2. Heterogeneous Applications for MOST - Design Tools for Complex Systems

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
http://st.inf.tu-dresden.de/teaching/most
WS 21-0.2, 20.11.21

1) Motivation for MOST
2) Design Tools for Complex Software Systems
3) Design of CPS with Domain-Specific CPS tool chain
    1) Cyber-physical systems (CPS)
4) Why Software Factories?

Obligatory Literature

- [Preevision] Vector. Modellbasierte Elektrik-/Elektronik-Entwicklung vom Architekturentwurf bis zur Serienreife. Preevision Handbuch
  - http://vector.com/portal/medien/cmc/marketing_items/web/91106.pdf
- [Reichmann] Clemens Reichmann, Daniel Gebauer, Klaus D. Müller-Glaser. Model Level Coupling of Heterogeneous Embedded Systems. Technical Report, FZI, 2008
  - http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.366
- [ETAS] Ulrich Lauff, Christoph Stoermer, Thomas Dollmaier, Mathias Klauda. ETAS GmbH, Stuttgart, Germany. Development Tools for Hybrids and Electric Cars.
  - http://www.etas.com/download-center-files/products_ASCET_Software_Products/1002_ATZ_elektronik_Entwicklungswerkzeuge_fuer_HEV_EV_EN.pdf

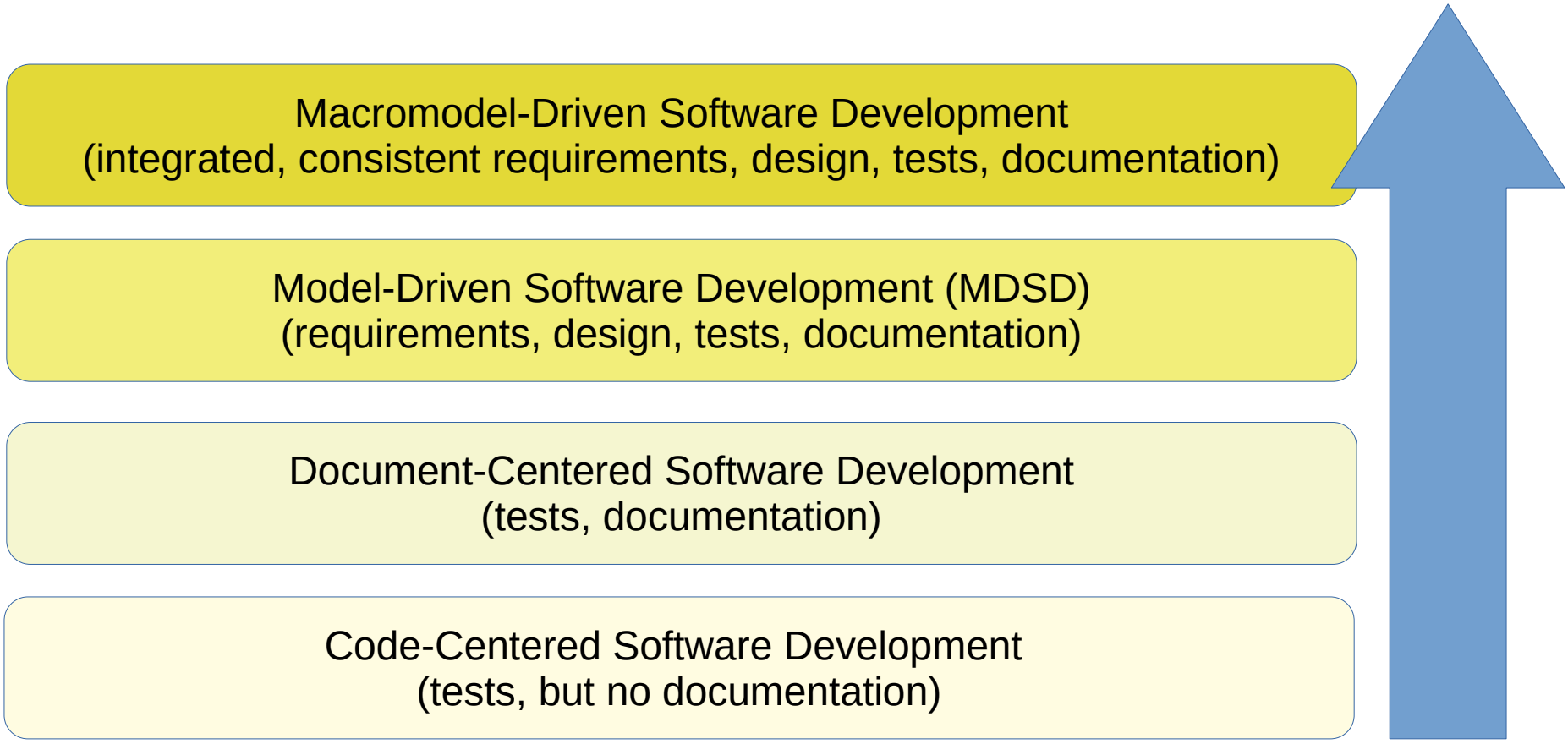
Literature

- [Zverlov] Sergey Zverlov. Comparison of two level-based Approaches for the Development of Embedded Systems. Bachelor Thesis in Computer Science. TU München, 2008.
- [Wurman] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. AI Magazine Volume 29 Number 1 (2008) (© AAAI)
- [MüGl09] Prof. Dr.-Ing. Klaus D. Müller-Glaser. Slide set. Model-Driven Engineering for Automotive Systems. UCSD SAASE 2009
  - http://jacobsschool.ucsd.edu/GordonCenter/g_leadership/l_summer/docs/saase/symposium-presentations/KlausMuellerGlaser.pdf

2.1 Example for Heterogeneous Software Factories: Integrated Development Environments for Large Software Systems (MDSD-Software-IDE)

Change in Software Development

- From code-only to documents to models to macromodels (integrated consistent multimodels)



What is needed for MDSD: Tool, Language, Process, Workflow and Method Engineering

Product-Line Engineering is the discipline of constructing domain-specific product families.

Software Ecosystem Engineering is the discipline of constructing open product platforms with appstores.

Tool Engineering is the discipline of constructing company-specific, domain-specific tools.

Language Engineering is the discipline of constructing company-specific, domain-specific languages for tools, processes, and workflows.

Process Engineering (Method Engineering) is the discipline of specifying and constructing methods and processes for a team of people to conduct a project.
Software Process Engineering (Software Method Engineering) focuses on software development processes.

Workflow Engineering is the discipline of running executable processes (workflows)
- For a team, in an application

Workflow engineering uses **behavioral languages.**

Design Tools:
Integrated Development Environment (IDE)
Software-Entwicklungsumgebungen (SEU)

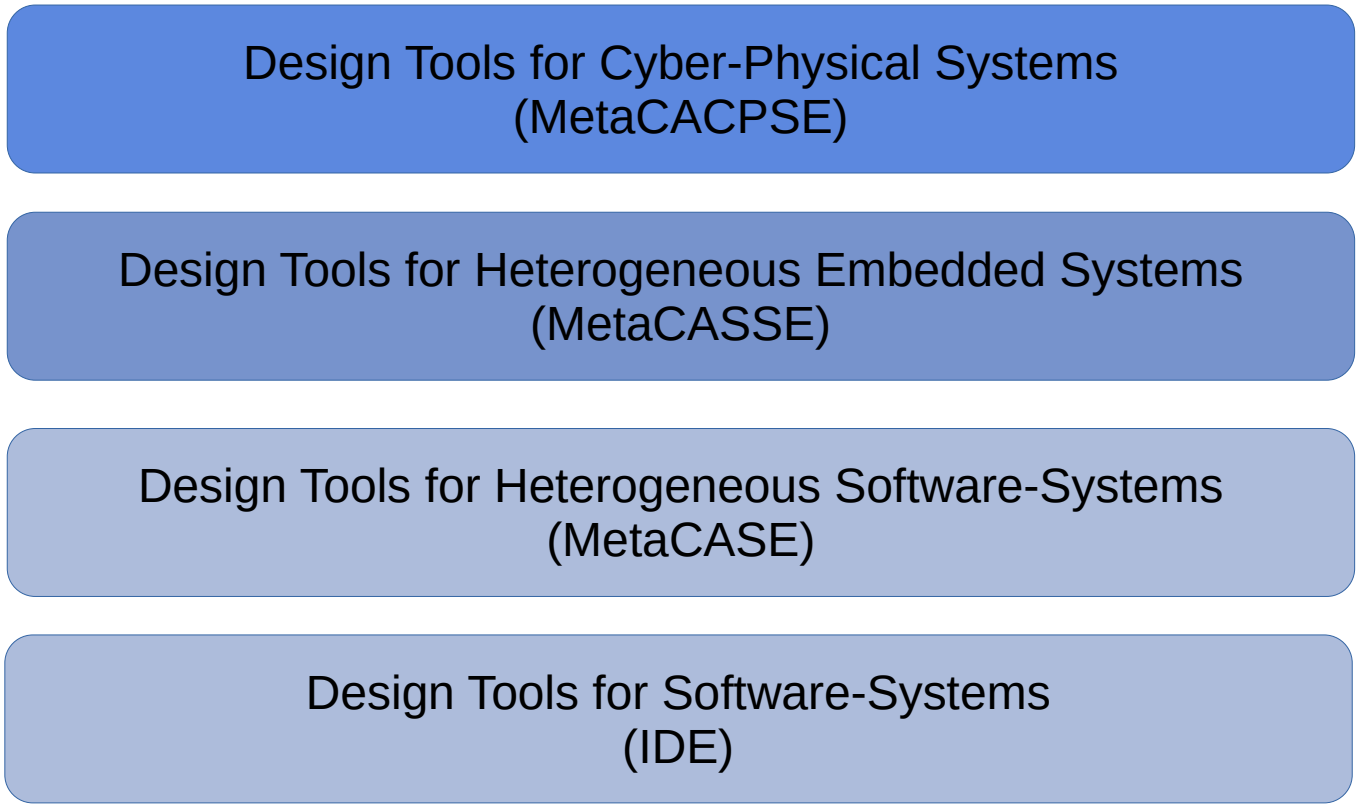
An **integrated development environment (IDE, Software-Entwicklungsumgebung, SEU)** consists of a **structured set of integrated standalone tools**
- to support a team in software development (process engineering)
- to construct a multimodel or macromodel.

- IDE support Computer aided Software Engineering (CASE)
- A MDSD-IDE (Meta-CASE) is complex software machine tool (Software-Werkzeugmaschine), an IDE for model-driven software development supporting
  - Many languages (DSL, metamodels) in a technical space
  - Heterogeneous software development
  - Model management system and Macromodel
- Other terms
  - Design Tools
  - Integrated Computer Aided Software Engineering (I-CASE)
  - Integrated Software Factory (ISF)
  - Software Engineering Environment System (SEES)
  - Integrated Project Support Environment (IPSE)
  - Integrated Software Engineering Environment (ISEE)

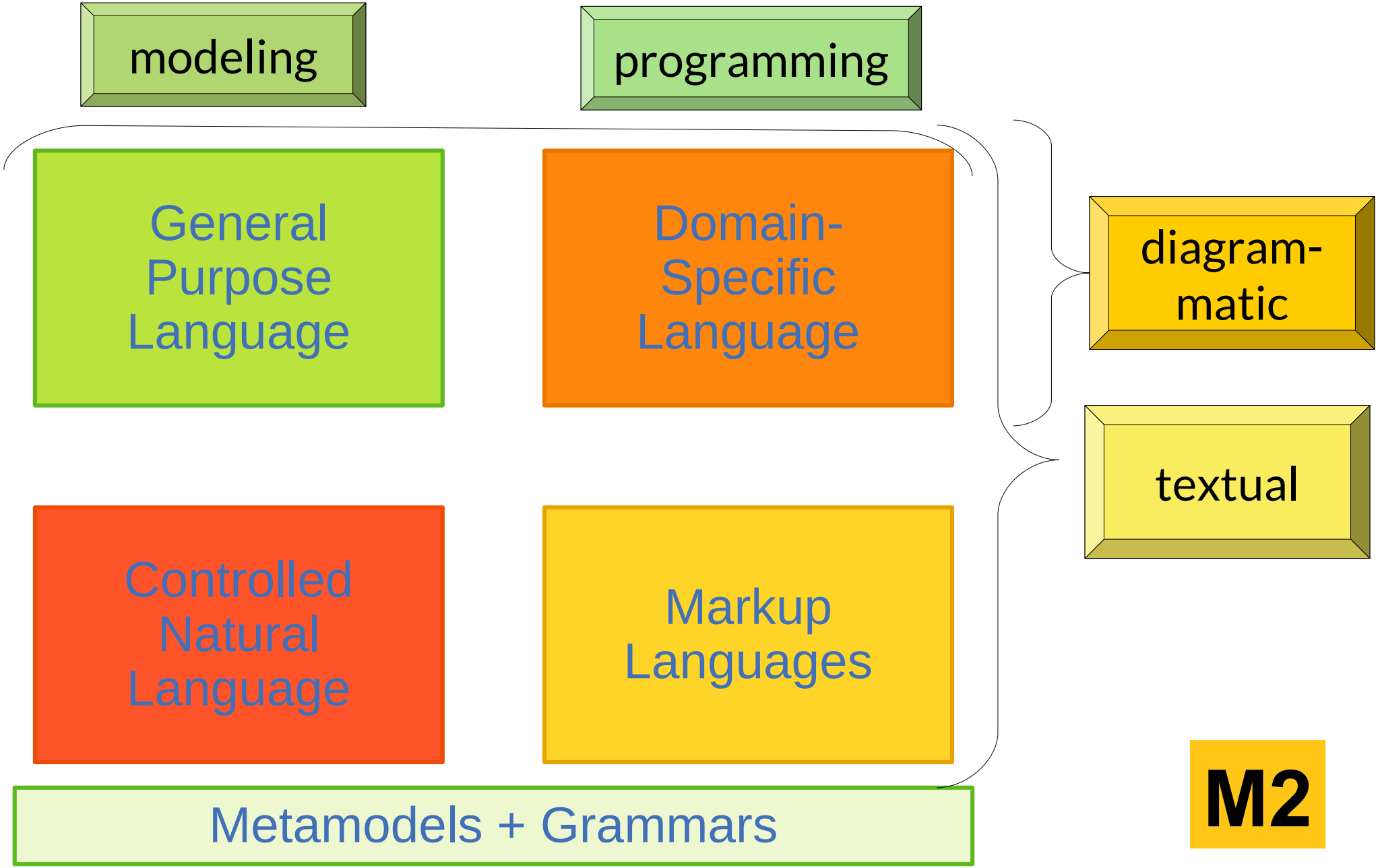
Nagl. M.: Software-Entwicklungsumgebungen: Einordnung und zukünftige Entwicklungslinien; Informatik-Spektrum 16(1993) H.5, S. 273-280

Design Tools can be Heterogeneous (Heterogeneous Software Factories)

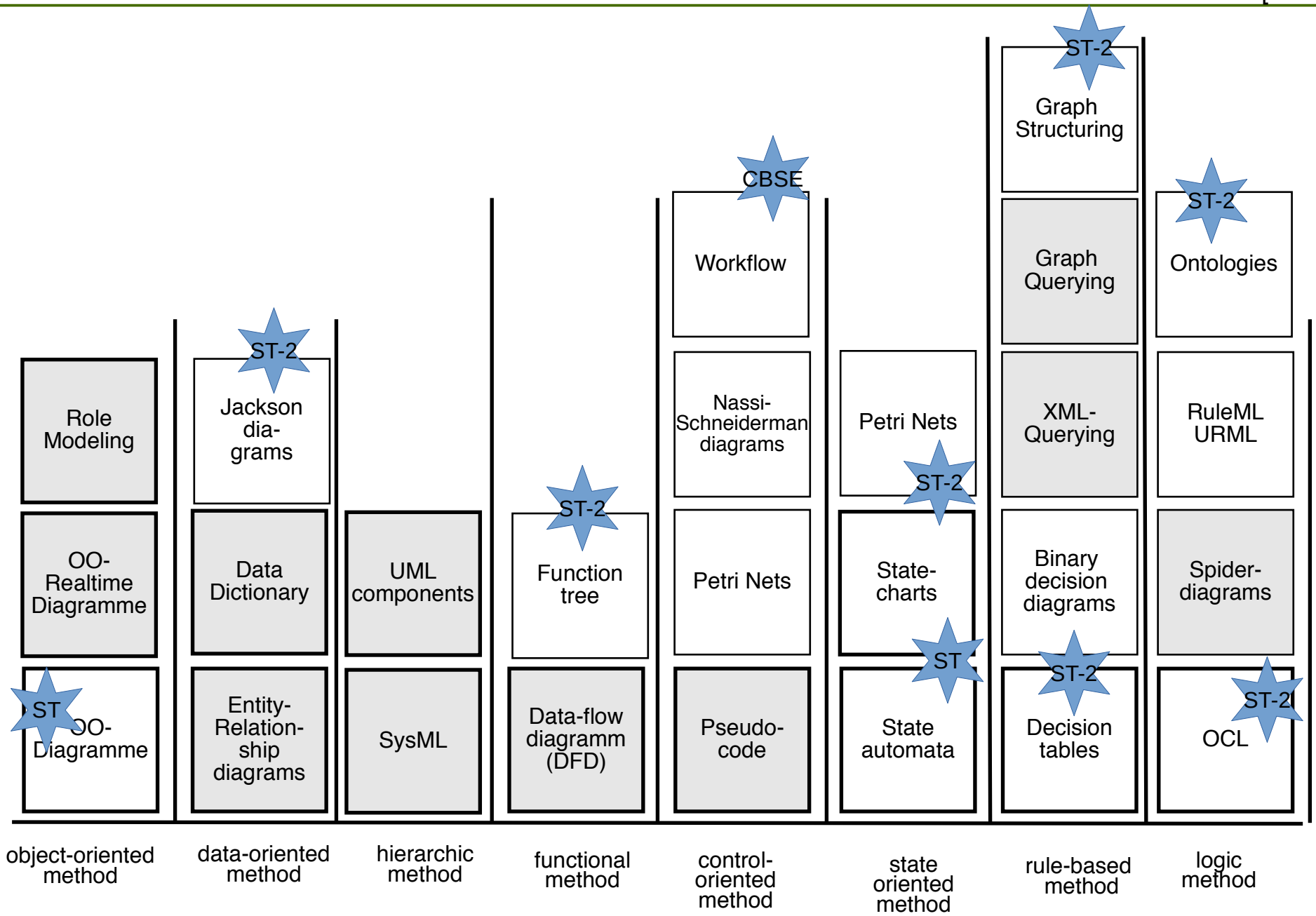
- While IDE are hosted in *one* technical space, (Heterogeneous) Software Factories span several ones.



Q16: Languages in Software Factories are Built on Metamodels and Grammars



Basic Languages for Design Tools



Problem for Companies: Building Domain- and Company-Specific Software Tools is Expensive

Tool	Person years	Cost in kEuro
Compiler	1-2	100
Optimizer	1-3	150
Back-End	0.5-1	100
Compiler component framework	20	1000
UML-IDE	5	250
Java-Refactorer	2-4	200
Energy Unit Test-Framework	1	50
Tool for Requirments management	2-4	200
Mobile Phone Test-Framework	2	100

How to Master Tool, Language, Process, Method Engineering in a Company?

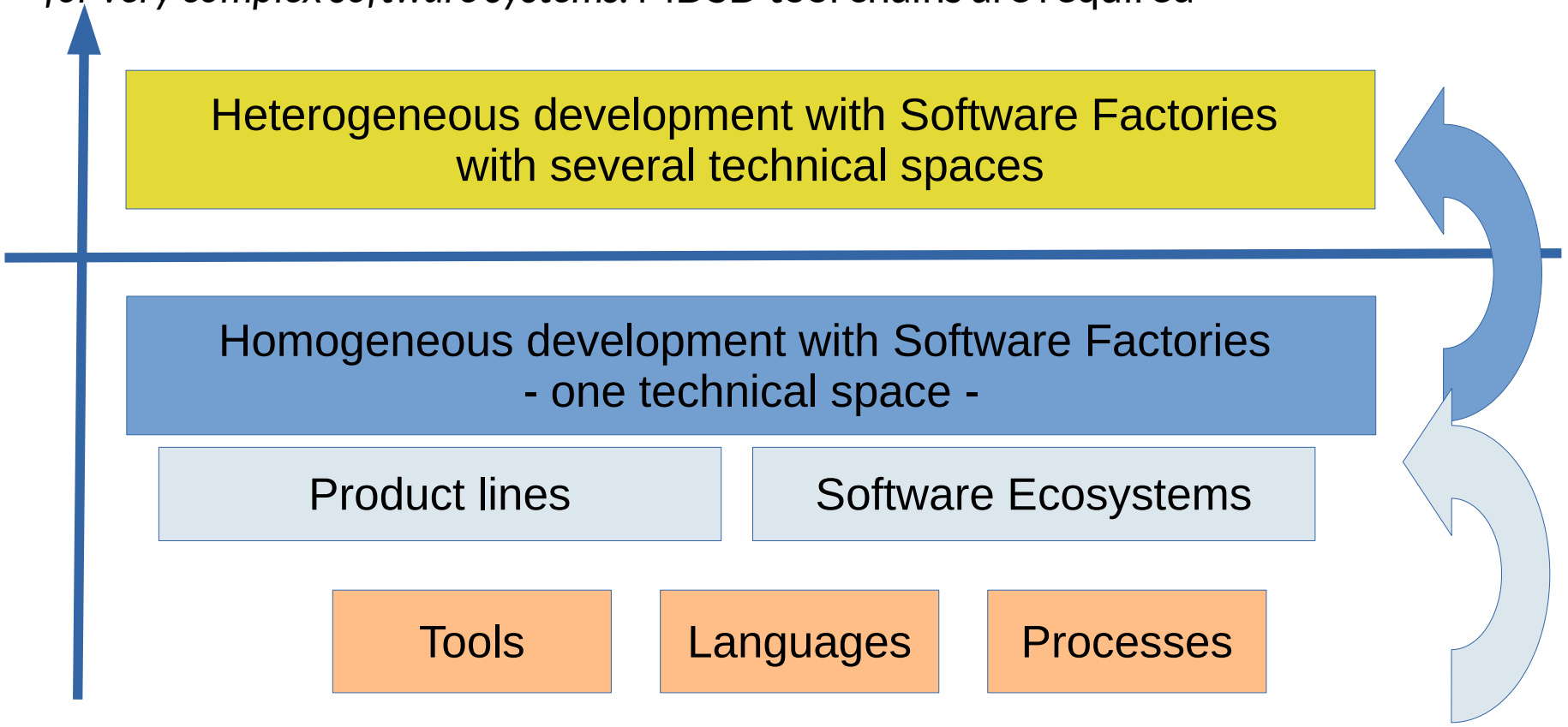
How can I create
simple tools, methods, languages,
and
reuse them for more complex tools, methods, languages
in my company?

Answer:
Mastering a **software factory** in a **technical space**
creating and composing

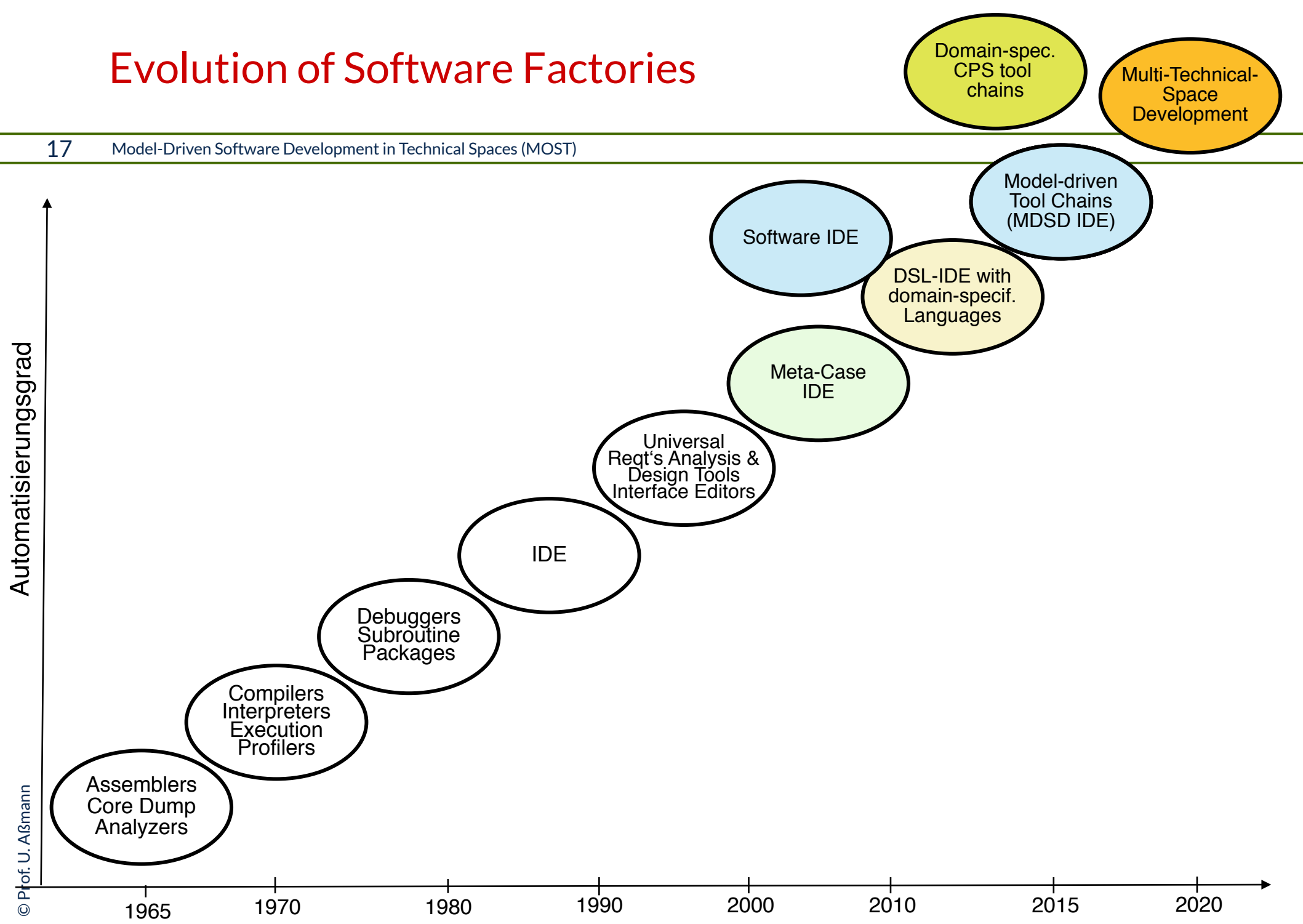
- Metamodels of base languages (on M2)
- Models (on M1)
- Repositories (on M0)

Maturity Levels of Software Companies

- Many companies do not know technical spaces nor software factories
- Many companies work with *homogeneous software development* in *one technical space*
- Some companies master *heterogeneous software development in one technical spaces for complex software systems*. Tools are required
- Some companies master *heterogeneous software development in several technical spaces for very complex software systems*. MDSD tool chains are required



Evolution of Software Factories



After [Alan S. Fisher. 1991. Case: Using Software Development Tools (2nd ed.). John Wiley & Sons, Inc., New York, NY, USA., S.20]
https://archive.org/details/caseusingsoftwar00fish

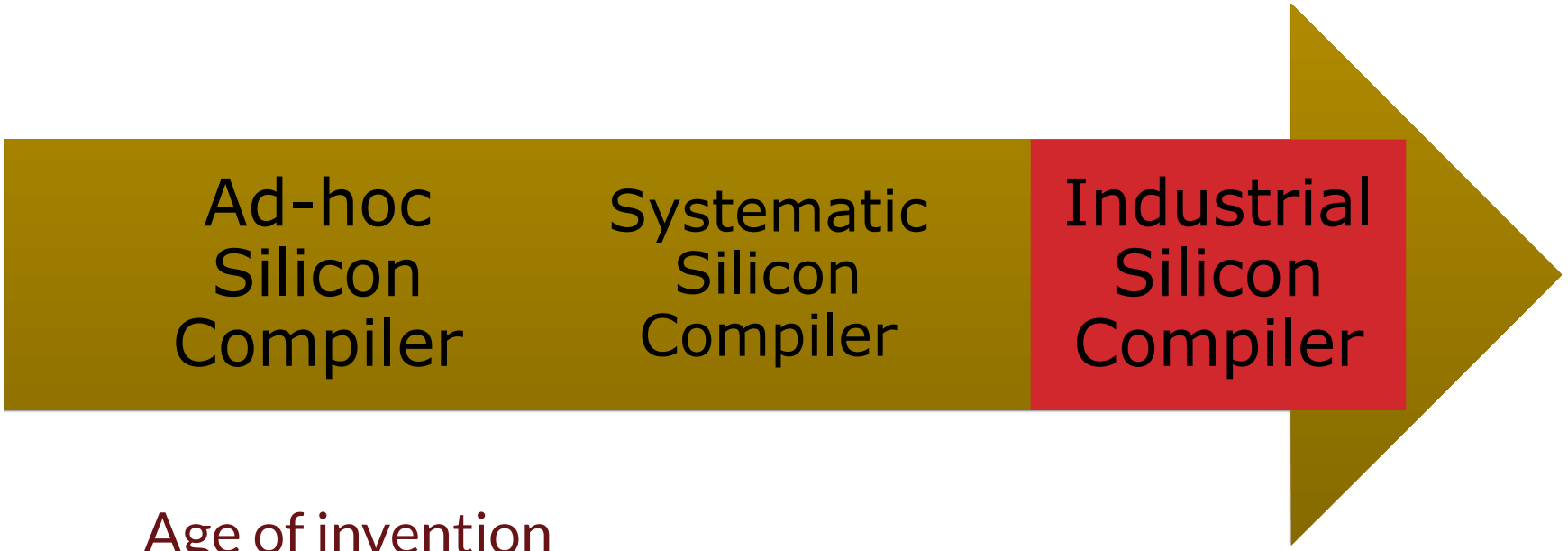
2.2. Example 2 of Software Factory: “Silicon Compilers”

Example 1: MDSD ToolChain: Silicon Compilers

- [Wikipedia:Silicon_Compiler] A **silicon compiler** is a software system that takes a user's specifications and automatically generates an integrated circuit (IC). The process is sometimes referred to as hardware compilation.
- [Wikipedia:Design_flow_(EDA)]
- Alberto Sangiovanni-Vincentelli distinguished three periods of EDA [Tides]:
- **"The Age of Invention:** During the invention era, routing, placement, static timing analysis and logic synthesis were invented.
- **The Age of Implementation:** In the age of implementation, these steps were drastically improved by designing sophisticated data structures and advanced algorithms. This allowed the tools in each of these design steps to keep pace with the rapidly increasing design sizes. However, due to the lack of good predictive cost functions, it became impossible to execute a design flow by a set of discrete steps, no matter how efficiently each of the steps was implemented.
- **The Age of Integration:** This led to the age of integration where most of the design steps are performed in an integrated environment, driven by a set of incremental cost analyzers."

Example 1: How the Silicon Compiler Industry Matured over Time

- Sangiovanni-Vincentelli claims that other industries (e.g., for CPS) will go the same way[20]



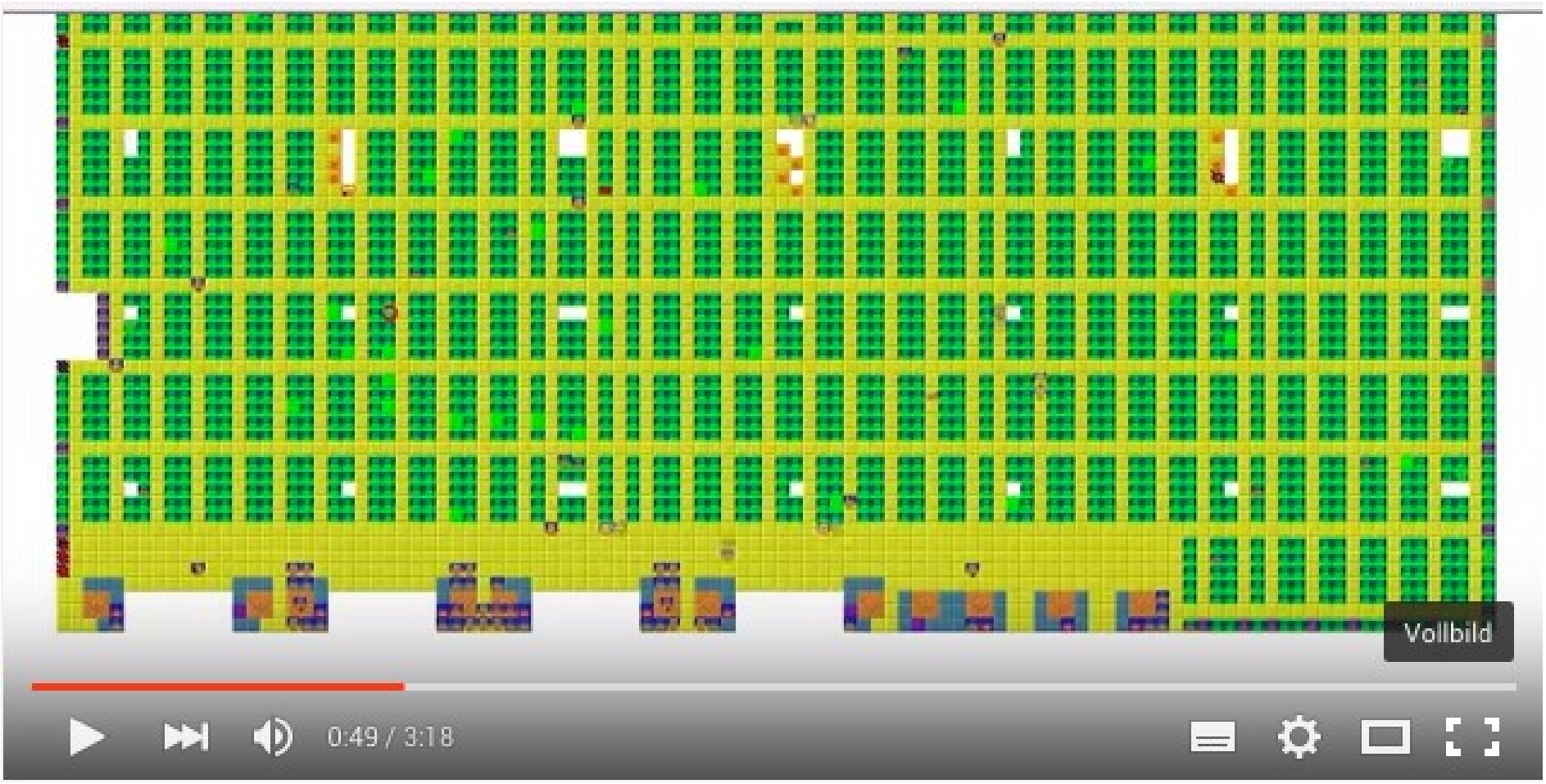
[Sangiovanni-Vincentelli Tides]

2.3. Example 3: Software Factories for Cyber-Physical Systems

2.3.1. What is a Cyber-Physical System (CPS)?

Kiva Bots for Logistics

- [Wurmer] Just search on YouTube for Kiva Systems
- https://www.youtube.com/watch?v=8gy5tYVR-28
- https://www.youtube.com/watch?v=6KRjuuEVEZs

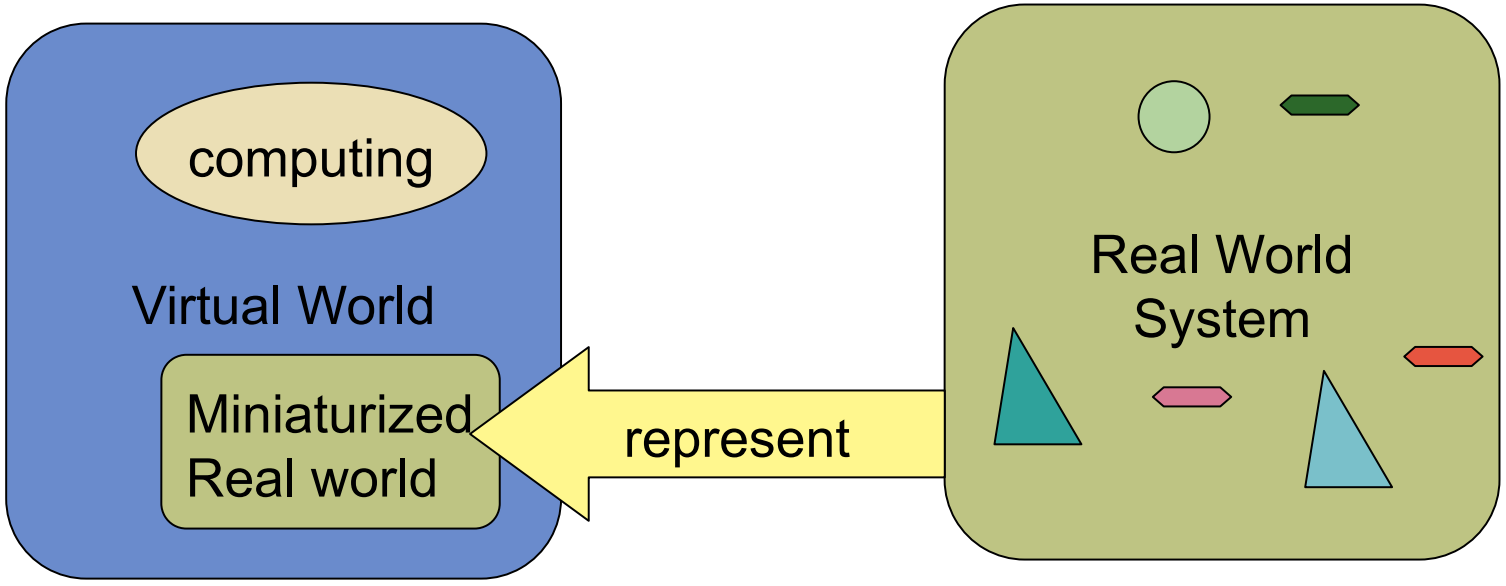


Smart Parking

http://commons.wikimedia.org/wiki/File:Bundesarchiv_Bild_183-H0605-0007-001,_Rostock,_Ernst-Th%C3%A4lmann-Platz,_Parkplatz,_Marienkirche.jpg#mediaviewer/File:Bundesarchiv_Bild_183-H0605-0007-001,_Rostock,_Ernst-Th%C3%A4lmann-Platz,_Parkplatz,_Marienkirche.jpg

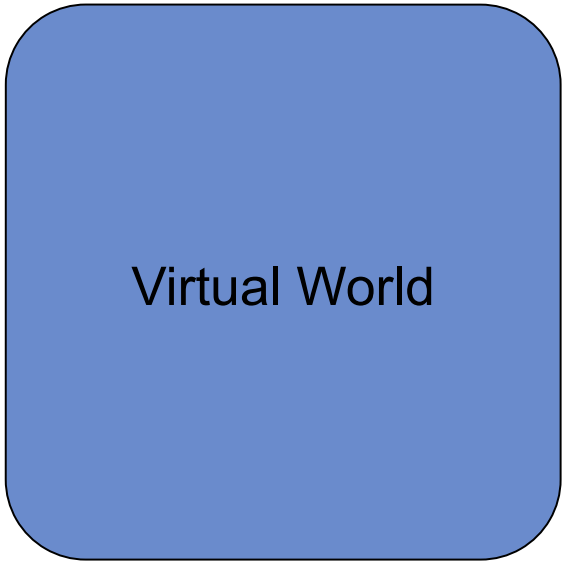
„Standard“ Computing

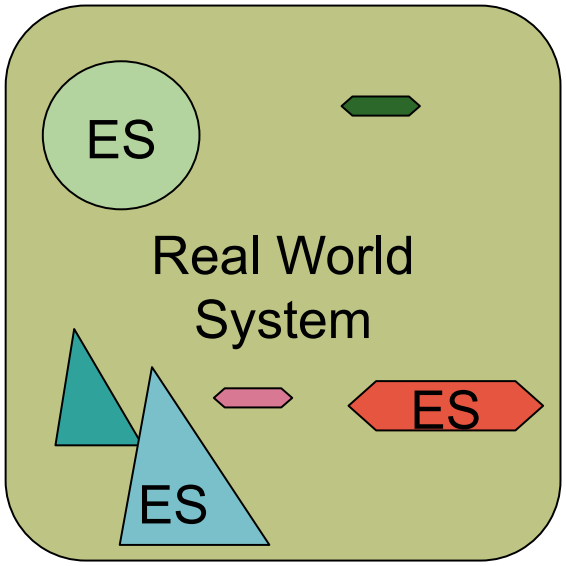
- „Standard“ Computing maps the real world into the computer and computes about it by simulation



Embedded System

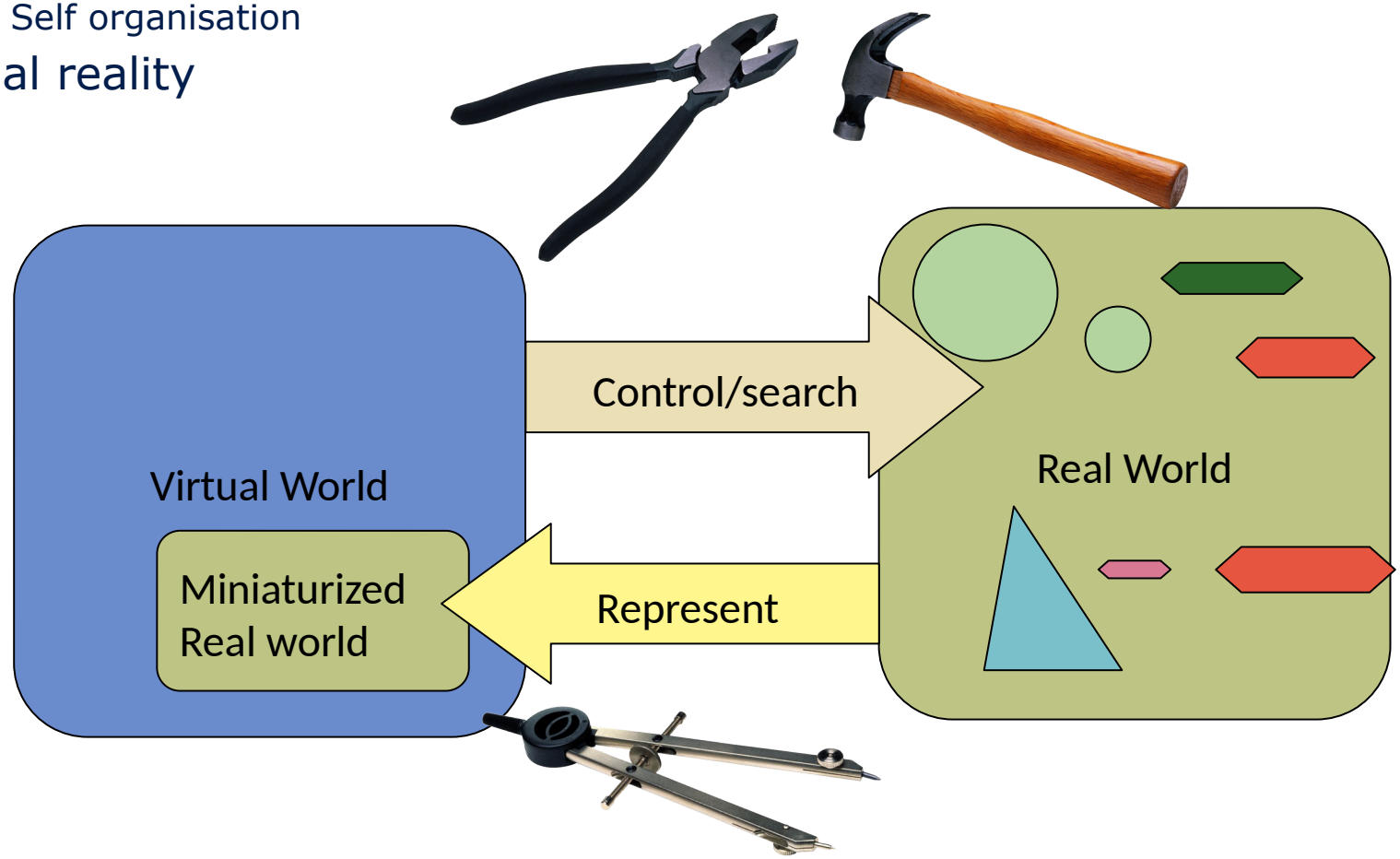
- The computer is integrated into the real-life object





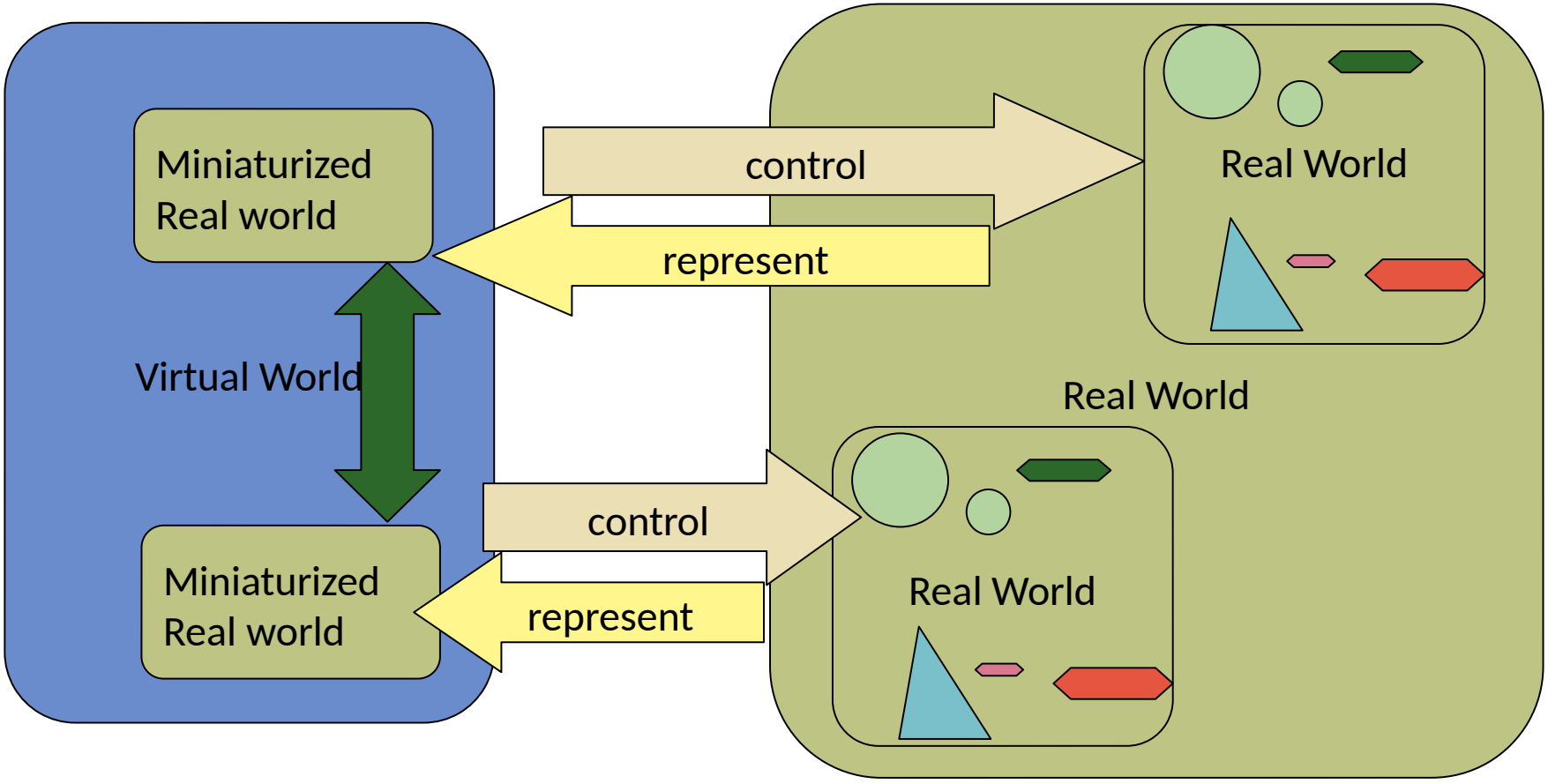
Cyber-Physical System (CPS)

- Simulation of intelligent things in space and time
  - Search possible
- Control of the intelligent things in space and time
  - Self regulation
  - Self optimization
  - Self organisation
- Dual reality



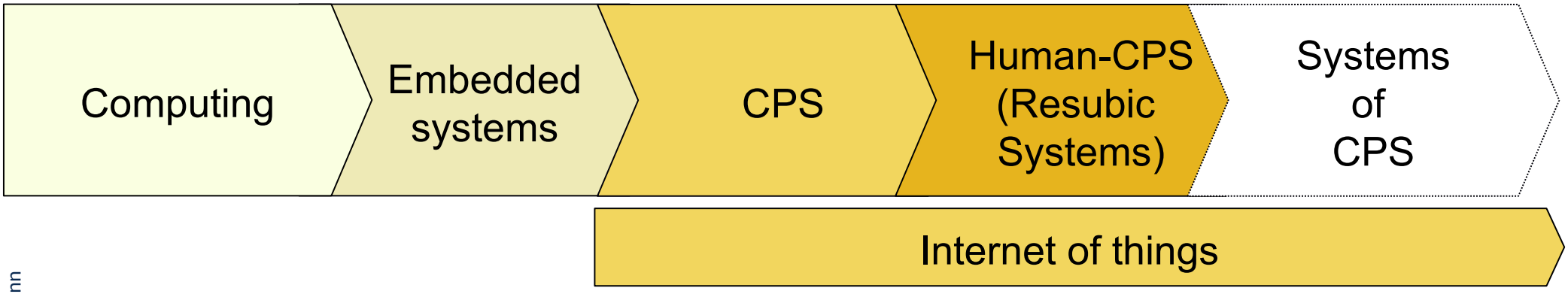
The Internet of Things

- Systems of CPS, i.e., remote tools

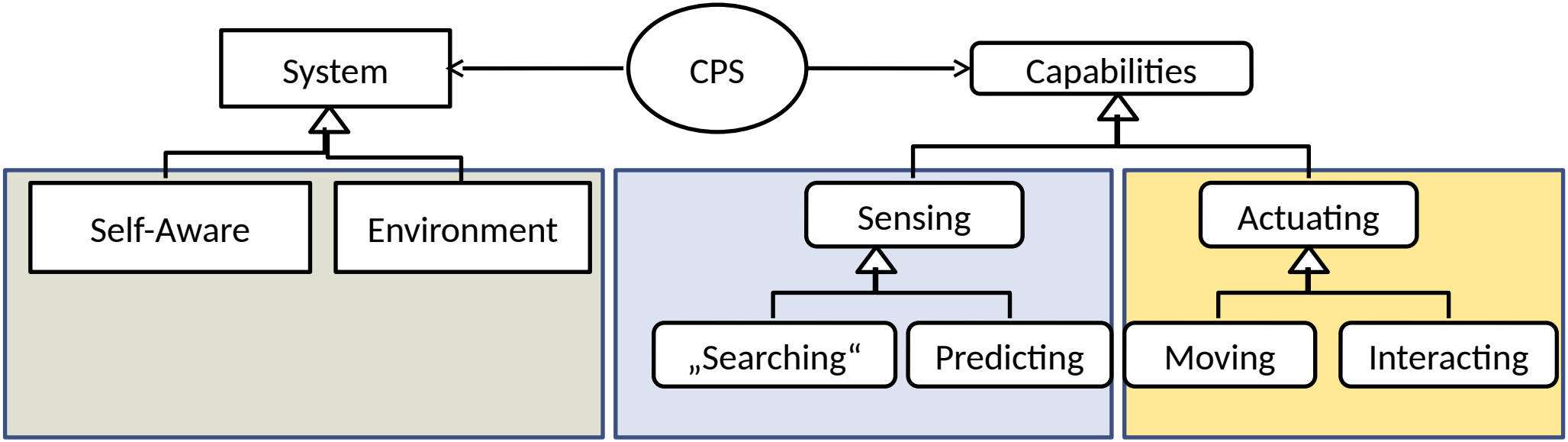


Trend CPS

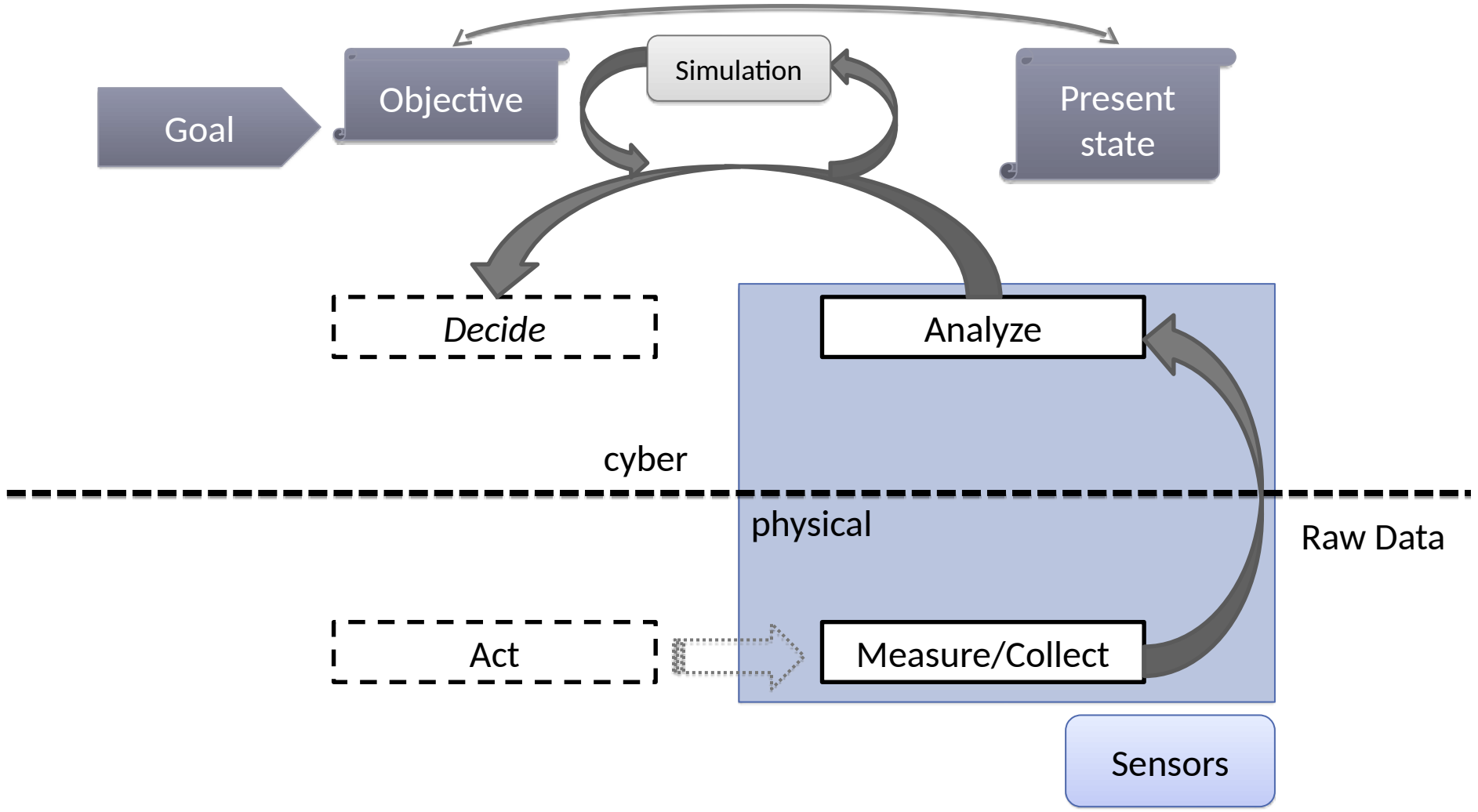
- Cyber-physical systems are the first step in the internet of things



Two Classes of Cyber-Physical Systems for Cyber-Physical **Search** and **Management**



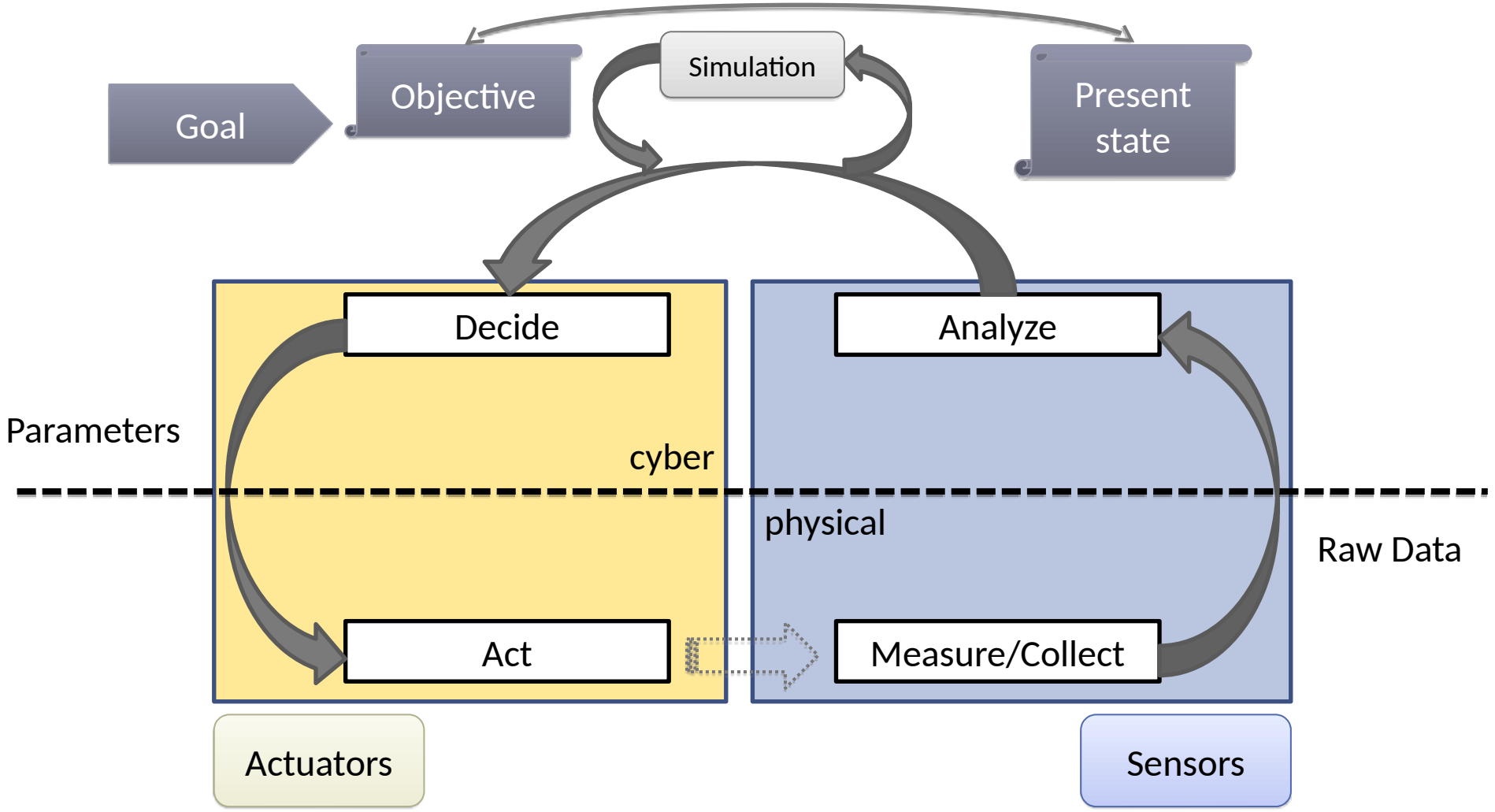
Cyber-Physical Database Systems = Analysis, Simulation and Prediction



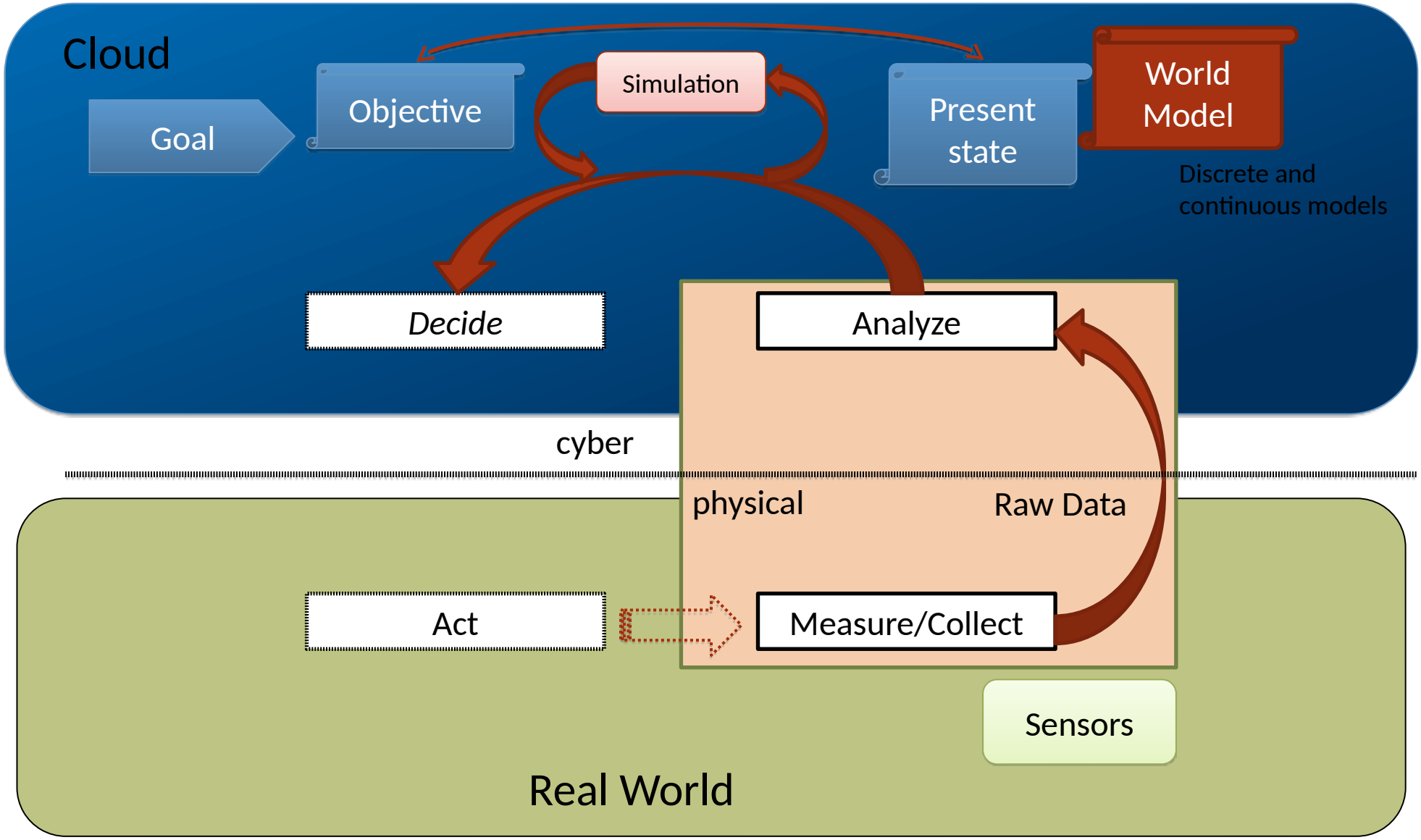
A Cyber-Physical System

http://commons.wikimedia.org/wiki/File:Traffic_seen_from_top_of_Arc_de_Triomphe.JPG

Cloud Robots =
Cyber-Physical Management Systems

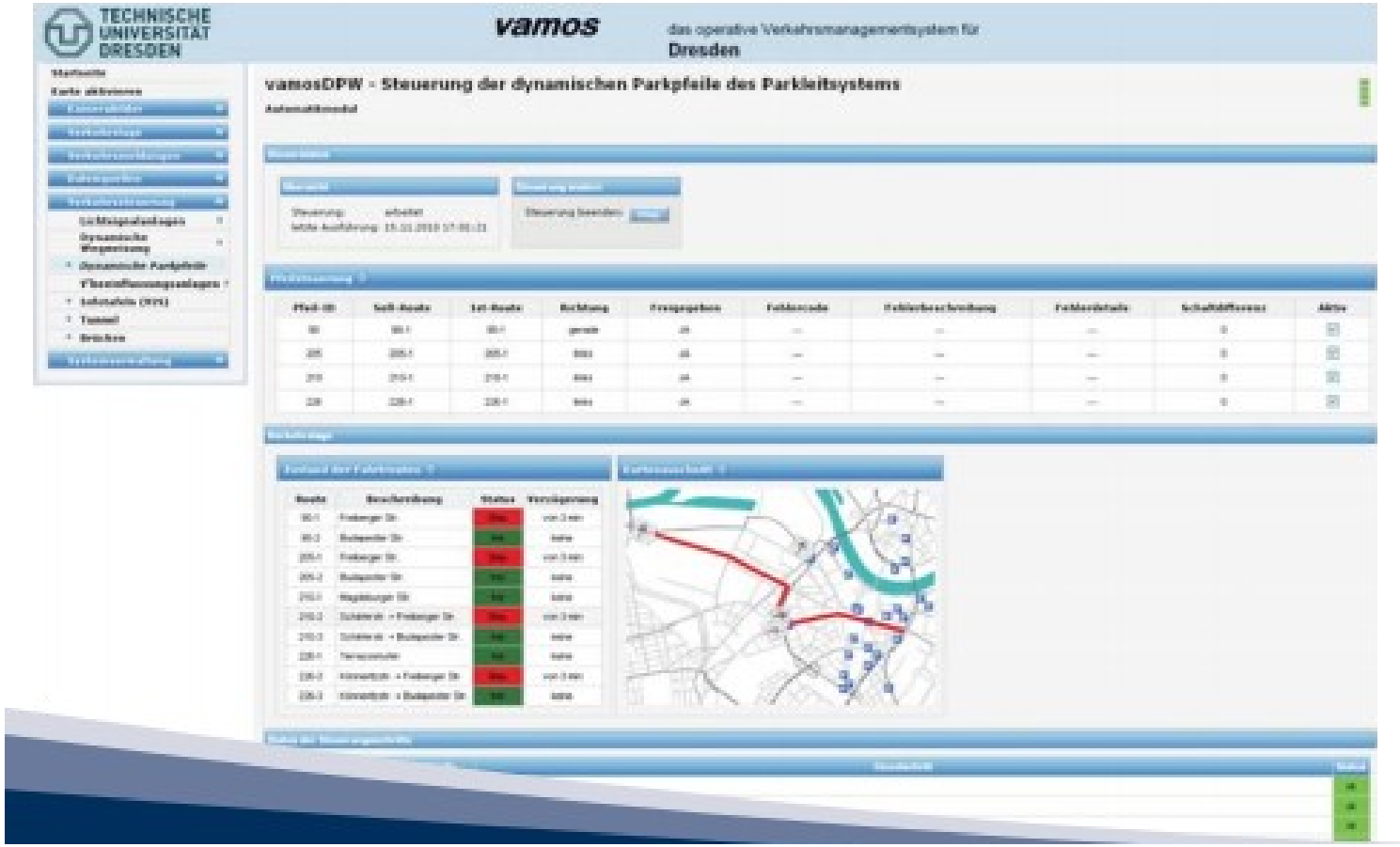


World Database Systems are Monitoring CPS (Analysis, Simulation and Prediction)

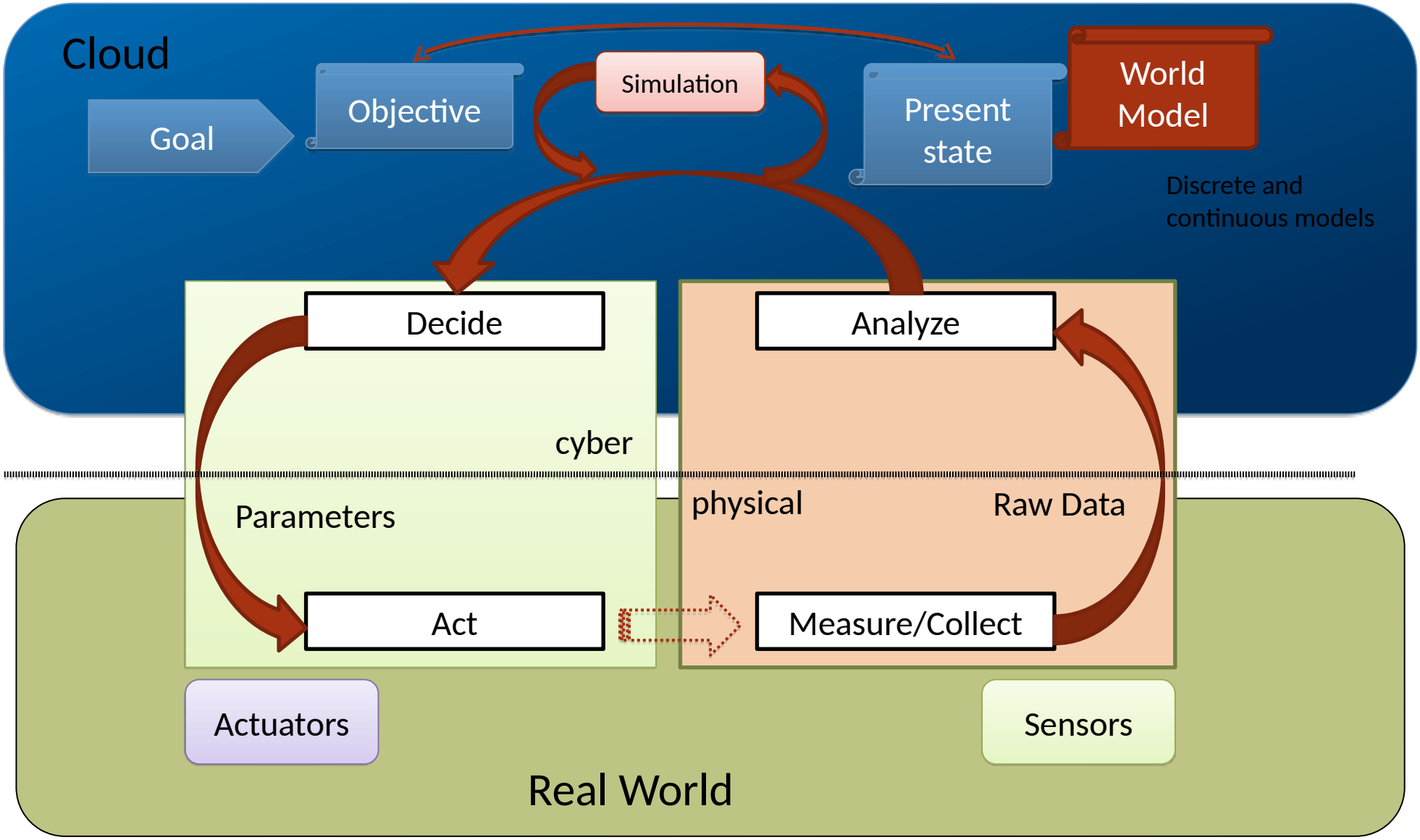


Ex.: The VAMOS Traffic Management System (Verkehrsleitsystem) Dresden

- Realtime data from the city‘s traffic
- http://www.vamosportal.de/
- http://wwwpub.zih.tu-dresden.de/~vamos/flyer/vamos_web.pdf



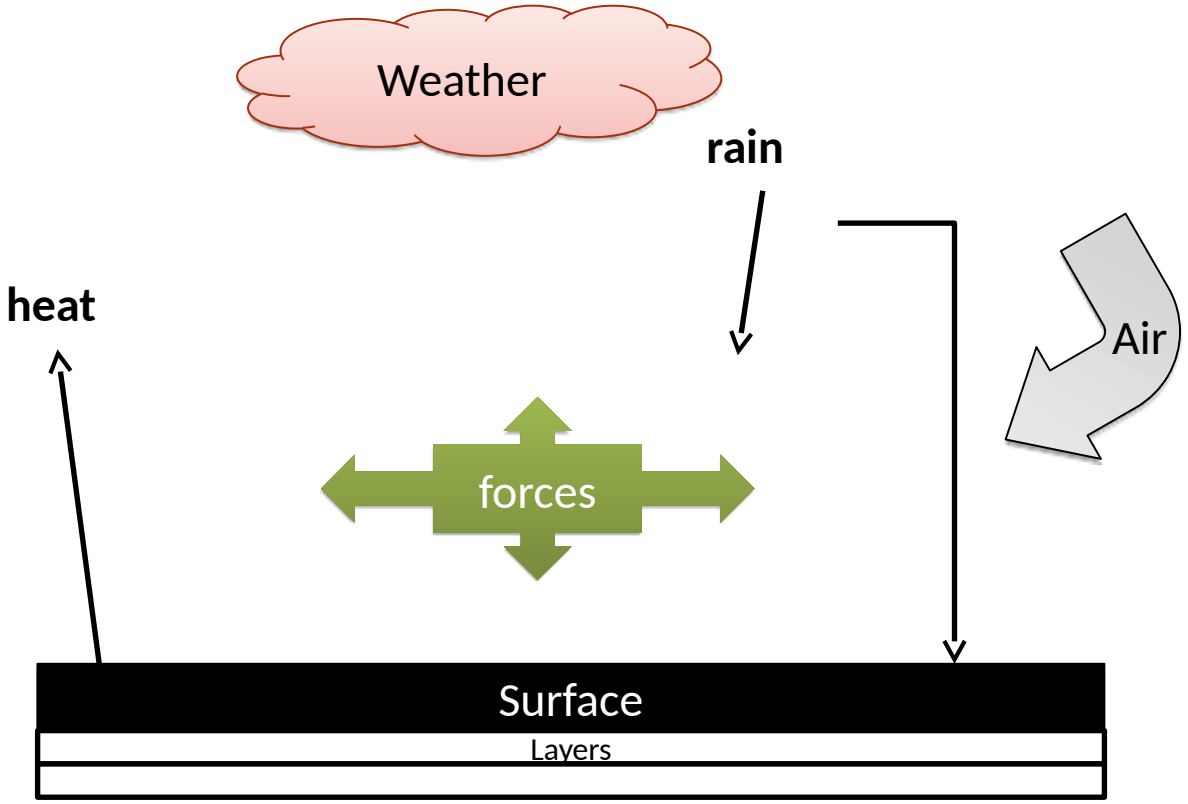
Cloud Robots are Controlling CPS



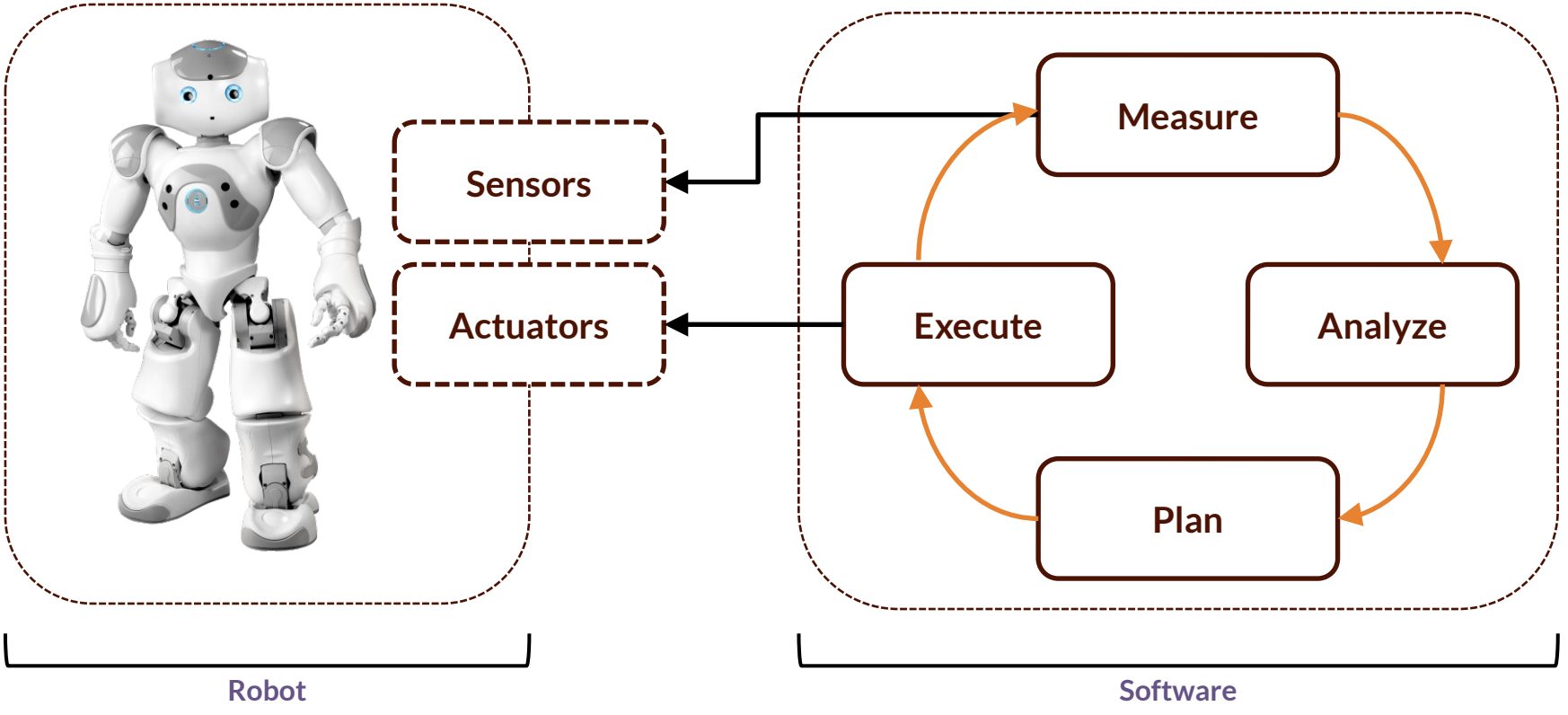
Physical Dynamics (Movement) of Cloud Robot

- How can I **control** a cloud robot move in space?

Cloud Robots
Need
World Models

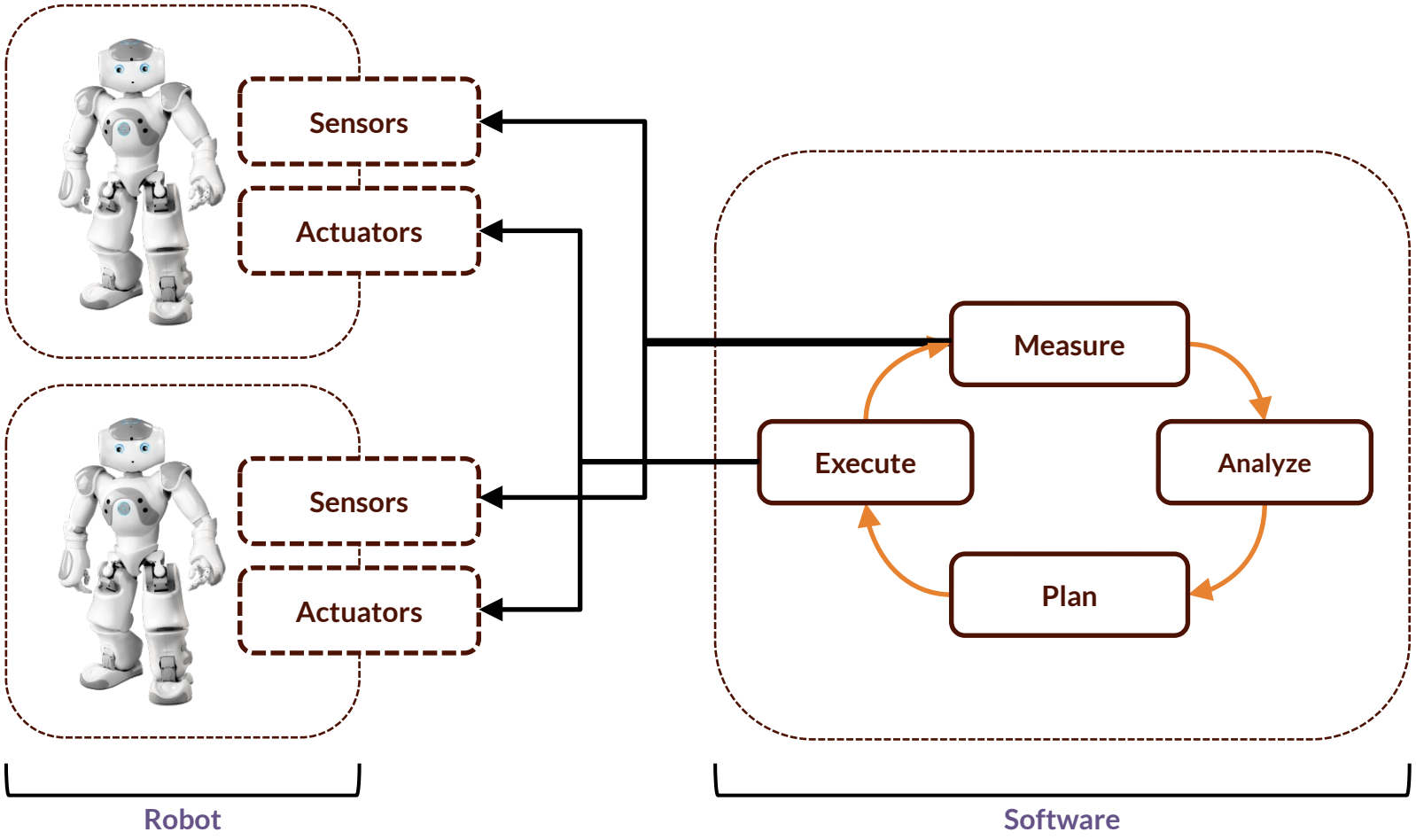


Cloud Robots are Adaptive Systems (MAPE Loop), and run a Dynamic Software Product Lines



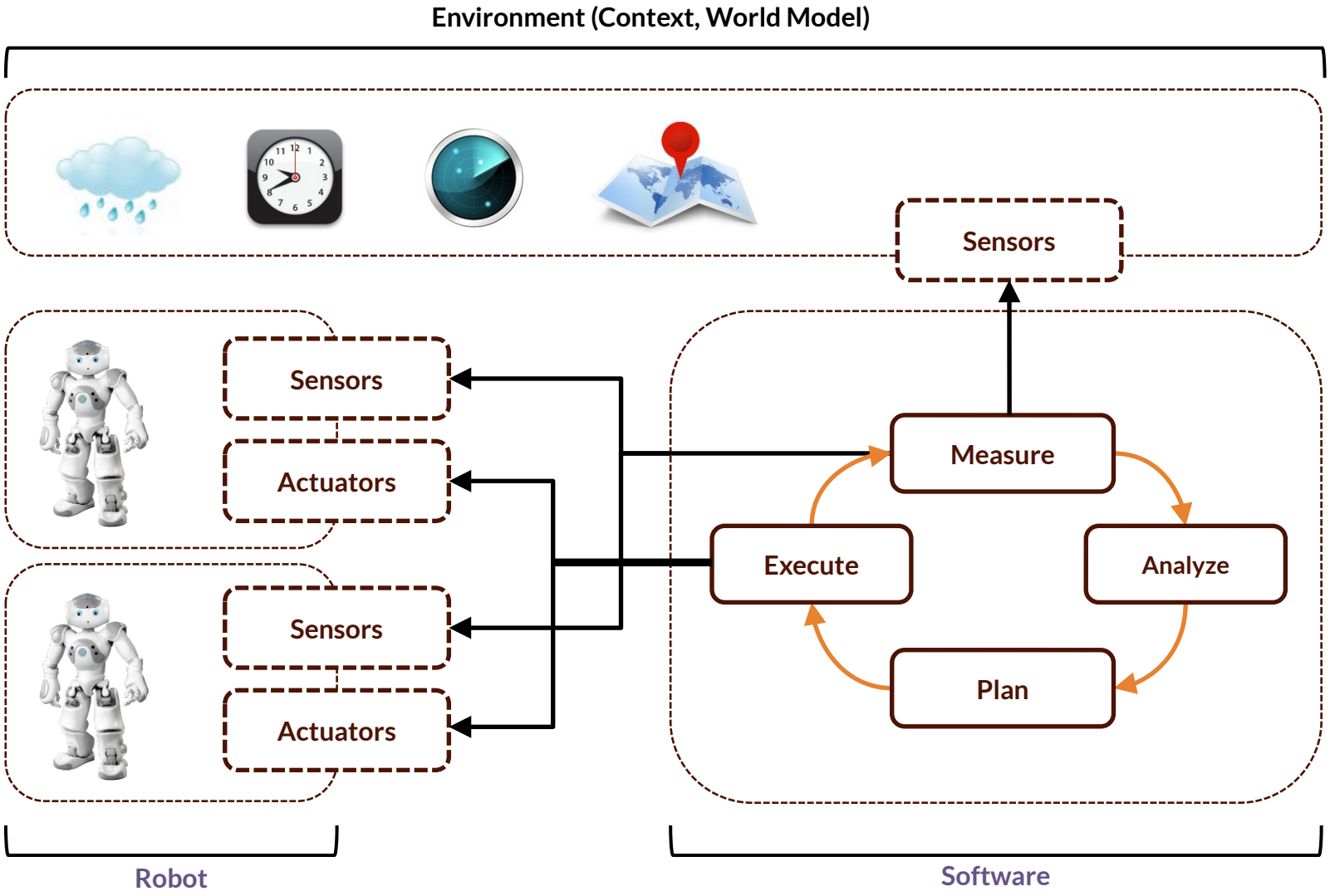
Elastic Architecture

Cloud Robots are Multi-Adaptive Systems



Elastic Architecture

Cloud Robots are Context-Adaptive Systems



Elastic Architecture

Industrie-4.0 (Smart Factory) with CPS

- Embedded System: machines, robots, presses, transport systems
- CPS: Autonomous control of the factory
  - Self assembly of the products
  - Autonomous control of logistics
  - Pull of products instead of push

http://commons.wikimedia.org/wiki/File:Mail_sorting_assembly_line.jpg

http://commons.wikimedia.org/wiki/File:Factory_Automation_Robotics_Palettizing_Bread.jpg?uselang=de

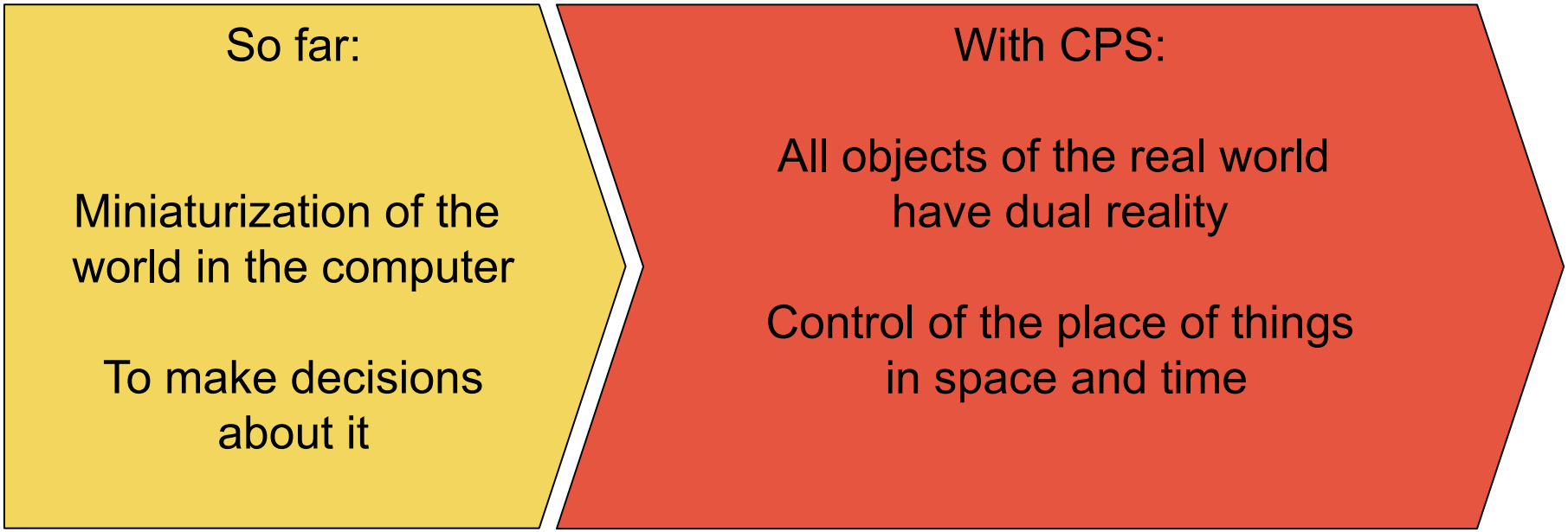
Smart Traffic/Transport/Logistics mit CPS

- Embedded System: Railcabs are autonomous train cars (Paderborn)
- CPS: Optimization of the German logistics

http://www.railcab.de

The Revolution of CPS

- All domains in transport, logistics, assembly, housing, cities will change
- Nothing will stay as it is
- All engineering disciplines will change until 2020



Questions

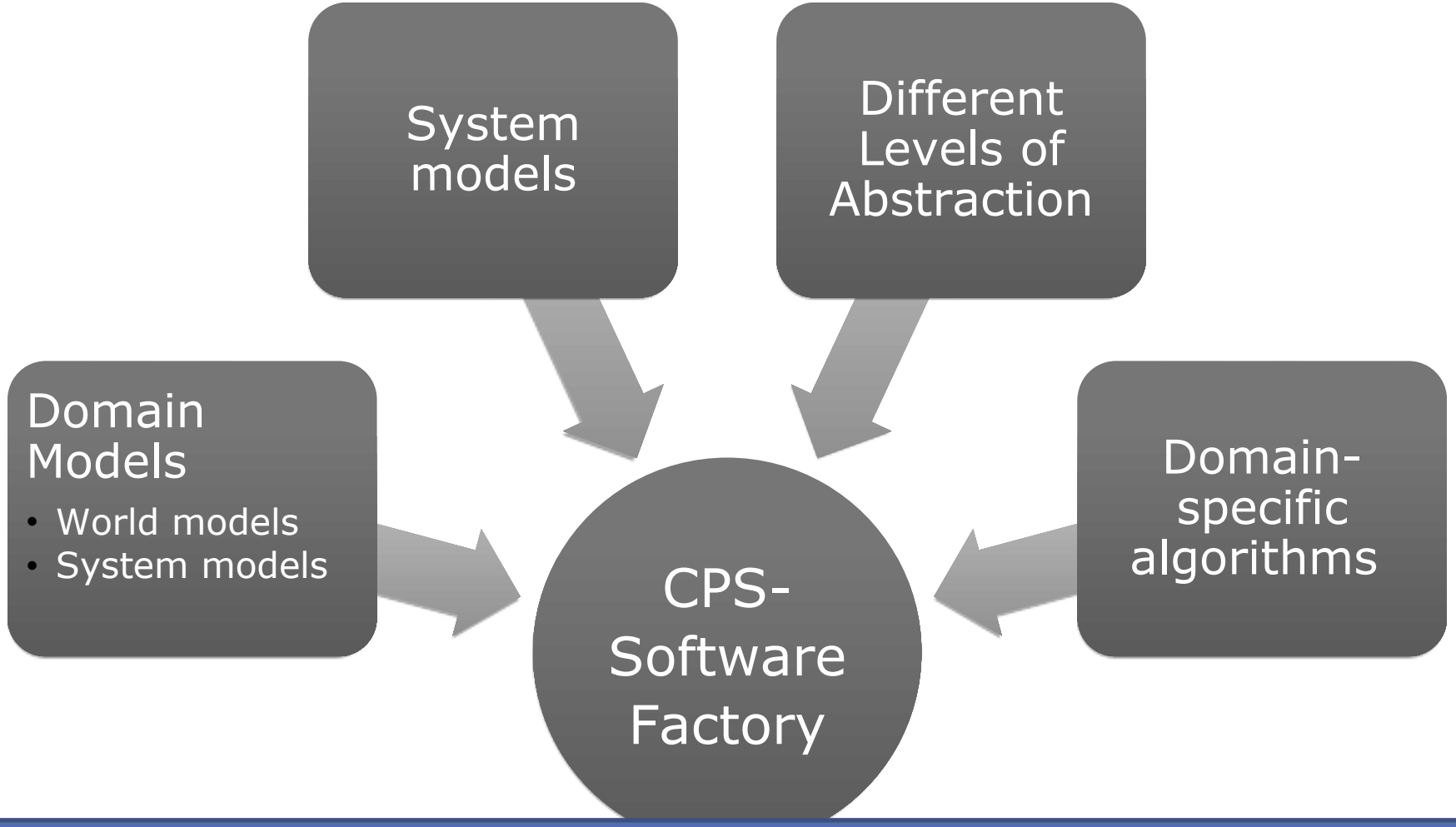
How can we build such complex tool suites for CPS (CPS-IDE)?

Answer: By Model-Driven Software Development (MDSD) for software **and** system, with

- Metamodels of languages (on M2)
- Models (on M1)
- Repositories (on M0)

2.3.2 Domain-Specific Software Factories (Design Tools) for Design of Cyber-Physical Systems

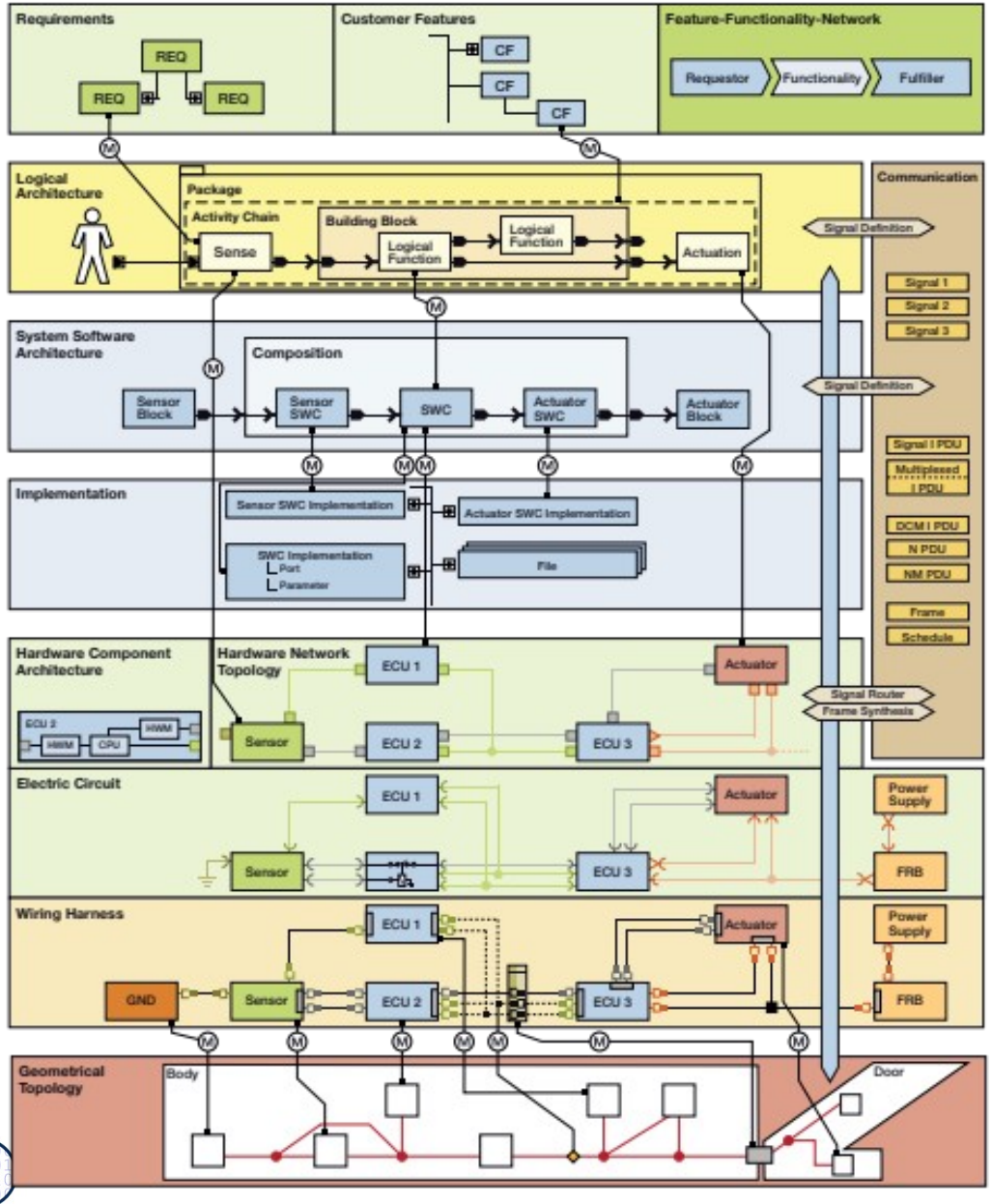
Domain-Specific CPS-Software Factories

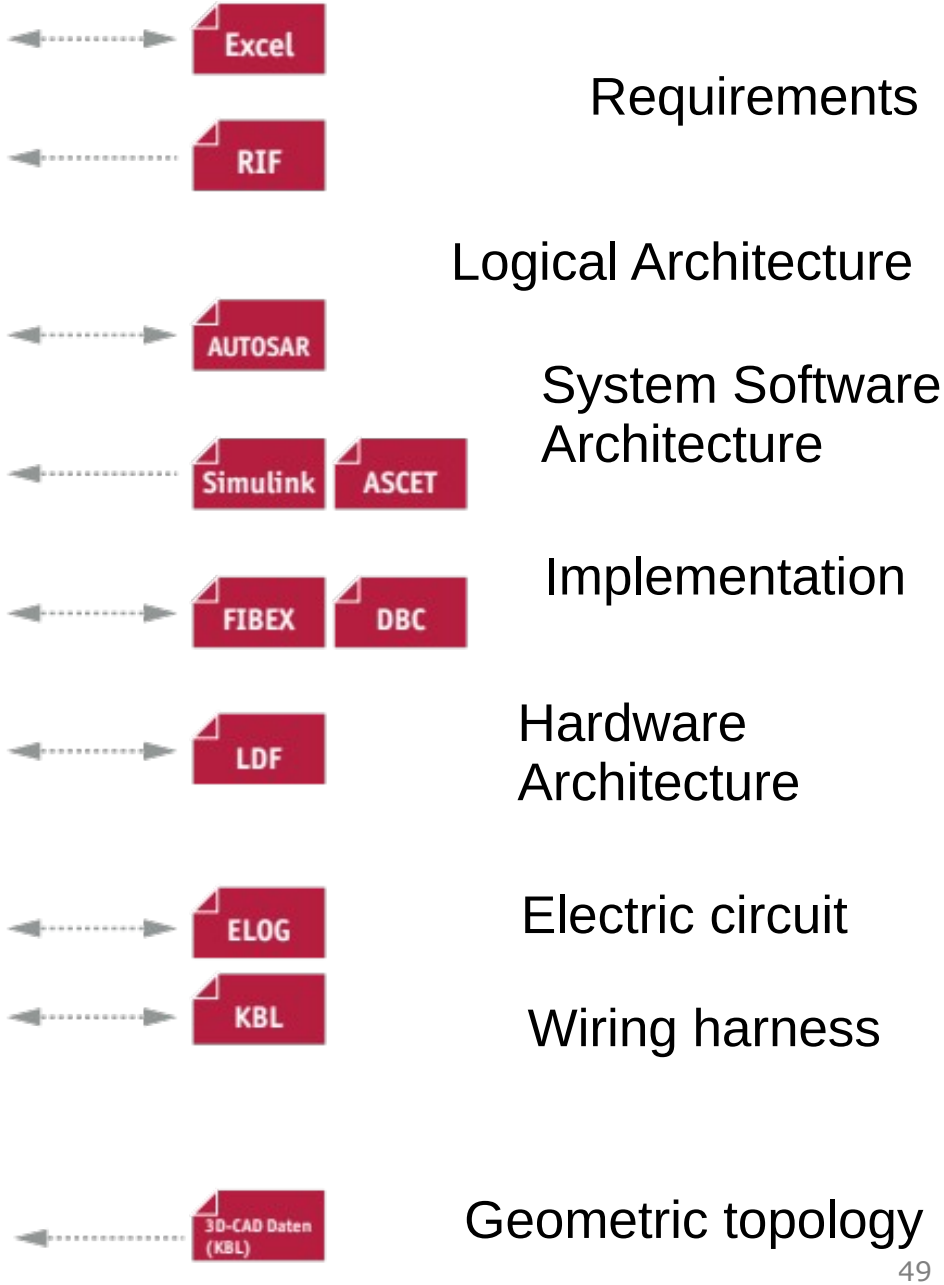


CPS-Software Factories are domain-specific

Example: Car Design with PREEVision (Vector)

[Preevision]



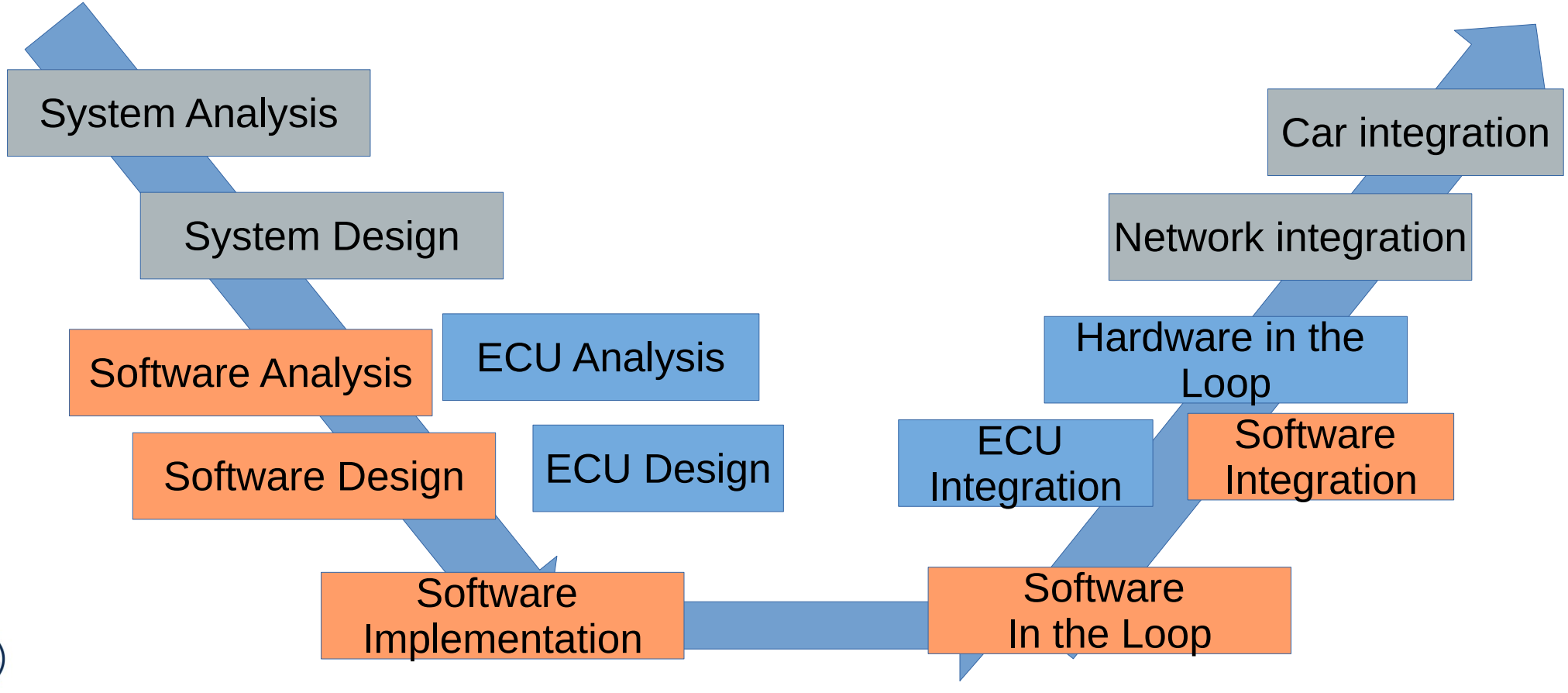


PreeVision has 3 Tools Steered by Metamodels

[MüGl09]

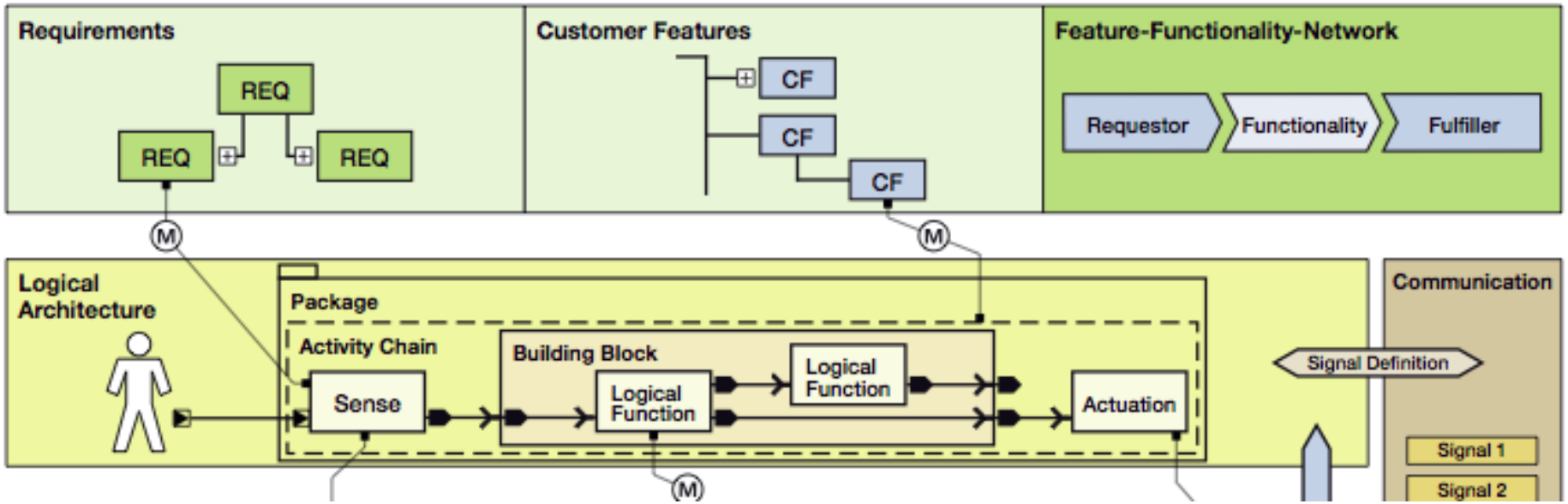
- PREEvision Architect
- PREEvision Function Designer
- PREEvision Electric Designer

- With options:
  - vTESTcenter
  - PREEvision Collaboration Platform
- All involved models are metamodeled



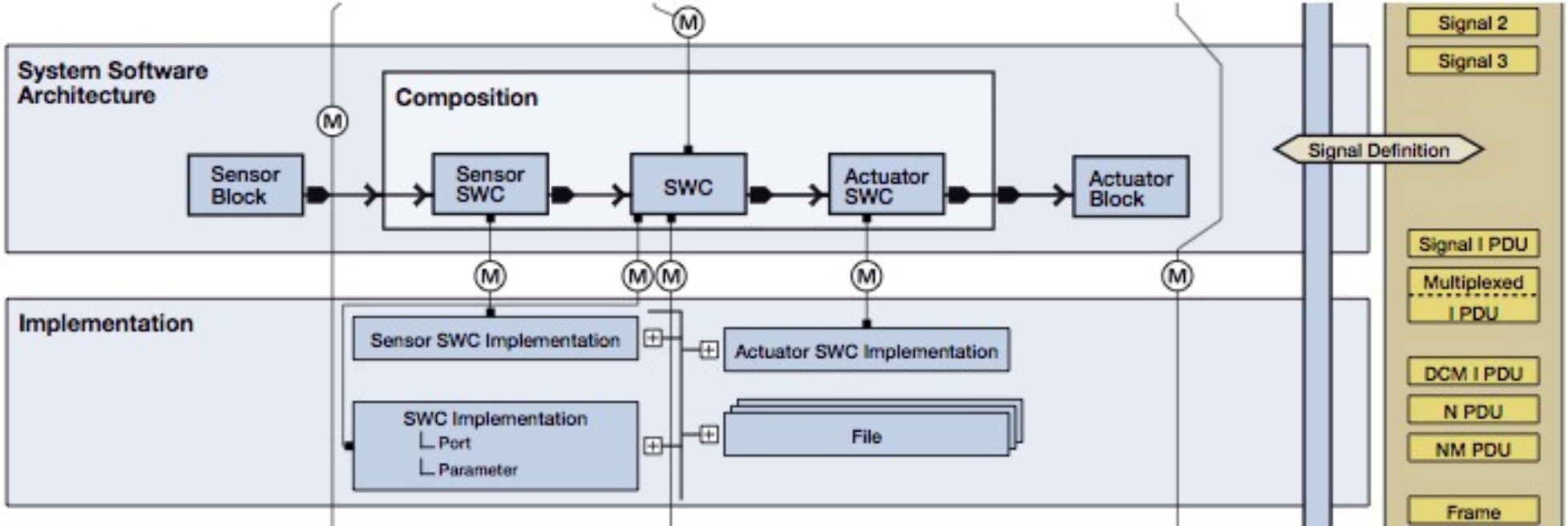
PreeVision Models in More Details

- Requirements specification with Excel and Requirements Interchange Format (RIF)
- Logical architecture with AUTOSAR components



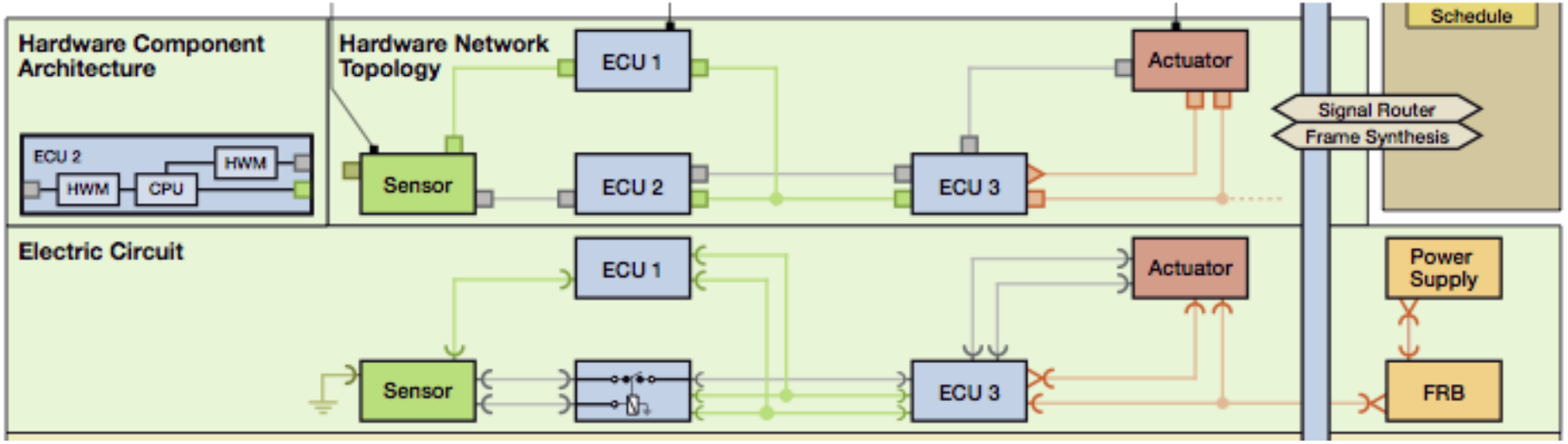
PreeVision Models in More Details

- Software Architecture with Simulink components (blocks) and ASCET model components (from ETAS)
- Implementation (generated or hand written)



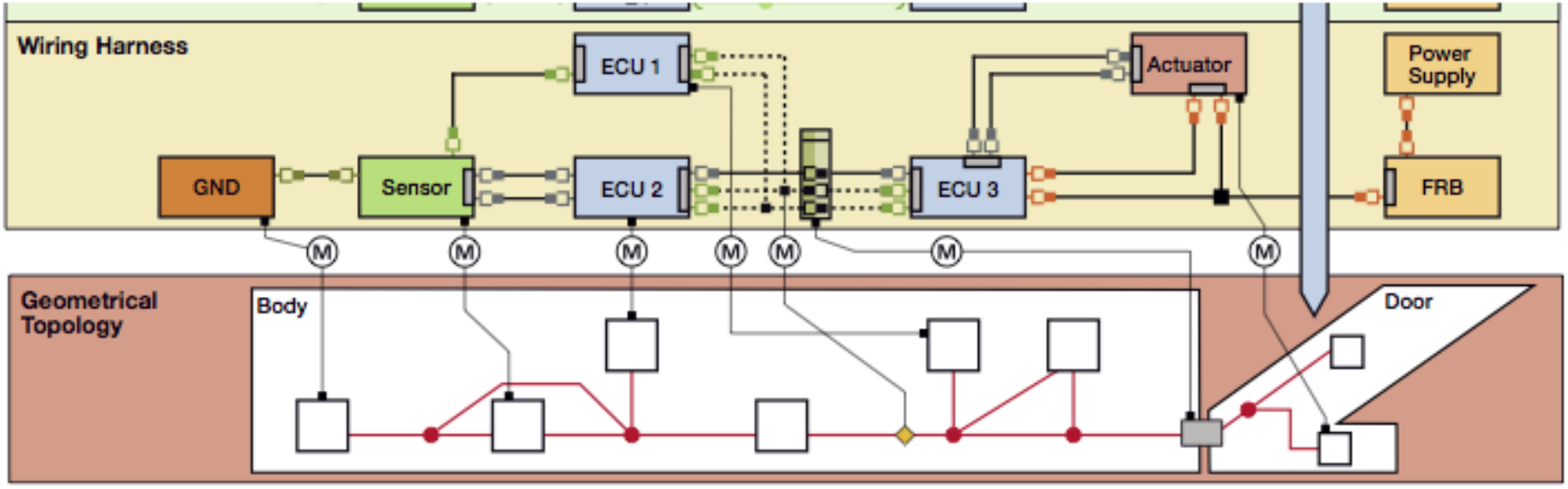
PreeVision Models in More Details

- Hardware architecture with LDF component model
- Electronic circuit design in ECU by ELOG



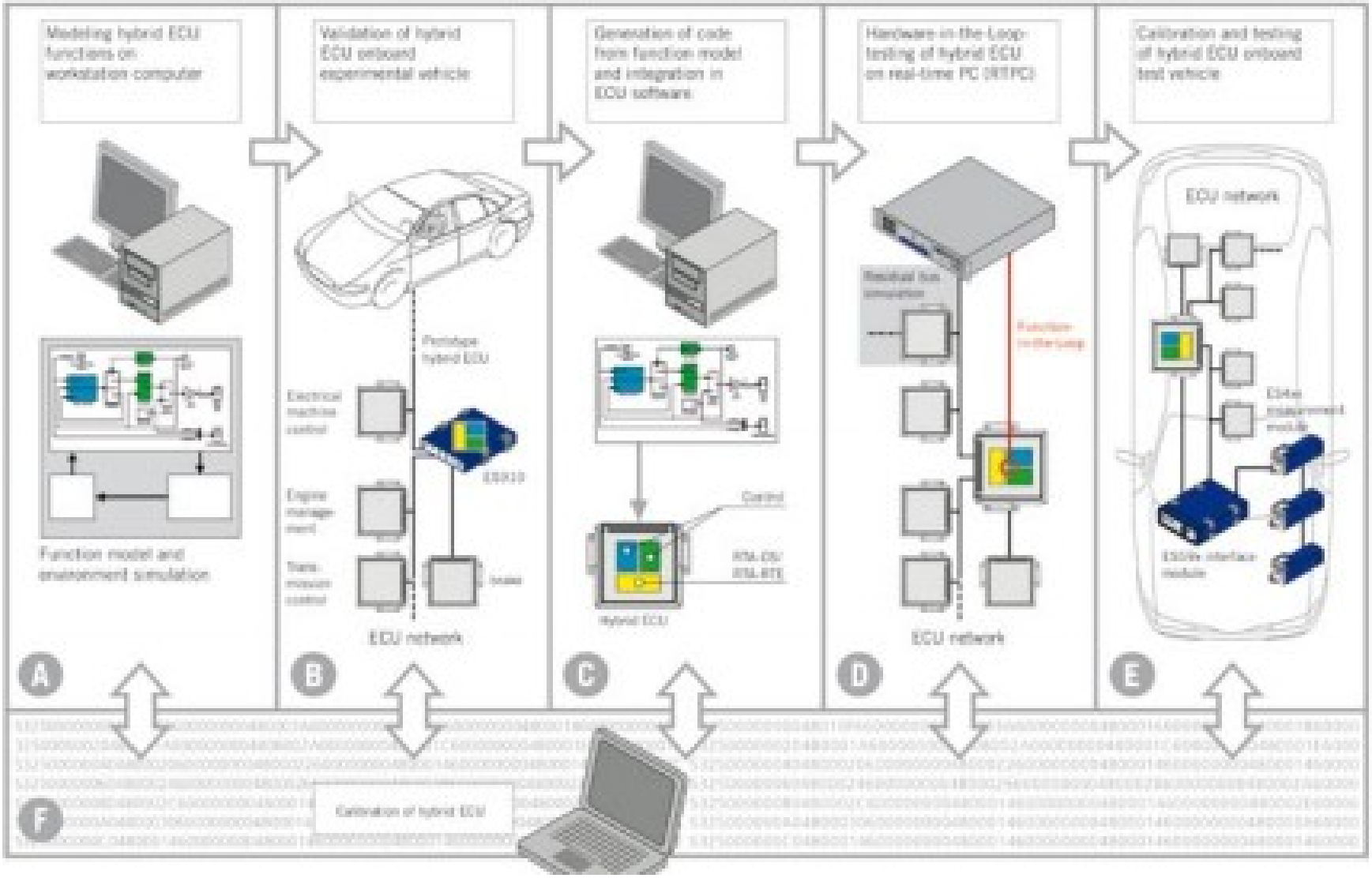
PreeVision Models in More Details

- Wiring in the car (physical network) with KBL
- 3-D CAD drawings for geometrical topology



Electric Cars (ETAS)

[ETAS] http://www.etas.com/en/products/ascet_md_modeling_design.php



CPS Software Factories (CPS IDE, Design Tools, CPS Tool Chains) are a Sign of a Maturing Productivity Industry

Age of invention

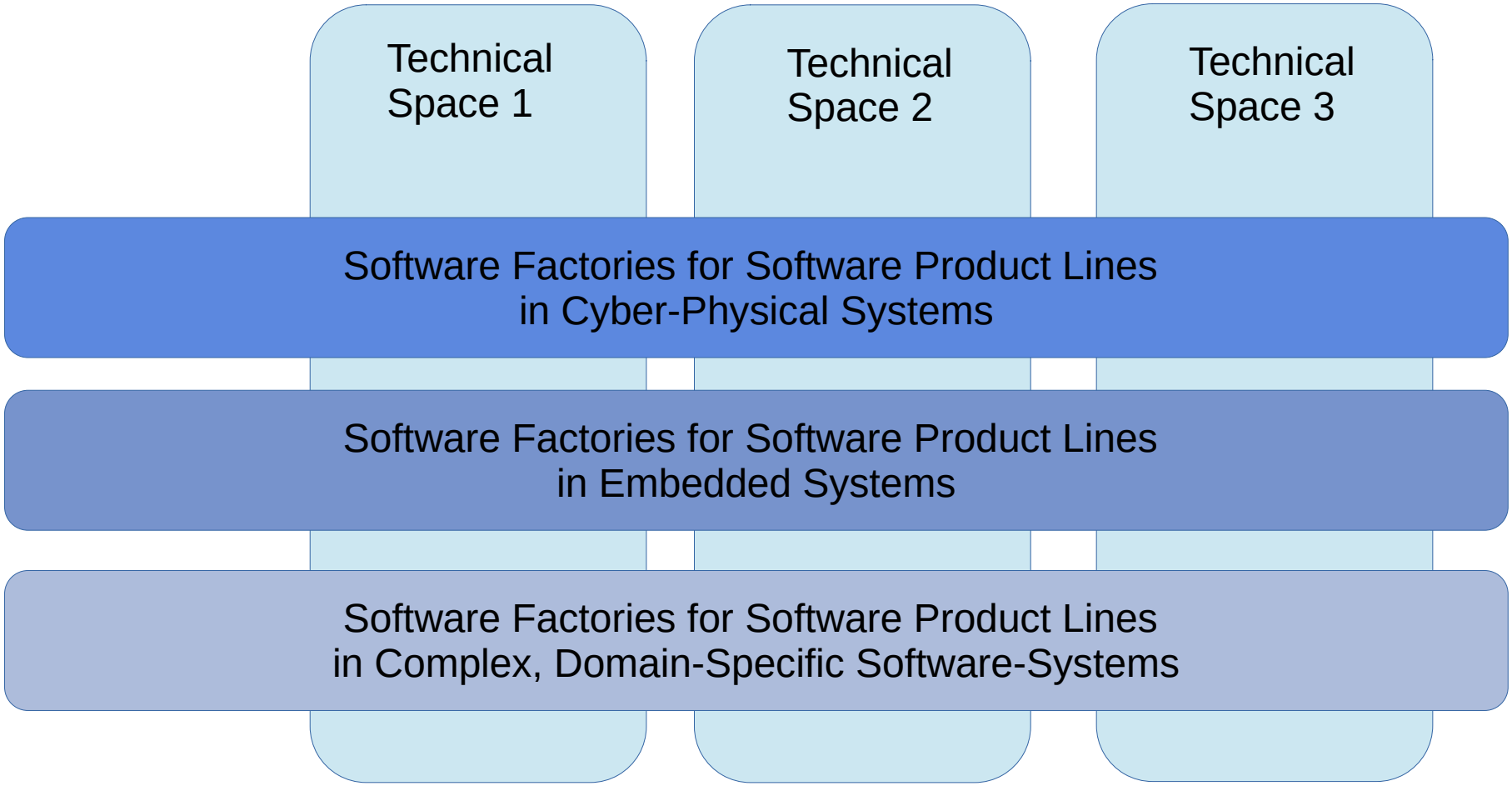
Age of implementation

Age of integration

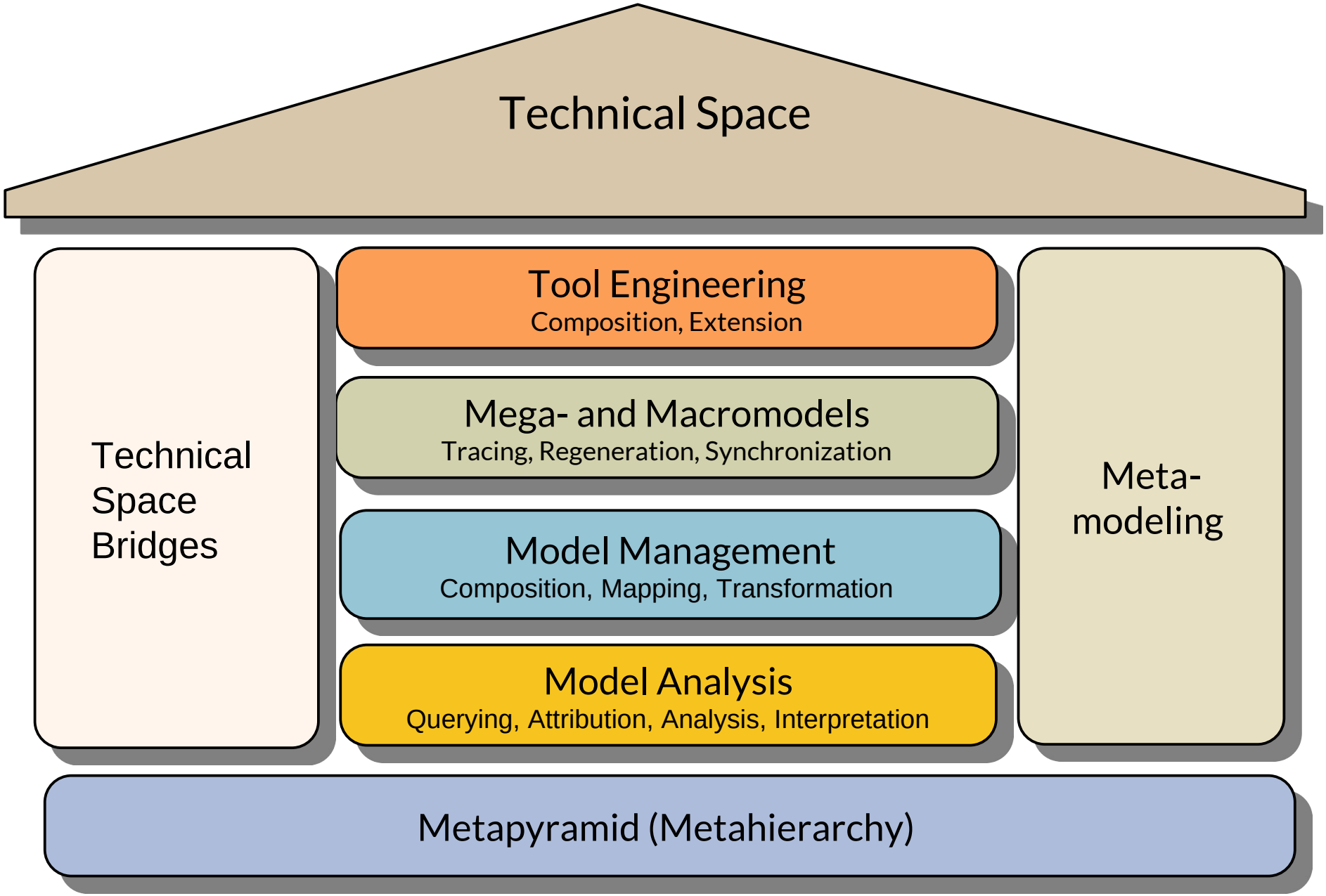
Will hold for all domains of CPS!

2.4 Why Do We Need Software Factories and MDSD in TS?

(Heterogeneous) Software Factories



Q10: The House of a Technical Space

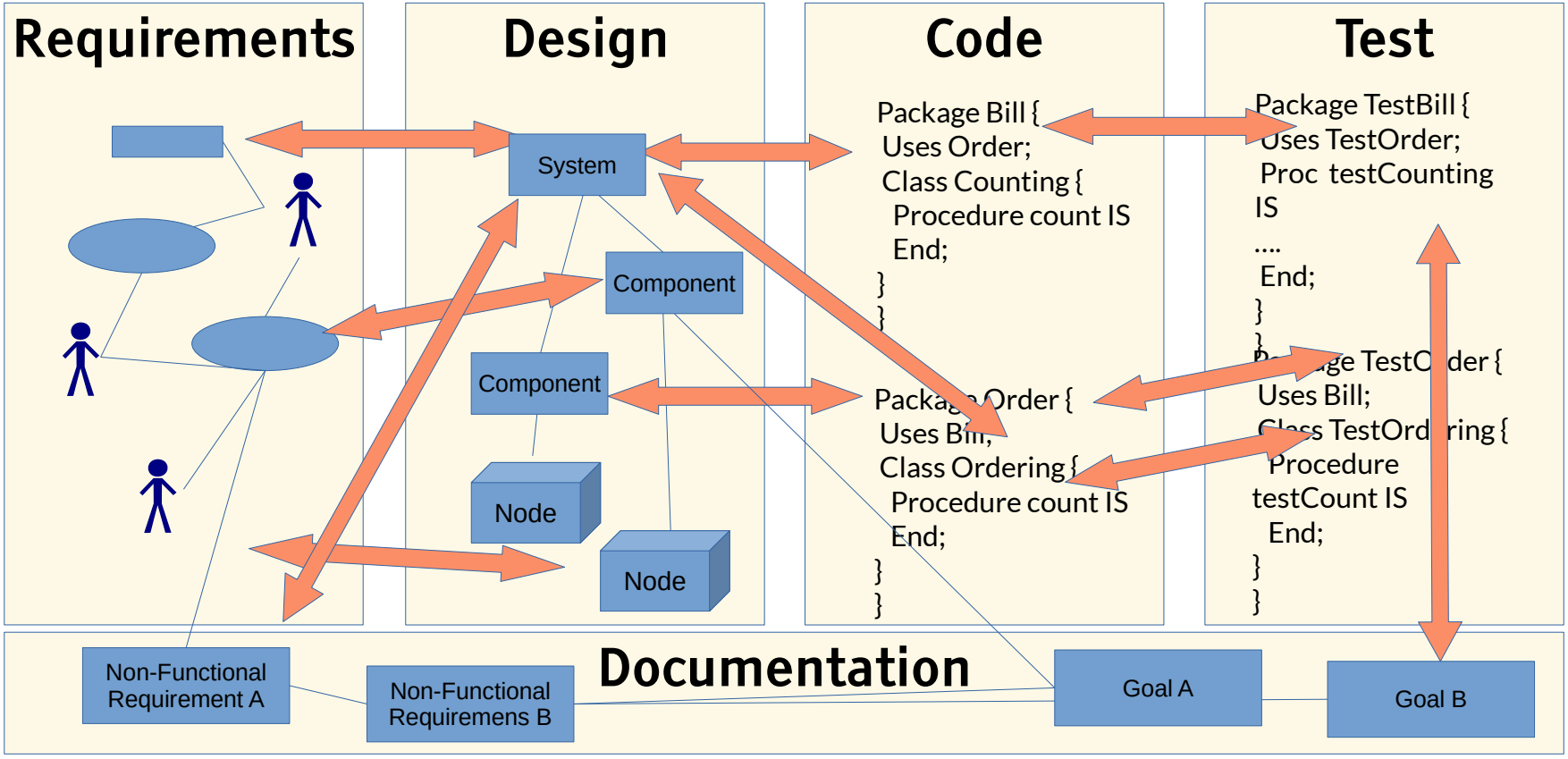


Q11: Overview of Technical Spaces in the Classical Metahierarchy

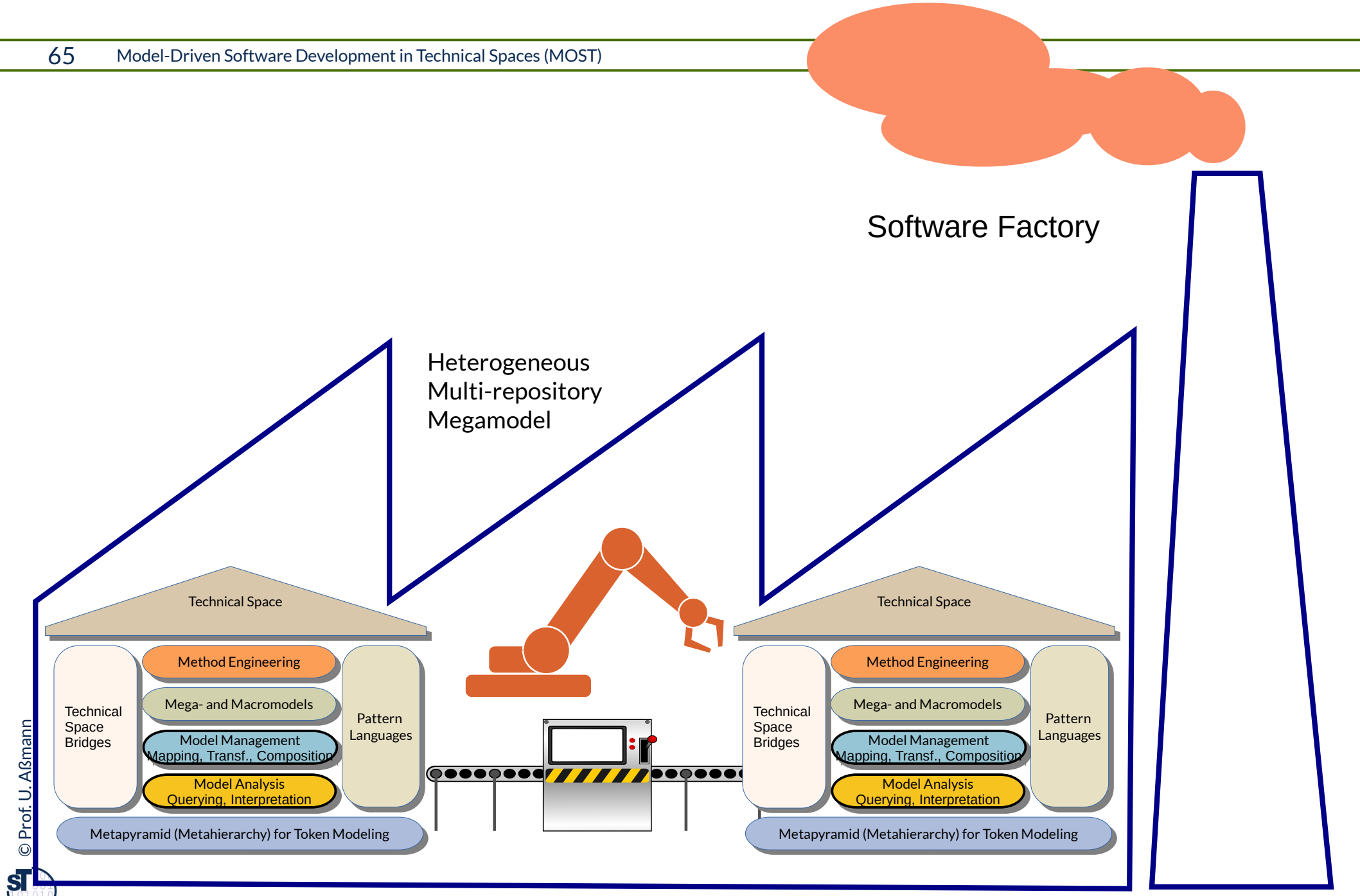
	Grammarware (Strings)	Text-ware	Table-ware		Treeware (trees)	Link-Tree-ware		Graphware/ Modelware			Role-Ware	CROM-Ware	Ontology-ware
	Strings	Text	Text-Table	Relational Algebra	NF2	XML	Link trees	MOF	Eclipse	CDIF	MetaEdit+	Context-role graphs	OWL-Ware
M3	EBNF	EBNF		CWM (common warehouse model)	NF2-language	XSD	JastAdd, Silver	MOF	Ecore, EMOF	ERD	GOPPR	CROM	RDFS OWL
M2	Grammar of a language	Grammar with line delimiters	csv-header	Relational Schema	NF2-Schema	XML Schema, e.g. xhtml	Specific RAG	UML-CD, -SC, OCL	UML, many others	CDIF-languages	UML, many others	CROM	HTML XML MOF UML DSL
M1	String, Program	Text in lines	csv Table	Relations	NF2-tree relation	XML-Documents	Link-Syntax-Trees	Classes, Programs	Classes, Programs	CDIF-Models	Classes, Programs	CROM models	Facts (T-Box)
M0	Objects	Sequences of lines	Sequences of rows	Sets of tuples	trees	dynamic semantics in browser		Object nets	Hierarchical graphs	Object nets	Object nets	Context-Object-Role Nets	A-Box (RDF-Graphs)

Q12: The ReDoDeCT Problem and its Macromodel

- The **ReDoDeCT problem** is the problem how requirements, documentation, design, code, and tests are related (→ V model)
- Mappings between the Requirements model, Documentation files, Design model, Code, Test cases
- A **ReDoDeCT macromodel** has maintained mappings between all 5 models



Q13: A Software Factory's Heart: the Multi-TS Megamodel



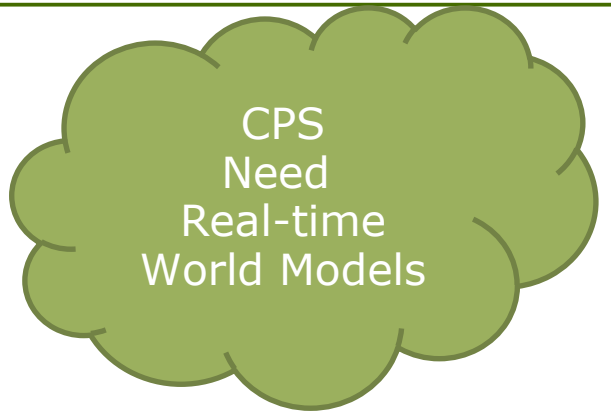
The End

- Why are future CPS a good application area for model-driven software development?
- Explain the model-driven tool chain Preevision, which problems about heterogeneous software systems it solves
- Why are CPS based on collaboration, contexts and roles?
- Why is modeling important for CPS?

Important World Models of “World Databases” (Monitoring CPS)

Physical Location of Thing in Environment

- Where is my thing in space?
  - Model of Physical Environment required
  - spatial, real-timed
  - magnetic, heat, humidity, user-defined
  - Continuous models

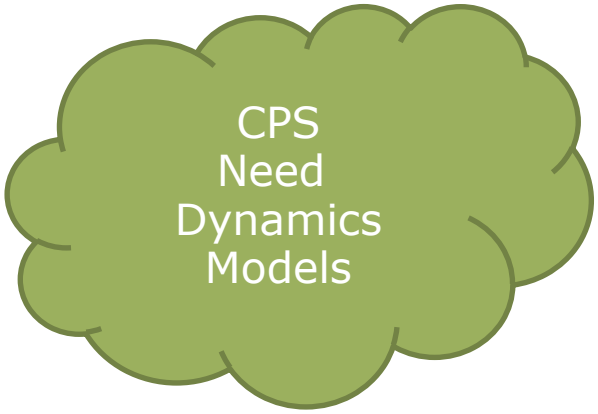


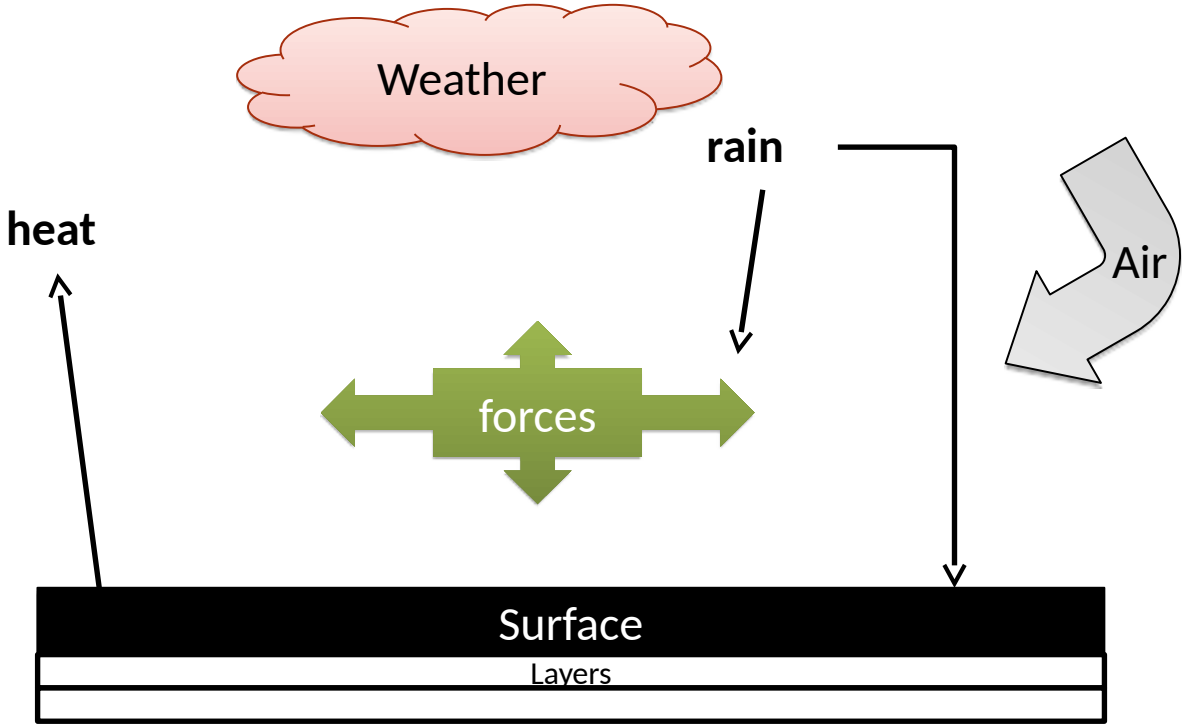
3D office models
Building models
City models
http://www.turbosquid.com

http://tf3dm.com/3d-model/the-city-39441.html

Physical Dynamics (Movement) of Thing

- How does it move in space?
  - Continuous modeling languages (Modelica)
  - Www.modelica.org, www.openmodelica.org

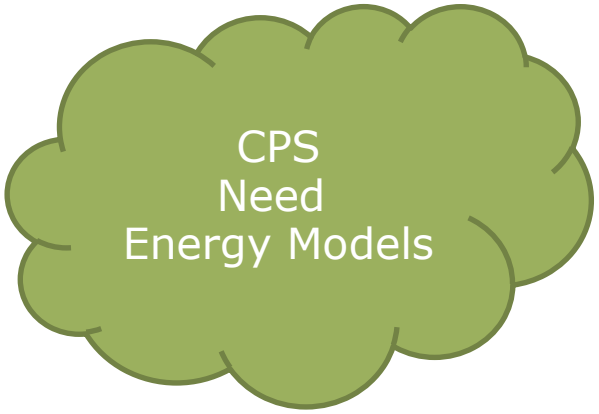


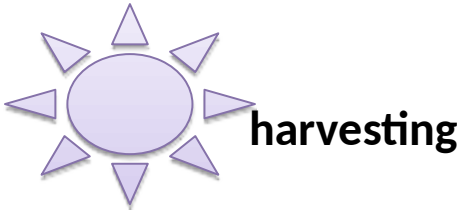


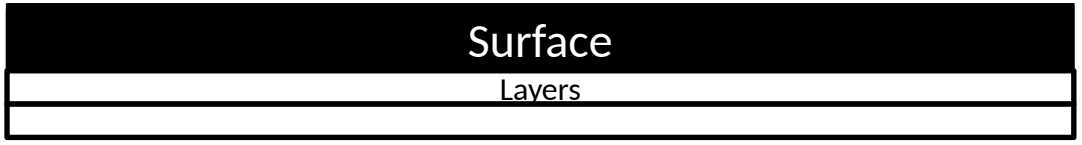
complex interplay of
- surface props
- weather: wind, rain, heat

Energy Consumption of Thing

- How much energy is left for its tasks?

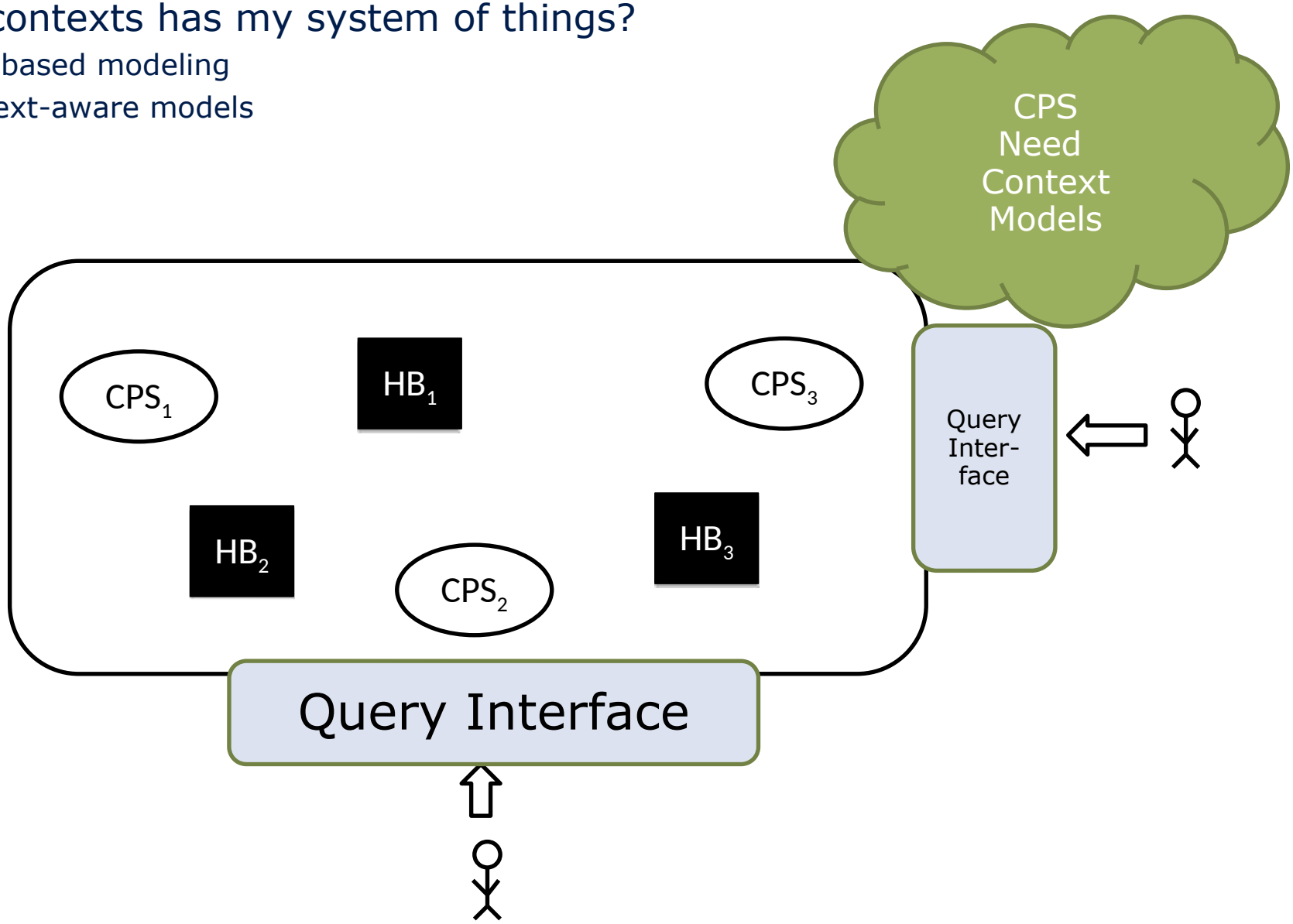






Current Physical Composition of a Thing

- Which contexts has my system of things?
  - Role-based modeling
  - Context-aware models



A Simple CPS: Cloud Robots

A Cloud Robot uses a Standard Robotic Platform
Hello, I'm NAO

Made by

- ALDEBARAN Robotics Paris, Frankreich
  [http://www.aldebaran-robotics.com/]

Application fields

- Teaching (Robot programming)
- Research
  - Robotics, AI
  - RoboCup
  - **Software Engineering**

Price

- 9.000 – 12.000 €

Nao Fact Sheet

Length: 58cm

Weight: 5kg

Hardware:
- x86 AMD GEODE 500MHz
- 256MB RAM
- 21 motors
- Battery 55Wh

OS:
Embedded Linux 32bit

Turtle Bot

Kinect

Netbook

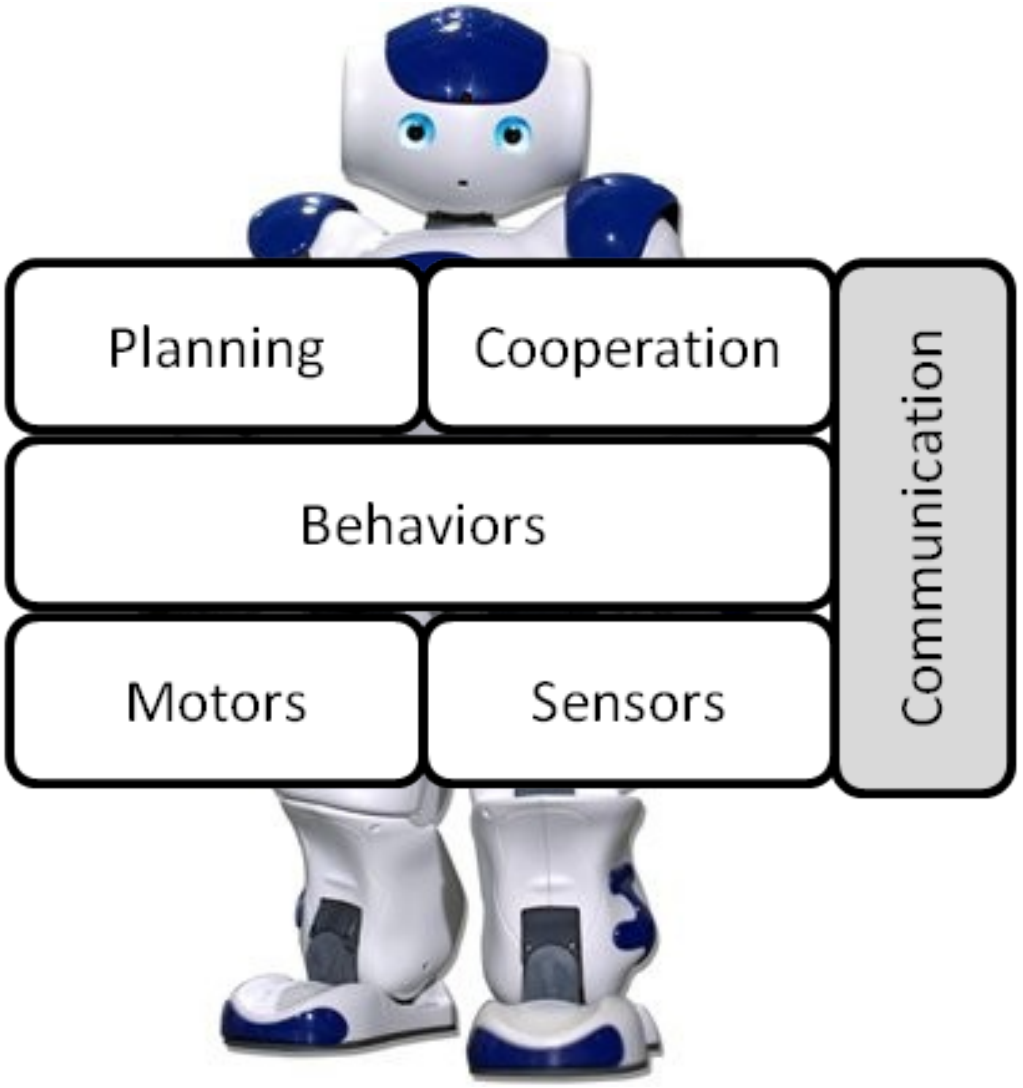
Roomla

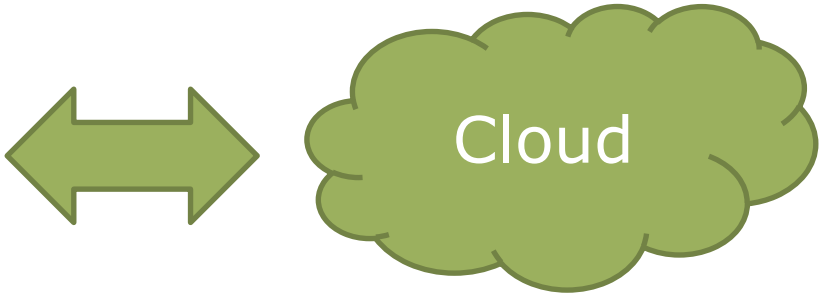
Kobuki

50kHz Sensor data rate

http://wiki.ros.org/Robots/TurtleBot
http://www.turtlebot.com

ResUbic Lab: NAO Web Service Architecture





http://code.google.com/p/naoservice/

NAO Web Service and Communication Framework

Runs in Cloud

Generated

Client (Java, NAOText)

Nao Utility Classes (Java)

Java Nao Web Service Proxies (Java)

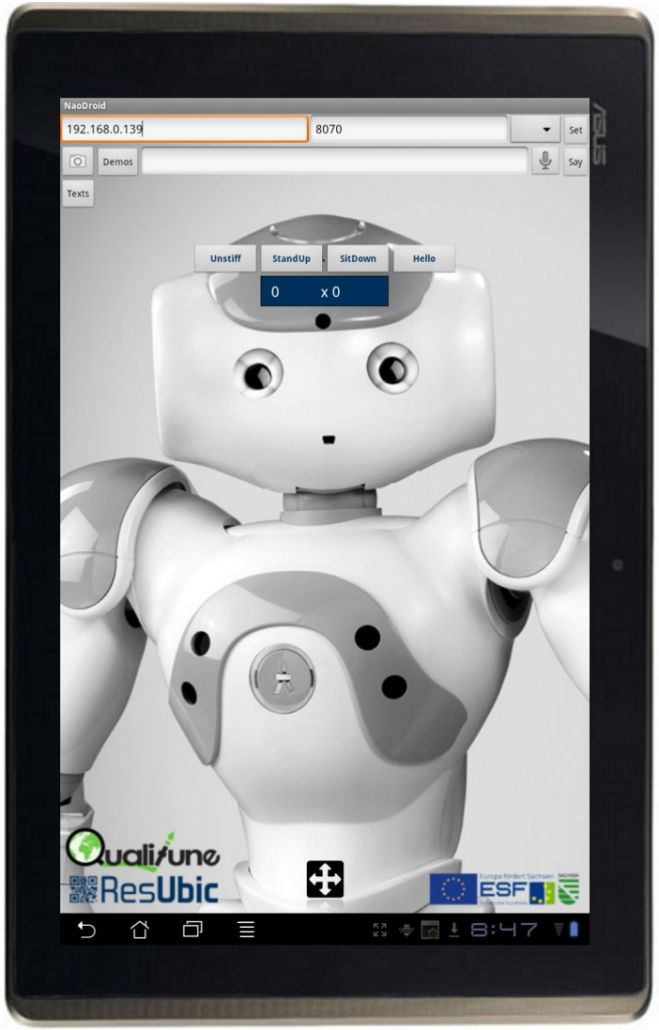
HTTP

Runs on Nao

Nao Web Service (Python, reflective)

Python Bridge for NaoQi (Python)

NaoQi (C++)



https://github.com/max-leuthaeuser/naoservice

A Killer App for Cloud Robots: Donut Production in „Nachtsprung“

Donuts Should be Individual....

And

https://www.flickr.com/photos/amiga-commodore/10059167335/

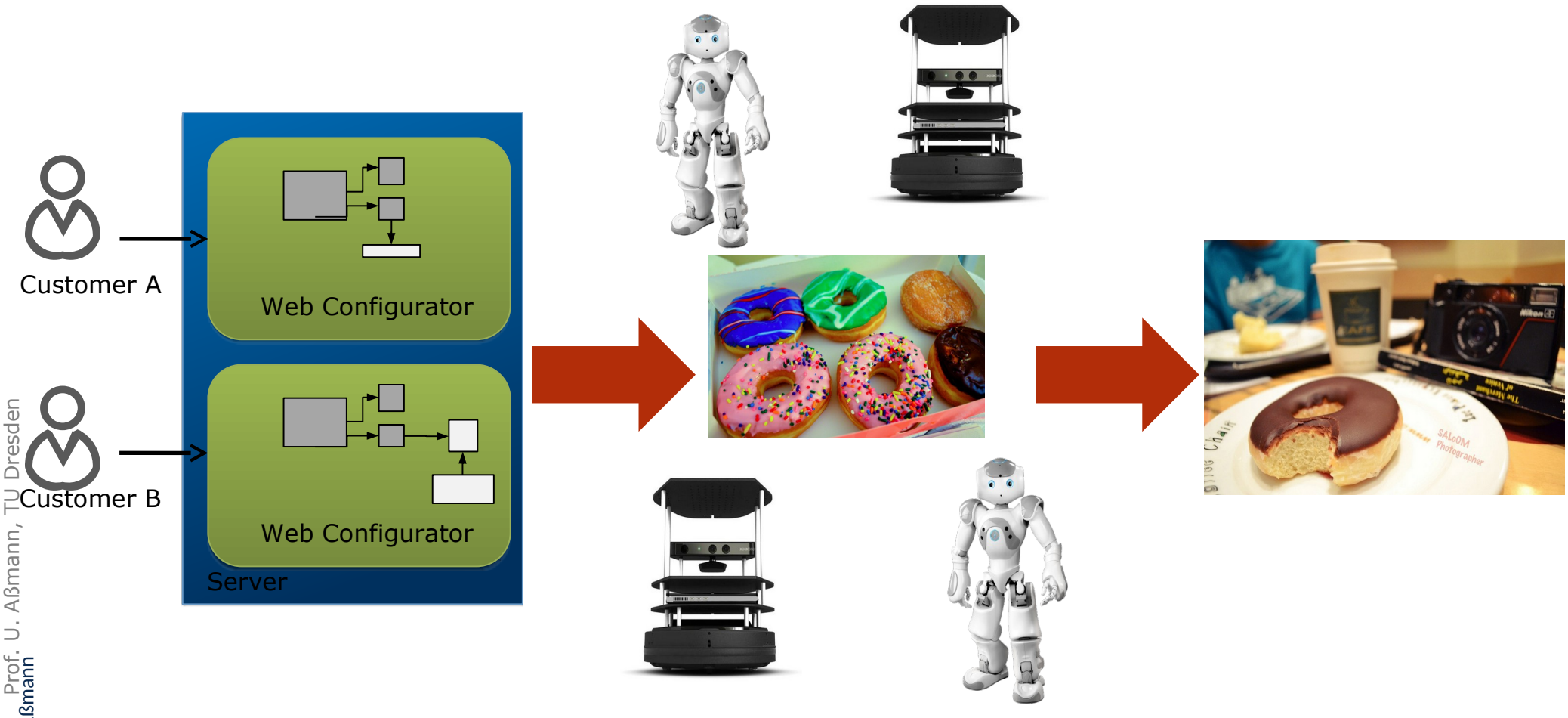
Situation Today

https://www.flickr.com/photos/jeades/2383525381/

- Mass production
- No individual configuration
- No fast, individualized production
- No „Nachtsprung“

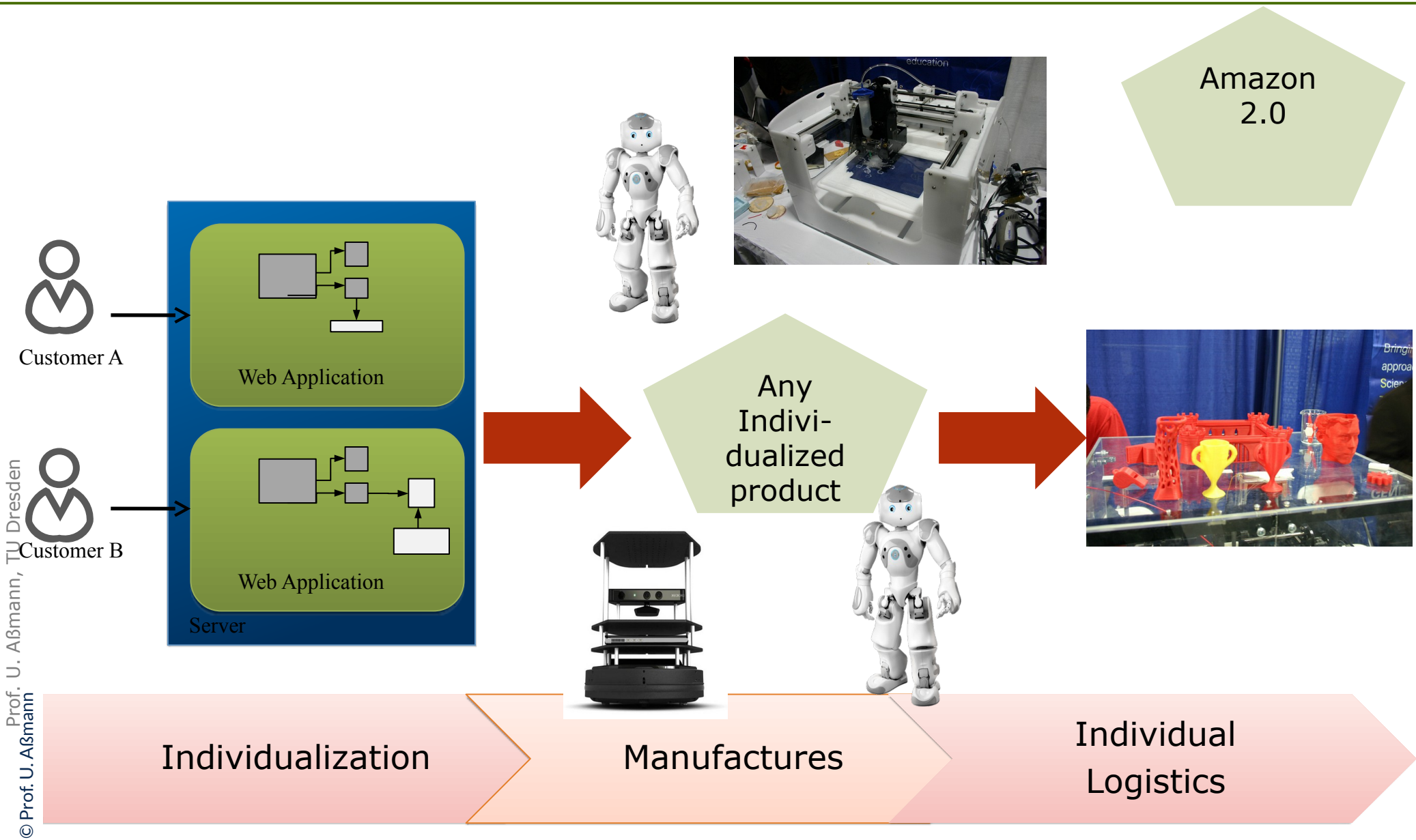
Donut Industry-4.0: Pulling Individual Donuts out in Nachtsprung

https://www.flickr.com/photos/soso__1991/7179199134/



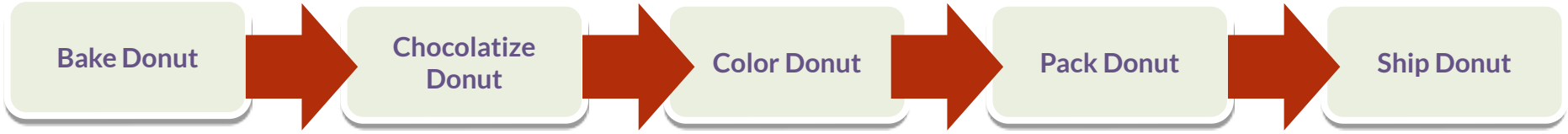
Configuring in the evening → Producing in the night → Shipping in the early morning

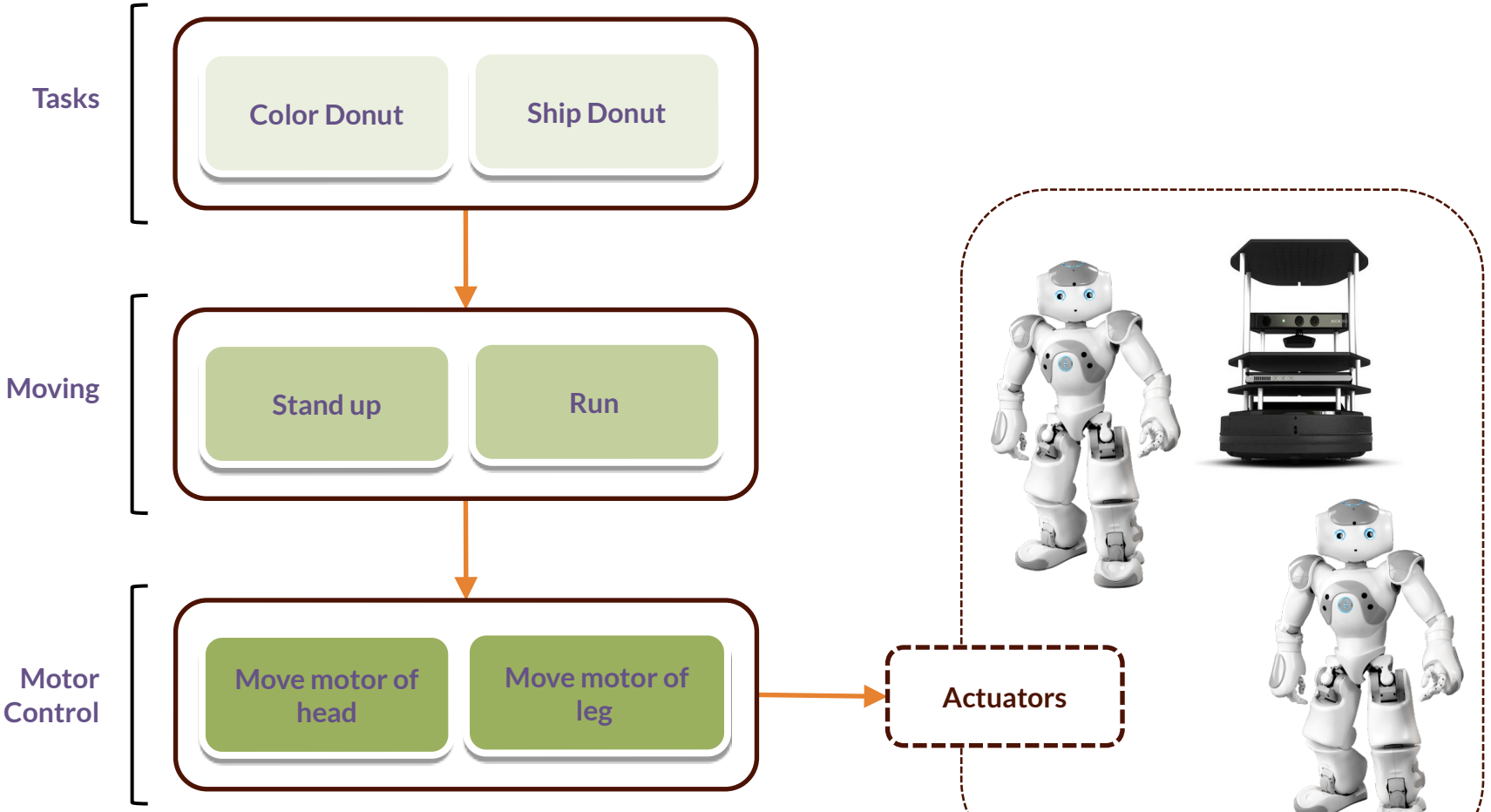
Industry-4.0: Economic Consequences



https://www.flickr.com/photos/ideonexus/7311856946/
https://www.flickr.com/photos/ideonexus/7311859510

Industry-4.0:
Cloud Robots Produce Things in Workflows





2. Heterogeneous Applications for MOST - Design Tools for Complex Systems

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
http://st.inf.tu-dresden.de/teaching/most
WS 21-0.2, 20.11.21

1) Motivation for MOST
2) Design Tools for Complex Software Systems
3) Design of CPS with Domain-Specific CPS tool chain
    1) Cyber-physical systems (CPS)
4) Why Software Factories?

Obligatory Literature

- [Preevision] Vector. Modellbasierte Elektrik-/Elektronik-Entwicklung vom Architekturentwurf bis zur Serienreife. Preevision Handbuch
  - http://vector.com/portal/medien/cmc/marketing_items/web/91106.pdf
- [Reichmann] Clemens Reichmann, Daniel Gebauer, Klaus D. Müller-Glaser. Model Level Coupling of Heterogeneous Embedded Systems. Technical Report, FZI, 2008
  - http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.366
- [ETAS] Ulrich Lauff, Christoph Stoermer, Thomas Dollmaier, Mathias Klauda. ETAS GmbH, Stuttgart, Germany. Development Tools for Hybrids and Electric Cars.
  - http://www.etas.com/download-center-files/products_ASCET_Software_Products/1002_ATZ_elektronik_Entwicklungswerkzeuge_fuer_HEV_EV_EN.pdf

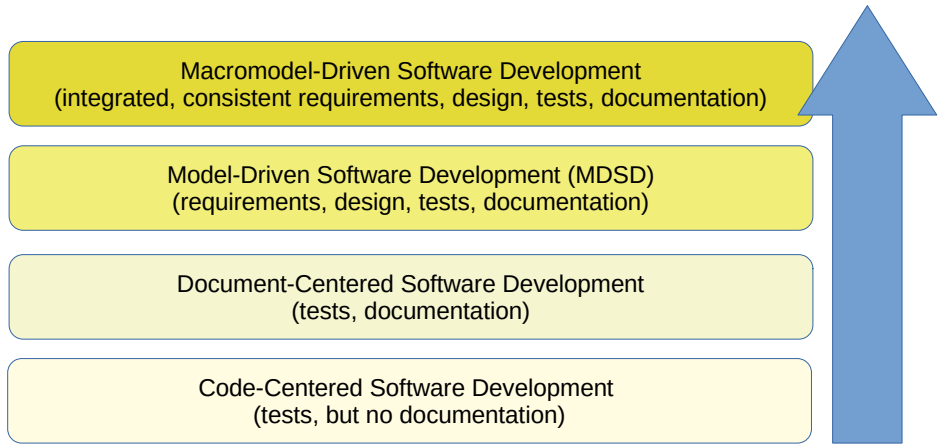
Literature

- [Zverlov] Sergey Zverlov. Comparison of two level-based Approaches for the Development of Embedded Systems. Bachelor Thesis in Computer Science. TU München, 2008.
- [Wurman] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. AI Magazine Volume 29 Number 1 (2008) (© AAAI)
- [MüGl09] Prof. Dr.-Ing. Klaus D. Müller-Glaser. Slide set. Model-Driven Engineering for Automotive Systems. UCSD SAASE 2009
    - http://jacobsschool.ucsd.edu/GordonCenter/g_leadership/l_summer/docs/saase/symposium-presentations/KlausMuellerGlaser.pdf

2.1 Example for Heterogeneous Software Factories: Integrated Development Environments for Large Software Systems (MDSD-Software-IDE)

Change in Software Development

- From code-only to documents to models to macromodels (integrated consistent multimodels)



What is needed for MDSD: Tool, Language, Process, Workflow and Method Engineering

Product-Line Engineering is the discipline of constructing domain-specific product families.

Software Ecosystem Engineering is the discipline of constructing open product platforms with appstores.

Tool Engineering is the discipline of constructing company-specific, domain-specific tools.

Language Engineering is the discipline of constructing company-specific, domain-specific languages for tools, processes, and workflows.

Process Engineering (Method Engineering) is the discipline of specifying and constructing methods and processes for a team of people to conduct a project.
Software Process Engineering (Software Method Engineering) focuses on software development processes.

Workflow Engineering is the discipline of running executable processes (workflows)
- For a team, in an application

Workflow engineering uses **behavioral languages.**

Design Tools:
Integrated Development Environment (IDE)
Software-Entwicklungsumgebungen (SEU)

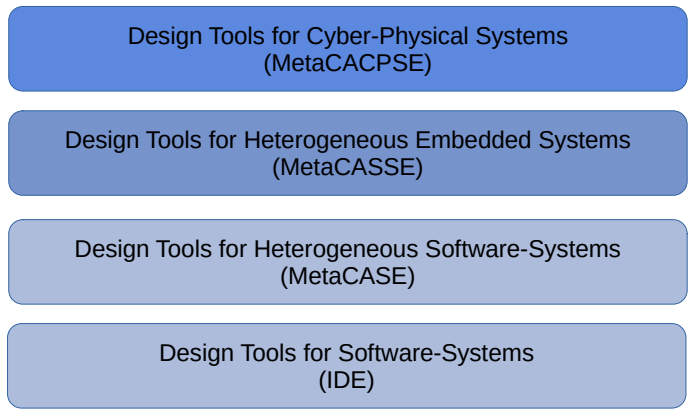
An **integrated development environment (IDE, Software-Entwicklungsumgebung, SEU)** consists of a **structured set of integrated standalone tools**
- to support a team in software development (process engineering)
- to construct a multimodel or macromodel.

- IDE support Computer aided Software Engineering (CASE)
- A MDSD-IDE (Meta-CASE) is complex software machine tool (Software-Werkzeugmaschine), an IDE for model-driven software development supporting
  - Many languages (DSL, metamodels) in a technical space
  - Heterogeneous software development
  - Model management system and Macromodel
- Other terms
  - Design Tools
  - Integrated Computer Aided Software Engineering (I-CASE)
  - Integrated Software Factory (ISF)
  - Software Engineering Environment System (SEES)
  - Integrated Project Support Environment (IPSE)
  - Integrated Software Engineering Environment (ISEE)

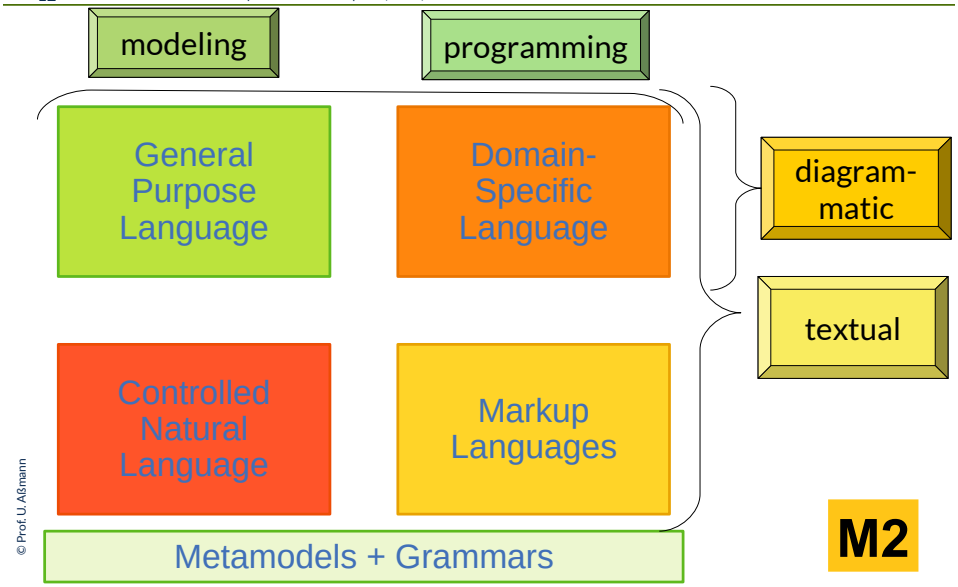
Nagl. M.: Software-Entwicklungsumgebungen: Einordnung und zukünftige Entwicklungslinien; Informatik-Spektrum 16(1993) H.5, S. 273-280

Design Tools can be Heterogeneous (Heterogeneous Software Factories)

- While IDE are hosted in *one* technical space, (Heterogeneous) Software Factories span several ones.

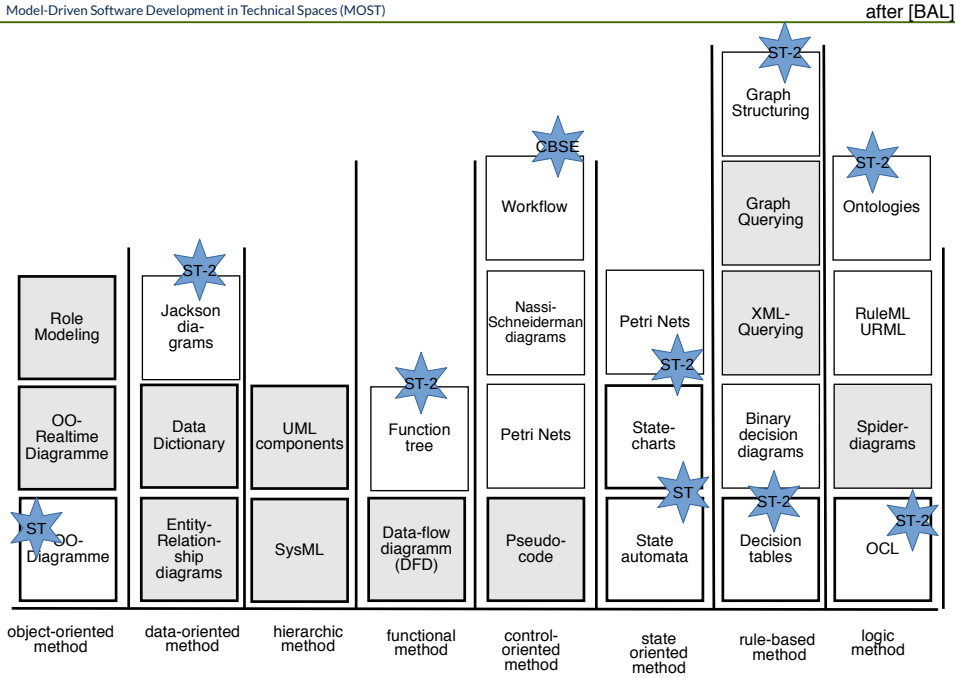


Q16: Languages in Software Factories are Built on Metamodels and Grammars



Basic Languages for Design Tools

after [BAL]



Problem for Companies: Building Domain- and Company-Specific Software Tools is Expensive

Tool	Person years	Cost in kEuro
Compiler	1-2	100
Optimizer	1-3	150
Back-End	0.5-1	100
Compiler component framework	20	1000
UML-IDE	5	250
Java-Refactorer	2-4	200
Energy Unit Test-Framework	1	50
Tool for Requirments management	2-4	200
Mobile Phone Test-Framework	2	100

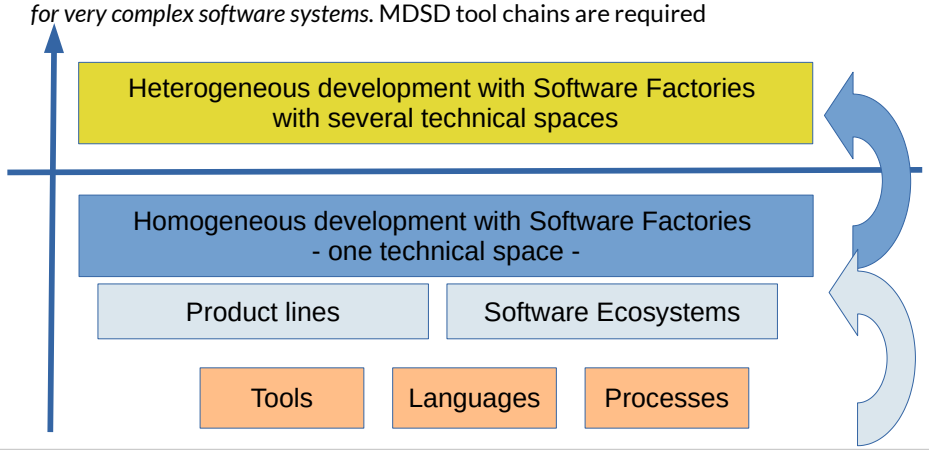
How to Master Tool, Language, Process, Method Engineering in a Company?

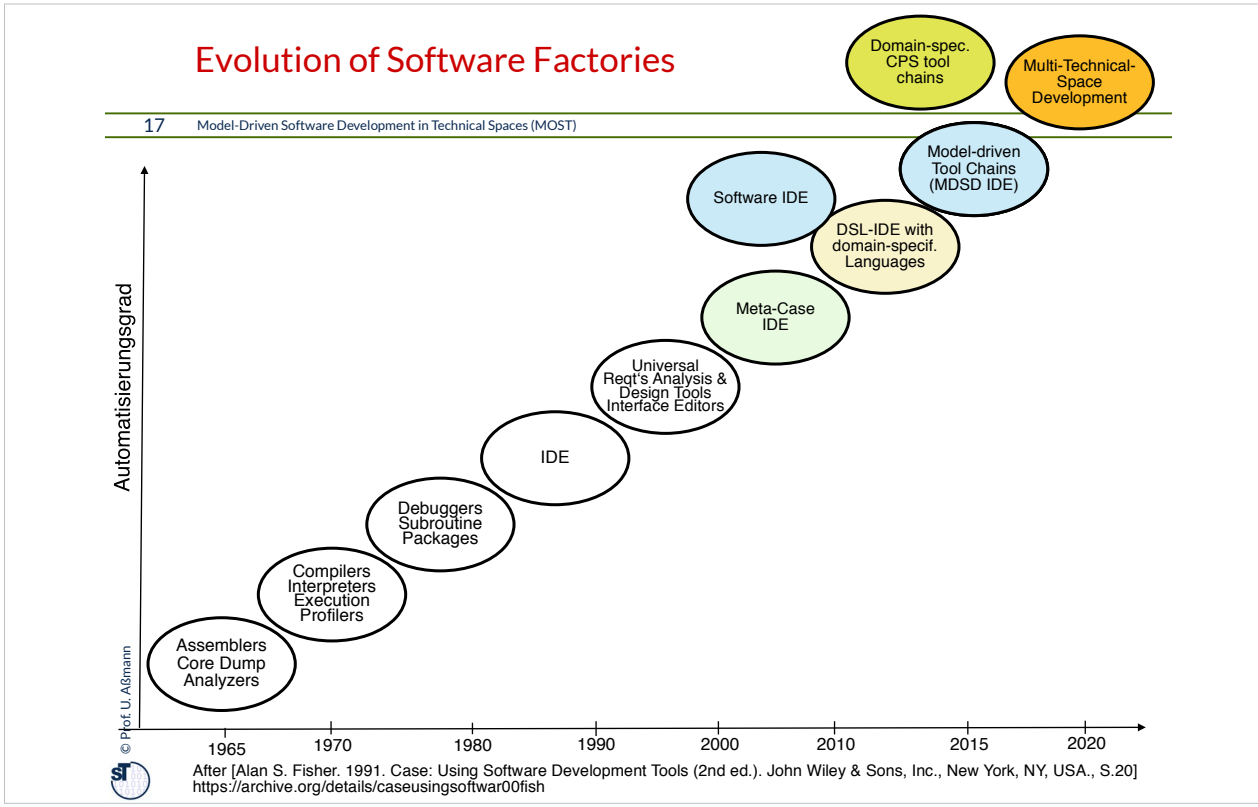
How can I create
simple tools, methods, languages,
and
reuse them for more complex tools, methods, languages
in my company?

Answer:
Mastering a **software factory** in a **technical space**
creating and composing
- Metamodels of base languages (on M2)
- Models (on M1)
- Repositories (on M0)

Maturity Levels of Software Companies

- Many companies do not know technical spaces nor software factories
- Many companies work with *homogeneous software development* in *one technical space*
- Some companies master *heterogeneous software development in one technical spaces for complex software systems.* Tools are required
- Some companies master *heterogeneous software development in several technical spaces for very complex software systems.* MDSD tool chains are required





Development with multiple technical spaces comes into focus (heterogeneous software development)

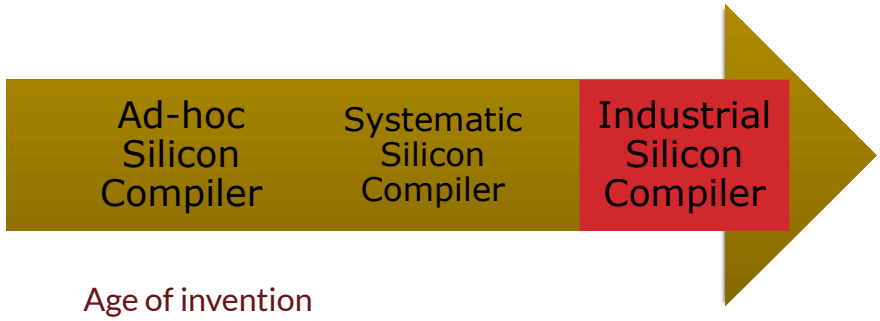
2.2. Example 2 of Software Factory: “Silicon Compilers”

Example 1: MDSD ToolChain: Silicon Compilers

- [Wikipedia:Silicon_Compiler] A **silicon compiler** is a software system that takes a user's specifications and automatically generates an integrated circuit (IC). The process is sometimes referred to as hardware compilation.
- [Wikipedia:Design_flow_(EDA)]
- Alberto Sangiovanni-Vincentelli distinguished three periods of EDA [Tides]:
- **"The Age of Invention:** During the invention era, routing, placement, static timing analysis and logic synthesis were invented.
- **The Age of Implementation:** In the age of implementation, these steps were drastically improved by designing sophisticated data structures and advanced algorithms. This allowed the tools in each of these design steps to keep pace with the rapidly increasing design sizes. However, due to the lack of good predictive cost functions, it became impossible to execute a design flow by a set of discrete steps, no matter how efficiently each of the steps was implemented.
- **The Age of Integration:** This led to the age of integration where most of the design steps are performed in an integrated environment, driven by a set of incremental cost analyzers."

Example 1: How the Silicon Compiler Industry Matured over Time

- Sangiovanni-Vincentelli claims that other industries (e.g., for CPS) will go the same way[20]



Age of invention

Age of implementation

Age of integration

[Sangiovanni-Vincentelli Tides]

2.3. Example 3: Software Factories for Cyber-Physical Systems

2.3.1. What is a Cyber-Physical System (CPS)?

Kiva Bots for Logistics

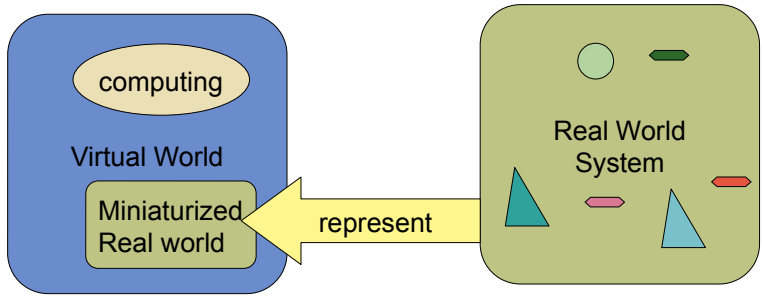
- [Wurmer] Just search on YouTube for Kiva Systems
- https://www.youtube.com/watch?v=8gy5tYVR-28
- https://www.youtube.com/watch?v=6KRjuuEVEZs

Smart Parking

http://commons.wikimedia.org/wiki/File:Bundesarchiv_Bild_183-H0605-0007-001,_Rostock,_Ernst-Th%C3%A4lmann-Platz,_Parkplatz,_Marienkirche.jpg#mediaviewer/File:Bundesarchiv_Bild_183-H0605-0007-001,_Rostock,_Ernst-Th%C3%A4lmann-Platz,_Parkplatz,_Marienkirche.jpg

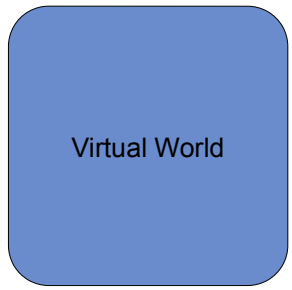
„Standard“ Computing

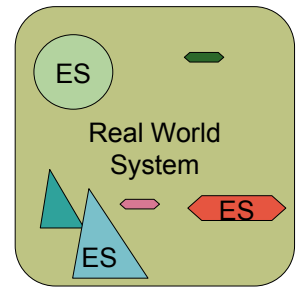
- „Standard“ Computing maps the real world into the computer and computes about it by simulation



Embedded System

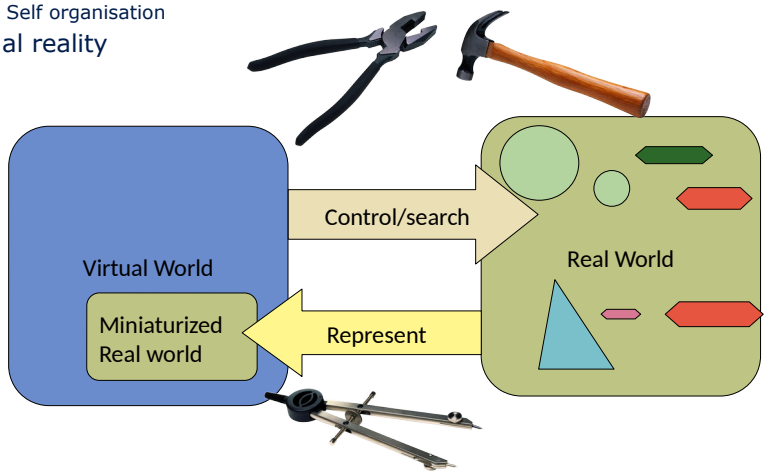
- The computer is integrated into the real-life object





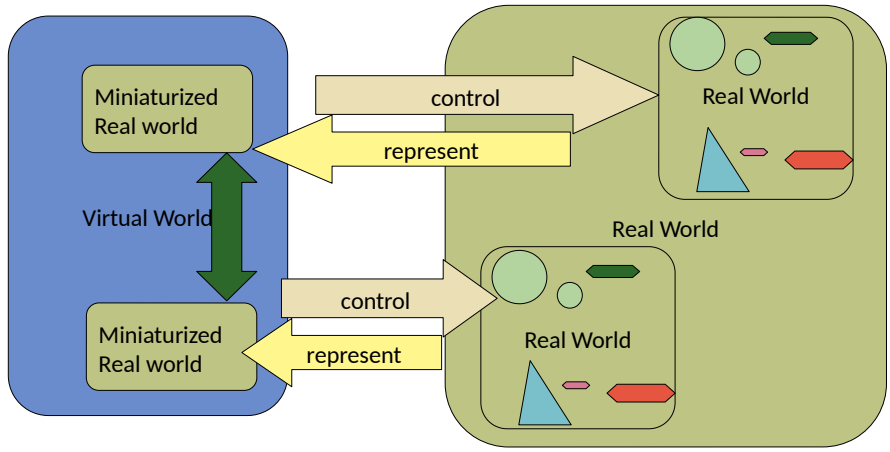
Cyber-Physical System (CPS)

- Simulation of intelligent things in space and time
  - Search possible
- Control of the intelligent things in space and time
  - Self regulation
  - Self optimization
  - Self organisation
- Dual reality



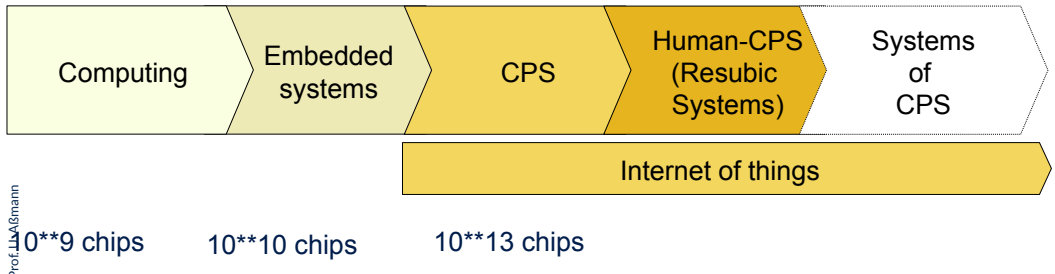
The Internet of Things

- Systems of CPS, i.e., remote tools



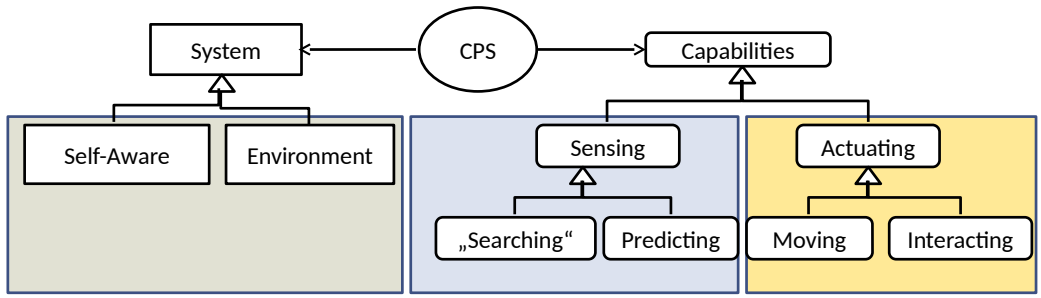
Trend CPS

- Cyber-physical systems are the first step in the internet of things

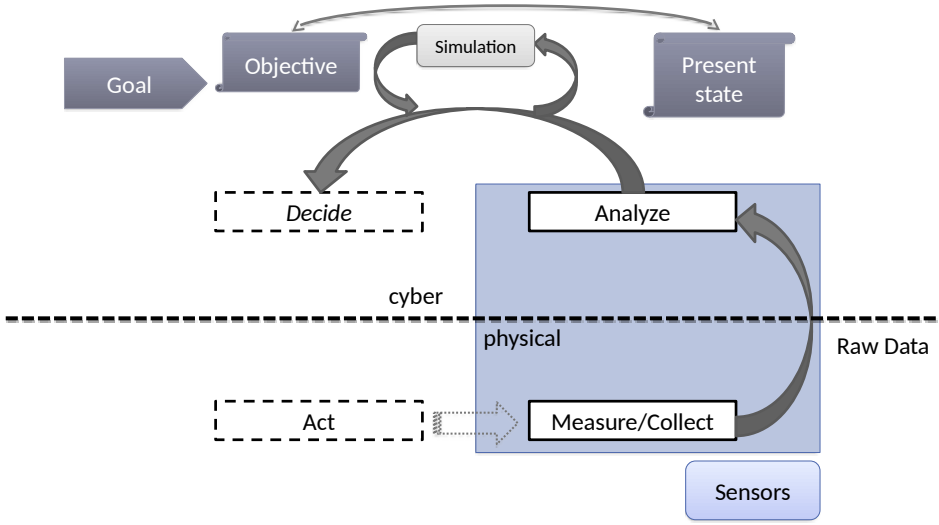


10**9 chips 10**10 chips 10**13 chips

Two Classes of Cyber-Physical Systems for Cyber-Physical **Search** and **Management**



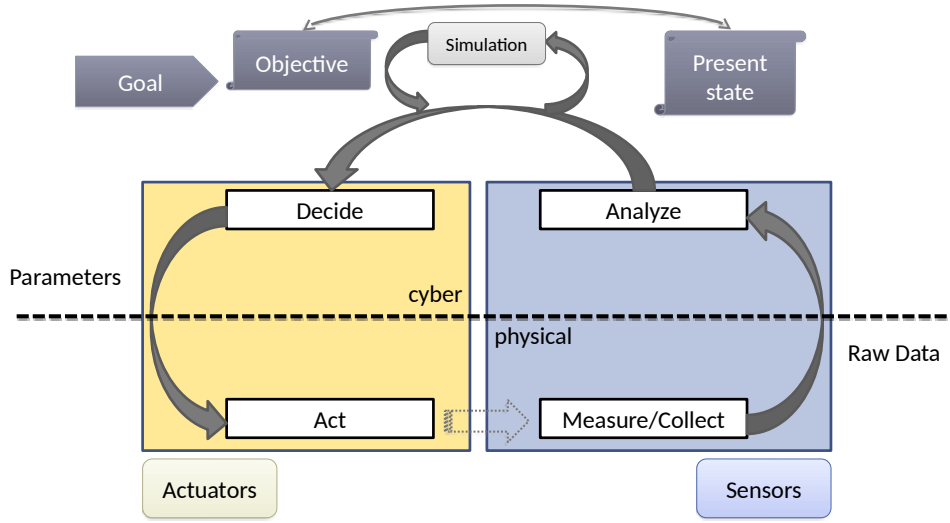
Cyber-Physical Database Systems = Analysis, Simulation and Prediction



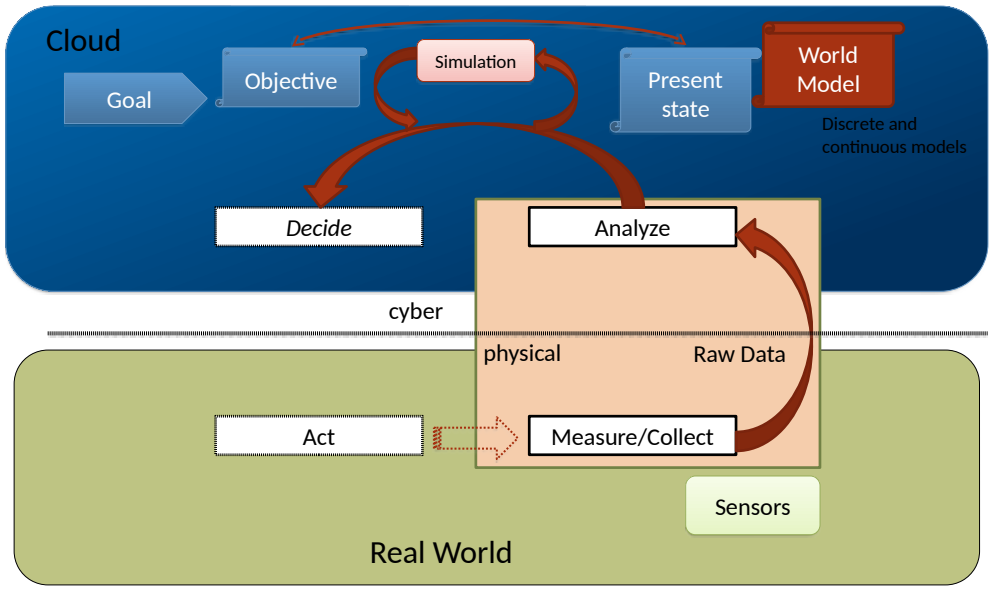
A Cyber-Physical System

http://commons.wikimedia.org/wiki/File:Traffic_seen_from_top_of_Arc_de_Triomphe.JPG

Cloud Robots =
Cyber-Physical Management Systems

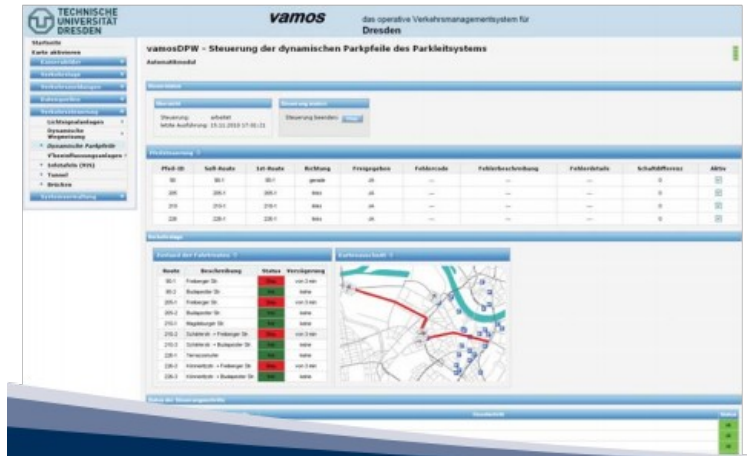


World Database Systems are Monitoring CPS (Analysis, Simulation and Prediction)

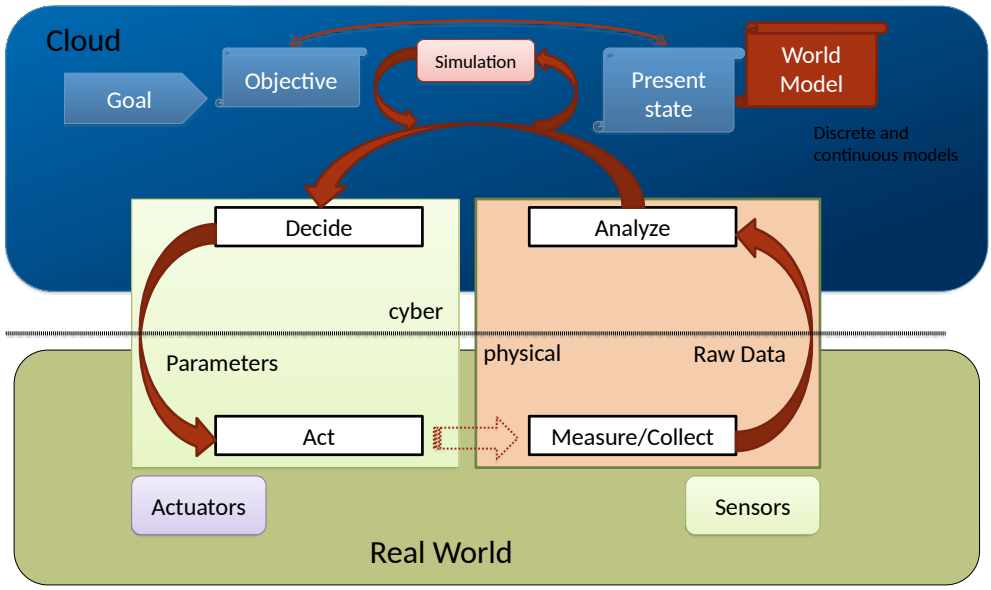


Ex.: The VAMOS Traffic Management System (Verkehrsleitsystem) Dresden

- Realtime data from the city's traffic
- http://www.vamosportal.de/
- http://wwwpub.zih.tu-dresden.de/~vamos/flyer/vamos_web.pdf

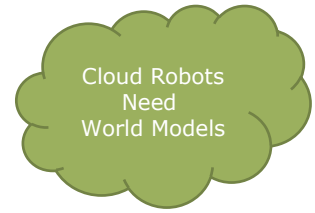


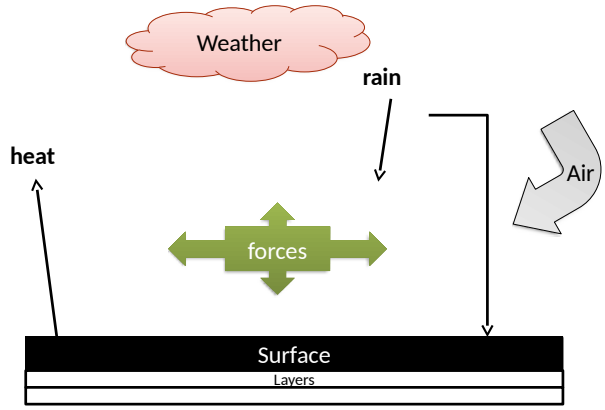
Cloud Robots are Controlling CPS



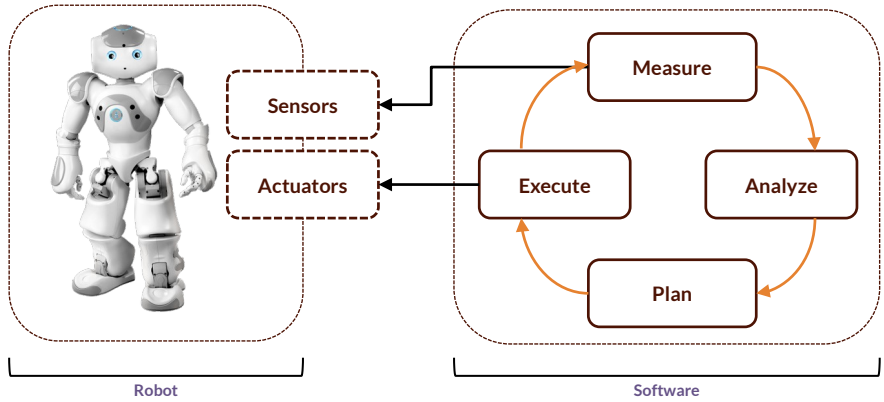
Physical Dynamics (Movement) of Cloud Robot

- How can I **control** a cloud robot move in space?



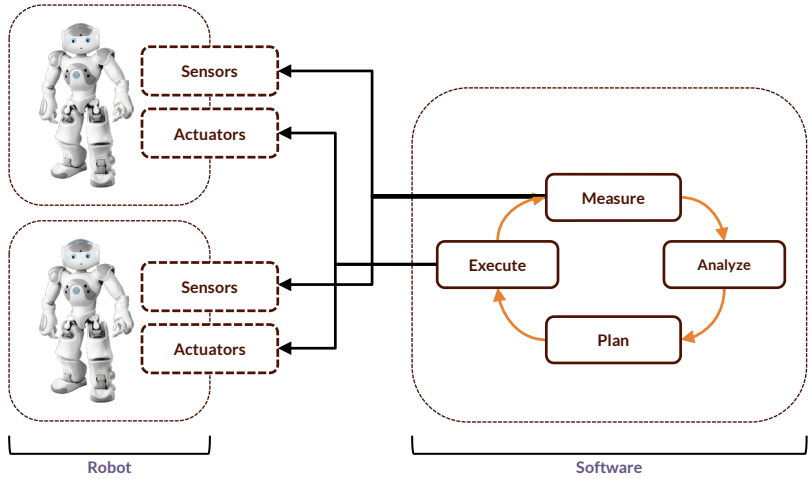


Cloud Robots are Adaptive Systems (MAPE Loop), and run a Dynamic Software Product Lines



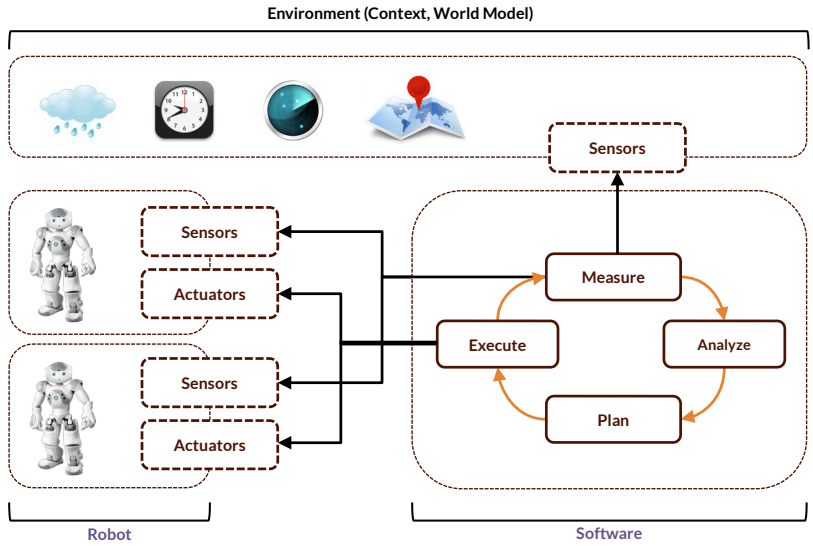
Elastic Architecture

Cloud Robots are Multi-Adaptive Systems



Elastic Architecture

Cloud Robots are Context-Adaptive Systems



Elastic Architecture

Industrie-4.0 (Smart Factory) with CPS

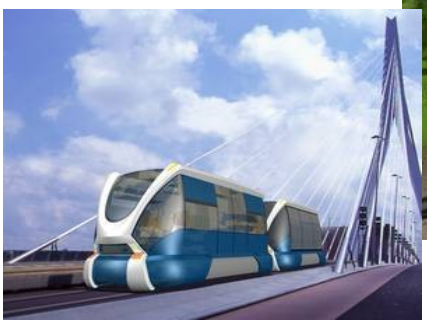
- Embedded System: machines, robots, presses, transport systems
- CPS: Autonomous control of the factory
  - Self assembly of the products
  - Autonomous control of logistics
  - Pull of products instead of push

http://commons.wikimedia.org/wiki/File:Mail_sorting_assembly_line.jpg
http://commons.wikimedia.org/wiki/File:Factory_Automation_Robotics_Palettizing_Bread.jpg?uselang=de

http://commons.wikimedia.org/wiki/File:Mail_sorting_assembly_line.jpg
http://commons.wikimedia.org/wiki/
File:Factory_Automation_Robotics_Palettizing_Bread.jpg?uselang=de

Smart Traffic/Transport/Logistics mit CPS

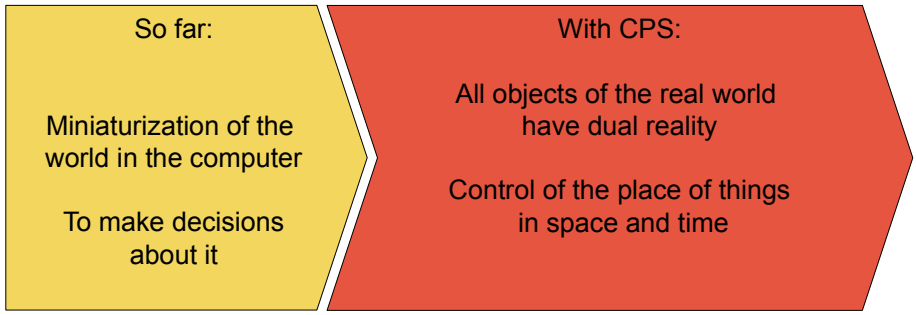
- Embedded System: Railcabs are autonomous train cars (Paderborn)
- CPS: Optimization of the German logistics



http://www.railcab.de

The Revolution of CPS

- All domains in transport, logistics, assembly, housing, cities will change
- Nothing will stay as it is
- All engineering disciplines will change until 2020

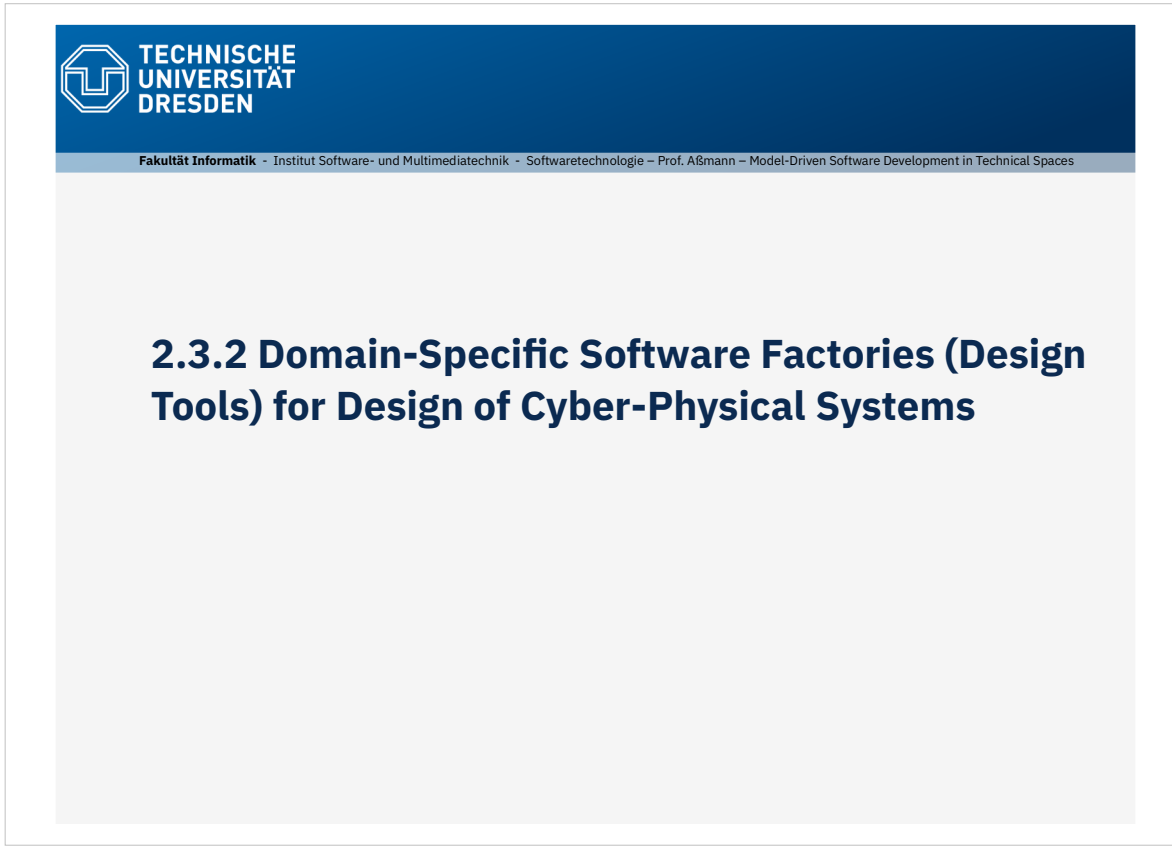


Questions

How can we build such complex tool suites for CPS (CPS-IDE)?

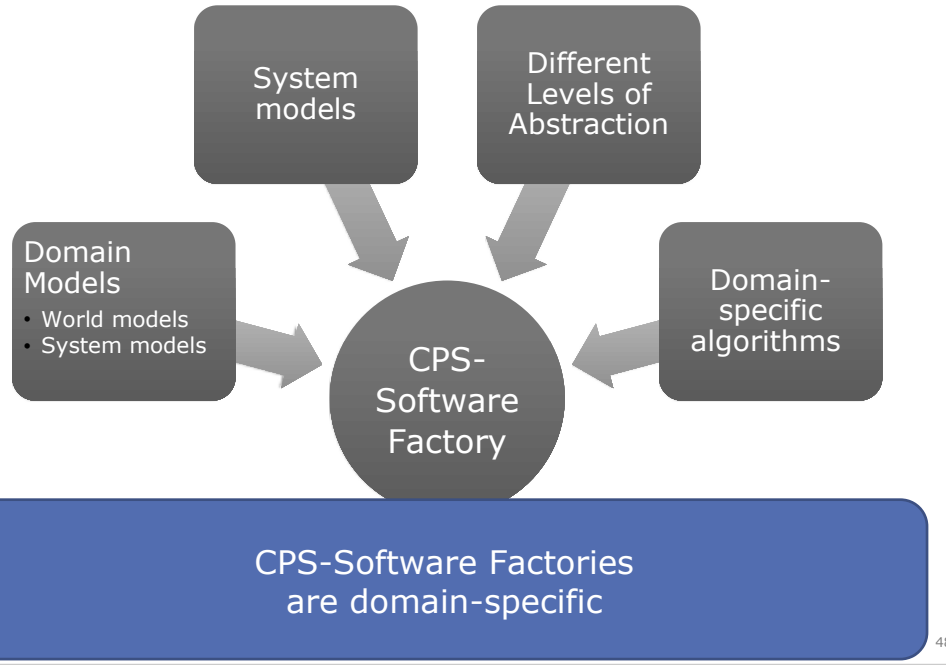
Answer: By Model-Driven Software Development (MDSD) for software **and** system, with

- Metamodels of languages (on M2)
- Models (on M1)
- Repositories (on M0)



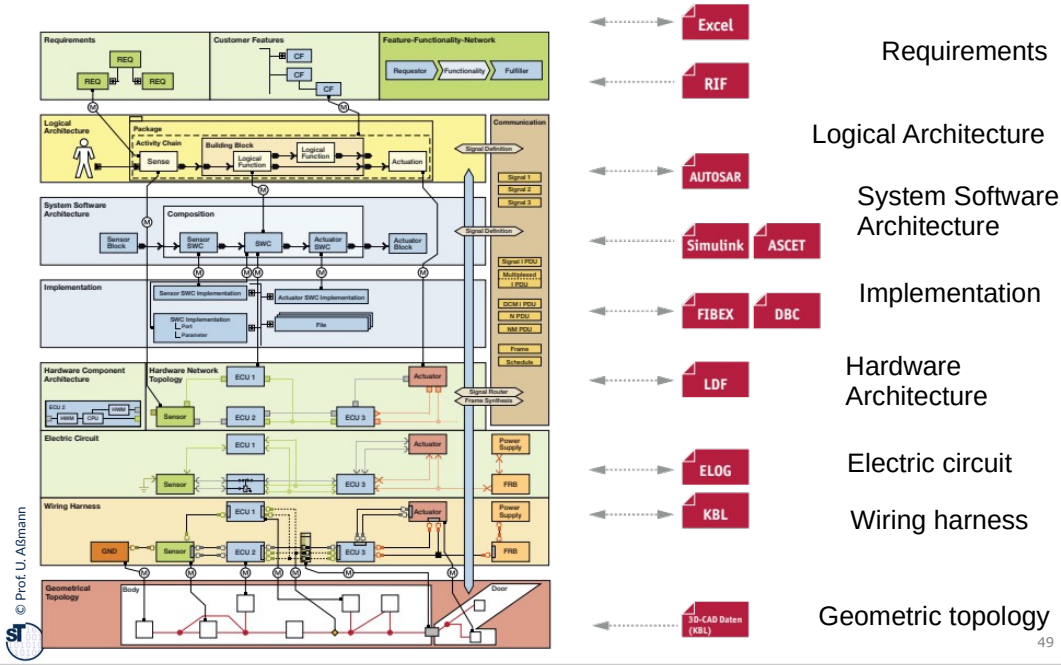
0.7#
Software ist die stärkste gesellschaftsverändernde Kraft heute

Domain-Specific CPS-Software Factories



Example: Car Design with PREEVision (Vector)

[Preevision]

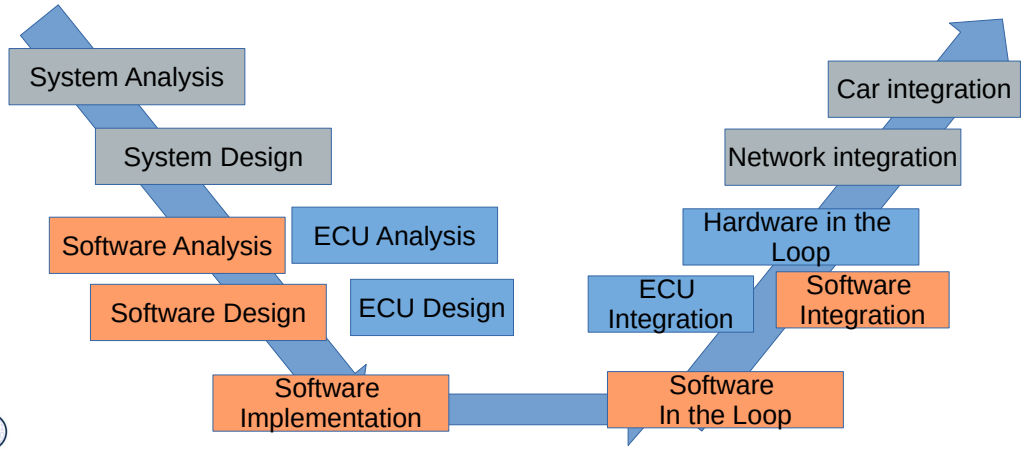


PreeVision has 3 Tools Steered by Metamodels

[MüGI09]

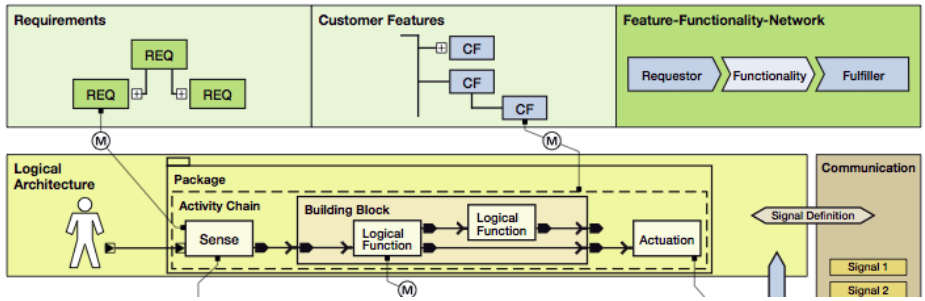
- PREEvision Architect
- PREEvision Function Designer
- PREEvision Electric Designer

- With options:
  - vTESTcenter
  - PREEvision Collaboration Platform
- All involved models are metamodeled



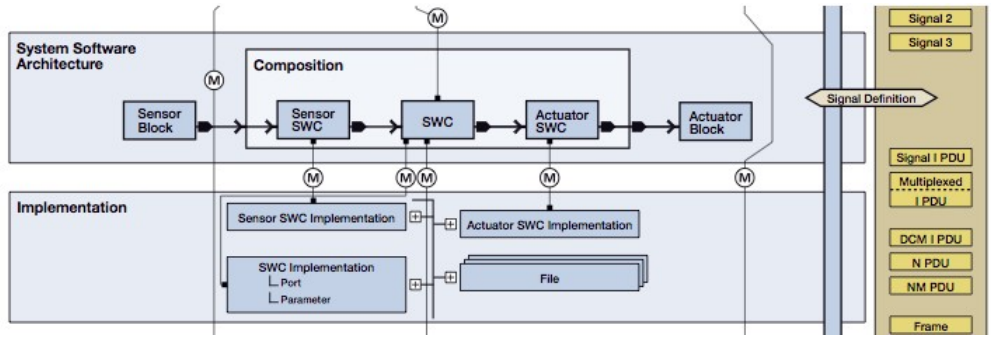
PreeVision Models in More Details

- Requirements specification with Excel and Requirements Interchange Format (RIF)
- Logical architecture with AUTOSAR components



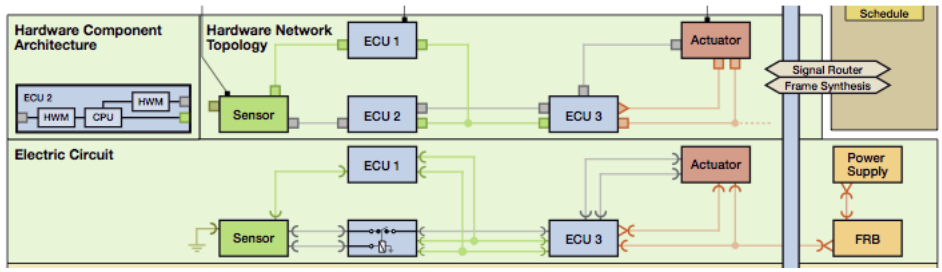
PreeVision Models in More Details

- Software Architecture with Simulink components (blocks) and ASCET model components (from ETAS)
- Implementation (generated or hand written)



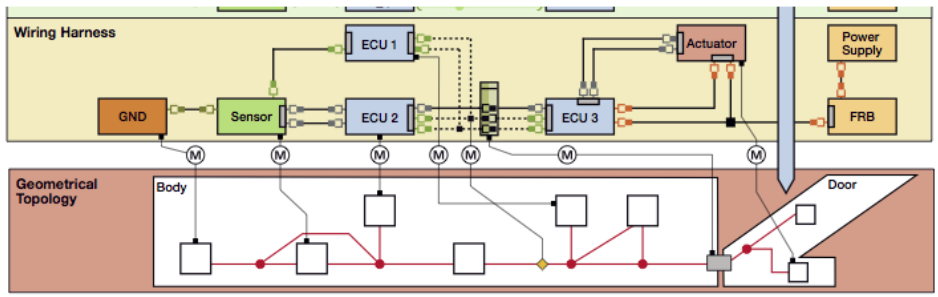
PreeVision Models in More Details

- Hardware architecture with LDF component model
- Electronic circuit design in ECU by ELOG



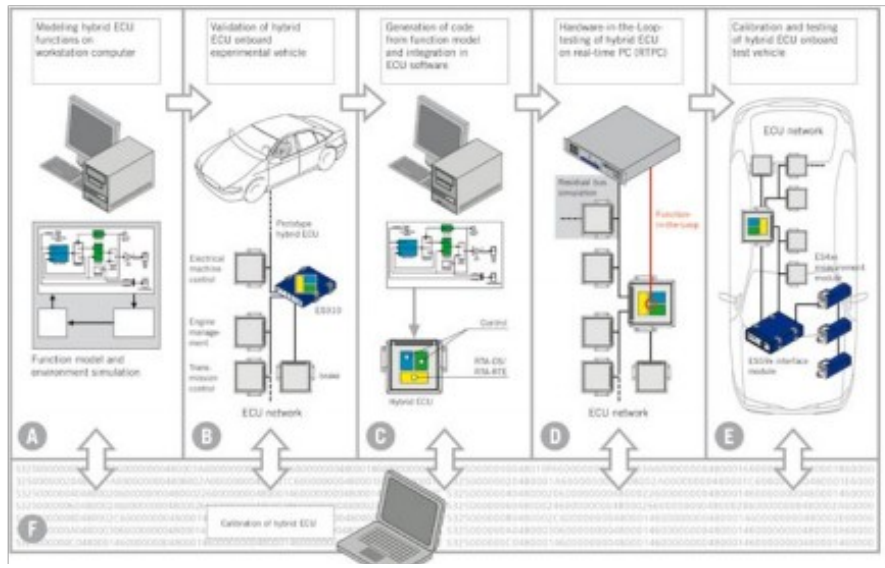
PreeVision Models in More Details

- Wiring in the car (physical network) with KBL
- 3-D CAD drawings for geometrical topology

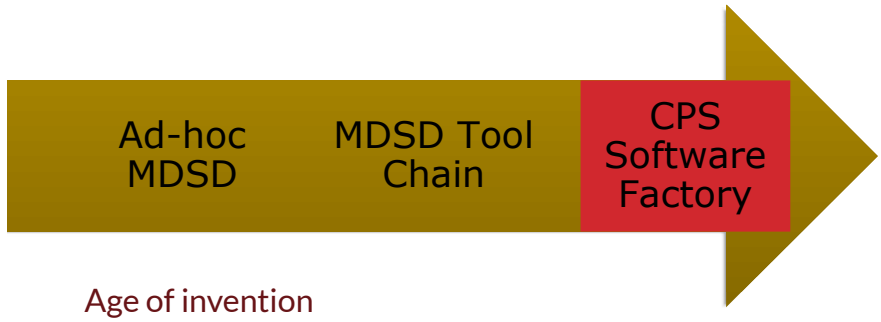


Electric Cars (ETAS)

[ETAS] http://www.etas.com/en/products/ascet_md_modeling_design.php



CPS Software Factories (CPS IDE, Design Tools, CPS Tool Chains) are a Sign of a Maturing Productivity Industry



Age of invention

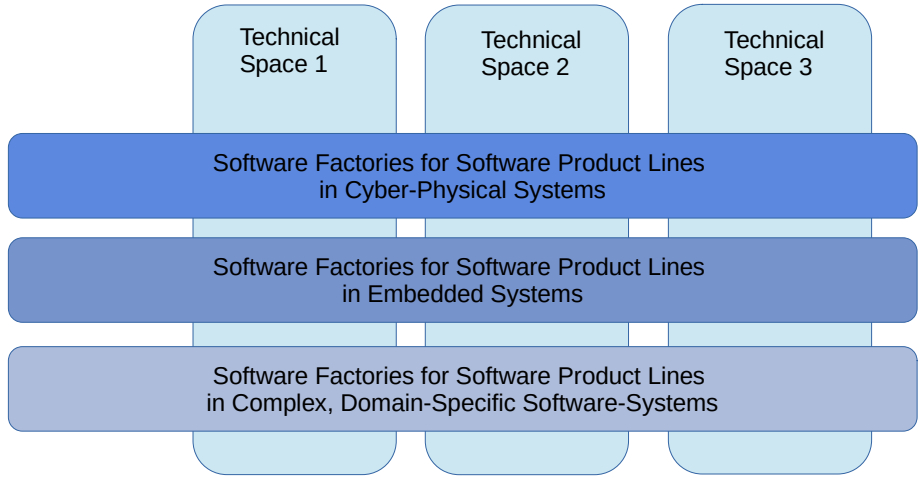
Age of implementation

Age of integration

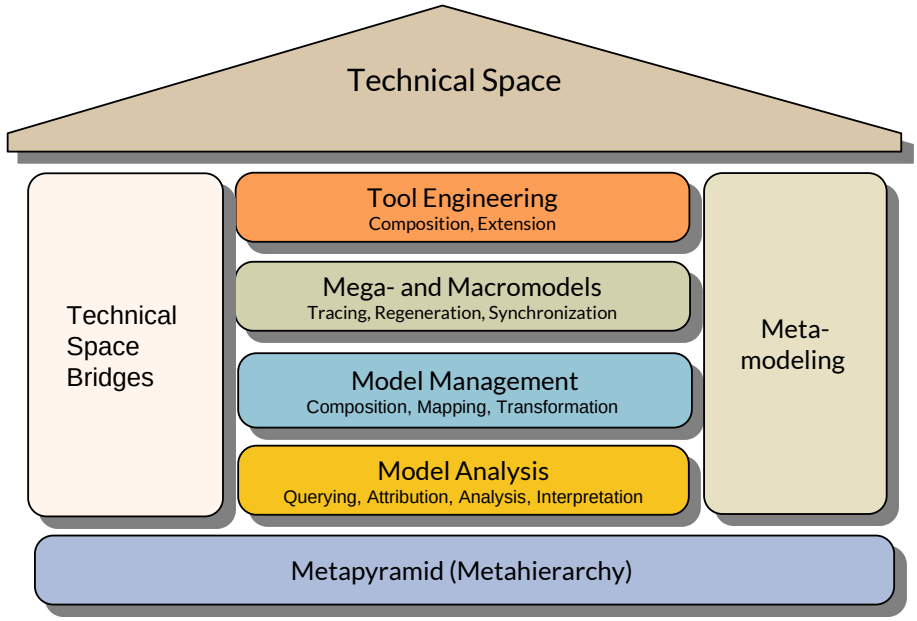
Will hold for all domains of CPS!

2.4 Why Do We Need Software Factories and MDSD in TS?

(Heterogeneous) Software Factories



Q10: The House of a Technical Space

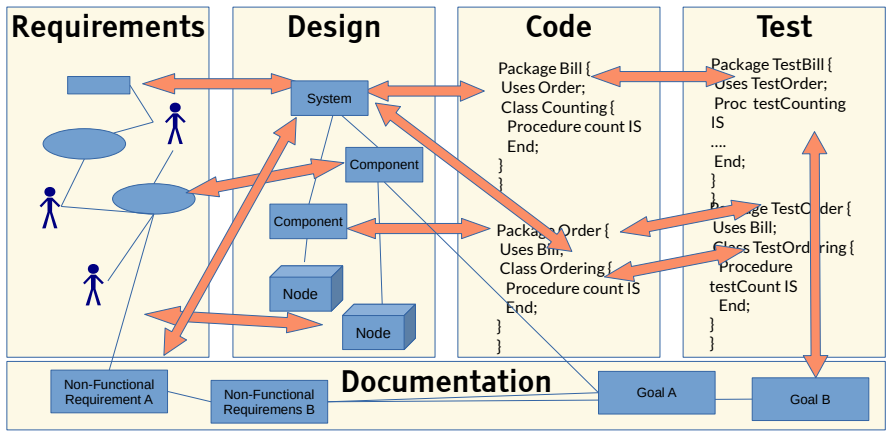


Q11: Overview of Technical Spaces in the Classical Metahierarchy

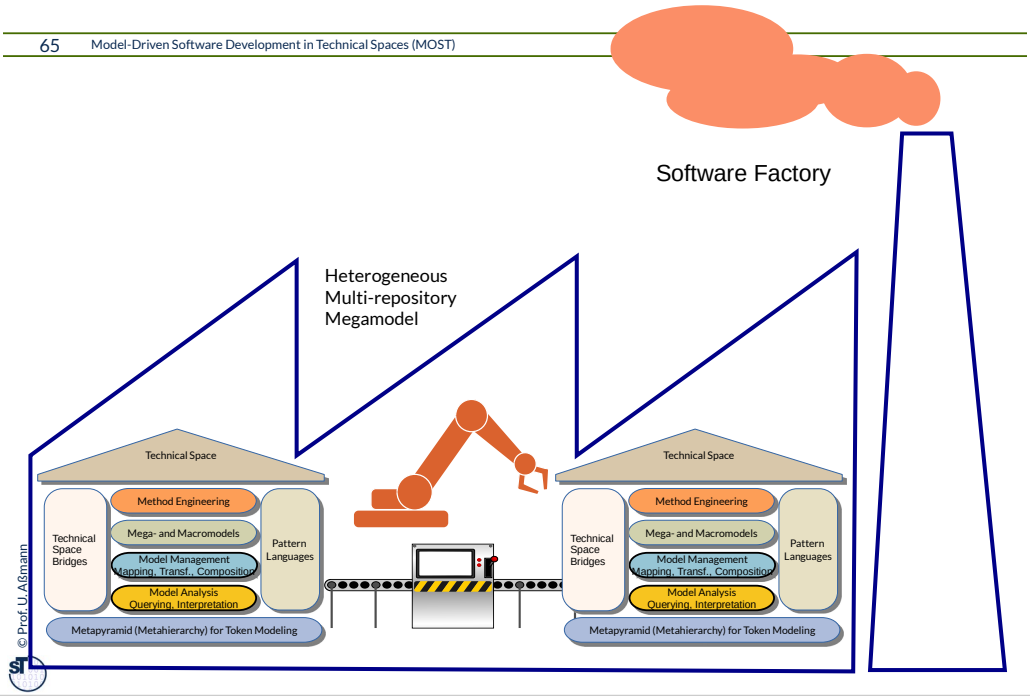
	Grammarware (Strings)	Text-ware	Table-ware		Treeware (trees)	Link-Tree-ware		Graphware/ Modelware			Role-Ware	CROM-Ware	Ontology-ware
	Strings	Text	Text-Table	Relational Algebra	NF2	XML	Link trees	MOF	Eclipse	CDIF	MetaEdit+	Context-role graphs	OWL-Ware
M3	EBNF	EBNF		CWM (common warehouse model)	NF2-language	XSD	JastAdd, Silver	MOF	Ecore, EMOF	ERD	GOPPR	CROM	RDFS OWL
M2	Grammar of a language	Grammar with line delimiters	csv-header	Relational Schema	NF2-Schema	XML Schema, e.g. xhtml	Specific RAG	UML-CD, -SC, OCL	UML, many others	CDIF-languages	UML, many others	CROM	HTML XML MOF UML DSL
M1	String, Program	Text in lines	csv Table	Relations	NF2-tree relation	XML-Documents	Link-Syntax-Trees	Classes, Programs	Classes, Programs	CDIF-Models	Classes, Programs	CROM models	Facts (T-Box)
M0	Objects	Sequences of lines	Sequences of rows	Sets of tuples	trees	dynamic semantics in browser		Object nets	Hierarchical graphs	Object nets	Object nets	Context-Object-Role Nets	A-Box (RDF-Graphs)

Q12: The ReDoDeCT Problem and its Macromodel

- The **ReDoDeCT problem** is the problem how requirements, documentation, design, code, and tests are related (→ V model)
- Mappings between the Requirements model, Documentation files, Design model, Code, Test cases
- A **ReDoDeCT macromodel** has maintained mappings between all 5 models



Q13: A Software Factory's Heart: the Multi-TS Megamodel



The End

- Why are future CPS a good application area for model-driven software development?
- Explain the model-driven tool chain Preevision, which problems about heterogeneous software systems it solves
- Why are CPS based on collaboration, contexts and roles?
- Why is modeling important for CPS?

Important World Models of “World Databases” (Monitoring CPS)

Physical Location of Thing in Environment

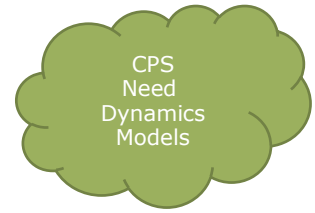
- Where is my thing in space?
  - Model of Physical Environment required
  - spatial, real-timed
  - magnetic, heat, humidity, user-defined
  - Continuous models

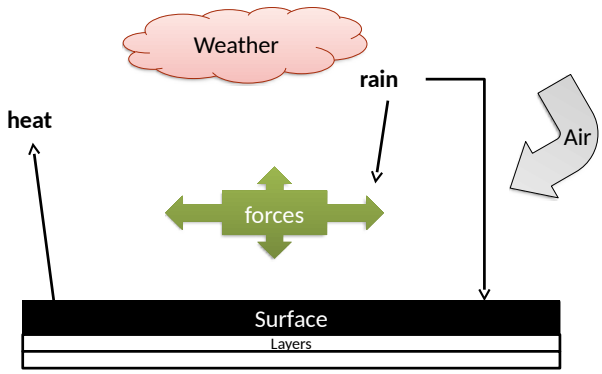
3D office models
Building models
City models
http://www.turbosquid.com

http://tf3dm.com/3d-model/the-city-39441.html

Physical Dynamics (Movement) of Thing

- How does it move in space?
  - Continuous modeling languages (Modelica)
  - Www.modelica.org, www.openmodelica.org

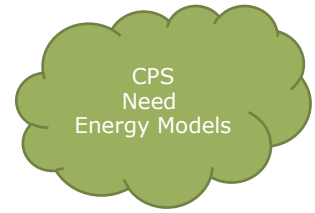


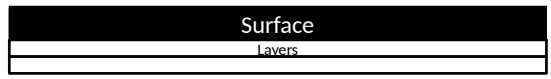


complex interplay of
- surface props
- weather: wind, rain, heat

Energy Consumption of Thing

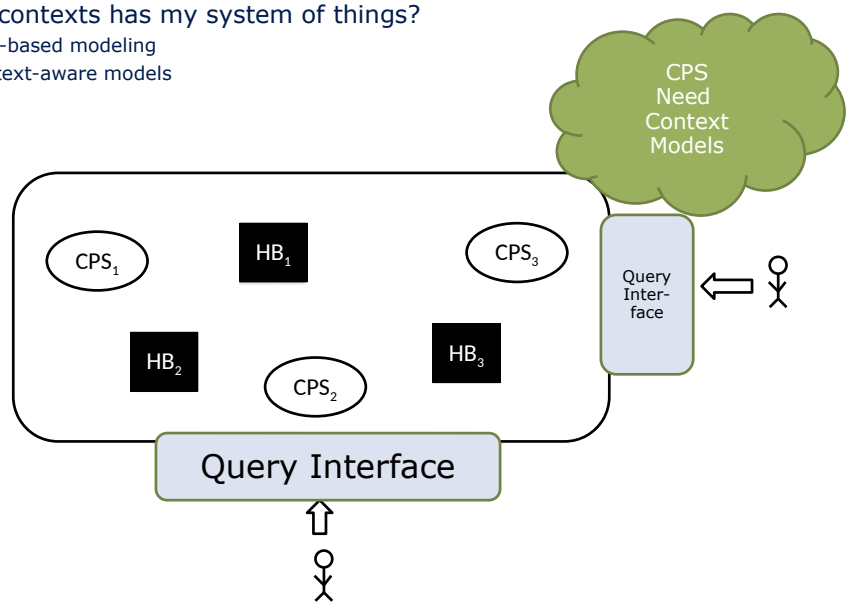
- How much energy is left for its tasks?





Current Physical Composition of a Thing

- Which contexts has my system of things?
  - Role-based modeling
  - Context-aware models



A Simple CPS: Cloud Robots

A Cloud Robot uses a Standard Robotic Platform
Hello, I'm NAO

Made by

- ALDEBARAN Robotics Paris, Frankreich
  [http://www.aldebaran-robotics.com/]

Application fields

- Teaching (Robot programming)
- Research
  - Robotics, AI
  - RoboCup
  - **Software Engineering**

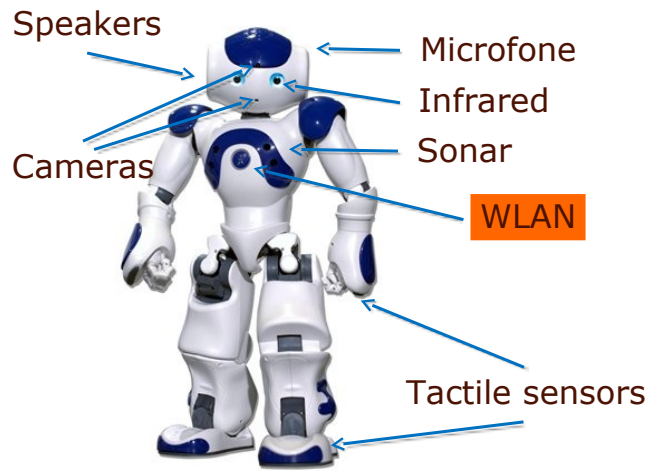
Price

- 9.000 – 12.000 €

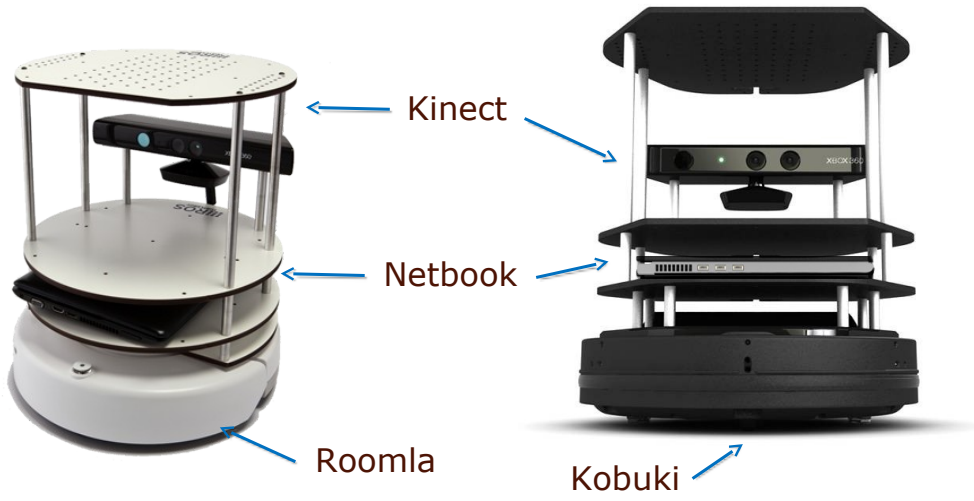
Nao Fact Sheet

Length: 58cm
Weight: 5kg
Hardware:
- x86 AMD GEODE 500MHz
- 256MB RAM
- 21 motors
- Battery 55Wh

OS:
Embedded Linux 32bit



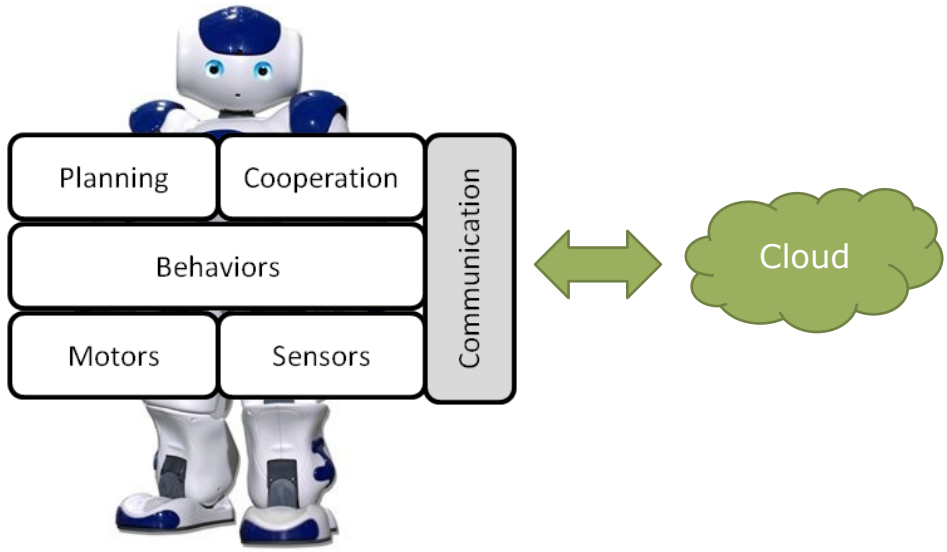
Turtle Bot



50kHz Sensor data rate

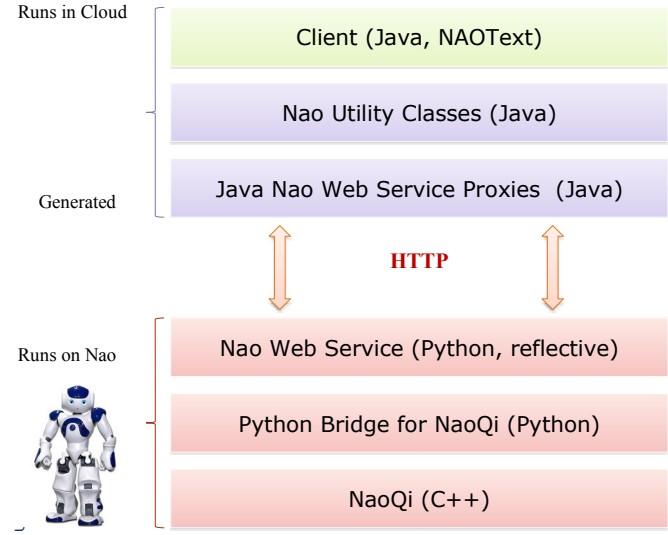
http://wiki.ros.org/Robots/TurtleBot
http://www.turtlebot.com

ResUbic Lab:
NAO Web Service Architecture



http://code.google.com/p/naoservice/

NAO Web Service and Communication Framework



https://github.com/max-leuthaeuser/naoservice

A Killer App for Cloud Robots: Donut Production in „Nachtsprung“

Donuts Should be Individual....

And

https://www.flickr.com/photos/amiga-commodore/10059167335/

The Dough Is the Same, but the Topping Makes the Difference

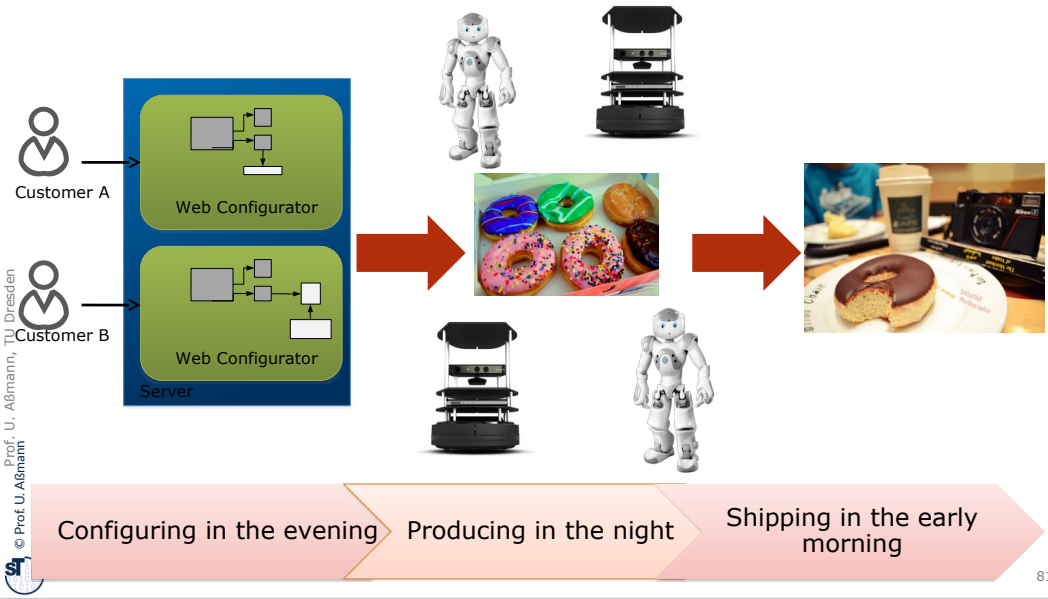
Situation Today

https://www.flickr.com/photos/jeades/2383525381/

- Mass production
- No individual configuration
- No fast, individualized production
- No „Nachtsprung“

Donut Industry-4.0: Pulling Individual Donuts out in Nachtsprung

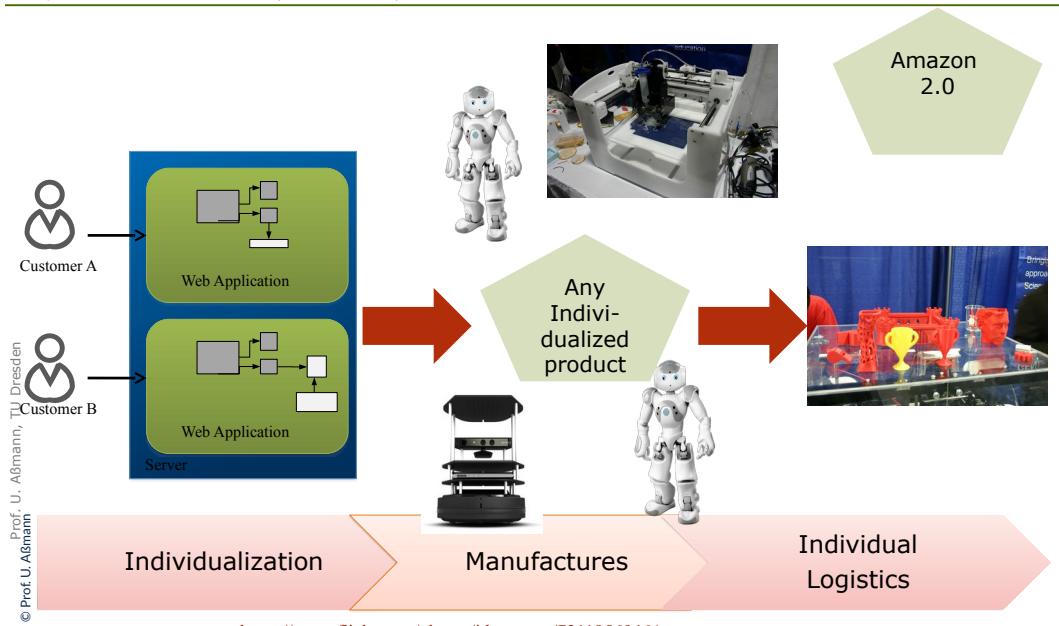
https://www.flickr.com/photos/soso__1991/7179199134/



Configuring in the evening | Producing in the night | Shipping in the early morning

3-d-printer

Industry-4.0: Economic Consequences



https://www.flickr.com/photos/ideonexus/7311856946/
https://www.flickr.com/photos/ideonexus/7311859510

3-d-printer

Industry-4.0:
Cloud Robots Produce Things in Workflows