Design-Abstraction and Processes: Examples and Experiences of Academia-Industry Collaboration

Prof. Dr. Christian Schlegel

Computer Science Department
University of Applied Sciences Ulm

http://www.zafh-servicerobotik.de/ULM/index.php
http://www.hs-ulm.de/schlegel
http://smart-robotics.sourceforge.net/

Hochschule Ulm
University of Applied Sciences

Academia-Industry Collaboration

- a two-way street where
  - both stakeholders play different roles
  - there is a third player: government / public funding
  - all bring in talent, resources, differentiated perspectives
  - overall aim: create a robust whole in addressing problems / projects

- challenges: different interests due to different roles, e.g.
  - commercialization versus public access (IPR)
  - producer (academia) / consumer (industry) relationship ?
  - market pull ? application driven ? technology driven ?
  - market driven research versus pre-competitive research ?
  - academia or industry in the driver's seat ?
  - ...

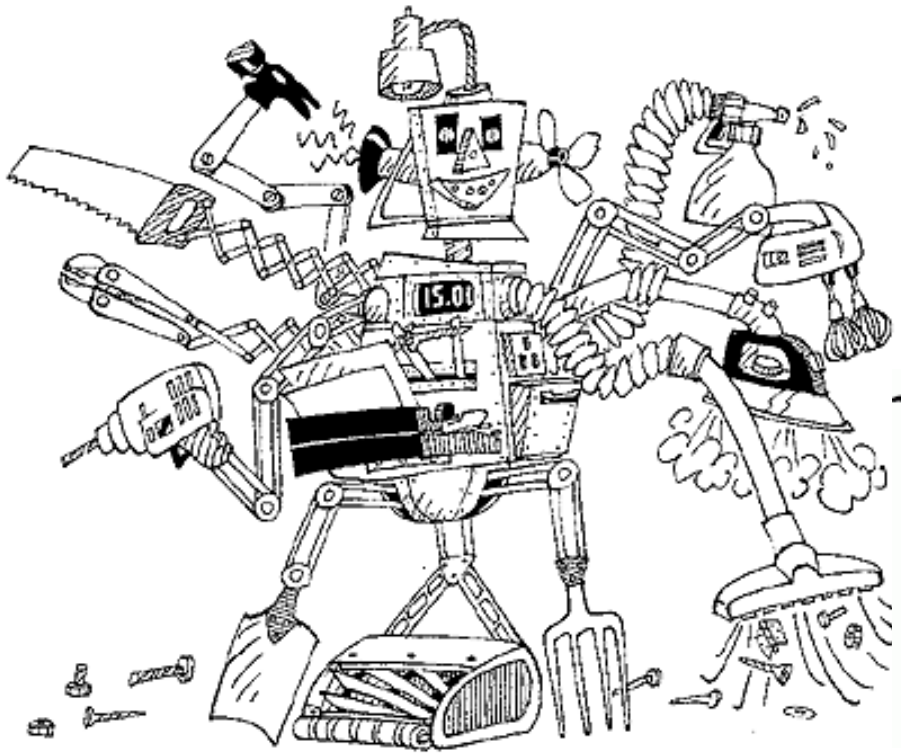
Hochschule Ulm

Academia-Industry Collaboration

- robotics
  - is a cross-cutting discipline / interdisciplinary challenge
- experience
  - it is far too easy to say “details can be worked out”
  - it often is comfortable to focus on one aspect in isolation
- interdisciplinary / integrational challenges are often neglected
  - system architectures, system engineering tools
  - safety, security, diagnostics and monitoring
  - resource awareness, quality of service
  - ➔ *system engineering is challenging for industry and academia*
  - ➔ *industry has experience, academia extends methods*
  - ➔ *it requires industrial expertise and offers scientific challenges*
  - ➔ *often perceived as “doing” or “just implementation” but that is wrong*
  - ➔ *ignoring system level challenges looks like a gap between industrial needs and academic offerings*

Approach: Design Abstraction

A development process often applied in robotics ...

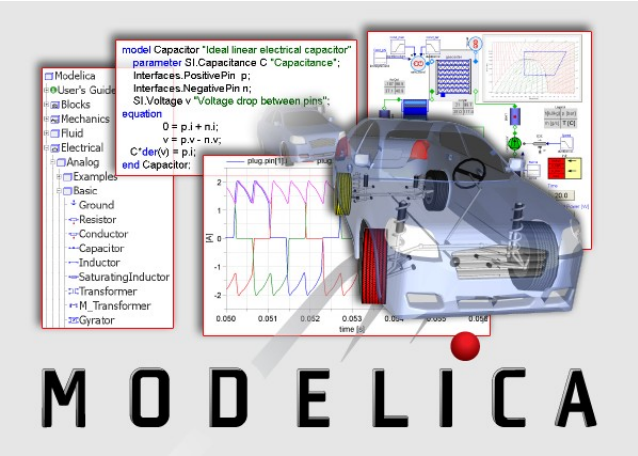


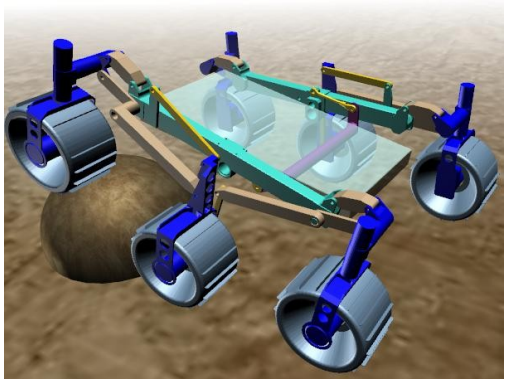
Hochschule Ulm

Approach: Design Abstraction

Systematic engineering processes look different ...

... also in case of software intensive systems





UML MARTE

Hochschule Ulm

Approach: Design Abstraction

We need a systematic engineering approach for robotics software!

- robots are complex systems that depend on systematic engineering
- so far fundamental properties of robotic systems have not been made detailed enough nor explicit (e.g. QoS)
- tremendous code-bases (libraries, middleware, etc.) coexist without any chance of interoperability and each tool has attributes that favors its use

➔ ***rely, as for every engineering endeavour, on the power of models***
➔ ***nowadays, robotics functionality is foremost based on software***
➔ ***make the step towards MDSD***







etc.

Hochschule Ulm

Approach: Design Abstraction

http://www.willowgarage.com/blog

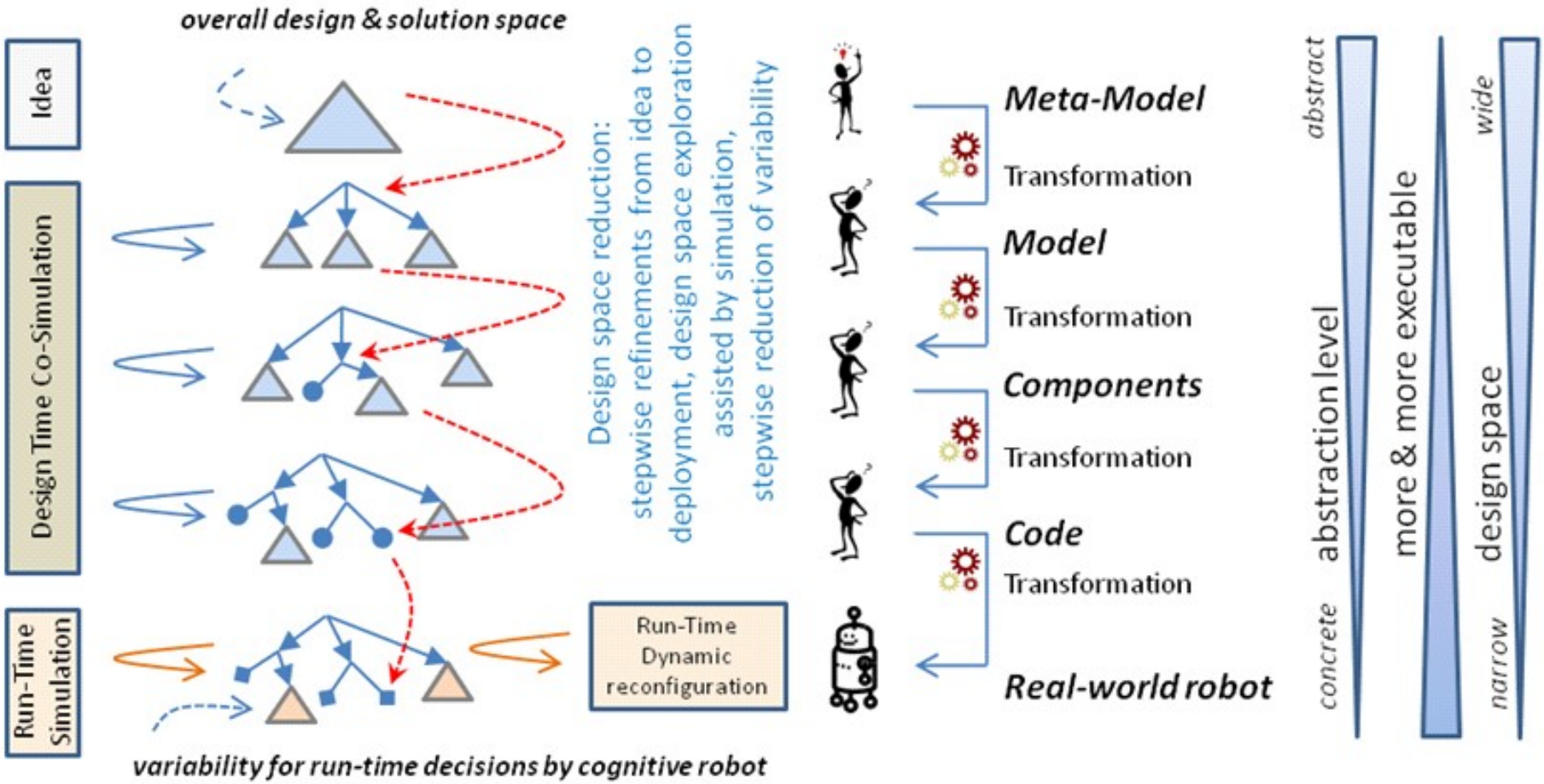
- separation of robotics knowledge from short-cycled implementational technologies
- providing sophisticated and optimized software structures to robotics developers not requiring them to become a software expert

how to achieve this?

- *make the step from code-driven to model-driven designs*
- since recently, there are matured open source tools, etc. available that can be applied in / tailored to robotics!

Hochschule Ulm

Approach: Design Abstraction

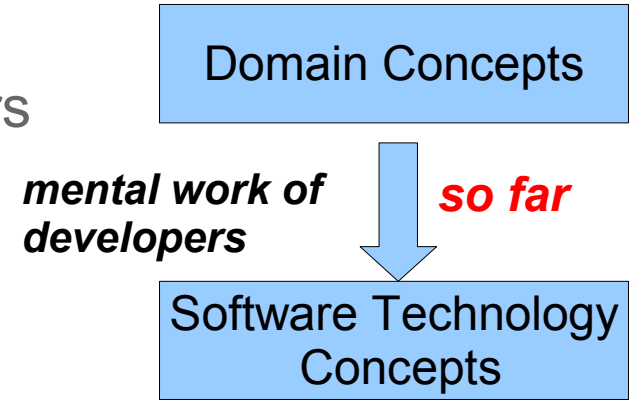


Hochschule Ulm

Approach: Design Abstraction

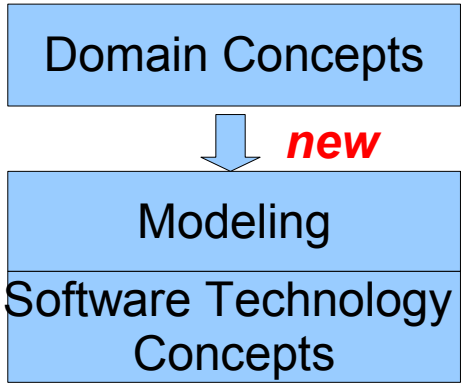
What is different in robotics?

- ***not*** the huge number of different sensors and actuators
- ***not*** the various hardware platforms
- ***not*** real-time requirements etc.



but

- the context and situation dependant reconfiguration of interactions
- a prioritized assignment of restricted resources to activities again depending on context and situation
- the tremendous complexity of design space / solution space
- deviations between design-time and run-time optimality



- **analysis / requirements models are *non-computational*, MDSD models are *computational***
- ***MDSD models are no "paperwork", they are the solution which is translated into code automatically***

Hochschule Ulm

Example 1:
Collaborative Center for Applied Research

- ZAFH Servicerobotics
- http://www.zafh-servicerobotik.de/
- collaborative center for applied research on service robotics

servicerobotik ZAFH

RWB EFRE

2008-2010 / 1.5 M€
(optional: 2011-2012 / 1 M€)

Universities of Applied Sciences
- Ulm
- Ravensburg-Weingarten
- Mannheim

- *adaptive intelligent control*
- *verification of safety properties*
- *software technology*
- *localization / mapping*
- *information-optimized object recognition*
- *adaptive real-time image processing*

Hochschule Ulm

Example 1:
Collaborative Center for Applied Research

- ZAFH Servicerobotics
  - further strengthening collaboration of
    - universities of applied sciences with
    - SMEs
  - in particular, SMEs should take the opportunity of improving their competitiveness by applying research results

- no funding of industry / SMEs in this program
  - Landesstiftung Baden-Württemberg
    (foundation to support projects of general public benefit linked by the common aim of securing the future capabilities of the State of Baden-Württemberg)
  - co-funded by EFRE (European Regional Development Fund)

RWB EFRE

service robotik ZAFH

Hochschule Ulm

Example 1:
Collaborative Center for Applied Research

service robotik ZAFH

- **Expectations**
  - sustainability of efforts
  - do not start from scratch again and again
  - composition out of building blocks
  - allow to focus on USPs
- Meta-Models and Models as means to interact with industry
  - *collaborate* at system level, interfaces, design methodologies, reference / prototype implementations of tools / components based on Meta-Models and Models *(open access)*
  - *compete* at the level of implementations (refinements of models), underneath algorithms, tailored tools *(proprietary)*

inovel innovative elektronik...

ifm

InMach Intelligente Maschinen GmbH

AGCO

Hochschule Ulm

Example 2:
Bilateral Academia-Industry Collaboration

Software development is too *complex* and too *expensive* because:

- there is *too little reuse*
- *technology changes* faster than developers can learn
- knowledge and practices are *hardly captured explicitly* and made available for reuse
- domain experts cannot understand all the *technology stuff* involved in software development

Overall aim:

- get rid of hand-crafted single unit service robot systems
- compose them out of standard components with explicitly stated properties
- be able to reuse / modify solutions expressed at a model level
- take advantage from the knowledge of software engineers that is encoded in the code transformators
- be able to verify properties (or at least provide conformance checks)

Lessons learned:

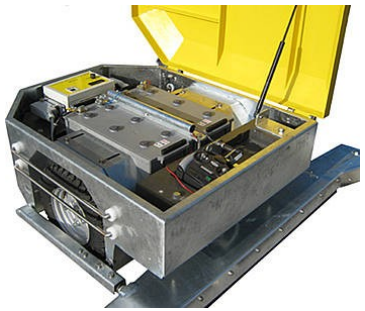
- *again, Meta-Models and Models as means to interact and share knowledge between industry and academia without requiring to use the same implementations while still sharing the same overall system architectures etc.!*

Hochschule Ulm

Example 3: SME

- they know about their domain and they are unmatched experts in their domain
- need to make the next step of automation of their products in their domain
  - next step of automation is being expected as evolution
  - they do not want to become a robotics expert
  - however, this step often is a revolution
    - high risks
    - not clear when additional sensors etc. pay off due to multi-use
- ***Example: Dung Removal Systems***
  - jump onto a design process which allows to compose robotics components from third-parties and just provide (even sell) those specialized components to others that comprise special domain / application know-how
  - ***again, basis for this is a model-based software development process***






<just arbitrary examples of dung removal systems / not the ones we are involved in>

Hochschule Ulm

Design Abstraction and Processes

- provide engineering methodology for robotics
  - **bridging the gap between academia & industry**
  - **decouple system knowledge from implementational technologies**
- provide the ground for processes for the overall lifecycle
  - **development**
  - **simulation**
  - **testing**
  - **deployment**
  - **maintenance**
- collaborate at the meta-models, transformations etc. and compete at their implementation
- provide freedom to choose whatever implementational technology is most adequate in target domain
- provides the ground for a component market: models, interoperability, services
  [in particular, SMEs: value-adding expertise can be made available by custom-models]

Hochschule Ulm

Design Abstraction and Processes

- **Design Abstraction**
  - integration / system engineering as science
  - interdisciplinary collaboration
  - training and education, workshops, summer schools
- **SMEs**
  - simpler involvement
  - small bilateral projects attached to large scale projects
  - roof under which even bilateral and short running activities are possible
- **Robotics Test Sites**
  - equipped with state-of-the-art robotic systems and equipment
  - maintained by technicians and operators
  - simulators / remote access for experiment preparation
  - places to collaborate and exchange components
  - accessible by different stakeholders
- **in general**
  - depreciation problem
  - sustainability after project duration (maintenance, expandable repositories)
- **shared design methodology / open reference implementations**
  - competition at the level of tools, models etc.

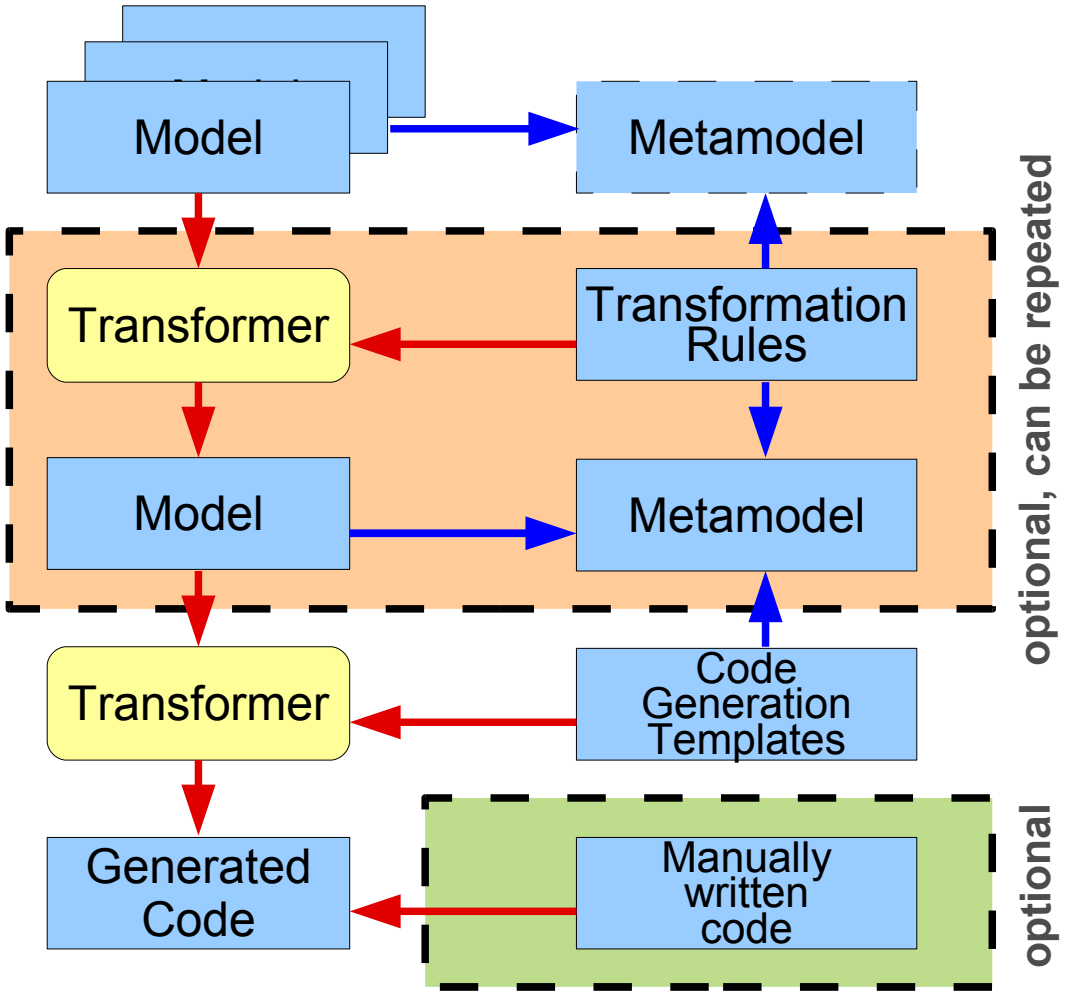
Hochschule Ulm

Addendum

Model Driven Software Development
Introduction and Motivation

How MDSD works

- Developer develops model(s) based on certain metamodel(s), expressed using a DSL
- Using code generation templates, the model is transformed to executable code
  - alternative: interpretation
- Optionally, the generated code is merged with manually written code
- One or more model-to-model transformation steps may precede code generation



http://www.voelter.de/services/mdsd.html

Model Driven Software Development
SmartMDSD

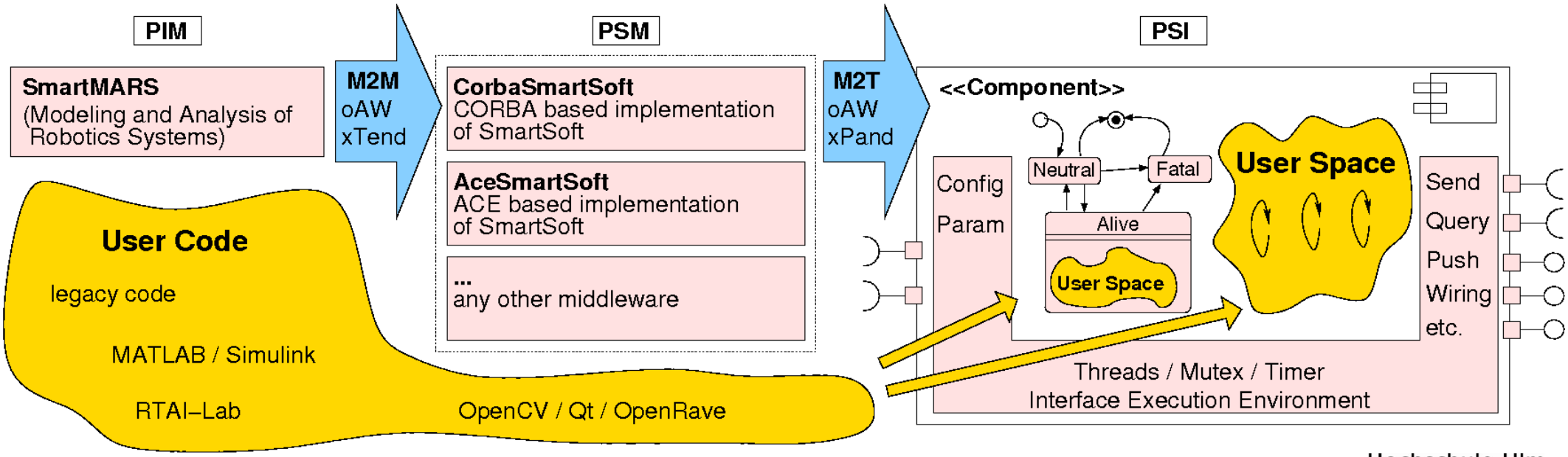
Benefits of this development process:

- systematically handle integration of systems of the complexity of service robots and to overcome plumbing
- tools like OpenArchitectureWare, Eclipse etc. are matured enough to be used in robotics
- there are many experienced people out there being already familiar with the tools, can start immediately using them and can just focus on robotics
- design patterns, best practices, approved solutions can be made available within the code generators such that even novices can immediately take advantage from that coded and immense experience
- SmartSoft provides the perfect granularity for system design, component development, composability, freedom within components, tool support etc.

Model Driven Software Development SmartMDSD

Illustration of our development process

- UML 2.0 profile for robotics component model
- covers component development, system composition, deployment
- based on standards: UML 2.0, Open Architecture Ware, Eclipse, etc.
- different runtime platforms, middleware systems etc.



Hochschule Ulm

Model Driven Software Development SmartMDSD

PIM

SmartMARS – Metamodel

(**M**odeling and **A**nalysis of **R**obotics **S**ystems)

- UML2-Profile
- platform independent stereotypes
  - SmartComponent
  - SmartTask
  - SmartMutex
  - SmartQueryServer
  - SmartEventClient
  - ...

M2M
oAW
xTend
check

PSM

CorbaSmartSoft
CORBA based implementation of SmartSoft

AceSmartSoft
ACE based implementation of SmartSoft

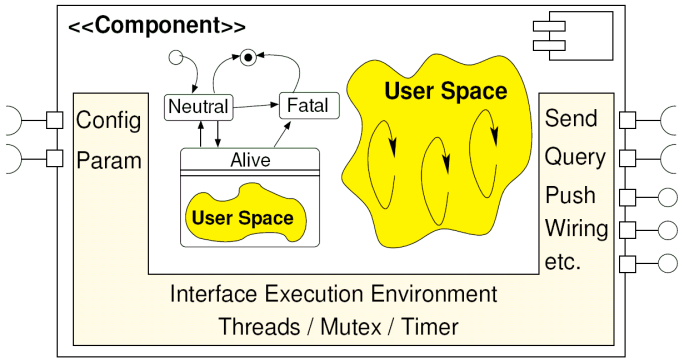
...
any other middleware

- UML2-Profile
- platform specific stereotypes

has to be created by a middleware expert

M2T
oAW
xPand
check

PSI



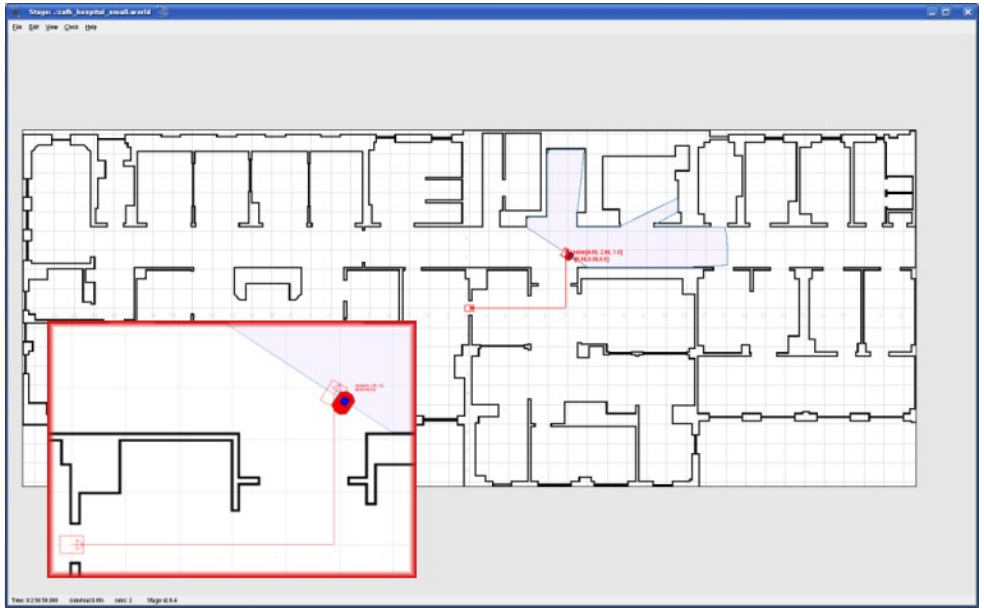
The user space can contain arbitrary code and libraries

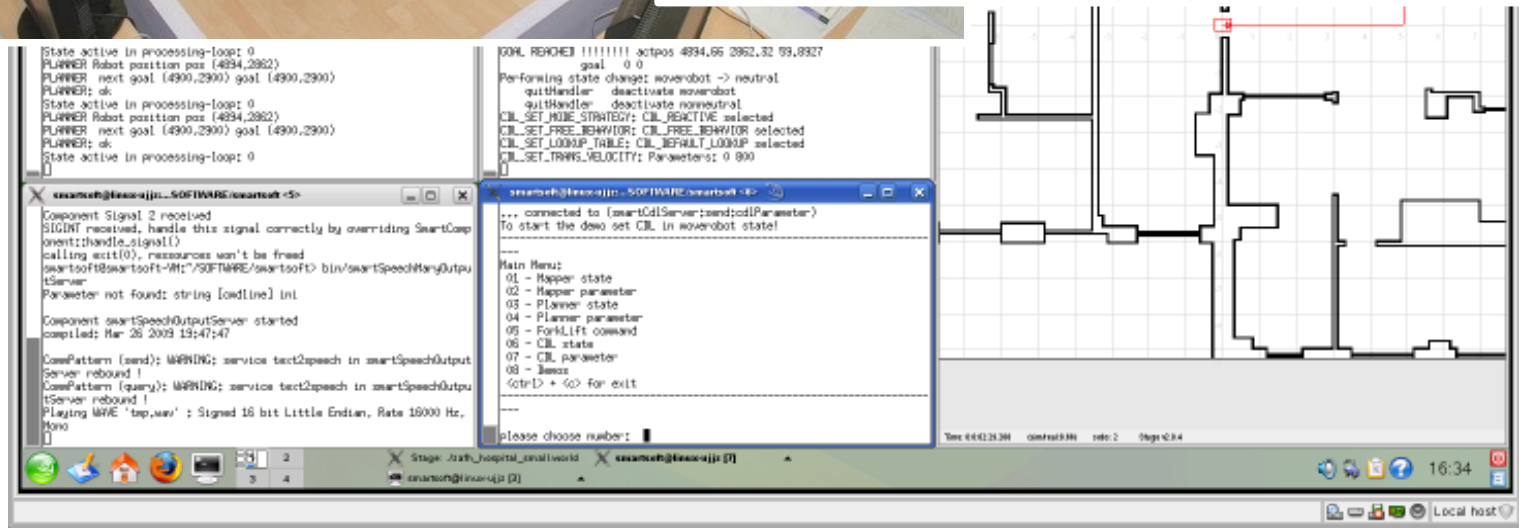
The user space remains the same independent of the different platform specific models

Just the component hull will be created

Hochschule Ulm

Model Driven Software Development
Examples etc.





Hochschule Ulm