

# Achieving Separation of Roles and Separation of Concerns in Robotics Software by Model-Driven Software Development

euRobotics Forum 2012, Odense, Denmark

Alex Lotz, Andreas Steck and Christian Schlegel

*Computer Science Department  
University of Applied Sciences Ulm, Germany*

<http://www.hs-ulm.de/lotz>

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

<http://www.zafh-servicerobotik.de/>

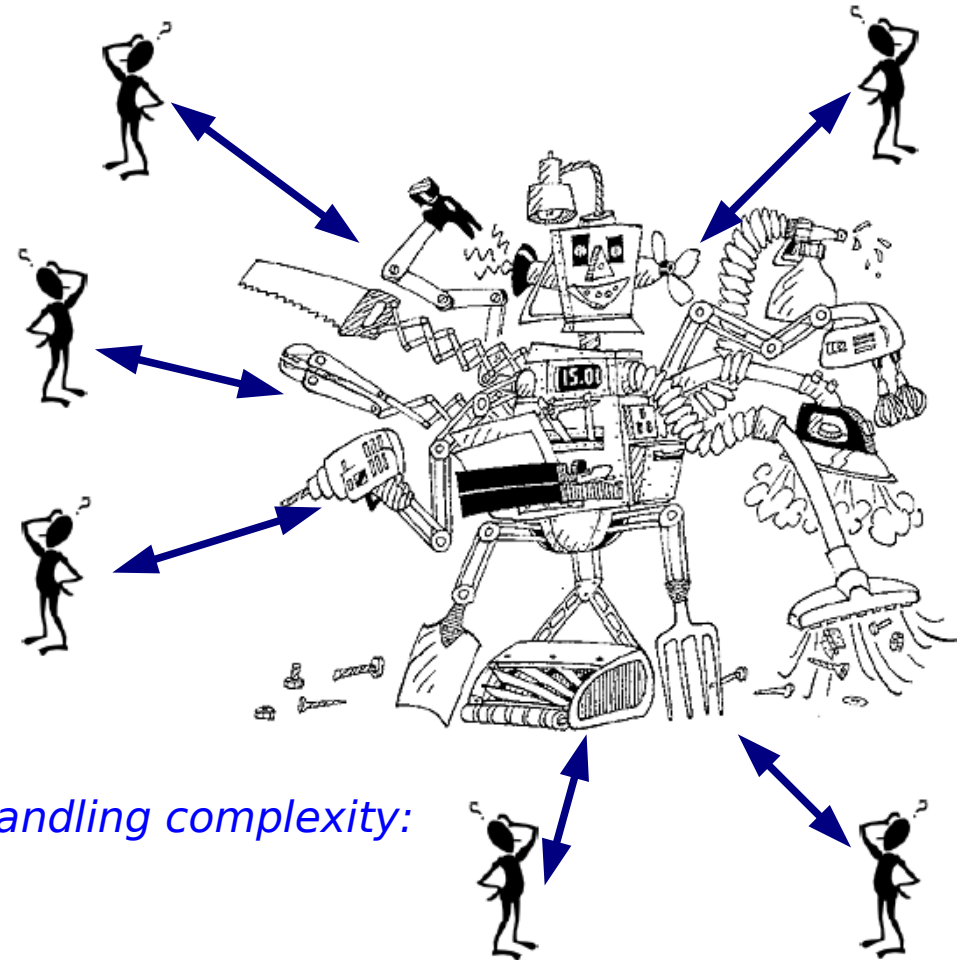


# What is the Challenge in Robotics?

- The current situation in software for robotics can be compared with the early times of the *World Wide Web* where one had to be a computer engineer to setup web pages.
- The *World Wide Web* turned into a universal medium only since the availability of tools
  - which have made it accessible to everyone
  - which allow domain experts (like journalists) to provide content without bothering with technical details
  - which ensure sustainability / availability of contents independently of preferred operating systems, browsers etc.

=> *separation of roles and separation of concerns*

=> *this is a universal approach towards successfully handling complexity: applications, markets, sharing efforts / risks*



# What is the Challenge in Robotics?

- The current situation in software for robotics can be compared with the early times of the *World Wide Web* where one had to be a computer engineer to setup web pages.
- The *World Wide Web* turned into a universal medium only since the availability of tools
  - which have made it accessible to everyone
  - which allow domain experts (like journalists) to provide content without bothering with technical details
  - which ensure sustainability / availability of contents independently of preferred operating systems, browsers etc.

=> *separation of roles and separation of concerns*

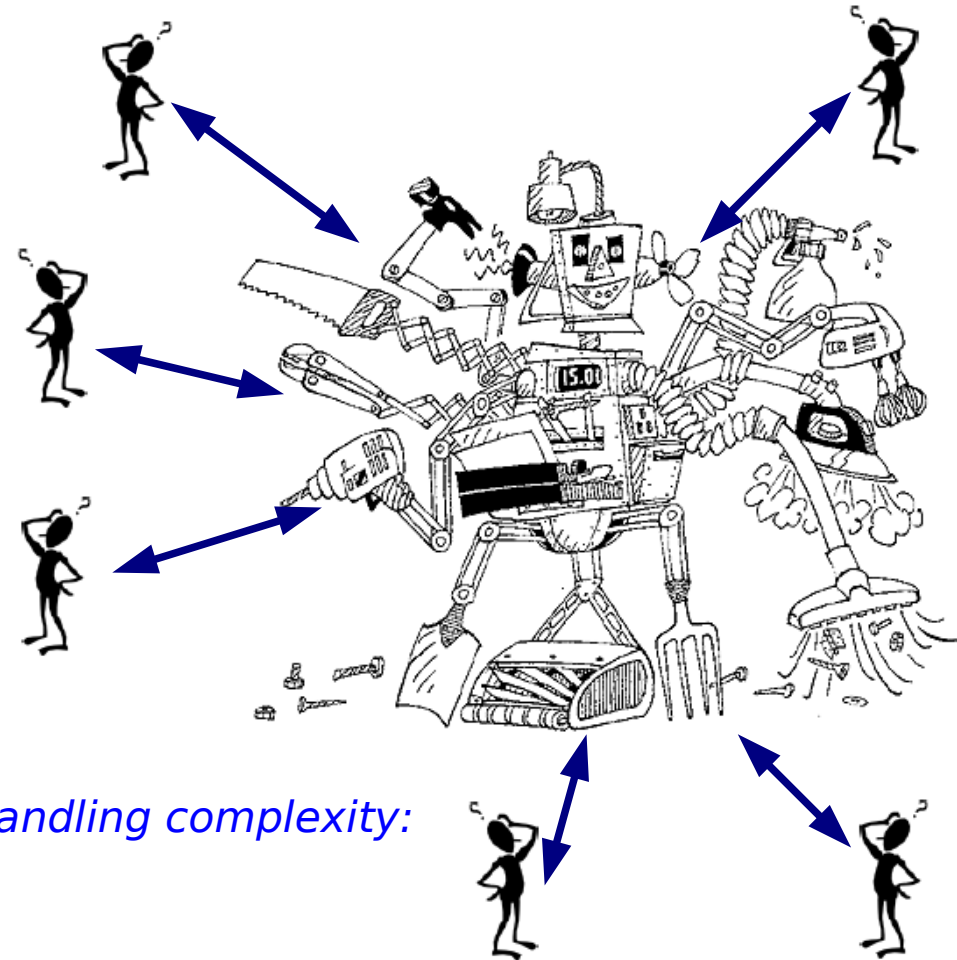
=> *this is a universal approach towards successfully handling complexity: applications, markets, sharing efforts / risks*

*separation of concerns*

=> *e.g. model-based approaches like MDSD to explicate structures / properties*

*separation of roles*

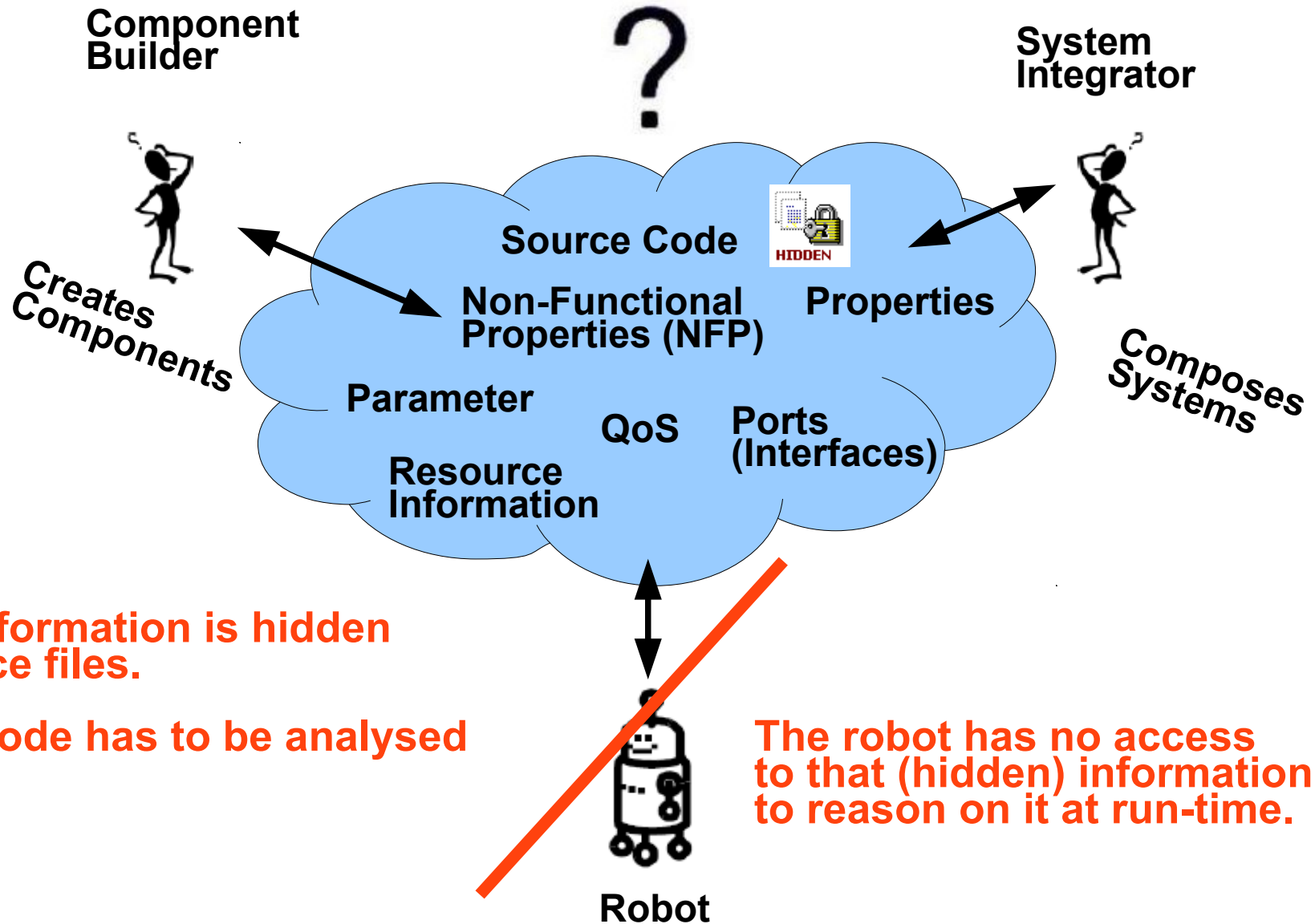
=> *e.g. DSLs to allow non-roboticists to use robotics technology*





# Separation of Roles

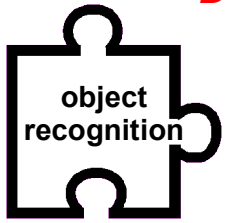
## What is the problem?



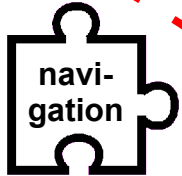


# Separation of Roles Separation of Concerns

*Component  
Builder*



**„freedom from choice“  
in order to ensure  
system-level conformity**

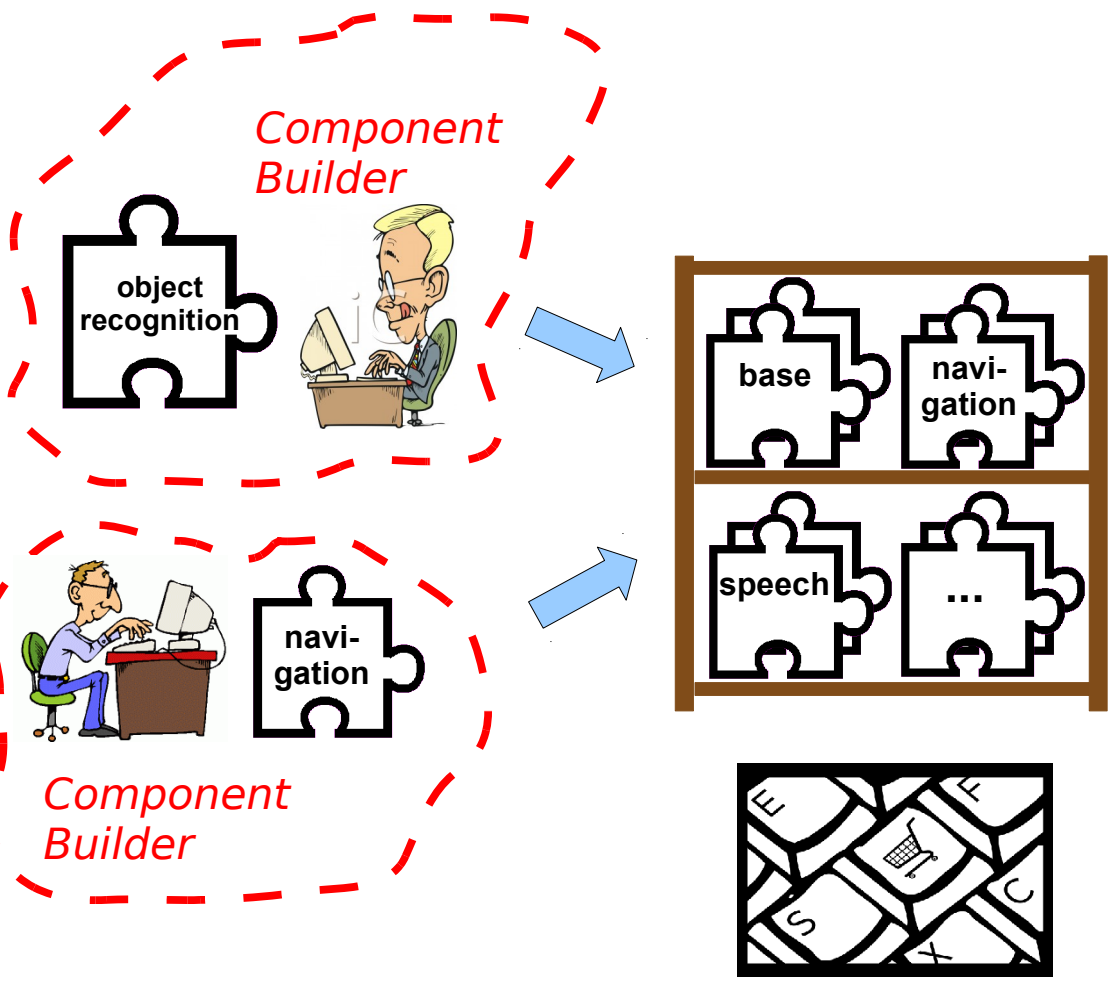


*Component  
Builder*

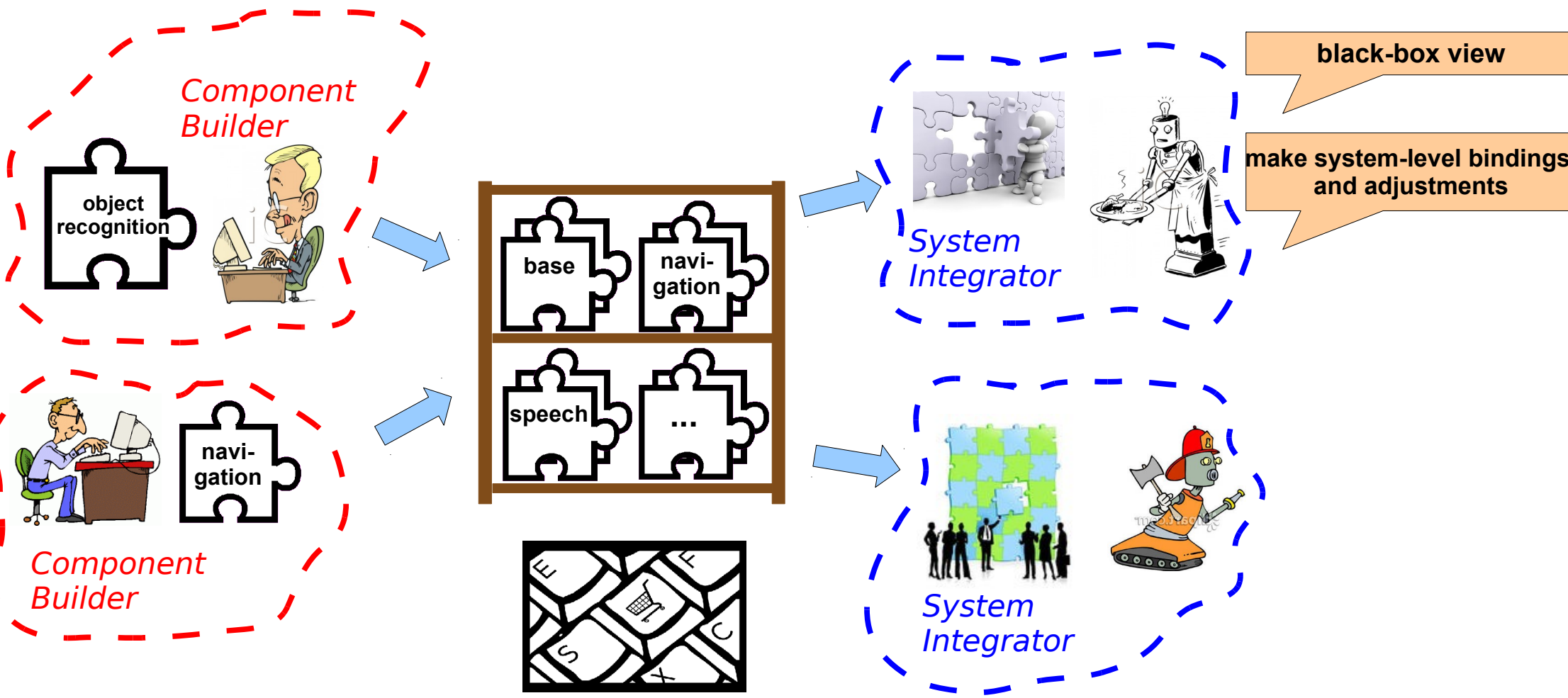


# Separation of Roles

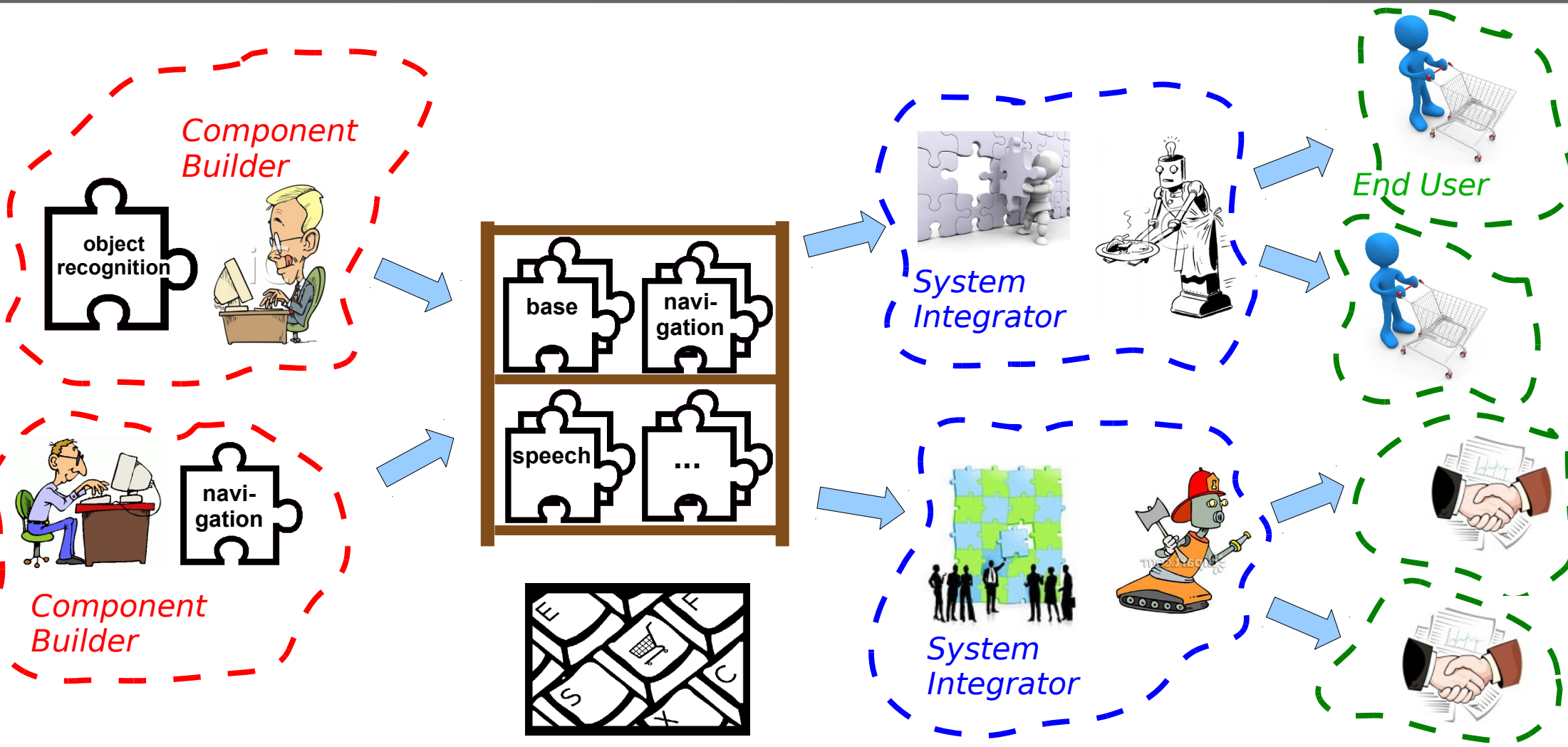
# Separation of Concerns



# Separation of Roles Separation of Concerns



# Separation of Roles Separation of Concerns



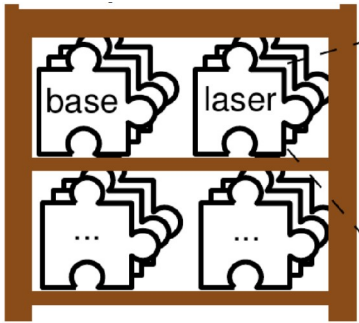




# Separation of Roles

**How it works? → Use Component Models!**

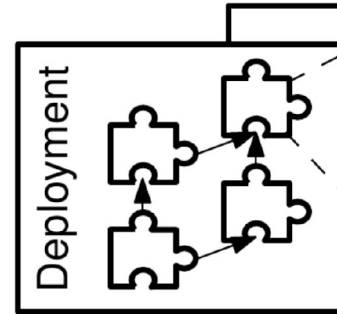
Component Shelf



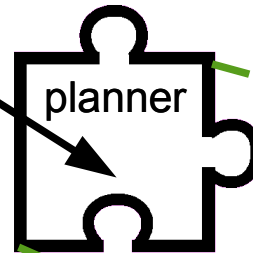
Component Builder



System Integrator



inner view



outer view

Parameters, Properties,  
Ports (Interfaces), Resource  
Information, ...

**ARE EXPLICATED IN THE MODELS**

**AND ARE NOT HIDDEN IN THE SOURCE FILES**

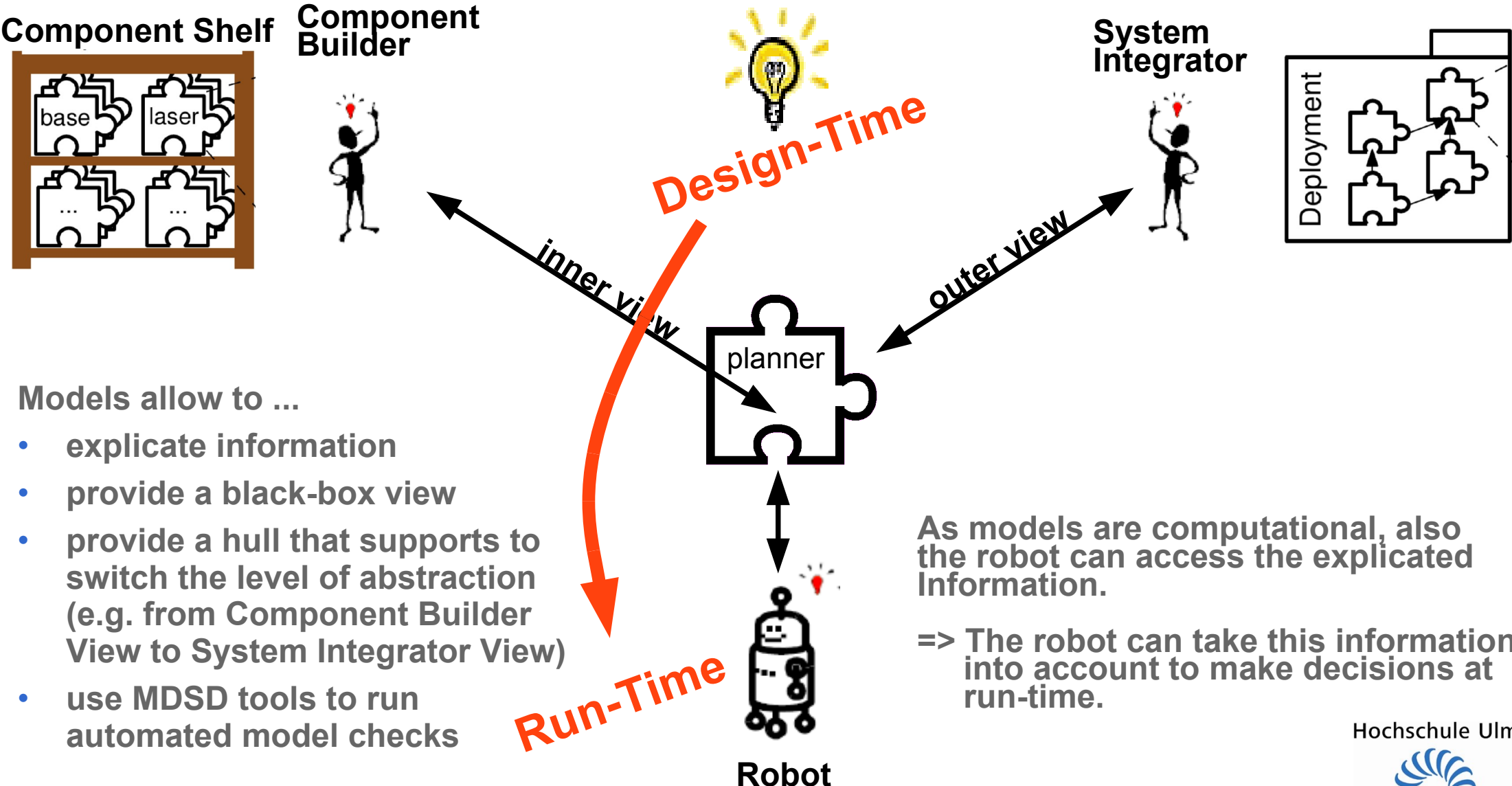
```

Name: planner
Ports:
  mapClient      : pushNewestClient<gridmap>
  goalServer     : pushNewestServer<goal>
  plannerEvent   : eventServer<goalStatus>
  state          : stateServer
  param          : parameterServer
...
States:
  active, neutral, ...
    
```



# Separation of Roles

## Bridge between Designtime- and Runtime-Models



Models allow to ...

- explicate information
- provide a black-box view
- provide a hull that supports to switch the level of abstraction (e.g. from Component Builder View to System Integrator View)
- use MDSD tools to run automated model checks

As models are computational, also the robot can access the explicated Information.

=> The robot can take this information into account to make decisions at run-time.



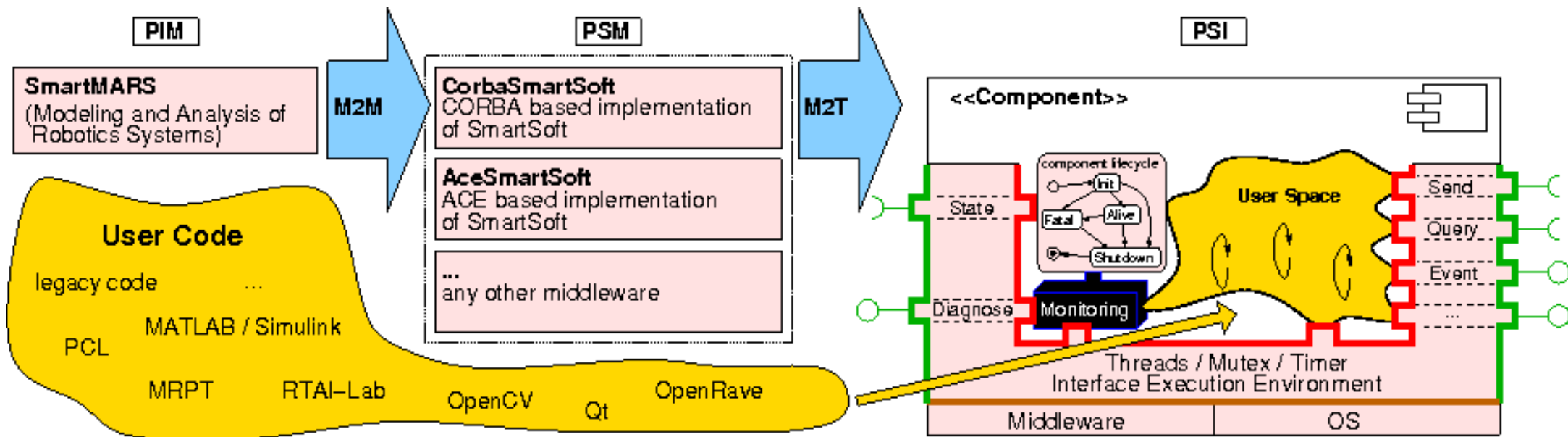


# Model-Driven Approach

## SmartMDSD

### Illustration of the Development Process

- Implemented as UML 2.0-Profile for Robotics Software Components
- supports Component Development, System Integration, Deployment
- based on standards: UML 2.0, Papyrus, Eclipse Modeling Project, etc.
- different Runtime-Platforms, Middleware-Systems etc.

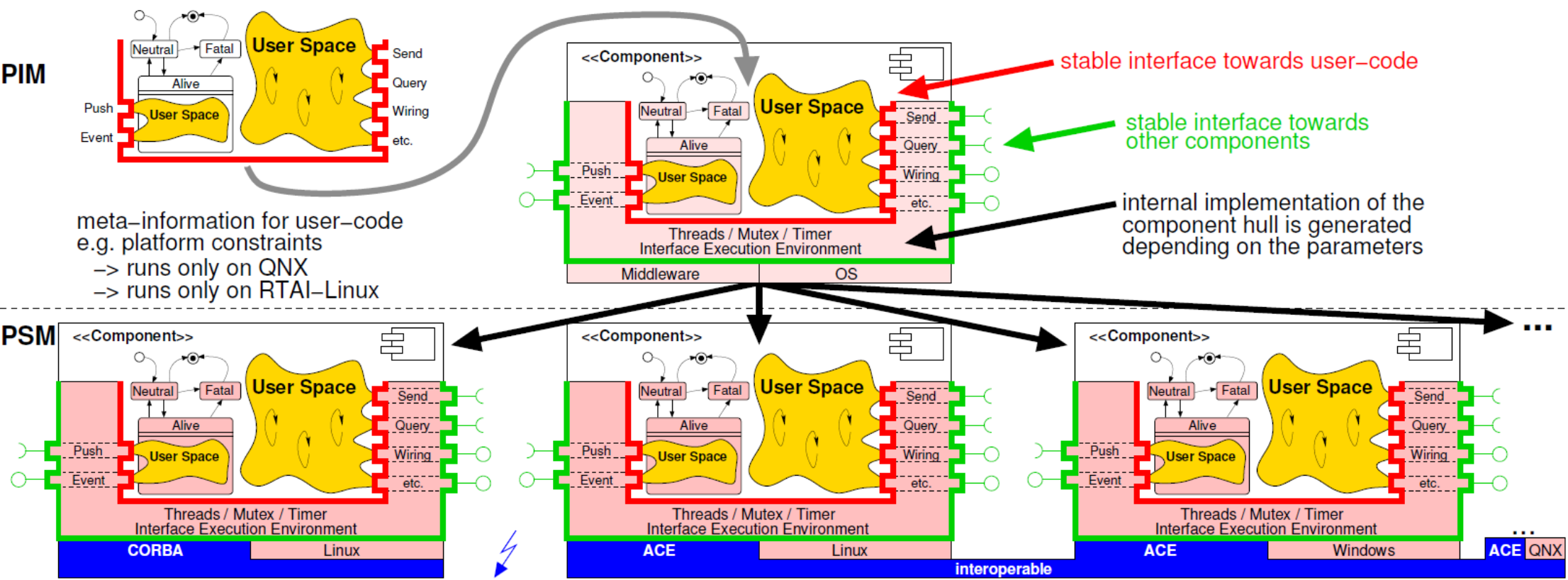


2-step transformation workflow (framework builder view)



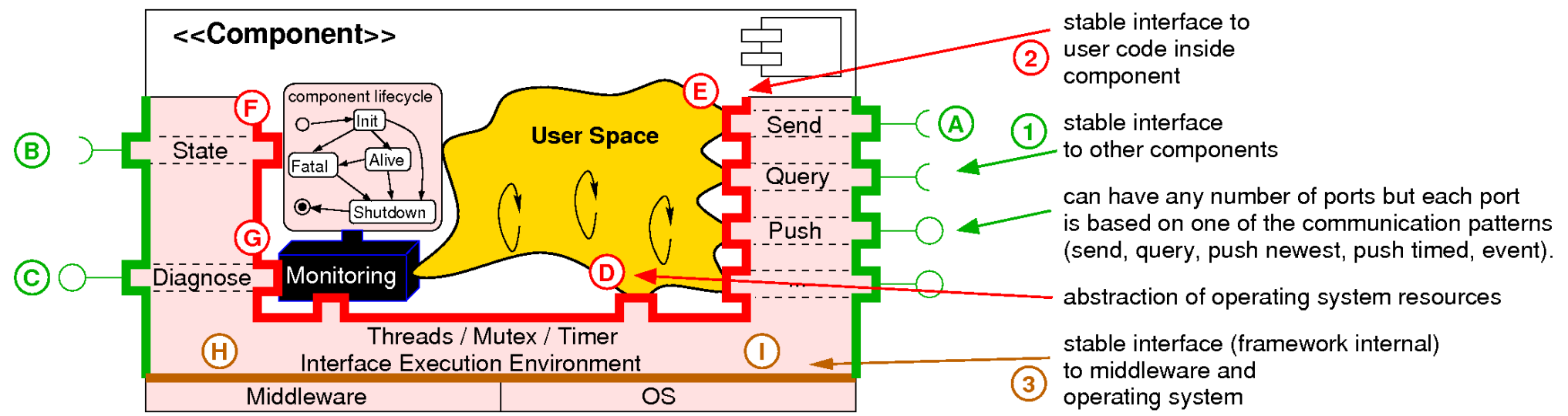
# The SmartSoft Component Model

## Mapping to different Middlewares



# The SmartSoft Component Model

## Decoupled Sphere of Influence



- Services are defined by a *Communication Pattern* and *Communication Objects*
- *Communication Objects* are communicated between components: platform-independent, by-value
- Services are offered / used by components via *Ports*

### The SmartSoft Communication 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 |

### The SmartSoft Services

|   |  |
|---|--|
| param   | component configuration                |
| state   | activate/deactivate component services |
| wiring  | dynamic component wiring               |
| diagnose  | introspection of components            |
| <i>(internally based on communication patterns)</i> |  |



# Model-Driven Approach

## Component Builder View

The screenshot displays the Papyrus IDE interface for a SmartFaceRecognition project. The main workspace shows a PIM Graphical Representation of the SmartFaceRecognition component, which includes several sub-components like VisualizationThread, ParameterHandler, FaceRecognitionEventTest, and StateChangeHandler. A dashed line highlights the imageNewestClient interface, which is detailed in the Properties panel below. The Properties panel shows the SmartFaceRecognition\_pim::SmartFaceRecognition::imageNewestClient interface with applied stereotypes and tagged values for serverName, serviceName, and commObject.

**Button** (Toolbar)

**PIM Files** (Navigator)

- SmartFaceRecognition
  - model
    - SmartFaceRecognition\_pim.di2
    - SmartFaceRecognition\_pim.uml
  - psm
  - src
    - gen
    - obj
      - CompHandler.cc
      - CompHandler.hh
      - DefaultStateChangeHandler.cc
      - DefaultStateChangeHandler.hh

**PSI Files** (Navigator)

**PIM outline** (Outline)

- SmartFaceRecognition\_pim
  - SmartFaceRecognition
    - imageNewestClient: <Undefined>
    - stateServer: <Undefined>
    - paramServer: <Undefined>
    - faceRecogEventServer: <Undefined>
    - VisualizationThread

**PIM Graphical Representation**

**Palette**

- Select
- Marquee
- UML Links
- UML Elements
- SmartSoft Deployment
- SmartSoft Component
- SmartTask
- SmartMainState
- SmartMutex
- SmartTimer
- SmartIniParameterGroup
- SmartSendClient
- SmartPushNewestClient
- SmartPushTimedClient

**Attributes / Tagged Values**

SmartFaceRecognition\_pim::SmartFaceRecognition::imageNewestClient

Applied stereotypes:

- SmartPushNewestClient (from SmartMARS)
  - serverName: String [1..1] = SmartUnicapImageServer
  - serviceName: String [1..1] = imageNewest
  - commObject: Class [1..1] = CommVideoImage

# Model-Driven Approach

## Screencast "Simple Navigation"

Papyrus - DeploySimpleNavigation/model/DeploySimpleNavigation.di2 - itemis openArchitectureWare distribution

File Edit View Navigate Search Project Run Window Help

Navigator

- CommBasicObjects [trunk/smartssoft/src/InterfaceClasses/Co...
- DeploySimpleNavigation
  - META-INF
  - model
    - DeploySimpleNavigation.di2
    - DeploySimpleNavigation.uml
    - .classpath
    - build.properties
- SmartLaserLMS200Server [trunk/smartssoft/src/components/S...
- SmartPioneerBaseServer [trunk/smartssoft/src/components/Sr...
- SmartSimpleNavigation

Palette

- Select
- Marquee
- UML Links
  - Dependency
  - Generalization
  - Association
  - Link
- UML Elements
  - Package
  - Component
  - Artifact
  - Port
- SmartSoft Deployment
  - CorbaNamingService
  - RTAISetup
  - Connection
- SmartSoft Component
  - SmartTask
  - SmartMainState
  - SmartMutex
  - SmartTimer

Deployment Diagram: DeploySimpleNavigation

- CorbaNamingService
- SmartLaserLMS200Server
  - laserServer: <Undefined> [1]
- SmartSimpleNavigation
  - laserClient: <Undefined> [1]
  - navVelClient: <Undefined> [1]
- SmartPioneerBaseServer
  - navigationVelocityServer: <Undefined> [1]

Properties

COMPOUNDIAGRAM\_NODE\_COMPONENT

|            |             |   |
|------------|-------------|---|
| General    | Name:       | SmartSimpleNavigation   |
| Profile    | Visibility: | <input checked="" type="radio"/> public <input type="radio"/> protected <input type="radio"/> private <input type="radio"/> package |
| Comments   | Cl.Behav:   | ClassifierBehavior Unset  |
| Appearance | Abstract:   | <input type="checkbox"/>  |
| Advanced   | Active:     | <input type="checkbox"/>  |
|            | Use Cases:  |   |

Birdview

- BaseStateQueryHandler
- ParameterHandler
- BatteryEventHandler
- import SmartSimpleNavigation
  - SmartSimpleNavigation
    - navVelClient: <Undefined>
    - laserClient: <Undefined>
  - NavigationTask
- Applied Profiles (3)
- DeploySimpleNavigation

<http://youtu.be/04SqzrS6Udw>





# Model-Driven Approach

## System Integrator View

Papyrus - DeployNavTask/model/DeployNavTask.di2 - itemis openArchitectureWare distribution

File Edit View Navigate Search Project Run Window Help

Button

**Deployment Model**

- DeployNavTask.di2
- DeployNavTask.uml

**Imported Components**

- import SmartCdlServer
- SmartCdlServer
- import SmartMapperGridMap
- SmartMapperGridMap
- import SmartPioneerBaseServer
- SmartPioneerBaseServer
- import SmartPlannerBreadthFirstSearch
- SmartPlannerBreadthFirstSearch
- import SmartLaserLMS200Server
- SmartLaserLMS200Server
- import SmartRobotConsole
- SmartRobotConsole
- import SmartAmcl
- SmartAmcl

**Graphical Representation of Deployment Model**

**Palette**

- Select
- Marquee
- UML Links
- UML Elements
- SmartSoft Deployment
- CorbaNamingService
- RTAISetup
- Connection

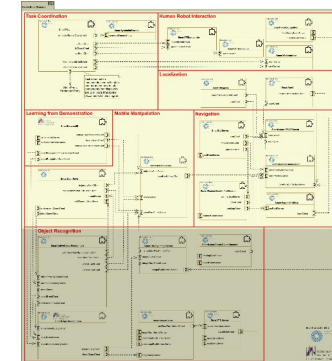
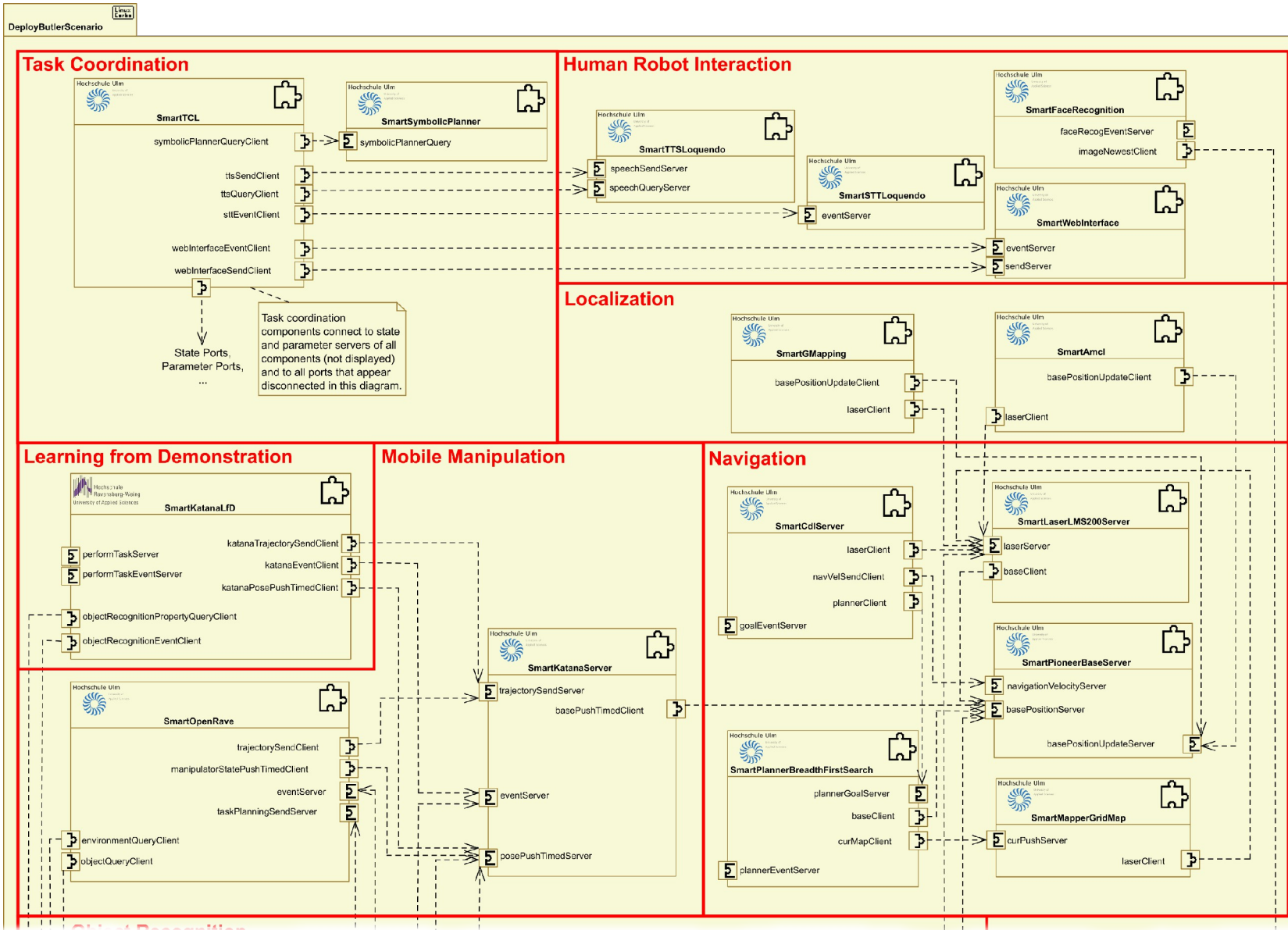
**Deployment Properties**

- ip: String [1..1] = 192.168.31.115
- deployed: DeployType [1..1] = remote
- username: String [1..1] = student
- directory: String [1..1] = tmp/autms



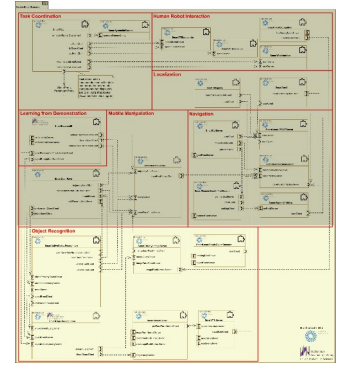
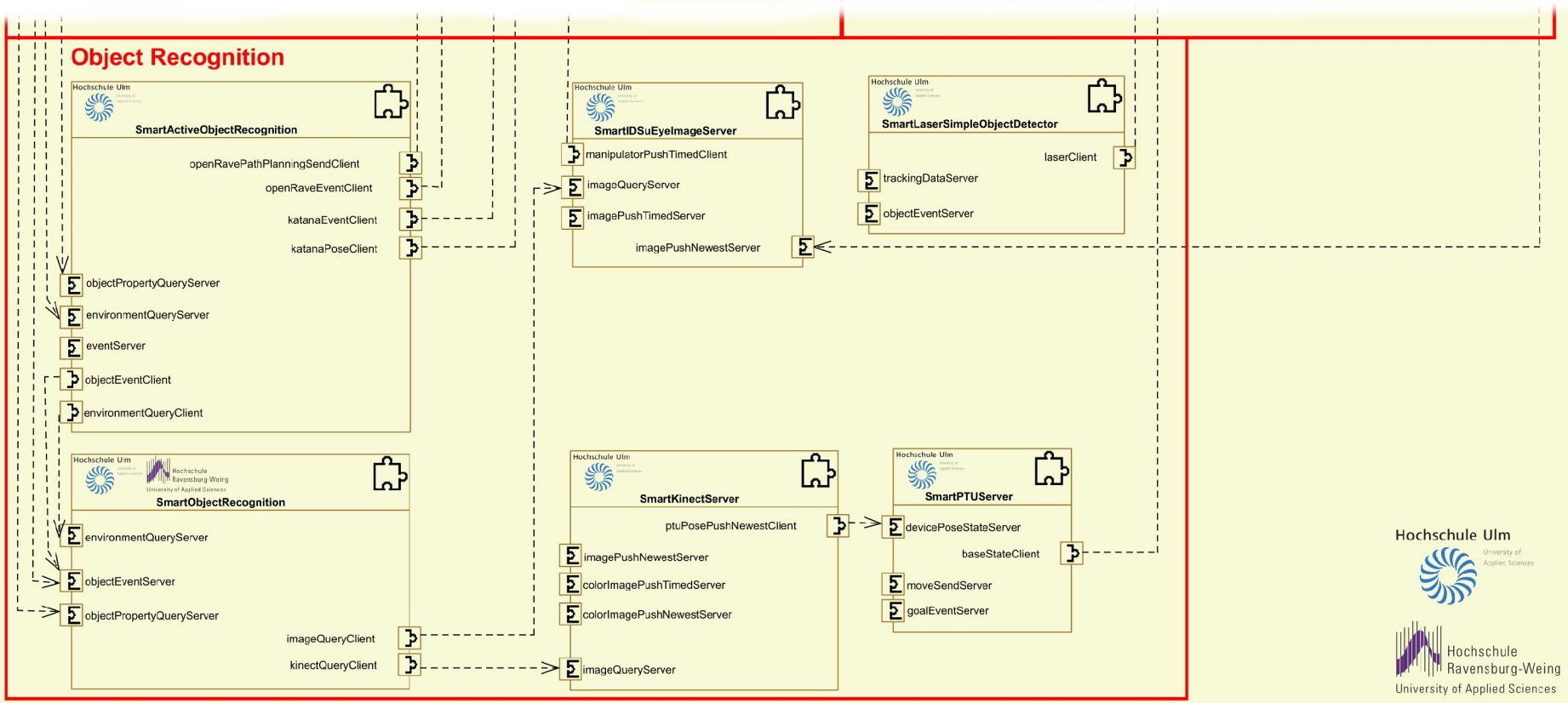


# Model-Driven Approach System Integrator View – Buttler Scenario





# Model-Driven Approach System Integrator View – Buttler Scenario

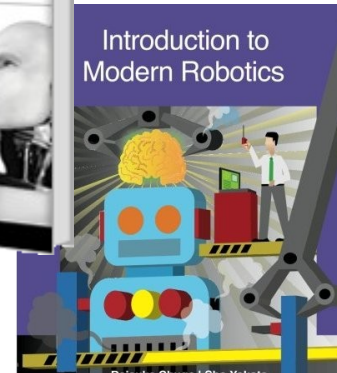
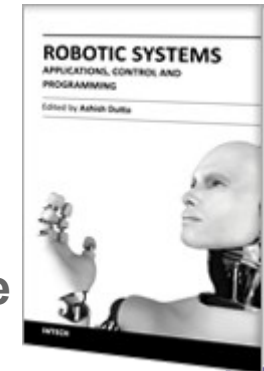




# Further References

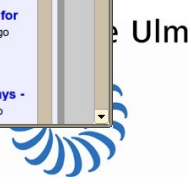
## Recent book chapters (open access PDFs)

- Christian Schlegel, Andreas Steck and Alex Lotz, "Robotic Software Systems: From Code-Driven to Model-Driven Software Development", in *Robotic Systems - Applications, Control and Programming*, ISBN 978-953-307-941-7, InTech, 2012 [Download as PDF](#)
- Christian Schlegel, Andreas Steck, and Alex Lotz. "Model-Driven Software Development in Robotics: Communication Patterns as Key for a Robotics Component Model", in *Introduction to Modern Robotics*, ISBN 978-0980733068, iConcept Press, 2011 [Download as PDF](#)



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

<http://www.youtube.com/user/RoboticsAtHsUlm>





# Selected publications for **Model usage at Run-Time**

- Christian Schlegel, Andreas Steck, Davide Brugali, Alois Knoll. “**Design Abstraction and Processes in Robotics: From Code-Driven to Model-Driven Engineering**”, in *2nd International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN)*, Pages 324-335, Darmstadt, Springer LNAI 6472, ISBN-10 3-642-17318-7, 2010  
**Get as PDF**
- Andreas Steck, Alex Lotz and Christian Schlegel, "**Model-Driven Engineering and Run-Time Model-Usage in Service Robotics**", in *Proc of the 10th ACM international conference on Generative programming and component engineering (GPCE '11)*, Portland, Oregon, USA, October 2011  
**Get as PDF**
- Andreas Steck and Christian Schlegel, "**Towards Quality of Service and Resource Aware Robotic Systems through Model-Driven Software Development**", in *Proc. 1st International Workshop on Domain-Specific Languages and models for ROBotic systems (DSLRob/IROS)*, Taipei, Taiwan, October 2010  
**Get as PDF**

