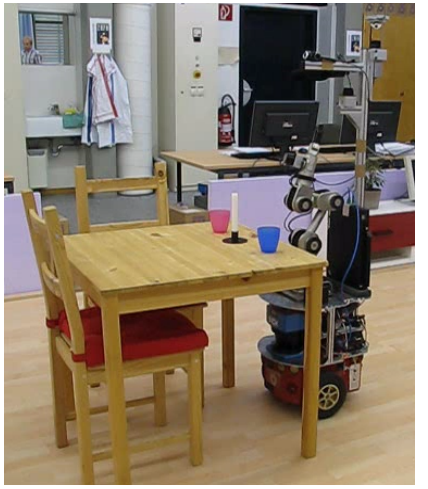
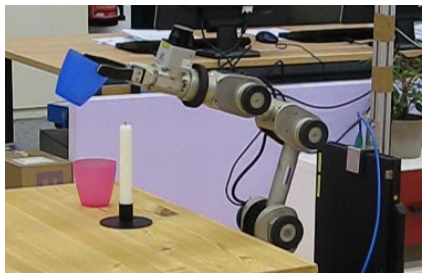
# A Robotics Task Coordination Language: Mastering Execution Variants at Run-Time

***M. Sc. Andreas Steck***
***Prof. Dr. Christian Schlegel***

***Institut für Informatik***
***Hochschule Ulm***
***Germany***

*http://www.hs-ulm.de/schlegel*
*http://www.zafh-servicerobotik.de/ULM/index.php*
*http://smart-robotics.sf.net/*
*http://www.youtube.com/user/roboticsathsulm*

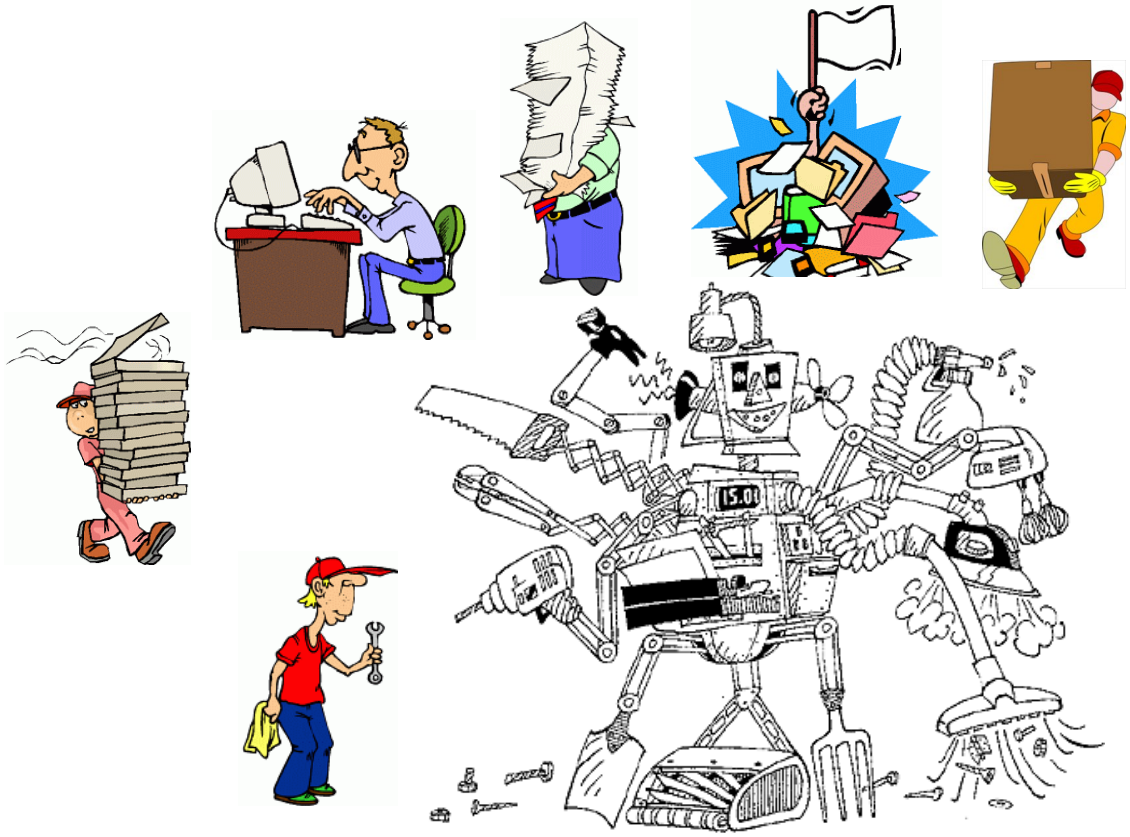




Hochschule Ulm

# What is the challenge in robotics?

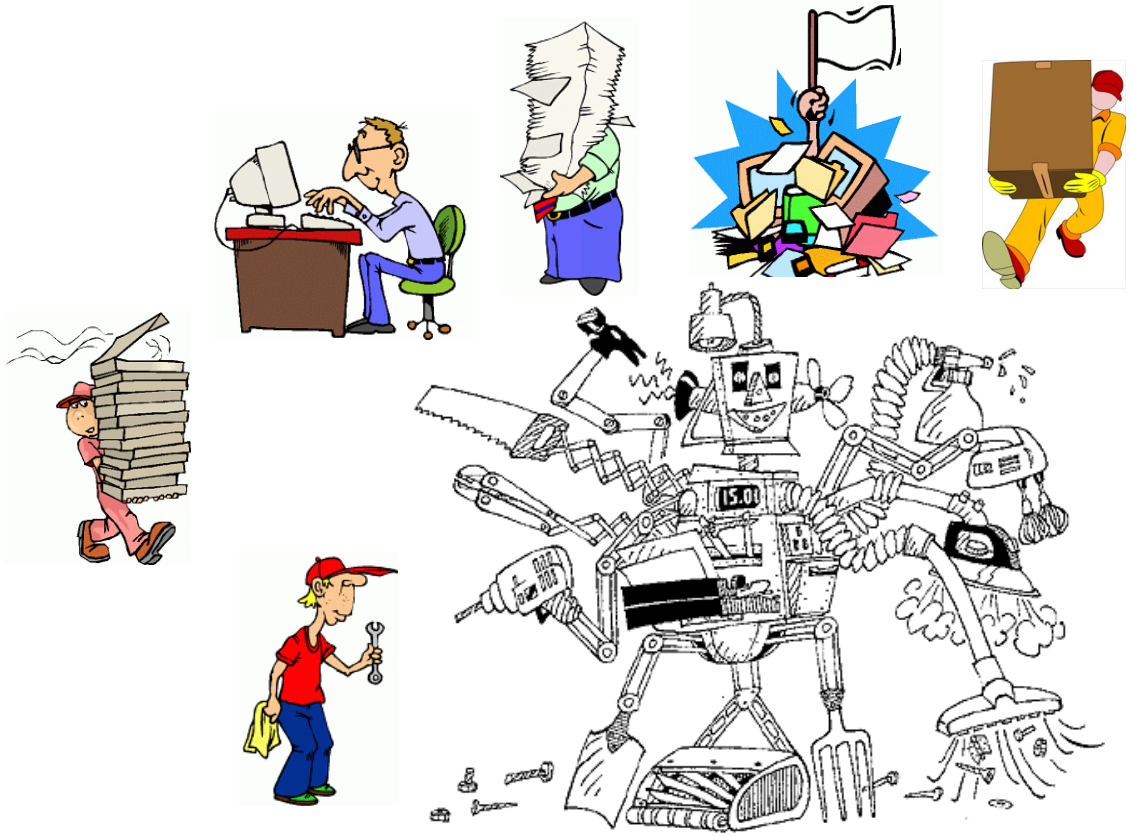
*Management of variants / variability / complexity in advanced robotics at design-time and at run-time*



# What is the challenge in robotics?

*Management of variants / variability / complexity in advanced robotics at design-time and at run-time*

- current situation:
  - *no separation of roles*
    - end users
    - system integrators
    - component builders
    - framework builders
  - *no separation of concerns*
    - computation
    - communication
    - configuration
    - coordination



*... but what are the reasons for this?*
*... how to address the above challenge?*
*... what do we need in robotics?*
*... what is different in robotics compared to e.g. automotive, avionics?*

# 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 is a universal approach towards successful applications and markets*

*=> what does this mean for robotics?*

# 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.

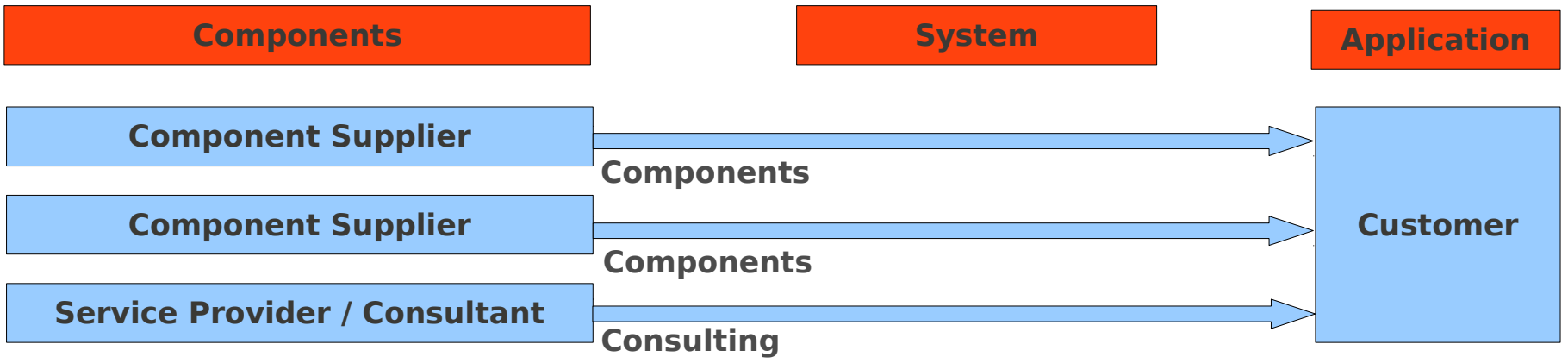
*=> separation of roles and separation of concerns is a universal approach towards successful applications and markets*

*=> what does this mean for robotics?*

*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
# Separation of Concerns

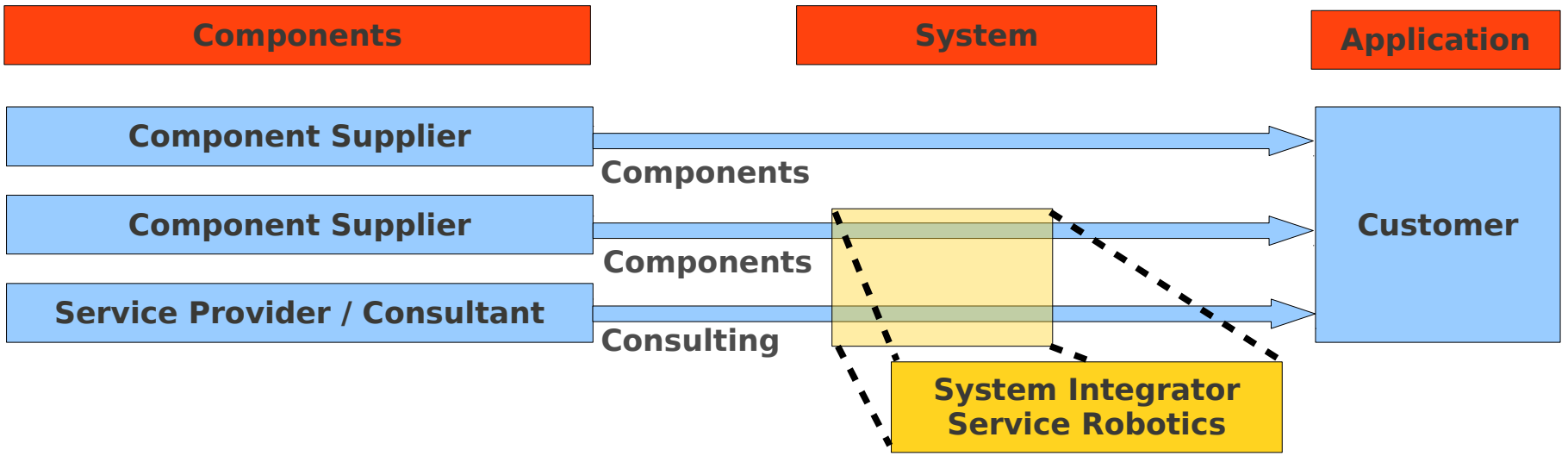
***Integration: unclear allocation of roles in service robotics / advanced robotics***




Hochschule Ulm

# Separation of Roles
# Separation of Concerns

***Integration: unclear allocation of roles in service robotics / advanced robotics***




Hochschule Ulm

# Separation of Roles
# Separation of Concerns

***Integration: unclear allocation of roles in service robotics / advanced robotics***

„freedom from choice“
to achieve
system level conformance

**Components** | **System** | **Application**

**Component Supplier** — Components →
**Component Supplier** — Components →
**Service Provider / Consultant** — Consulting →

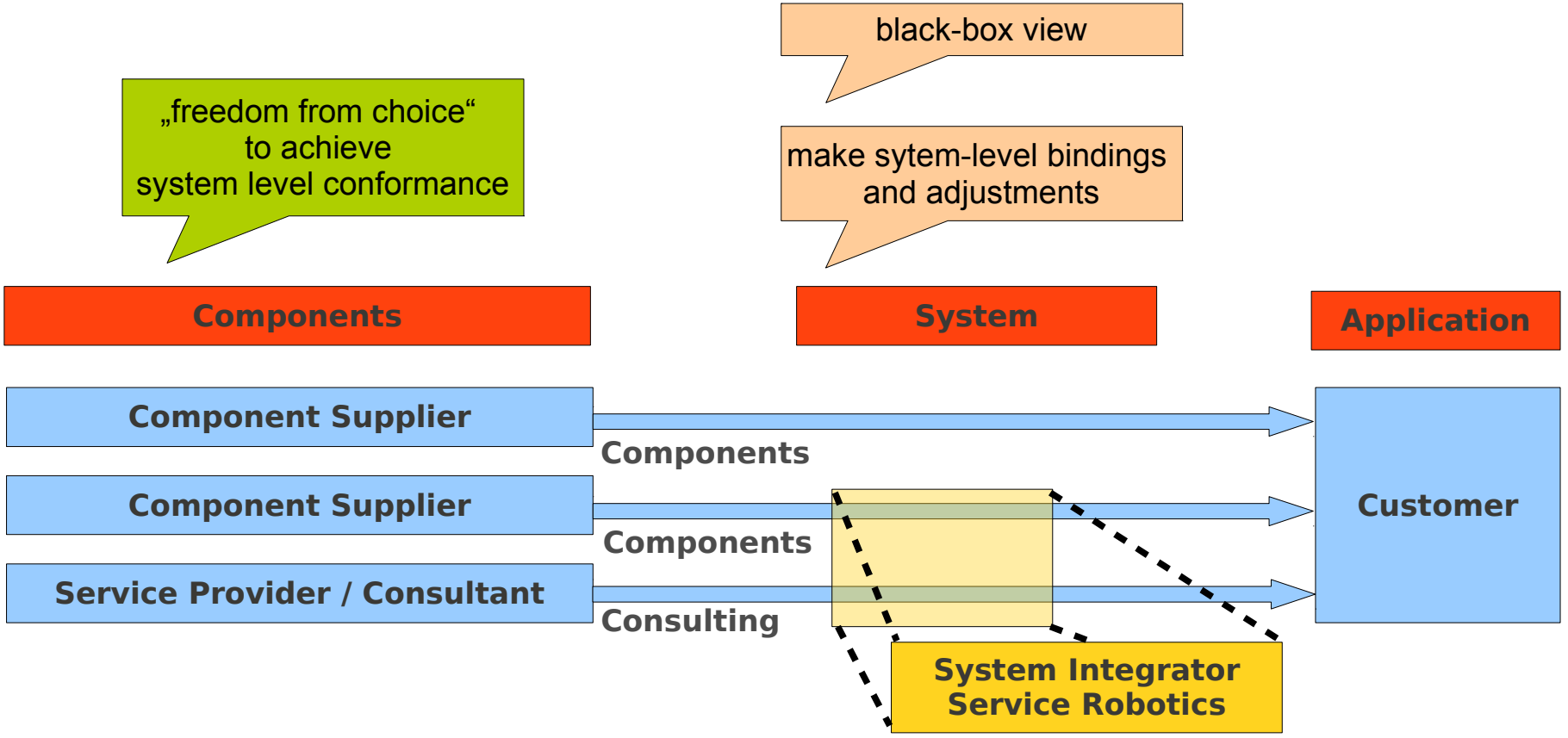
→ **Customer**

**System Integrator Service Robotics**

Hochschule Ulm

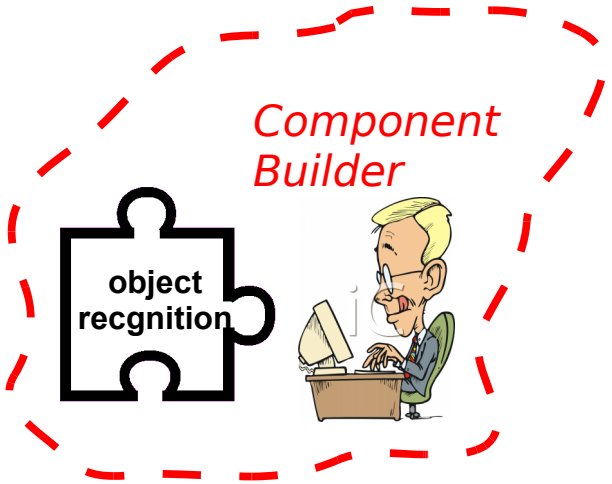
# Separation of Roles
# Separation of Concerns

***Integration: unclear allocation of roles in service robotics / advanced robotics***

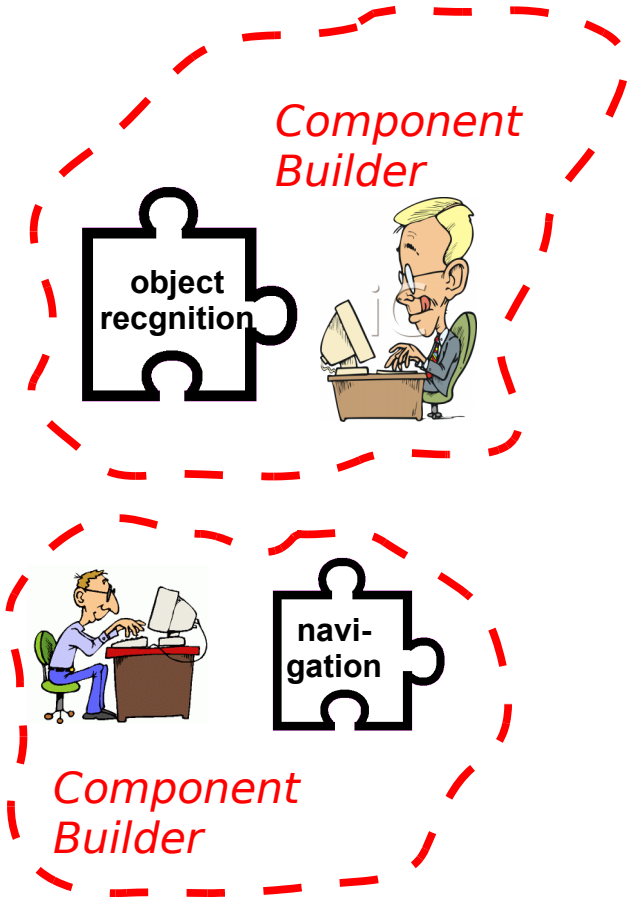



Hochschule Ulm

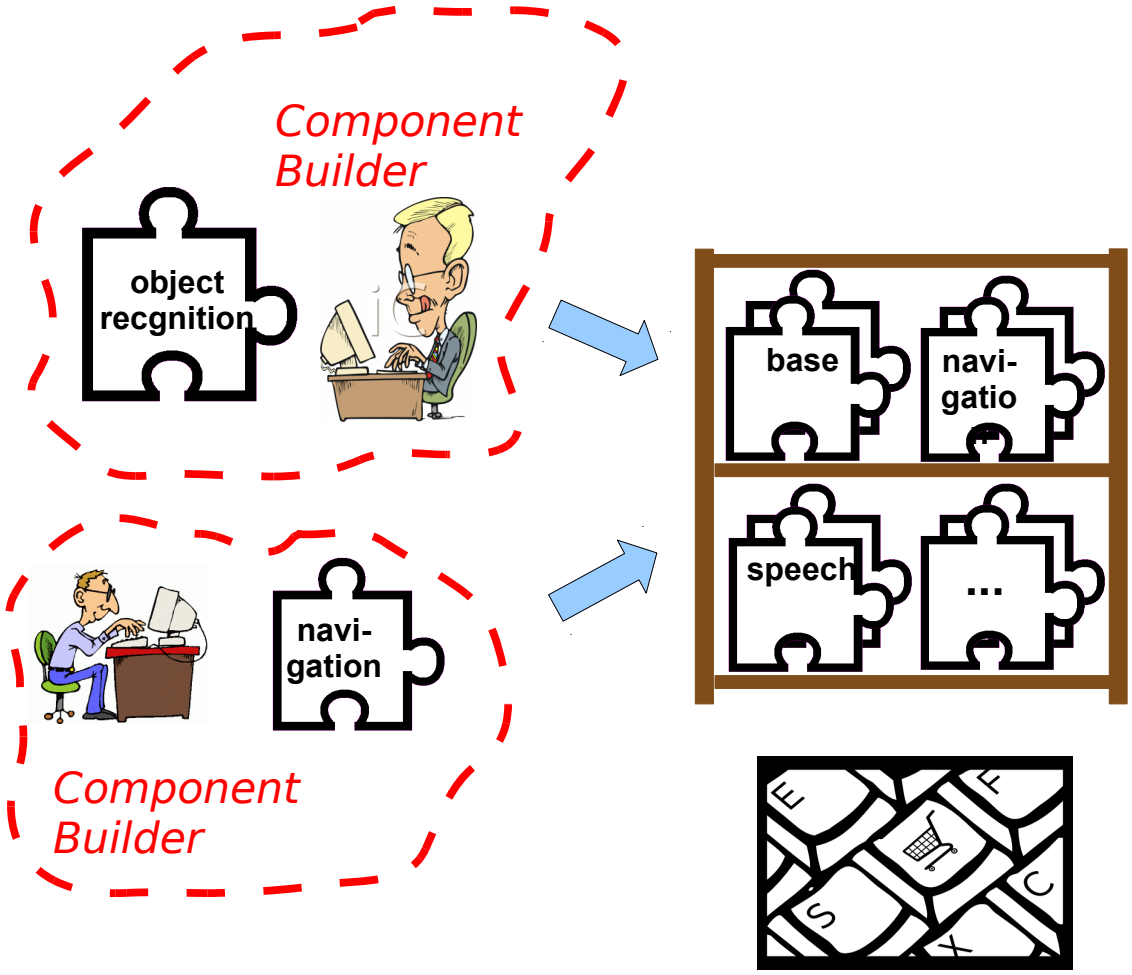
# Separation of Roles
# Separation of Concerns



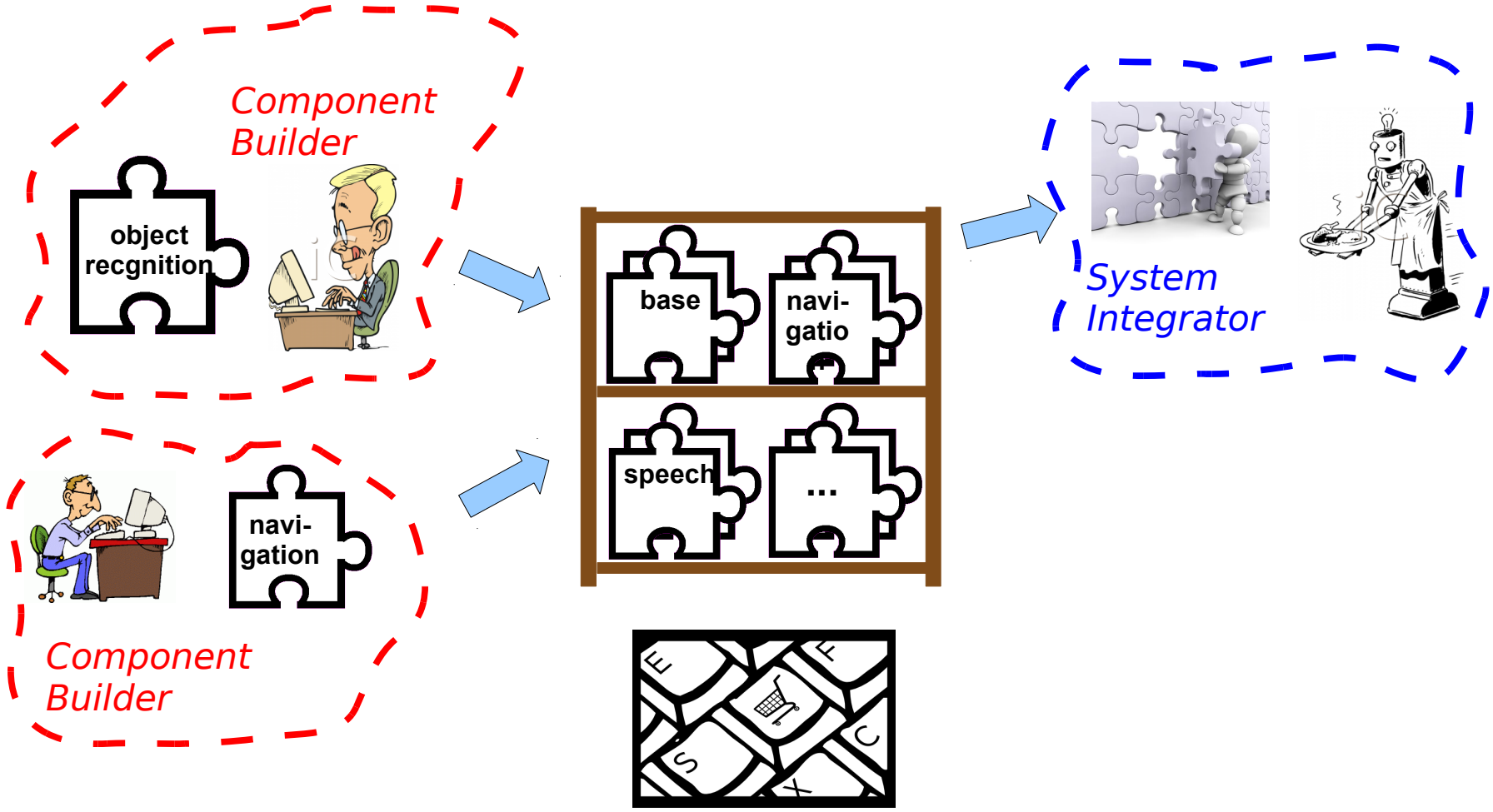
# Separation of Roles
# Separation of Concerns



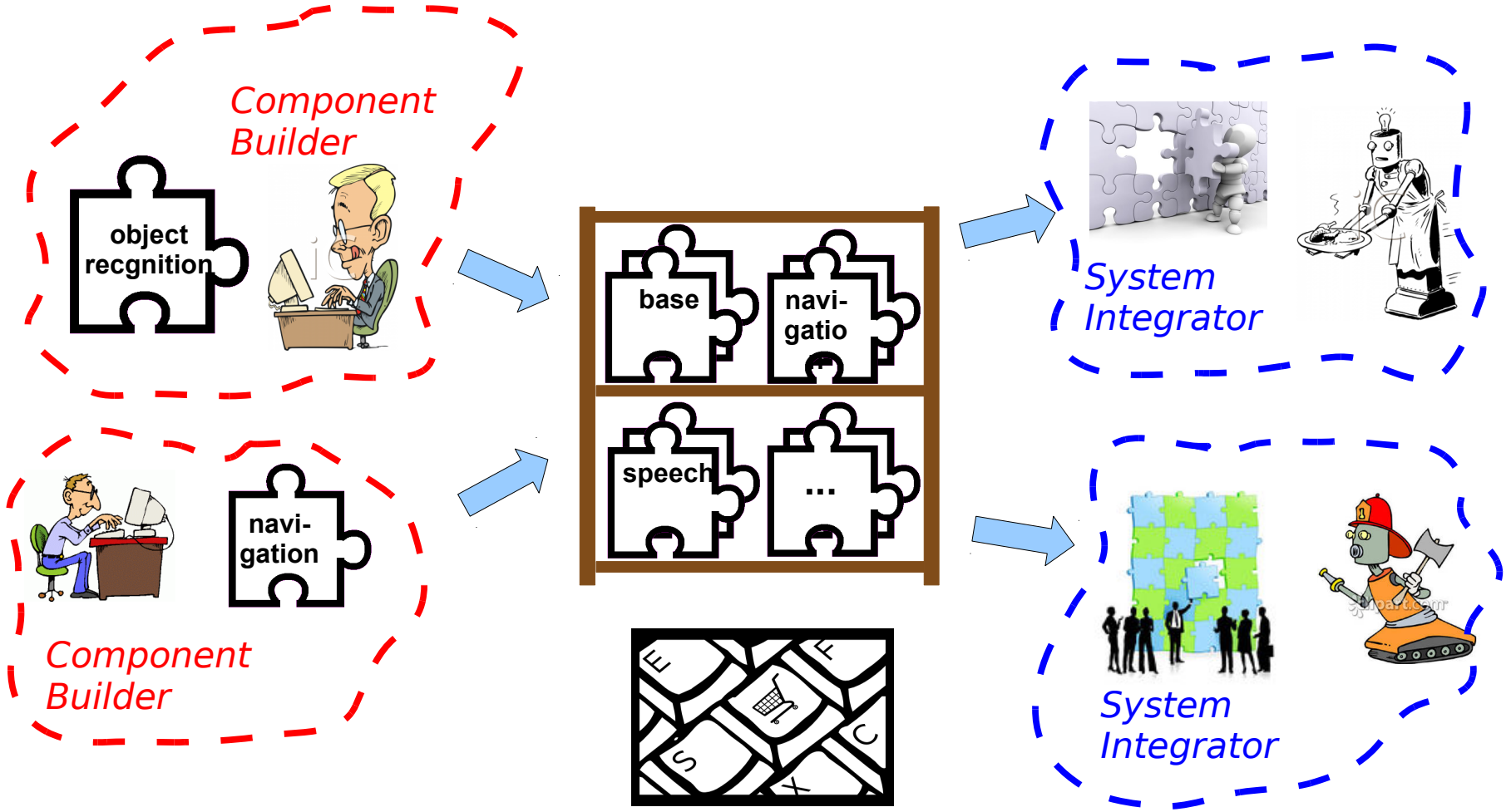
# Separation of Roles
# Separation of Concerns



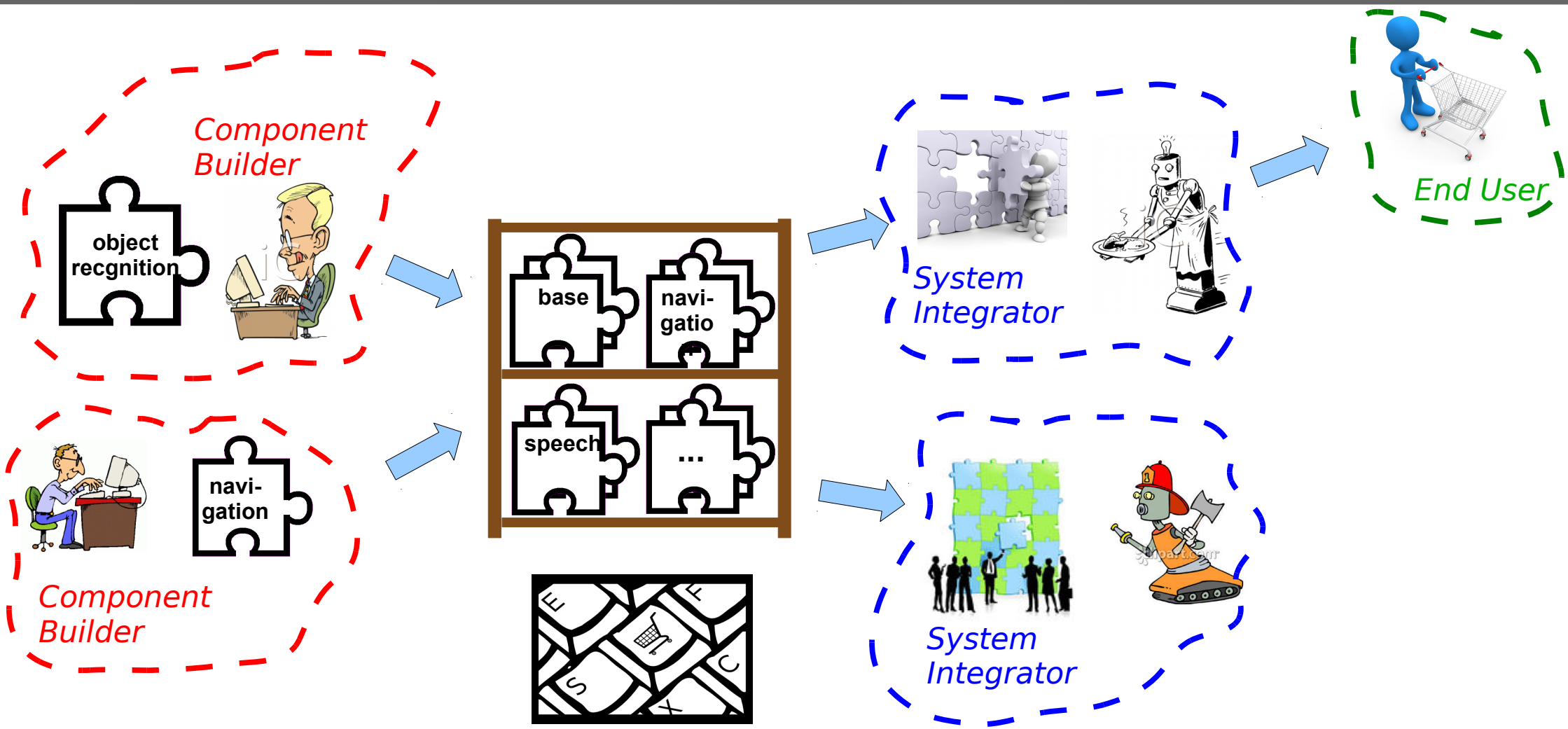
# Separation of Roles
# Separation of Concerns



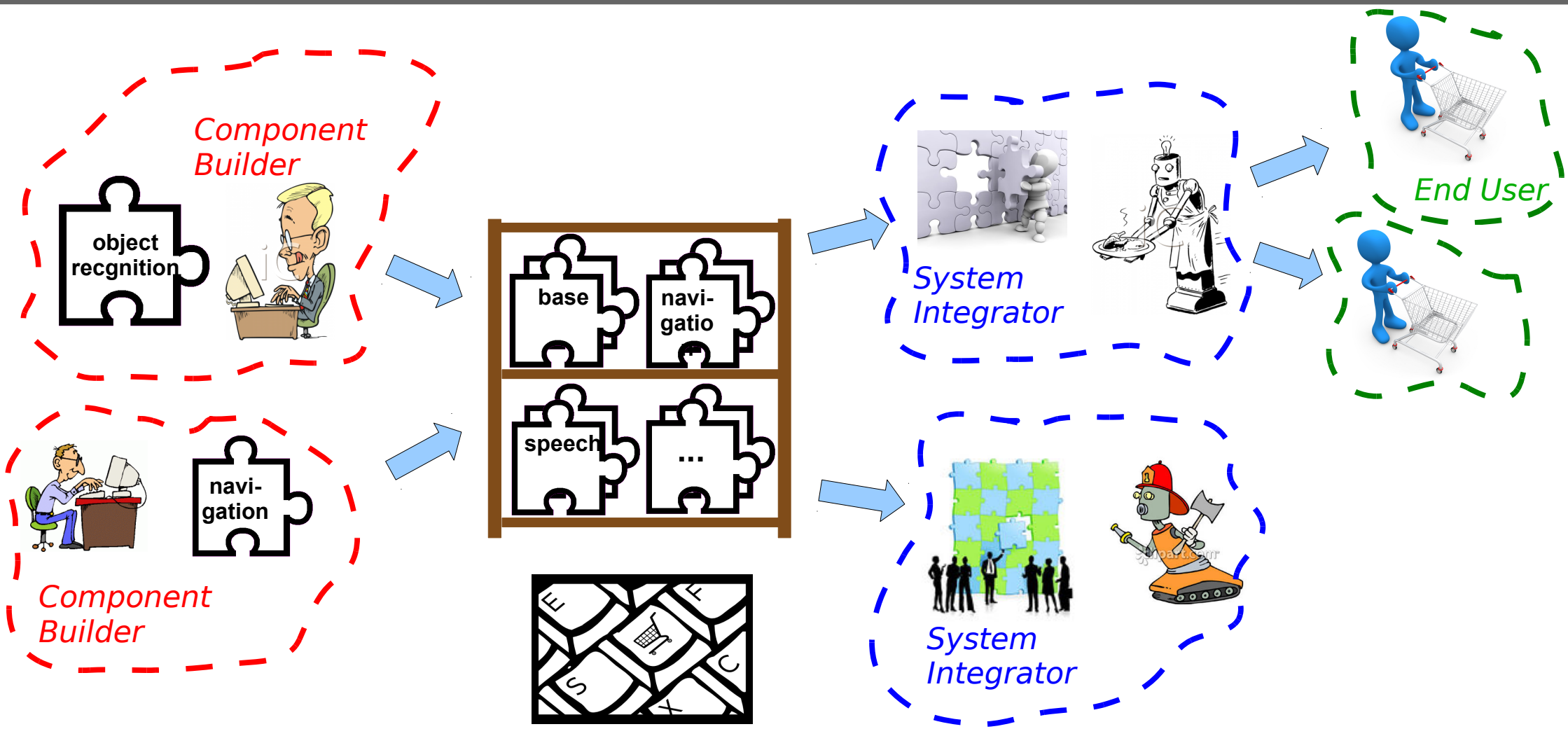
# Separation of Roles
# Separation of Concerns



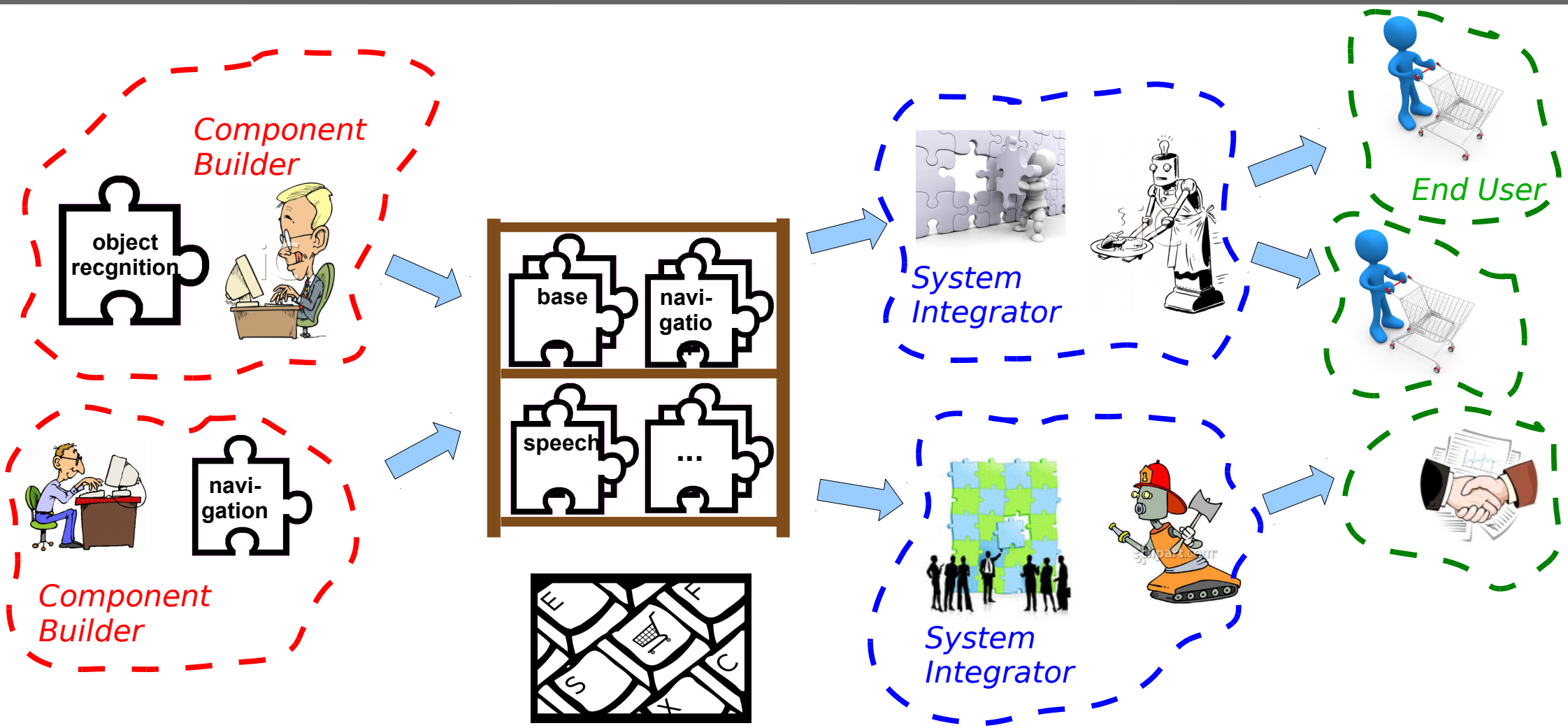
# Separation of Roles
# Separation of Concerns



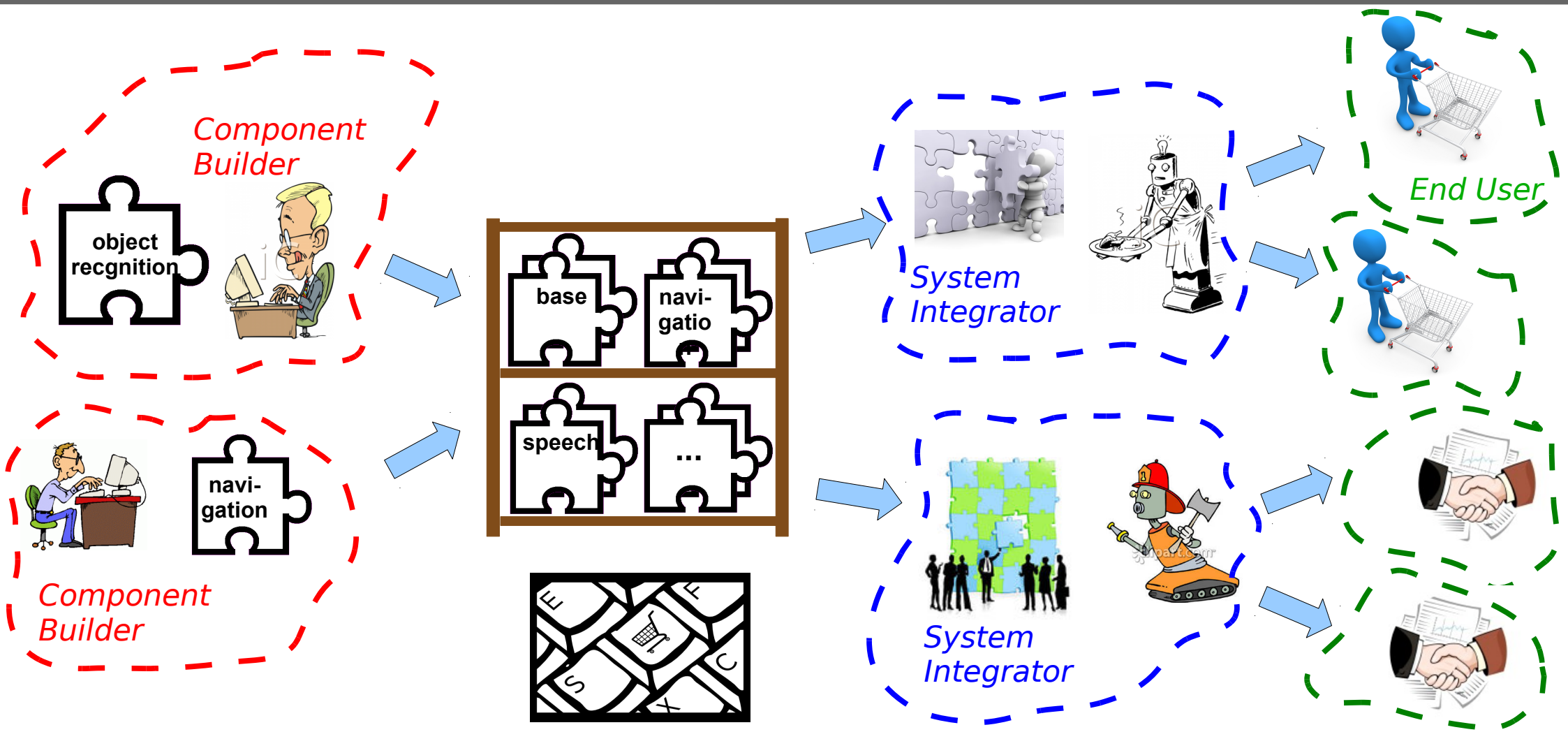
# Separation of Roles
# Separation of Concerns



# Separation of Roles
# Separation of Concerns



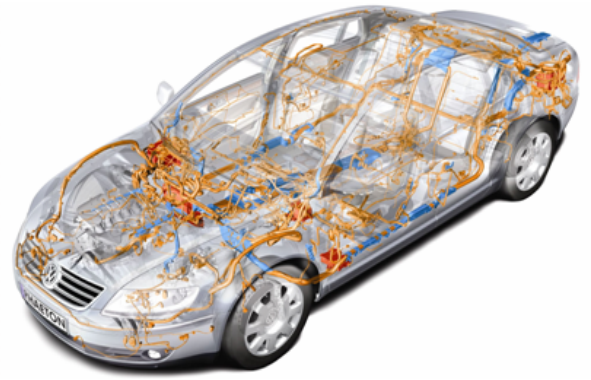
# Separation of Roles
# Separation of Concerns

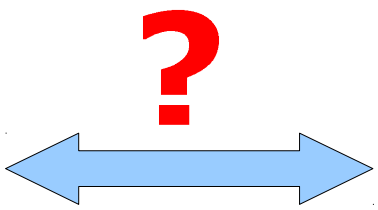


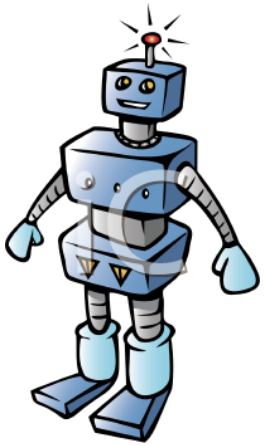
# What is different in robotics?

- The *difference* of robotics compared to other disciplines (e.g. automotive, avionics) is *neither* the huge variety of different sensors, actuators, hardware platforms *nor* the number of different disciplines being involved.

- We are convinced that *differences* of robotics compared to other domains *originate from* the need of a robot to cope with *open-ended environments while having* only *limited resources* at its disposal.



# What is different in robotics?

- The *difference* of robotics compared to other disciplines (e.g. automotive, avionics) is *neither* the huge variety of different sensors, actuators, hardware platforms *nor* the number of different disciplines being involved.

- We are convinced that *differences* of robotics compared to other domains *originate from* the need of a robot to cope with *open-ended environments while having* only *limited resources* at its disposal.

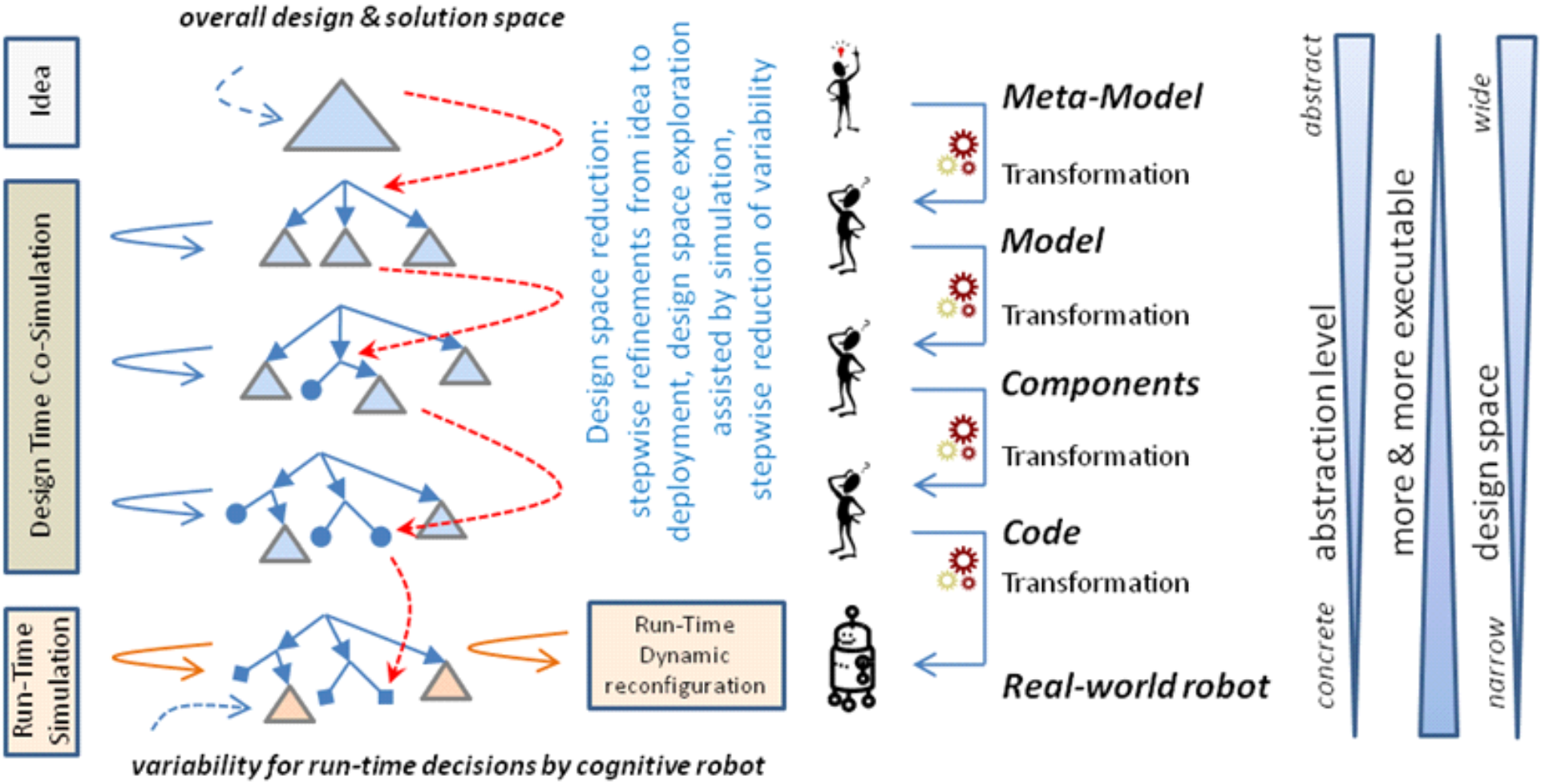
- *Limited resources* require decisions: when to assign which resources to what activity taking into account perceived situation, current context and tasks to be fulfilled.

- Due to *open-ended real-world environments*, it is impossible to statically assign resources in advance in such a way that all potential situations arising at runtime are properly covered.

- Due to the *enormeous sizes of the problem space and the solution space* in robotics, there will *always be a deviation between design-time and run-time optimality*.

- Therefore, there is a need for dynamic resource assignments at runtime: managing variants / variability at runtime by late bindings of purposefully left-open variation points *(models@runtime, accessible via MDSD + DSLs)*

# The Big Picture …
## … Design-time / Run-time Model Usage

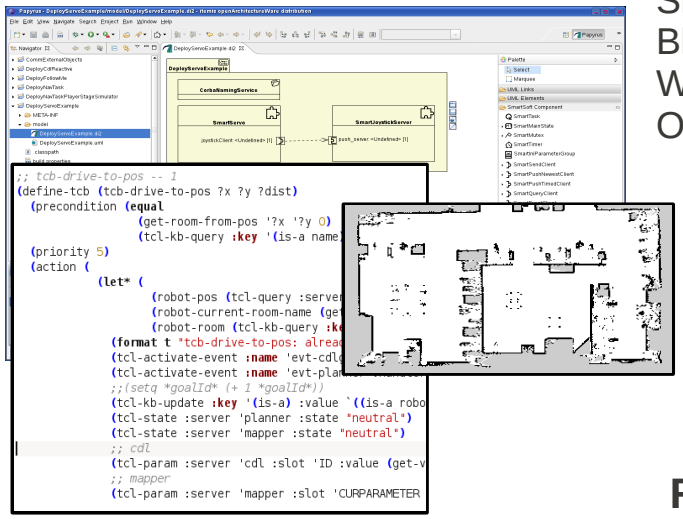


# The Big Picture ...
# ... Model-Centric Robotic Systems

service robotik

*Developer* → *Robot*

## Create/ Modify Models



SmartMDSD Toolchain,
Blender, Solid Works,
World / Map Editor,
Ontosaurus, ...

**Model Pool**
different views/
representations
of the models

## Modify Models

Manipulate models at run-time

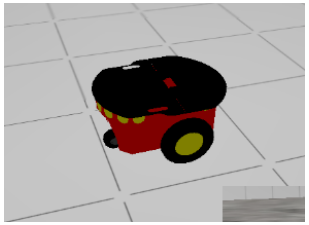
Reflect current state of the world and robot in the models

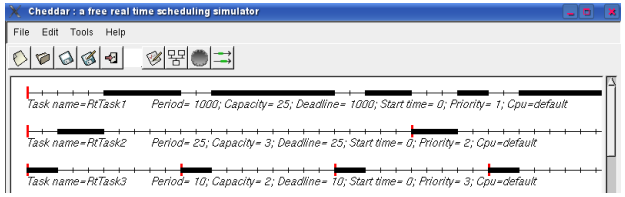
Make decisions at run-time depending on the models

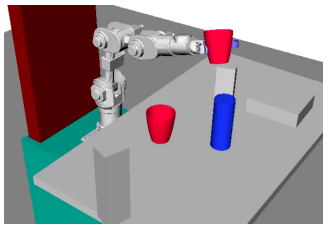
```
name          : CDL
  state       : "neutral"
  parameter
    strategy  : followPerson
    freebehavior : activate
    lookuptable  : default
    goalmode  : person
    approachdist : 500
    id        : –
  resources
    task–1
      period  : 100ms
```

## Reason on the Models

Analysis, Simulation, Planning, ...





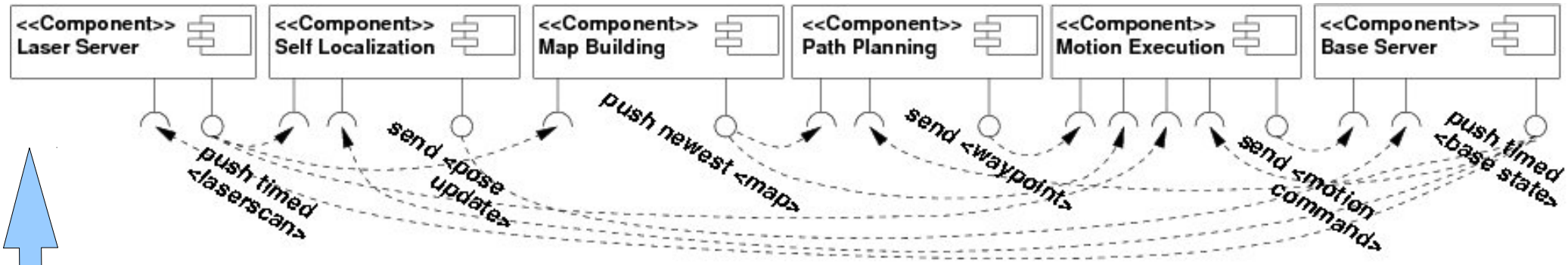
CHEDDAR, OpenRAVE, Gazebo, Metric-FF, LAMA, ...

*Design-Time* → *Run-Time*

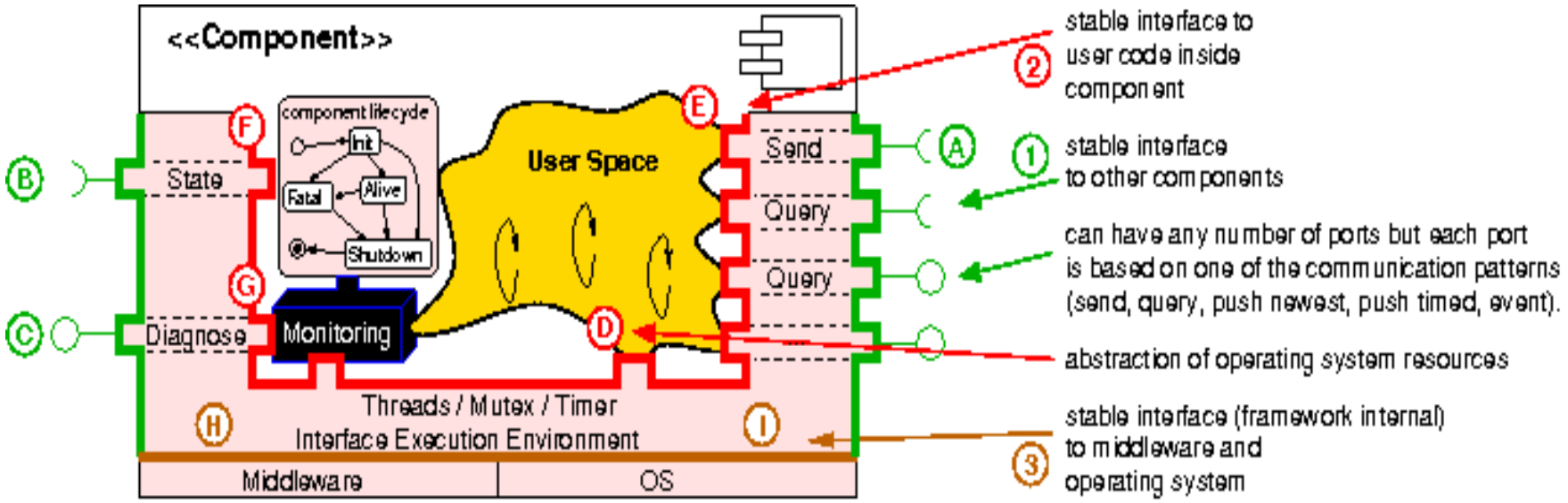
Hochschule Ulm

service robotik

# Model Driven Software Development
# Separation of Roles / Concerns

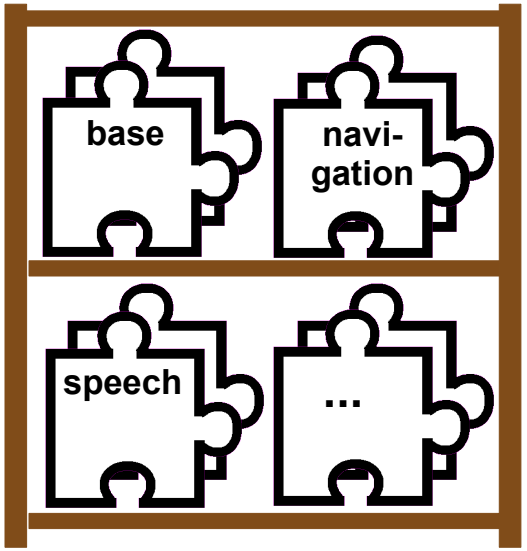



service-oriented component approach with loose couplings: ensuring separation of concerns / roles




Hochschule Ulm

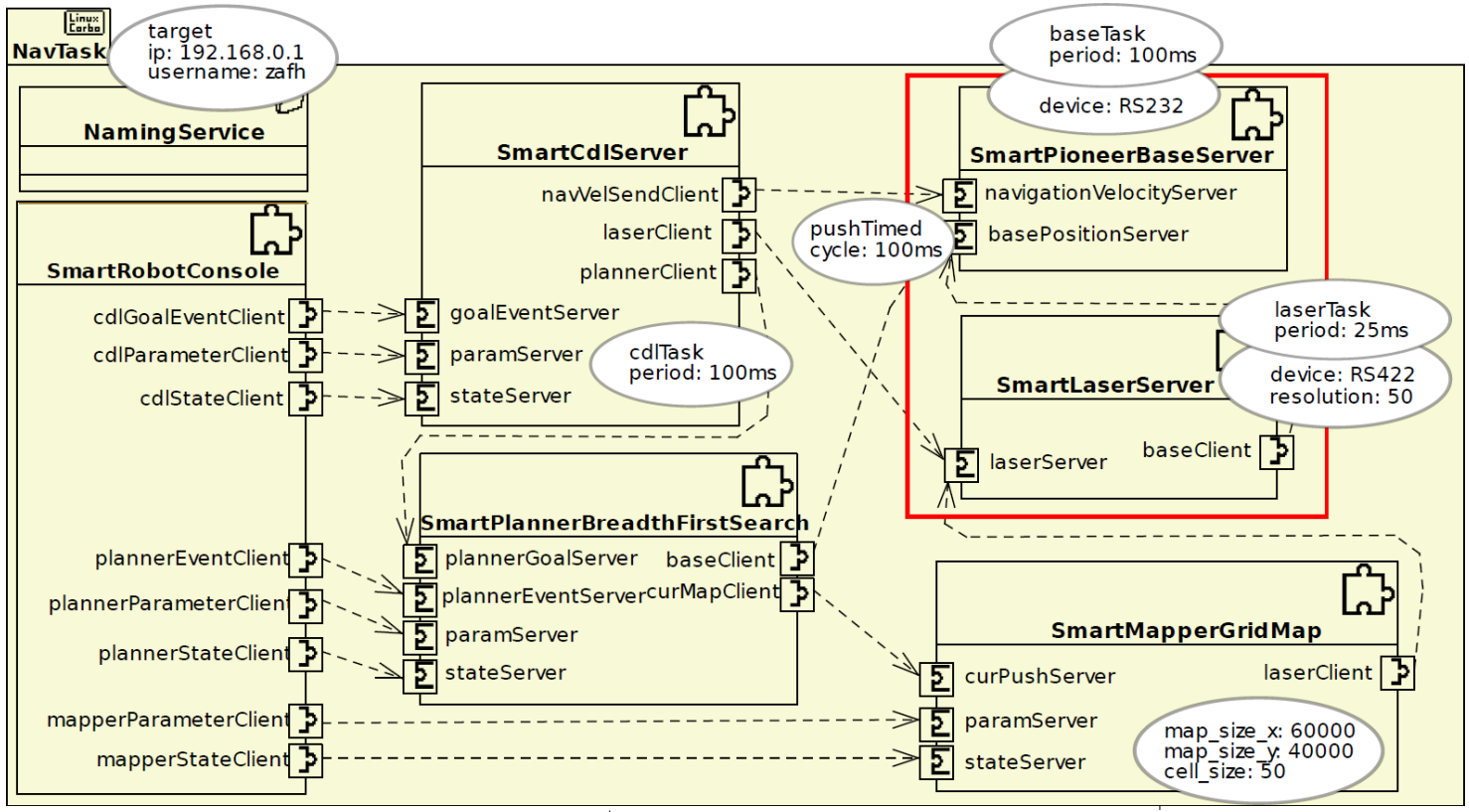
# Model Driven Software Development
# Separation of Roles / Concerns




**Component Shelf**
**Reusable Components**

- ***Separation of Roles***
- ***Separation of Concerns***

## *System Level Properties / Bindings / Conformance Checks*





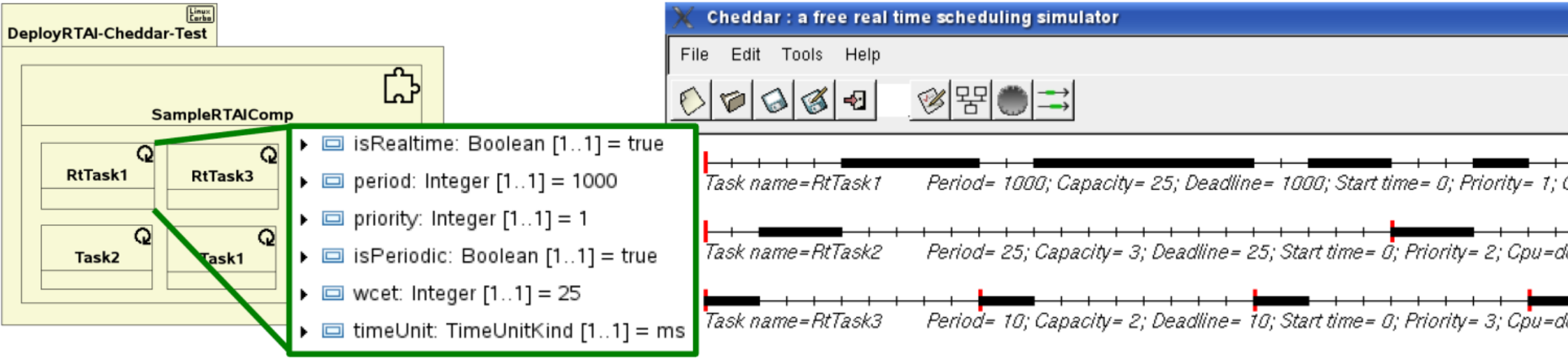

Hochschule Ulm

# Model Driven Software Development
# Separation of Roles / Concerns

## *System Level Properties / Bindings / Conformance Checks*

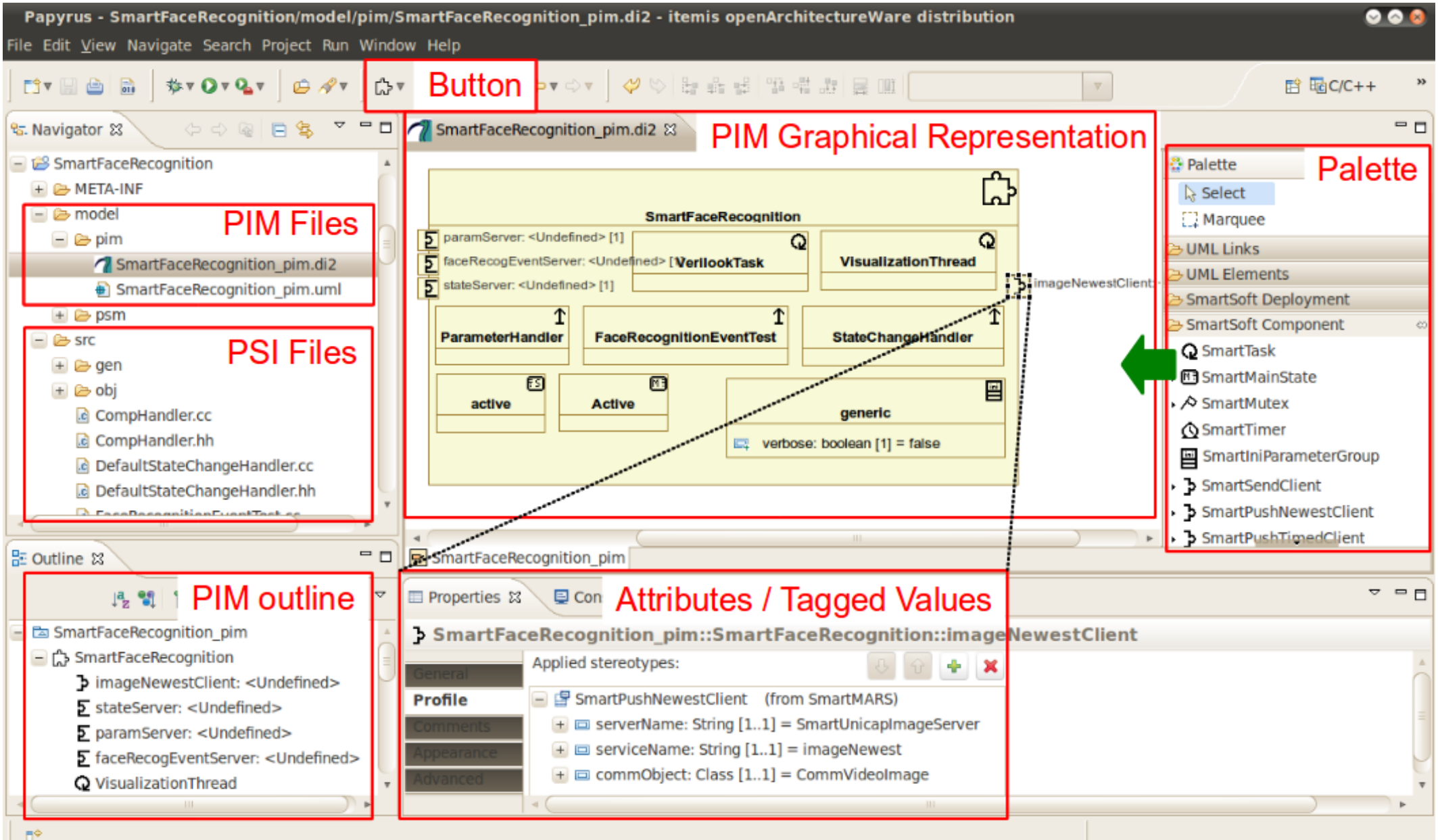
Resource Awareness and Quality of Service

- Example: Schedulability Analysis (CHEDDAR)



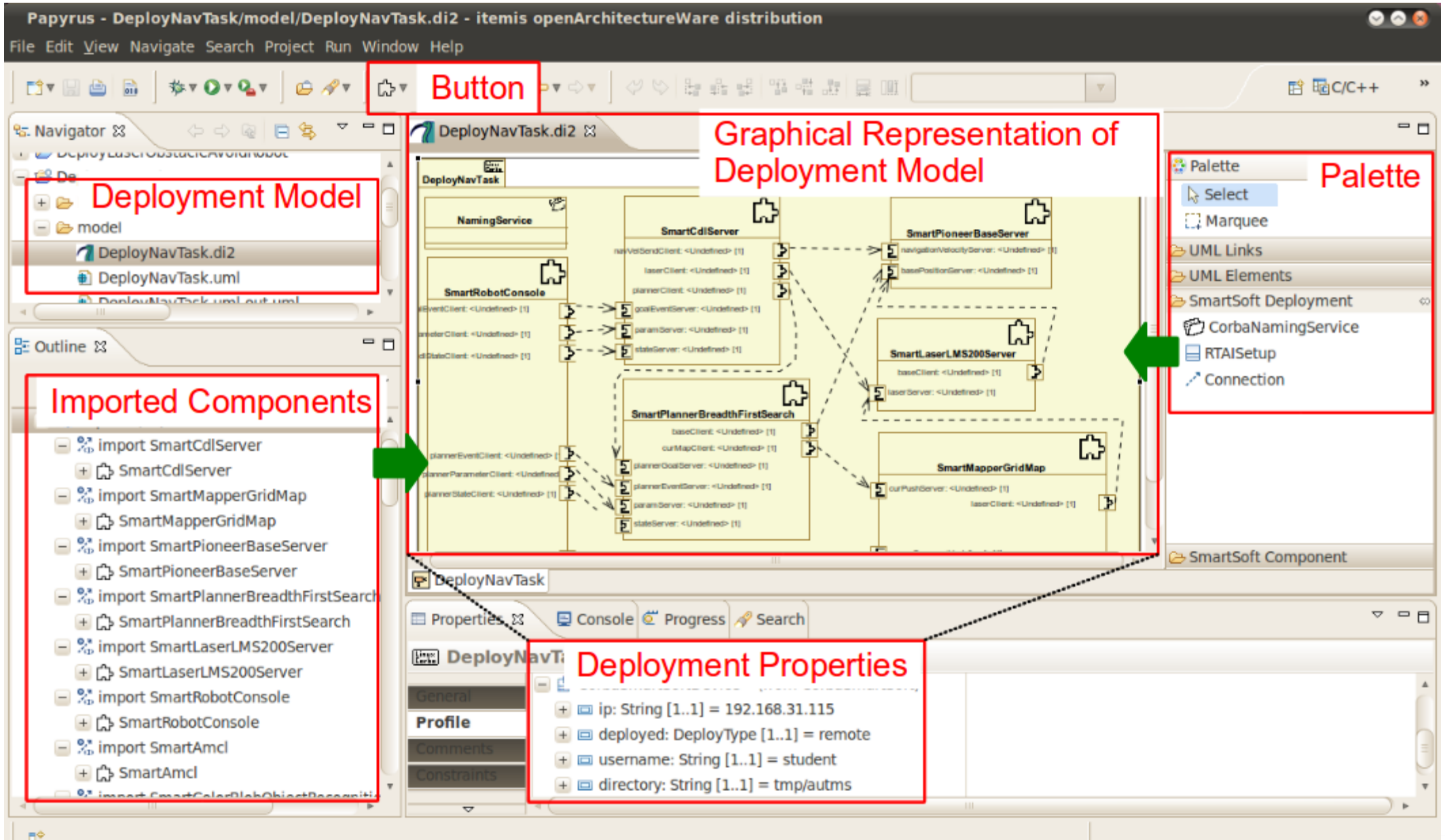
service robotik

# Model Driven Software Development
# Component Builder View



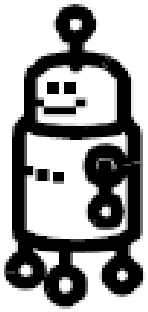
service robotik

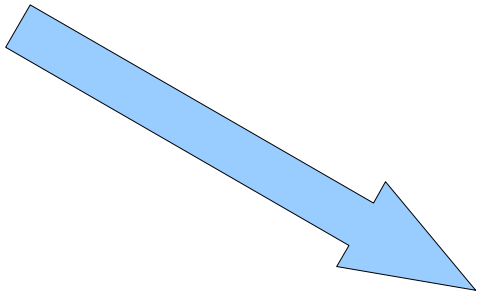
# Model Driven Software Development
# System Integrator View

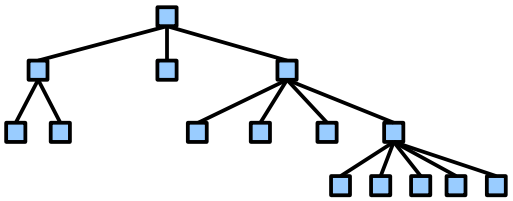


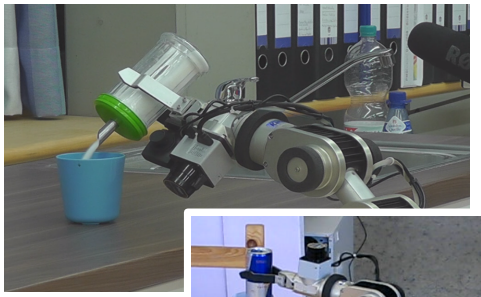
# Model Driven Software Development
# Robot View

Run-Time
Dynamic
reconfiguration





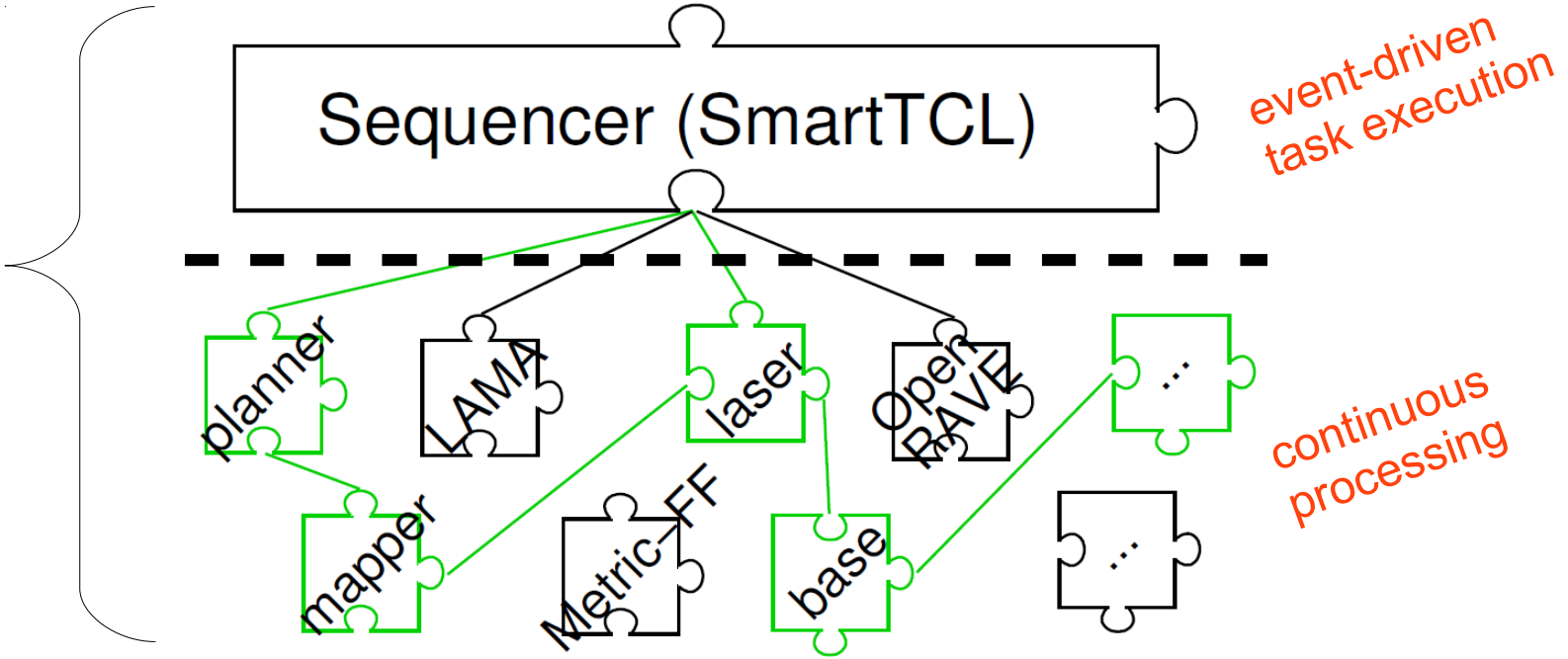




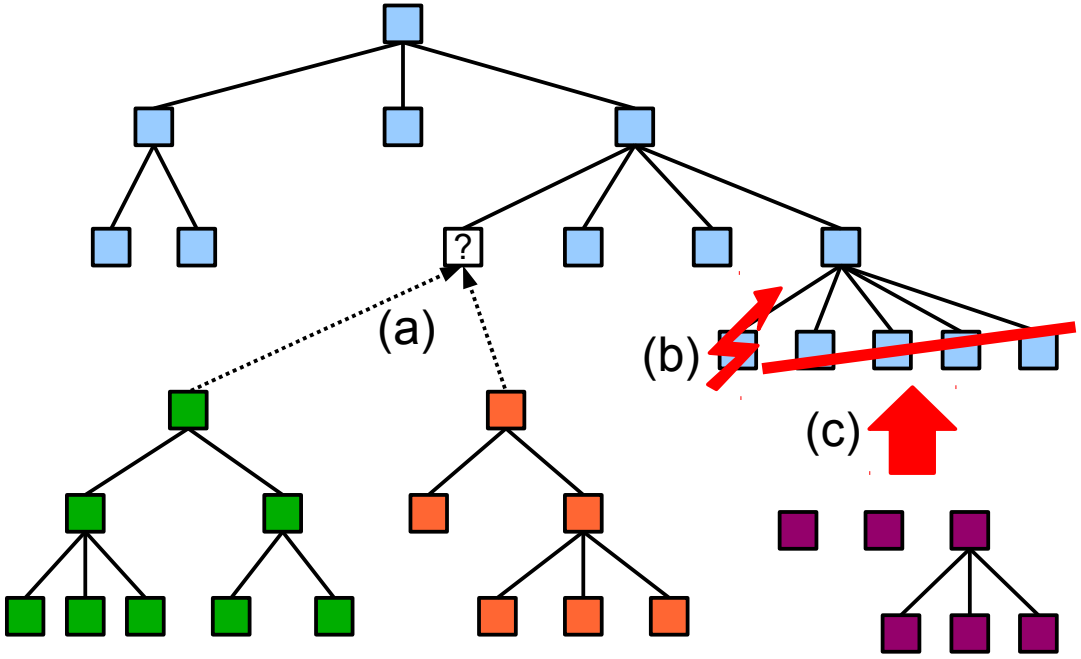
Hochschule Ulm

# Run-Time: Managing Execution Variants
## Sequencer Orchestrates the Components

- bridges between continuous processing and event-driven task execution
- the sequencer orchestrates the software components in the system:
  - send parameters / configurations
  - switch components on/off to manage resources
  - change the wiring between the components
  - query information / wait for events



# Run-Time: Managing Execution Variants
## Sequencer: SmartTCL Task-Tree



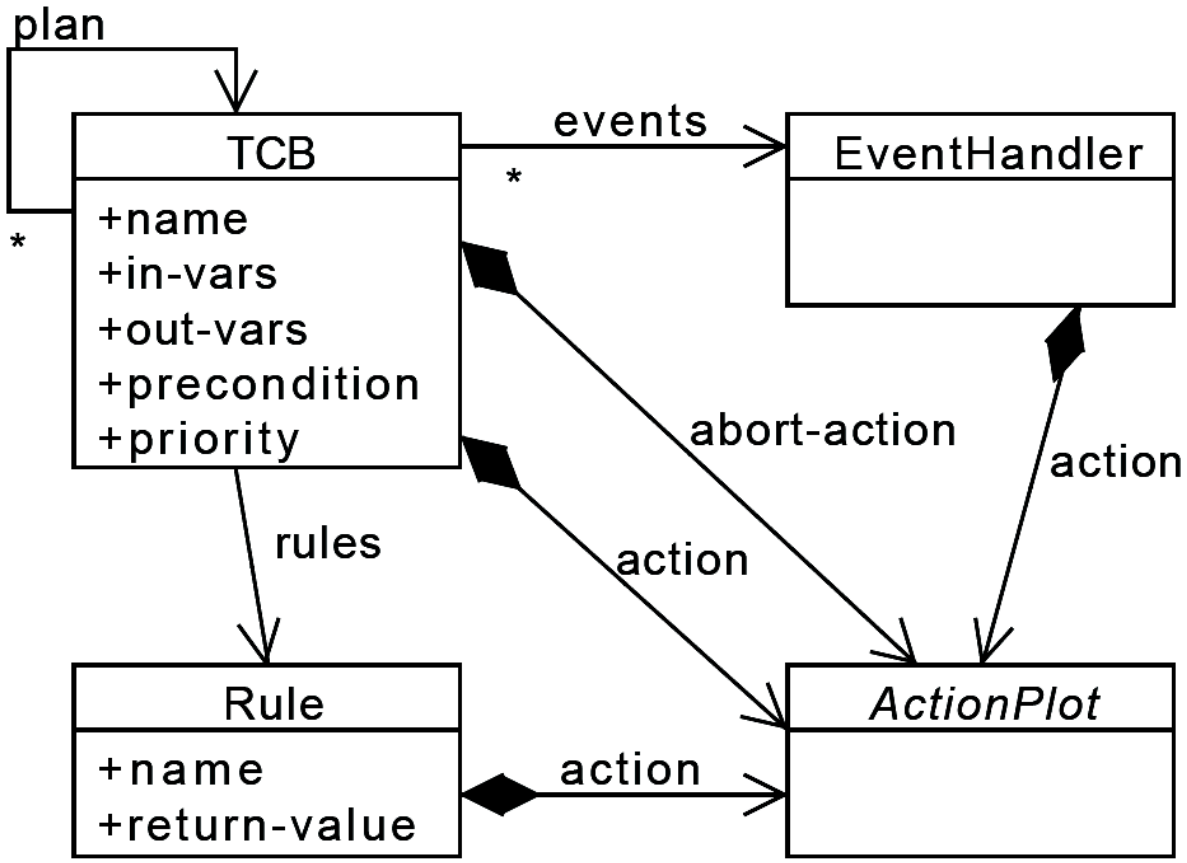

(a) select between alternatives at runtime

(b) handle contingencies

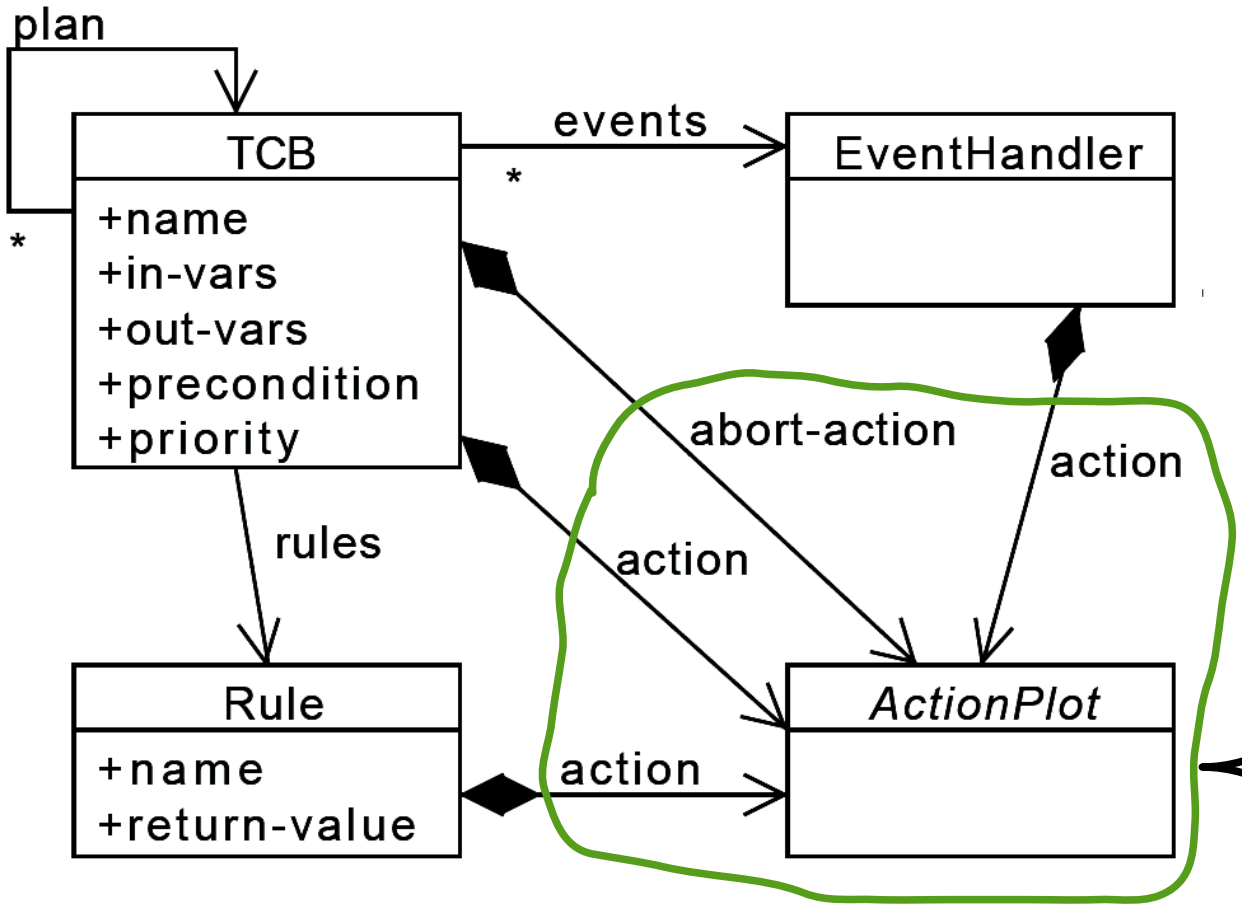
(c) delete, add or replace parts of the task-tree at runtime

- at runtime a task-tree is dynamically created, modified and executed
- composes reusable action-plots to complex behaviors
- manages execution variants and contingencies of real world environments
- provides context and situation-driven task execution
- mediates between symbolic and subsymbolic mechanisms of information processing

# Run-Time: Managing Execution Variants
## The SmartTCL Meta-Model



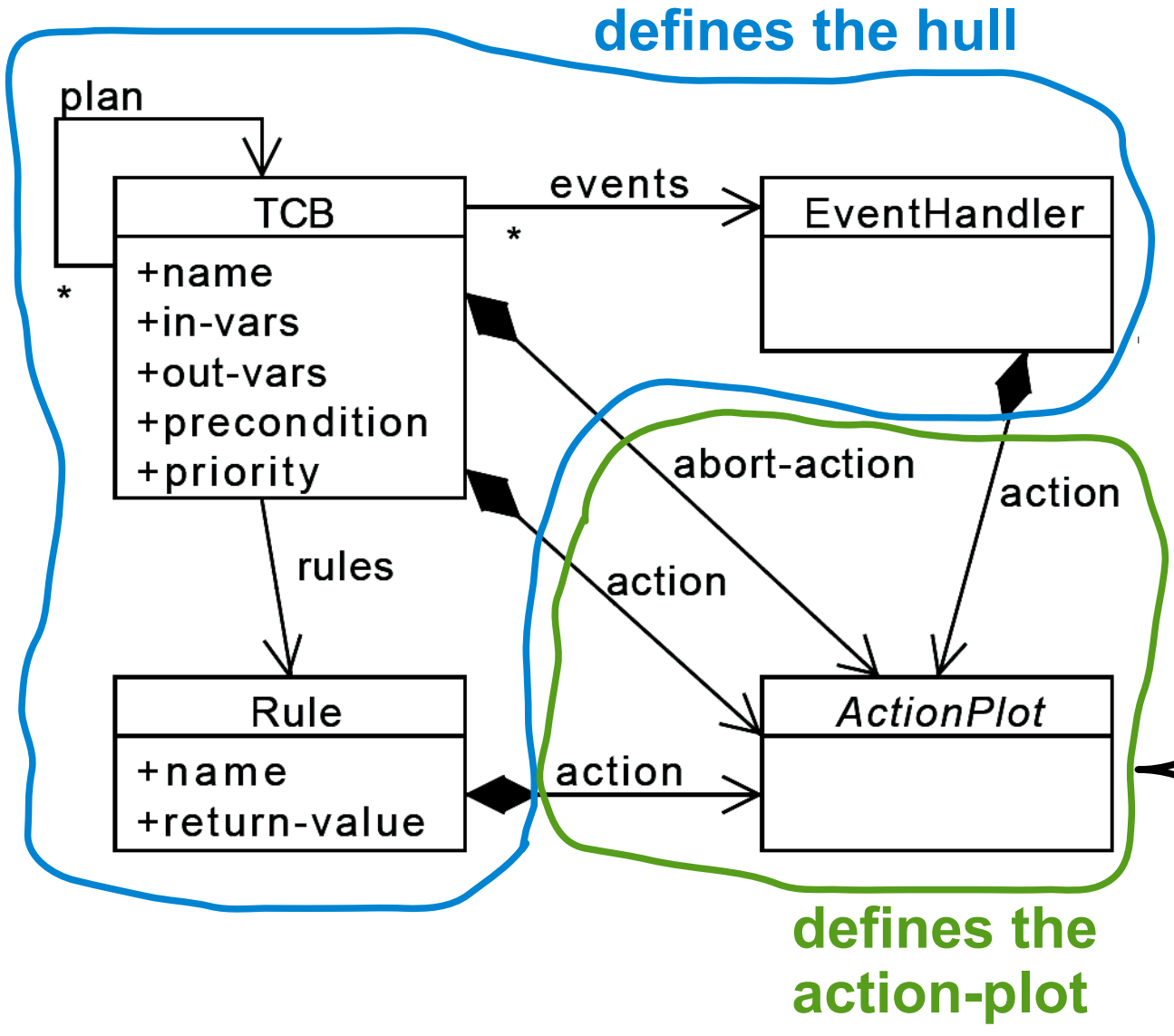
# Run-Time: Managing Execution Variants
## The SmartTCL Meta-Model




Lisp code (with restrictions):

- actions should not invoke blocking calls that take a long time relative to the reactivity which is expected from SmartTCL
- SmartTCL specific function:
  - tcl-param, tcl-state
  - tcl-wiring, tcl-query
  - tcl-activate-event
  - tcl-delete-event
  - ...

# Run-Time: Managing Execution Variants
## The SmartTCL Meta-Model




Actions are encapsulated by a hull:

- TCB
- EventHandler
- Rule

Lisp code (with restrictions):

- actions should not invoke blocking calls that take a long time relative to the reactivity which is expected from SmartTCL

- SmartTCL specific function:
  - tcl-param, tcl-state
  - tcl-wiring, tcl-query
  - tcl-activate-event
  - tcl-delete-event
  - ...

service robotik

# Run-Time: Managing Execution Variants
## TCB Programming

The *Hull* provides a stable structure that allows a black-box view on the action-plots and thus ensures reusability and composability → **Seperation of Roles**

**defines the hull**

```
(define-tcb (tcb-get-coffe-machine-cup-pose ?coffeMachineId => ?x ?y ?z)
  (rules nil)
  (precondition nil)
  (action (
```

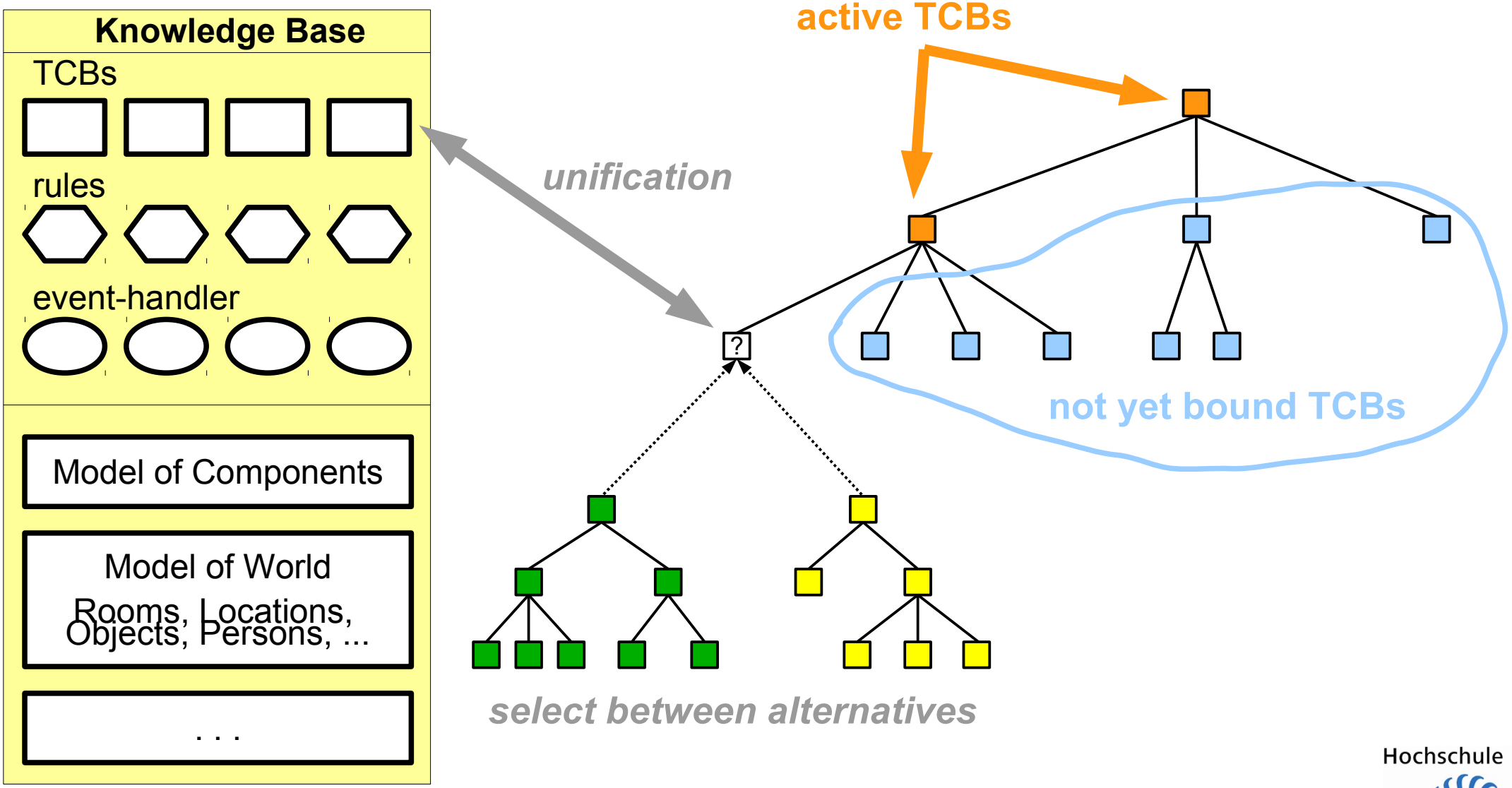
```
(format t "=========================>>> tcb-get-coffe-machine-cup-pose ~d ~%" '?coffeMachineId)
;; query coffe machine pose and cup-offset from KB
(let* ((coffeeMachine (tcl-kb-query :key '(is-a id) :value '((is-a object)(id ?coffeMachineId))))
       (coffeeMachinePose (get-value coffeeMachine 'pose))
       (cup-offset (get-value coffeeMachine 'cup-offset))
       (pose nil))
  ;; transform pose to point
  (setf pose (eval (append '(transformPoseToPoint) coffeeMachinePose cup-offset)))
  ;; bind output variables
  (tcl-bind-var :name '?x :value (first pose))
  (tcl-bind-var :name '?y :value (second pose))
  (tcl-bind-var :name '?z :value (third pose))
  '(SUCCESS ())))))
```

**defines the action-plot**

To programm the Action-Plots the developers are free, for example, to do calculations, query for information from components or the KB.

# Run-Time: Managing Execution Variants

## TCB Selection at Run-Time



# Scenario: Robot “Kate” prepares and delivers Coffee

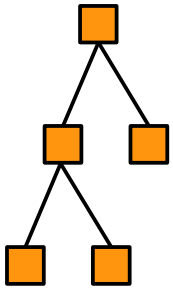
Watch Video on YouTube
http://www.youtube.com/roboticsathsulm





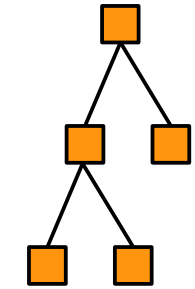



# Run-Time: Managing Execution Variants
## Separation of Roles / Reusability / Composability

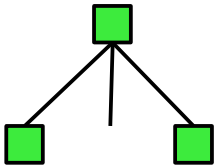


approach kitchen

# Run-Time: Managing Execution Variants
## Separation of Roles / Reusability / Composability

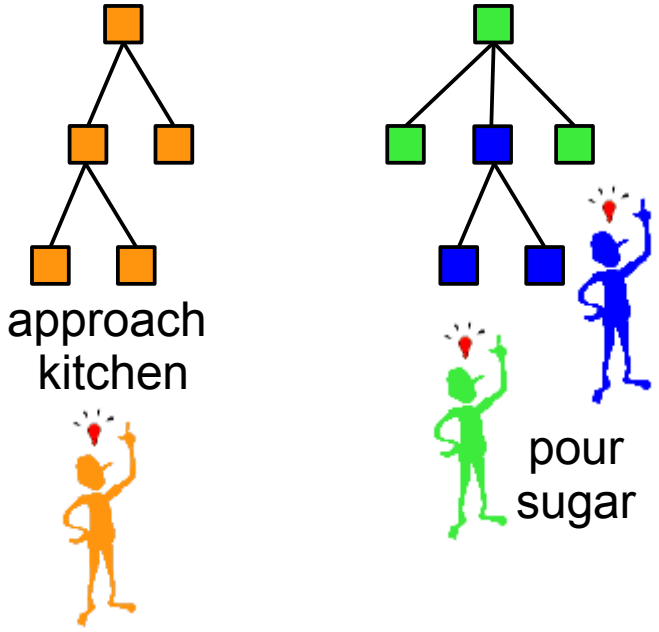


approach kitchen

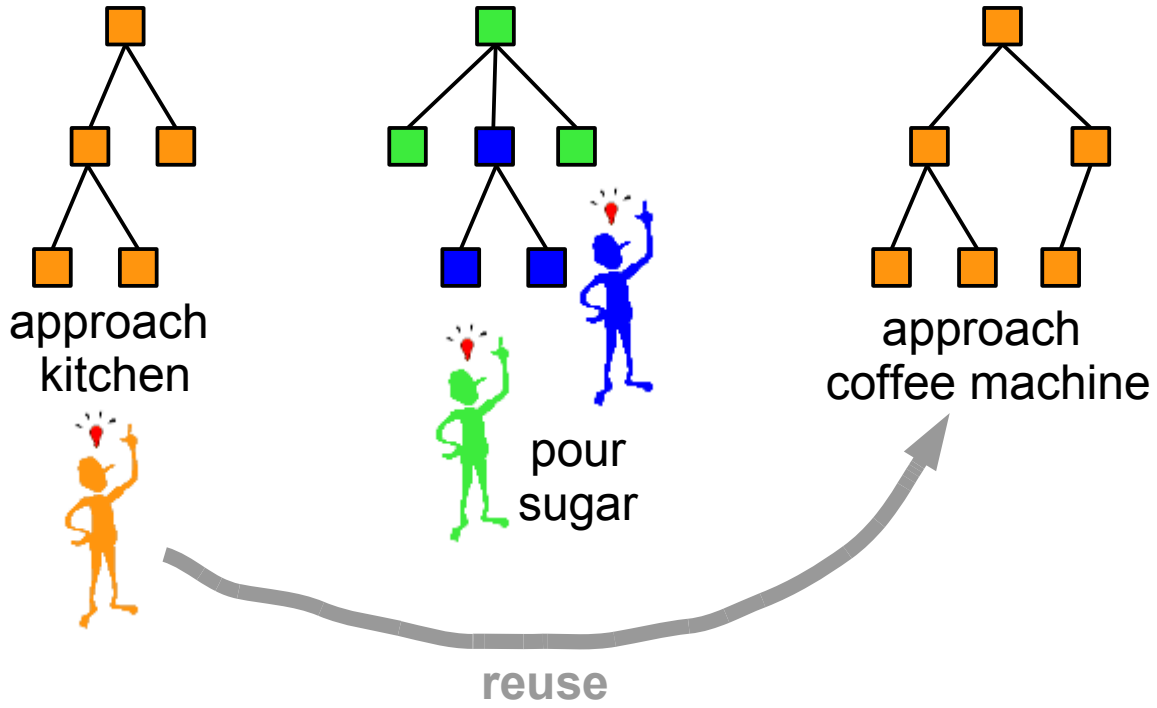


pour sugar

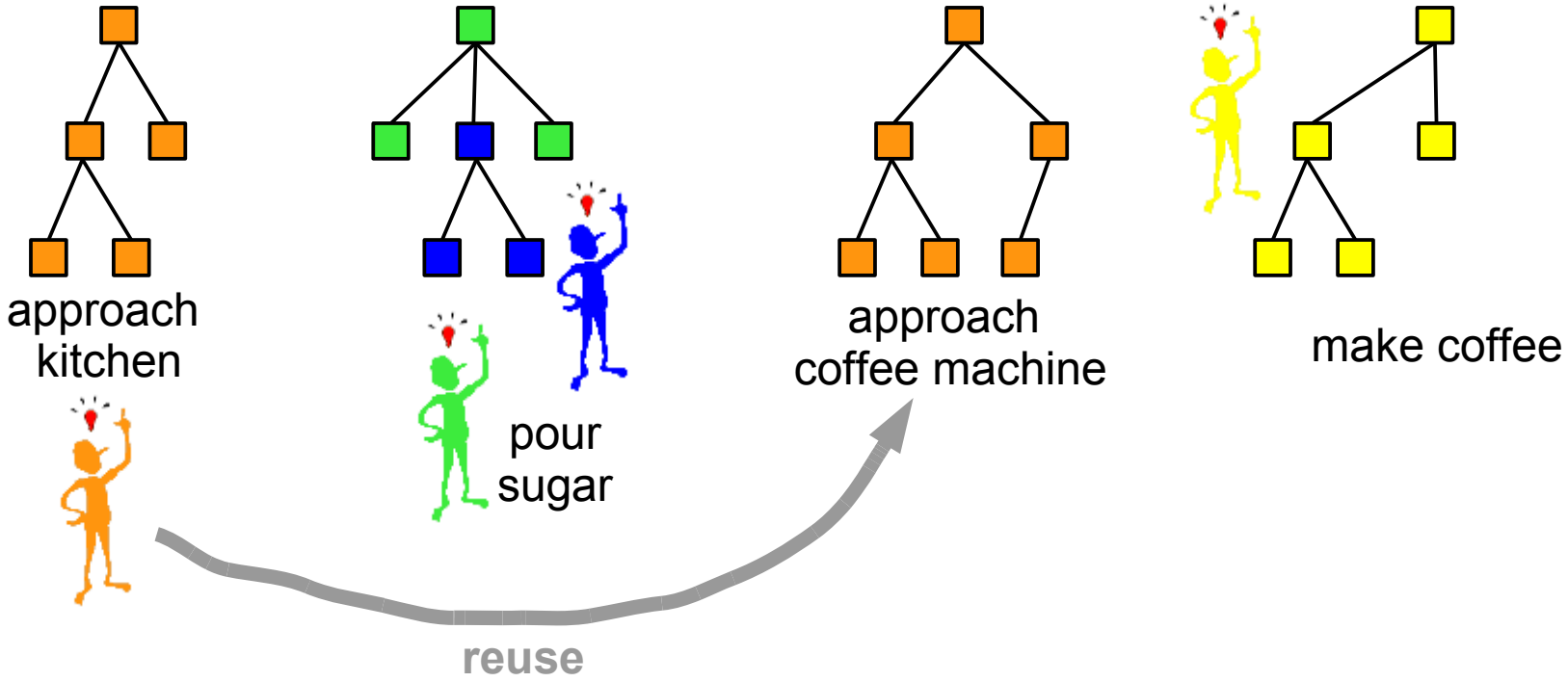
# Run-Time: Managing Execution Variants
## Separation of Roles / Reusability / Composability



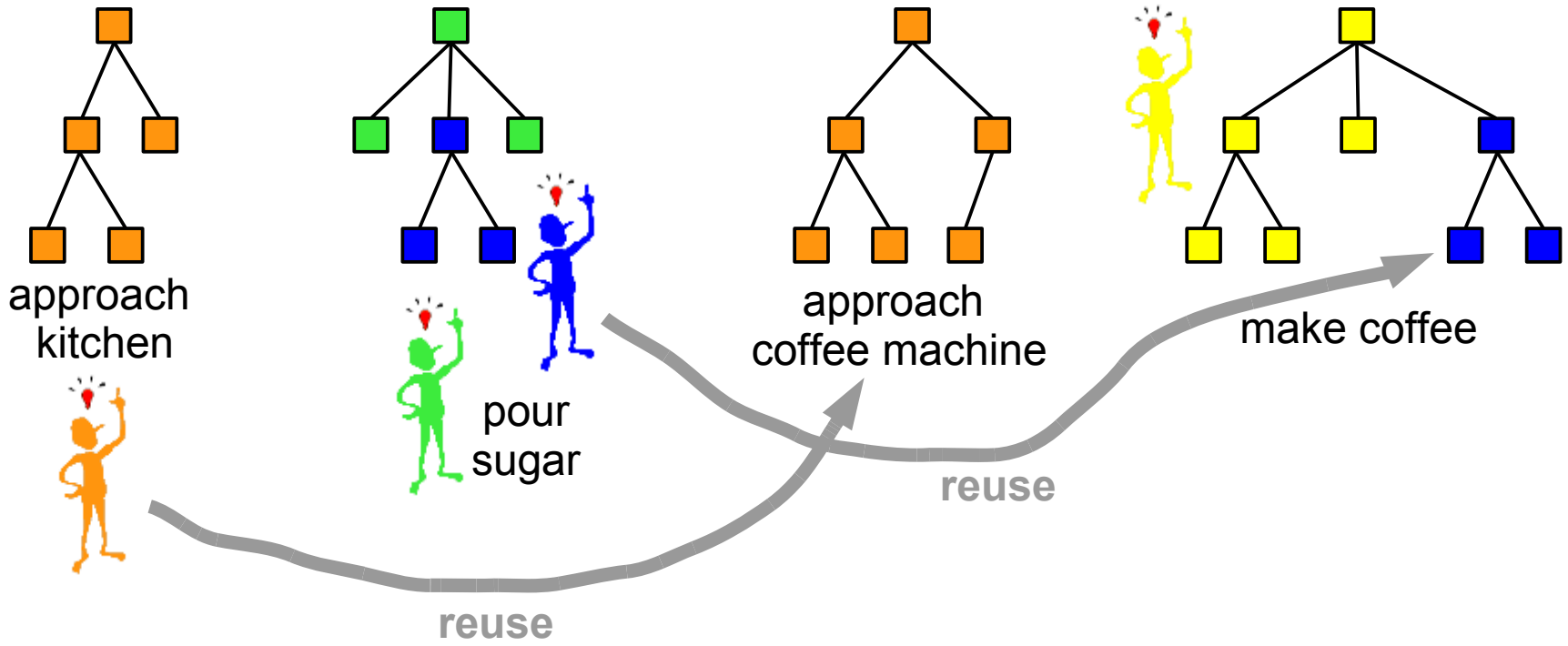
# Run-Time: Managing Execution Variants
## Separation of Roles / Reusability / Composability



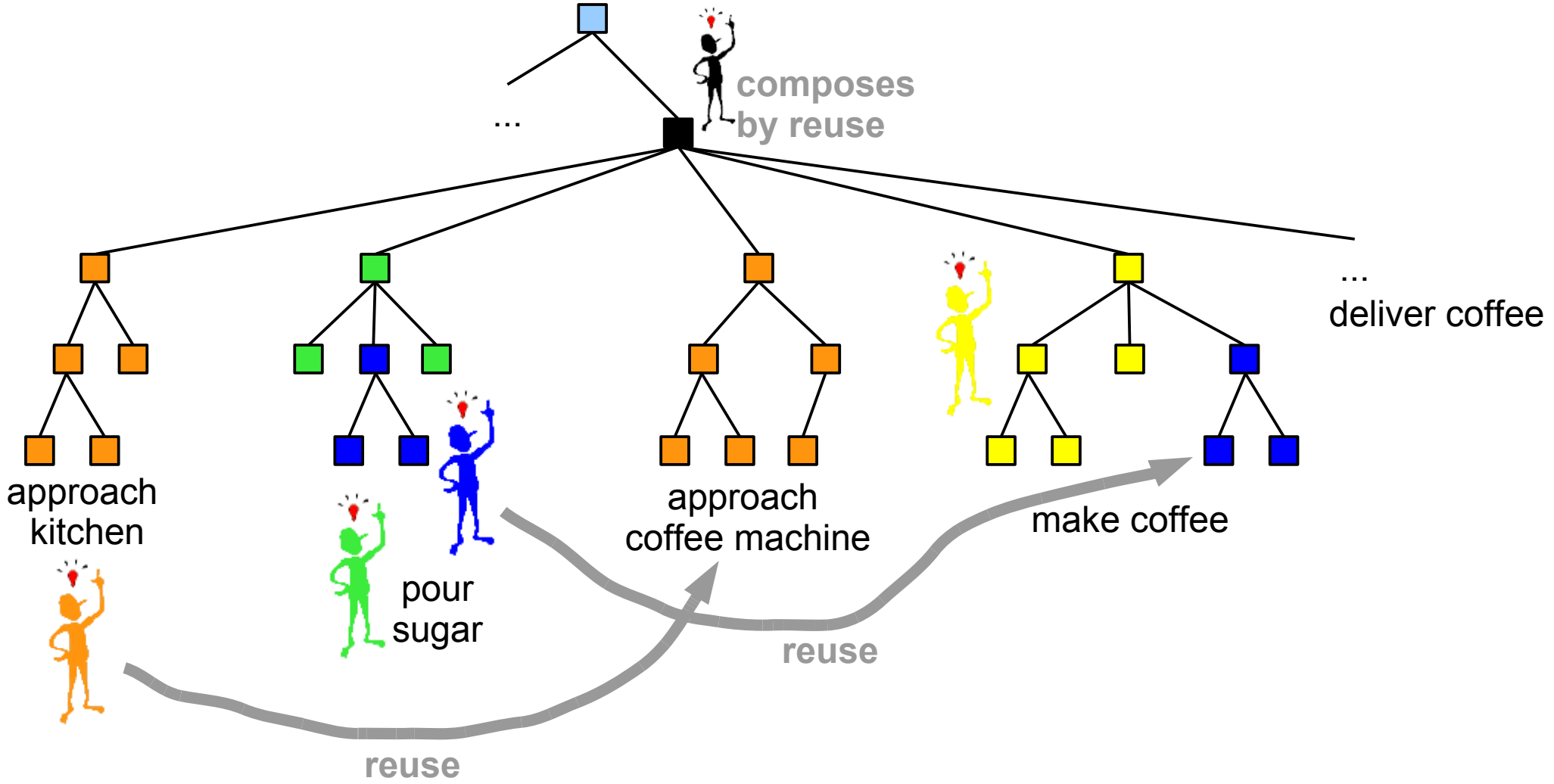
# Run-Time: Managing Execution Variants
## Separation of Roles / Reusability / Composability



# Run-Time: Managing Execution Variants
## Separation of Roles / Reusability / Composability



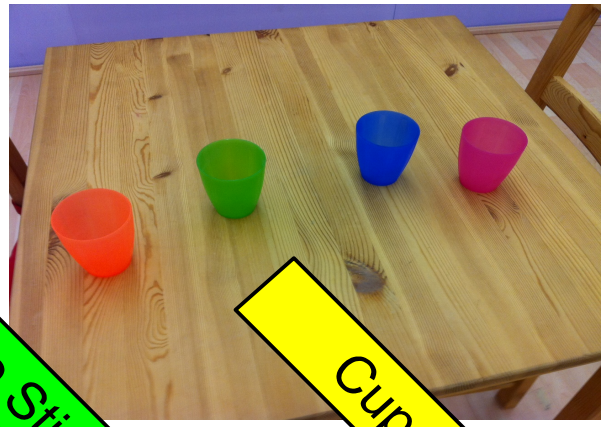
# Run-Time: Managing Execution Variants
## Separation of Roles / Reusability / Composability



# Run-Time: Managing Execution Variants

## “Kate” clean the table



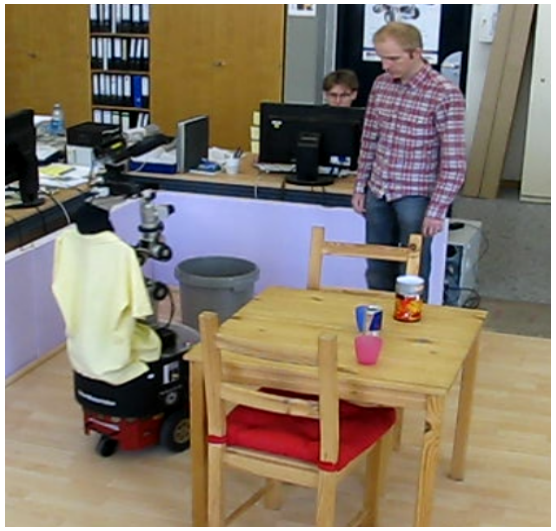
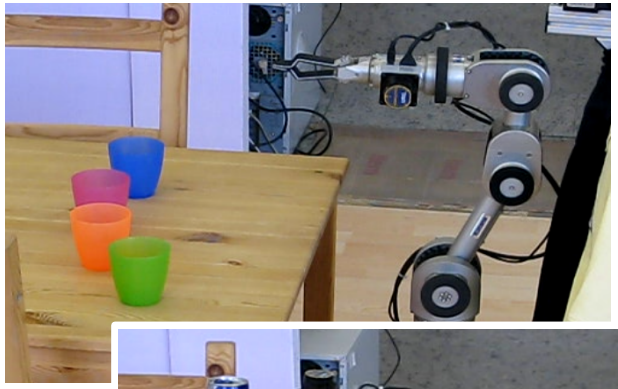


*cups* can be stacked into each other

*one red bull* can be stacked into one *potato sticks*

service robotik

# Scenario: Robot “Kate” cleans up a table
## Managing Execution Variants
## Handling Contingencies

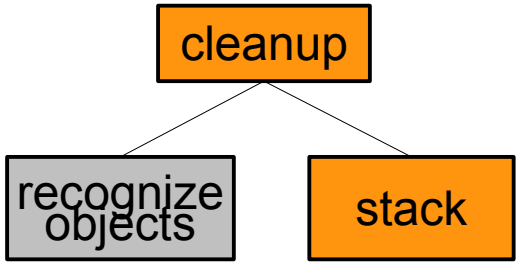




YouTube

Hochschule Ulm

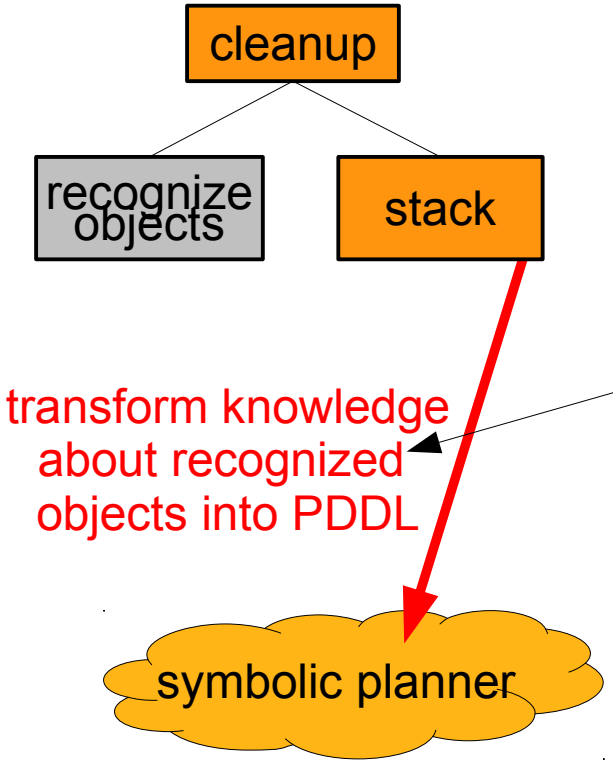
# Run-Time: Managing Execution Variants
## Calling a Symbolic Planner




stores recognized objects in KB

the different variants how to stack the different objects is huge
→ calling a symbolic planner in that specific situation helps the manage the combinatorial explosion

# Run-Time: Managing Execution Variants
## Calling a Symbolic Planner

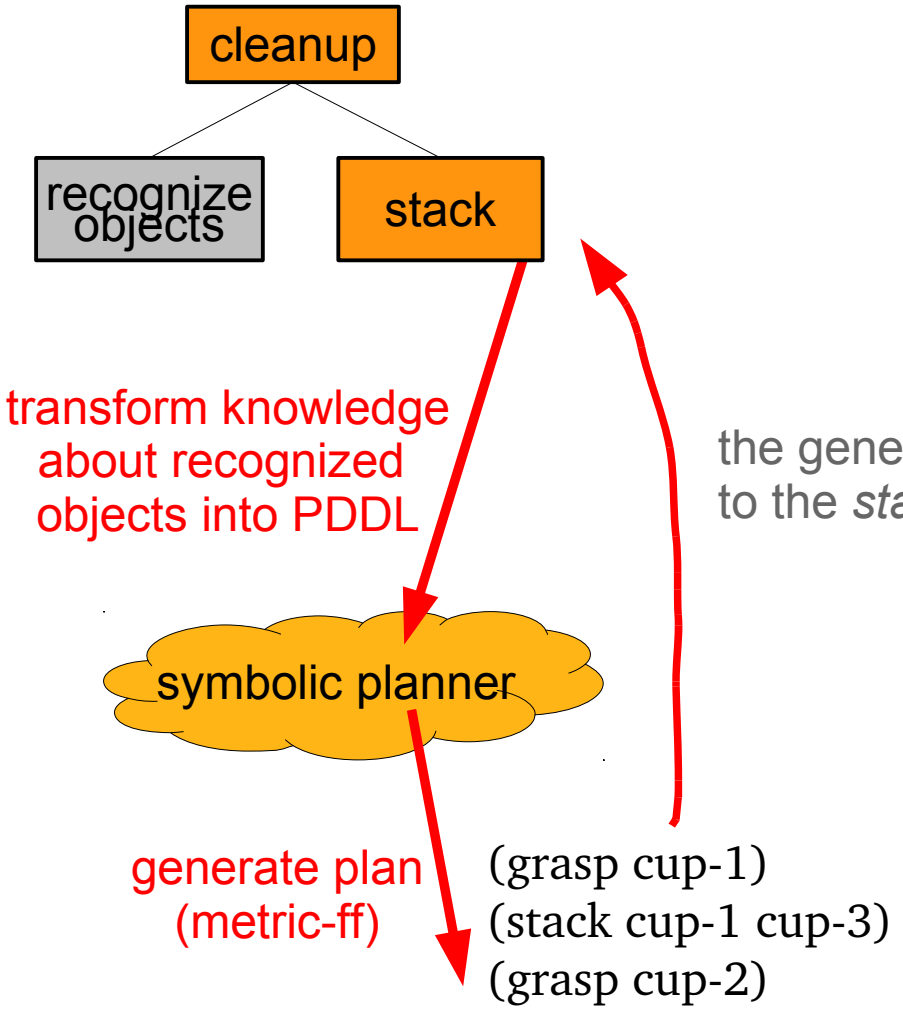



this is encoded in the action plot of the *stack* TCB:

- the recognized objects are queried from the KB
- the stable domain description (PDDL) as well as the situation specific fact description (PDDL) are created and forwarded to the symbolic planner

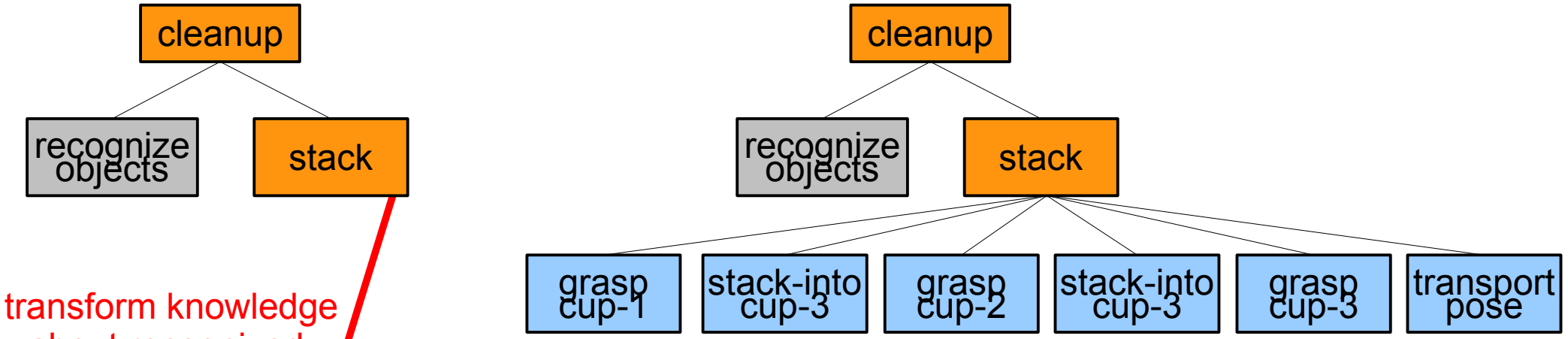
the *SmartSymbolicPlanner* component provides the service to call symbolic planners like ff, metric-ff, lama, ...

# Run-Time: Managing Execution Variants
## Calling a Symbolic Planner




the generated plan is returned to the *stack* TCB

# Run-Time: Managing Execution Variants
## Calling a Symbolic Planner




transform knowledge about recognized objects into PDDL

symbolic planner

generate plan (metric-ff)

(grasp cup-1)
(stack cup-1 cup-3)
(grasp cup-2)
(stack cup-2 cup-3)
(grasp cup-3)
(transport)

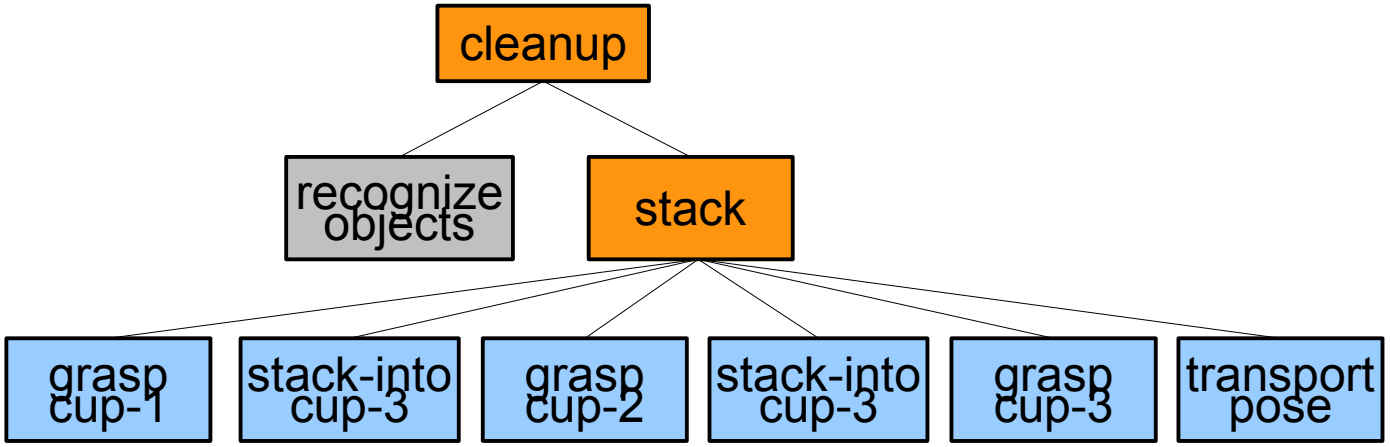
add TCBs to task-tree

**reusable *TCBs***

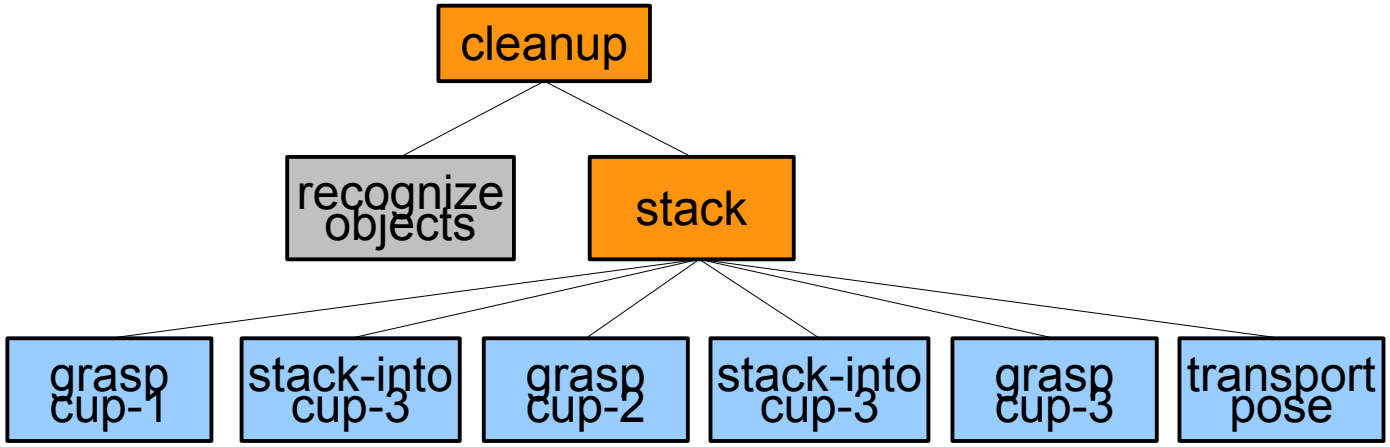
the knowledge how to transform the plan steps into TCBs is encoded in the action plot of the *stack* TCB

transform plan steps into SmartTCL Task Coordination Blocks (TCBs)

# Run-Time: Managing Execution Variants
## Calling a Symbolic Planner

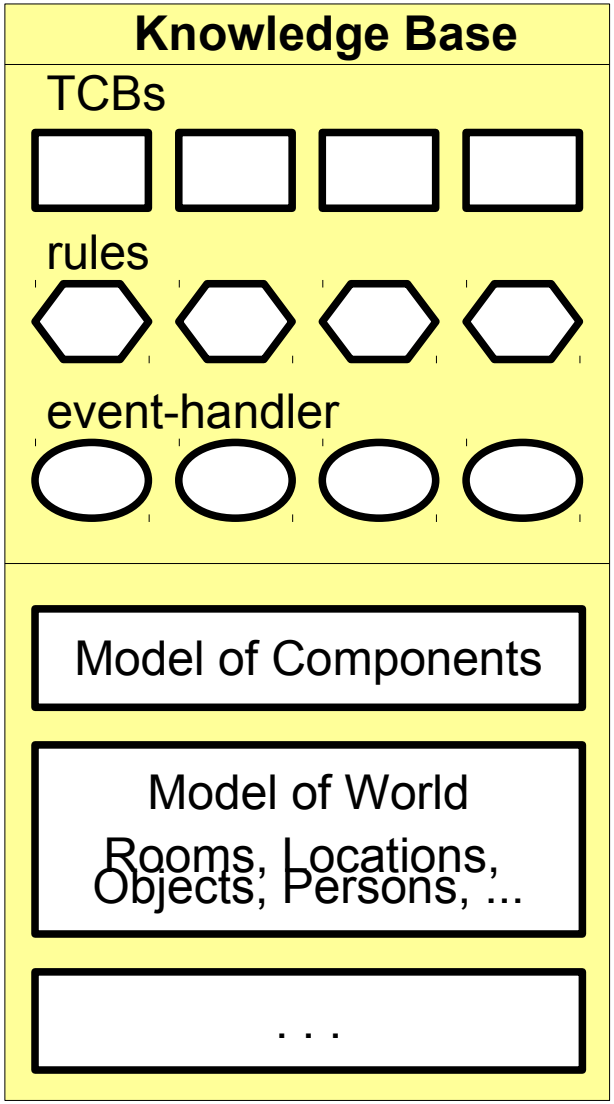


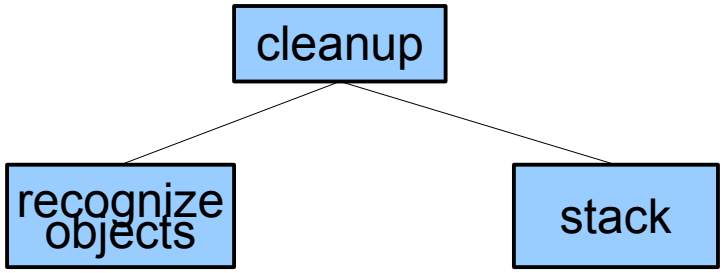
# Run-Time: Managing Execution Variants
## Calling a Symbolic Planner



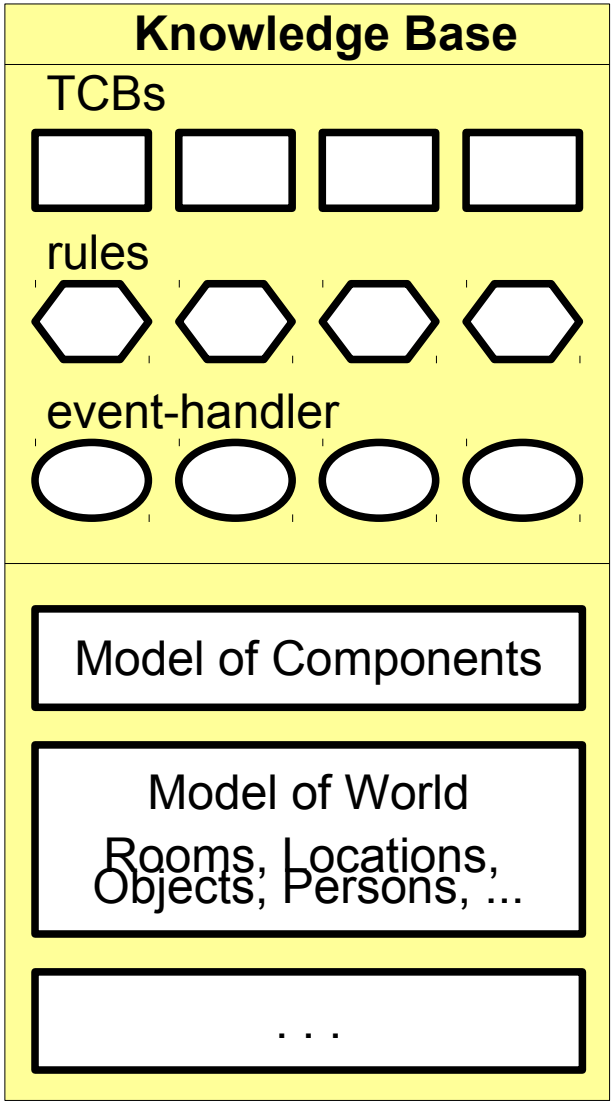

"Anything that can go wrong will go wrong"
[Edward A. Murphy]

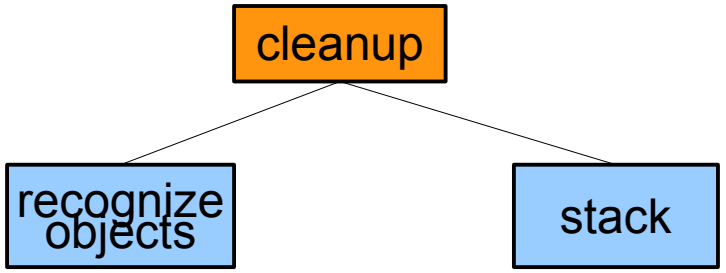
# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies



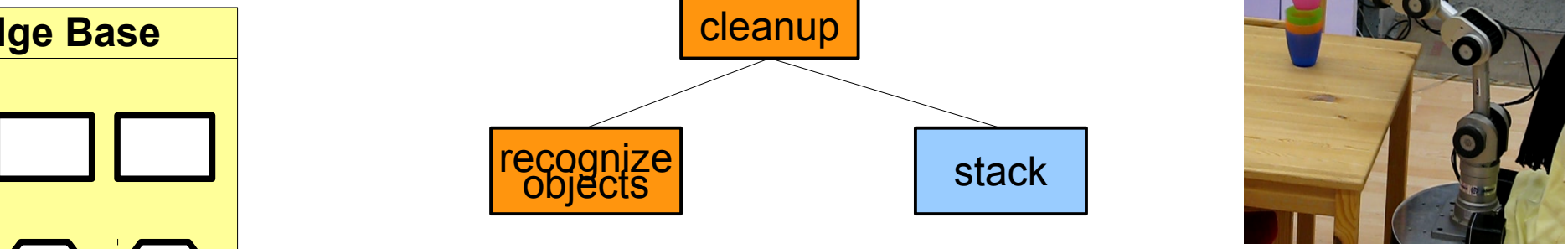




# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies

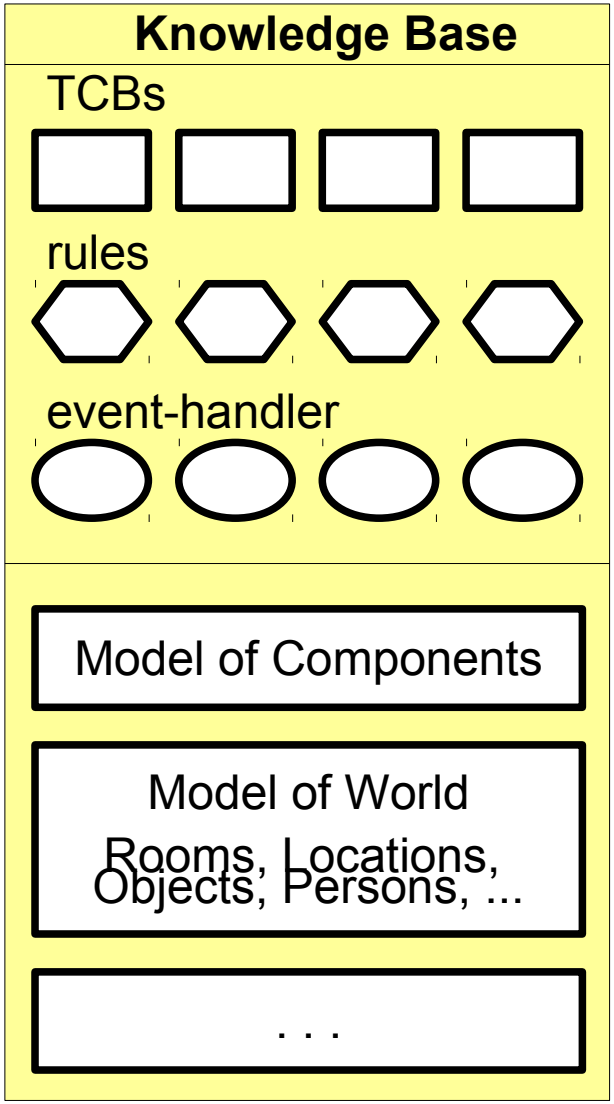


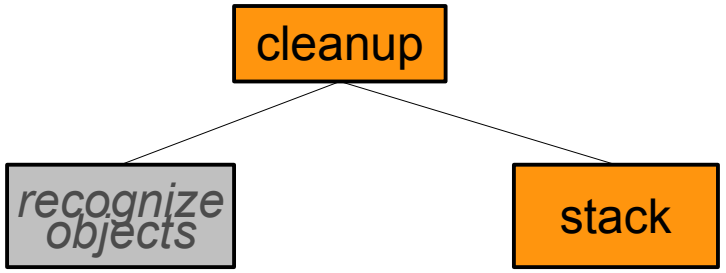



# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies



# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies

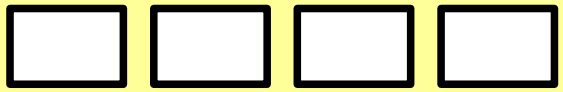


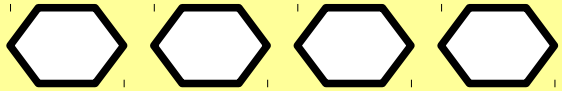
# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies

**Knowledge Base**

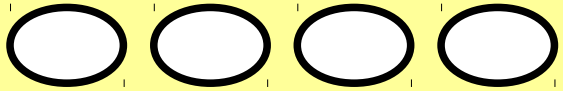
TCBs



rules



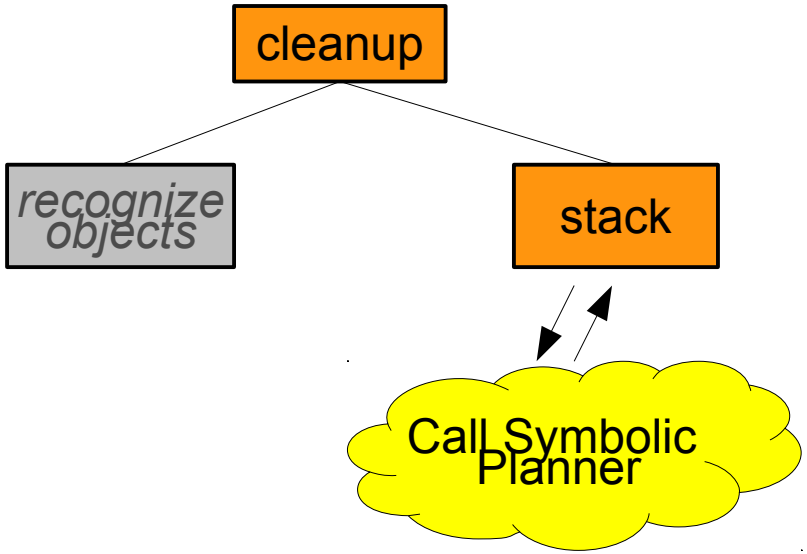
event-handler



Model of Components

Model of World
Rooms, Locations, Objects, Persons, ...

. . .



# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies

**Knowledge Base**

TCBs

rules

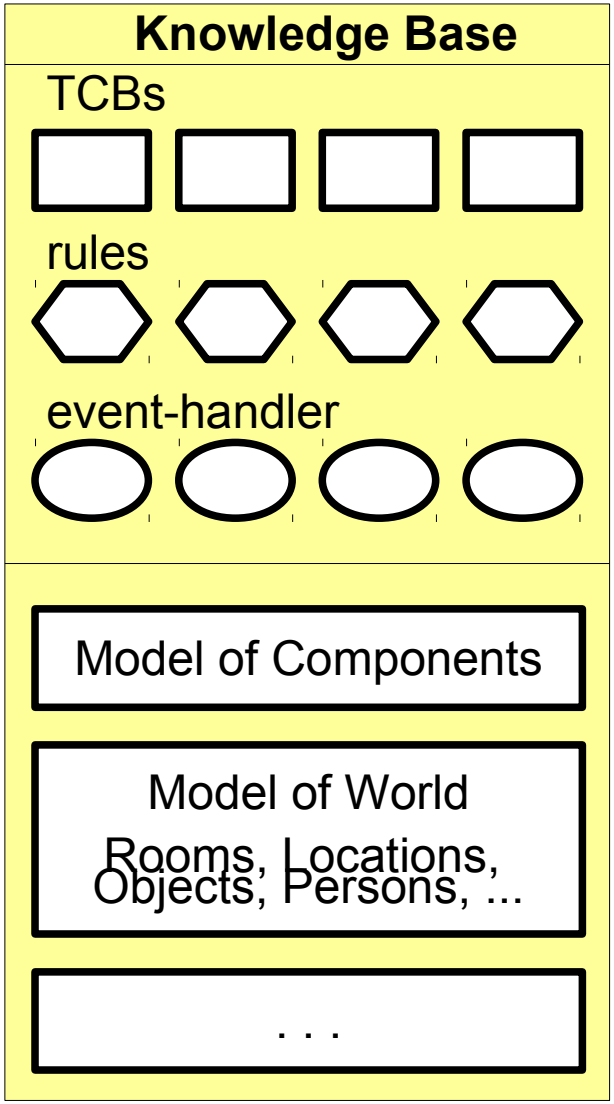
event-handler

Model of Components

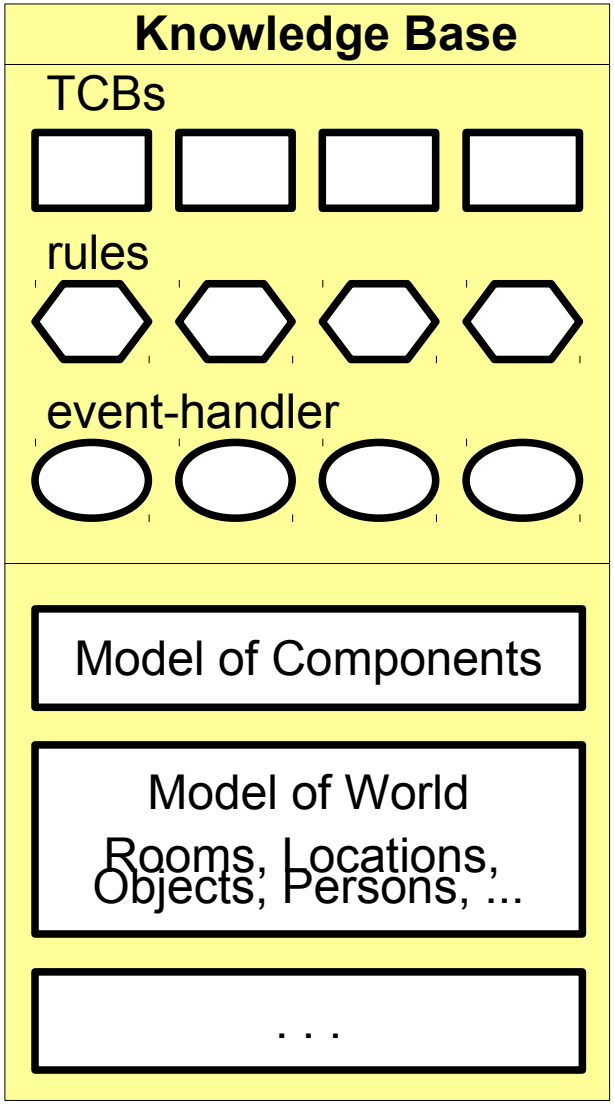
Model of World
Rooms, Locations, Objects, Persons, ...

. . .

# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies



# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies



# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies

**Knowledge Base**

TCBs

rules

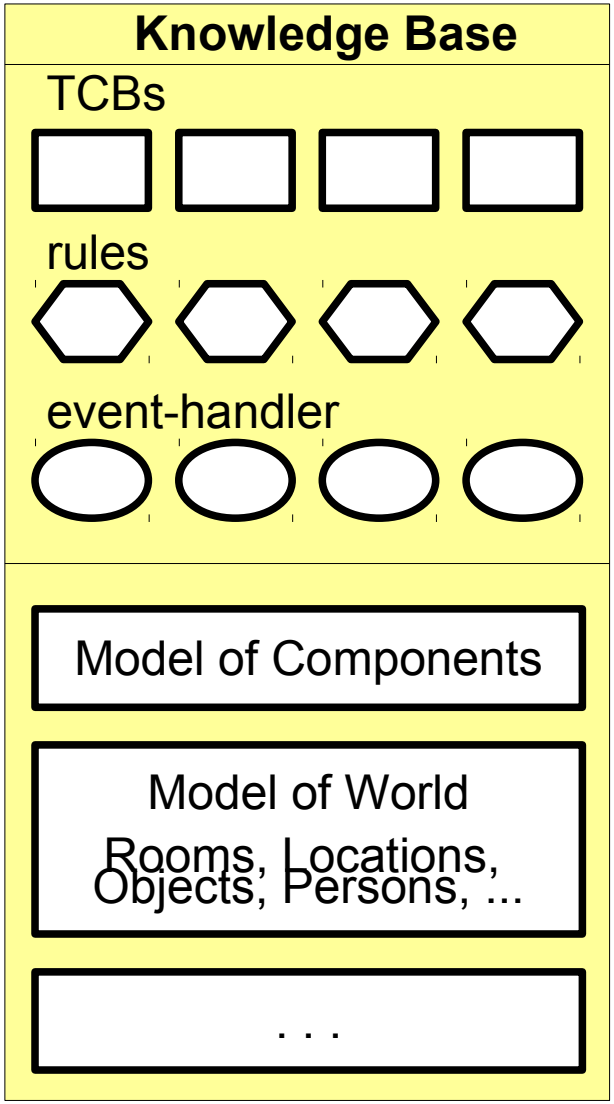
event-handler

Model of Components

Model of World
Rooms, Locations, Objects, Persons, ...

. . .

# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies

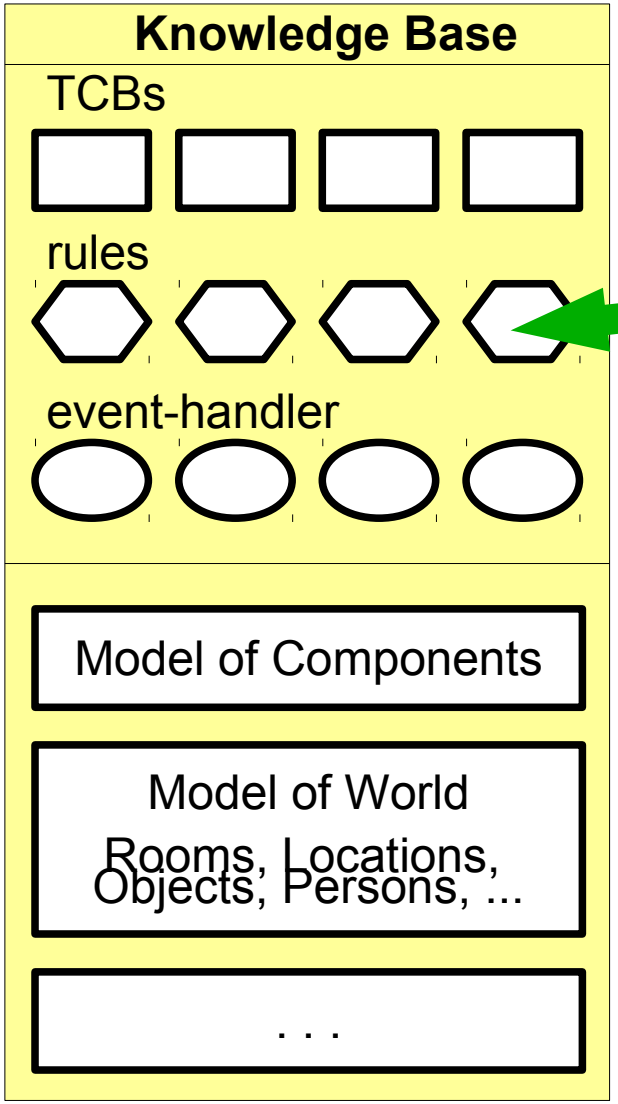






- contingencies are handled by the parent node
- return-values (e.g. errors) are passed along the node hierarchy similar as it is done by catch-throw mechanisms
- rules can modify the task-tree by adding or deleting nodes
- principle of subsidiarity

# Run-Time: Managing Execution Variants
## Rules to Recover from Contingencies

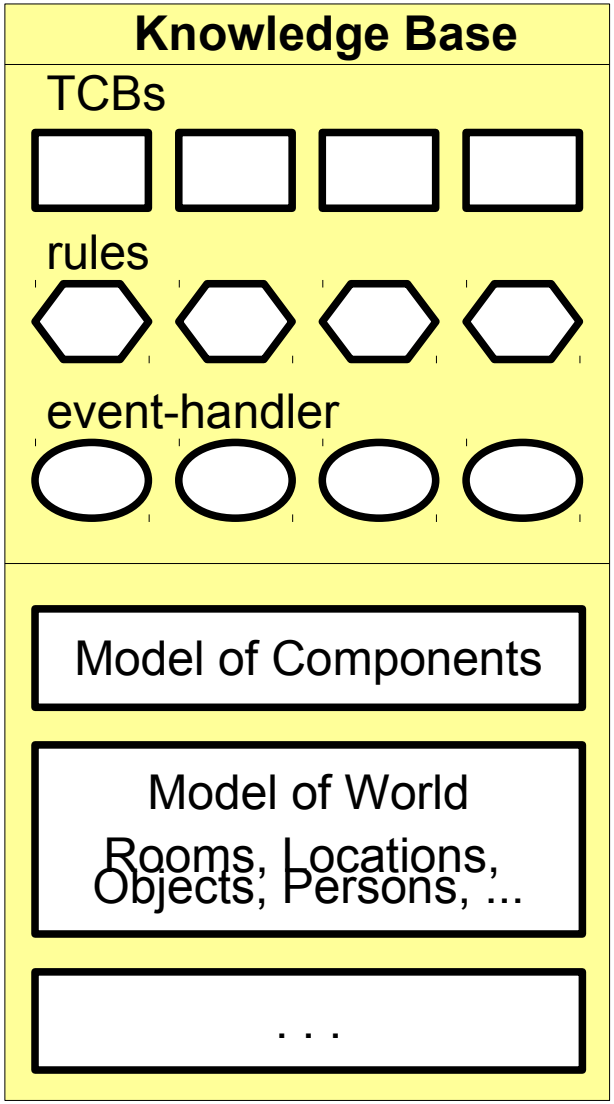


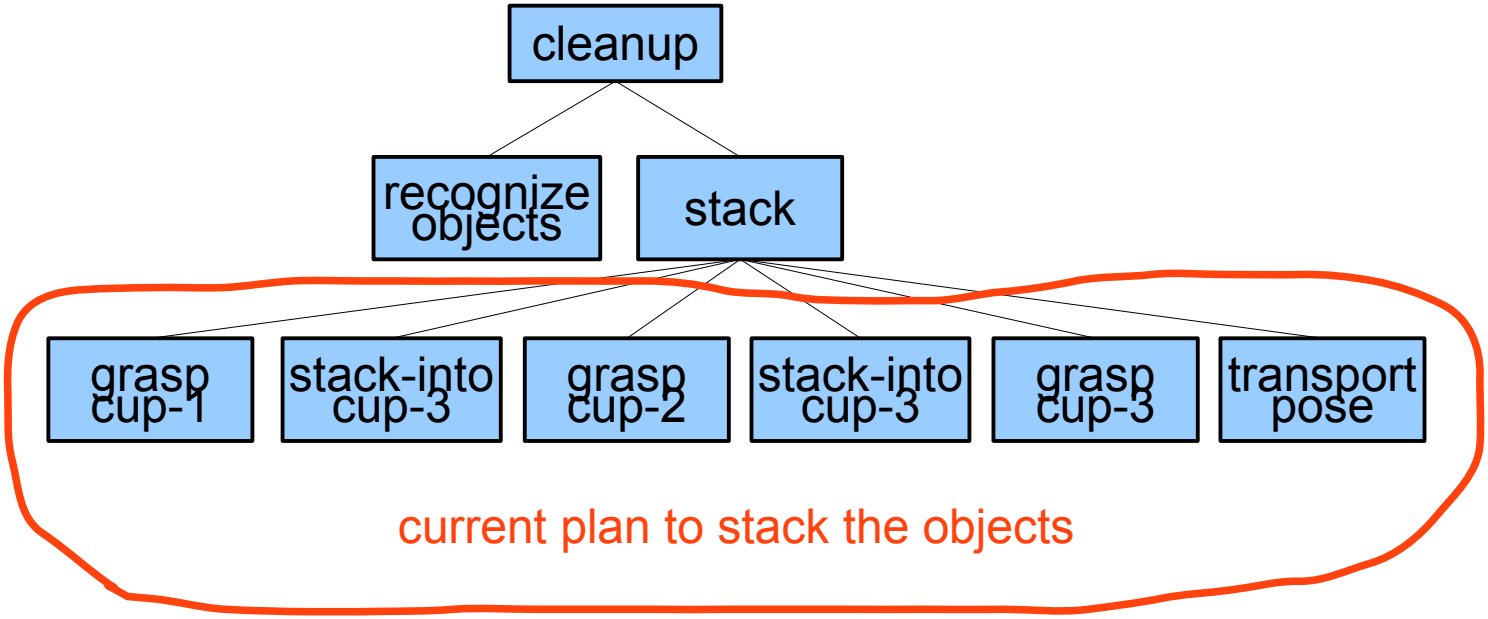




- contingencies are handled by the parent node
- return-values (e.g. errors) are passed along the node hierarchy similar as it is done by catch-throw mechanisms
- rules can modify the task-tree by adding or deleting nodes
- principle of subsidiarity

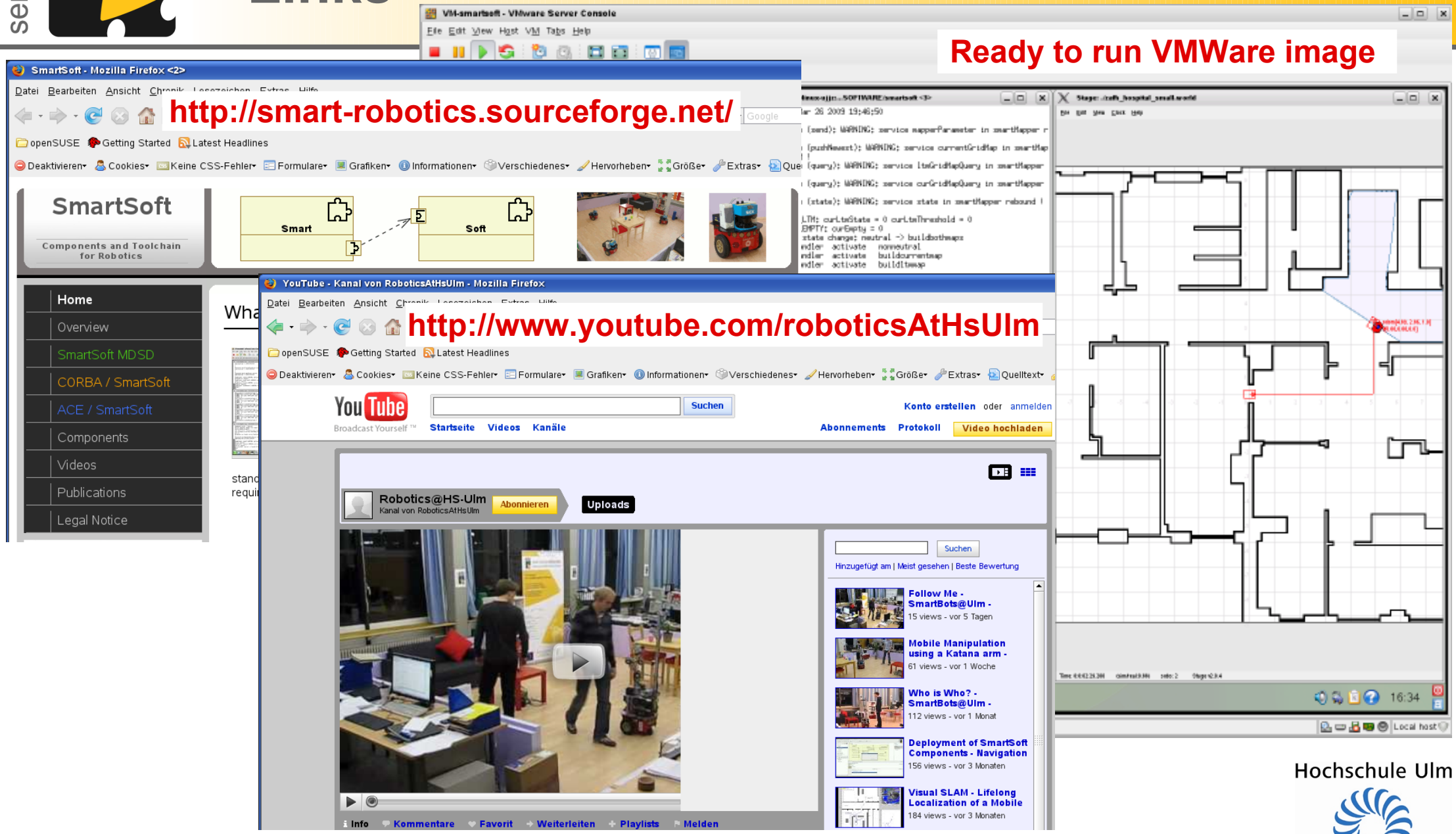
# Run-Time: Managing Execution Variants
## Handling Contingencies




- as rules are associated to the parent node (*stack*), the contingency handling works independent of the concrete plan which was generated
- rules “know” whether to repair the plan locally or to delete the plan and generate a new one.

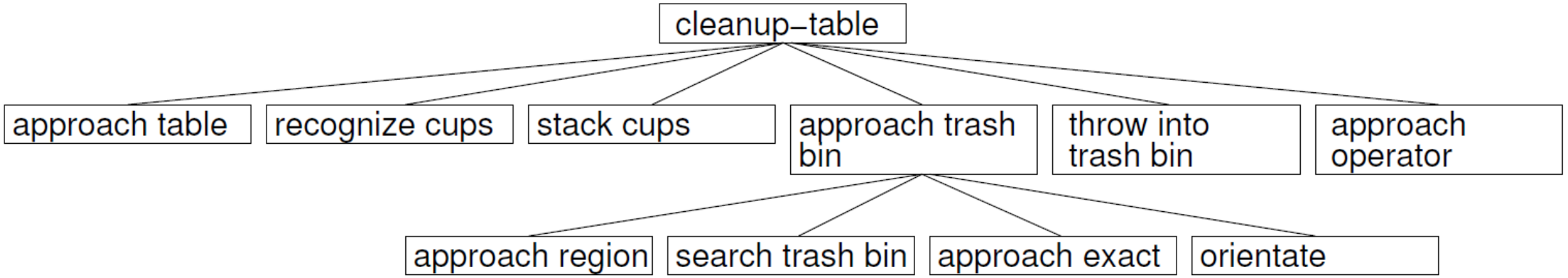
# SmartSoft MDSD Toolchain Links



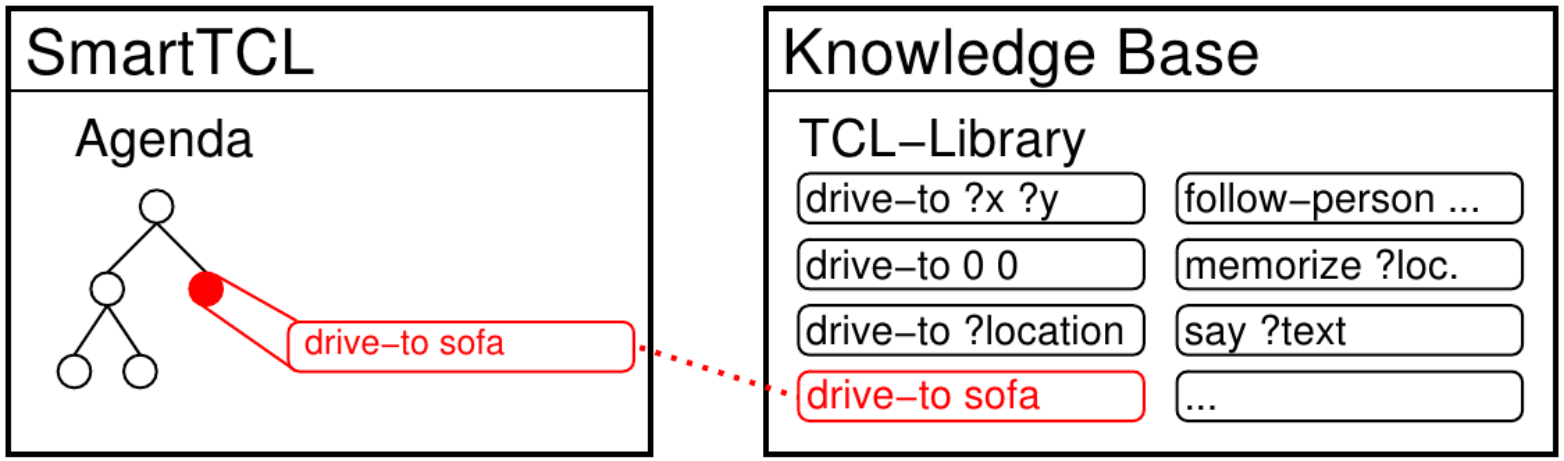

Hochschule Ulm

# Addendum

# Run-Time: Managing Execution Variants
## Hierarchical Task Decomposition



# Run-Time: Managing Execution Variants
## TCB Selection at Run-Time



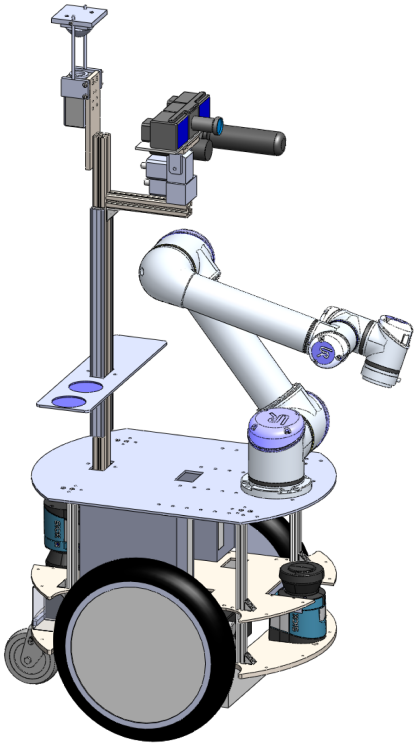

**Selection steps:**

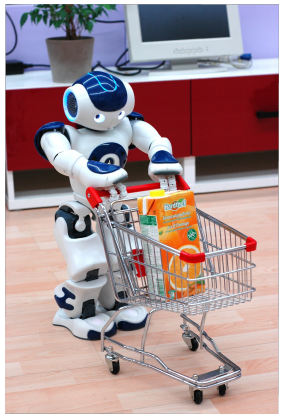
- name of TCB, number of input/output variables and binding of input variables is matched against TCBs stored in KB (***unification***)
- precondition clause is evaluated
- out of remaining TCBs the one with the highest priority is chosen

service robotik

ZAFH

# Mastering Execution Variants at Run-Time
# What is different in Robotics?

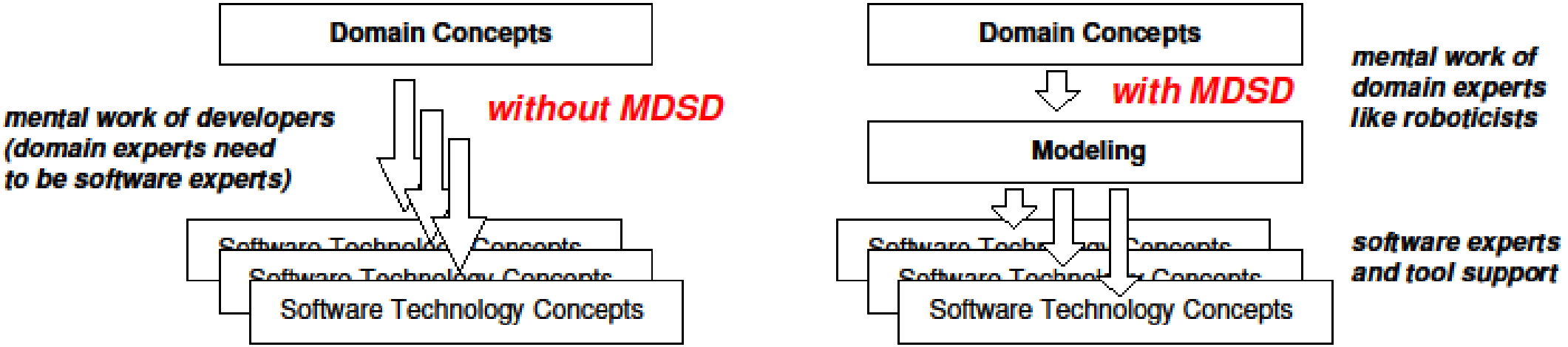





Hochschule Ulm

# Model Driven Software Development Service Robotics

- analysis models / requirement models are ***non-computational***, MDSD-models are ***computational***
- MDSD-models are not just paper work, they ***are*** the solution which can be translated into code via tool support
- *"freedom from choice"* instead of *"freedom of choice"*
- identification of stable structures and variability points



service robotik

# Model Driven Software Development
# Service Robotics

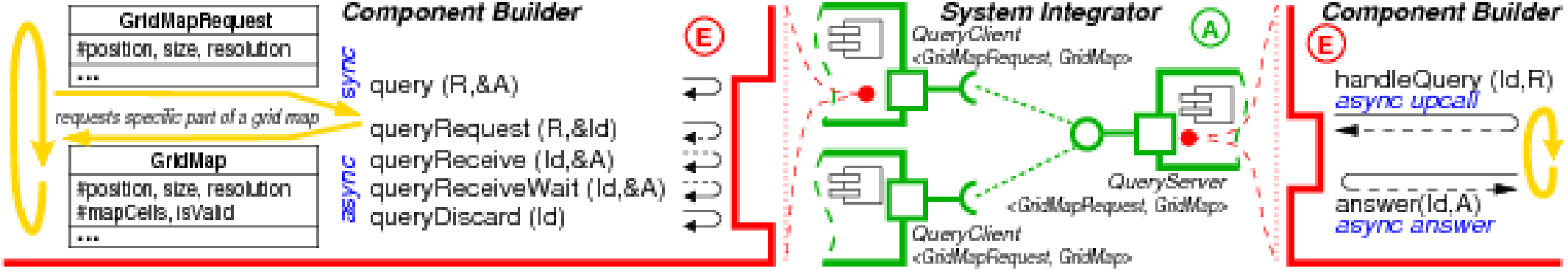
**The SmartSoft Communication Patterns**

| | |
|---|---|
| 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**

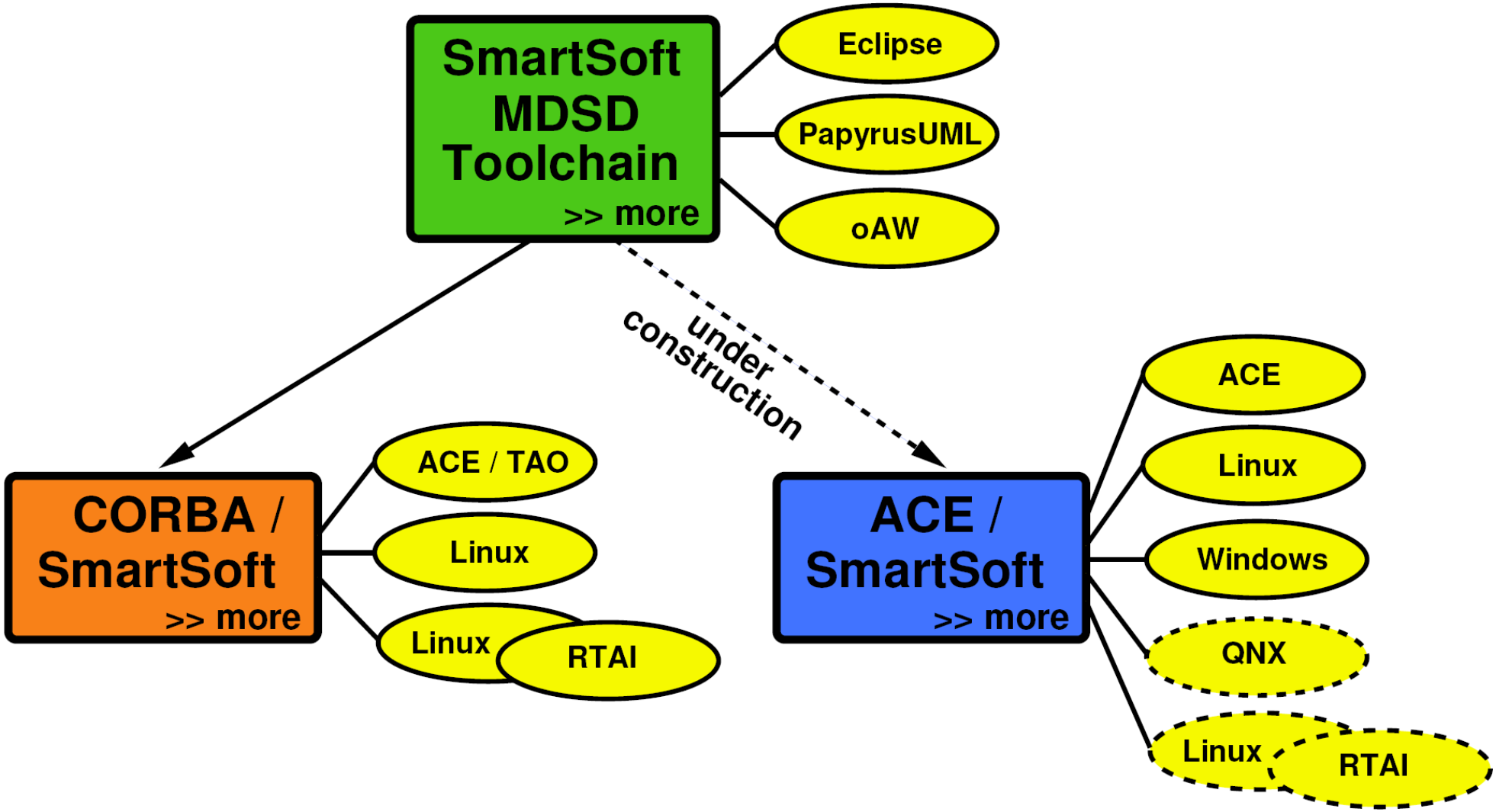
| | |
|---|---|
| param | component configuration |
| state | activate/deactivate component services |
| wiring | dynamic component wiring |
| diagnose | introspection of components |
| *(internally based on communication patterns)* | |

- **these communication patterns are sufficient since they support both, a *request/response* interaction as well as *asynchrone notifications / push services***
- **the communication patterns and their semantics is independent of the underlying / used middleware**
- **the communication patterns ensure a certain level of granularity of the services as well as the component architecture**




Hochschule Ulm

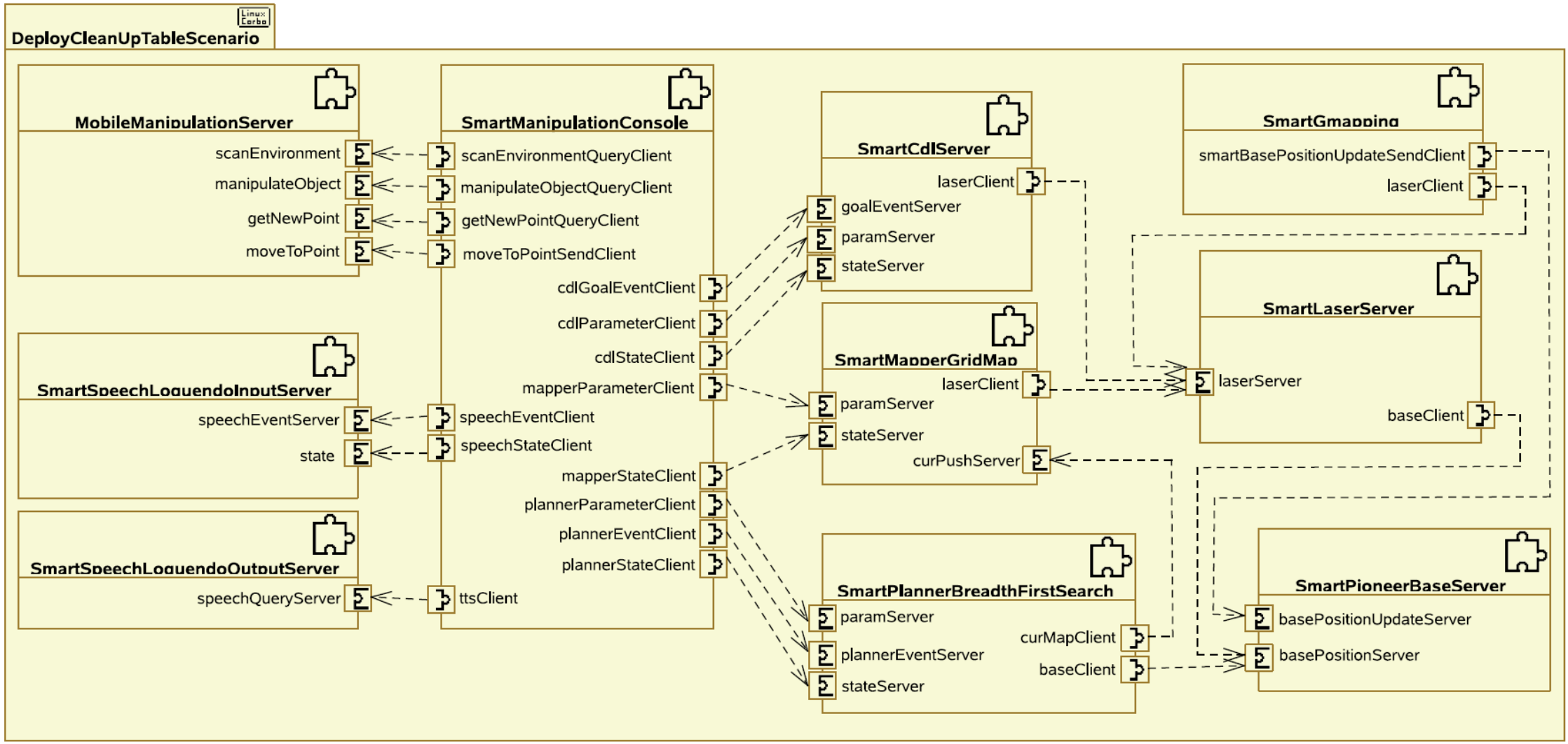
# Model Driven Software Development
# Service Robotics

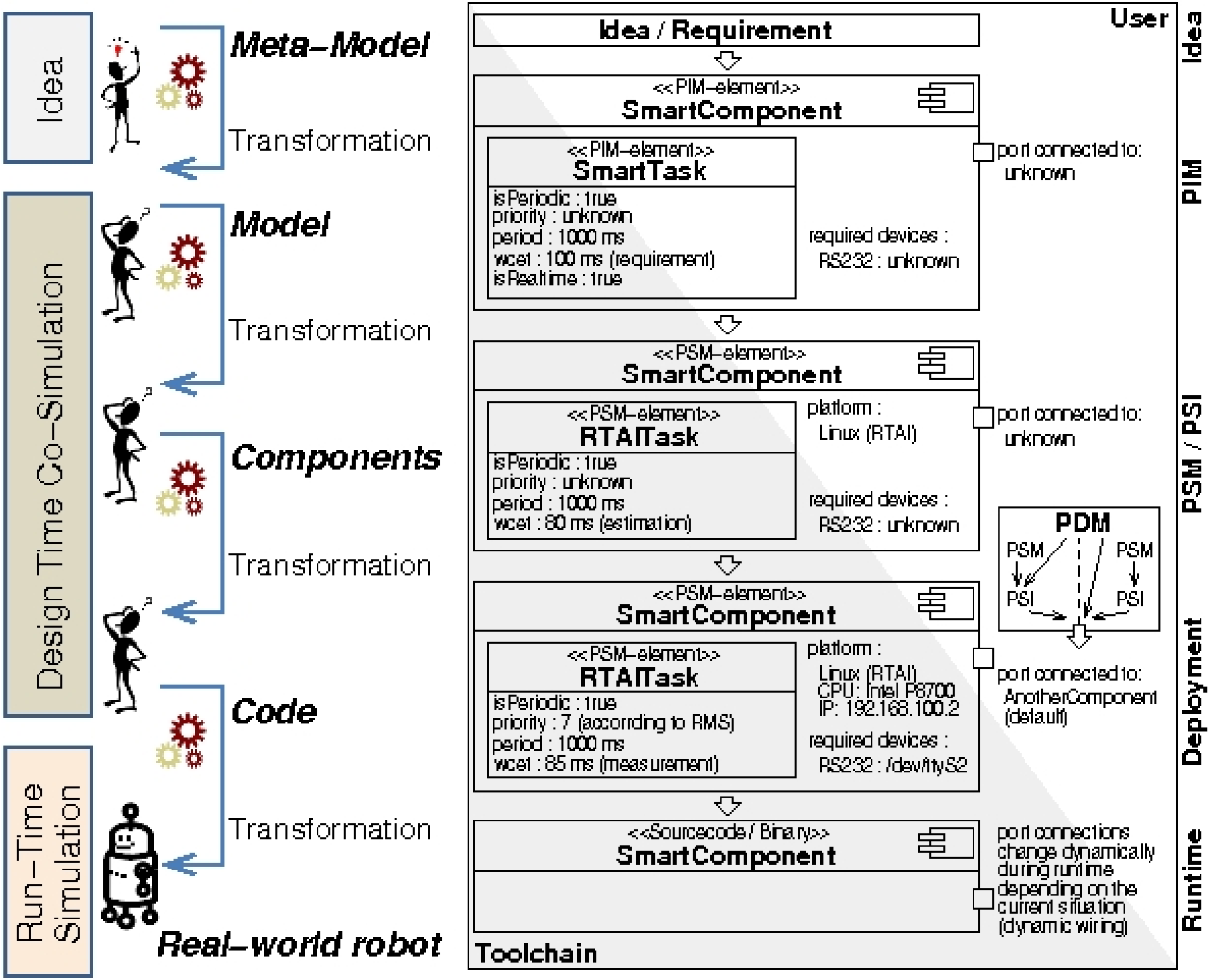



Hochschule Ulm

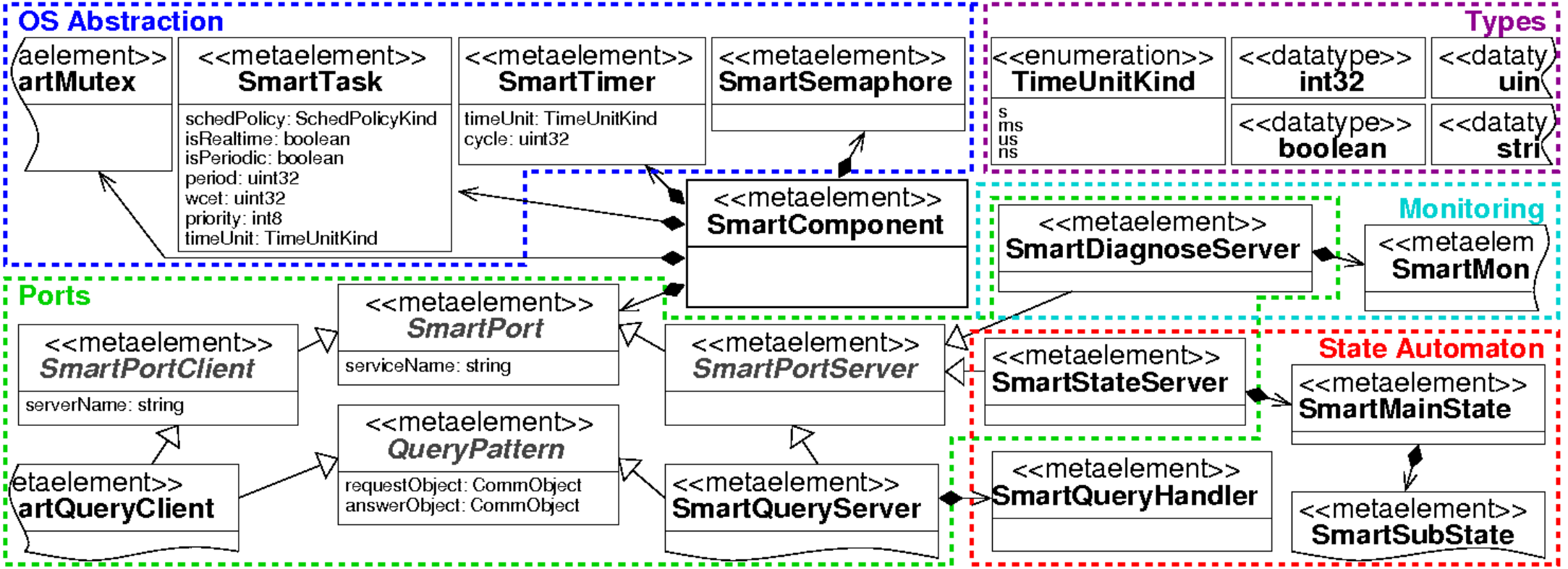
# Example: “Cleanup Table Scenario”






Idea
Design Time Co-Simulation
Run-Time Simulation
Meta-Model
Transformation
Model
Transformation
Components
Transformation
Code
Transformation
Real-world robot
Idea / Requirement
User
<<PIM-element>>
SmartComponent
<<PIM-element>>
SmartTask
isPeriodic : true
priority : unknown
period : 1000 ms
wcet : 100 ms (requirement)
isRealtime : true
required devices :
RS232 : unknown
port connected to:
unknown
<<PSM-element>>
SmartComponent
<<PSM-element>>
RTAITask
isPeriodic : true
priority : unknown
period : 1000 ms
wcet : 80 ms (estimation)
platform :
Linux (RTAI)
required devices :
RS232 : unknown
port connected to:
unknown
PDM
PSM
PSM
PSI
PSI
<<PSM-element>>
SmartComponent
<<PSM-element>>
RTAITask
isPeriodic : true
priority : 7 (according to RMS)
period : 1000 ms
wcet : 85 ms (measurement)
platform :
Linux (RTAI)
CPU: Intel P8700
IP: 192.168.100.2
required devices :
RS232 : /dev/ttyS2
port connected to:
AnotherComponent
(default)
<<Sourcecode / Binary>>
SmartComponent
port connections
change dynamically
during runtime
depending on the
current situation
(dynamic wiring)
Toolchain
Idea
PIM
PSM / PSI
Deployment
Runtime

# Model Driven Software Development Service Robotics



service robotik

# Model Driven Software Development Service Robotics