Modellgetriebene Softwareentwicklung in der Servicerobotik

Prof. Dr. Christian Schlegel

Fakultät Informatik
Hochschule Ulm

http://www.zafh-servicerobotik.de/ULM/index.php
http://www.hs-ulm.de/schlegel

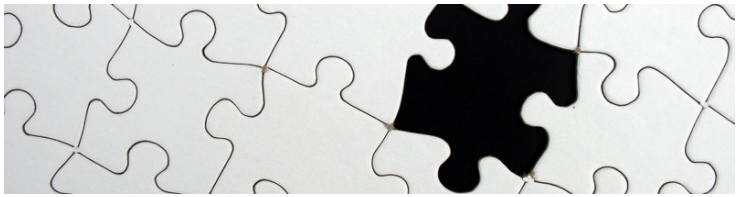
AG Schlegel:
M.Sc. Siegfried Hochdorfer, B.Sc. Matthias Lutz, B.Sc. Dennis Stampfer, B.Sc. Andreas Steck

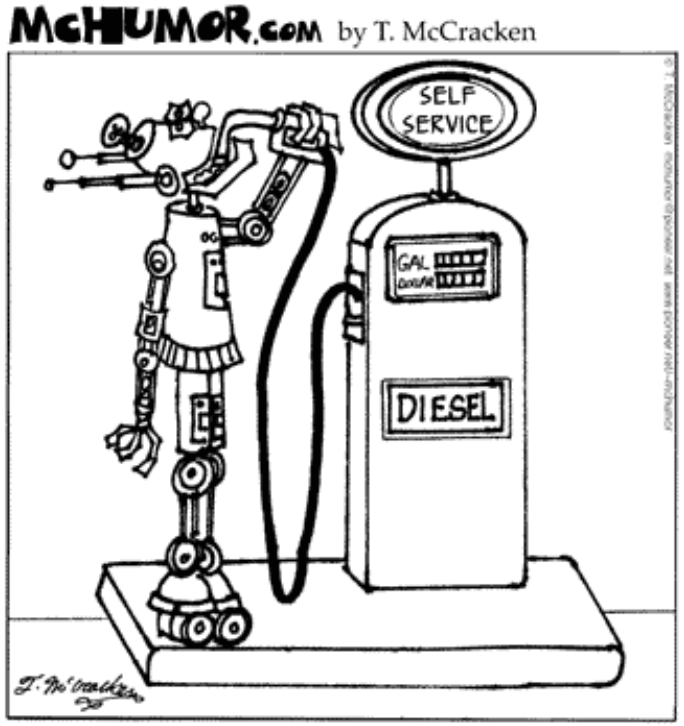
Hochschule Ulm

Modellgetriebene Softwareentwicklung
Einleitung und Motivation

- Die Beherrschung der Softwarekomplexität sowie des Softwareentwicklungsprozesses ist eine der grundlegenden Voraussetzungen, um den Massenmarkt Servicerobotik zu erschließen und zu entwickeln
- offensichtliche Lücken bspw. in Robustheit und Zuverlässigkeit zwischen aktuellen Systemen und dem benötigten Niveau können nur geschlossen werden durch
  - ***Vollziehen des Schrittes von code-zentrierten zu modell-getriebenen Systemen***
  - ***nicht weiterhin ignorieren von Echtzeitaspekten (oder allgemeiner: Einführen von "resource awareness")***

http://www.itee.uq.edu.au/~milford/smltract.gif

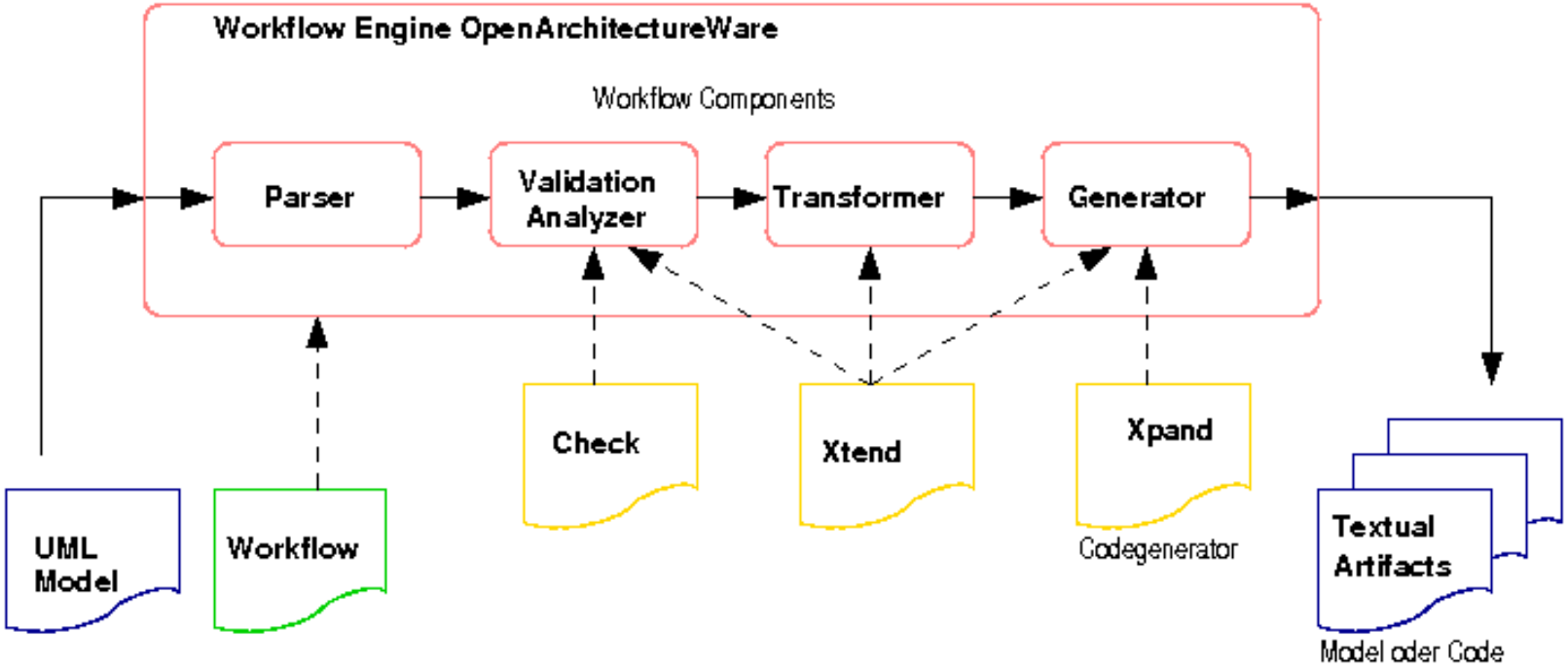




Modellgetriebene Softwareentwicklung
Zielsetzung und Fragestellung

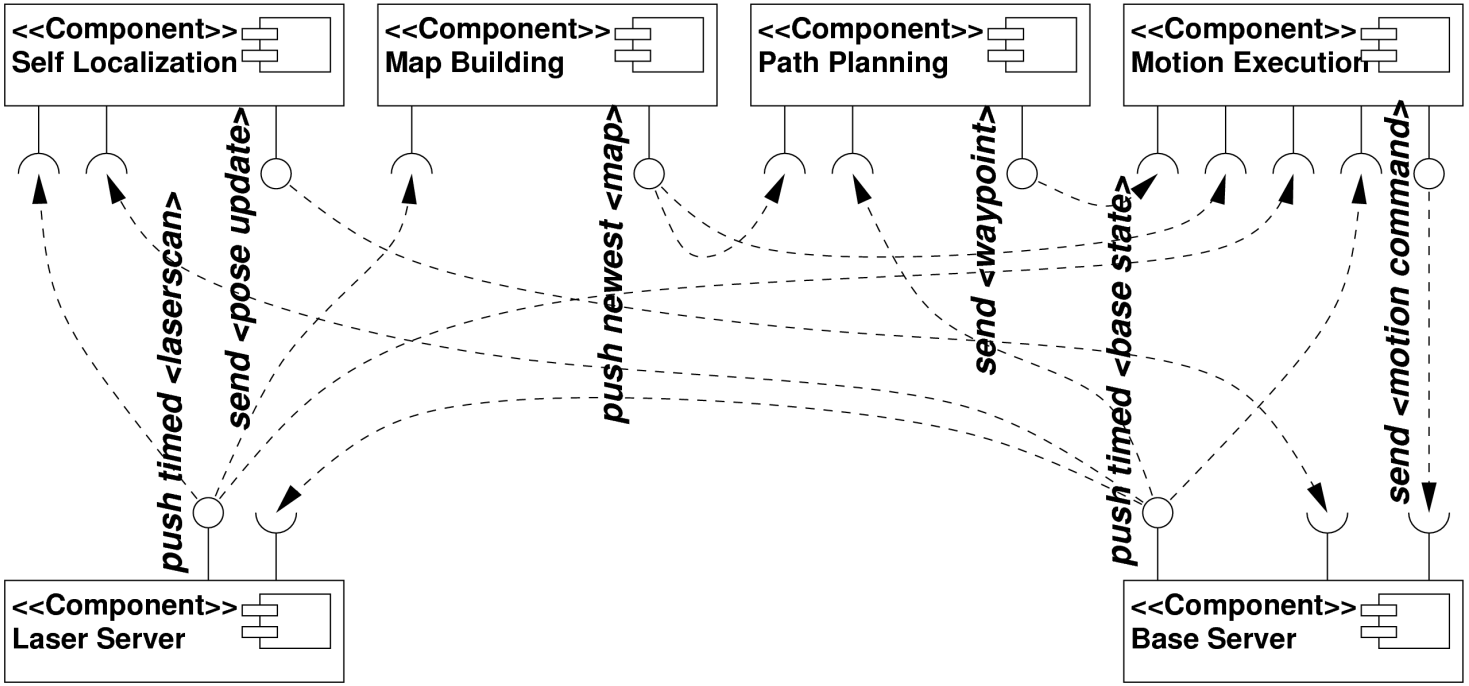
- weg von individuellen Einzelentwicklungen hin zu aus Standardbausteinen zusammengesetzten Systemen
- Modelle sind die Basis für Validierungs- und Verifikationstools
- Entkopplung des Problemlösungswissens von der Implementierungstechnologie

- wie modelliert man Robotics-Belange per UML ?
- welche domänenspezifischen Ergänzungen werden benötigt ?
- wie kann *Model Driven Software Development* in der Robotik funktionieren ?



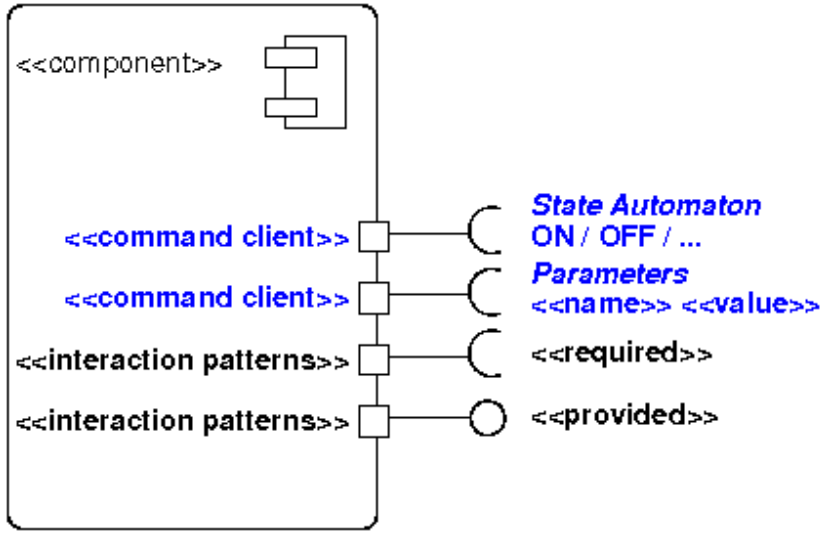
Hochschule Ulm

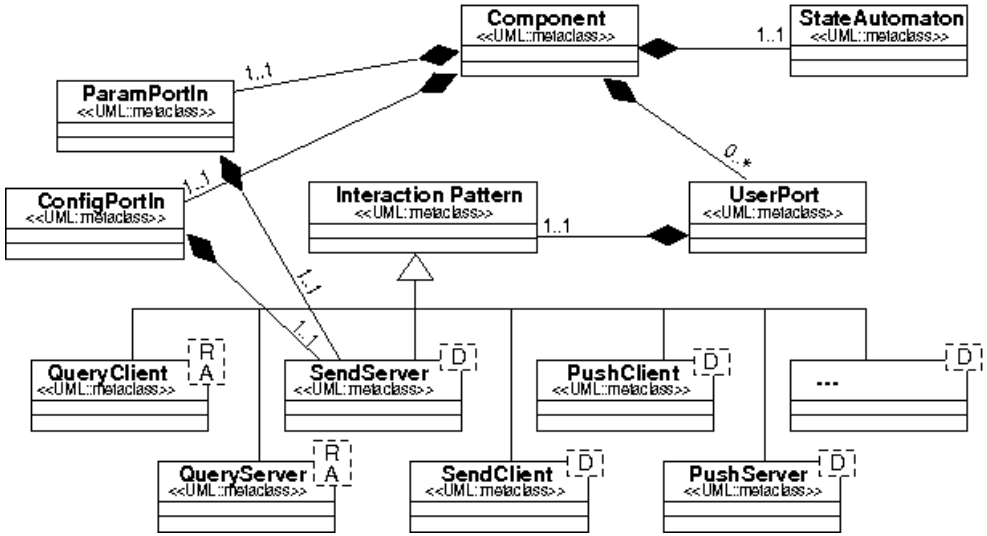
Modellgetriebene Softwareentwicklung
Idee und Ansatz

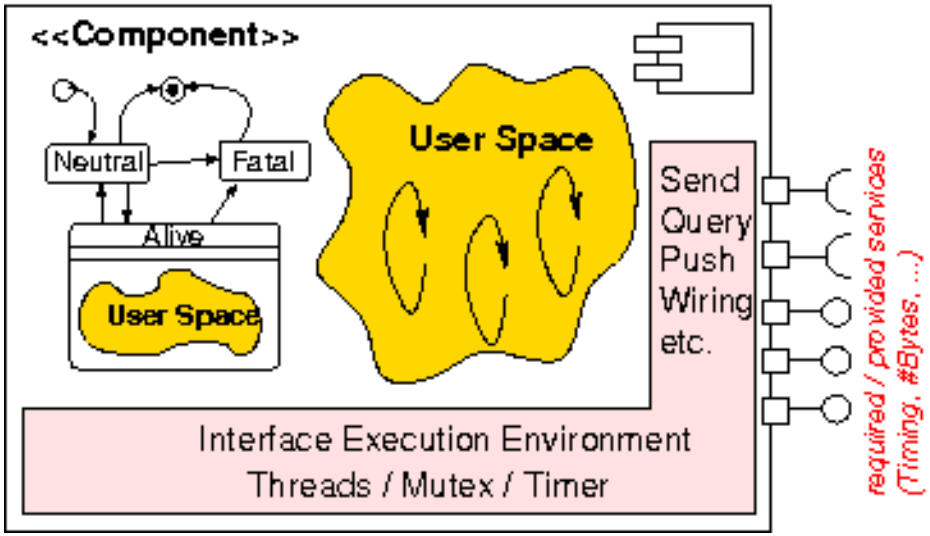


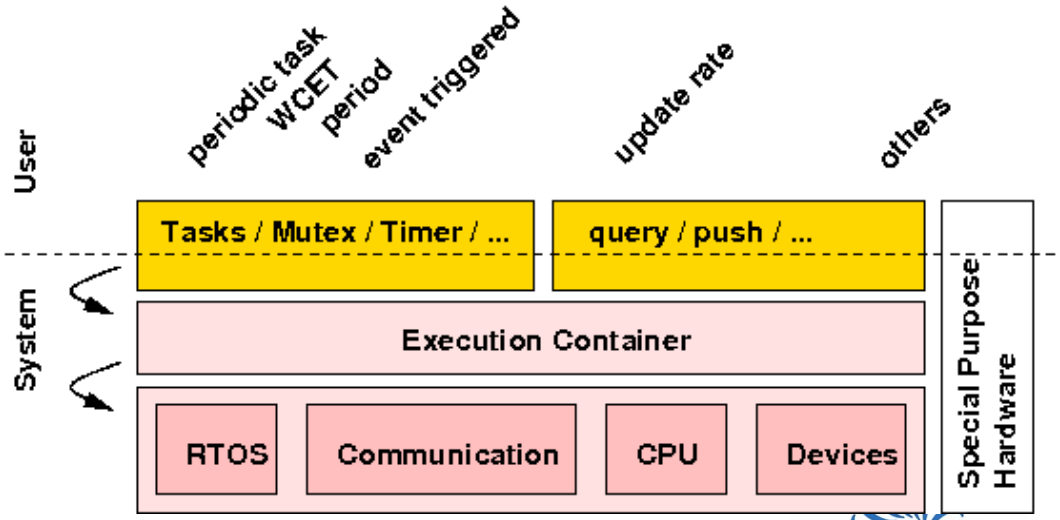
send	CHS::SendClient, CHS::SendServer, CHS::SendServerHandler
query	CHS::QueryClient, CHS::QueryServer, CHS::QueryServerHandler
push newest	CHS::PushNewestClient, CHS::PushNewestServer
push timed	CHS::PushTimedClient, CHS::PushTimedServer, CHS::PushTimedHandler
event	CHS::EventClient, CHS::EventHandler, CHS::EventServer, CHS::EventTestHandler
wiring	CHS::WiringMaster, CHS::WiringSlave

Modellgetriebene Softwareentwicklung
Idee und Ansatz

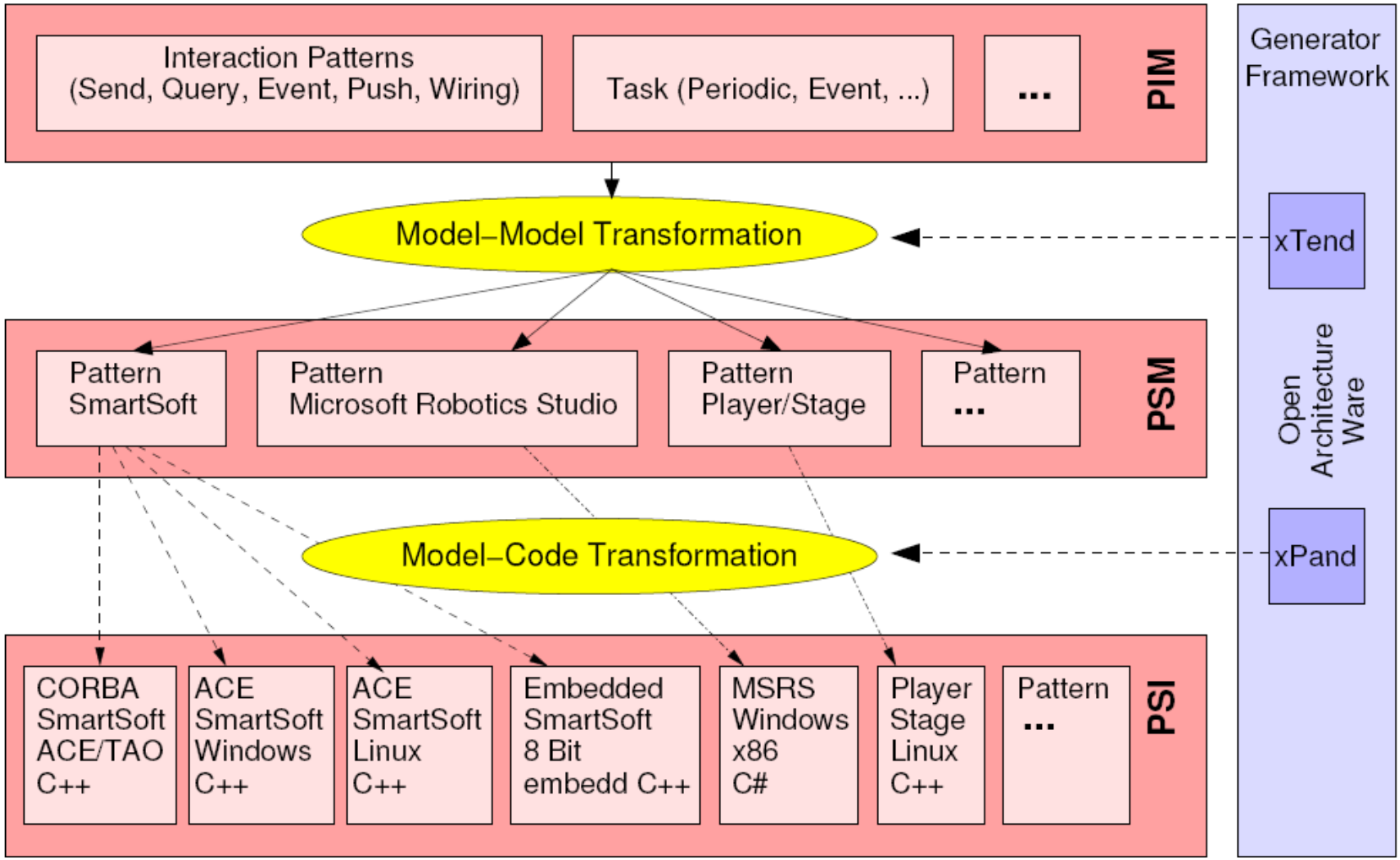






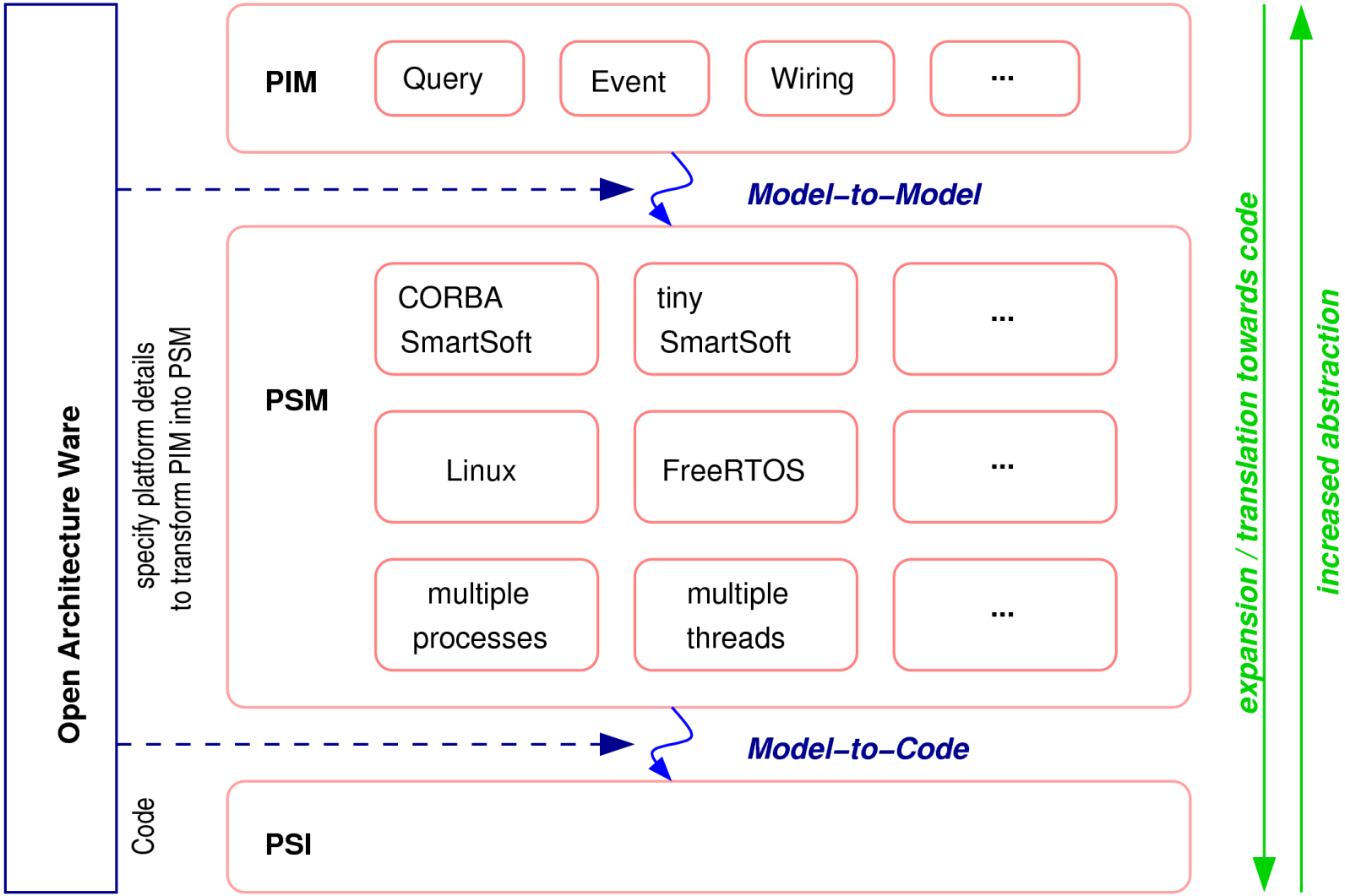


Modellgetriebene Softwareentwicklung
Aktuelle Arbeiten

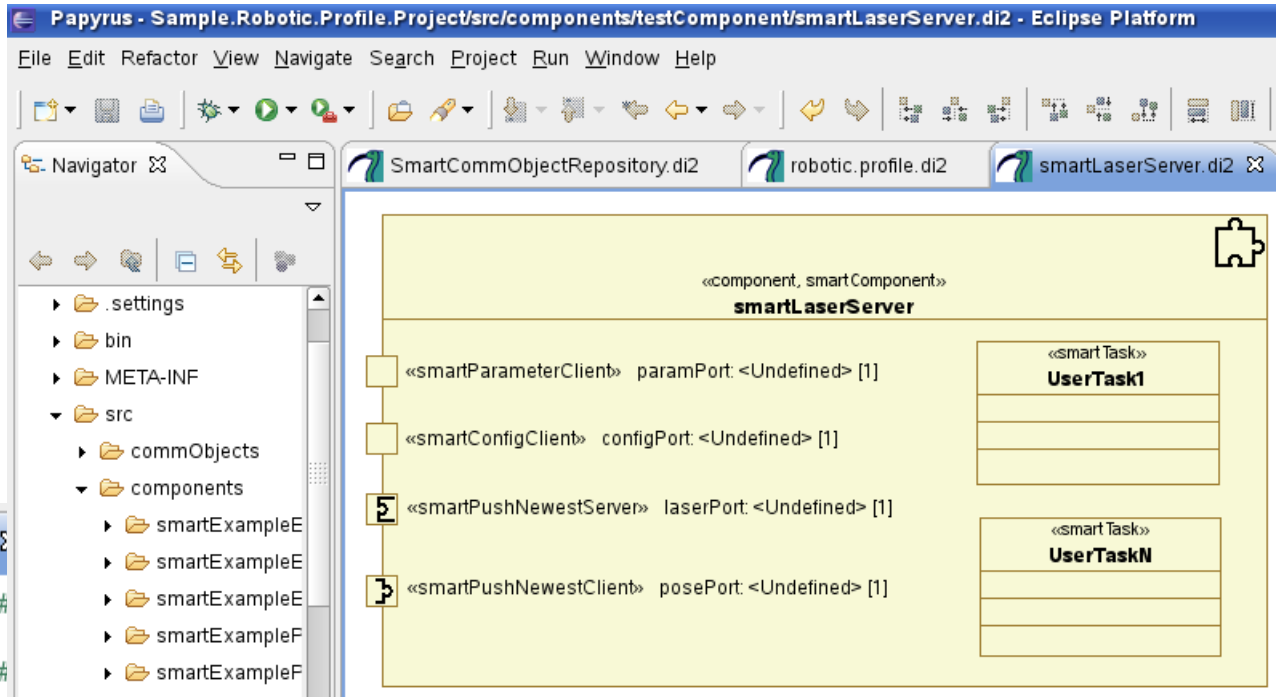


Hochschule Ulm

Modellgetriebene Softwareentwicklung
Aktuelle Arbeiten



Modellgetriebene Softwareentwicklung
Aktuelle Arbeiten



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»
```

Modellgetriebene Softwareentwicklung
Aktuelle Arbeiten

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() {};
  int svc(void);
};
```


Hochschule Ulm

Modellgetriebene Softwareentwicklung
Aktuelle Arbeiten

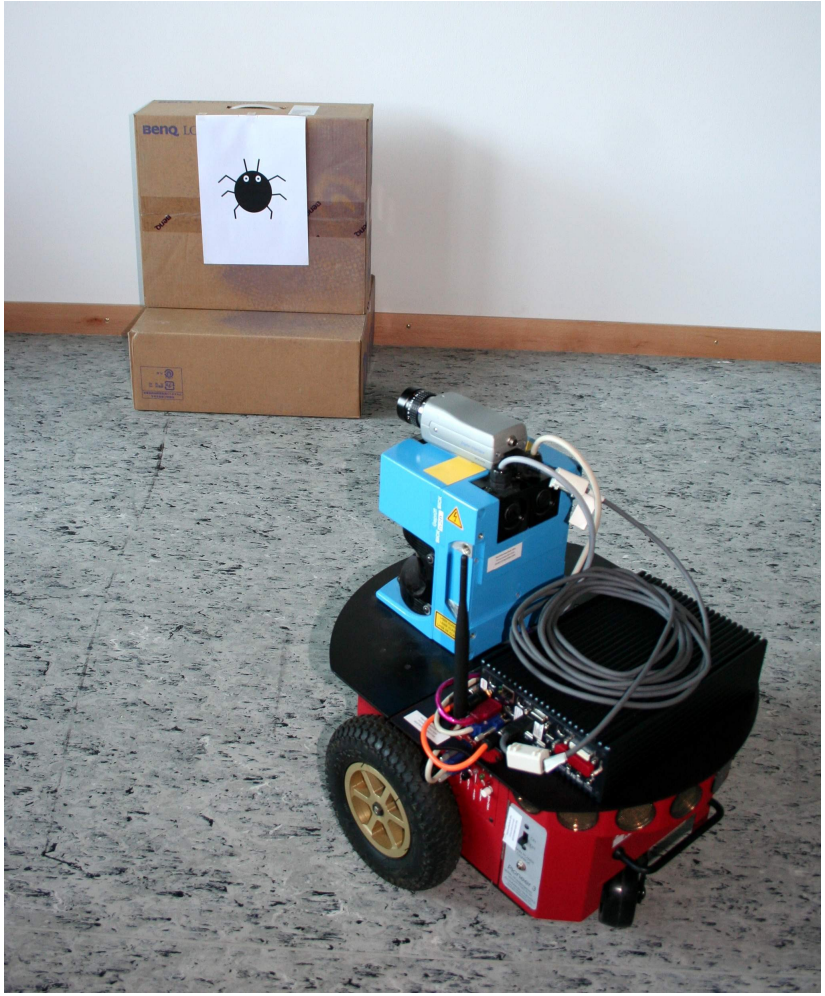
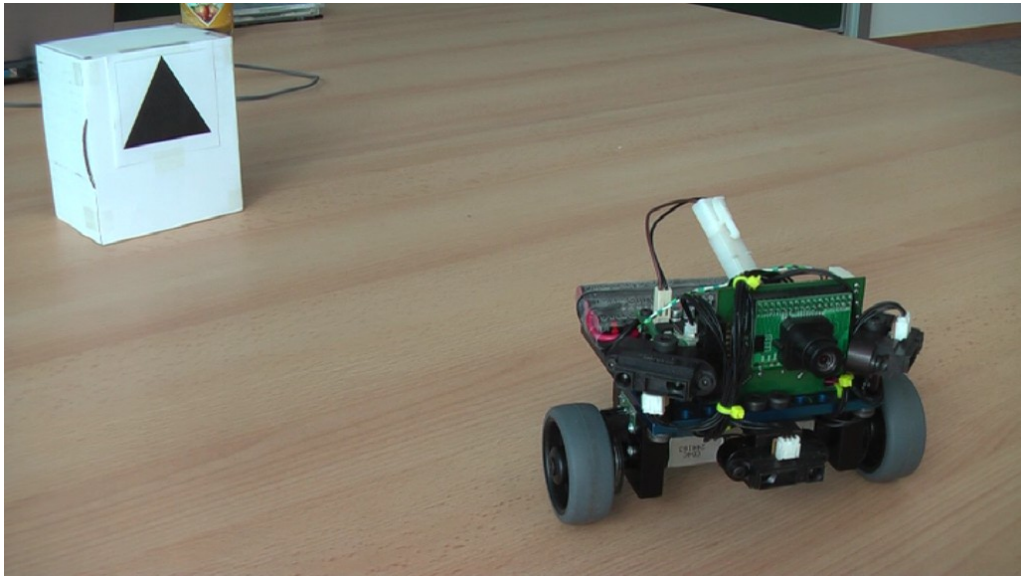
SmartCommObjectRepository.di2 | robotic.profile.di2 | smartLaserServer.di2 | workflow.oaw | smar

```
////////////////////////////////////////////////////////////////////////////////
// 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

Modellgetriebene Softwareentwicklung
Beispiel