



Servicerobotik

Autonome Mobile Serviceroboter

Software-Engineering in der Servicerobotik

Der Weg zum modellgetriebenen Softwareentwurf

Andreas Steck, Dennis Stampfer und Christian Schlegel

Institut für Informatik
Hochschule Ulm

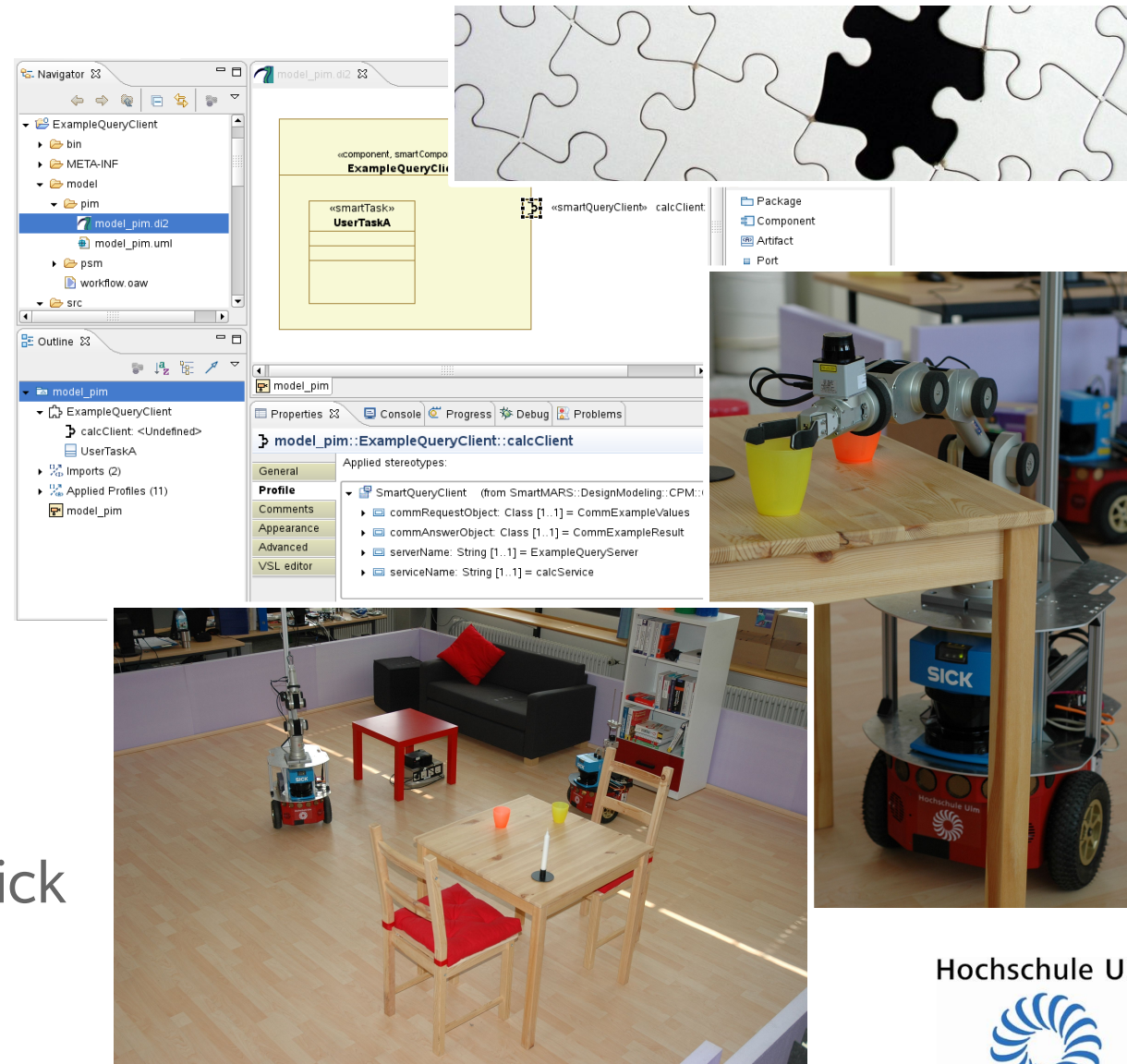
<http://smart-robotics.sourceforge.net/>

<http://www.zafh-servicerobotik.de/ULM/index.php>



Agenda

- ZAFH Servicerobotik
- Einleitung und Motivation
- Das SmartSoft Framework
- SmartSoft modellgetrieben
- Demonstration der Toolchain
- Zusammenfassung und Ausblick





Service robotik

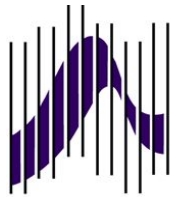
Autonome Mobile Serviceroboter

Hochschule Ulm



University of
Applied Sciences

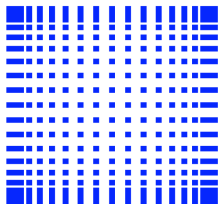
- **Hochschule Ulm**
University of Applied Sciences
Prof. Dr. Christian Schlegel



Hochschule
Ravensburg-Weingarten

University of Applied Sciences

- **Hochschule Ravensburg-Weingarten**
University of Applied Sciences
Prof. Dr. Wolfgang Ertel
Prof. Dr.-Ing. Holger Voos



hochschule mannheim

- **Hochschule Mannheim**
University of Applied Sciences
Prof. Dr.-Ing. Thomas Ihme
Prof. Dr. Bernhard Wirnitzer

Hochschule Ulm



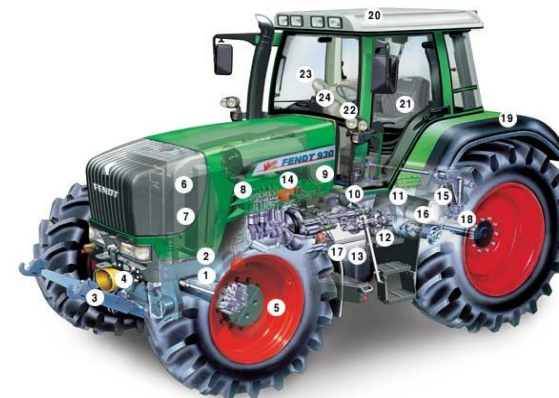
Service robotik

Autonome Mobile Serviceroboter

- **Lernfähige intelligente Steuerung**
AG Ertel / Ravensburg-Weingarten
- **Verifikation Sicherheitseigenschaften**
AG Voos / Ravensburg-Weingarten
- **Modellgetriebene Softwareentwicklung**
AG Schlegel / Ulm
- **Lokalisierung / Kartierung**
AG Schlegel / Ulm
- **Informationsoptimierte adaptive Objekterkennung**
AG Wirnitzer / Mannheim
- **Adaptive Realzeitbildverarbeitung**
AG Ihme / Mannheim



ZIEL:
Methodik
für Bau
alltagstauglicher
Serviceroboter
DEMONSTRATOR





Servicerobotik

Autonome Mobile Serviceroboter

Motivation:

- Serviceroboter sollen zukünftig selbständig Dienstleistungen in allen Lebensbereichen erbringen
- die Entwicklung eines Serviceroboters für den Einsatz in veränderlichen Umgebungen stellt große Herausforderungen an Algorithmik, Architekturen und Entwicklungsprozesse dar
- ausgereifte Algorithmen und etablierte Lösungen existieren für Teilaufgaben
- ein systematischer Ingenieursansatz zum Bau von Servicerobotern fehlt noch immer

Zielsetzung:

- Entwickeln einer Methodik für den Bau intelligenter Serviceroboter
- Entwicklung von autonomen mobilen Servicerobotern stark vereinfachen

Ansatz:

- Erweiterung und Zusammenführung bisher separierter Techniken unter dem Leitthema **Alltagstauglichkeit**





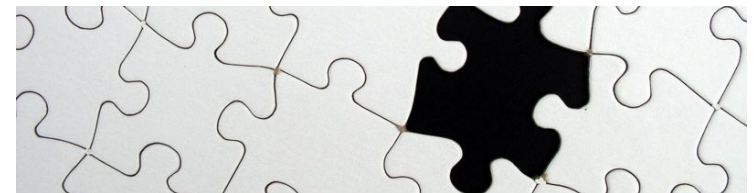
Modellgetriebene Softwareentwicklung

Einleitung und Motivation

Warum ist die modellgetriebene Softwareentwicklung wichtig für die Robotik?

- wegkommen von den von “Hand” programmierten Systemen
- entwerfen von Systemen aus Standardkomponenten mit explizierten Eigenschaften und Parametern
- wiederverwenden und modifizieren der Lösungen auf Modellebene
- von den Erfahrungen der Softwareingenieure profitieren Transformationen (M2M / M2T), optimierte Frameworks, effiziente Strukturen, ...
- analysieren und verifizieren der Parameter
- ... und viele weitere

Der Schritt zur modellgetriebenen Softwareentwicklung ist maßgeblich zur Erschließung breiter Anwendungspotentiale für (Service)-Roboter





Modellgetriebene Softwareentwicklung

Einleitung und Motivation

Vergleich zur *distributed real-time and embedded (DRE)* Domäne



Gemeinsamkeiten:

- Datenverarbeitung in verteilten sensomotorischen Systemen
- Berücksichtigung von Echtzeit
- hardwarenah
- komplexe Systeme

Unterschiede:

- hohe Dynamik bei den Konfigurationen und Verdrahtungen
- dynamic wiring – ändern der Verdrahtung zur Laufzeit
- System kann nicht als “Ganzes” betrachtet werden
- sehr lose gekoppelte Komponenten mit lokalen Verantwortlichkeiten





Modellgetriebene Softwareentwicklung

Einleitung und Motivation

Wir haben einige ausschlaggebende Entscheidungen getroffen:

- Granularität, wie ein System aus Komponenten zusammengesetzt wird
 - lose gekoppelte Komponenten mit lokalen Verantwortlichkeiten
 - Services werden zur Verfügung gestellt / benötigt
- Streng festgelegte Interaktionsmuster zwischen den Komponenten
 - präzise definierte Semantik der Interaktionen
 - diese sind als Richtlinien zu verstehen (auf jede Middleware abbildbar)
→ *unabhängig von der Implementierungstechnologie*
- minimales Komponentenmodell zur Systemintegration
 - dynamic wiring des Datenflusses zwischen den Komponenten
 - interner Zustandsautomat um Komponenten in unterschiedliche Betriebsmodi zu bringen
 - param-/ config-Port um Komponenten von außen zu orchestrieren
→ *sicherstellen der Zusammensetzbarkeit/ Systemintegration*
- unabhängig von der Ausführungsumgebung
 - Tasks (periodic, non-periodic, hard real-time, no realtime), Synchronisation
→ *kann wiederum auf verschiedene Betriebssysteme abgebildet werden*





Modellgetriebene Softwareentwicklung

Das SmartSoft Framework

SmartSoft:

- die Idee
 - wie Systeme aus Komponenten zusammengesetzt werden
 - wie die Komponentenhülle aussieht
 - wie Komponenten miteinander interagieren
- die konkreten Implementierungen, basierend auf:
 - CORBA (ACE/TAO) => CorbaSmartSoft
 - ACE only
 - ...

Die Auswahl der Muster ist ausreichend, da sowohl Request/Response Interaktionen als auch asynchrone Mitteilungen und Push Services zur Verfügung stehen.

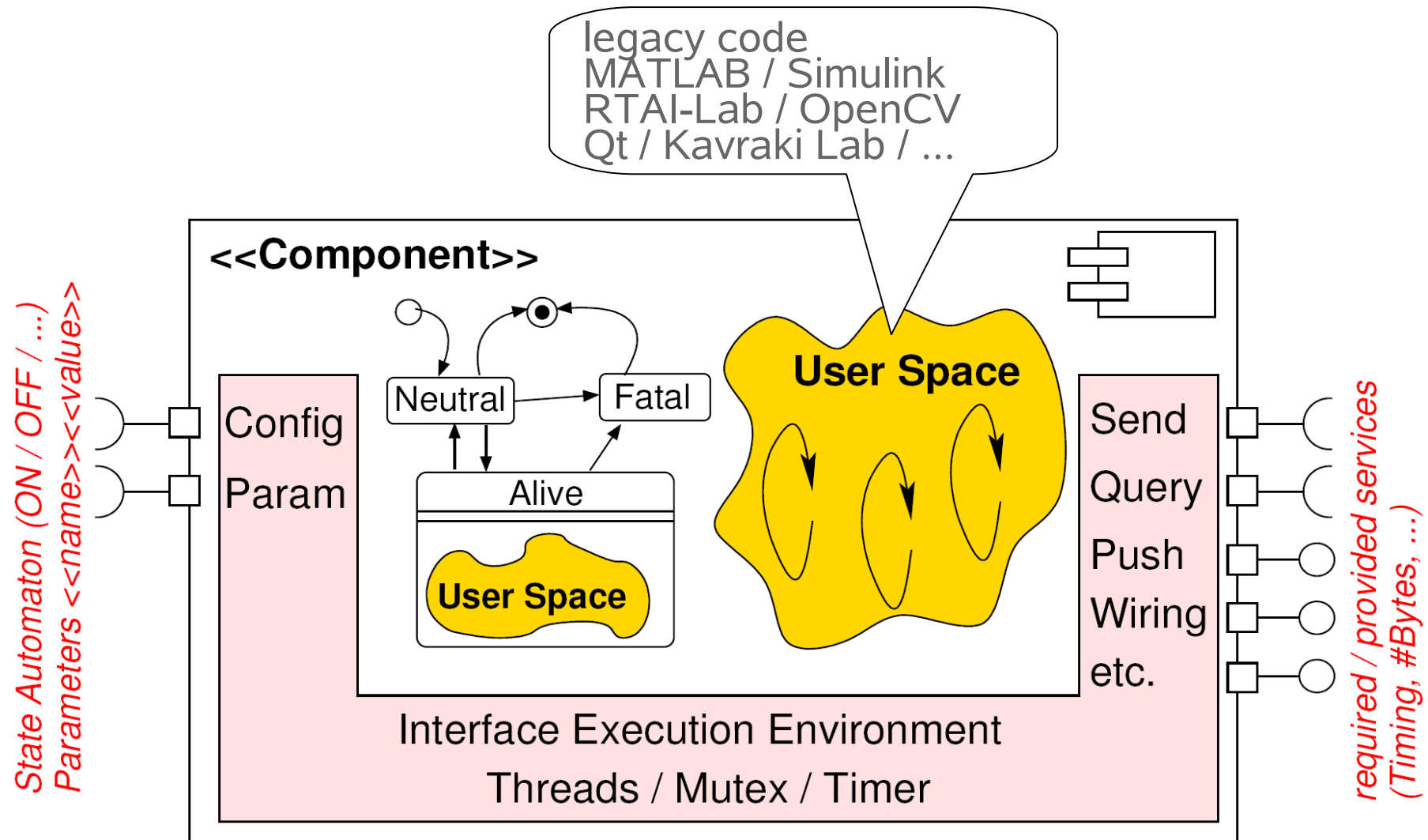
Die SmartSoft Interaktionsmuster

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



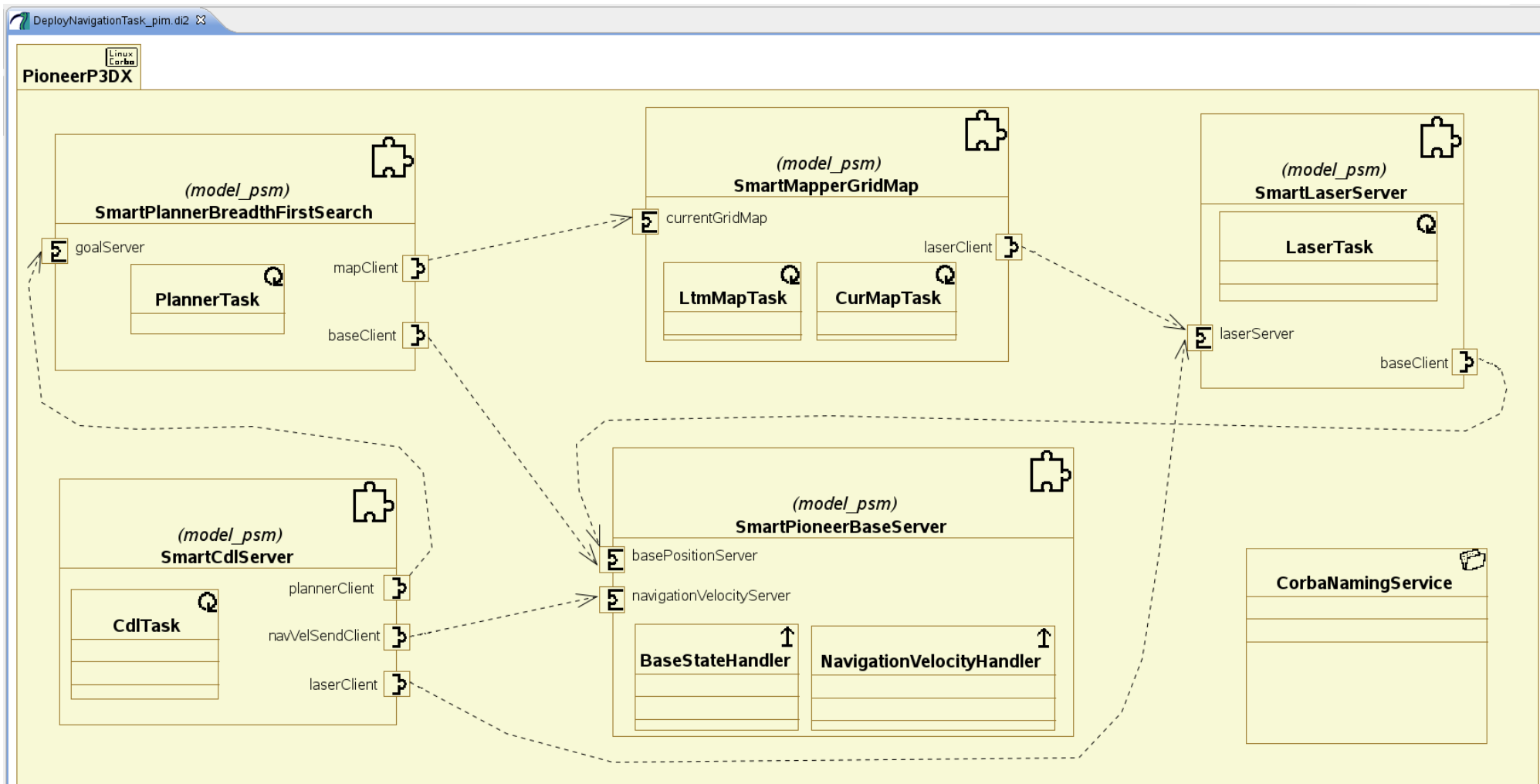
Modellgetriebene Softwareentwicklung

Das SmartSoft Framework



Modellgetriebene Softwareentwicklung

Das SmartSoft Framework



Modellgetriebene Softwareentwicklung

Das SmartSoft Framework

VIDEO



weitere Videos siehe:

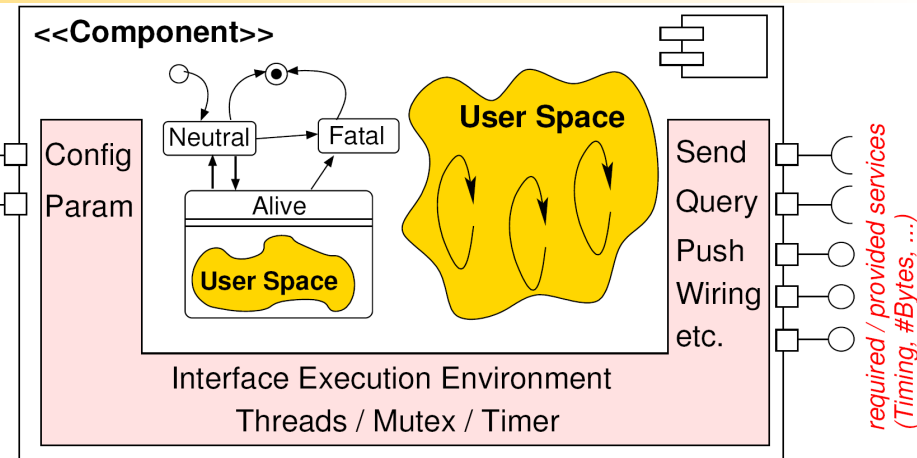
<http://smart-robotics.sourceforge.net/videos.php>

Modellgetriebene Softwareentwicklung

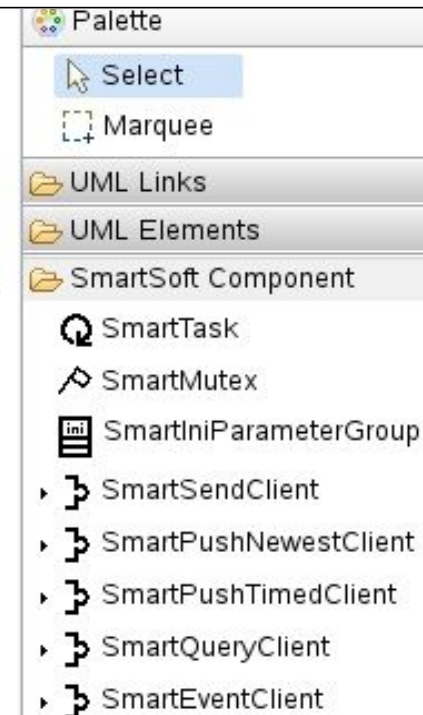
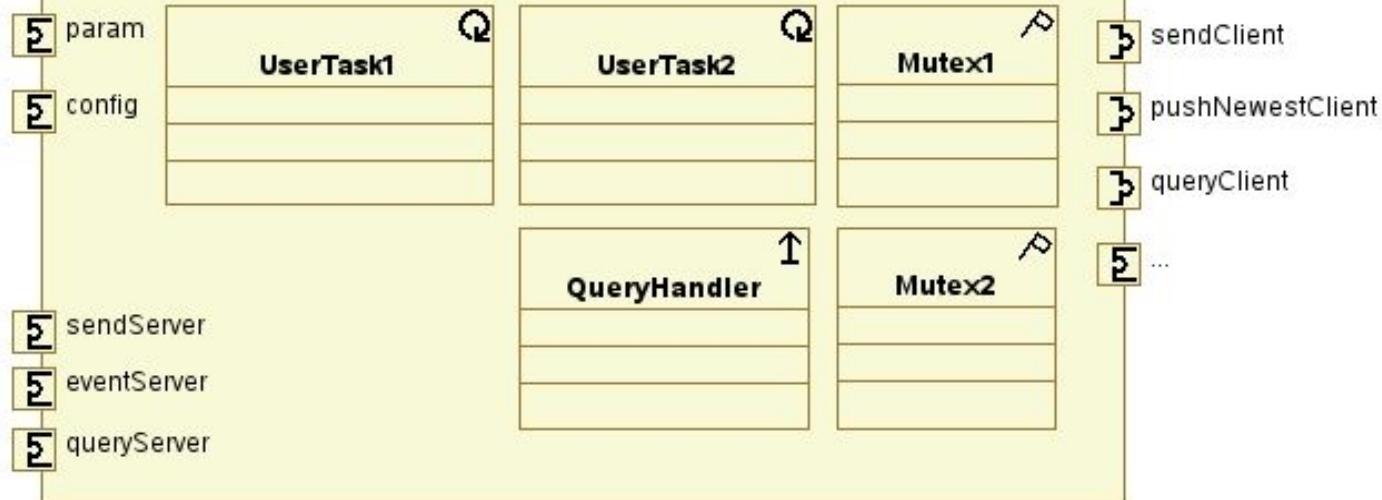
SmartSoft modellgetrieben

legacy code
MATLAB / Simulink
RTAI-Lab / OpenCV
Qt / Kavraki Lab / ...

State Automaton (ON / OFF / ...)
Parameters <<name>><<value>>



SampleComponent



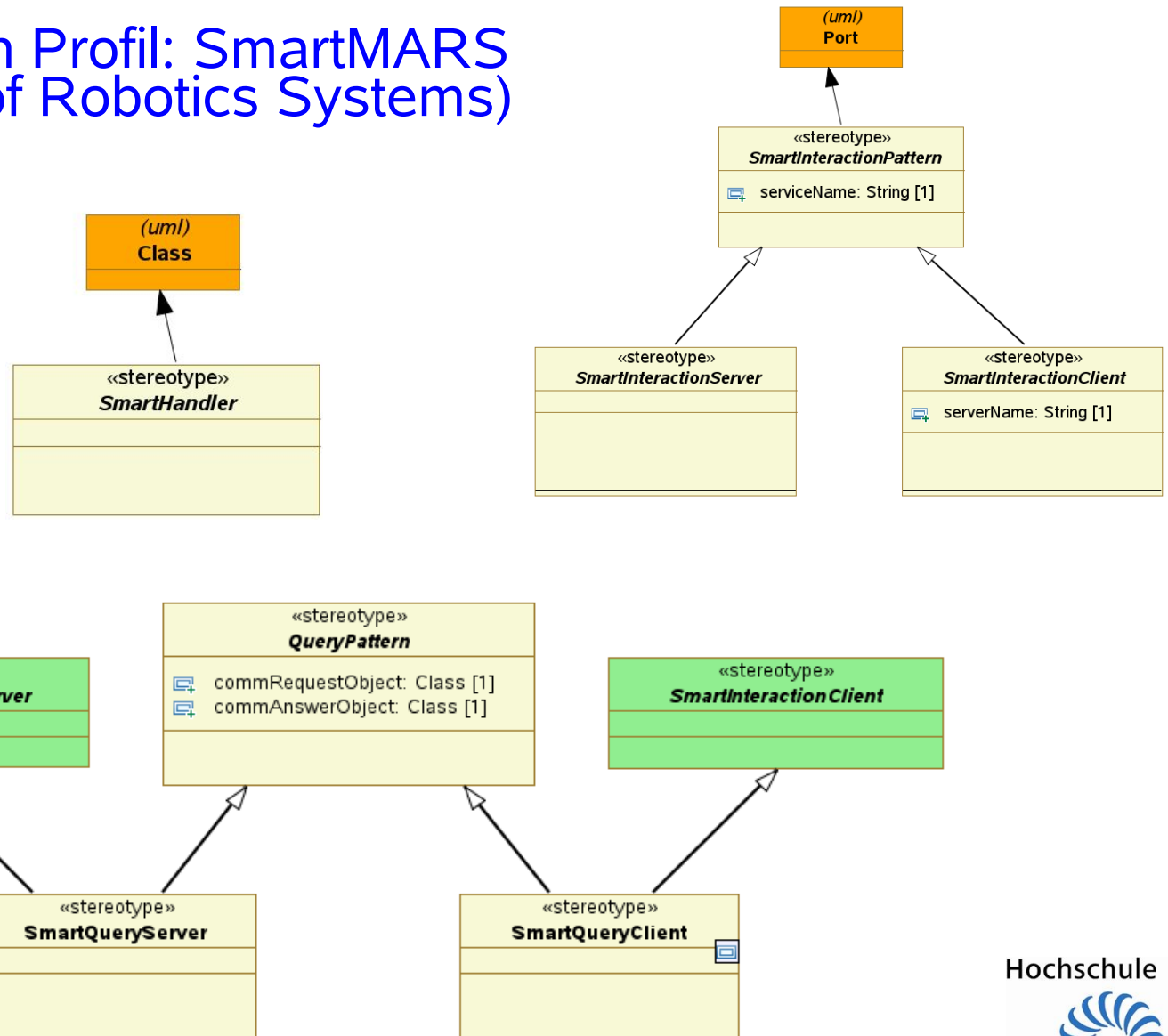
Modellgetriebene Softwareentwicklung

SmartSoft modellgetrieben

Erweiterung der UML durch Profil: SmartMARS (Modeling and Analysis of Robotics Systems)

- Stereotypen
- Tagged Values

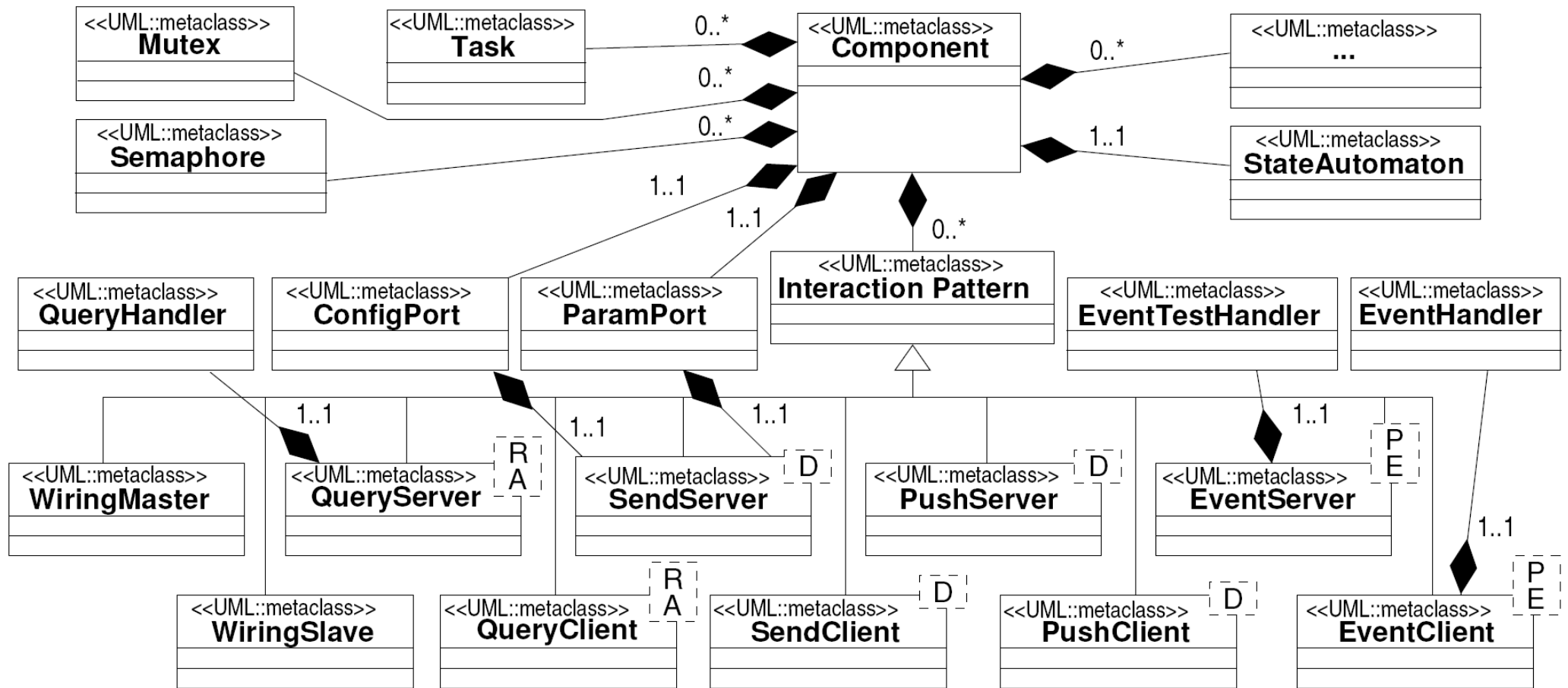
Auszug SmartMARS (Interaktionsmuster)



Modellgetriebene Softwareentwicklung

SmartSoft modellgetrieben

Ausschnitt SmartMARS (Modeling and Analysis of Robotics Systems)



Modellgetriebene Softwareentwicklung

SmartSoft modellgetrieben

PIM

SmartMARS – Metamodel
(Modeling and Analysis of Robotics Systems)

- UML2-Profil
- plattformunabhängige Stereotypen
 - SmartComponent
 - SmartTask
 - SmartMutex
 - SmartQueryServer
 - SmartEventClient
 - ...

M2M
oAW
xTend

PSM

CorbaSmartSoft
CORBA-basierte
Implementierung von SmartSoft

AceSmartSoft
ACE-basierte
Implementierung von SmartSoft

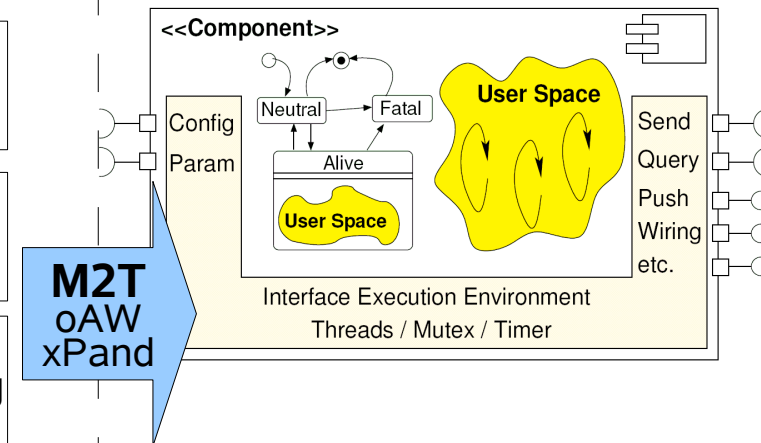
Microsoft Robotic Studio
MSRS-basierte Implementierung

...
jede andere Middleware

- UML2-Profil
- plattformspezifische Stereotypen

dies wird von
Middlewareexperten zur
Verfügung gestellt

PSI



M2T
oAW
xPand

Im Userspace können
beliebige Libraries
verwendet werden

Der Userspace muss nicht
angepasst werden, wenn
sich die Plattform ändert

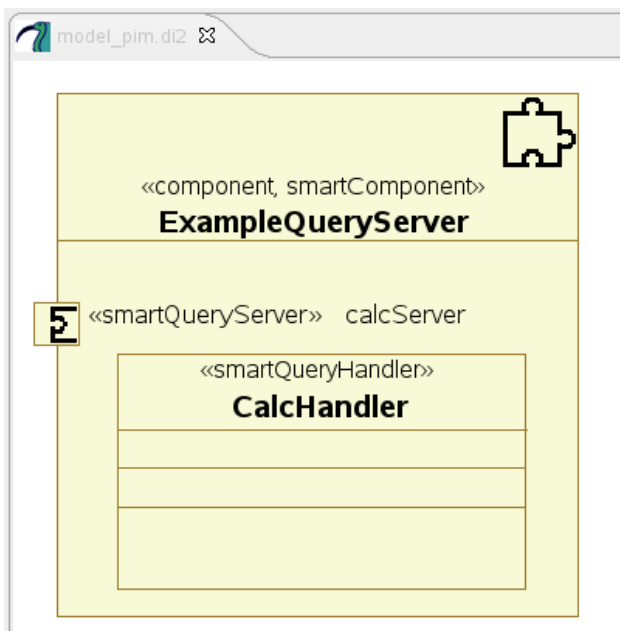
Es wird ausschließlich die
Komponentenhülle erzeugt

Modellgetriebene Softwareentwicklung

SmartSoft modellgetrieben

PIM

SmartMARS – Metamodel
(Modeling and Analysis of Robotics Systems)

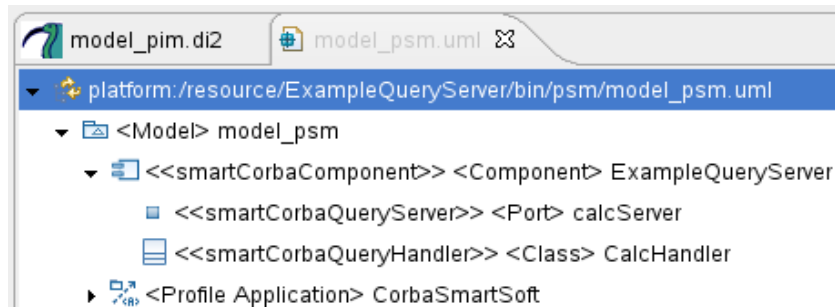


Benutzer erstellt PIM

PSM

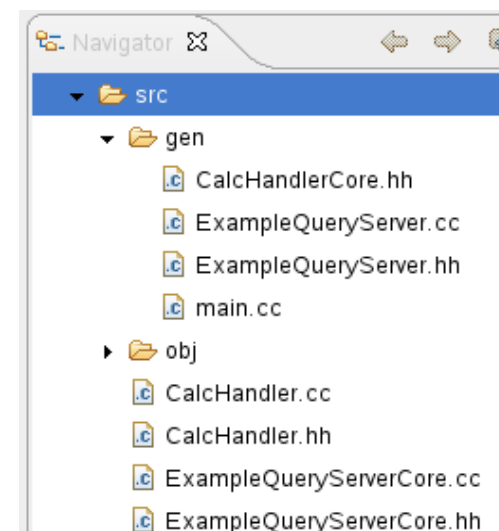
CorbaSmartSoft
CORBA-basierte
Implementierung von SmartSoft

M2M
oAW
xTend

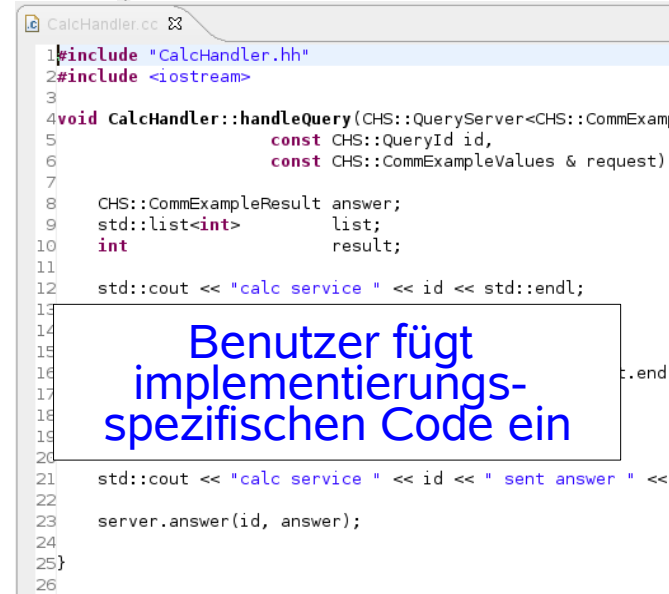


PSM wird vom
Workflow generiert

PSI



M2T
oAW
xPand



```

1 #include "CalcHandler.hh"
2 #include <iostream>
3
4 void CalcHandler::handleQuery(CHS::QueryServer<CHS::CommExample>
5     const CHS::QueryId id,
6     const CHS::CommExampleValues & request)
7
8     CHS::CommExampleResult answer;
9     std::list<int> list;
10    int result;
11
12    std::cout << "calc service " << id << std::endl;
13
14
15
16
17
18
19
20
21    std::cout << "calc service " << id << " sent answer " <<
22
23    server.answer(id, answer);
24
25 }
26
  
```

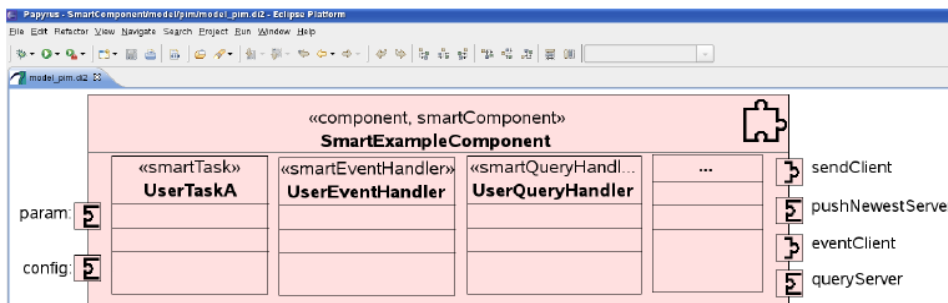
Benutzer fügt
implementierungs-
spezifischen Code ein

Modellgetriebene Softwareentwicklung

SmartSoft modellgetrieben

Aus Anwendersicht:

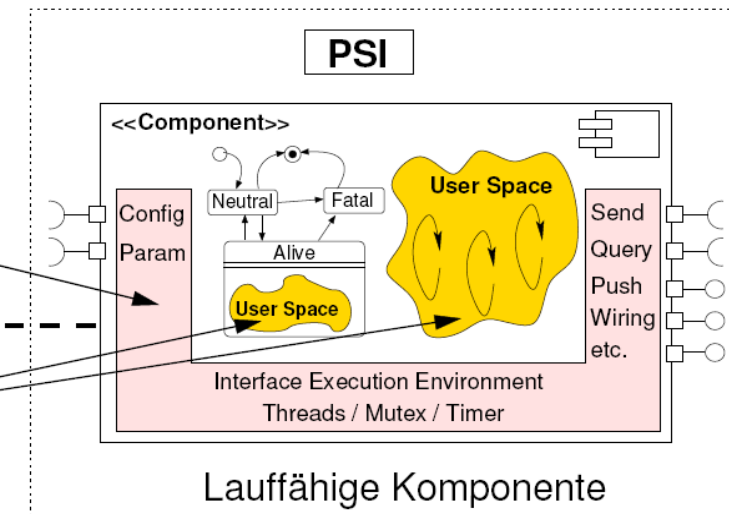
PIM der Komponentenhülle



Transformation
u. Verifikation
(z.B. QoS)

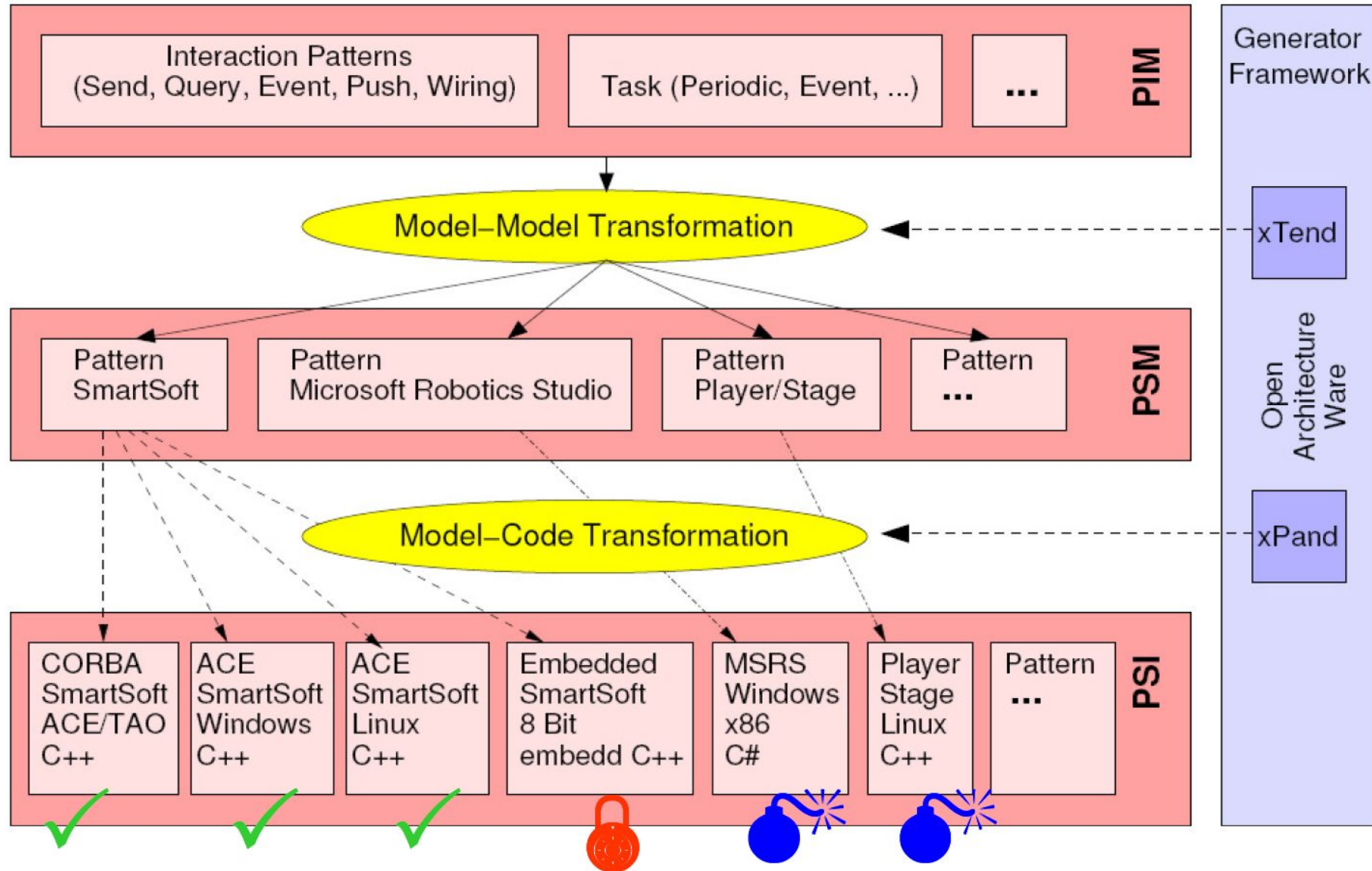
User Code

legacy code
MATLAB / Simulink
RTAI-Lab / OpenCV
Qt / Kavraki Lab / ...



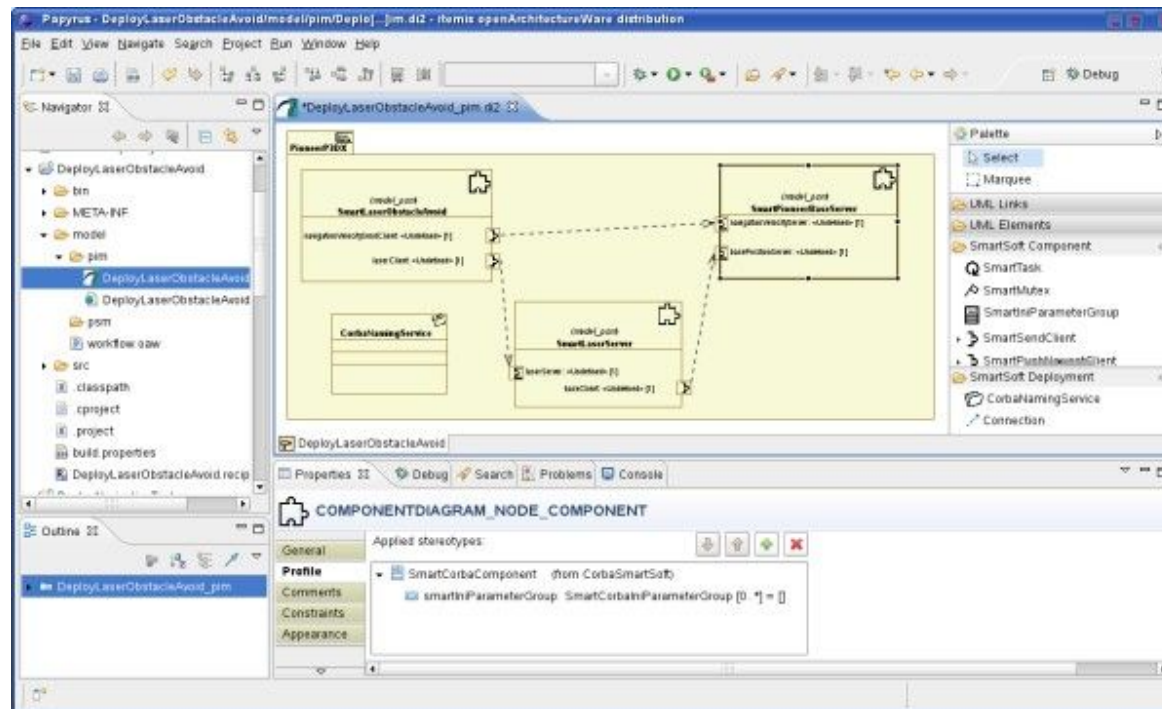
Modellgetriebene Softwareentwicklung

SmartSoft modellgetrieben



Modellgetriebene Softwareentwicklung SmartSoft modellgetrieben

MDSD Toolchain - Screencast



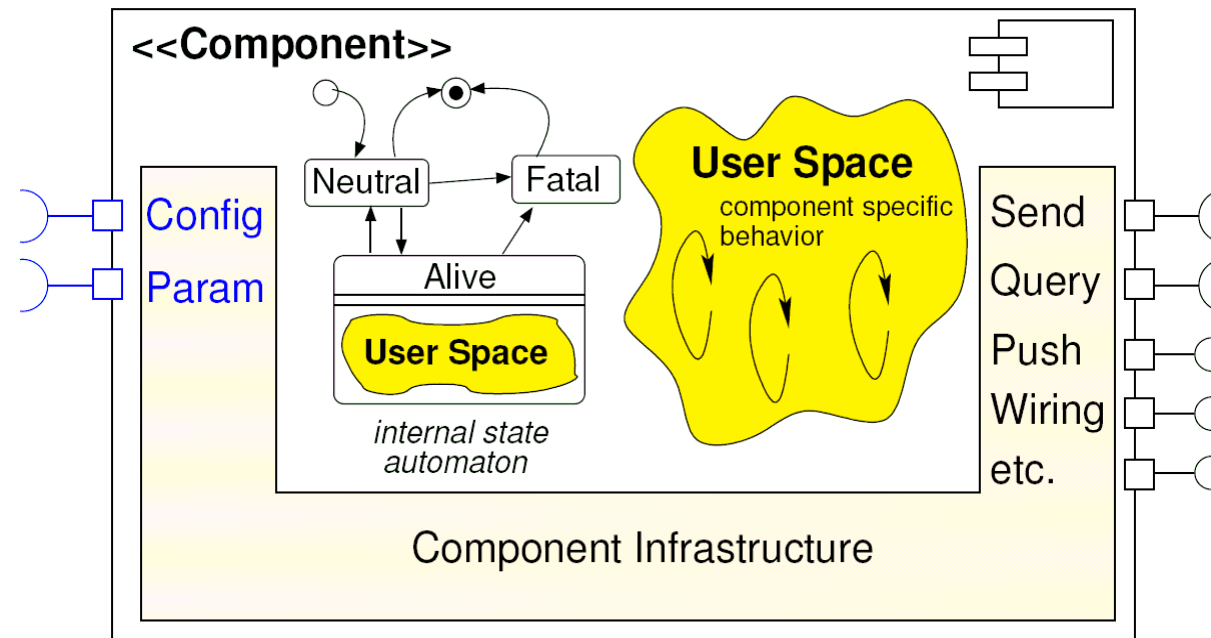
weitere Screencasts siehe

<http://smart-robotics.sourceforge.net/mdsdSmartSoft/screencasts.php>

Modellgetriebene Softwareentwicklung

SmartSoft modellgetrieben

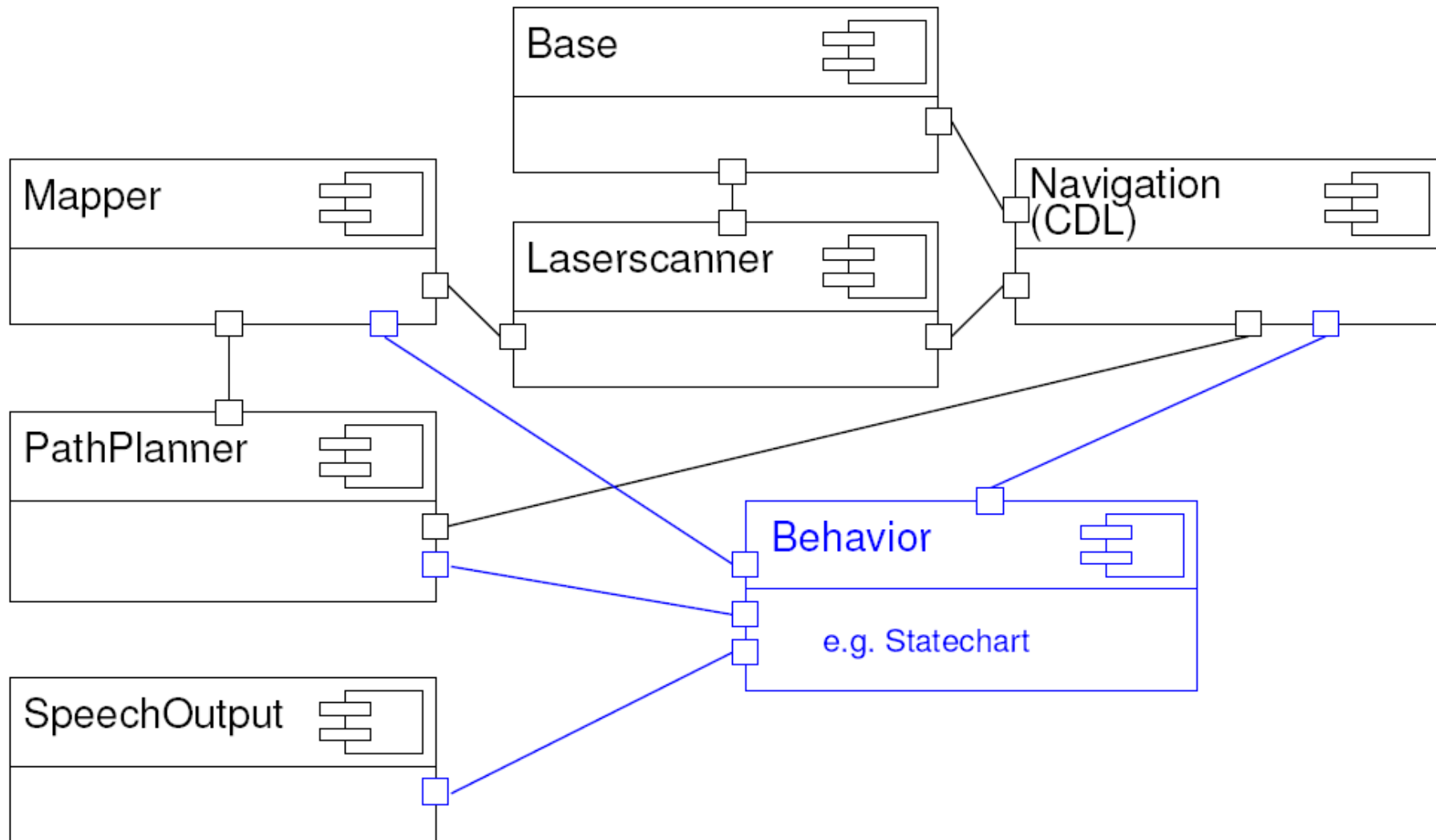
vereinheitlichter
Komponentenzugriff
von außen



- **Config:** setzen des internen Zustands der Komponente
 - navigation: **neutral; moverobot**
- **Param:** senden von Parametern
 - navigation: **(TRANSVEL(0)(500))**
 - pathPlanner: **(SETDESTINATIONCIRCLE(xPos)(yPos)(dist))**

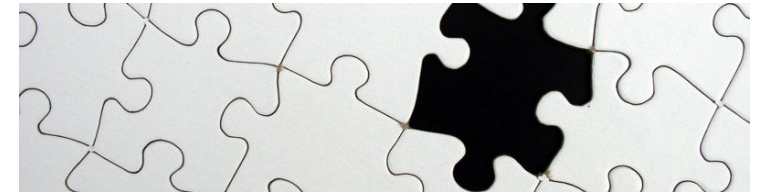
Modellgetriebene Softwareentwicklung

SmartSoft modellgetrieben



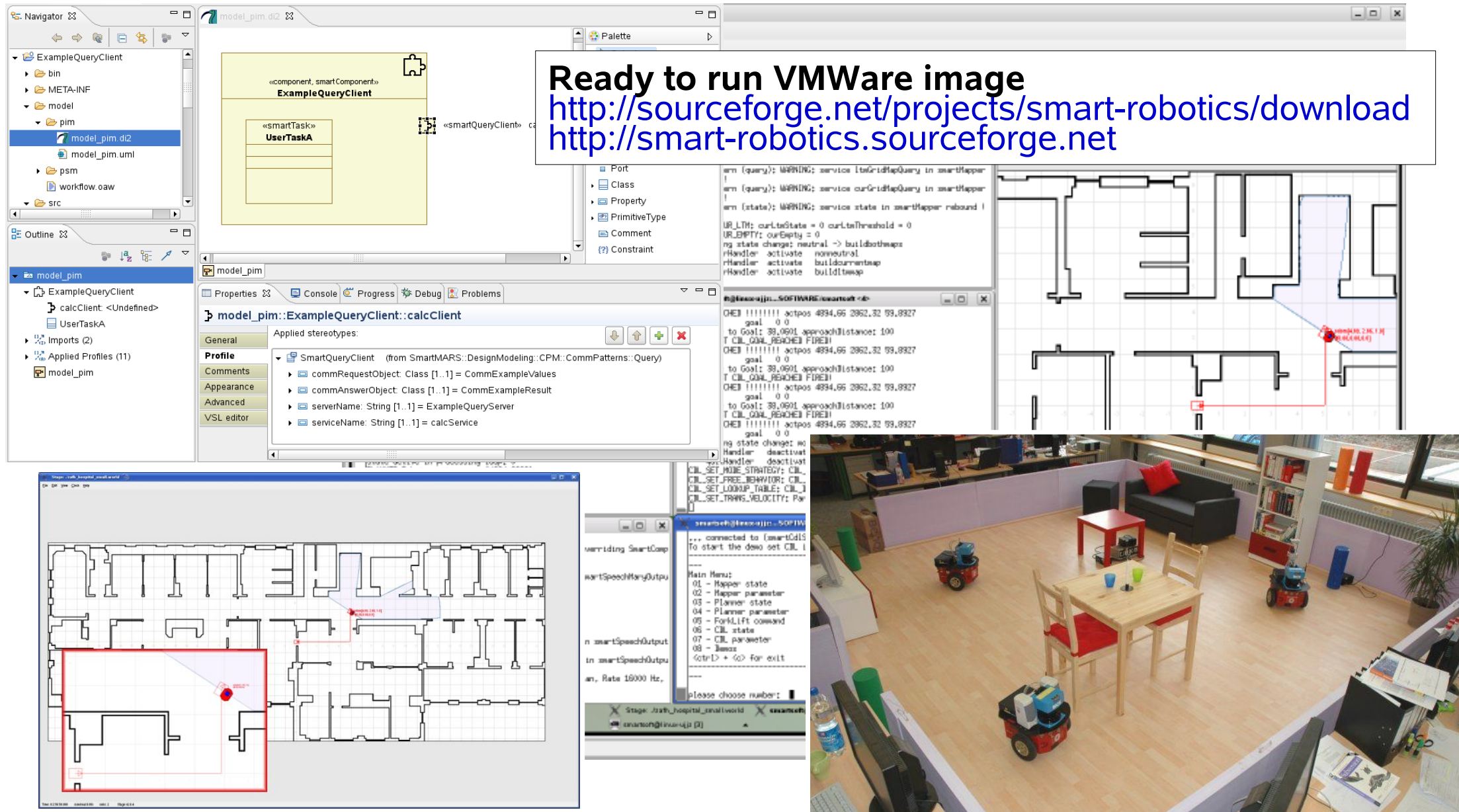
Model Driven Software Development Zusammenfassung und Ausblick

- **Toolchain basiert auf openArchitectureWare**
 - <http://www.openarchitectureware.org/>
 - UML-Tool: PapyrusUML
 - in Eclipse integriert
- **MDSD Toolchain**
 - PIM: SmartMARS Robotik Profil (Modeling and Analysis of Robotics Systems)
 - PSM: SmartSoft mit verschiedenen Technologien umgesetzt, jedoch mit derselben Semantik!
 - kann an weitere PSM's angepasst werden
- **SmartSoft [LGPL]**
 - <http://smart-robotics.sourceforge.net/>
 - CORBA (ACE/TAO) basierte SmartSoft Version
 - frei auf Sourceforge verfügbar
 - ACE (ohne CORBA) basierte SmartSoft Version
 - demnächst auf Sourceforge verfügbar [Linux, Windows, ...]
 - oAW Toolchain für SmartSoft
 - frei auf Sourceforge verfügbar



Model Driven Software Development Zusammenfassung und Ausblick

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



The screenshot illustrates the SmartMARS IDE, a tool for model-driven software development for service robotics. The interface is divided into several panes:

- Navigator:** Shows the project structure, including the ExampleQueryClient component and its associated files (bin, META-INF, model, psm, workflow.oaw, SRC).
- Outline:** Displays the model elements, such as the ExampleQueryClient component and its associated classes (calcClient, UserTaskA).
- Properties:** Shows the applied stereotypes and properties for the selected element (SmartQueryClient). The properties include:
 - commRequestObject: Class [1..1] = CommExampleValues
 - commAnswerObject: Class [1..1] = CommExampleResult
 - serverName: String [1..1] = ExampleQueryServer
 - serviceName: String [1..1] = calcService
- Console:** Displays the execution log, including the state changes and the robot's position. The log shows the robot's state (neutral) and its position (x=4934.66, y=2962.32, z=99.89327).
- 3D Rendering:** Shows a 3D rendering of the robot in a simulated environment, including a table, chairs, and a bookshelf.