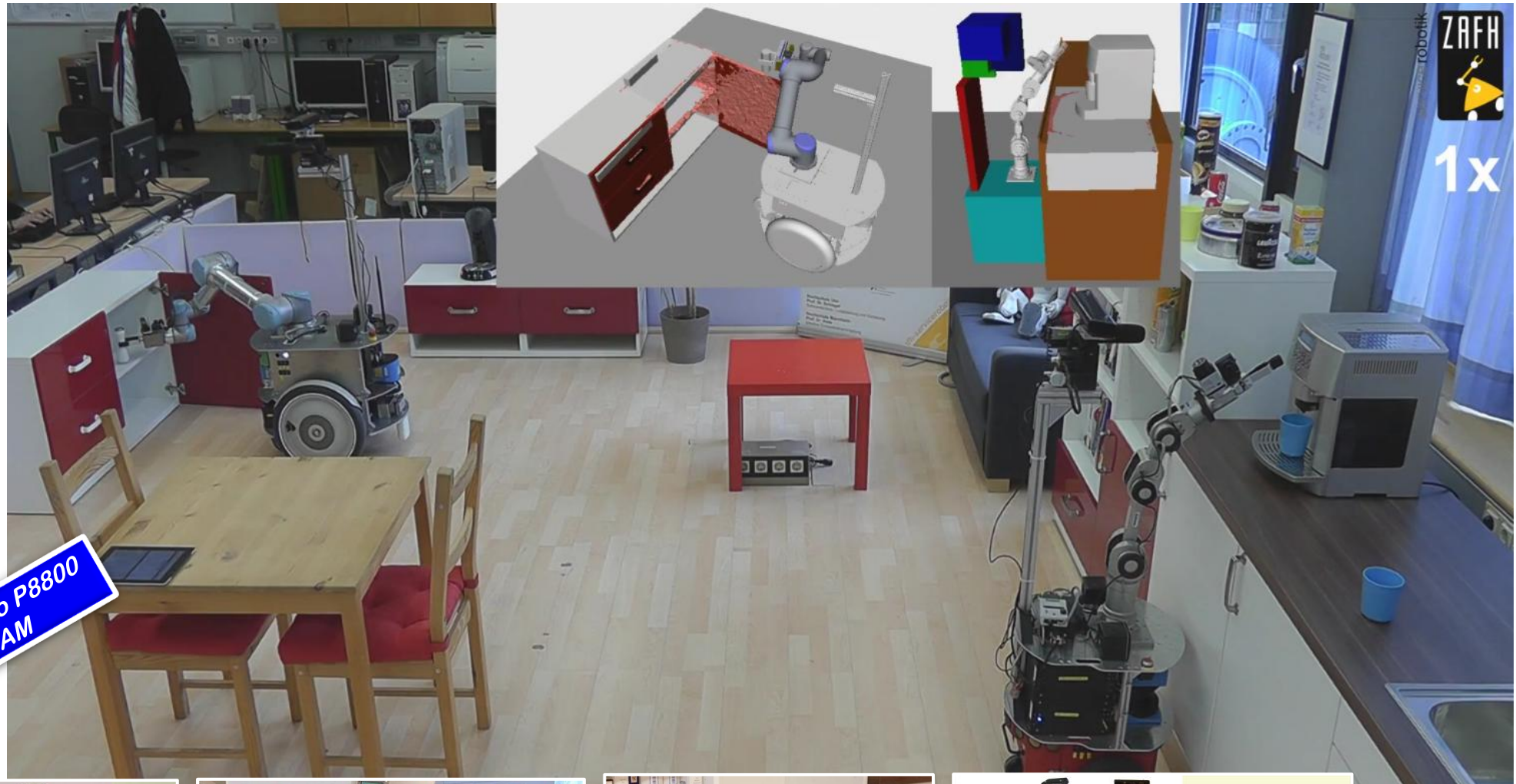
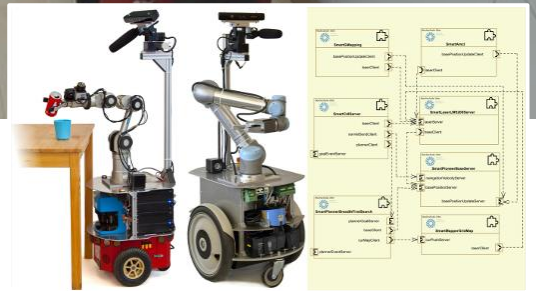
The SmartMDSD Toolchain: Supporting dynamic reconfiguration by managing variability in robotics software development

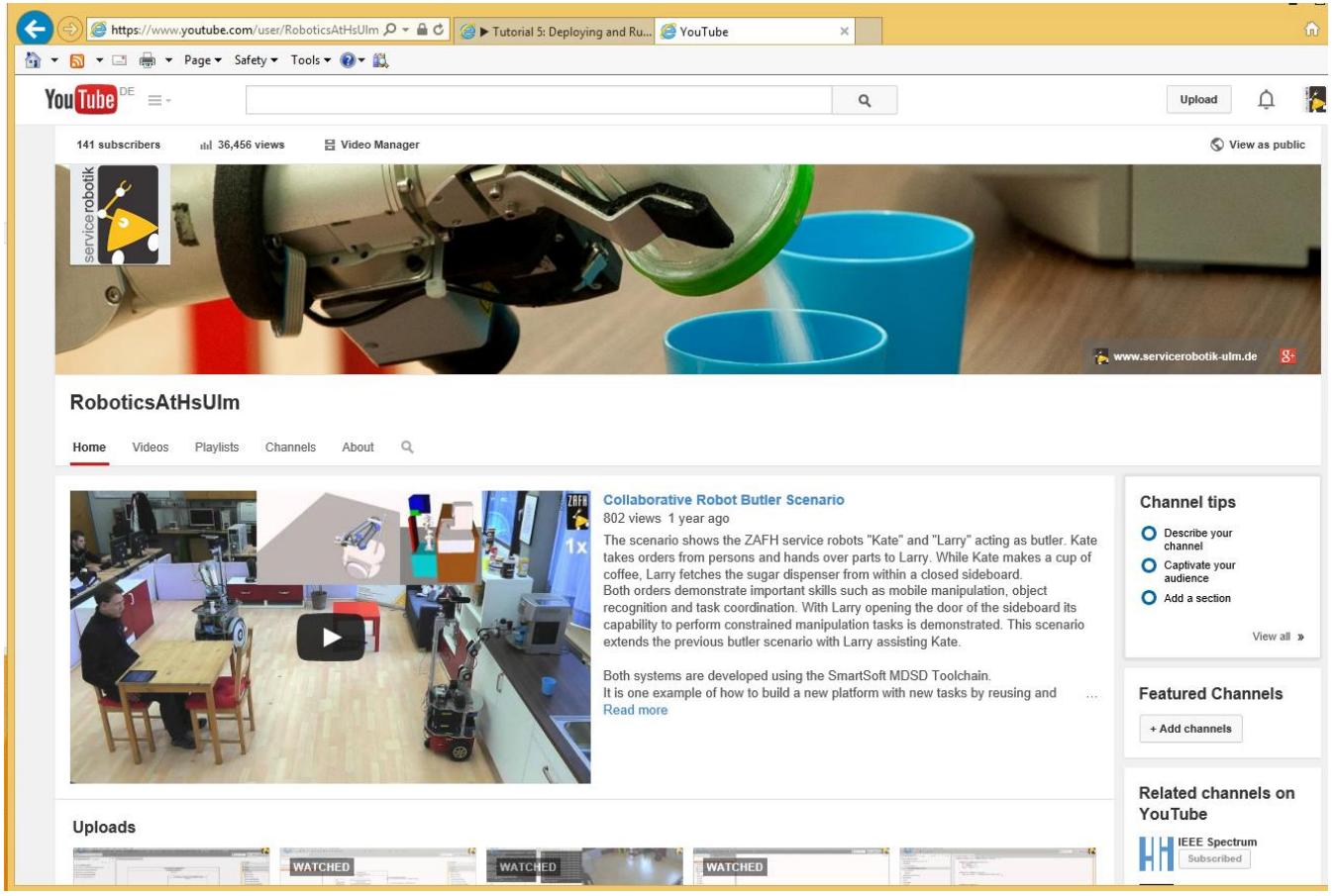
Christian Schlegel, Dennis Stampfer

Cooperative Robot Butler Scenario





Cooperative Robot Butler Scenario: Best Of Video



http://youtu.be/OmubgOiSAkc

This video shows clips of the 30-minute "Butler Scenario". The service robot "Kate" takes orders from persons and delivers coffee and juice. The orders demonstrate methods and research results of the Service Robotics Lab at University of Applied Sciences Ulm.

Business Ecosystem / Business Community

A business ecosystem describes the structure and behaviour of a network of high-tech organisations that share a key technological platform and the ways individual firms can flourish in such an environment.
[Moore, James F., 1993]

A business ecosystem is “a dynamic structure which consists of an interconnected population of organisations. A business ecosystem develops through self-organisation, emergence and co-evolution, which help it to acquire adaptability. In a business ecosystem there is both competition and cooperation present simultaneously”
[Peltoniemi & Vuori2005]

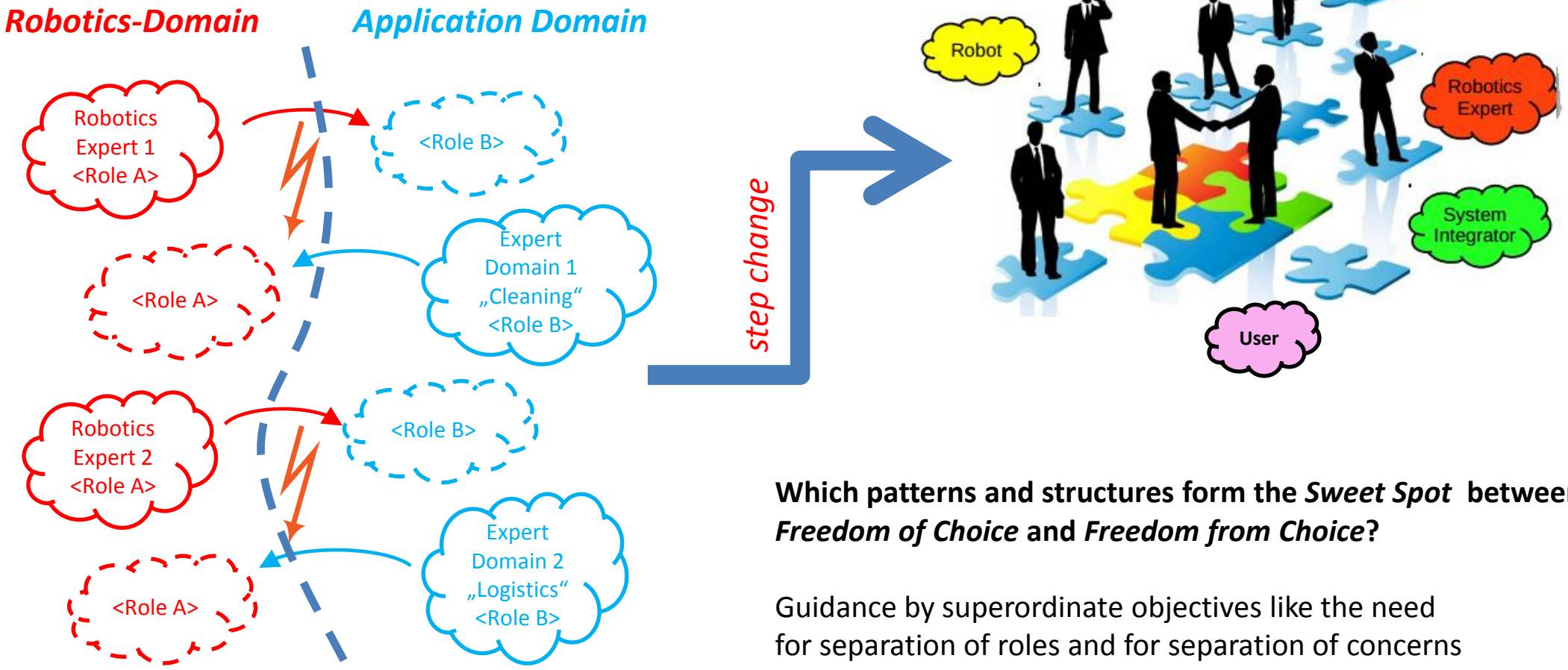
Benefits:

- *share and lower risks, efforts & costs*
- *improve robustness, quality, time-to-market, cost-efficiency*
- *agile and effective collaboration to support commercialization*
- *easy entry for niche players to effectively fill gaps and to evolve*
- *delay design decisions to the latest point that is economically feasible*

Structures:

- *Separation of Roles*
- *Separation of Concerns*
- *Composability and Composition of building blocks*
- *Freedom of Choice versus Freedom from Choice (Standards)*

Business Ecosystem / Business Community

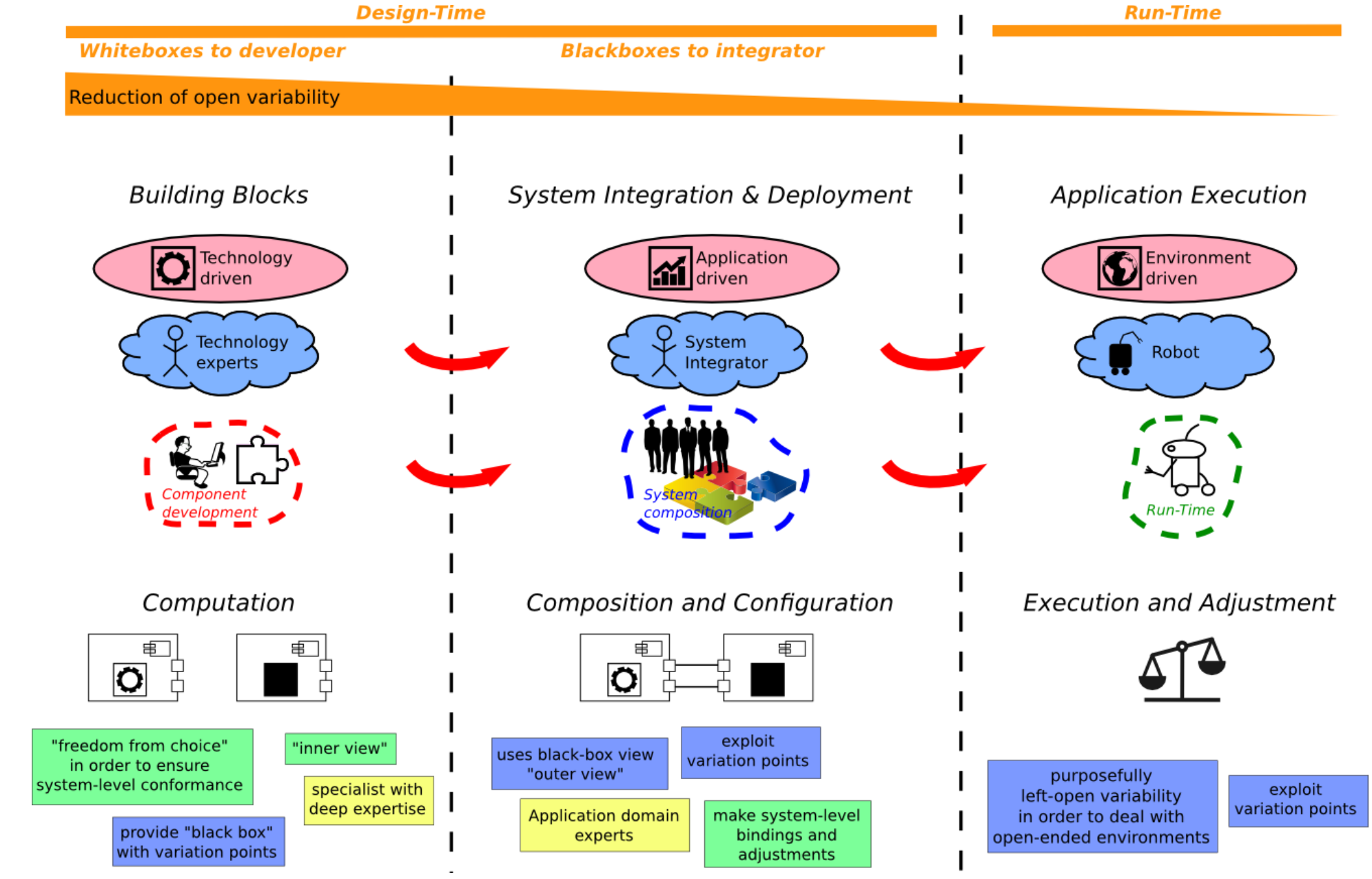


Which patterns and structures form the *Sweet Spot* between *Freedom of Choice* and *Freedom from Choice*?

Guidance by superordinate objectives like the need for separation of roles and for separation of concerns

Support as much freedom as possible while still ensuring composability despite separation of roles

Variability Management in the Lifecycle



Variability Management in the Lifecycle

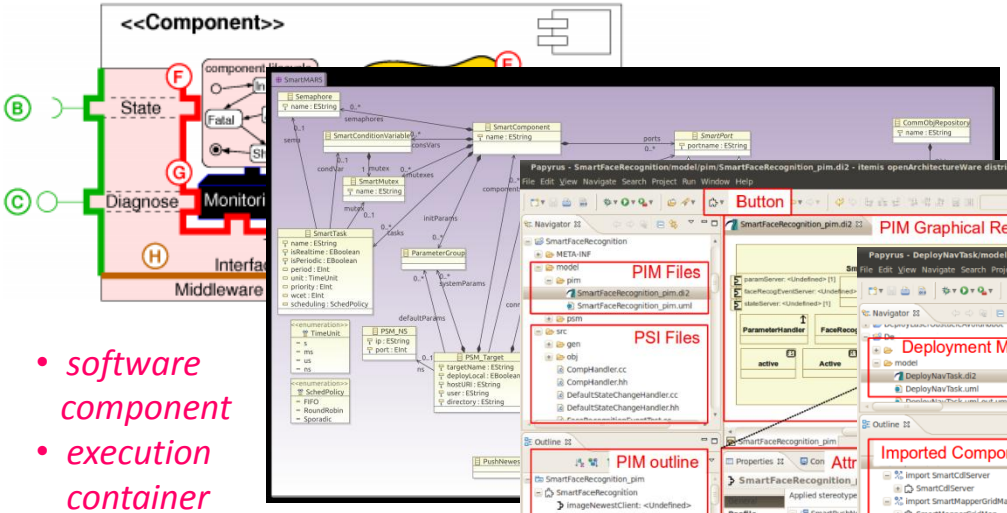
SmartMDSD
(service oriented component model)
- Meta-Model
- Toolchain

SmartSoft
(implementation)
- CORBA / SmartSoft
- ACE / SmartSoft
- Linux, Windows, etc.

SmartTCL
(Task Coordination Language)

VML
(Variability Modeling Language)

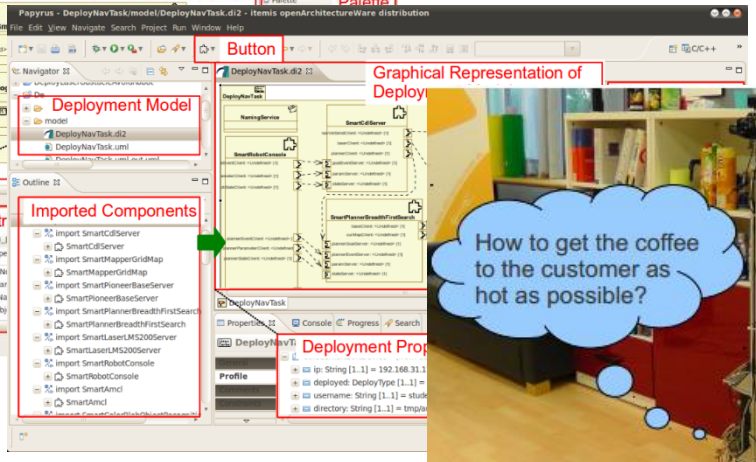
Domain Specific Languages



- *software component*
- *execution container*

Meta-Model

Eclipse-Toolchain
- *component developer*

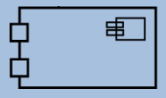
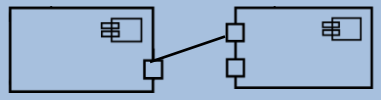


Eclipse-Toolchain
- *system integrator*
- *deployment*

Real robot in real world

Part II

Software Variability and Runtime Reconfiguration

	Design Time		Run-Time
	Technology Driven	Application Driven	Environment Driven
Software Variability:	 deliver product with explicated variation points.	***Components*** ***Services*** ***System of Components*** ***Meta Model***	 Exploit and bind left open variation points.
Composability			
Variability in Operation:	task sequencing, composable subtasks	***DSL***	**How to do it?** • pure task achievement • robot functionality
			
Maintain a high success rate in task fulfillment => sequencing actions			
Variability in Quality:	contexts, rules, properties, variation points	***DSL***	**What is a good way to do it?** • do it efficiently • which possibilities are better than others in terms of non-functional properties
			
Improve overall execution quality => optimizing non-functional properties			

Software Variability and Runtime Reconfiguration

Design Time

Technology Driven | Application Driven

Run-Time

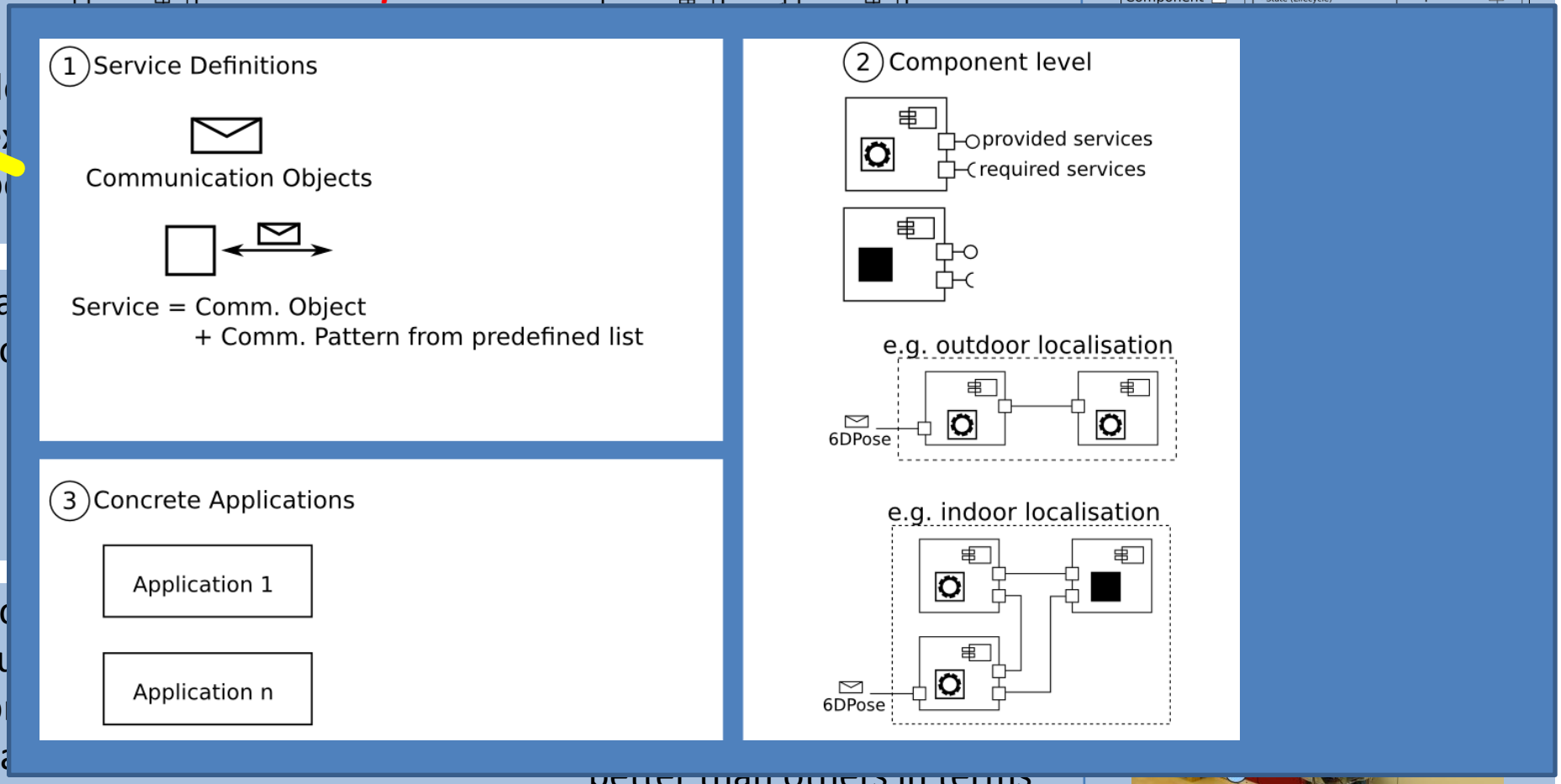
Environment Driven

Software Variability:

Variability in Operation:

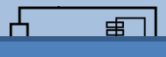
Variability in Quality:

Components



of non-functional properties

Software Variability and Runtime Reconfiguration

	Design Time		Run-Time
	Technology Driven	Application Driven	Environment Driven
Software Variability:	 **Components**  		
Variability in Operation:			
Variability in Quality:			

Services as basic architectural entities:

- guarantee that supplied components can be integrated into concrete applications
- keep the architecture and implementation containers flexible
- allow to identify white spots as early as possible (required services that no-one provides or provided services that no-one requires).

Component Level:

- aggregates services to components or groups of components
- might contain competing alternative components providing the same services from different suppliers with different characteristics
- both open source and closed source implementations can be used since one can rely on service descriptions
- even though components implement functionalities, the granularity of components is arbitrary and not decisive for the architecture. Component granularity might differ from application to application, whereas the granularity of services can remain the same.

Concrete Applications:

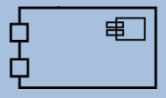
- are composed from building blocks

Software Variability and Runtime Reconfiguration

Desig

Technology Driven

Software Variability:



deliver product with explicated variation points.

Compo

Ser

Sys

Com

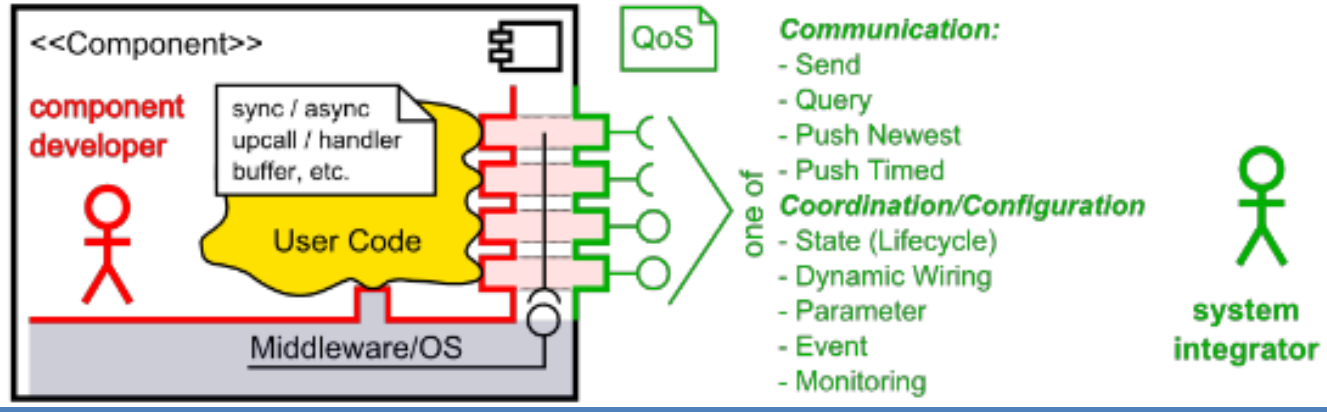
Met

Variability in Operation: task sequencing, composable subtasks

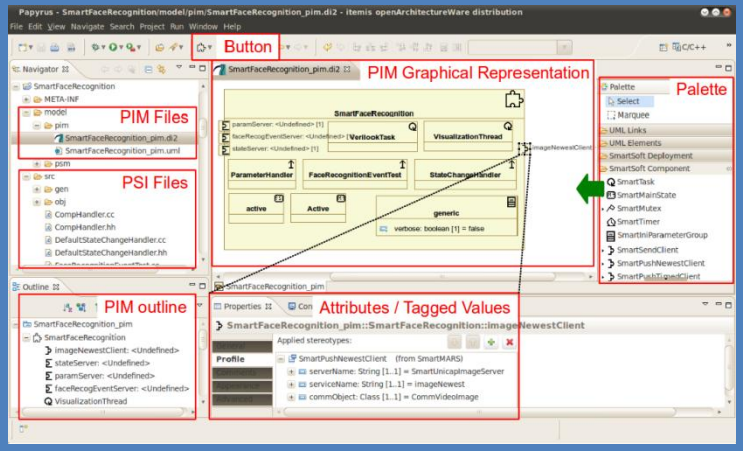
Variability in Quality: contexts, rules, properties, variation points

D

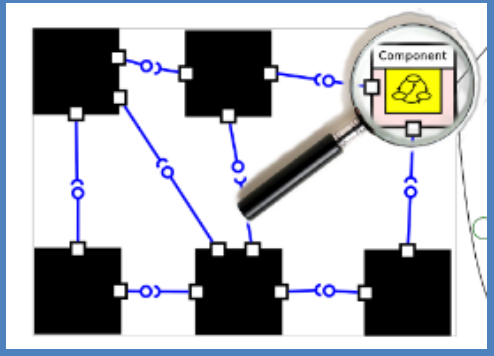
Service-oriented Component Model and MDSD-workflow



Eclipse based SmartMDSD Toolchain



Black-Box View



Software Variability and Runtime Reconfiguration

Design Time | Run-Time

Technolog

Software Variability:

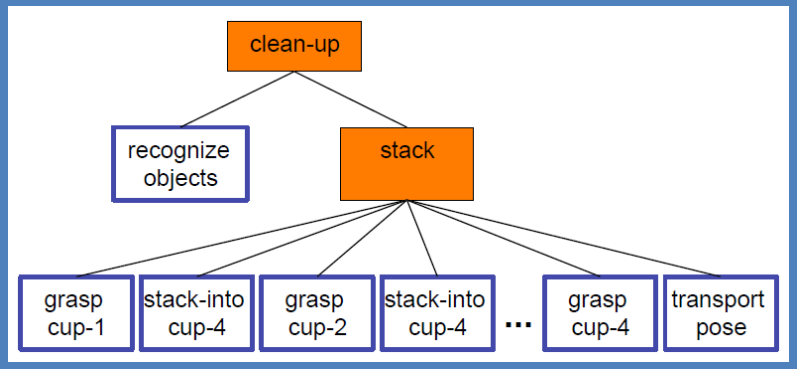
deliver pr
explicated
points.

Variability in Operation: task seque
compos

Variability in Quality: contexts, rules, properties, variation points

DSL

SmartTCL: Task Coordination Language




```
(define-tcb (cleanup-table ?location)
  (rules (rule-action-cleanup-table-failed rule-action-cleanup-table-empty
    rule-action-cleanup-say-success ))
  (plan (
    (tcb-approach-location ?location)
    (tcb-say "I have reached the table, and will look for objects to clean up.")
    (tcb-cleanup-table))))
```

What is a good way to do it?

- do it efficiently
- which possibilities are better than others in terms of non-functional properties

Software Variability and Runtime Reconfiguration

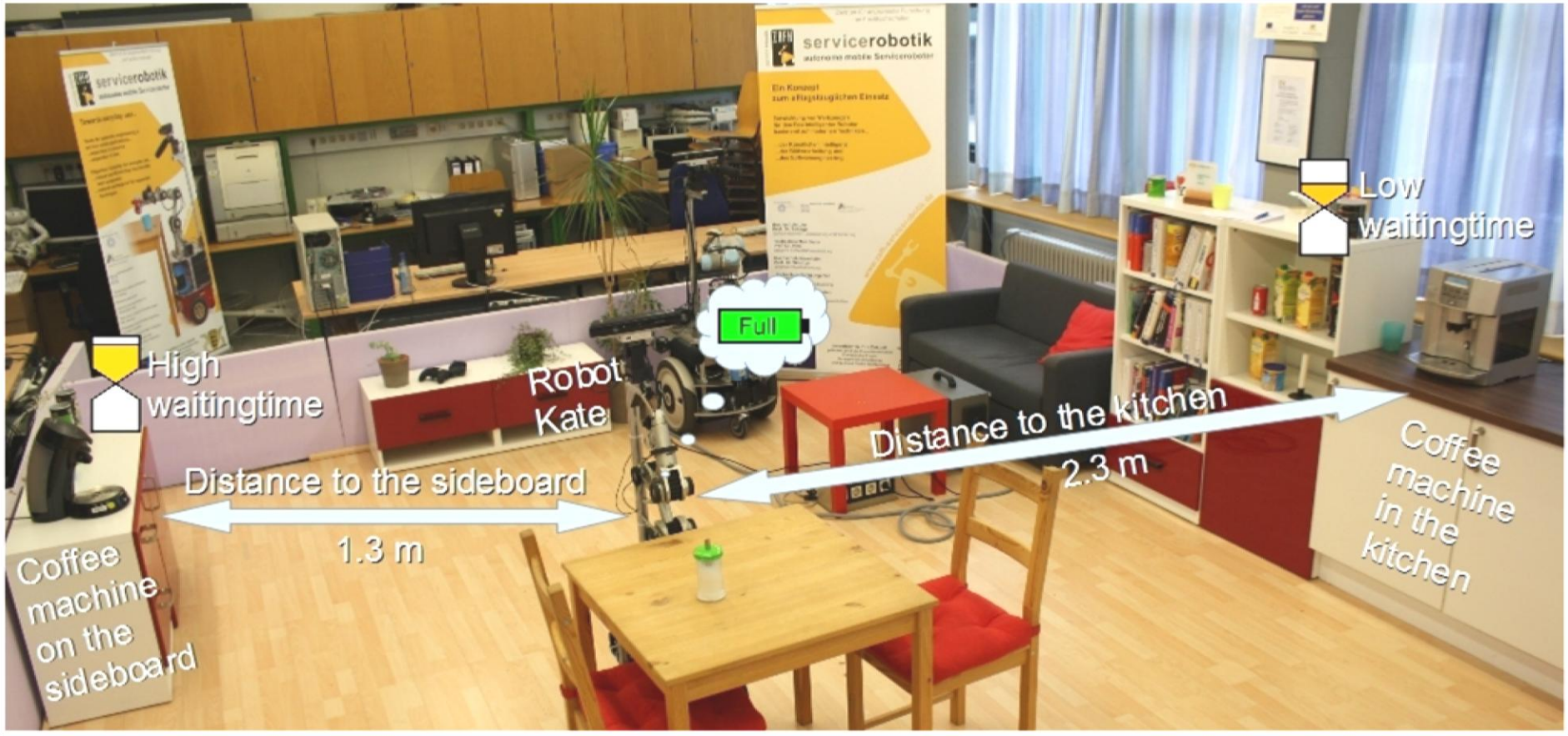
Software Variability

Variability in Operati

Variability in Quality:

rules,
properties,
variation po

VML: Variability Modeling Language




```
  priorities:
    f(ctx_battery) = exp(-1 * ctx_battery/15);
  definitions:
    f(maximumVelocity) = exp(maximumVelocity/150);
}
```

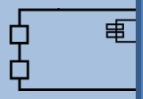
```
/* Variation points */
varpoint maximumVelocity : number [100:10:600];
varpoint coffeeMachine   : enum { COFFEE_MACHINE_A, COFFEE_MACHINE_B };
```

Variation points

Software Variability and Runtime Reconfiguration

Technology

Software Variability:



deliver prod
explicated v
points.

Variability in Operation: task sequen
composable

Variability in Quality: contexts, rules, properties, variation po

VML: Variability Modeling Language

```
context ctx_battery                : number [0:1:100];
context ctx_distanceMachine_A      : number [0:0.1:20];
context ctx_distanceMachine_B      : number [0:0.1:20];
context ctx_waitingTimeMachine_A   : number [10:1:300];
context ctx_waitingTimeMachine_B   : number [10:1:300];
```

Contexts

```
/* Auxiliary variables */
var timeMachine_A := ctx_waitingTimeMachine_A + ctx_distanceMachine_A/600;
var timeMachine_B := ctx_waitingTimeMachine_B + ctx_distanceMachine_B/600;
```

```
/* ECA rules */
rule lowBattery_NearMachineA :
   ctx_battery < 15 and ctx_distanceMachine_A < ctx_distanceMachine_B
   => coffeeMachine = @coffeeMachine.COFFEE_MACHINE_A;
rule lowBattery_NearMachineB :
   ctx_battery < 15 and ctx_distanceMachine_A >= ctx_distanceMachine_B
   => coffeeMachine = @coffeeMachine.COFFEE_MACHINE_B;
rule high_EFF_coffeeMachA :
   ctx_battery >= 15 and timeMachine_A  > timeMachine_B
   => coffeeMachine = @coffeeMachine.COFFEE_MACHINE_A;
rule high_EFF_coffeeMachB :
   ctx_battery >= 15 and timeMachine_A <= timeMachine_B
   => coffeeMachine = @coffeeMachine.COFFEE_MACHINE_B;
```

Rules

Universidad Politécnica de Cartagena

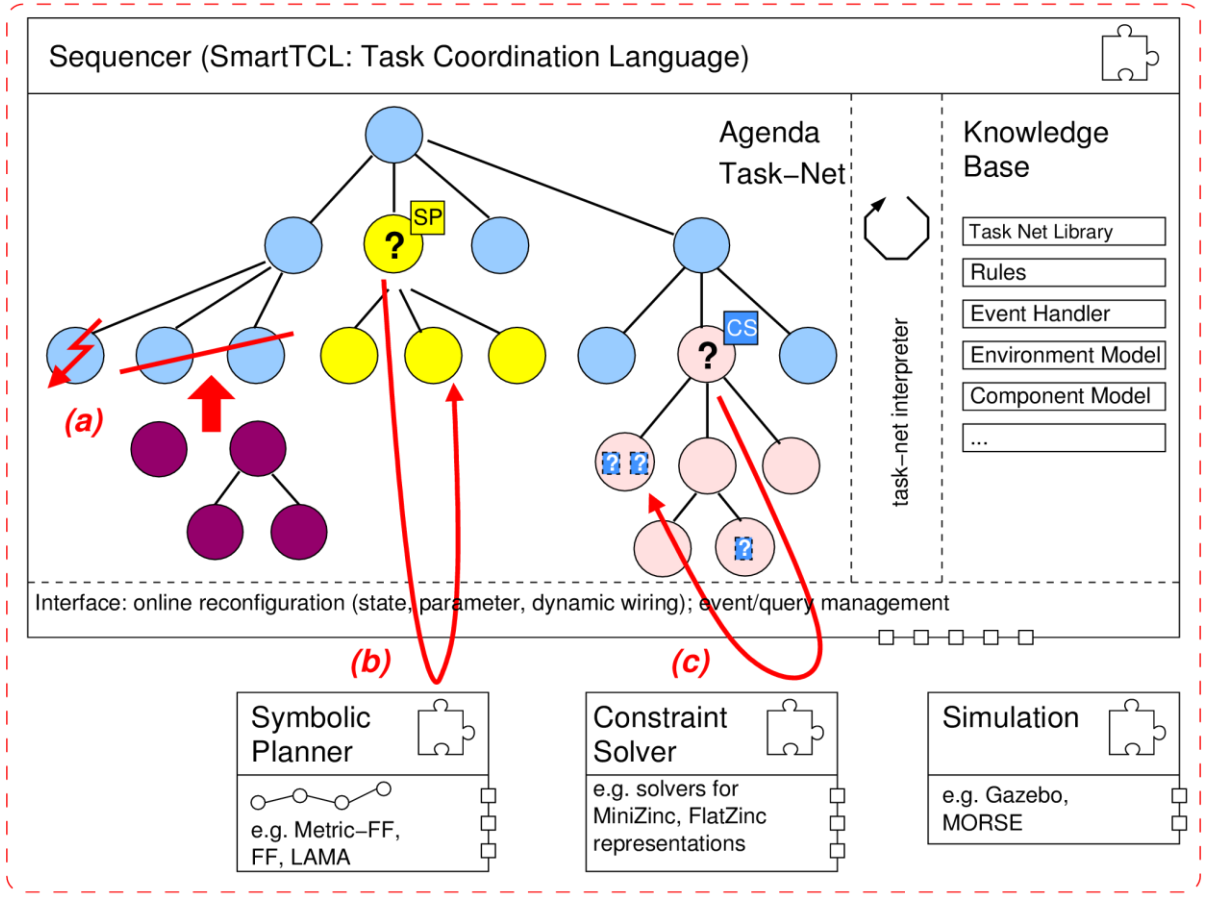
```
/* Properties to optimize */
property performance : number[0:1:100] maximized {
  priorities:
    f(ctx_battery) = max(exp(-1*ctx_battery/15), 10 sec) - exp(-1*ctx_battery/15);
  definitions:
    f(maximumVelocity) = maximumVelocity;
}
property powerConsumption : number[0:1:100] minimized {
  priorities:
    f(ctx_battery) = exp(-1 * ctx_battery/15);
  definitions:
    f(maximumVelocity) = exp(maximumVelocity/150);
}
```

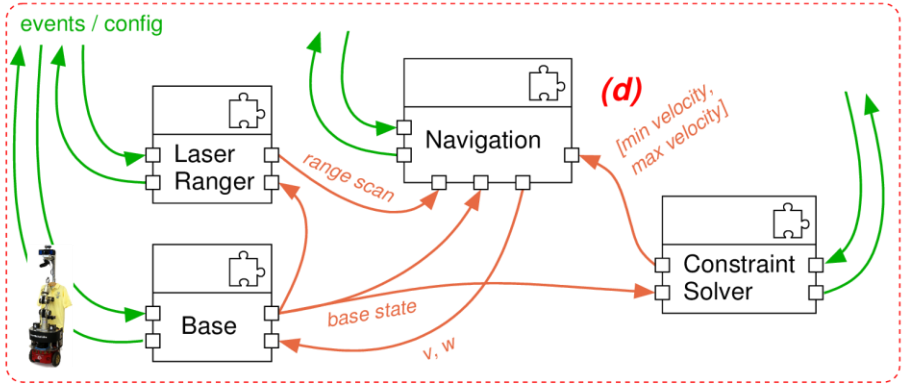
Properties

```
/* Variation points */
varpoint maximumVelocity : number [100:10:600];
varpoint coffeeMachine   : enum { COFFEE_MACHINE_A, COFFEE_MACHINE_B };
```

Variation points

System Architecture: Execution Variants at Run-Time





Integration of "Variability in Task **Sequencing**" and "Variability in Task **Execution Quality**"

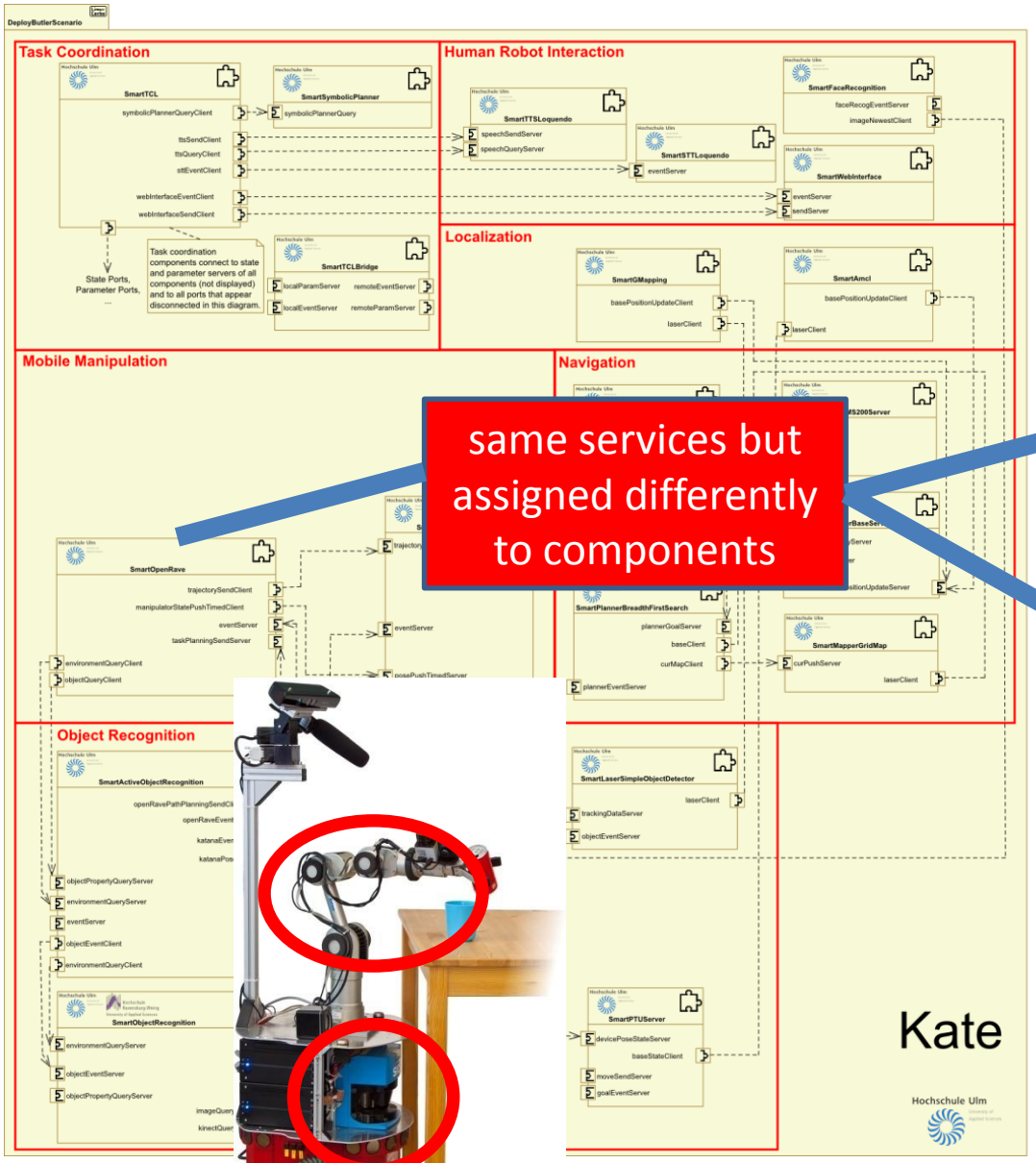
(a) SmartTCL handles a contingency by exchanging a sub-tree

(b) SmartTCL uses a symbolic planner to refine a sub-tree

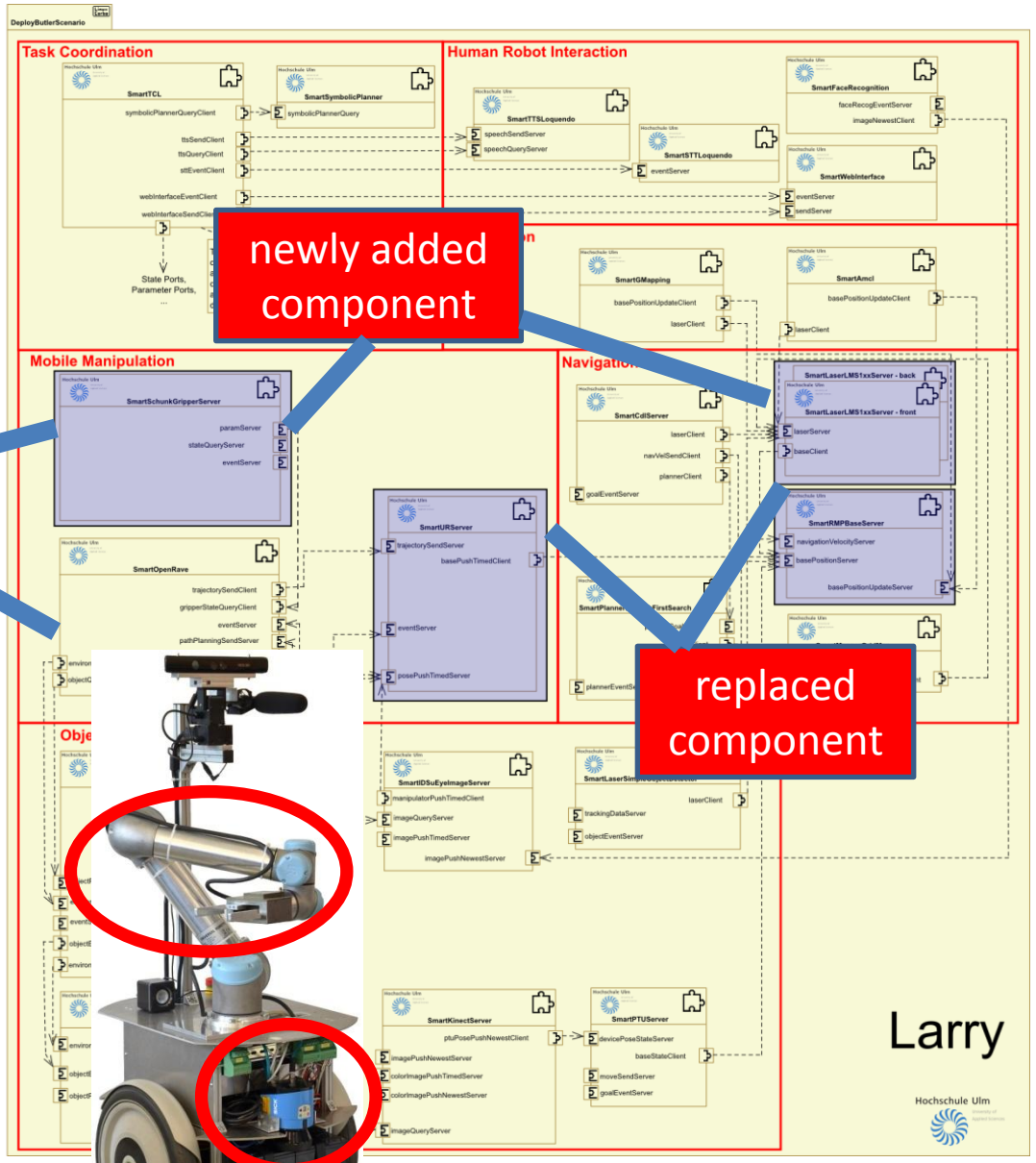
(c) SmartTCL triggers a constraint solver which executes the VML models

(d) VML binds left open variation point "max velocity" as a continuous service

Software Variability and Runtime Reconfiguration



same services but assigned differently to components



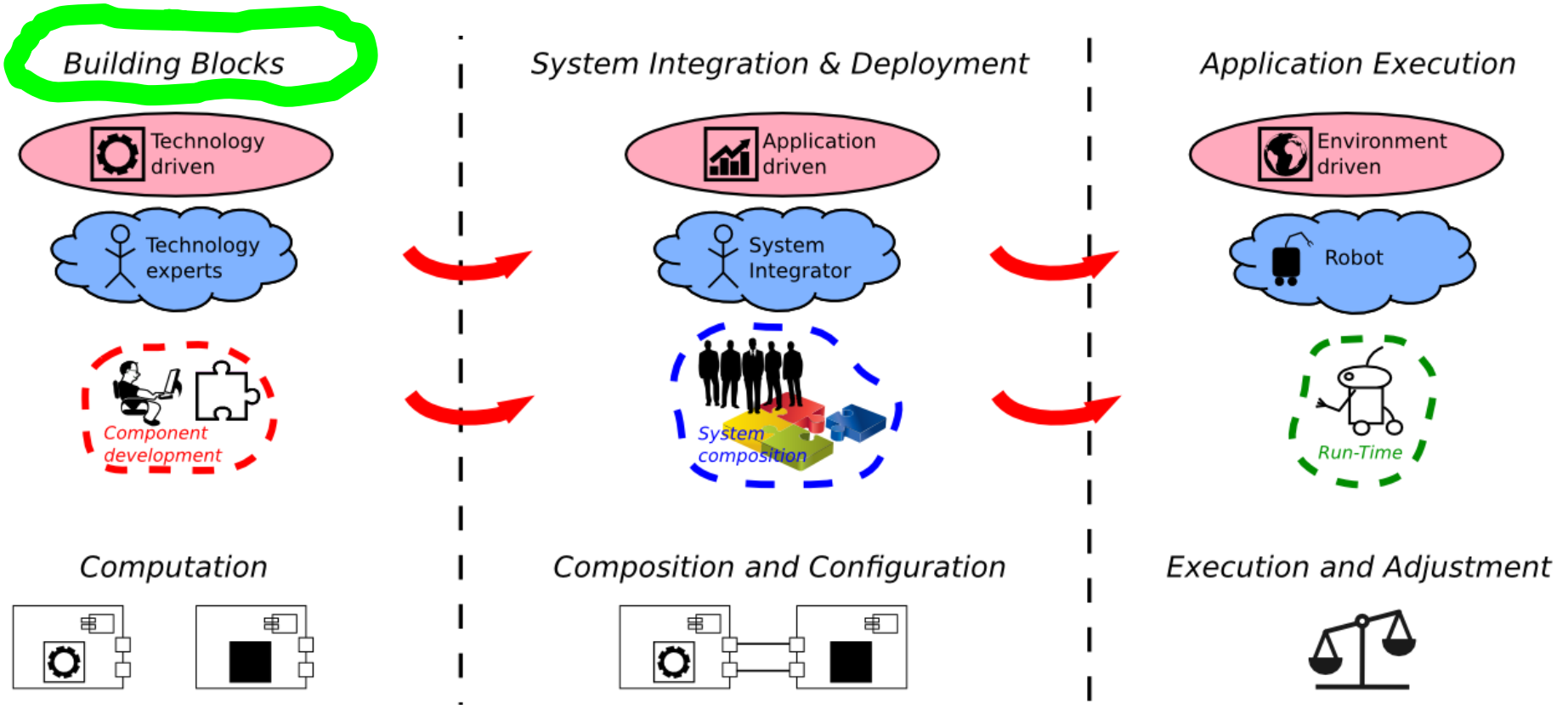
newly added component

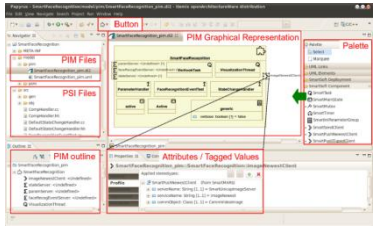
replaced component

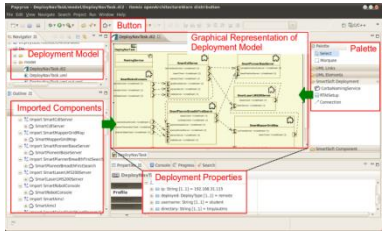
http://youtu.be/DjjNUPpj36E

Part III

SmartMDSD Toolchain: Hands-On Example







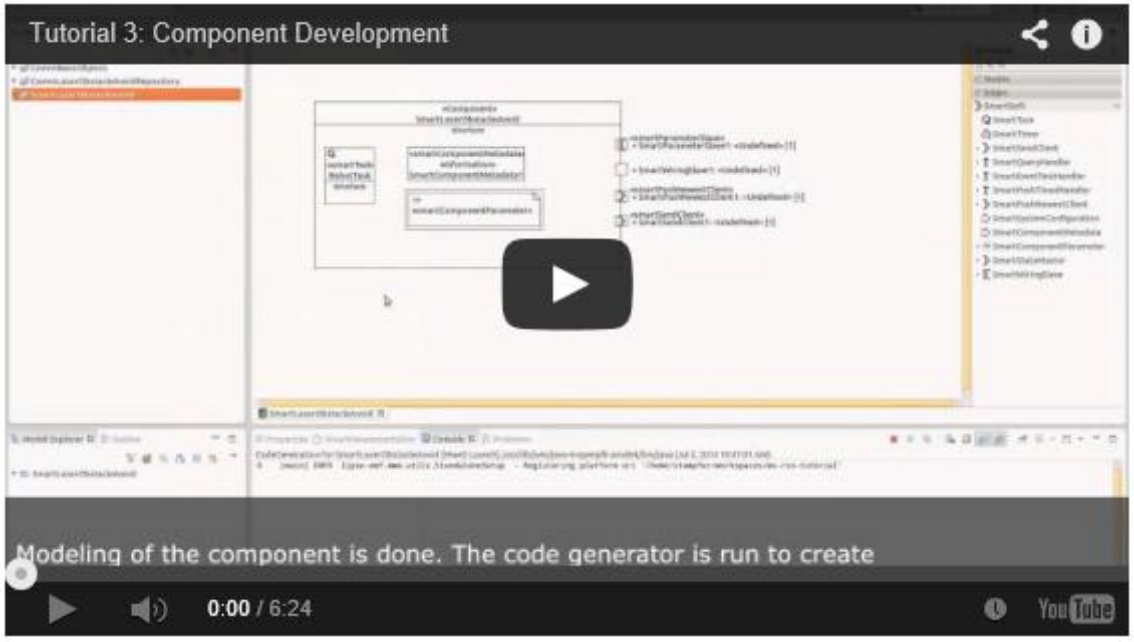
Demonstration – Step 1: Parameter Set Definition

http://youtu.be/2U4KxSgwtqY

This video demonstrates the modeling of a parameter using the SmartMDSD Toolchain.

The parameter represents a configurable maximum velocity of a robot. This parameter can later be instantiated by components. The maximum speed can then be configured through the parameter service.

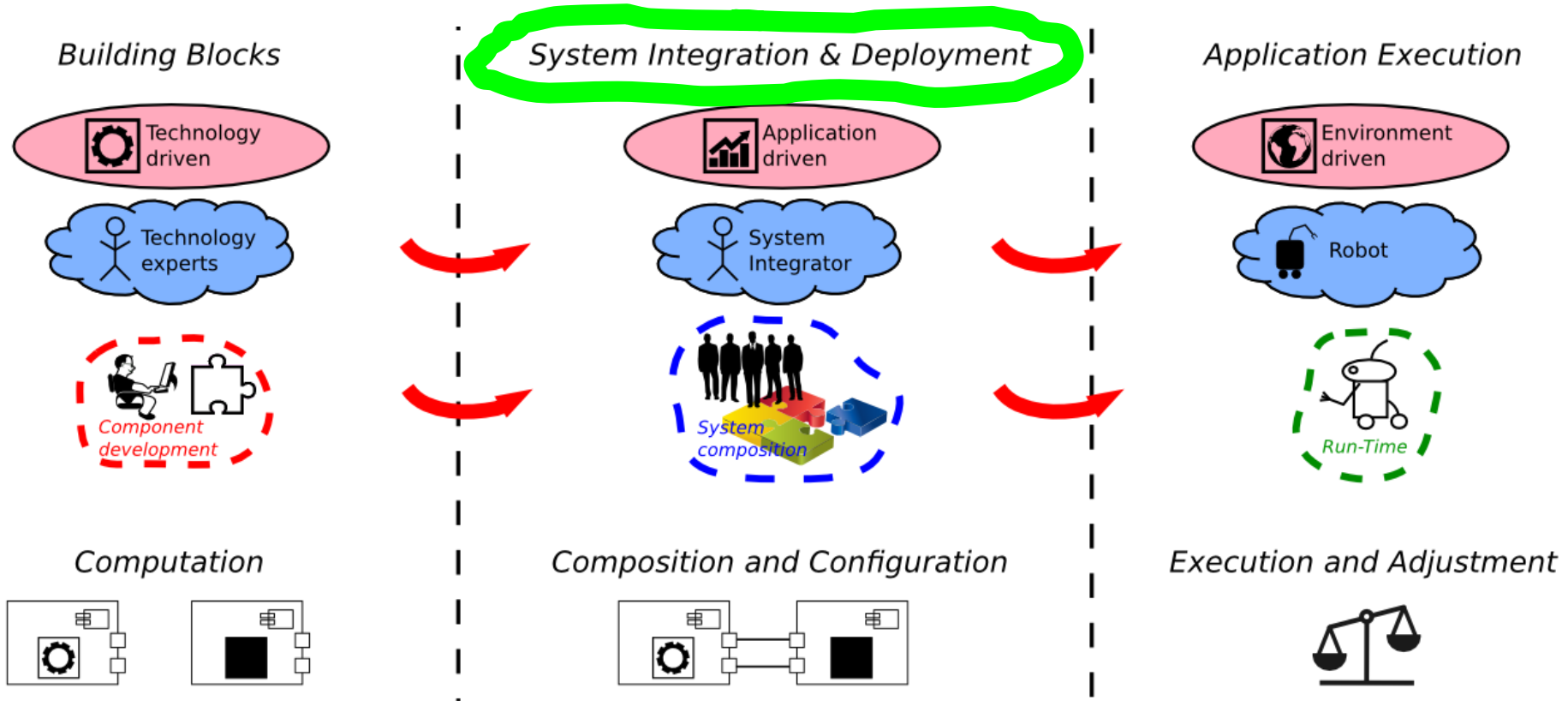
Demonstration – Step 2: Component Development

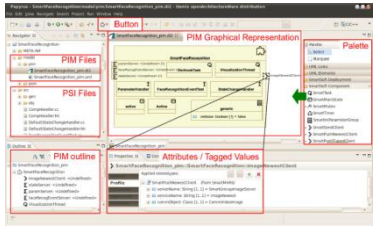


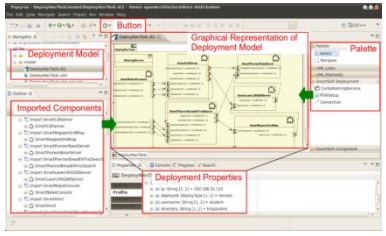
http://youtu.be/chyRCu4FCbs

This video demonstrates the modeling and implementation of a component using the SmartMDSD Toolchain.
The component receives laser scans. A simple obstacle avoidance algorithm outputs values for speed and direction. The component then thresholds the maximum speed according to a variation point (parameter "v_x", modeled in a previous video) before providing the navigation commands through one of its services. This parameter "v_x" can be configured during runtime of the component through its parameter service.

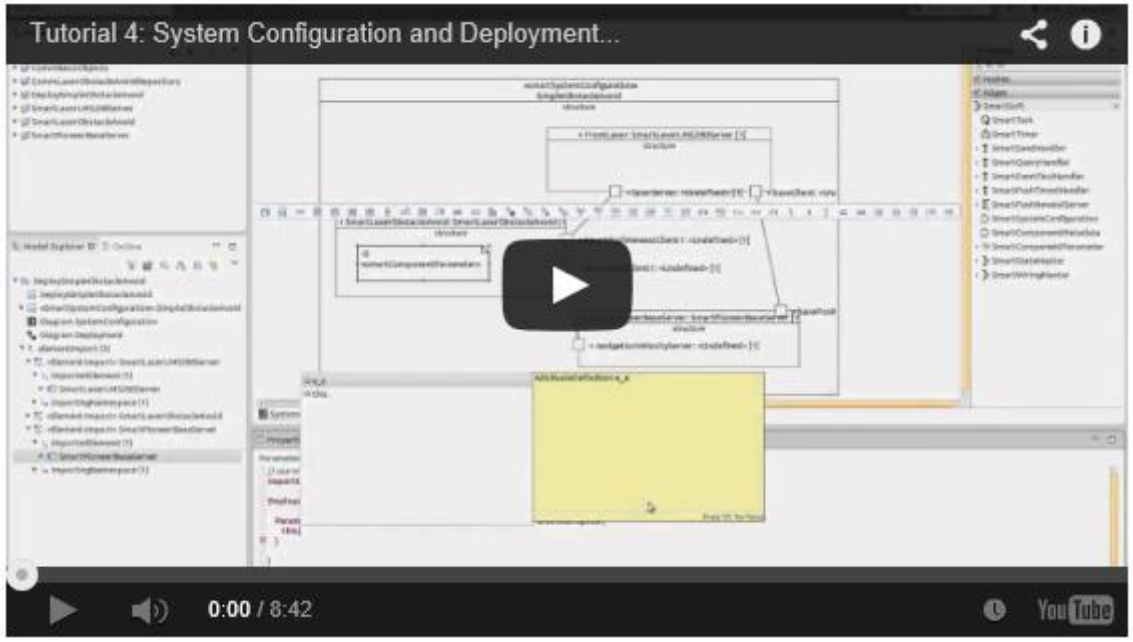
SmartMDSD Toolchain: Hands-On Example







Demonstration – Step 3: System Configuration and Deployment Model



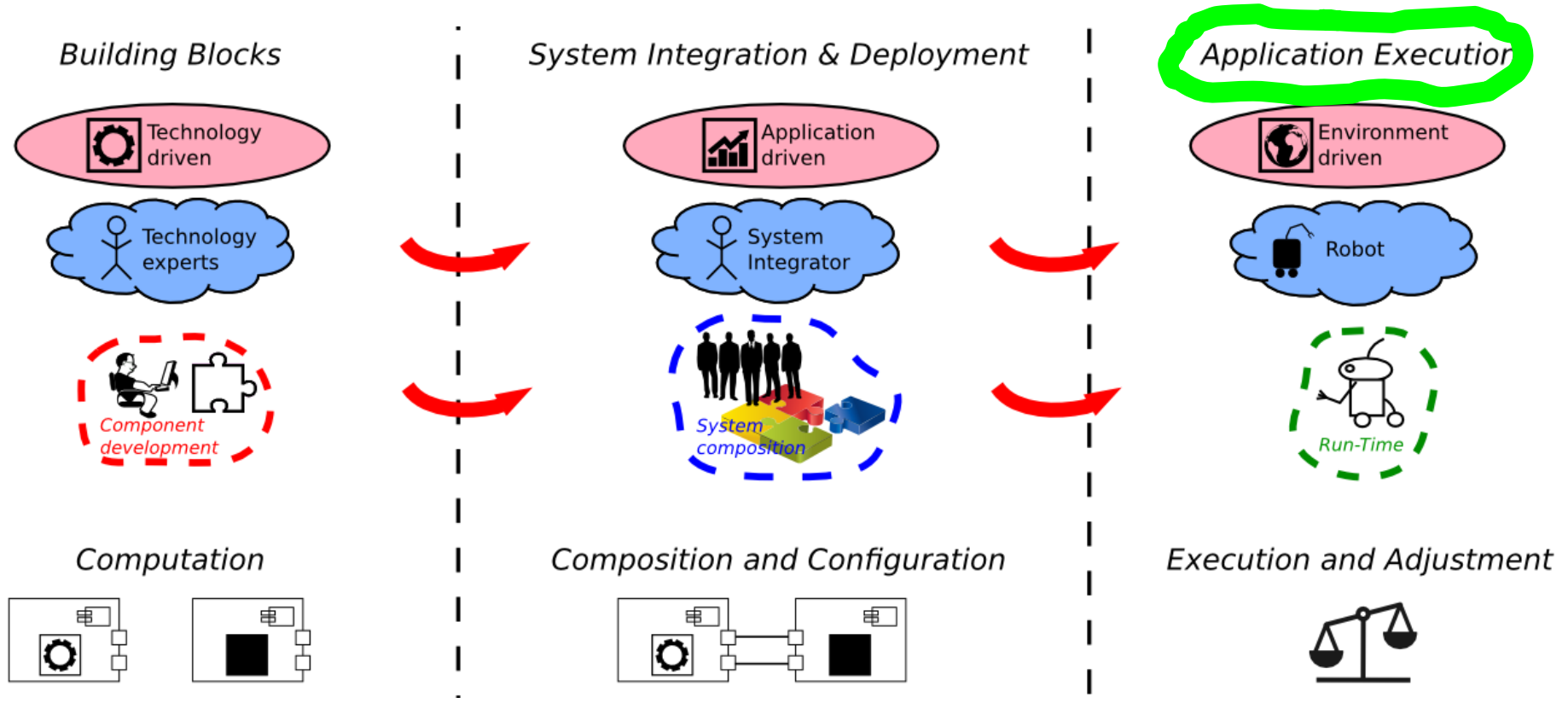
YouTube

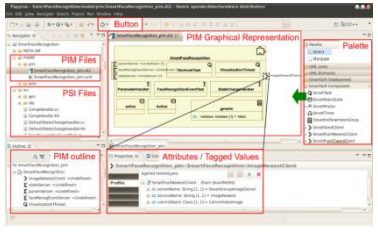
http://youtu.be/y-S33qaeNfI

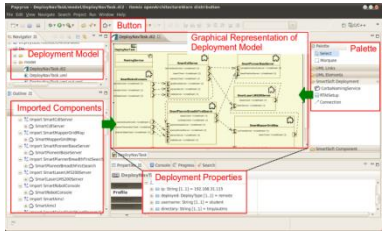
This video demonstrates the creation of system configuration and deployment model using the SmartMDSD Toolchain.

The scenario: a robot shall drive and avoid obstacles. It reuses (existing) components SmartLaserObstacleAvoid (see previous screencast), SmartLaserLMS200Server (laser ranger) and SmartPioneerBaseServer (robot). The system configuration model models the connection and configuration of components. The deployment model models the distribution of components on hardware.

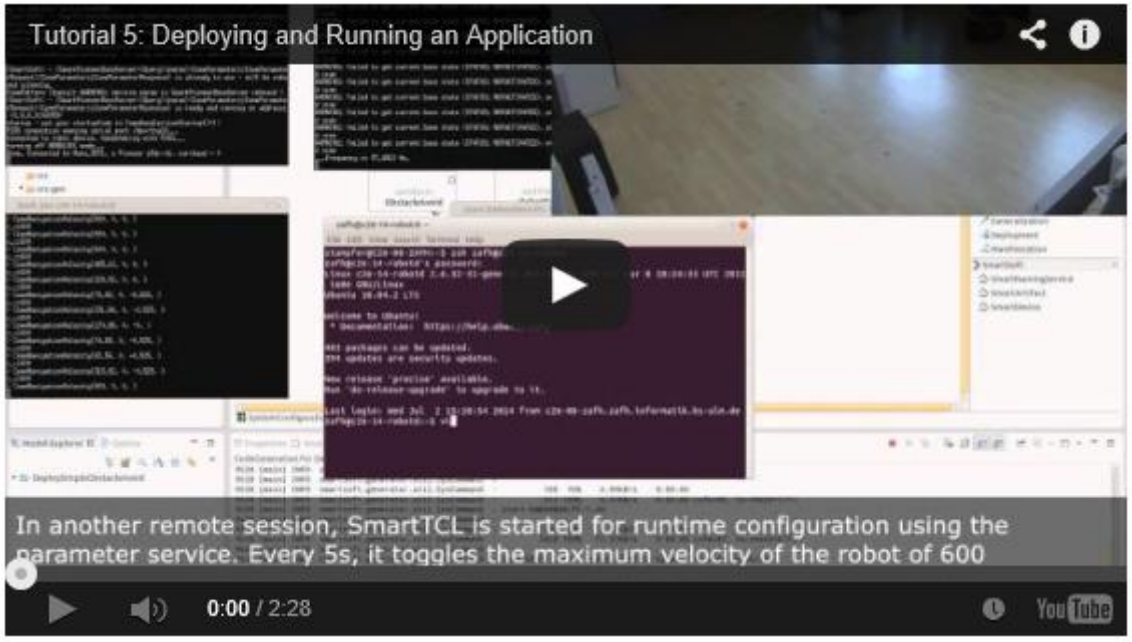
SmartMDSD Toolchain: Hands-On Example







Demonstration – Step 4: Deployment + Runtime



http://youtu.be/OZcC4ipt_BM

This video demonstrates the deployment and execution of an application developed using the SmartMDSD Toolchain.

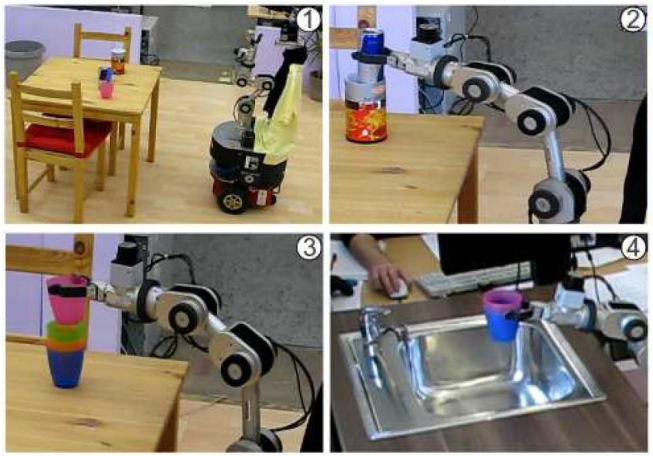
The application (laser obstacle avoidance from a previous video) is deployed using SSH. A remote session on the robot is established in order to run it. The robot will first drive with a maximum velocity of 600m/s (as configured in system configuration). Later, SmartTCL is used to change the maximum velocity of the component to 200 and back to 600 every 5s via the parameter service and explicated variation point v_x.

Software Variability and Runtime Reconfiguration

Butler Scenario

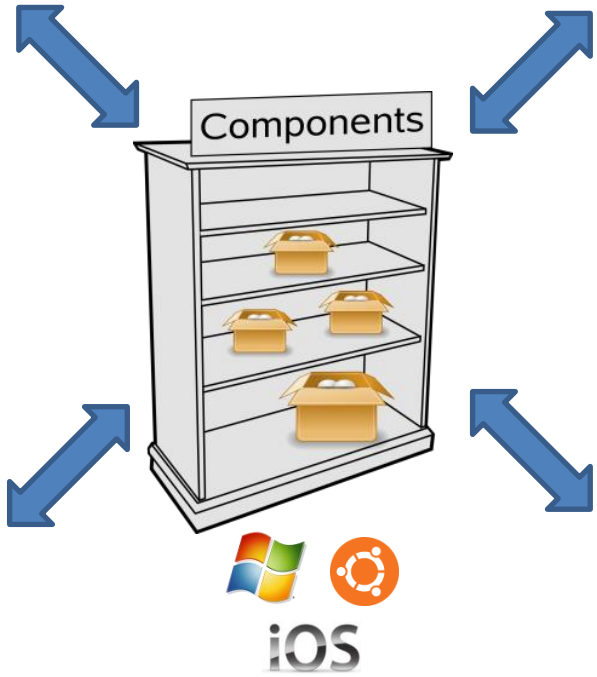
- Coffee Delivery
- Clean-up table
- Object Recognition
- States of objects

Runtime Reconfiguration



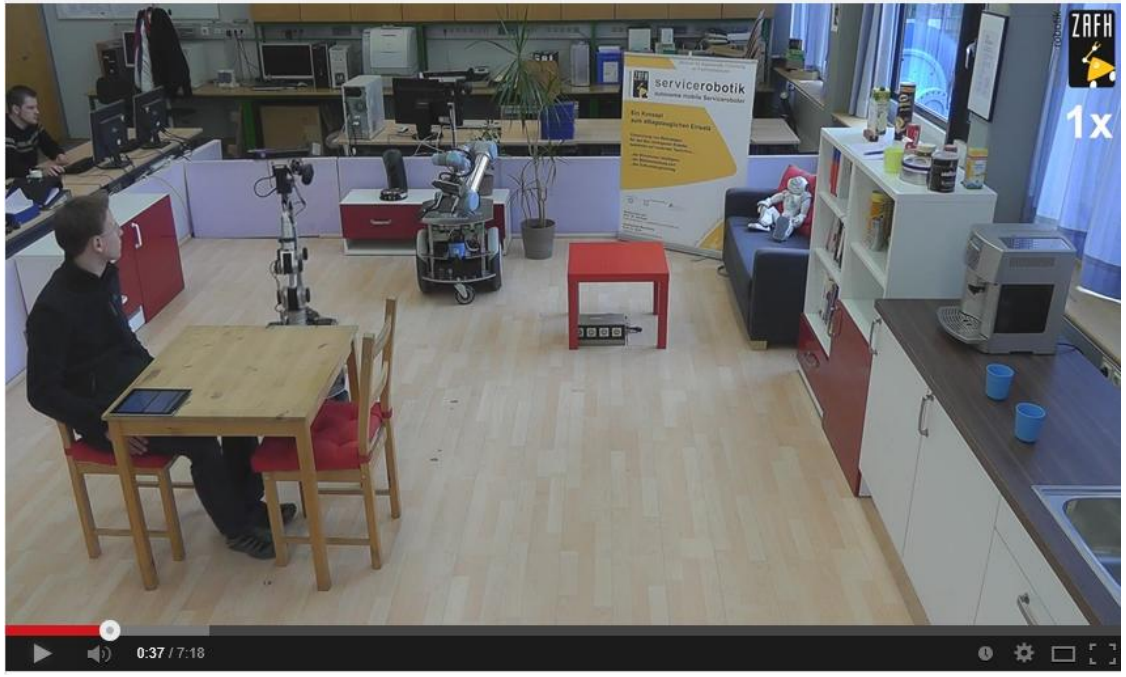
- Which coffee machine? Which velocity?
- Stacking cups and waste separation
- Active information-driven object recognition
- full or empty? Ready or problem?

Intralogistic Scenario



RoboCup@Home Student Team

Video: Real-World example



http://youtu.be/DjjNUPpj36E

The scenario shows the service robots "Kate" and "Larry" acting as butler. Kate takes orders from persons and hands over parts to Larry. While Kate makes a cup of coffee, Larry fetches the sugar dispenser from within a closed sideboard.

Links

- Portal
  - http://www.servicerobotik-ulm.de/
- Paper and Talks
  - http://www.servicerobotik-ulm.de/drupal/?q=node/15
- Videos
  - http://youtube.com/user/roboticsathsulm
- Software
  - http://www.servicerobotik-ulm.de/drupal/?q=node/7