Robotic Software Systems: From Code-Driven to Model-Driven Designs

Christian Schlegel, Thomas Haßler, Alex Lotz and Andreas Steck

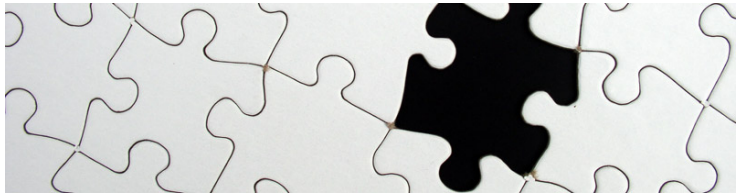
Computer Science Department
University of Applied Sciences Ulm

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

Hochschule Ulm

Model Driven Software Development
Introduction and Motivation

What is this talk about ?



- not just another software framework
- not just another middleware wrapper

➔ we have plenty of those ...

But

- 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
- using common open source tools for robotics !

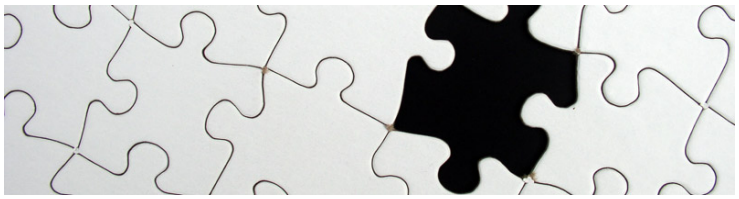
service robotik

Model Driven Software Development
Introduction and Motivation

Why is Model Driven Software Development important in Robotics ?

- 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 transformation rules / hidden structures
- be able to verify (or at least provide conformance checks) properties

and many many more good reasons



Engineering the software development process in robotics is one of the basic necessities towards industrial-strength service robotic systems

Model Driven Software Development
Idea and Approach

That sounds good but give me an example ...

we made some very simple but pivotal decisions:

- granularity level for system composition:
  - loosely coupled components
  - services provided and required
- strictly enforced interaction patterns between components
  - precisely defined semantics of intercomponent interaction
  - these are policies (and can be mapped onto any middleware mechanism)

  ➔ *independent of a certain middleware*
- minimum component model to support system integration
  - dynamic wiring of the data flow between components
  - state automaton to allow for orchestration / configuration

  ➔ *ensures composability / system integration*
- execution environment independently
  - tasks (periodic, non-periodic, hard real-time, no realtime), synchronization, resource access

  ➔ *again, can be mapped onto different operating systems*

Model Driven Software Development
Idea and Approach

SmartSoft can be seen as:

- the idea
  - how robotics systems should be composed out of components
  - how the components hull looks like
  - how the components interact with each other
- the concrete implementations based on
  - CORBA => CorbaSmartSoft
  - ACE only
  - ...

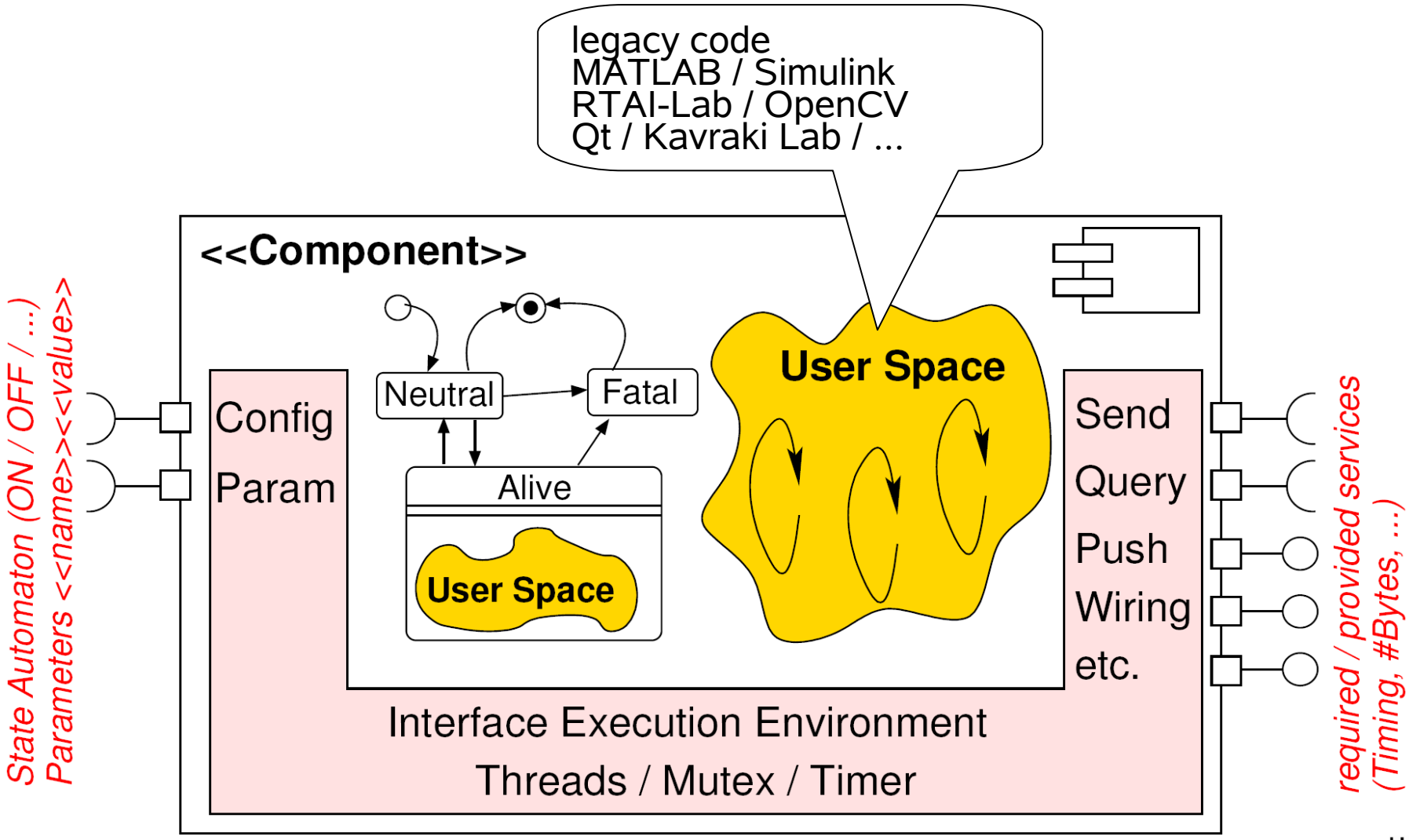
These patterns are sufficient since they offer request/response interaction as well as asynchronous notifications and push services.

The SmartSoft Interaction Patterns

send	one-way communication
query	two-way request/response
push newest	1-to-n distribution
push timed	1-to-n distribution
event	asynchronous conditioned notification
wiring	dynamic component wiring

service robotik

Model Driven Software Development
Idea and Approach



Hochschule Ulm

Model Driven Software Development
The Workflow

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

PSM

CorbaSmartSoft
CORBA based implementation of SmartSoft

AceSmartSoft
ACE based implementation of SmartSoft

Microsoft Robotic Studio
MSRS based implementation

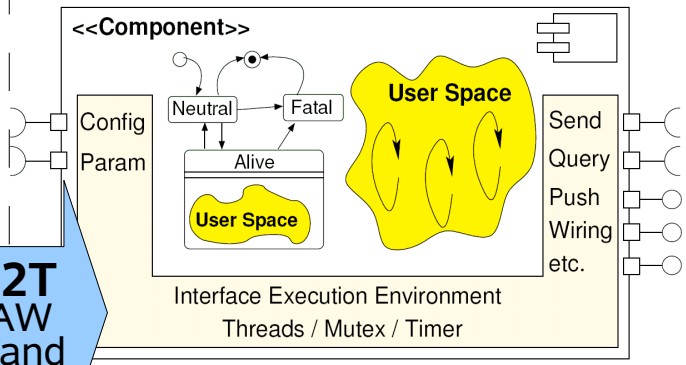
...
any other middleware

- UML2-Profile
- platform specific stereotypes

has to be created by a middleware expert

M2T
oAW
xPand

PSI



The User Space can contain arbitrary code and libraries

The User Space stays the same independent of the different platform specific models

Just the component hull will be created

Hochschule Ulm

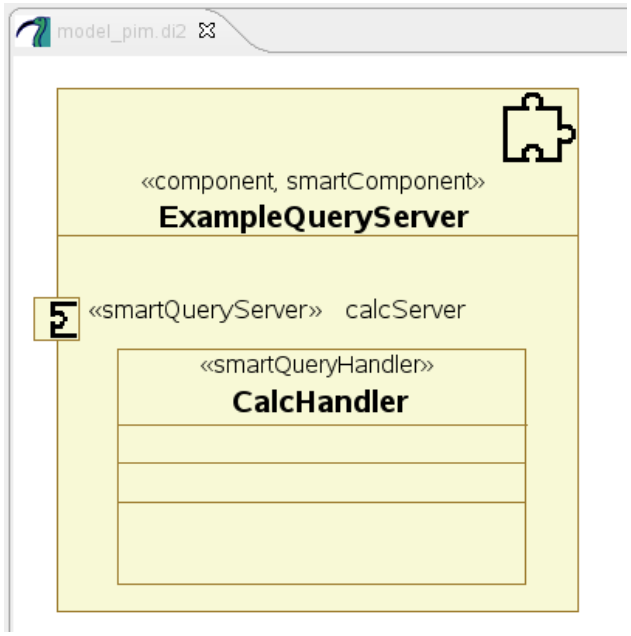
Model Driven Software Development Workflow Example (User View)

service robotik

PIM

SmartMARS – Metamodel

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

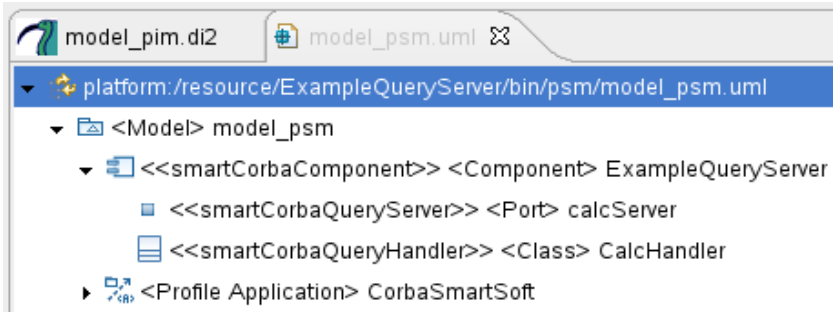


user has to create a PIM

M2M
oAW
xTend

PSM

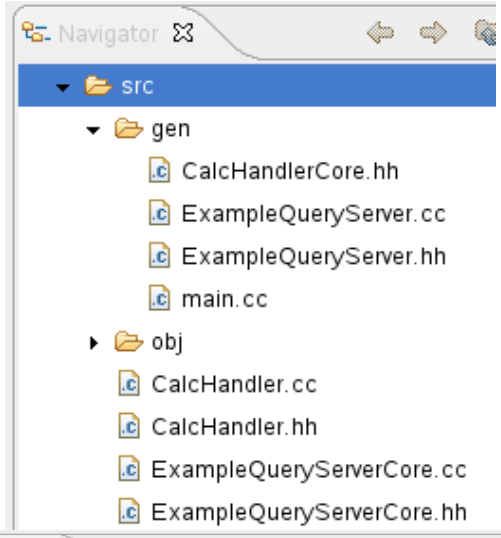
CorbaSmartSoft
CORBA based implementation of SmartSoft

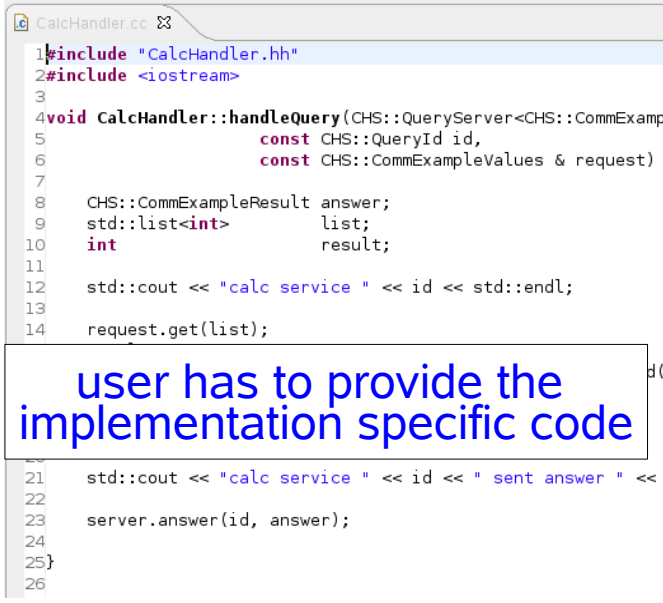


no need to change anything in the PSM

M2T
oAW
xPand

PSI

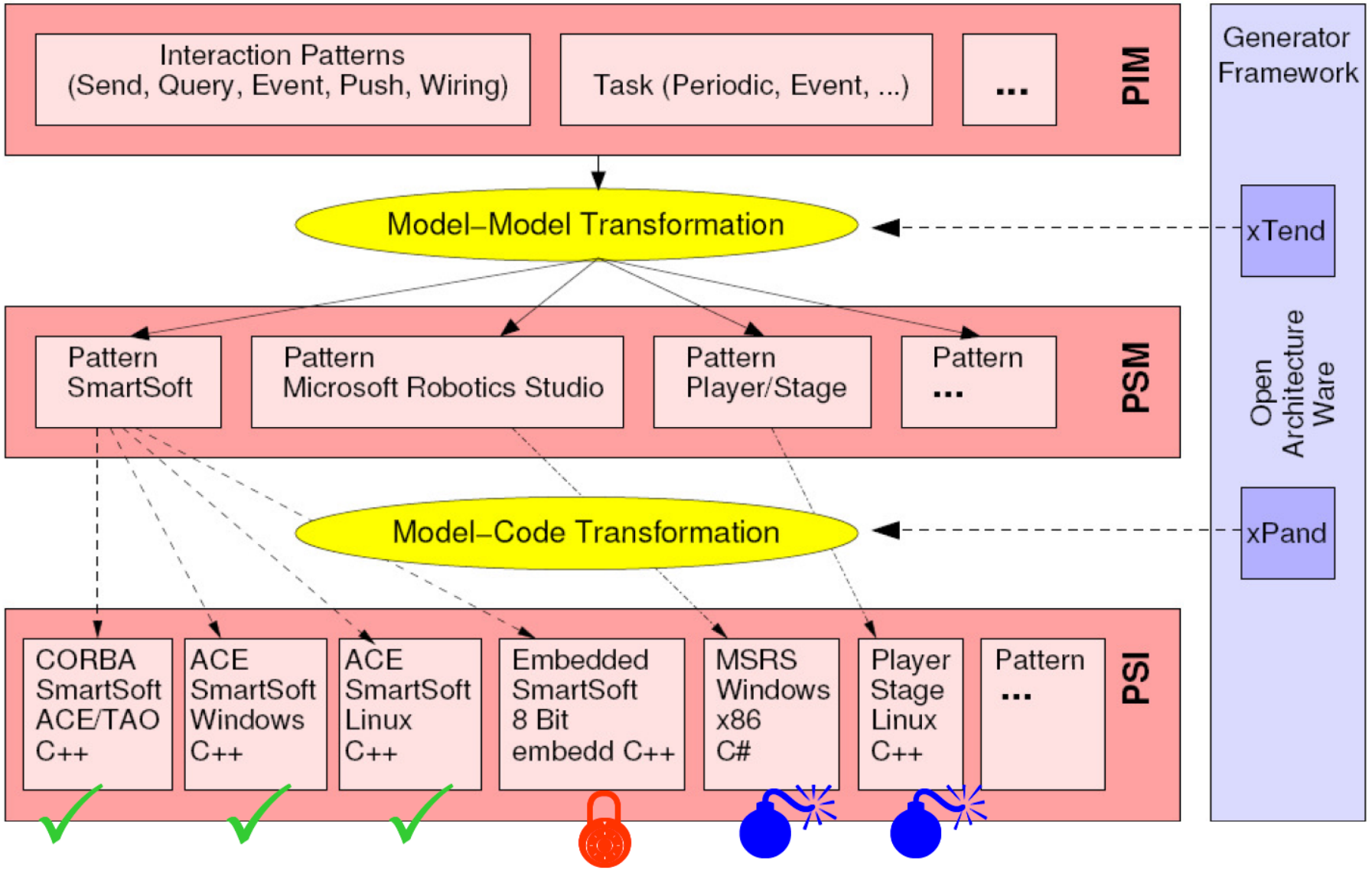




user has to provide the implementation specific code

service robotik

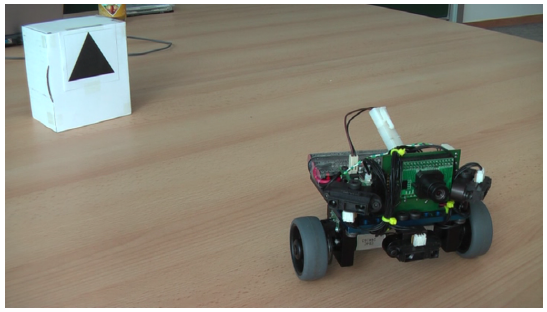
Model Driven Software Development Transformation



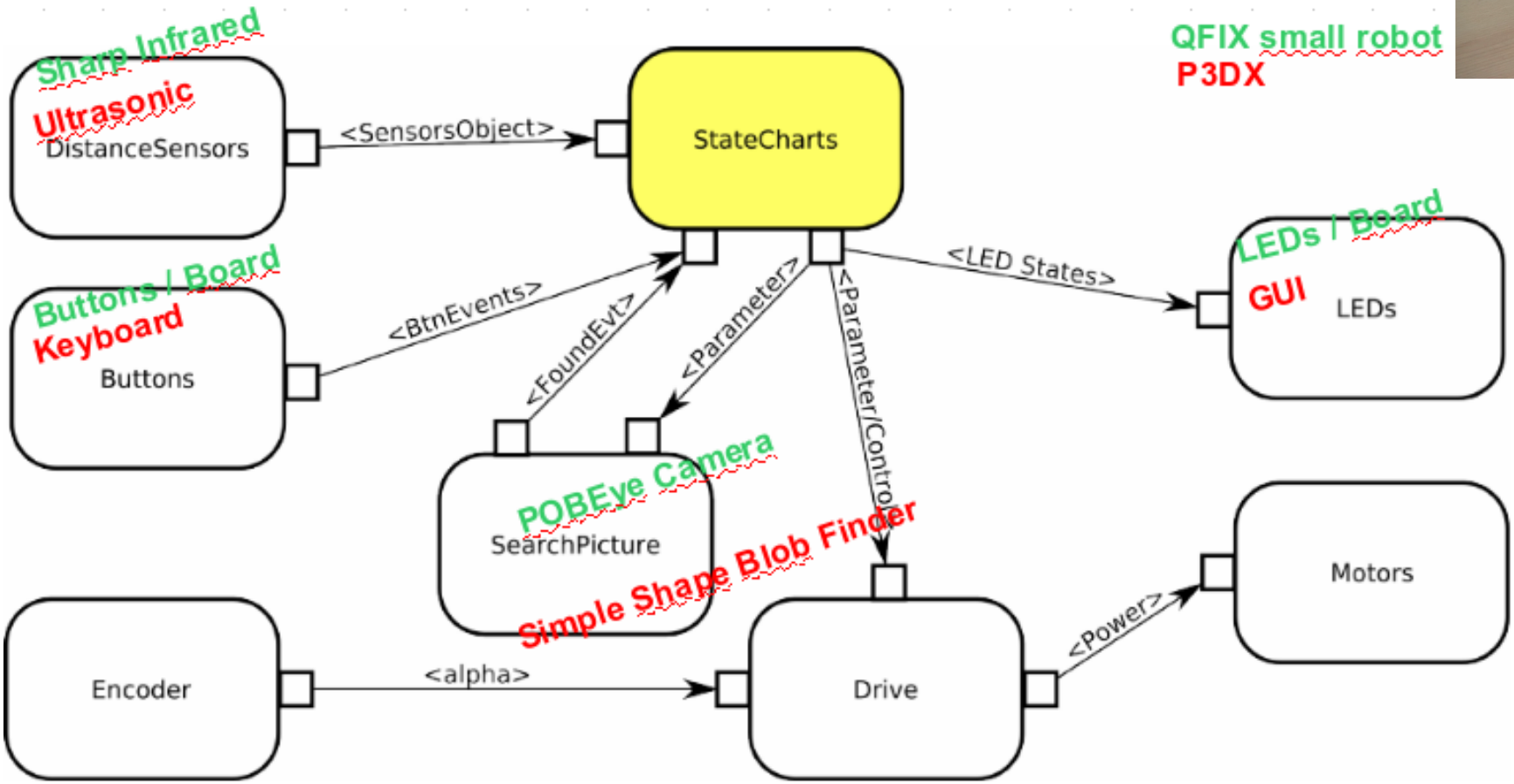
Hochschule Ulm

service robotik

Model Driven Software Development
Practical Example



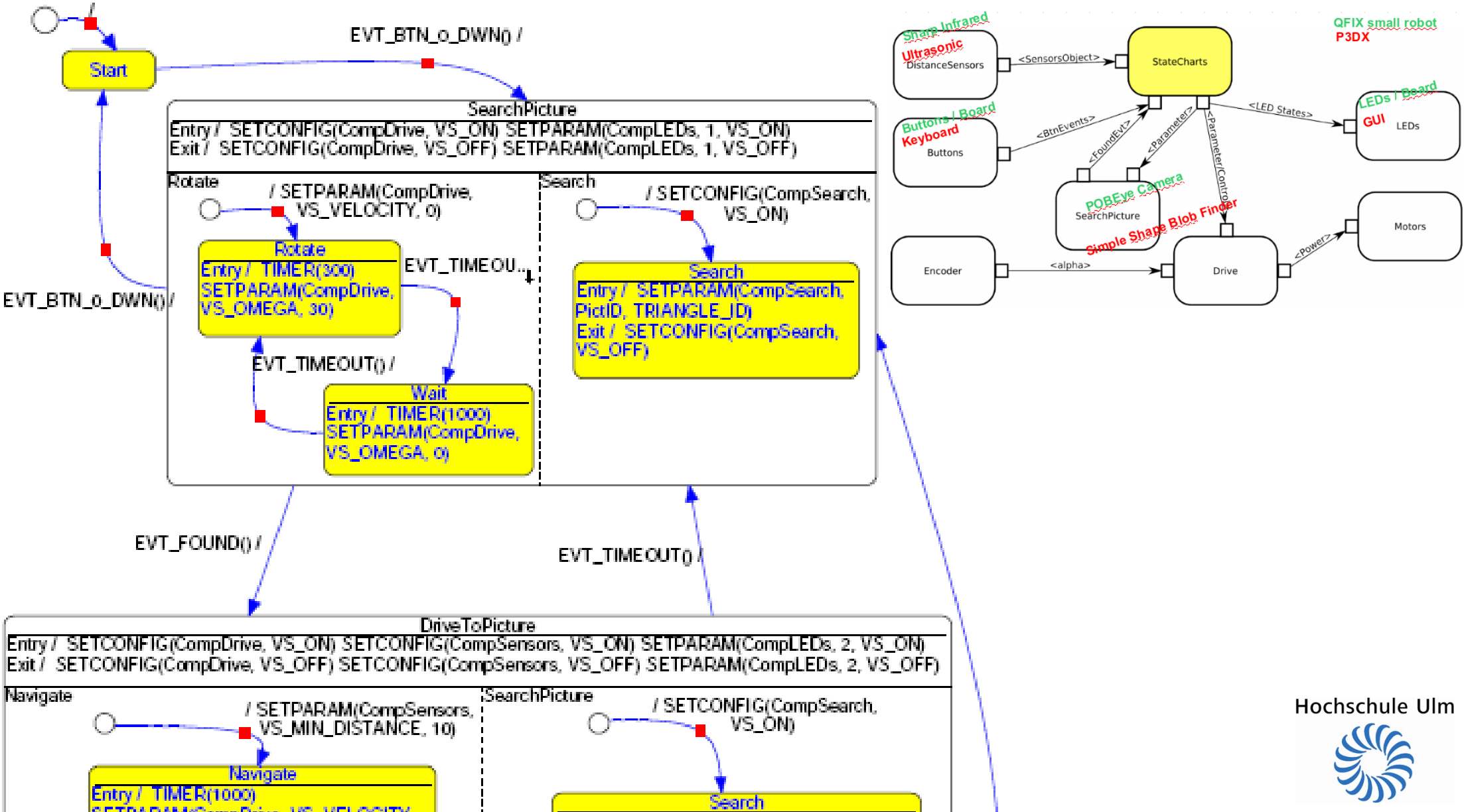
QFIX small robot
P3DX



Hochschule Ulm

service robotik

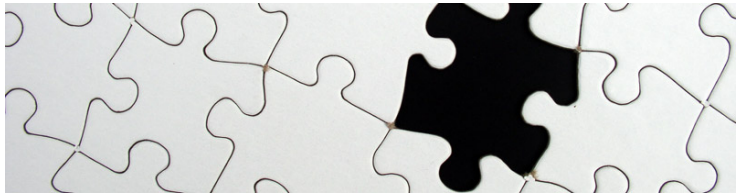
Model Driven Software Development
Practical Example



Hochschule Ulm

Model Driven Software Development
Summary and Conclusion

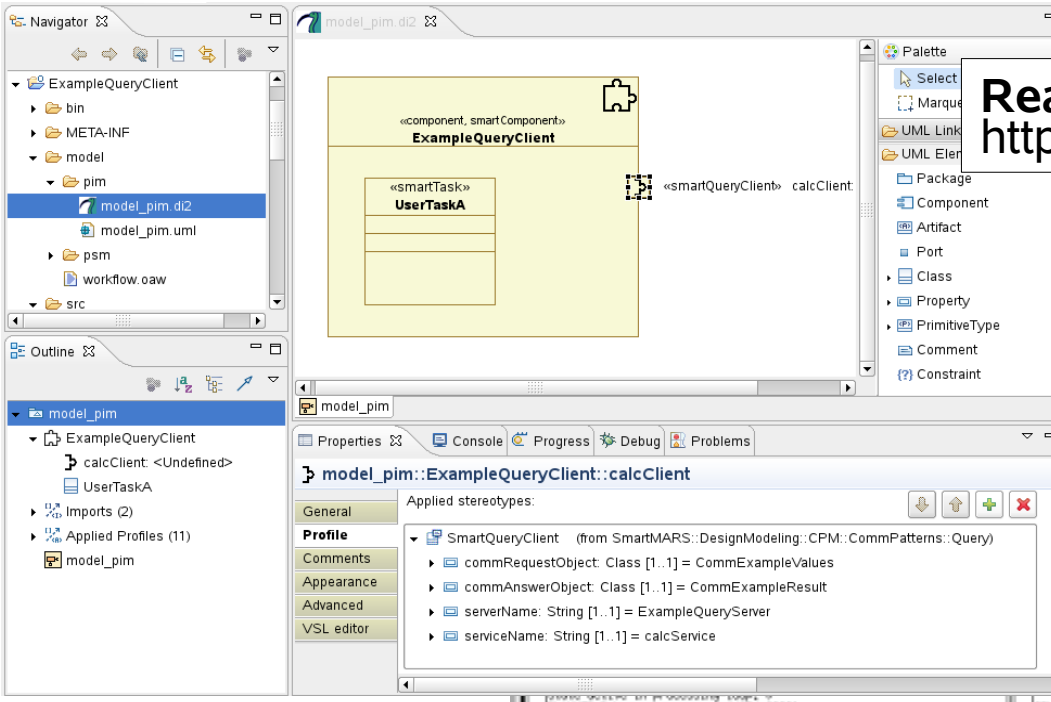
- Toolchain based on Open Architecture Ware
  - fully integrated into Eclipse
  - http://www.openarchitectureware.org/
- MDSD Toolchain Example
  - PIM: SmartMARS robotics profile (Modeling and Analysis of Robotics Systems)
  - PSM: SmartSoft in different implementations but with the same semantics !
  - can be easily adapted to different profiles / profile extensions / PSMs
- SmartSoft [LGPL]
  - http://smart-robotics.sourceforge.net/
  - http://www.zafh-servicerobotik.de/ULM/en/smartsoft.php
  - CORBA (ACE/TAO) based SmartSoft
    - on web with various robotics components
  - ACE (without CORBA) based SmartSoft
    - under stress testing
    - soon available [Linux, Windows, ...] on sourceforge
  - oAW Toolchain for SmartSoft
    - will soon be available on sourceforge

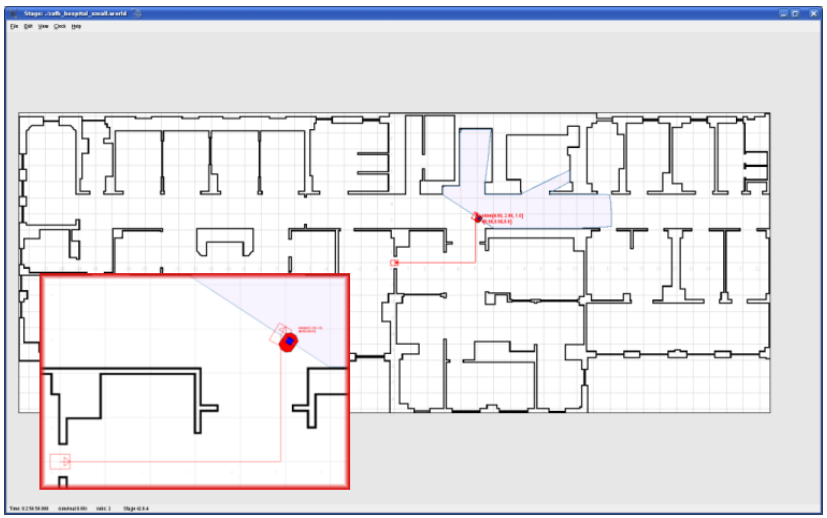


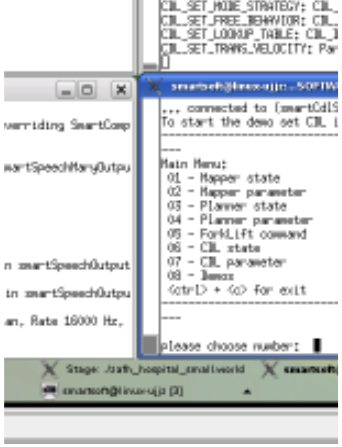
service robotik

Model Driven Software Development
Summary and Conclusion

Ready to run VMWare image
http://sourceforge.net/projects/smart-robotics/download







Model Driven Software Development
Summary and Conclusion

MDSD Toolchain - Screencast

ZAFH Ulm video2 05-2009.swf

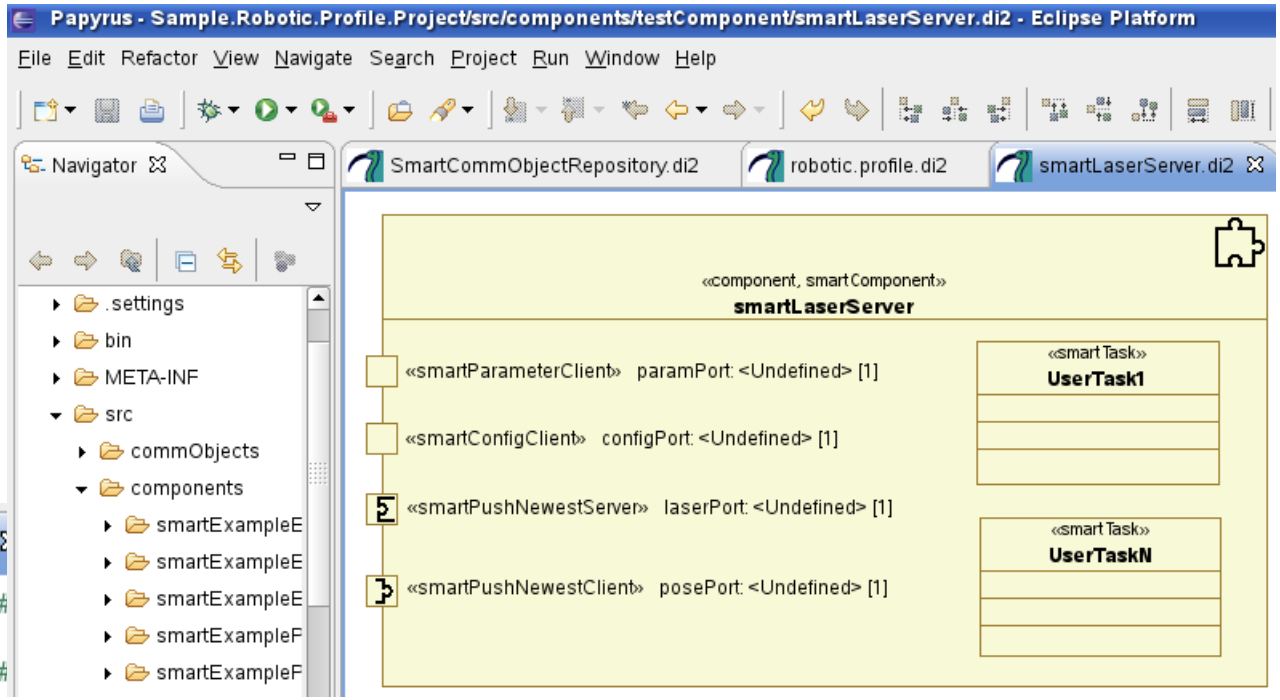
http://www.zafh-servicerobotik.de/ULM/en/dokumente/ZAFH Ulm video2 05-2009.swf

http://smart-robotics.sourceforge.net/

service robotik

Model Driven Software Development
Idea and Approach

Example of a
Model Transformation Template



smartLaserServer.di2 | smartLaserServer.cc | *Root.xpt

```
«REM»################################################
## creation of communication patterns
################################################
```

```
«REM»---- SmartPushNewestServer ----«ENDREM»
«DEFINE Create FOR robotic::DesignModeling::CPM::CommPatterns::PushNewest::SmartPushNewestServer»
        «this.name» = new CHS::PushNewestServer<CHS::«this.commObject.name»>(component,"«this.serviceName»");
«ENDDEFINE»

«REM»---- SmartPushNewestClient ----«ENDREM»
«DEFINE Create FOR robotic::DesignModeling::CPM::CommPatterns::PushNewest::SmartPushNewestClient»
        «this.name» = new CHS::PushNewestClient<CHS::«this.commObject.name»>(component);
        «this.name»->connect("«this.serverName»","«this.serviceName»");
        «this.name»->subscribe();
«ENDDEFINE»
```


Hochschule Ulm

service robotik

Model Driven Software Development
Idea and Approach

SmartCommObjectRepository.di2 | robotic.profile.di2 | smartLaserSe

```
#include "smartSoft.hh"
#include "commMobileLaserScan.hh"
#include "commBaseState.hh"

CHS::SmartComponent *component;
////////////////////////////////////////////////////////////////
// communication-patterns
CHS::PushNewestServer<CHS::CommMobileLaserScan> *laserPort;
CHS::PushNewestClient<CHS::CommBaseState> *posePort;

////////////////////////////////////////////////////////////////
// internal classes
class UserTaskN : public CHS::SmartTask {
public:
  UserTaskN() {};
  ~UserTaskN() {};
  int svc(void);
};

int UserTaskN::svc(void) {
    /*PROTECTED REGION ID(UserTaskN) ENABLED START*/
    // -- put your sourcecode here --

    return 0;
    /*PROTECTED REGION END*/
}

class UserTask1 : public CHS::SmartTask {
public:
  UserTask1() {};
  ~UserTask1() {};
```

Example of generated code with protected user sections not touched by the code generator

Hochschule Ulm

service robotik

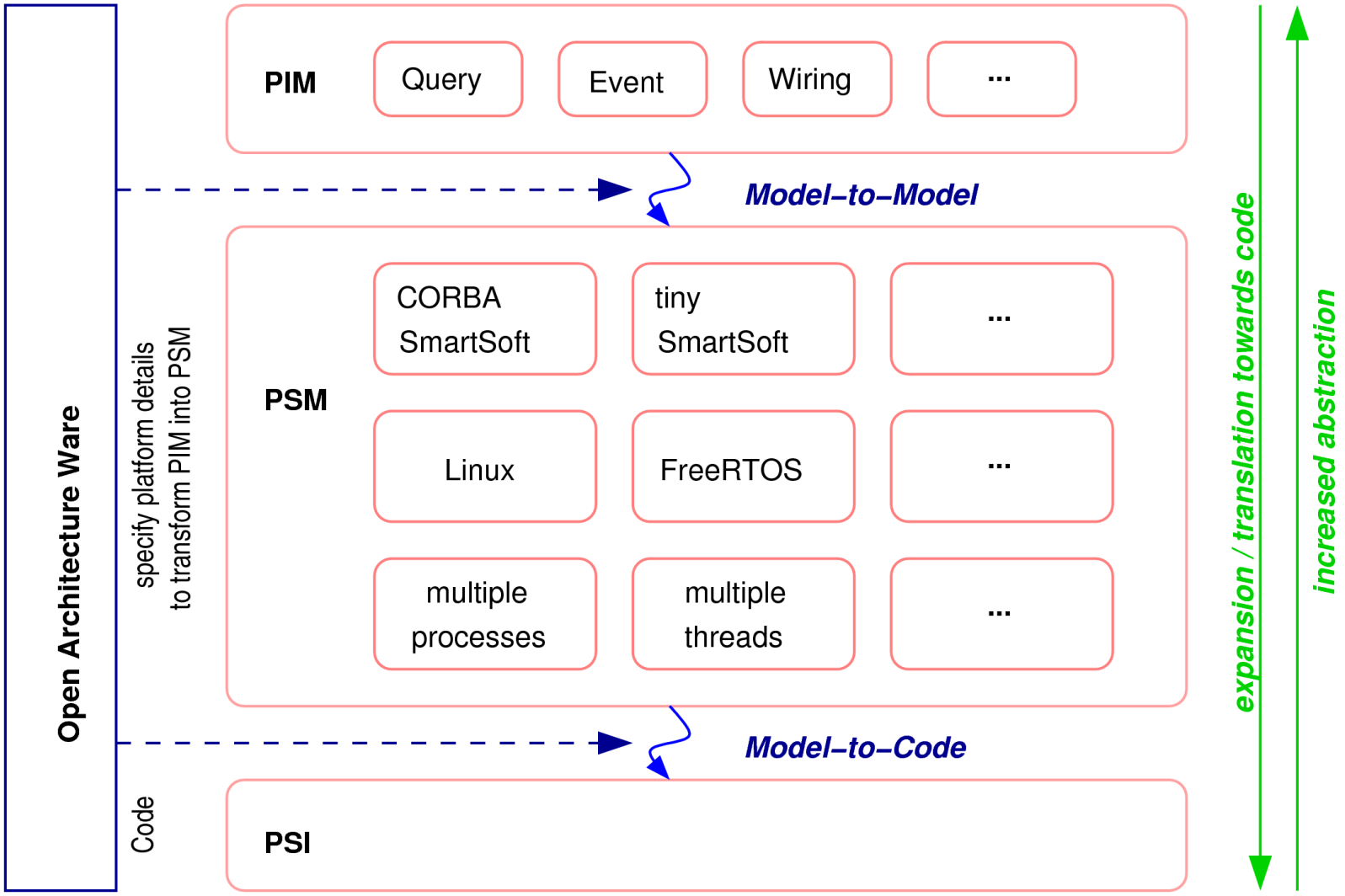
Model Driven Software Development
Idea and Approach

SmartCommObjectRepository.di2 | robotic.profile.di2 | smartLaserServer.di2 | workflow.oaw | smar

```
////////////////////////////////////////////////////////////////////////////////
// main
int main (int argc, char *argv[]) {
    try {
        component = new CHS::SmartComponent("smartLaserServer",argc,argv);
        laserPort = new CHS::PushNewestServer<CHS::CommMobileLaserScan>(component,"laser");
        posePort = new CHS::PushNewestClient<CHS::CommBaseState>(component);
        posePort->connect("smartBaseServer","pose");
        posePort->subscribe();

        UserTaskN userTaskN;
        UserTask1 userTask1;

        // run all
        userTaskN.open();
        userTask1.open();

        component->run();
    } catch (const CORBA::Exception &) {
        std::cerr << "Uncaught CORBA exception" << std::endl;
        return 1;
    } catch (...) {
        std::cerr << "Uncaught exception" << std::endl;
        return 1;
    }
    delete component;
    return 0;
}
```


Hochschule Ulm

Model Driven Software Development
Idea and Approach



service robotik

Model Driven Software Development
Idea and Approach