Workshop Roboterkontrollarchitekturen

Roboterkontrollarchitekturen: Herausforderungen - Einführungs- und Übersichtsvortrag









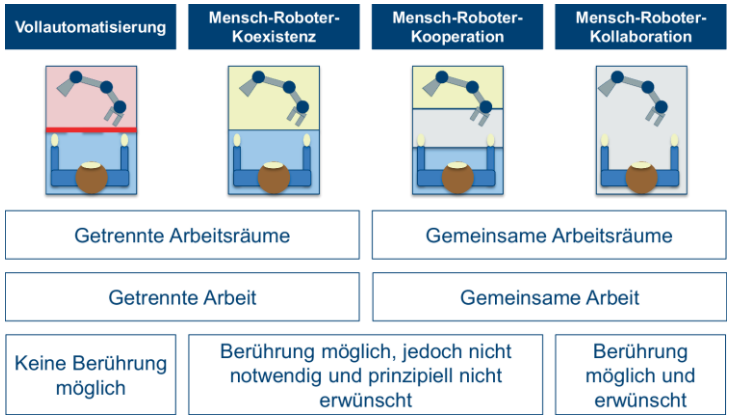






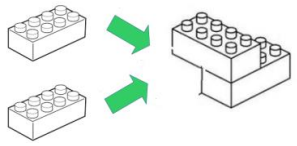
Automatisierte Systeme ⇔ Autonome Systeme
Flexibilität, Wandlungsfähigkeit, Einbindbarkeit in umgebende Infrastruktur
Vollautomatisierung, Mensch-Roboter-Koexistenz, Mensch-Roboter-Kooperation, Mensch-Roboter-Kollaboration
Physische und psychische Nähe von Robotern
Anwendungsdomänen ⇔ Bauformen

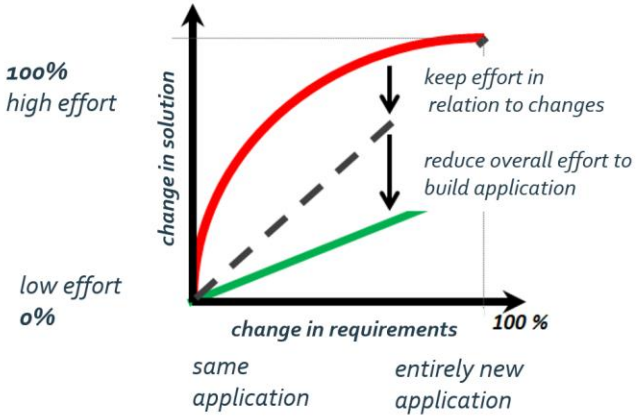
Workshop Roboterkontrollarchitekturen



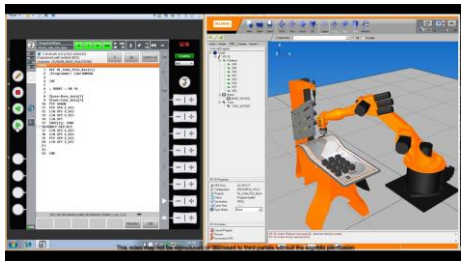
Regulative Anforderungen
Verantwortung, Haftung, Vertrauen

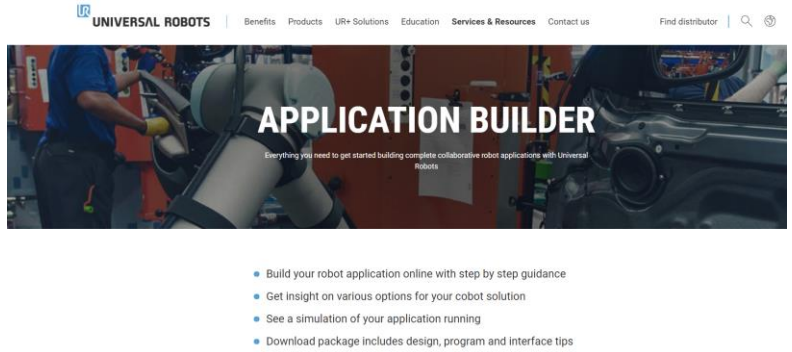
Roboter sind ein Flexibilitätsversprechen, welches Anwender selbst einlösen wollen

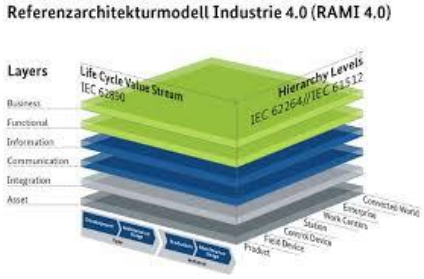




Roboterkontrollarchitekturen <=> Informationsarchitekturen <=> Softwarearchitekturen

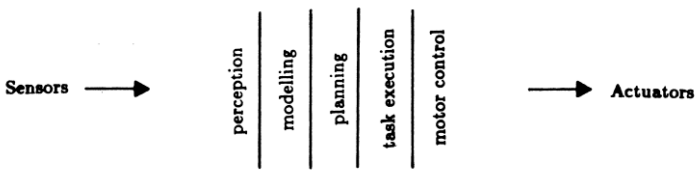


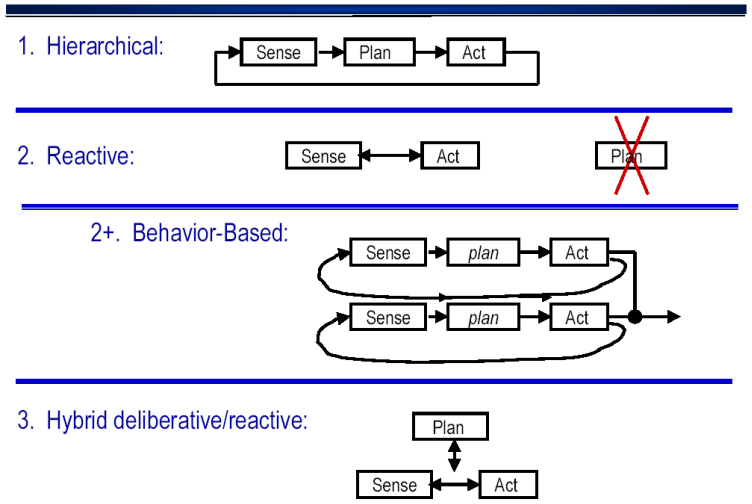




PLATTFORM INDUSTRIE 4.0

Workshop Roboterkontrollarchitekturen





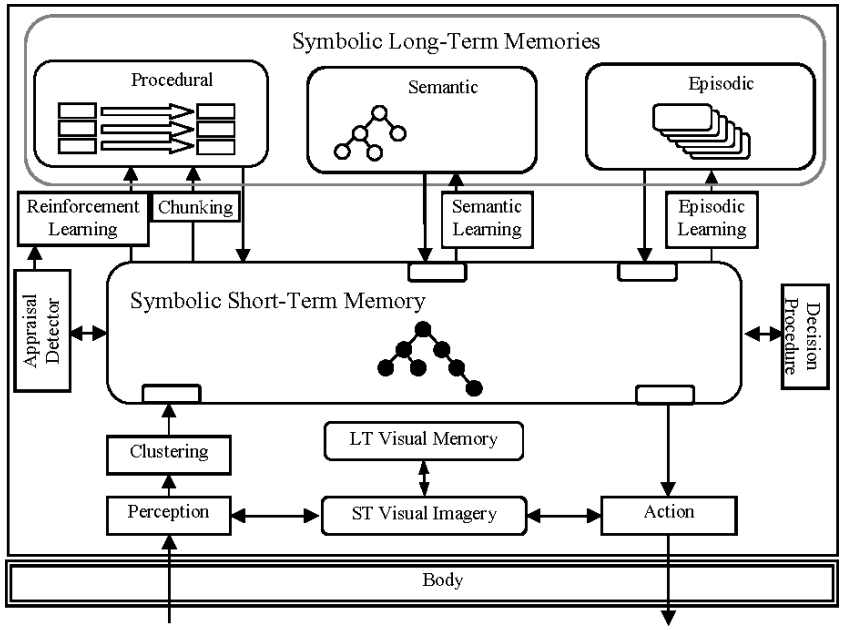
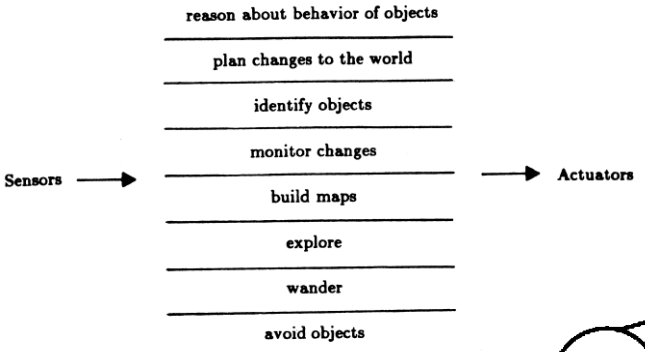
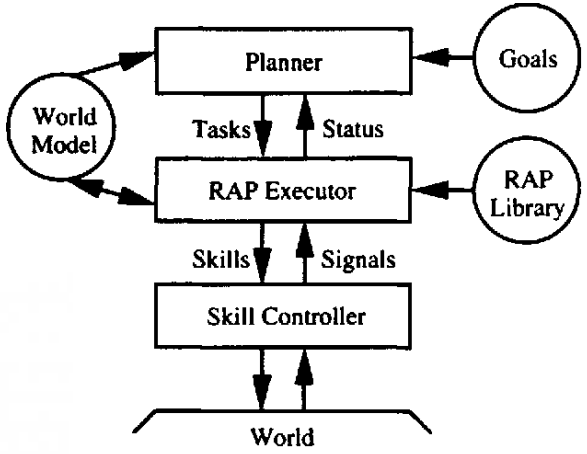
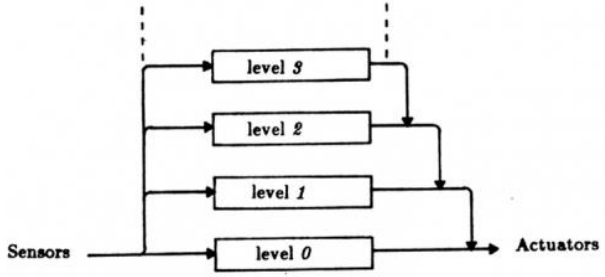


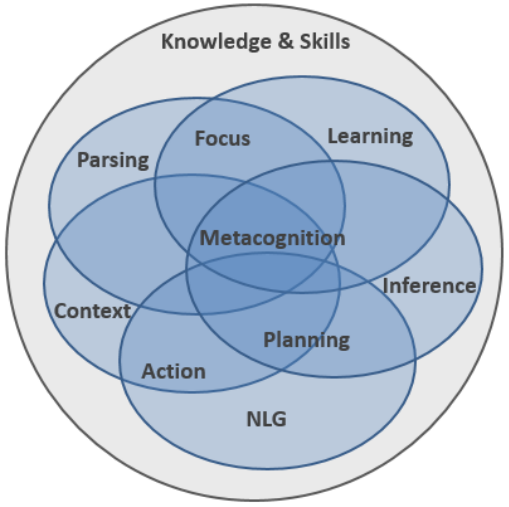
Figure 3: Soar 9







Highly Integrated Cognitive Architecture



Step 1: Models are used by persons

Step 2: Robots use models at run-time, e.g to monitor and explain what they are doing

Step 3: Robots adapt models and improve them

Workshop Roboterkontrollarchitekturen

Ziele und Herausforderungen

- *Was wollen wir mit einer Roboterkontrollarchitektur letztlich erreichen?*
- *Was ist davon aktuell bereits sehr gut erreicht / was ist davon aktuell noch komplett offen?*
- *Welche extra-funktionalen Aspekte müssen wir bei der Realisierung einer Roboterkontrollarchitektur beachten?*
- *Welche Methoden stehen uns für welche Aspekte zur Verfügung / fehlen?*
- *…*

Werden unterschiedliche Roboterkontrollarchitekturen benötigt für

- *Mobile Roboter, Manipulation, Mobile Manipulation?*
- *Art der Nähe des Roboters zu Personen?*
- *Art der Umgebung, welcher der Roboter ausgesetzt wird?*
- *Art der Programmierung durch welche Rollen?*
- *…*

Es gibt nicht *die eine* Roboterkontrollarchitektur!

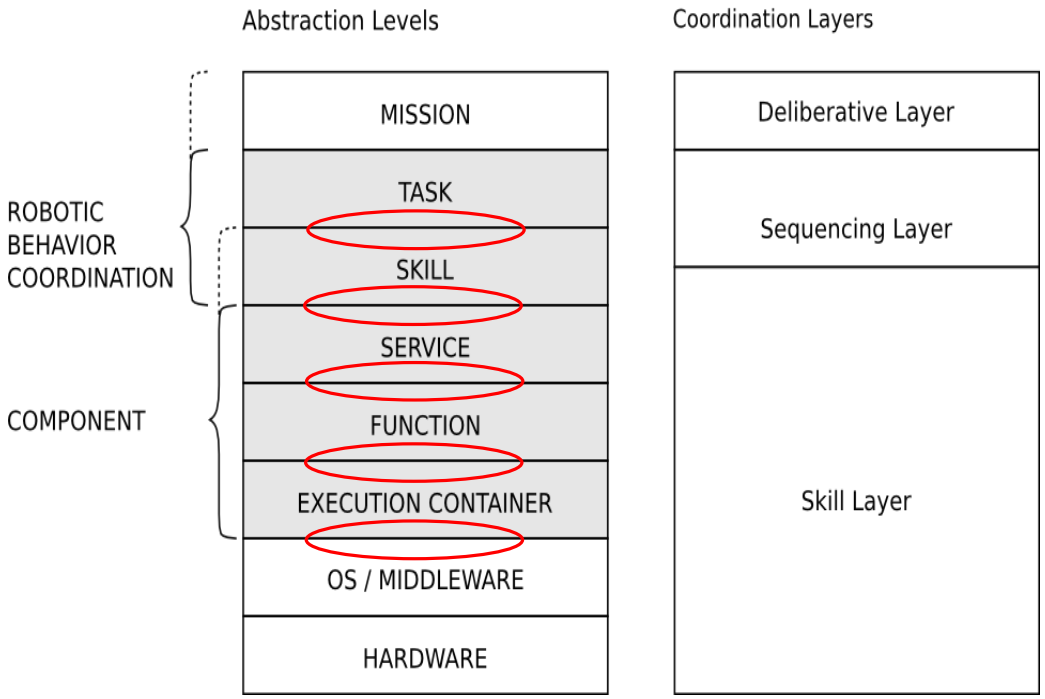
Auf jeden Fall gibt es zu jedem Zeitpunkt eine eindeutige Kontrollhierarchie. Diese kann aber zu unterschiedlichen Zeitpunkten unterschiedlich aussehen!

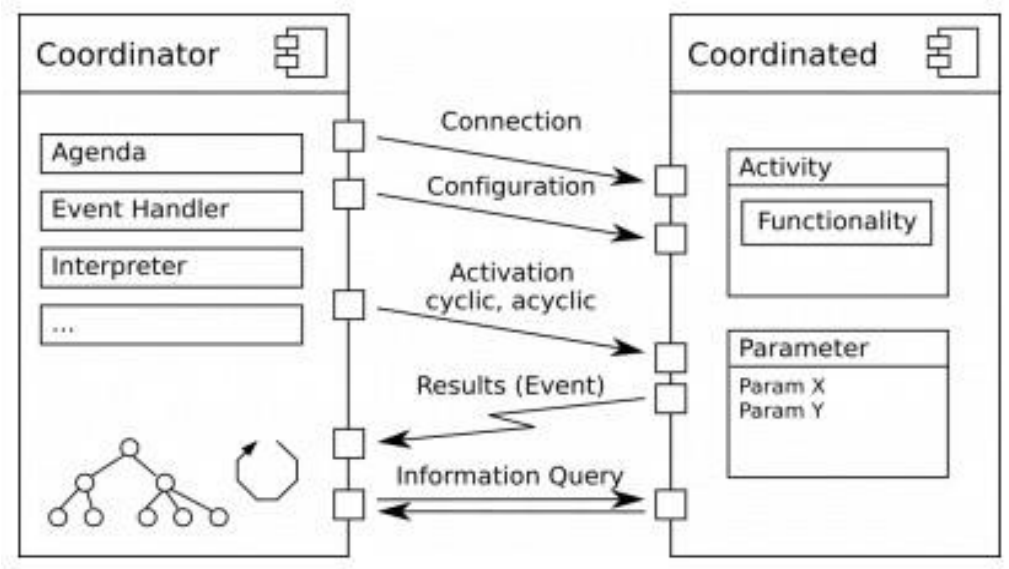
Wir müssen oftmals sogar zwischen verschiedenen Kontrollarchitekturen wechseln können!

Es sollte nicht komplizierter als nötig sein, aber versuche nicht, komplexe Probleme mit Ansätzen zu lösen, die dafür nicht mächtig genug sind!

Workshop Roboterkontrollarchitekturen

Was ist für welche Aspekte verfügbar?





locally synchronous,
globally asynchronous

Serviceorientierte Softwarearchitekturen
Komponentenorientierte Software

monitors and sanity checks

event driven ⇔ time triggered

Hierarchische Task-Netze
Dynamische hierarchische Zustandsautomaten
Endliche Automaten

Computation Models
- logical execution time (LET)
- synchronous data flow (SDF)
- event driven
- ...

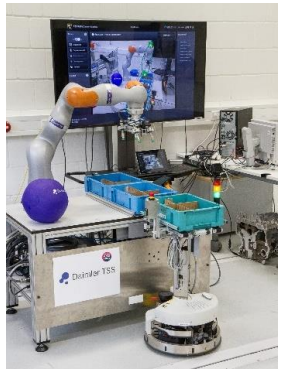
...

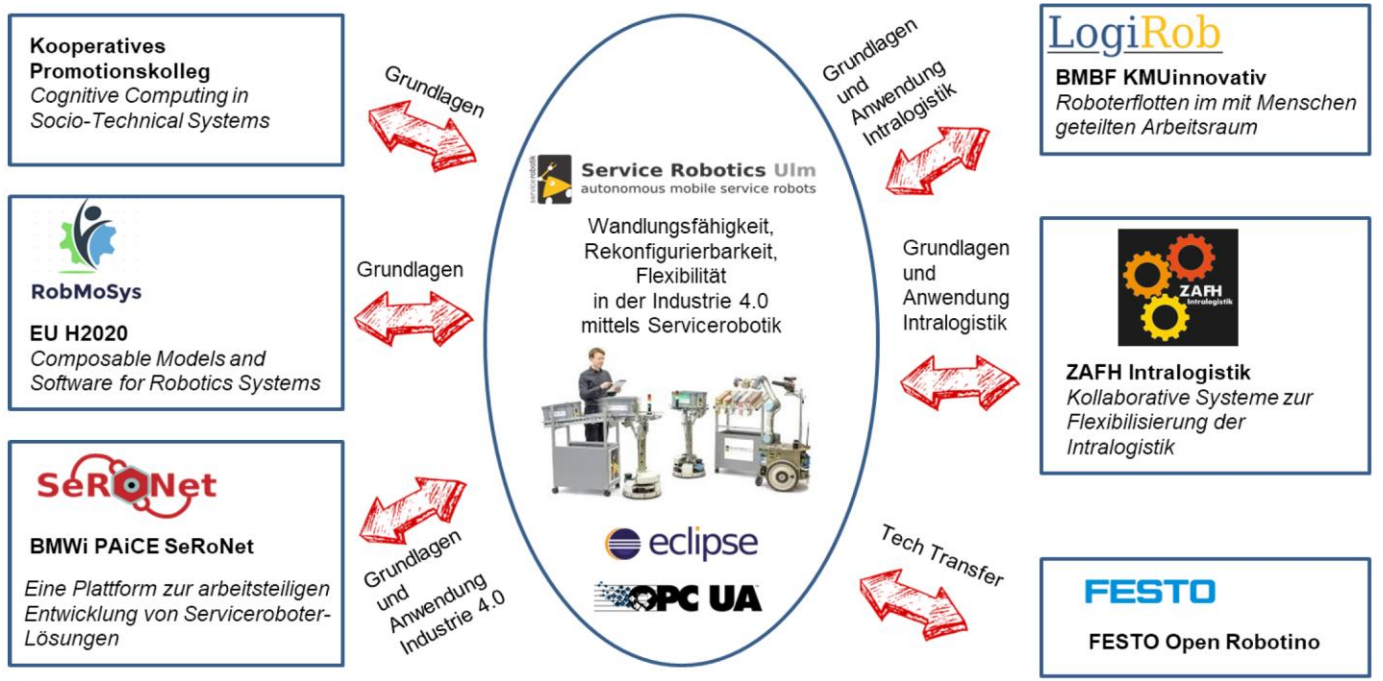
Modellgetriebene Ansätze für die Explizierung von Eigenschaften, Konfigurationsmöglichkeiten etc.

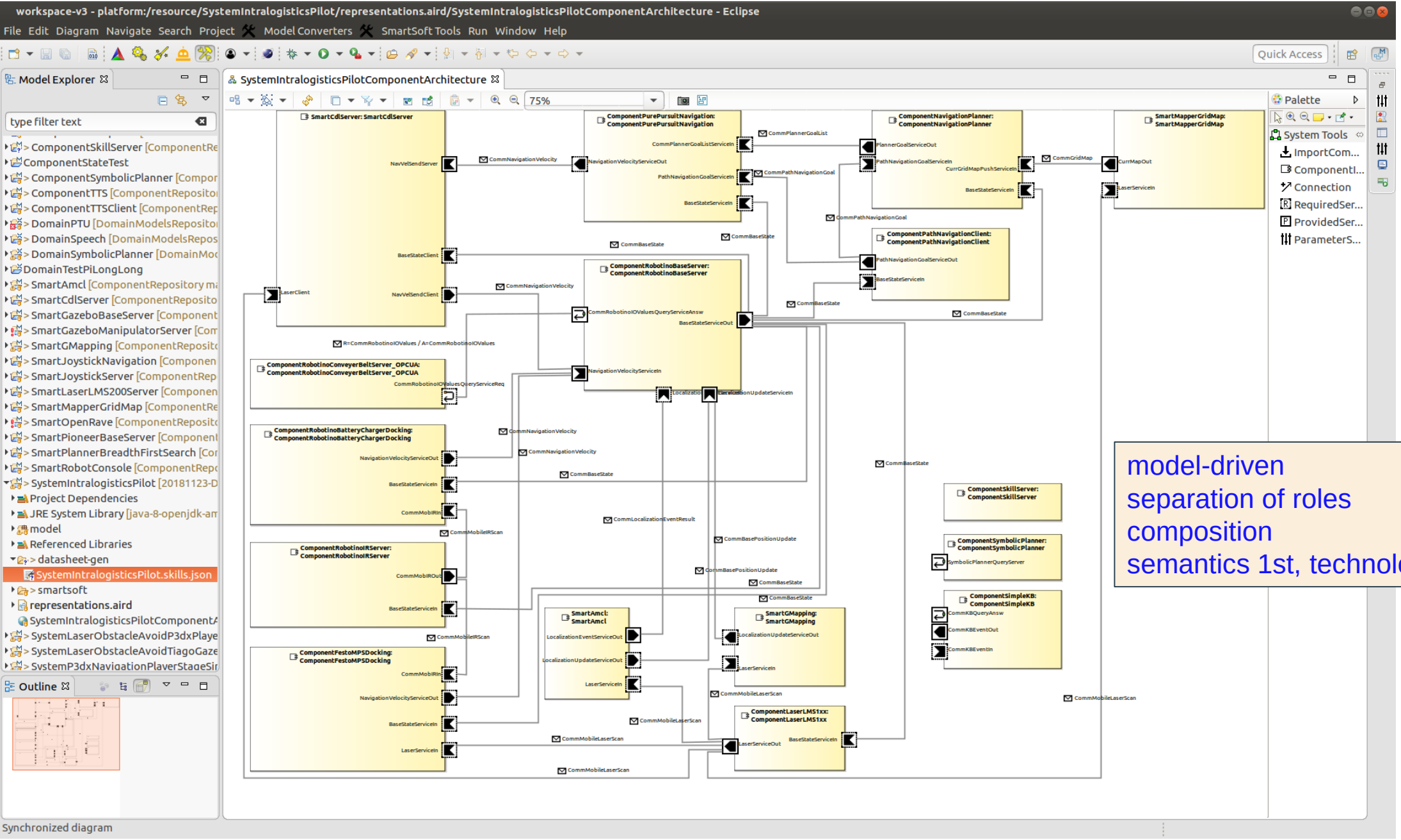
diskret ⇔ kontinuierlich

Composition, Separation of Roles and Model-Driven Approaches as Enabler of a Robotics Software Ecosystem

Towards an EU Digital Industrial Platform for Robotics





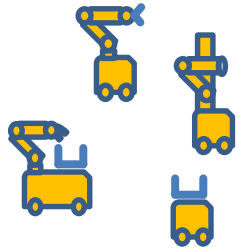


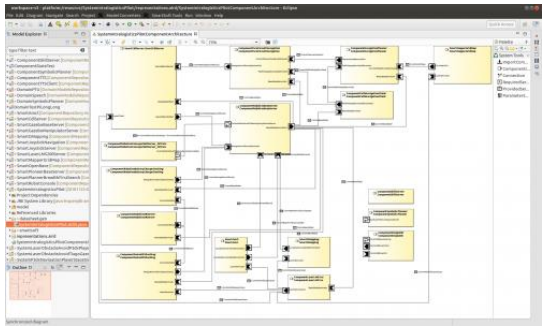
model-driven
separation of roles
composition
semantics 1st, technology mapping 2nd

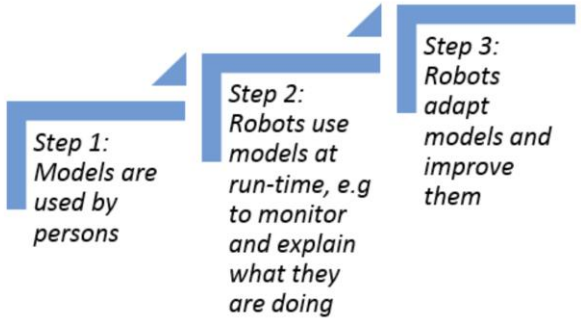
Towards an EU Digital Industrial Platform for Robotics

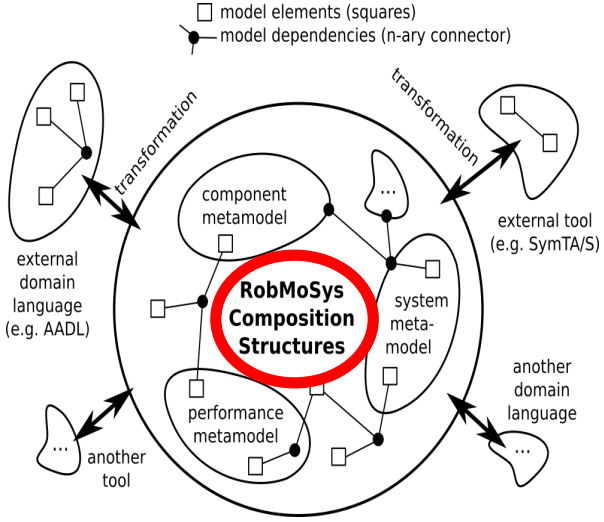
Models always have a purpose: overall purpose is consistency

- organize consistent abstraction for e.g. prediction
- better understanding in early phases avoids costs at later stages
- organize interfaces and ensure fits while decoupling roles, responsibiilities, scopes, etc.
- ensure traceability of properties, conformance by design and not just by discipline, etc.









A model-driven approach allows to

- ...secure your design and solution efforts
- ...decouple different paces of evolution
- ...be technology-agnostic ***(semantics: early binding / technology: late binding)***
- ...predict what you get before you build it
- ...exploit the power of combinatorics
- ...explicate otherwise hidden magic numbers
- ...benefit from low effort in modifications towards lot size 1
- ...achieve robust job fulfillment by context-aware run-time decisions

Towards an EU Digital Industrial Platform for Robotics
Projects EU H2020 RobMoSys and BMWi PAiCE SeRoNet

RobMoSys

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732410.

https://robmosys.eu
https://robmosys.eu/wiki/open-call-2
https://discourse.robmosys.eu
https://robmosys.eu/wiki

cea · Hochschule Ulm University of Applied Sciences · KU LEUVEN · TUM Technische Universität München · COMAU · PAL ROBOTICS · SIEMENS · EUnited Robotics European Robotics Association · eclipse

conforms to

SeRoNet

Gefördert durch:
Bundesministerium für Wirtschaft und Energie
aufgrund eines Beschlusses des Deutschen Bundestages

PAiCE

https://www.seronet-projekt.de/plattform/tooling.html

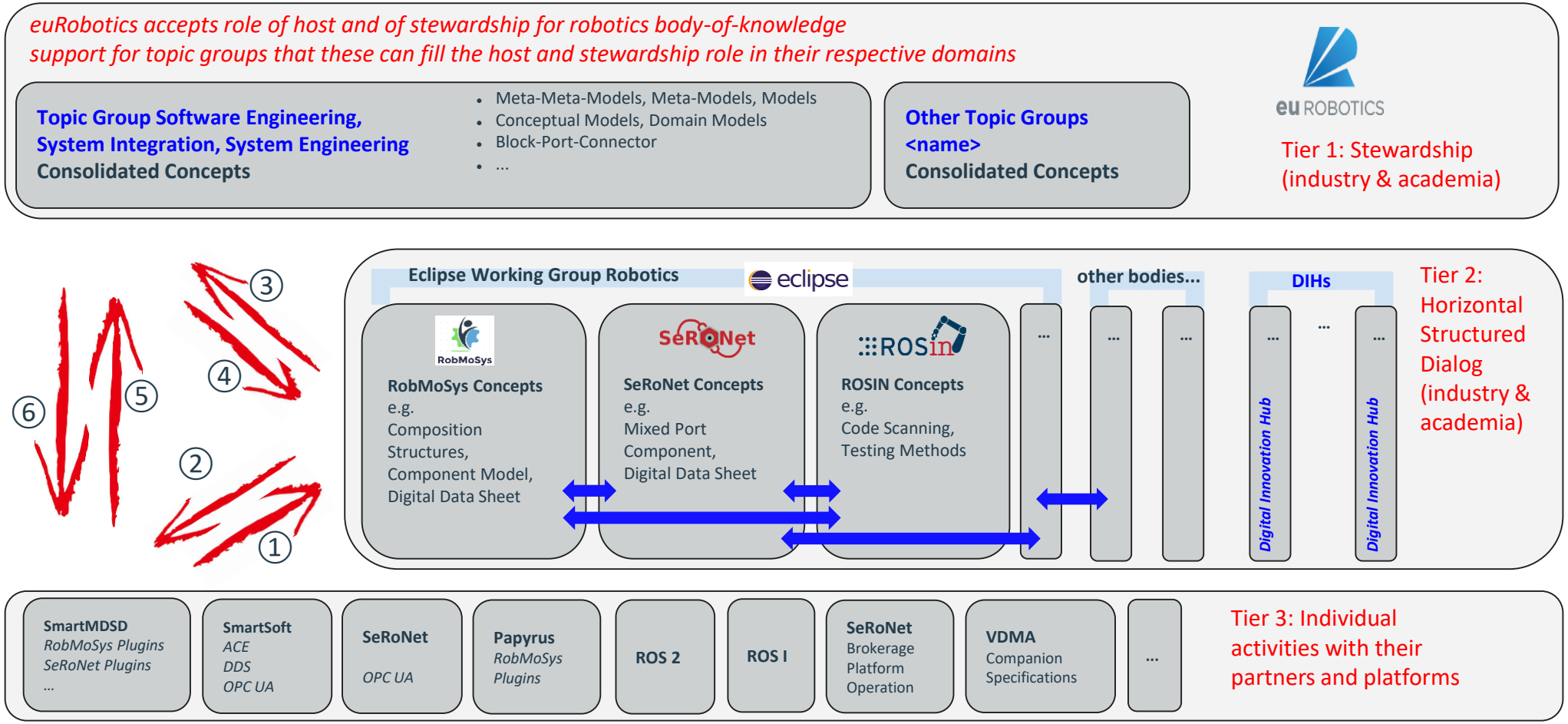
Fraunhofer IPA · ISW · THU Technische Hochschule Ulm · RUHRBOTICS We Program Your Success · TRANSPHARM competence.connection.courage

KUKA · FZI · Universität Paderborn · Daimler TSS · KMD Klinikum Mannheim Dienstleistungsgesellschaft mbH

Process: Sustainability

RobMoSys provides a ***concept & structure & mechanism***

- to deal with different coexisting levels of maturity, acceptance, innovation, …
- to achieve evolvement, be inclusive, to achieve trust, to go beyond project life-times, …

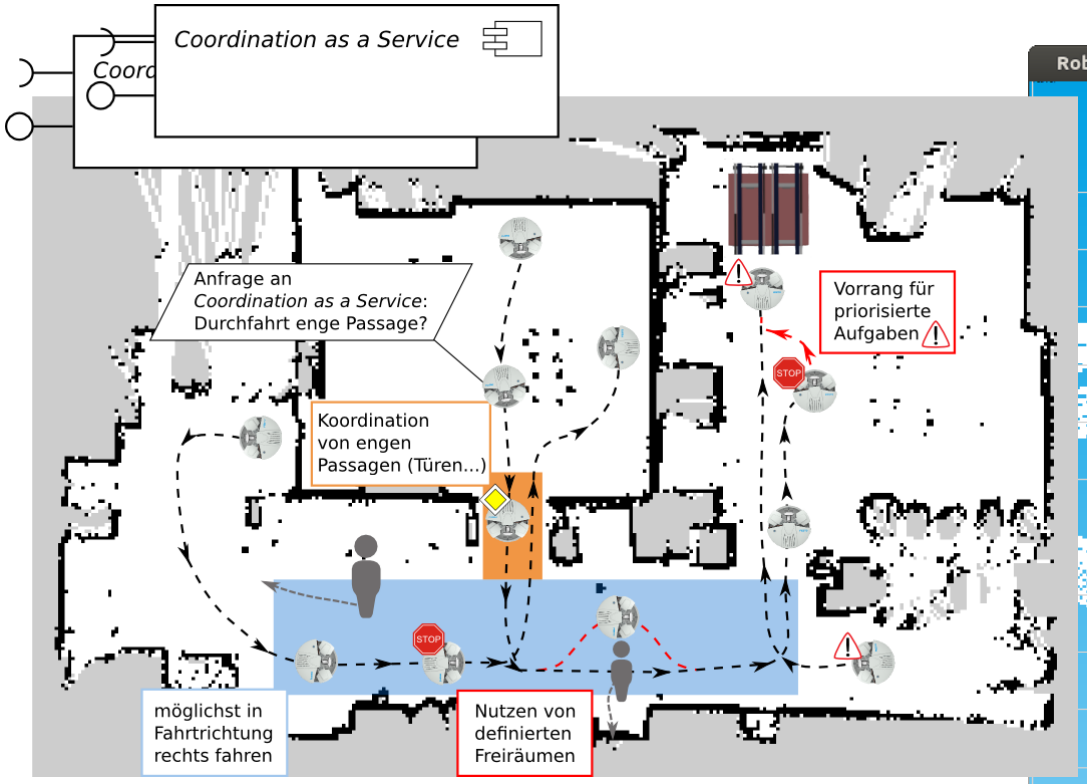


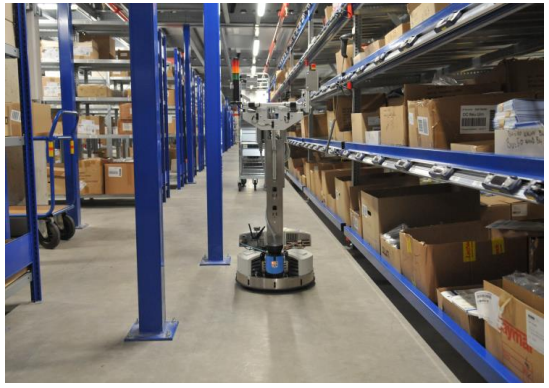
Service Robotics Ulm

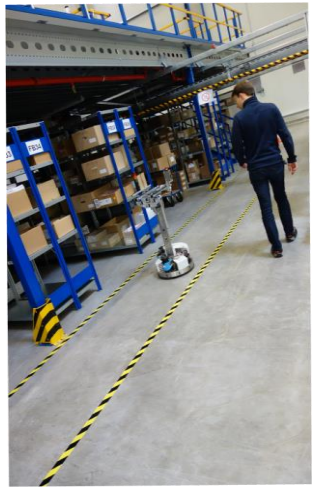
autonomous mobile service robots

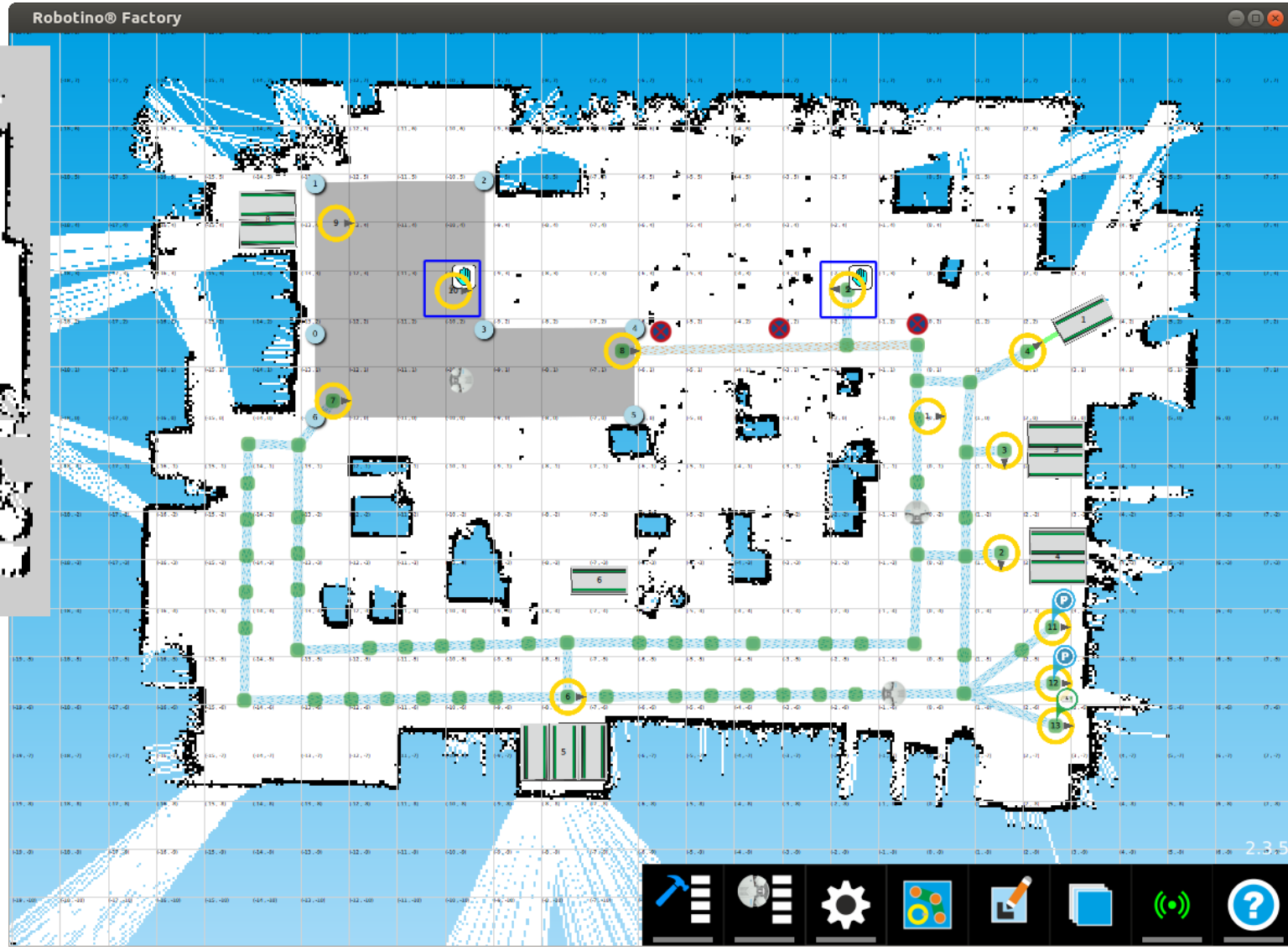
www.servicerobotik-ulm.de

https://youtu.be/RHvvb6lTHG4


Coordination as a Service
Anfrage an
Coordination as a Service:
Durchfahrt enge Passage?
Koordination
von engen
Passagen (Türen...)
Vorrang für
priorisierte
Aufgaben
möglichst in
Fahrtrichtung
rechts fahren
Nutzen von
definierten
Freiräumen



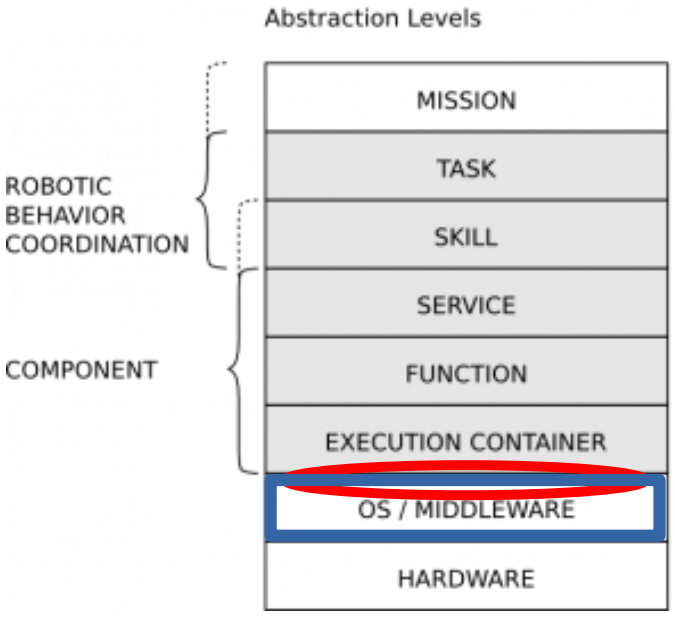


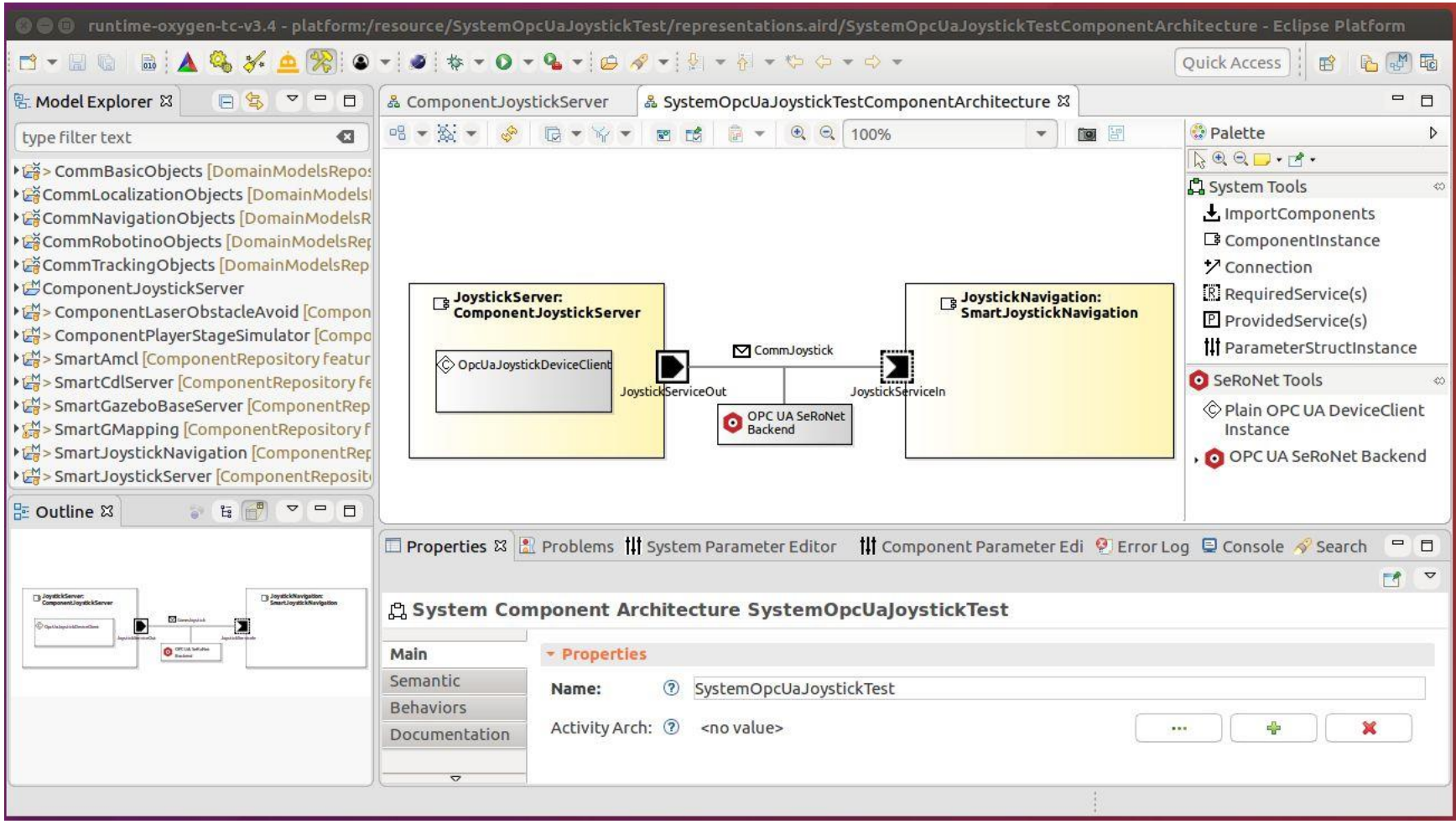

Robotino® Factory

SmartTCL

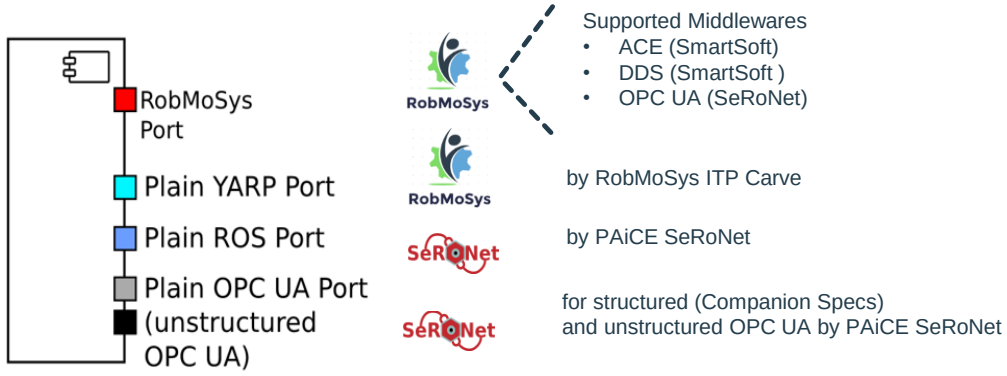
https://youtu.be/RHvvb6lTHG4

- *Mixture of different middlewares within a single system*
- *Middleware can be decided per connection*
- *Late binding of middleware at deployment without recompilation of components*

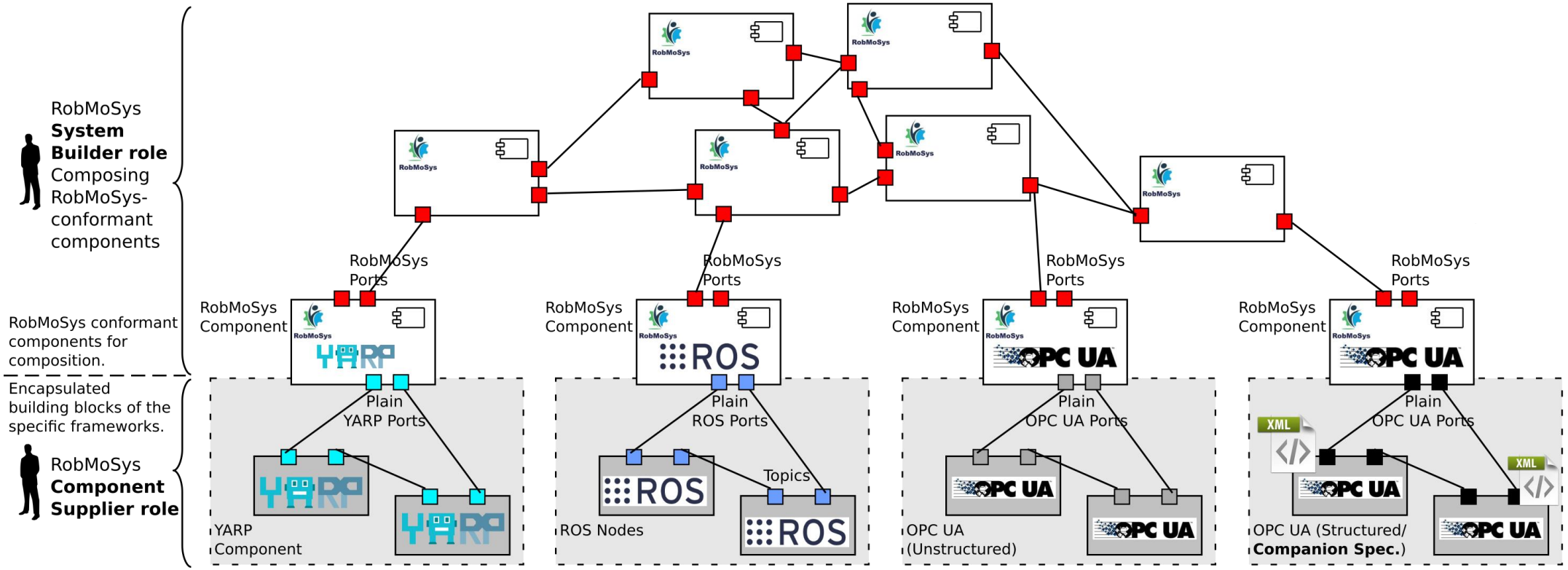




Mixed Port Component as Migration Path

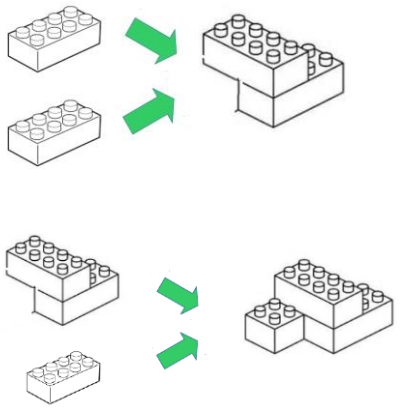


https://wiki.servicerobotik-ulm.de/tutorials:start



The way of thinking…

Composition, Blocks, Ports, Connectors, Data Sheets, Models



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

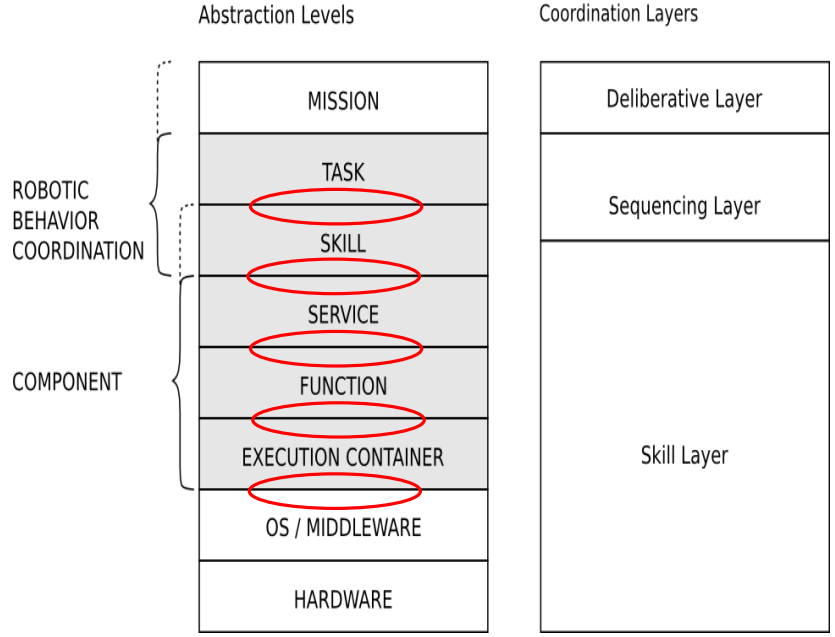
Separation of Concerns
- computation
- communication
- coordination
- configuration

splitting apart...

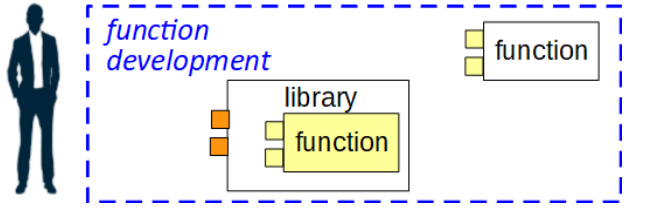
Composition, Composability, Compositionality
- computation
- communication
- coordination
- configuration

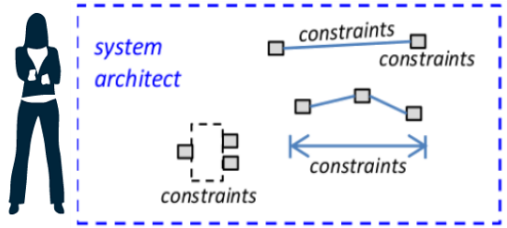
...putting together

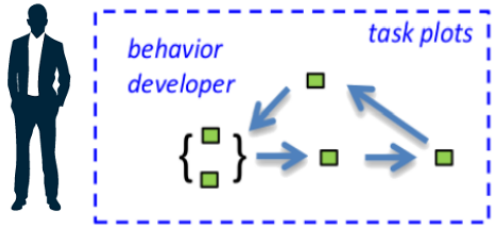
Separation of Roles!

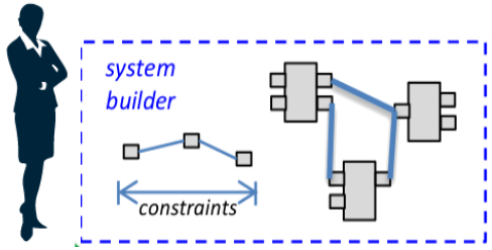


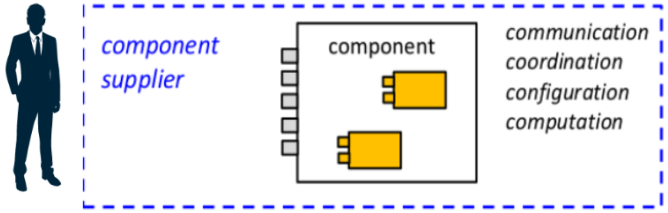
Which patterns and structures form the ***sweet spot*** between ***freedom of choice*** and ***freedom from choice***?

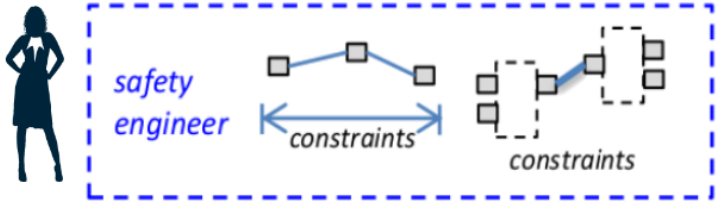






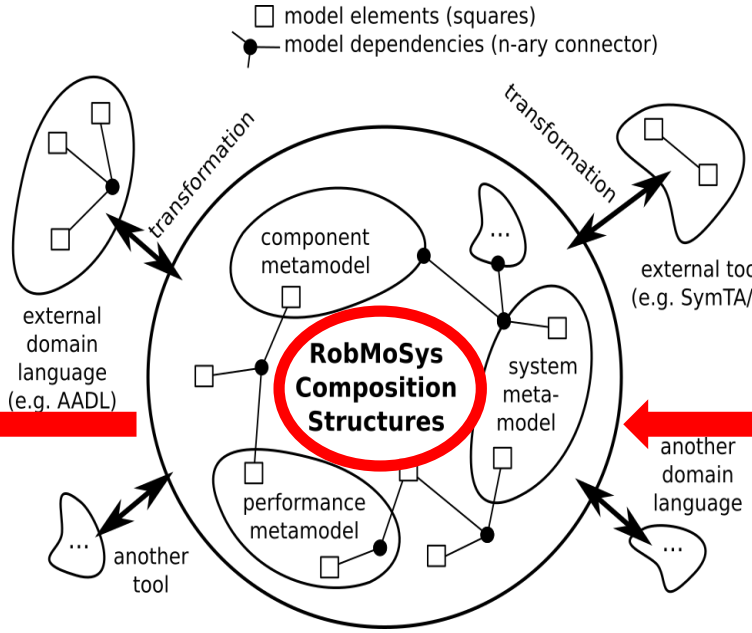


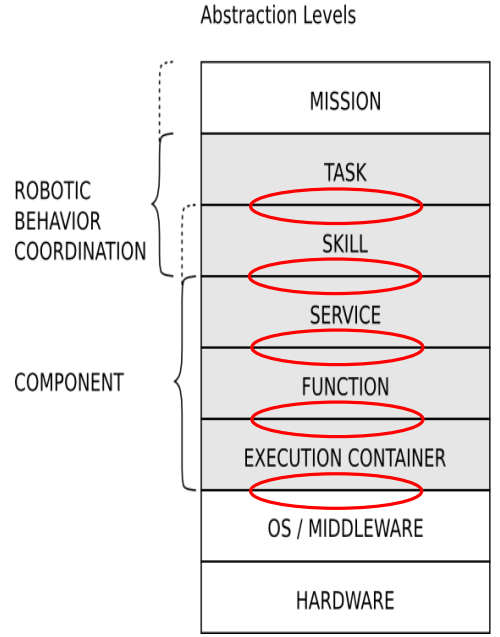




...

Composition, Blocks, Ports, Connectors, Data Sheets, Models



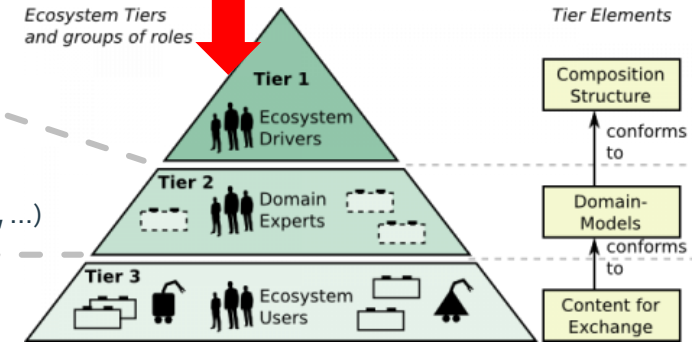


Architectural Pattern for Communication
Architectural Pattern for Component Coordination
Architectural Pattern for Software Components
Architectural Pattern for Managing Transition of System States
Architectural Pattern for Task-Plot Coordination (Robotic Behaviors)
Architectural Pattern for Service Definitions
Architectural Pattern for Stepwise Management of Extra-Functional Properties
Architectural Pattern for Coordinate-Frame Transformation
Architectural Pattern for Reservation Based Resource Management
...

Meta-Model:
domain-independent

Model:
domain-specific
(mobile robots, intralogistics, manipulation, ...)

Implementation:
components and systems



Toolings

Tier 1 content:
Modeling Foundations & Composition Structures

Tier 2 content:
Domain Models & Stacks

Tier 3 content:
Components & Systems

- Methodology
- Meta Models
- Models
- Implementation Technologies
- Toolings
- Building Blocks
- Pilot Applications
- Repositories
- Processes

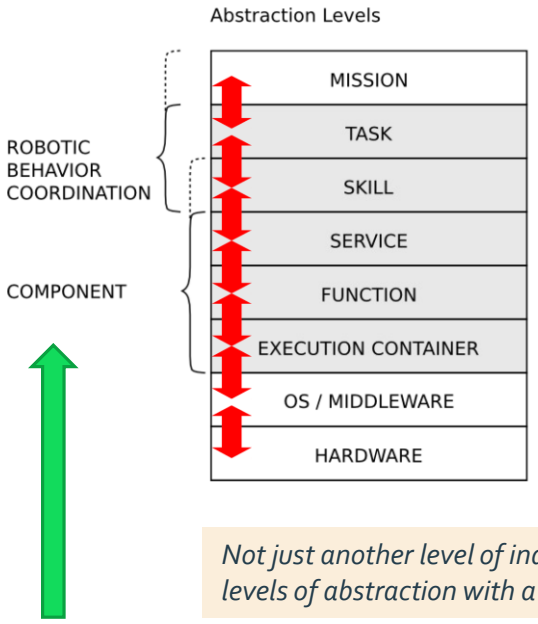
Example I4.0:

OPC UA

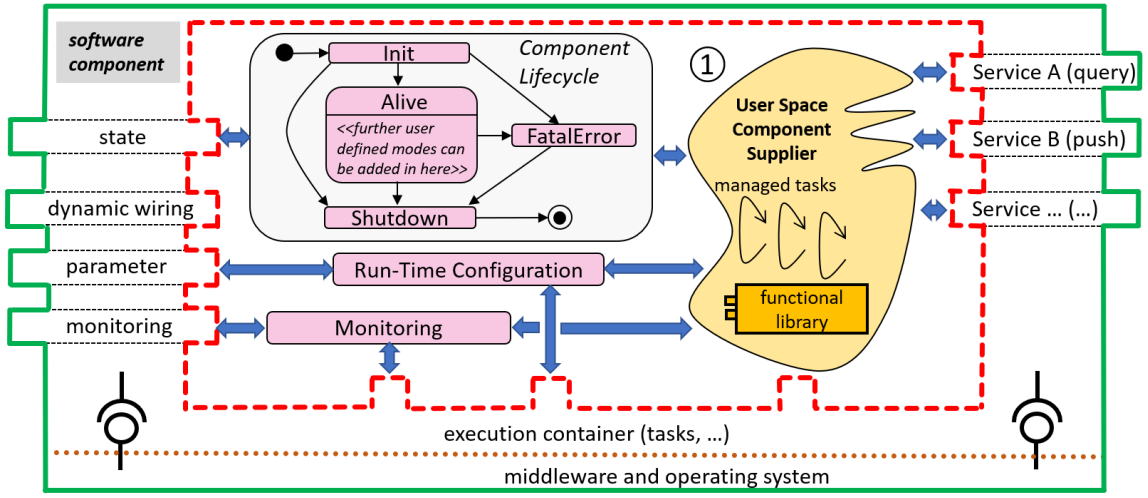
Companion Specification

Conformant Devices

Composition, Blocks, Ports, Connectors, Data Sheets, Models



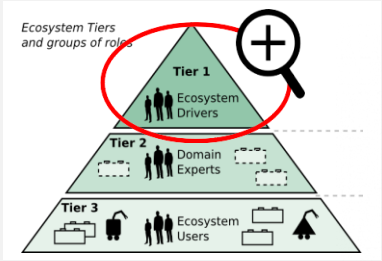
Not just another level of indirection, but levels of abstraction with a real benefit

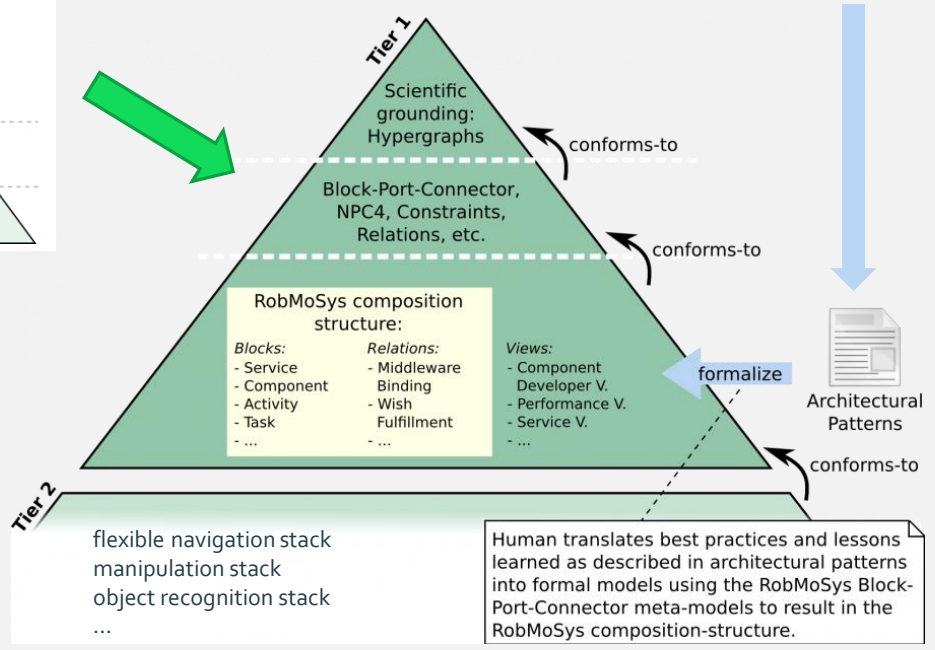


S/W component with communication (service-oriented ports), configuration (resources, parameters), coordination (modes, lifecycle), computation

Tier 1 provides the **general structures for composition**.
Three levels can be distinguished:

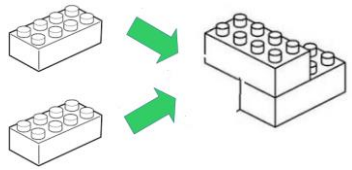
Architectural Pattern for Communication
Architectural Pattern for Component Coordination
Architectural Pattern for Software Components
Architectural Pattern for Managing Transition of System States
Architectural Pattern for Task-Plot Coordination (Robotic Behaviors)
Architectural Pattern for Service Definitions
Architectural Pattern for Stepwise Management of Extra-Functional Properties
Architectural Pattern for Coordinate-Frame Transformation
Architectural Pattern for Reservation Based Resource Management
...

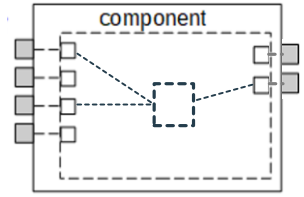




Composition, Blocks, Ports, Connectors, Data Sheets, Models

- ***all is interfaces***
  - context assumption => interface specification to the environment
  - compatibility => replacing behaves the same in that context
  - …
- ***blocks, ports, connectors***
  - different views of parts: state view, behavior view, architecture view, interface view, …
- ***composition, composability, compositionality***
  - you get the properties of the system by the properties of their parts
  - parts are to be used „as is" and they come with explicated variation points
- ***modularity***
  - you get modularity if it is real composition
- ***important:***
  - **not**:
    - have different views of full system first and then introduce blocks for parts hiding parts of the full system
    - refine uncoupled views of different concerns of a system first and then hope that you can consistently realize structure and behavior
  - **but**:
    - have blocks first which in a defined way have concerns combined which can be presented in different views





The Concept of a Digital Data Sheet

- *Data sheets are models*
- *A data sheet describes an outside view of an asset, including its variation points*
- *A data sheet includes internals only as far as you need to know them for using the asset and for predicting its fit (behavior, structure) for your context*
- *Data sheets are, by purpose, not rich enough for synthesis of the artefact*

Composition, Blocks, Ports, Connectors, Data Sheets, Models

PICTURE QUALITY
- Full HD 1080p
- Motion Rate 60
- Wide Color Enhancer

SMART
- Smart TV
- Full Web Browser

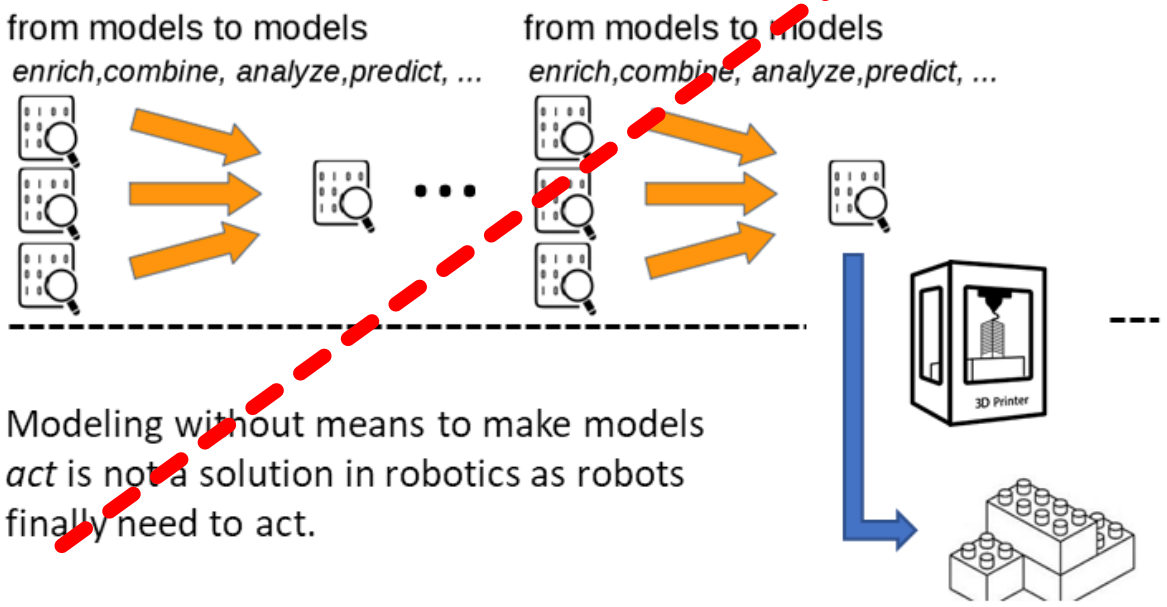
CONNECTIONS
- 2 HDMI® Connections
- 2 USB Connections
- 802.11n Wi-Fi Built In
- 1 Component in
- 1 Composite In (Shared with AV Component input)

Synthesis

Data Sheets are Models

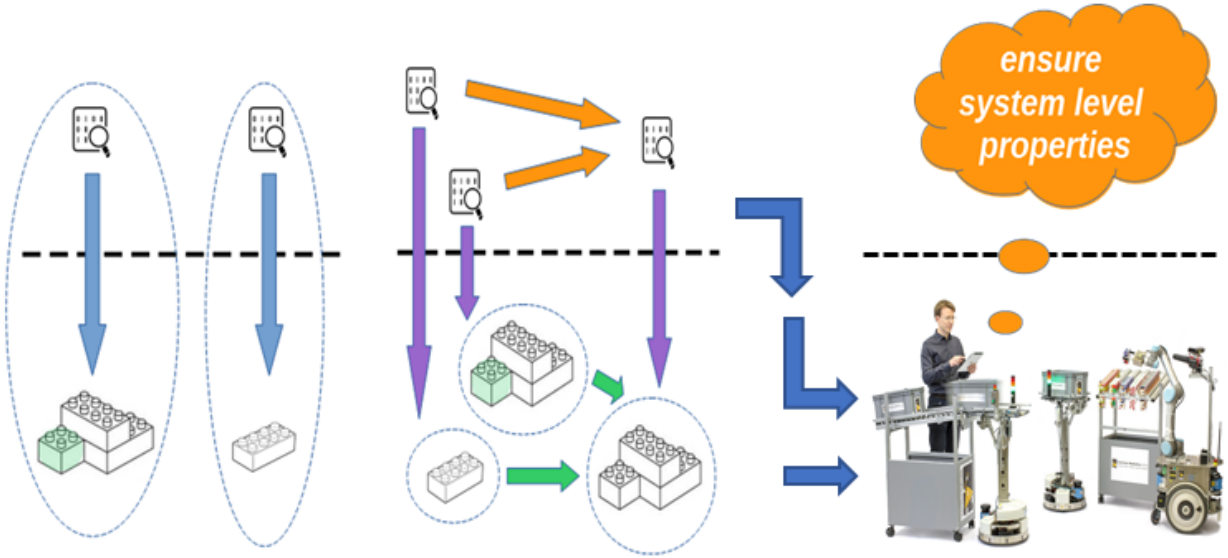
Describes outside view, including internals only as far as you need to know them for using the asset and for predicting its fit (behavior, structure) for your cóntext

Composing different models for a full-fledged model for synthesis as the last step in the workflow so far only works in selected use-cases of 3D-printing.



Modeling without means to make models *act* is not a solution in robotics as robots finally need to act.

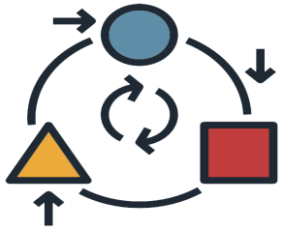
Data sheets (models of artefacs that act) *represent* components, sub-systems, task-plots etc. Suitability, traceability, simulation, etc. of system properties all via *composed data sheets*. When all is fine, then *compose* (put together and accordingly configure) the real artefacts to get the real system with properties as expected.



Composition, Blocks, Ports, Connectors, Data Sheets, Models

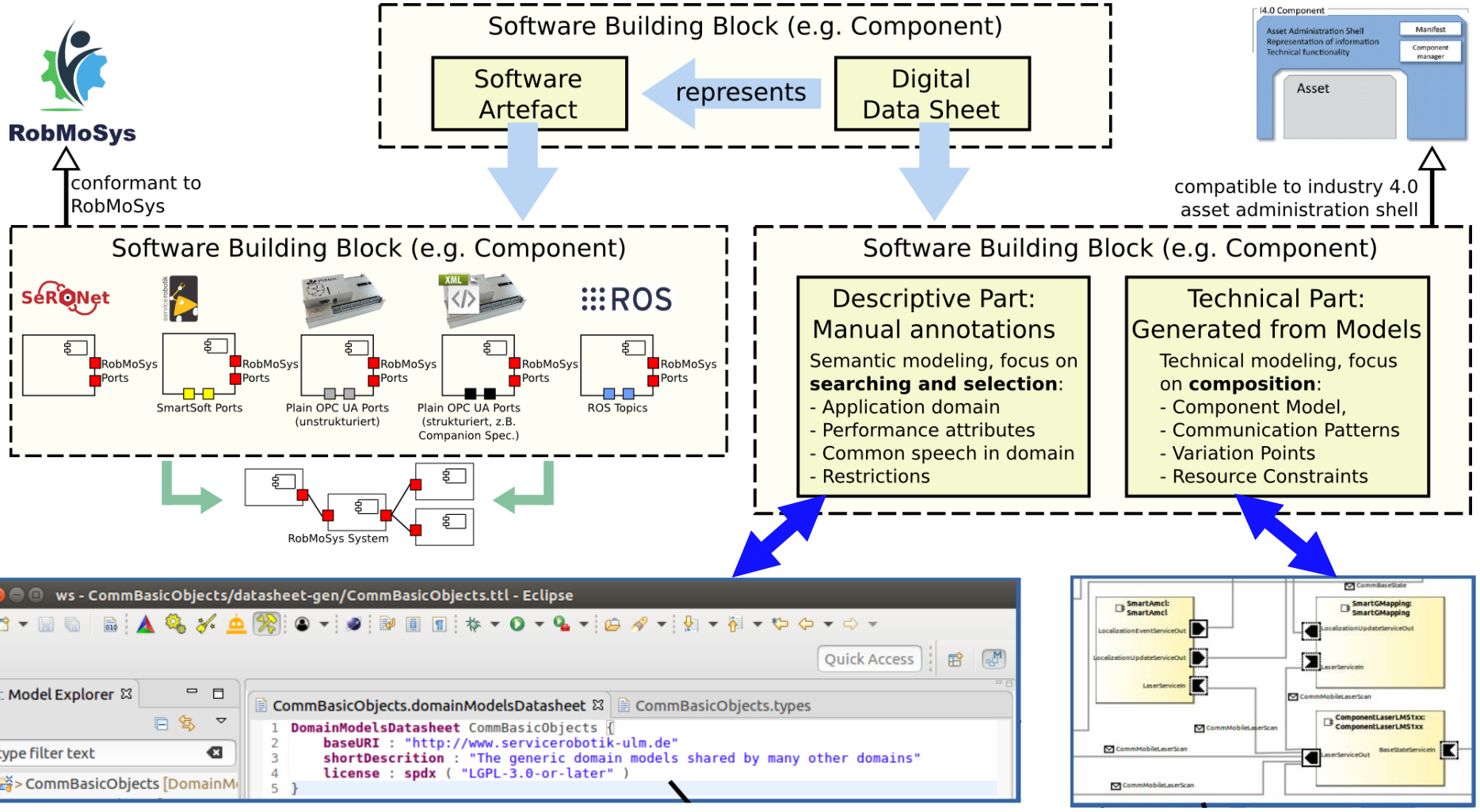
- Brokerage Platform
- Online marketplace

- Component selection
- Component composition
- Component configuration



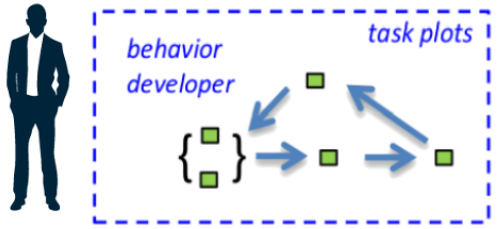
- Runtime adaptation
- Context awareness
- Robustness and self-X

Digital Data Sheet as Submodel of Industry 4.0 Asset Administration Shell

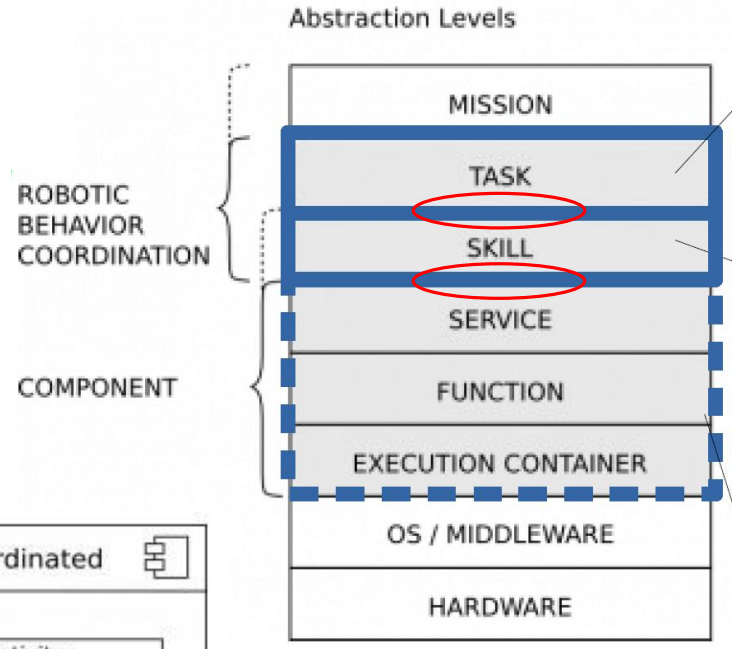


Behavior Coordination: Skill- / Task-Level

- not only „what to do“ (the task), but also „how to do it“ (quality of service, adequateness)
- data sheets for skills: reuse of task models with robots coming with different capabilities

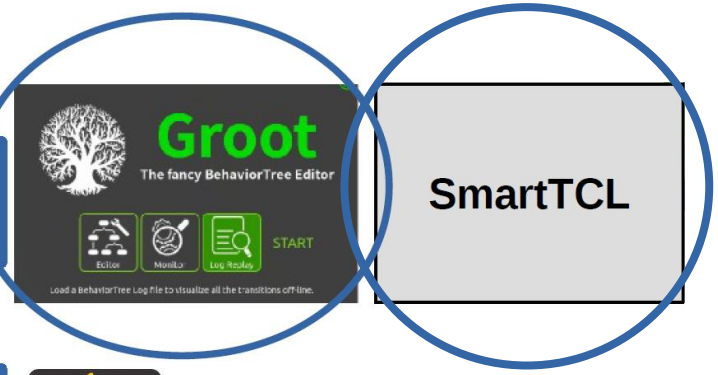


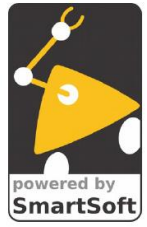
Behavior Coordination: Data Sheets for Skills

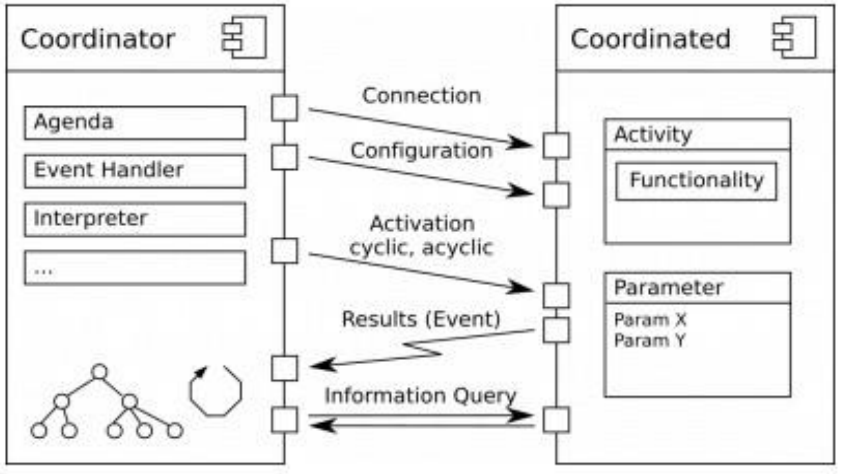


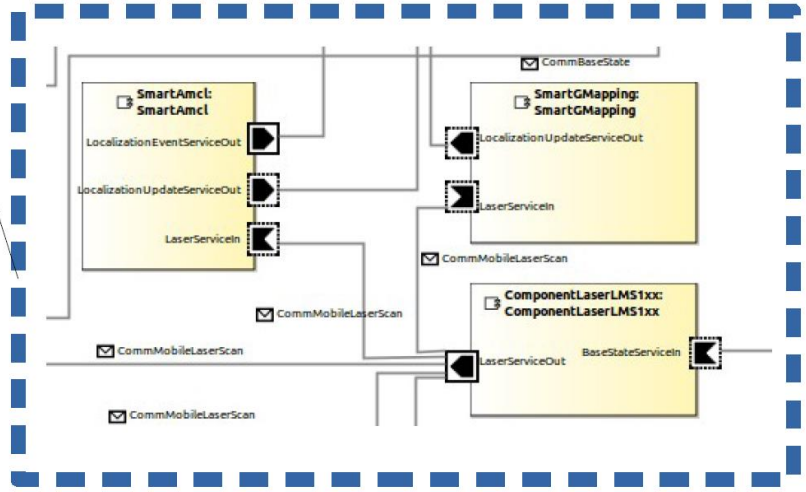
Tasks
Delivery

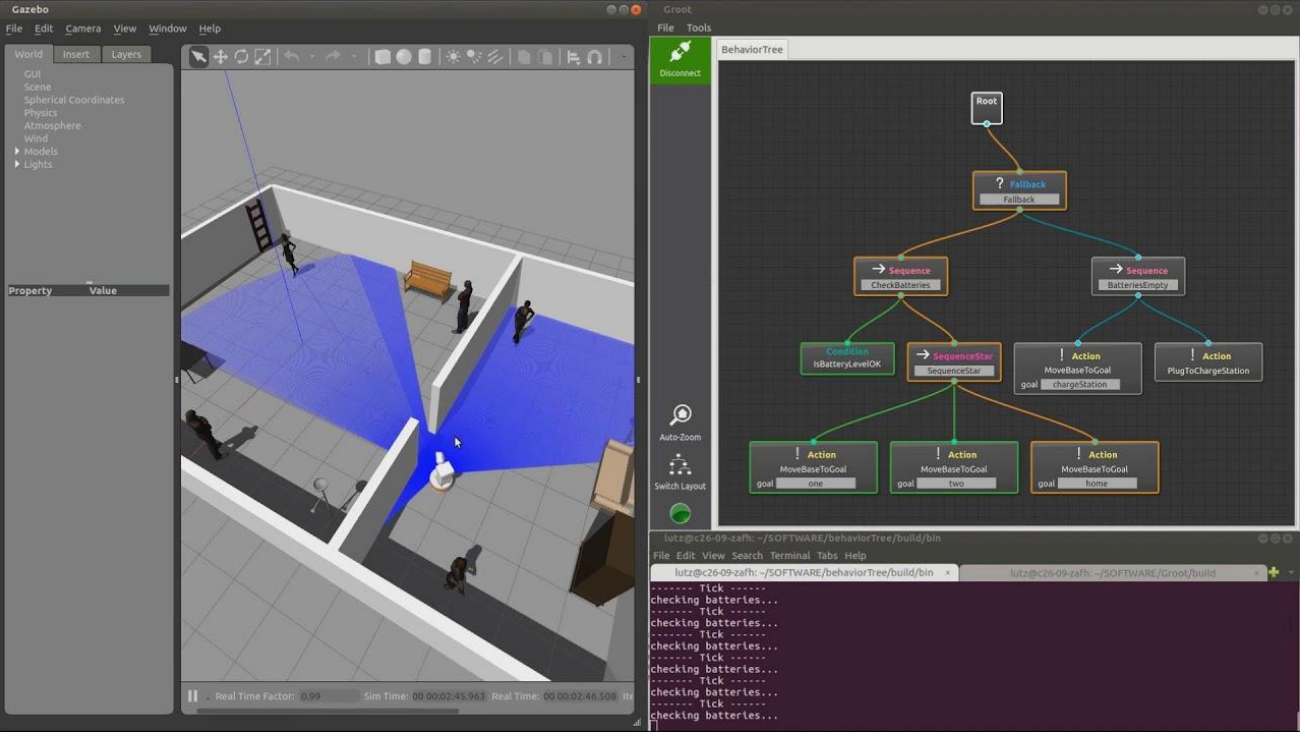
Skills:
pick up box
go to
drop box

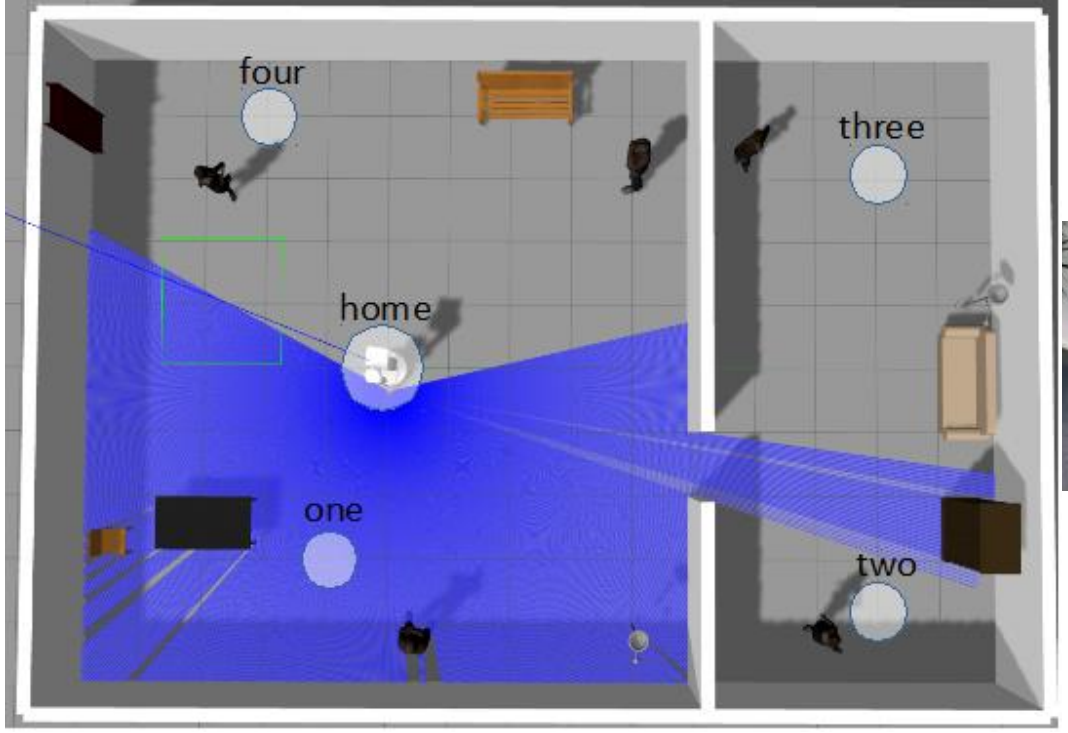




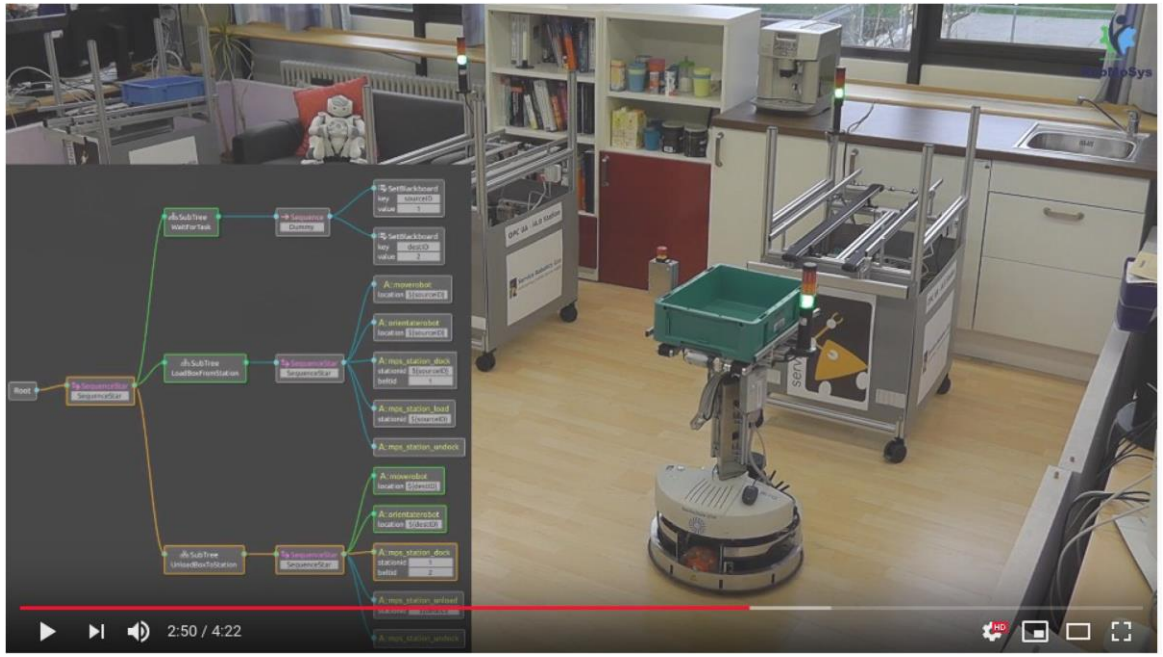






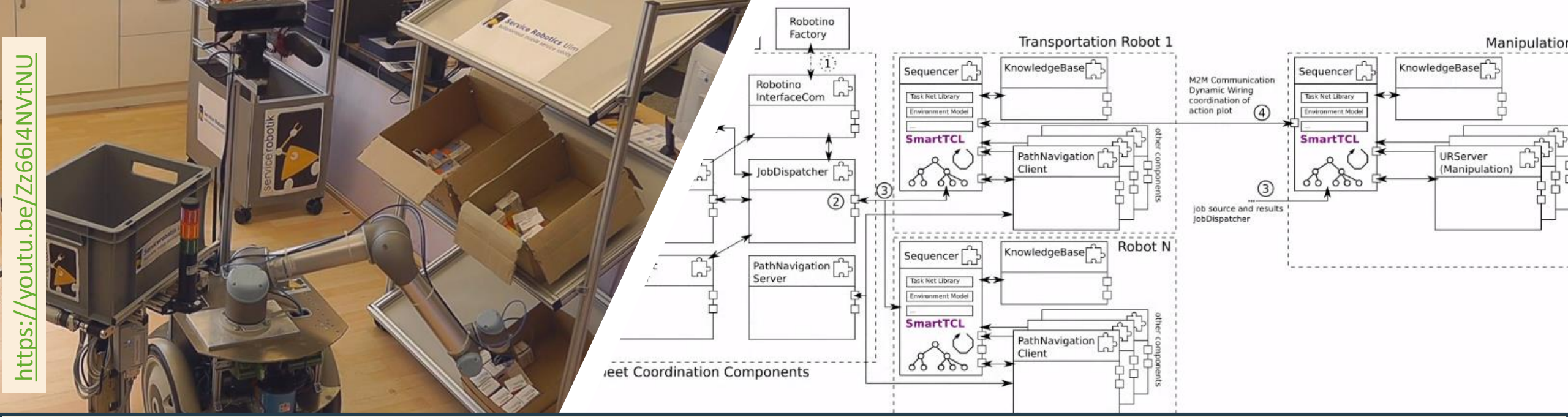


https://robmosys.eu/wiki/community:behavior-tree-demo:start



https://www.youtube.com/watch?v=54_skOuHsds

https://youtu.be/Zz66I4NVtNU



SeRoNet / RobMoSys conformant S/W building blocks

Behavior Coordination
Sequencer (SmartTCL)
SmartSimpleKB
SmartSeq2SeqCom

Navigation
SmartCdlServer
SmartNavigationPlanner
SmartPlannerBreadthFirstSea
SmartMapperGridMap
SmartPathNavigationClient
SmartPurePursuitNavigati

Hardware Abstraction
SmartLaserLMS1xxServer
SmartRobotinoBaseServer
SmartRobotinoImageServer
SmartRobotinoIRServer
SmartRobotinoLaserServer
SmartRobotinoConveyerBeltServer

Localization + SLAM
SmartAmcl
SmartGMapping

Relative Movement

Robot Fleet Co

ion Robot - Larry

vior Coordination
quencer (SmartTCL)
martSimpleKB
SmartSeq2SeqCom

Object Recognition
SmartBoxDetection
SmartRackDetection

Mobile Manipulation
SmartOpenRave

Localization + SLAM

Navigation
SmartMapperGridMap
SmartNavigationPlanner
SmartCdlServer

Hardware Abstraction
SmartPTUServer
SmartKinectV2Server
SmartRMPBaseServer
SmartLaserLMS1xxServer
SmartURServer
SmartVacuumGripper

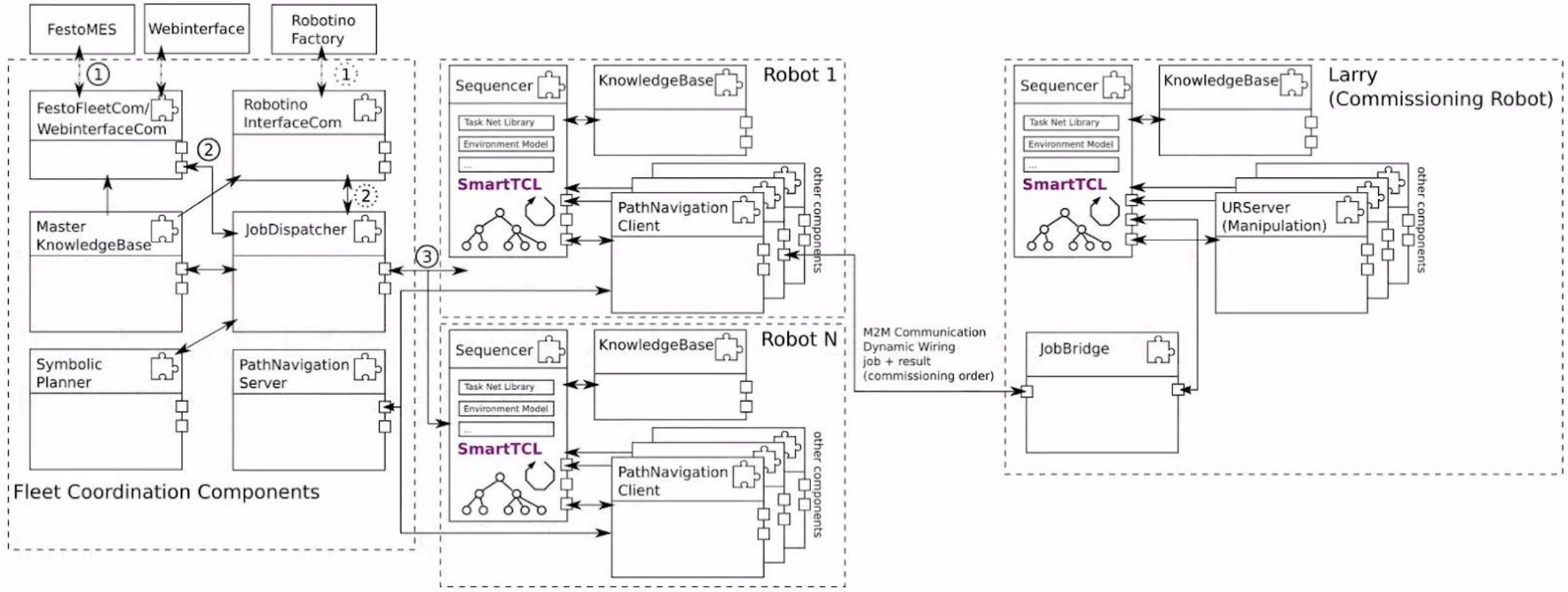
Robot Fleet Communication

Examples of System Composition: Robot Fleet

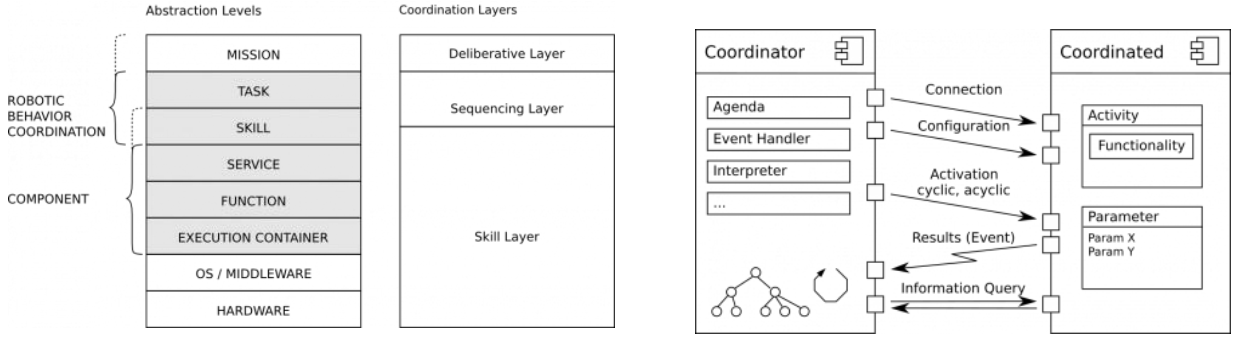
Management-Components (zentral)

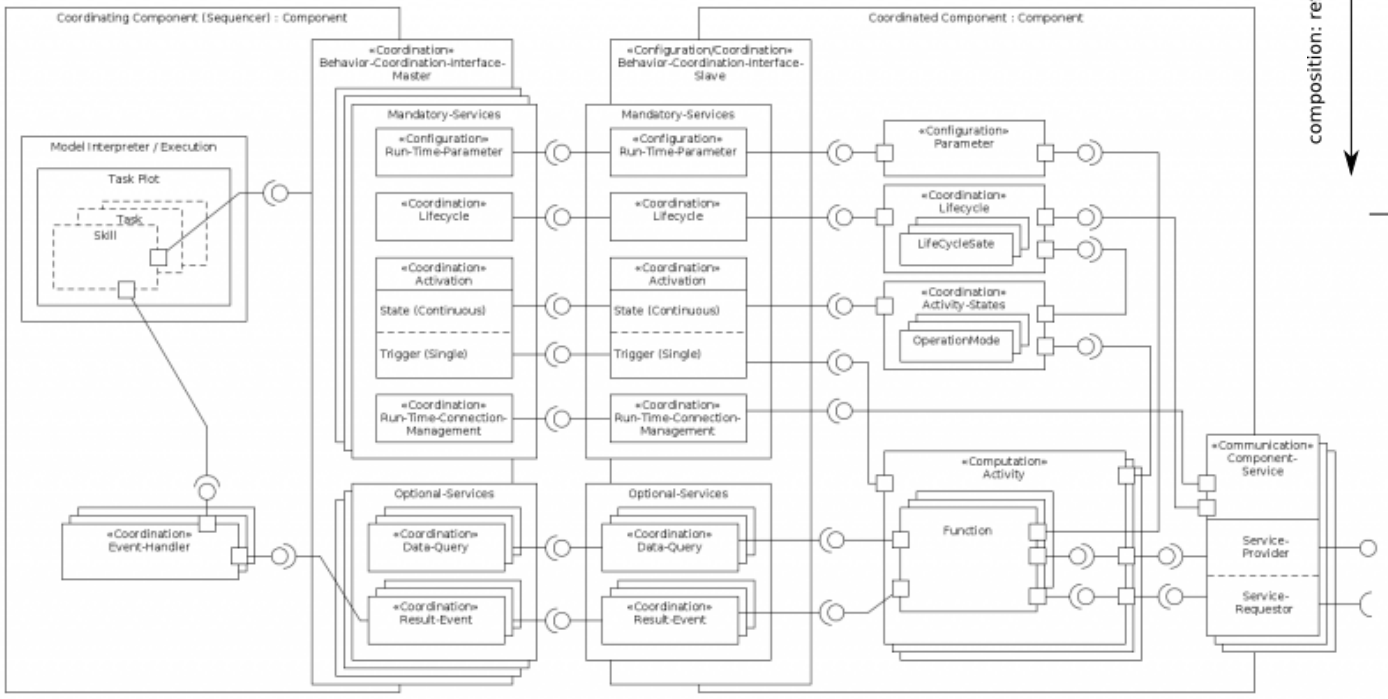
Transportroboter-Komponenten

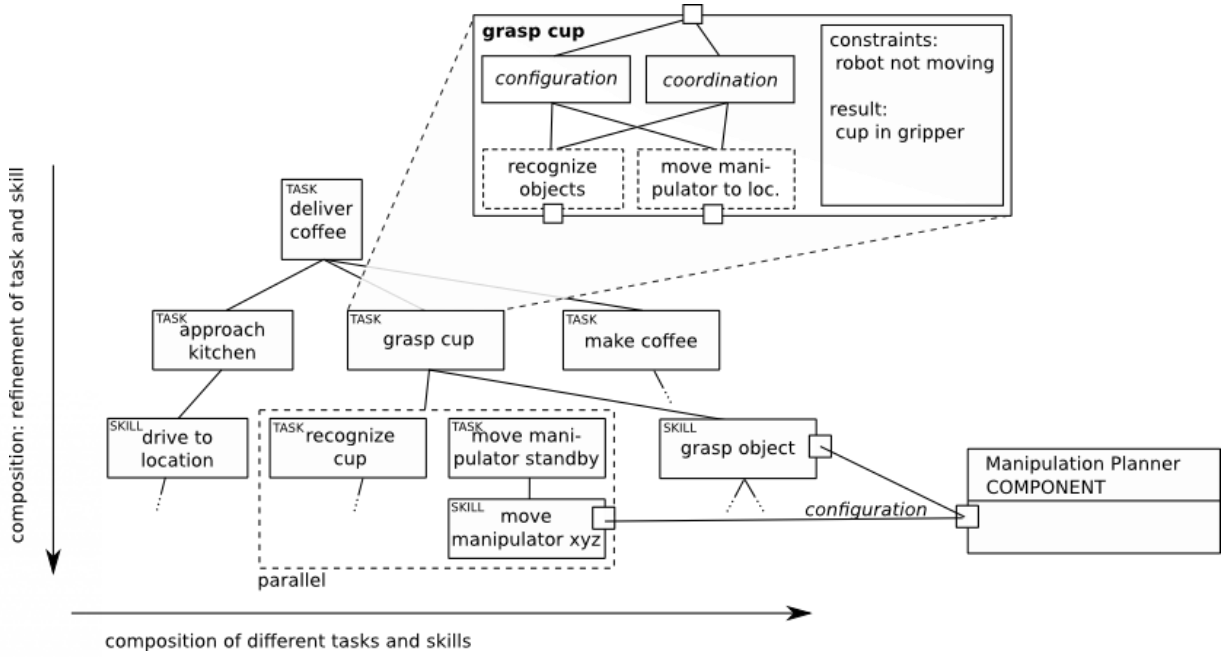
Pickroboter-Komponenten



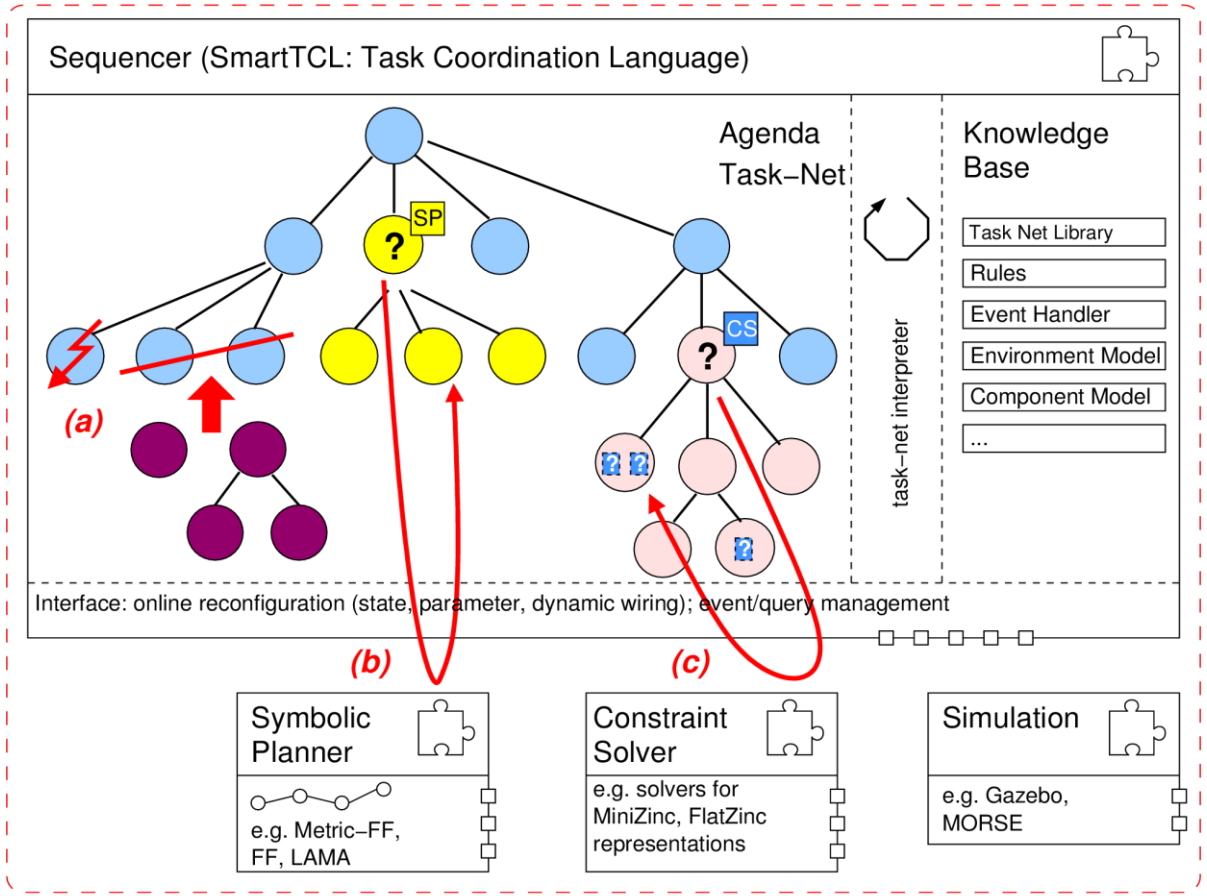
Examples of System Composition: Task Coordination

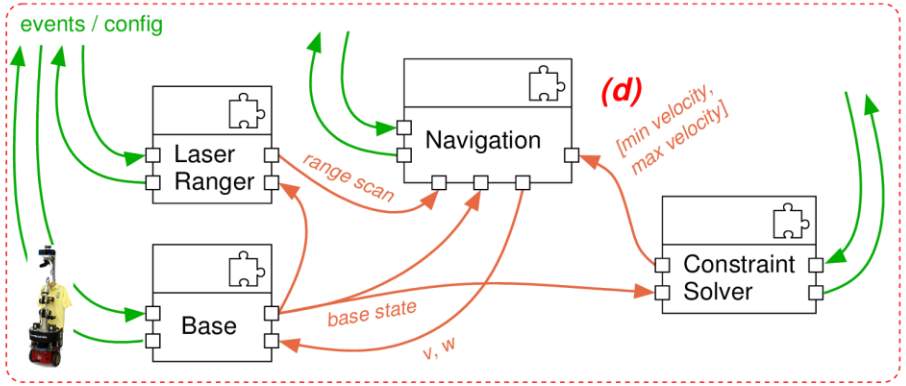






Management of resource shares, execution context constraints, configuration constraints in horizontal and vertical task interactions





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

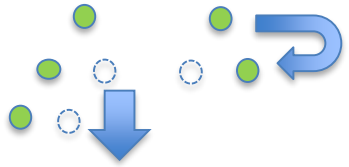
(b) SmartTCL uses a symbolic planner to refine a sub-tree

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

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

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

SmartTCL



- *component operating mode*
- *assignment of resource shares*
- *constraints forwarding*

- *resource gripper fully occupied*
- *constraint „hold always upside"*

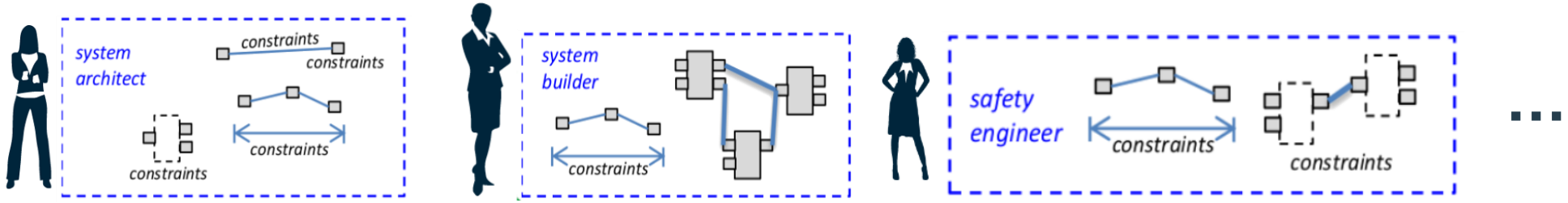
dynamic behavior tree
- parallel (one-of, all), sequential, expansion, …

horizontal
- e.g. resource share reservation in knowledge base

vertical
- e.g. expand task node under constraints and forward constraints

The Concept of Dependency Graphs

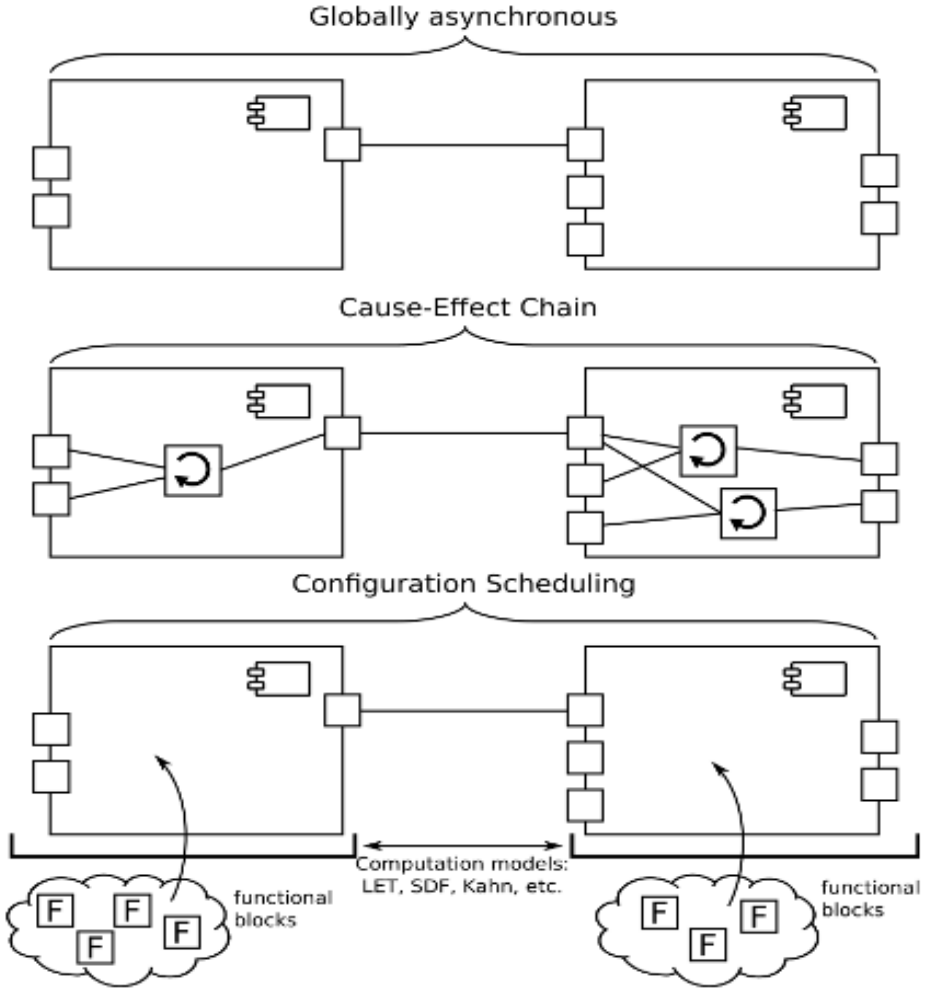
- *Horizontal / vertical composition and the challenge of managing resources*
- *Separation of control flow and data flow*
- *Composability:*
  - *Resource shares, reservation based mechanisms, constraints are composable*
  - *Priorities are not composable*

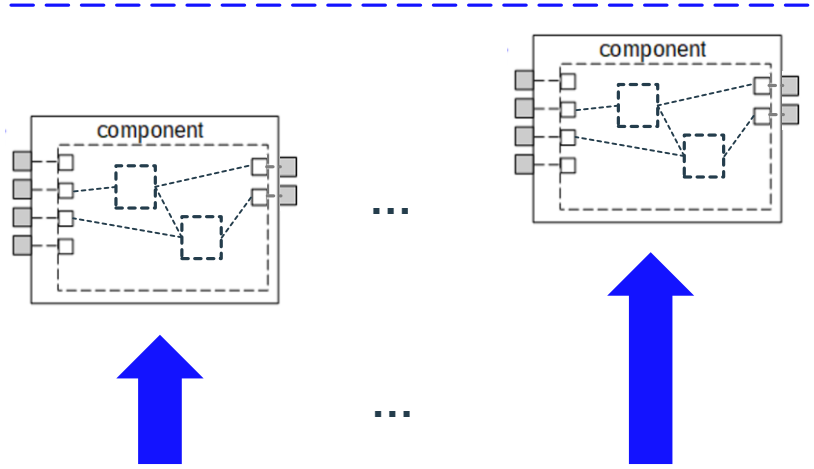


Horizontal / Vertical Composition: Dependency Graphs

Dependency graphs

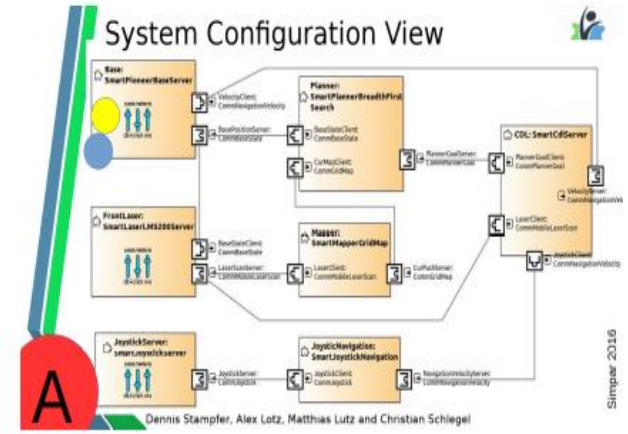
- to model needs for data consistency, data sync, data quality, data aging, cause-effect-chains, etc.
- to configure computation model

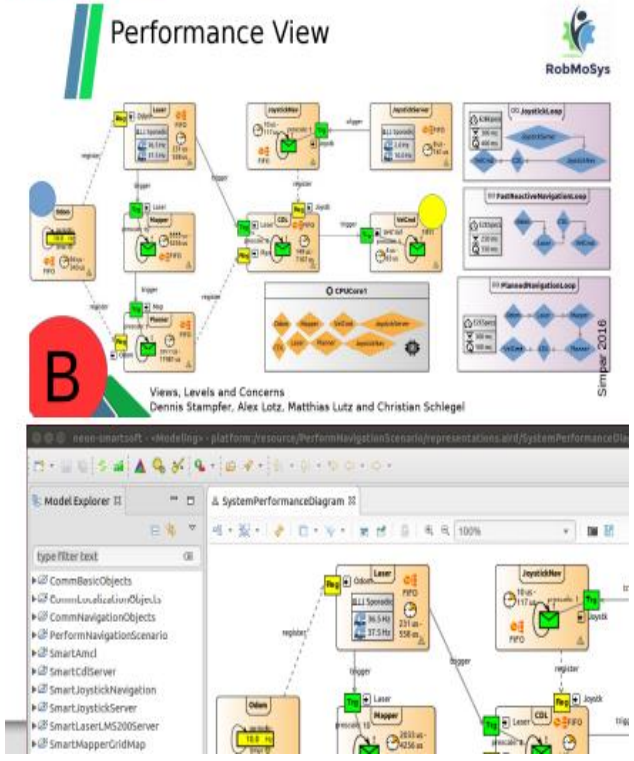




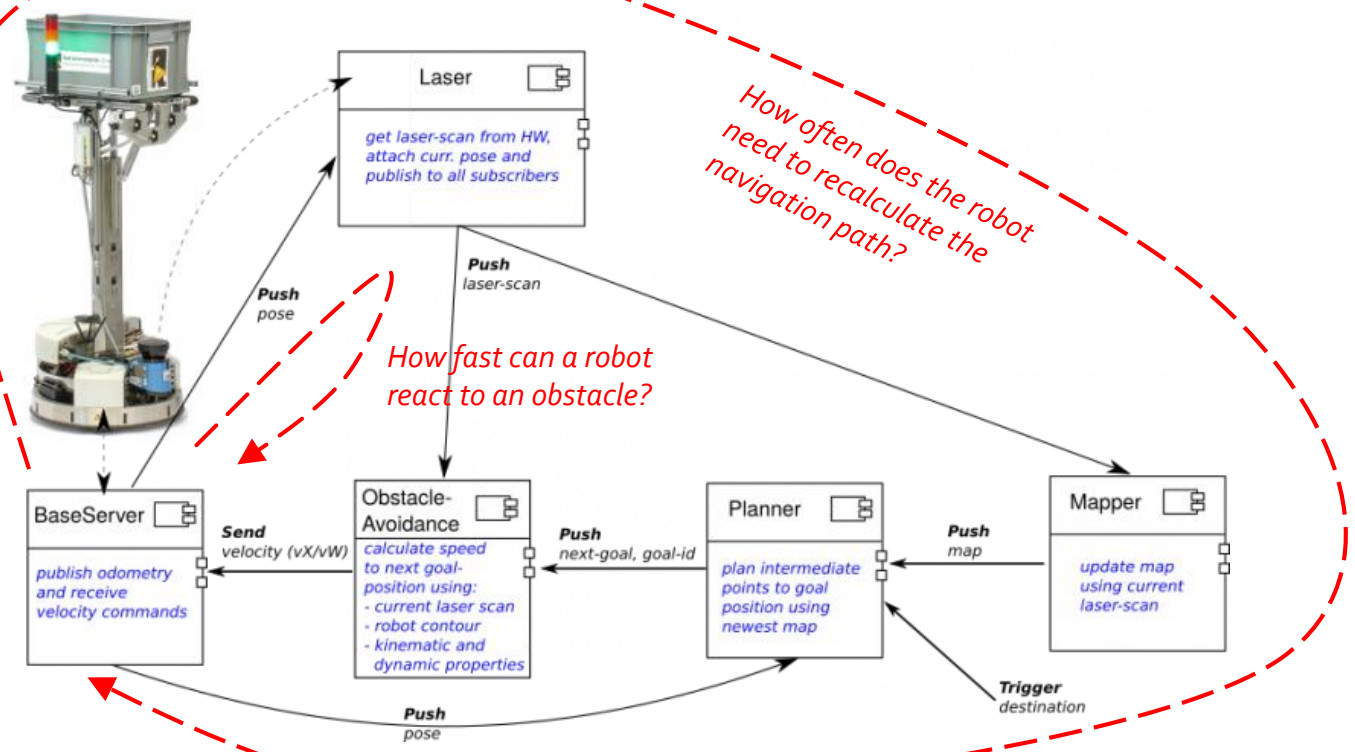
The software component model supports configuration of the following settings at system composition time (as represented in the data sheets) without recompilation:

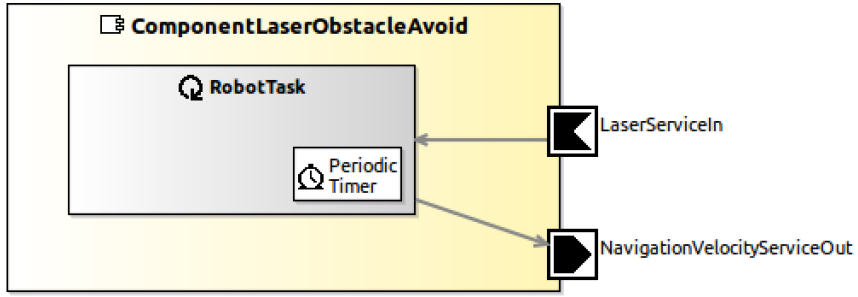
- register / trigger semantic for communication
- port trigger / timed trigger for computation
- scheduling constraints
- sanity checks and run-time monitors
- …

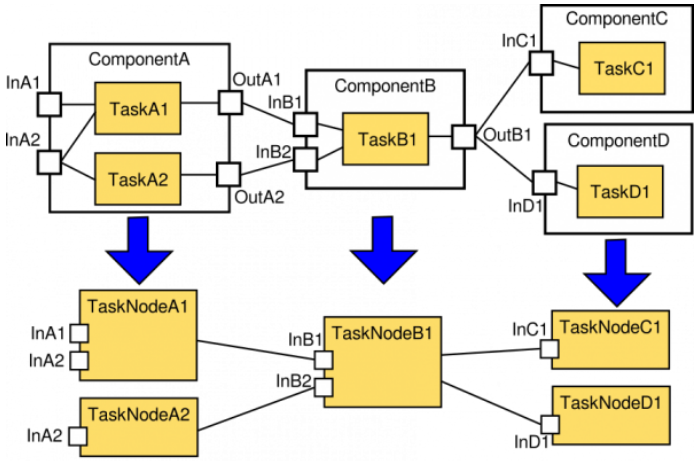


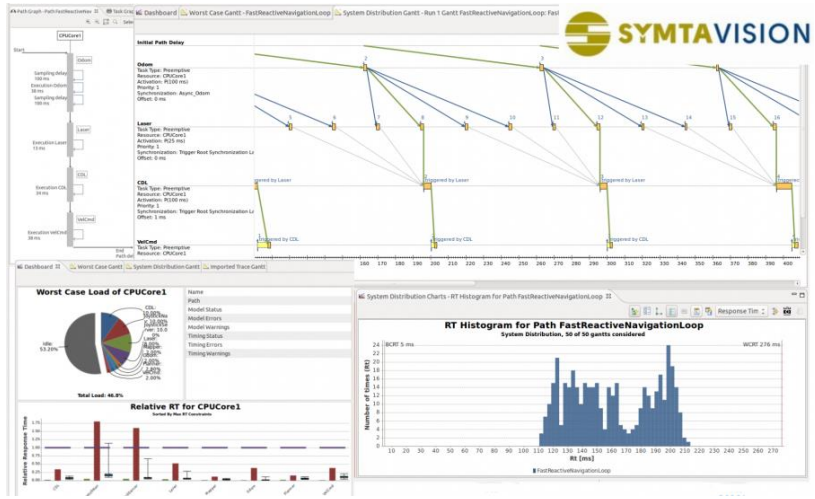


Horizontal / Vertical Composition: Dependency Graphs

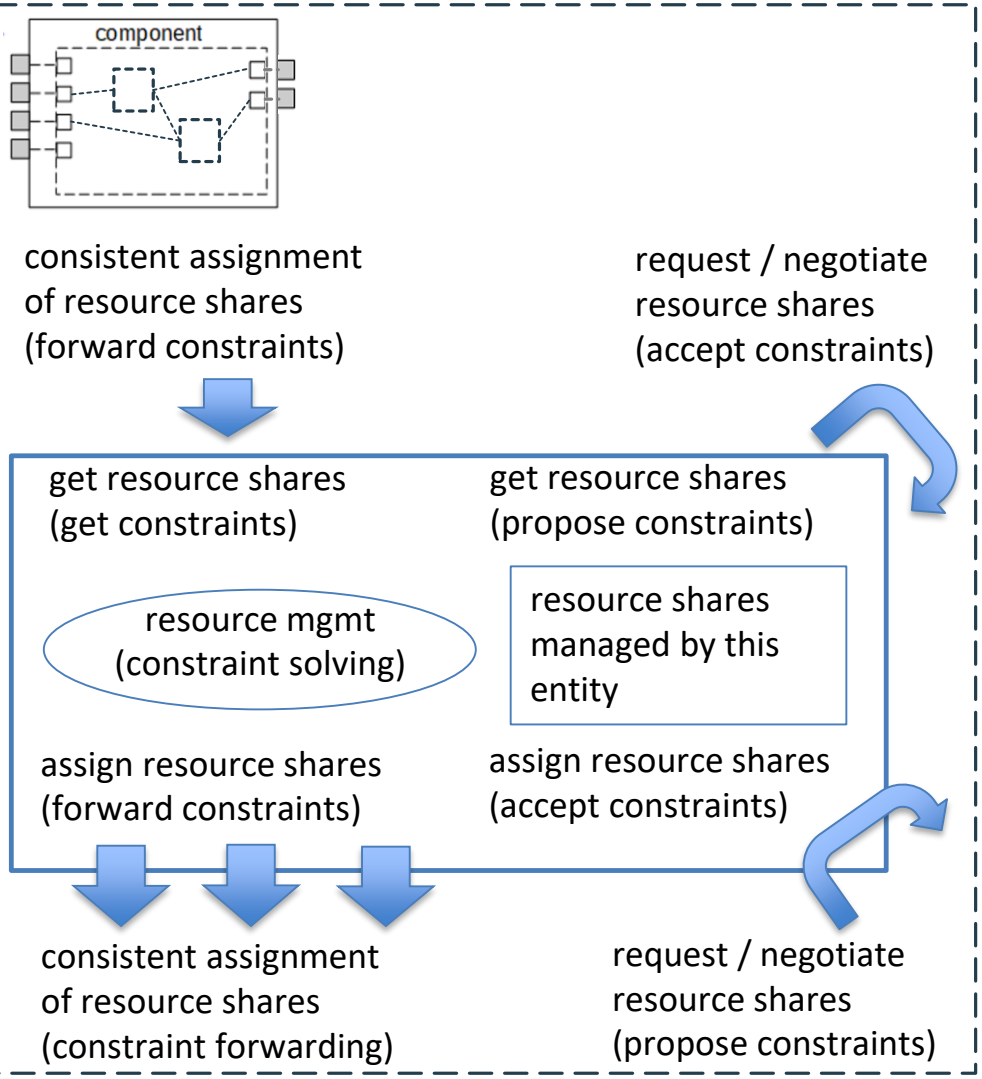


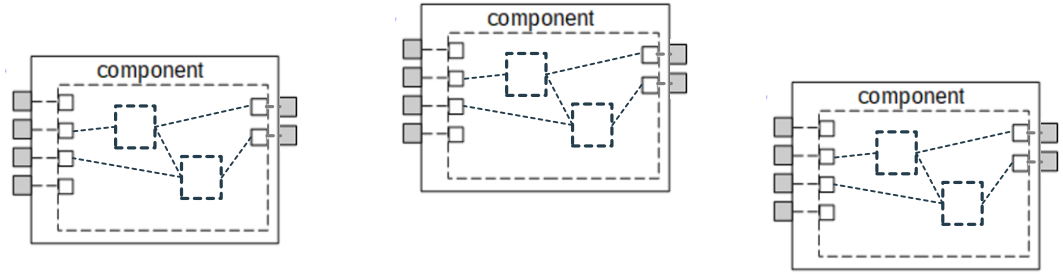


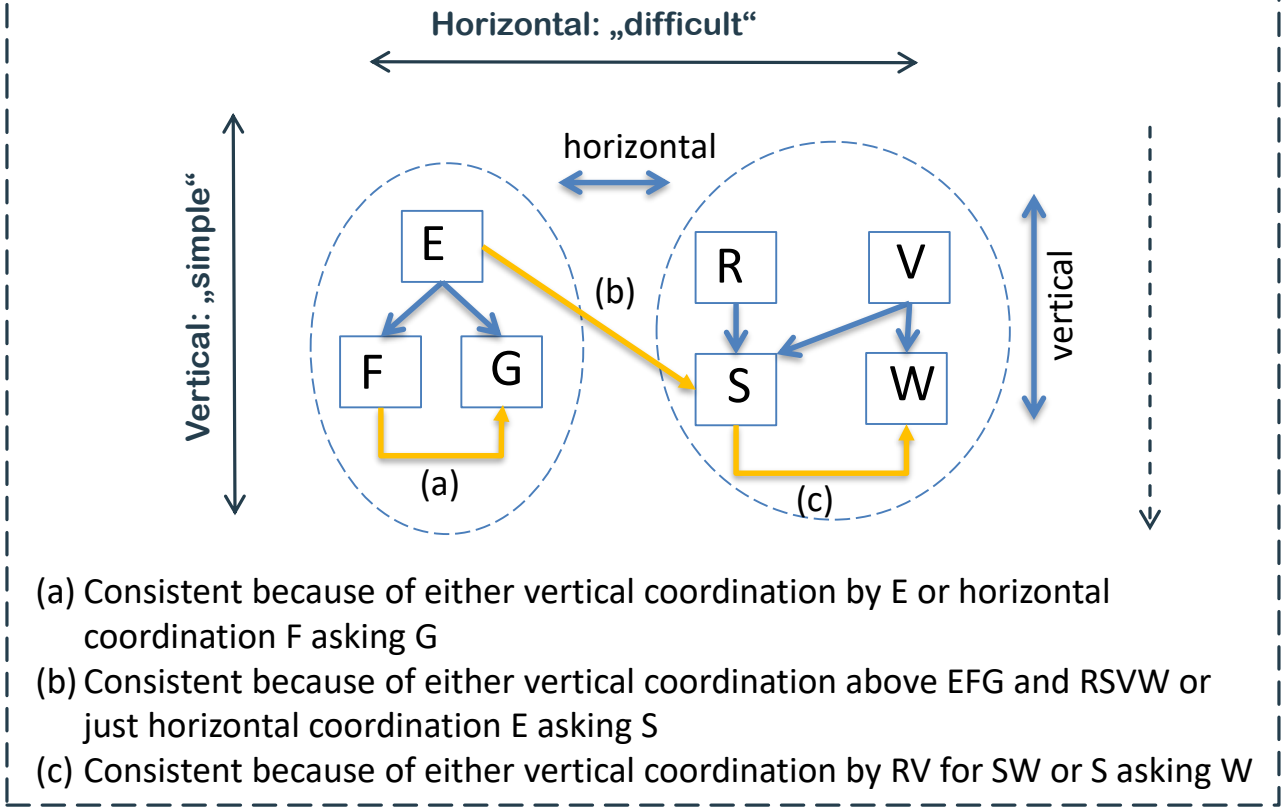




Horizontal / Vertical Composition: Separation of Control / Data Flow, Resource Management







(a) Consistent because of either vertical coordination by E or horizontal coordination F asking G
(b) Consistent because of either vertical coordination above EFG and RSVW or just horizontal coordination E asking S
(c) Consistent because of either vertical coordination by RV for SW or S asking W

I want to use it …
I want to contribute …
I want to …

- Methodology
- Meta Models
- Models
- Implementation Technologies
- Toolings
- Building Blocks
- Pilot Applications
- Repositories
- Processes

RobMoSys Open Access Repositories

RobMoSys

Tier 3 Systems

Name	Description	Purpose	Vendor	Tooling	Status	Figure
SystemTiagoNavigation	A pilot skeleton that covers the navigation aspect of the Intralogistics Industry 4.0 Robot Fleet Pilot and Assistive Mobile Manipulation Pilot. This system covers the TIAGo Robot in ulation/Gazebo.	Navigation	HSU	SmartMDSD Toolchain v3	Ready	
	ot skeleton that covers navigation aspect of ntralogistics Industry	Navigation	HSU	SmartMDSD Toolchain v3	Ready	-

https://robmosys.eu/wiki/model-directory:start

RobMoSys Wiki
http://www.robmosys.eu

Search

Recent Changes Media Manager Sitemap Mainpage Imprint

You are here: RobMoSys Wiki » RobMoSys Model Directory

RobMoSys Model Directory

A list of domain models, software components and systems for use with RobMoSys Tooling. Please see end of page for a legend.

Table of Co
- RobMoSys
- Tier 2 Do
- Tier 3 Co
- Tier 3 Sy
- Explanat

Tier 2 Domain Models

Name	Description	Purpose	Vendor	Toolin
CommBasicObjects	A collection of **very basic service definitions and communication objects** for use in almost every robotics system.	Universal	HSU	Smart Toolch
CommNavigationObjects	A collection of domain models for **wheeled robot navigation**.	Navigation	HSU	Smart Toolch
CommRobotinoObjects	A collection of domain models for use with the **FESTO Robotino** robot.	Mobile-Base	HSU	Smart Toolch
CommLocalizationObjects	A collection of domain models for **localization**.	Localization	HSU	Smart Toolch
CommManipulationPlannerObjects	A collection of domain models for (mobile) manipulation.	Mobile Manipulation	HSU	Smart Toolch
CommManipulatorObjects	A collection of domain models for manipulators.	Manipulation	HSU	Smart Toolch

Tier 3 Component Models

Name	Description	Purpose	Vendor	Tooling	Status
SmartCdlServer	Implements the Curvature Distance Lookup (CDL) algorithm for fast local obstacle avoidance. It considers the dynamics and kinematics of the robot, as well as its polygonal shape.	Navigation	HSU	SmartMDSD Toolchain v3	Ready
ComponentLaserObstacleAvoid	The SmartLaserObstacleAvoid component implements a simple reactive obstacle avoidance.	Navigation	HSU	SmartMDSD Toolchain v3	Ready
ComponentPlayerStageSimulator	The SmartPlayerStageSimulator simulates a robot in a 2D bitmapped environment using Player/Stage. It offers several services for controlling the robot, such as sending navigation commands, providing access to the robot's odometry and laser scans.	Simulation	HSU	SmartMDSD Toolchain v3	Ready

Composing a Robotics Application in a Day – A low code approach
We make Robotics Software Systems Engineering easier!

Service Robotics Ulm
autonomous mobile service robots

https://wiki.servicerobotik-ulm.de/start

https://wiki.servicerobotik-ulm.de/smartmdsd-toolchain:start

- ***one-click download*** *of the full Open-Source Eclipse-based development environment*
- ***start development with zero installation effort***
- *comes with Gazebo-Simulator and all kinds of components, stacks, pilot applications, tutorials, etc.*
- *skill-based engineering, task-level coordination, robot fleet coordination, graphical tools for end-users*
- *fully middleware-agnostic: ACE, DDS, OPC UA, etc.*
- *mixed-port component as migration path: link to ROS, I4.0 OPC UA, etc.*

https://robmosys.eu/wiki/baseline:start

https://robmosys.eu/wiki/jumppage

https://robmosys.eu/wiki/open-call-2