DE LA RECHERCHE À L'INDUSTRIE

# Developing ROS2 systems with Papyrus for Robotics

SHARC Day, GDR Robotique 09/07/2021

Matteo MORELLI, Ansgar RADERMACHER CEA-LIST/DILS/LSEA

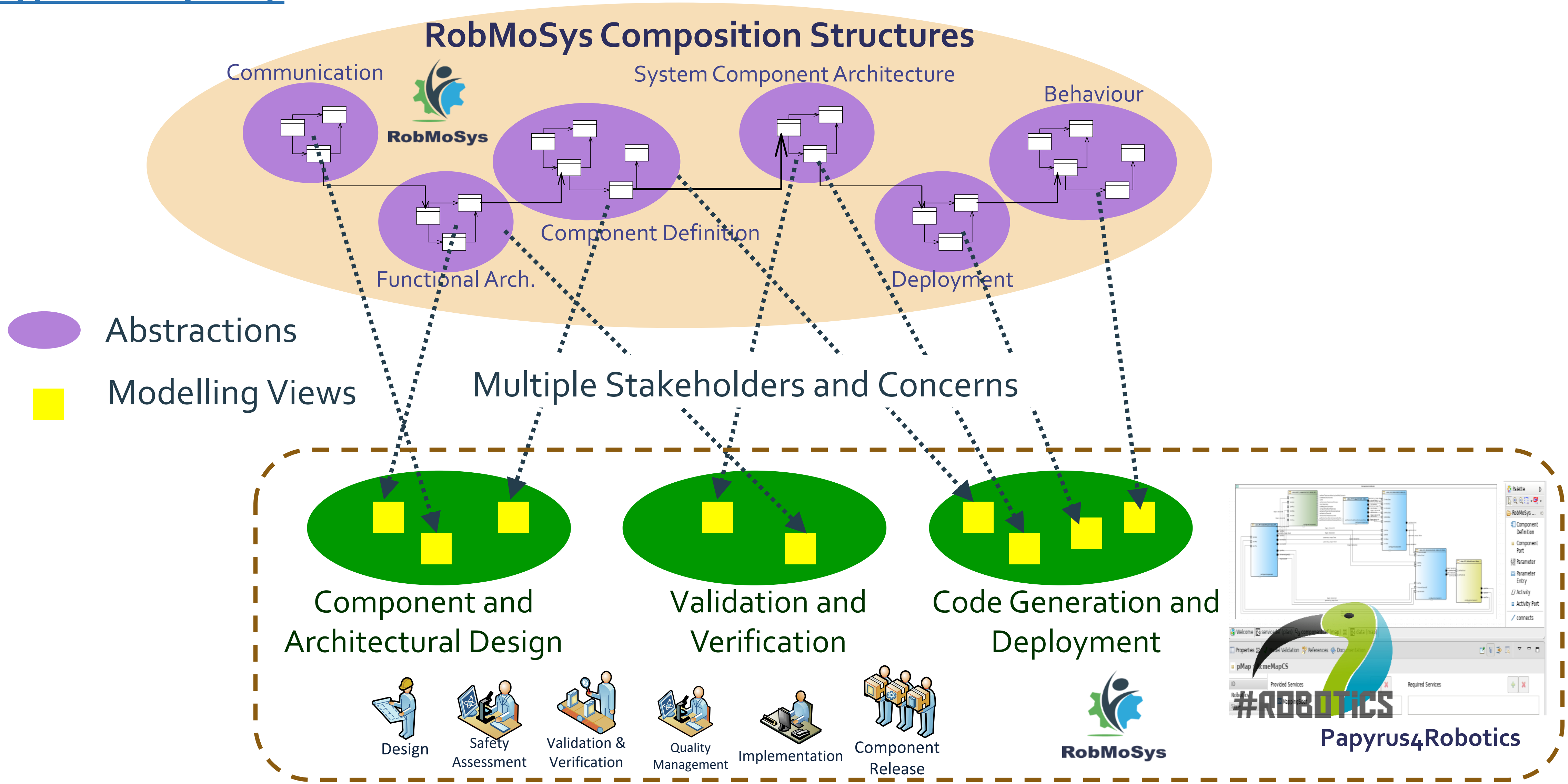
# Robotics software development challenges

**Robotics software engineering** is still in a **"craft" stage** compared to software engineering in more advanced domains such as **automotive** and **avionics**

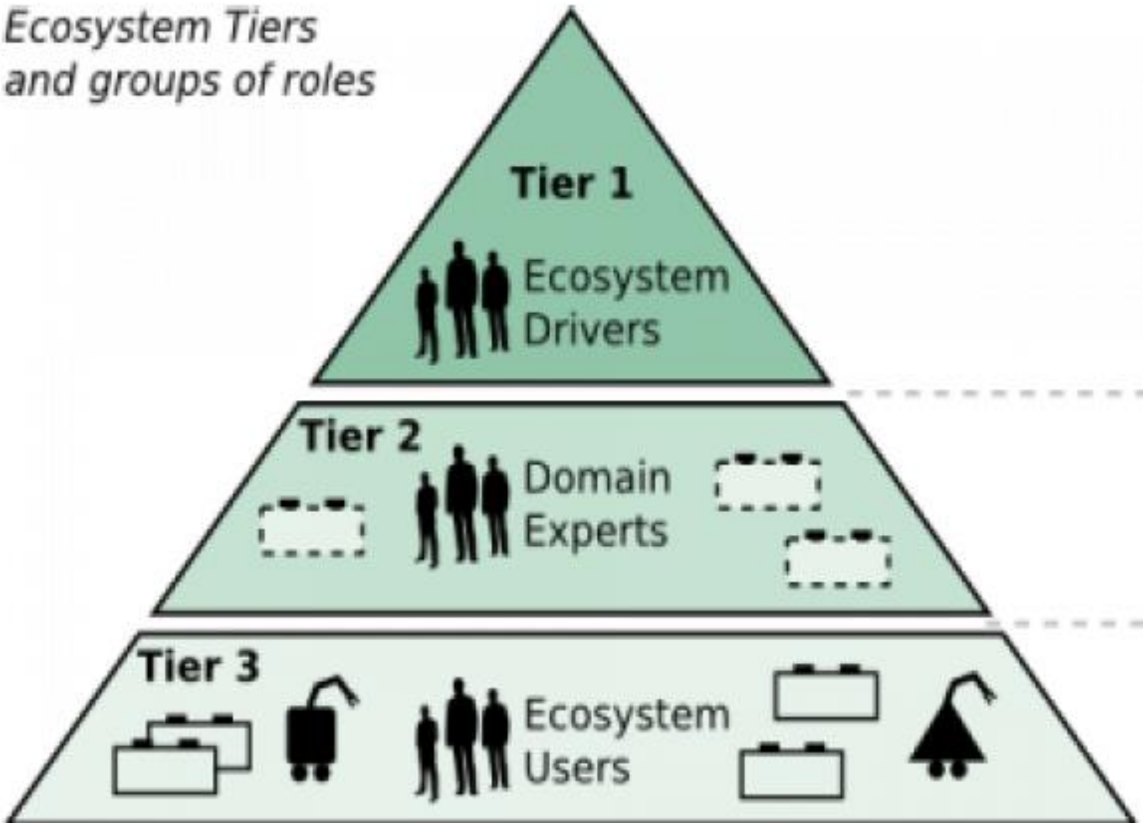
- Numerous and often incompatible component-based frameworks
- Code-centric software development approach
- No standard interfaces between roles involved in developing robotics systems
- Participants involved with heterogenous backgrounds and skills

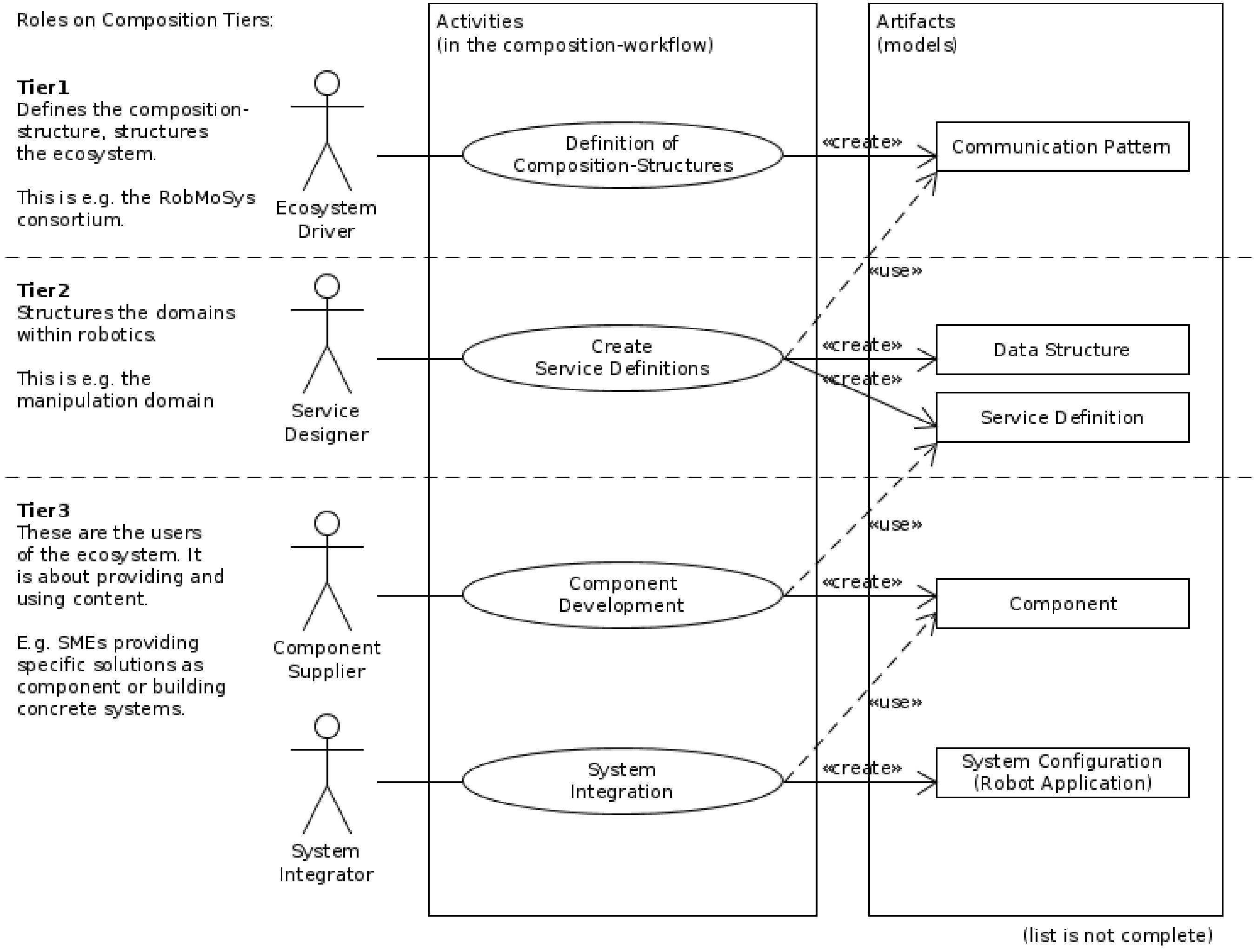
# RobMoSys H2020 — Composable Models and Software for Robotics Systems

https://robmosys.eu/



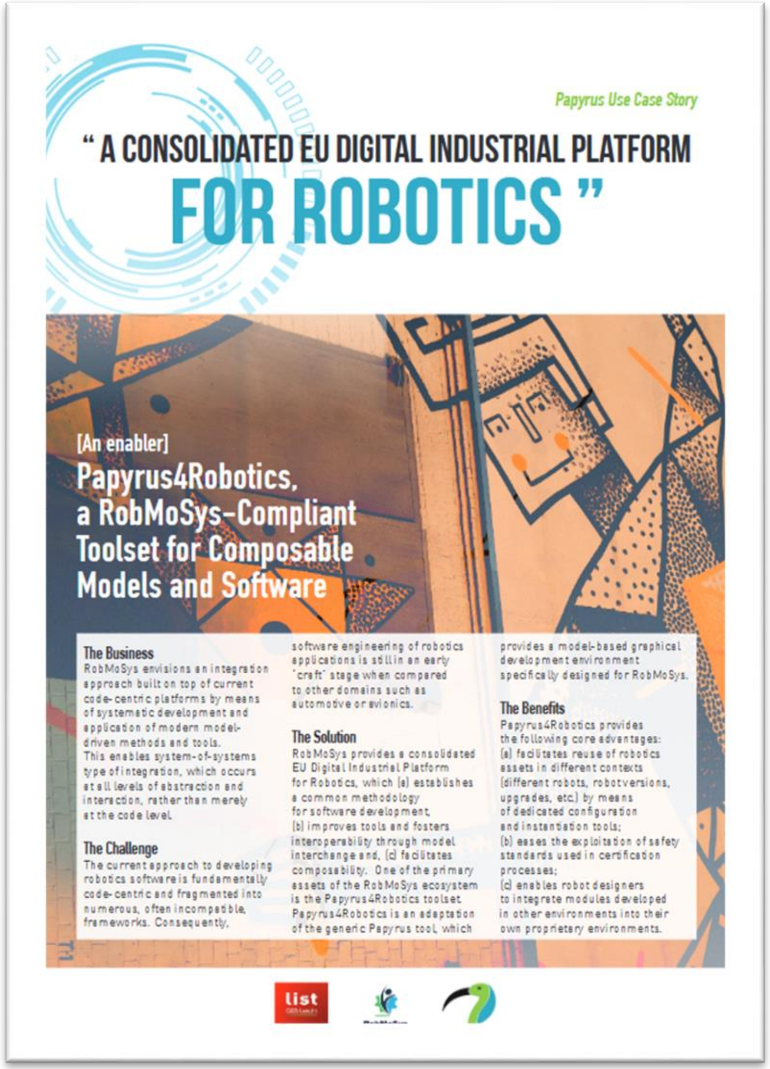
# Ecosystem Tiers and Roles



[ **Papyrus for Robotics** is a model-based development environment that **supports** the **RobMoSys** methodology ]

- **Modular and role-based design**
  provide dedicated views and abstractions for the different stakeholders in the robotics value chain
- **Code-generation**
  transform models of software architectures, platform descriptions and deployment specifications into code
- **Reverse engineering**
  build component and service models from existing systems
- **Behavior tree execution**
  enable modeling of reactive and composable robot behaviors
- **Safety analysis**
  perform dysfunctional analysis on component architectures


Papyrus Use Case Story

" A CONSOLIDATED EU DIGITAL INDUSTRIAL PLATFORM FOR ROBOTICS "

[An enabler]

Papyrus4Robotics, a RobMoSys-Compliant Toolset for Composable Models and Software

**The Business**

RobMoSys envisions an integration approach built on top of current code-centric platforms by means of systematic development and application of modern model-driven methods and tools. This enables system-of-systems type of integration, which occurs at all levels of abstraction and interaction, rather than merely at the code level.

**The Challenge**

The current approach to developing robotics software is fundamentally code-centric and fragmented into numerous, often incompatible, frameworks. Consequently, software engineering of robotics applications is still in an early "craft" stage when compared to other domains such as automotive or avionics.

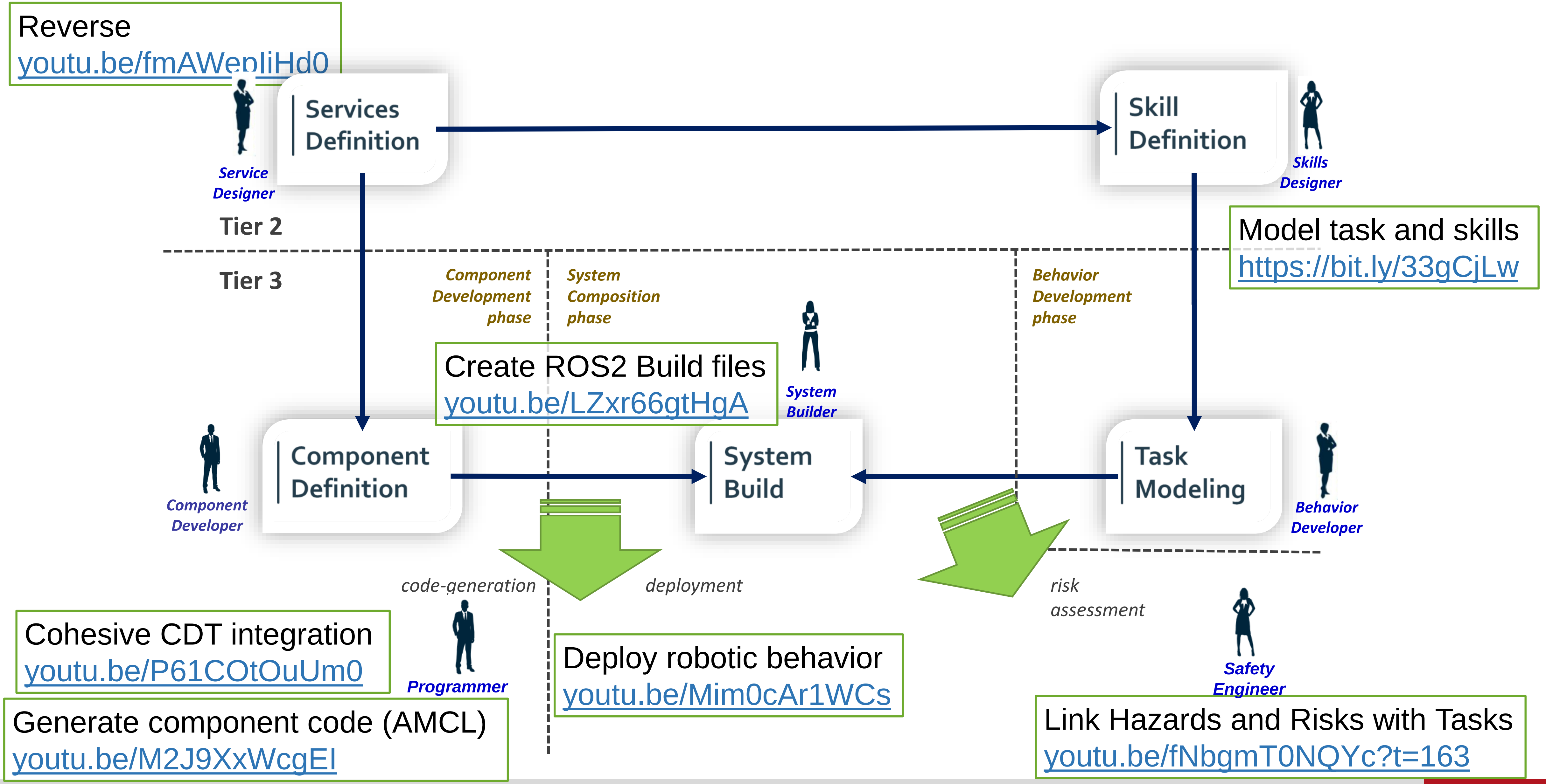
**The Solution**

RobMoSys provides a consolidated EU Digital Industrial Platform for Robotics, which (a) establishes a common methodology for software development, (b) improves tools and fosters interoperability through model interchange and, (c) facilitates composability. One of the primary assets of the RobMoSys ecosystem is the Papyrus4Robotics toolset. Papyrus4Robotics is an adaptation of the generic Papyrus tool, which provides a model-based graphical development environment specifically designed for RobMoSys.

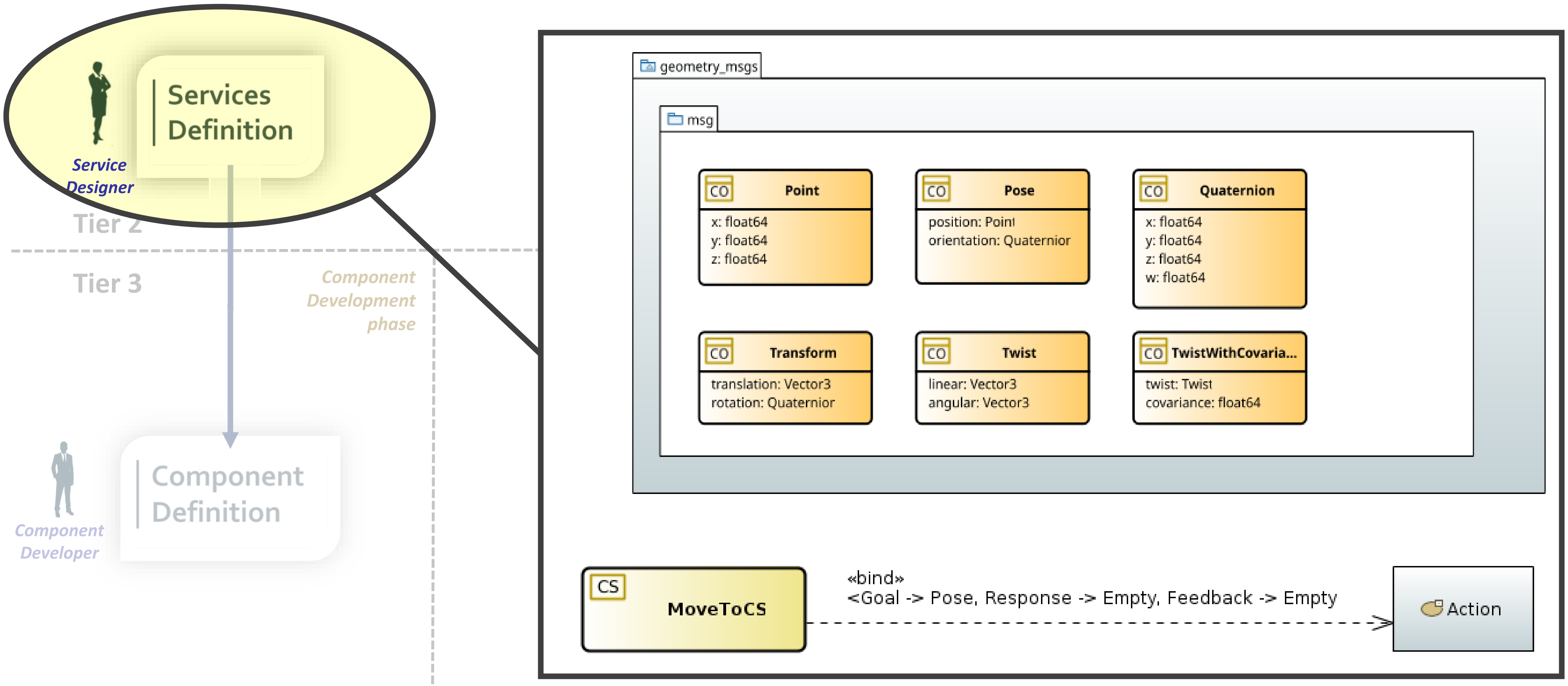
**The Benefits**

Papyrus4Robotics provides the following core advantages:
(a) facilitates reuse of robotics assets in different contexts (different robots, robot versions, upgrades, etc.) by means of dedicated configuration and instantiation tools;
(b) eases the exploitation of safety standards used in certification processes;
(c) enables robot designers to integrate modules developed in other environments into their own proprietary environments.

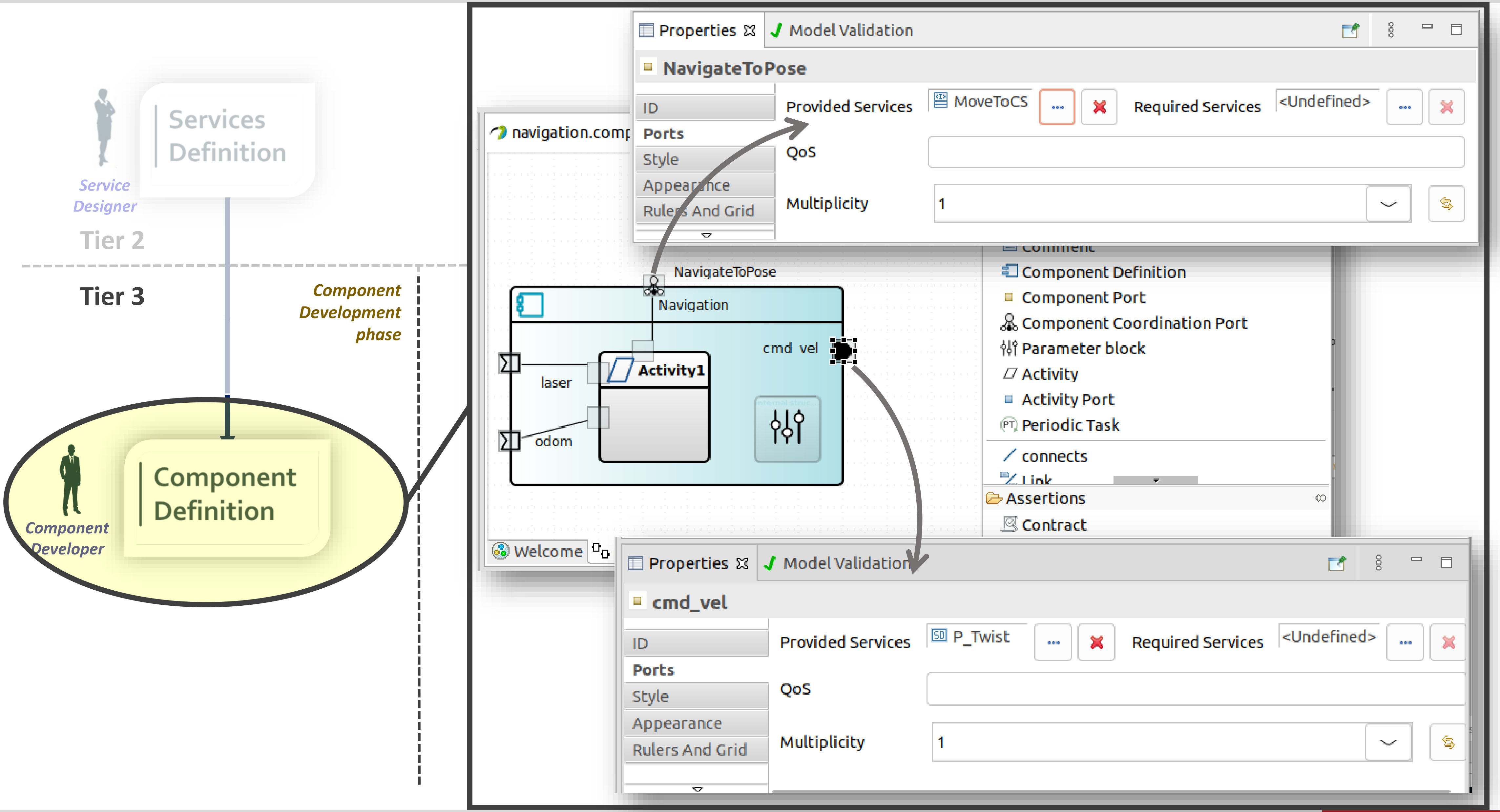
# Roles and interactions



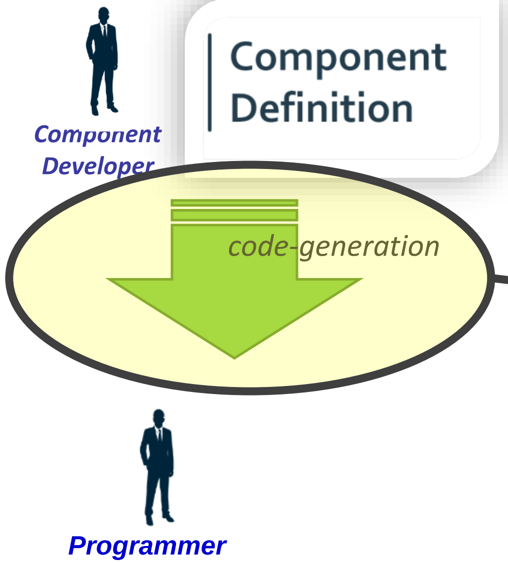
## Ros2 data types and services (components' data flows / configuration&coordination)

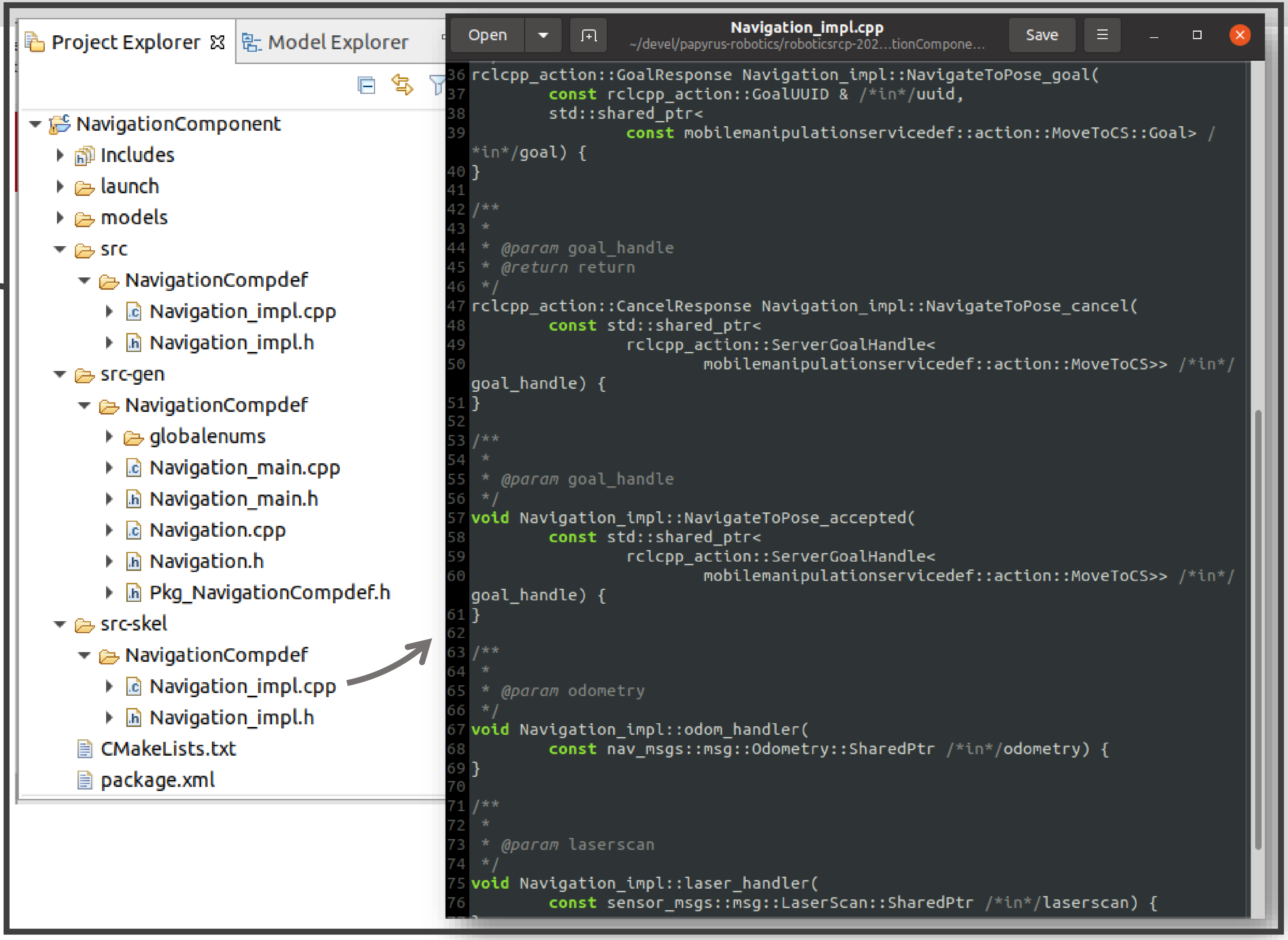


# Services and components



# Component code-generation

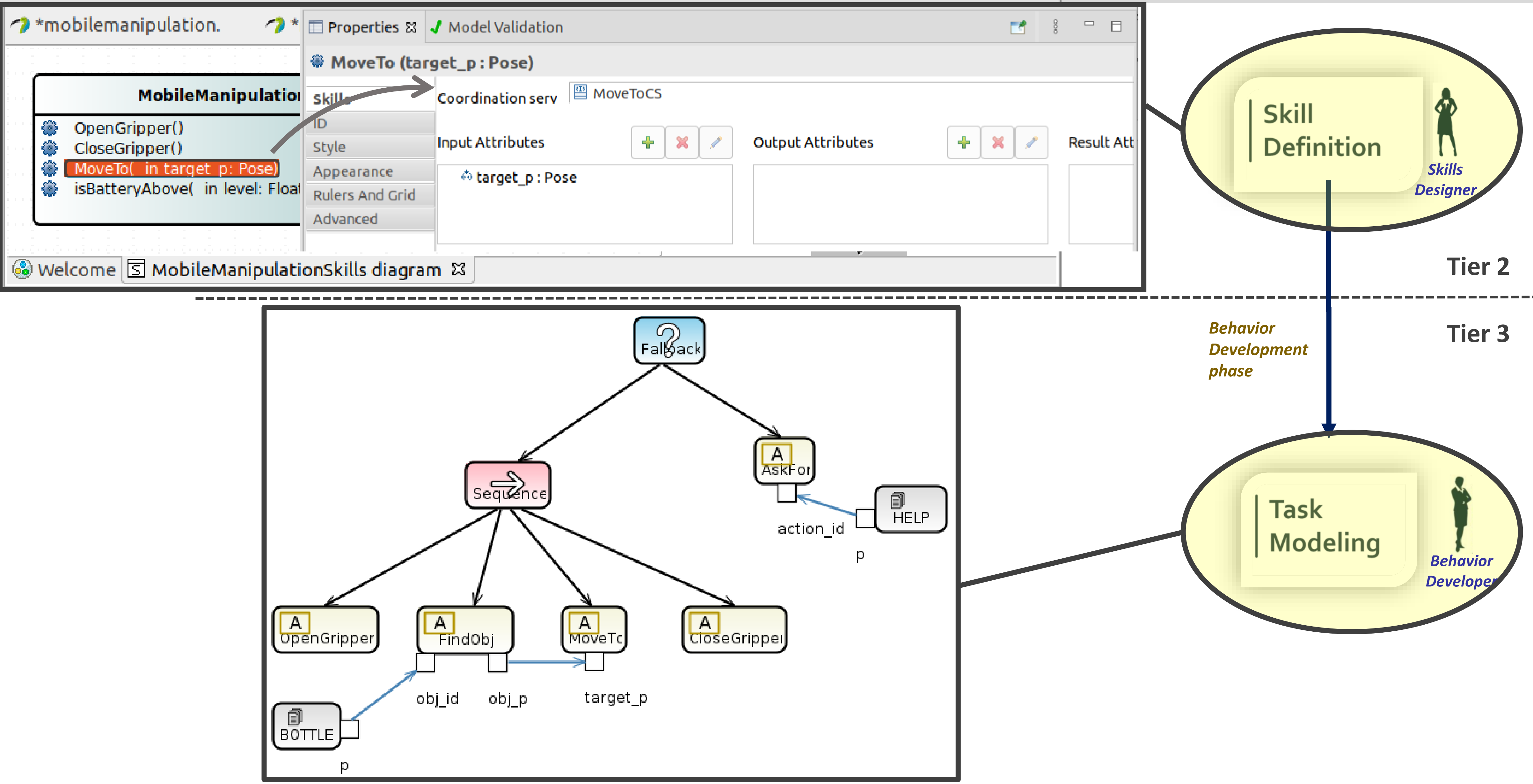


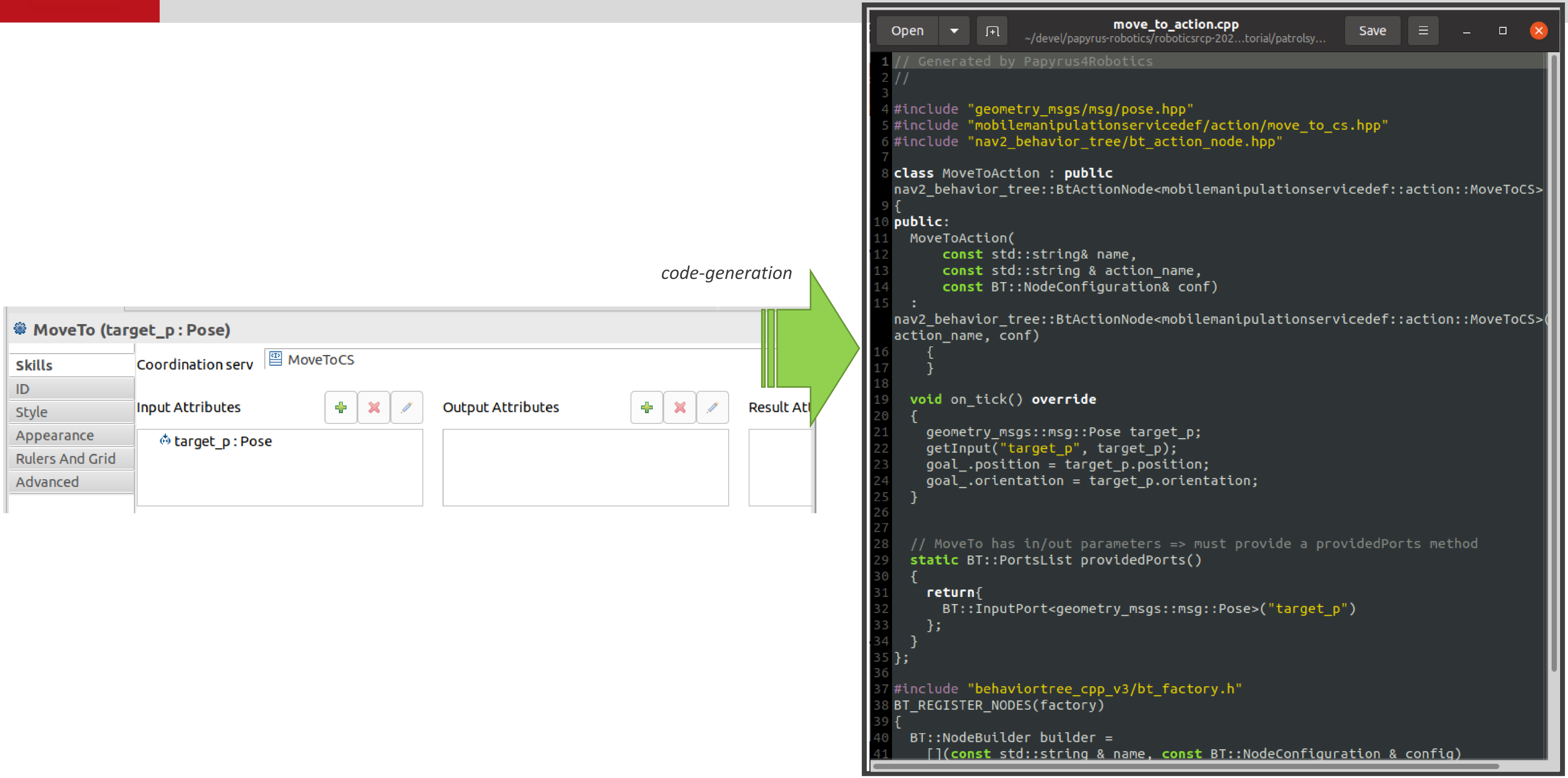

binds data from component interface to internals (algorithms, etc.)

- `const auto goal = goal_handle->get_goal();`
- `res = do_navigate_algo ( goal->position.x, goal->position.y, … );`
- `if (res == 1) { // algo OK } else {algo KO }`

# Skills and behaviors



# Skill code-generation

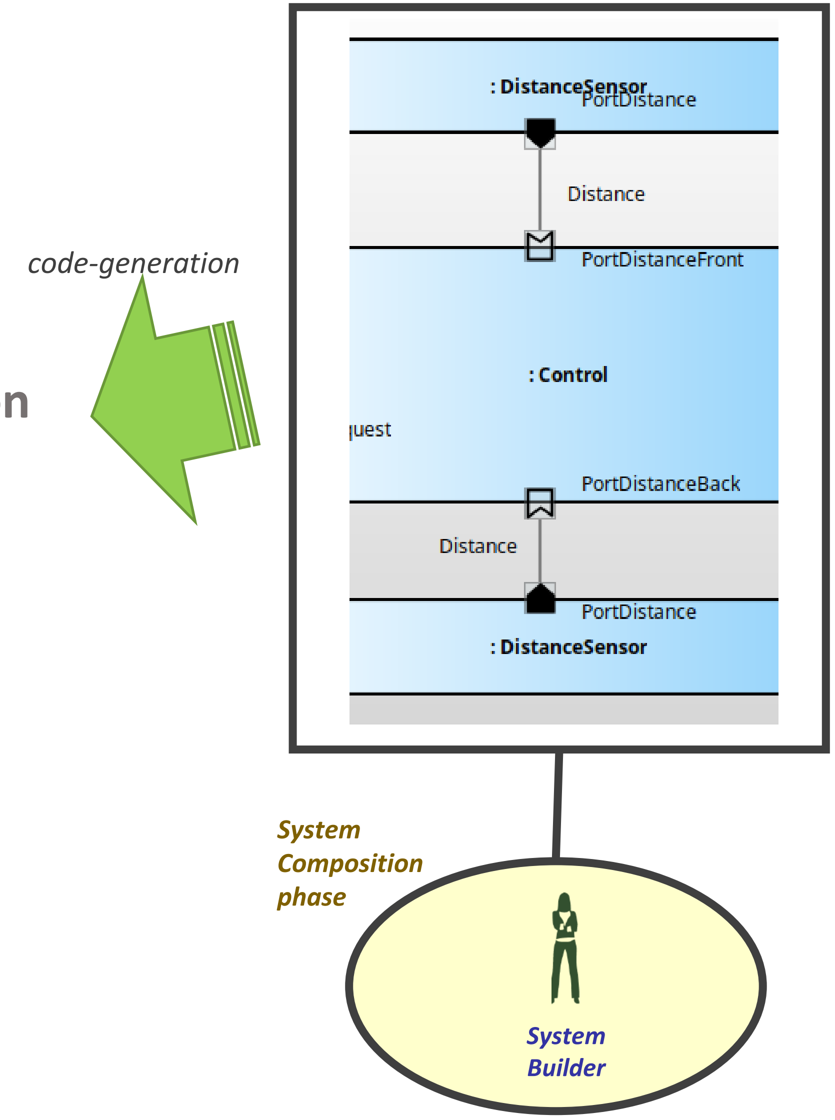


## Code-generation

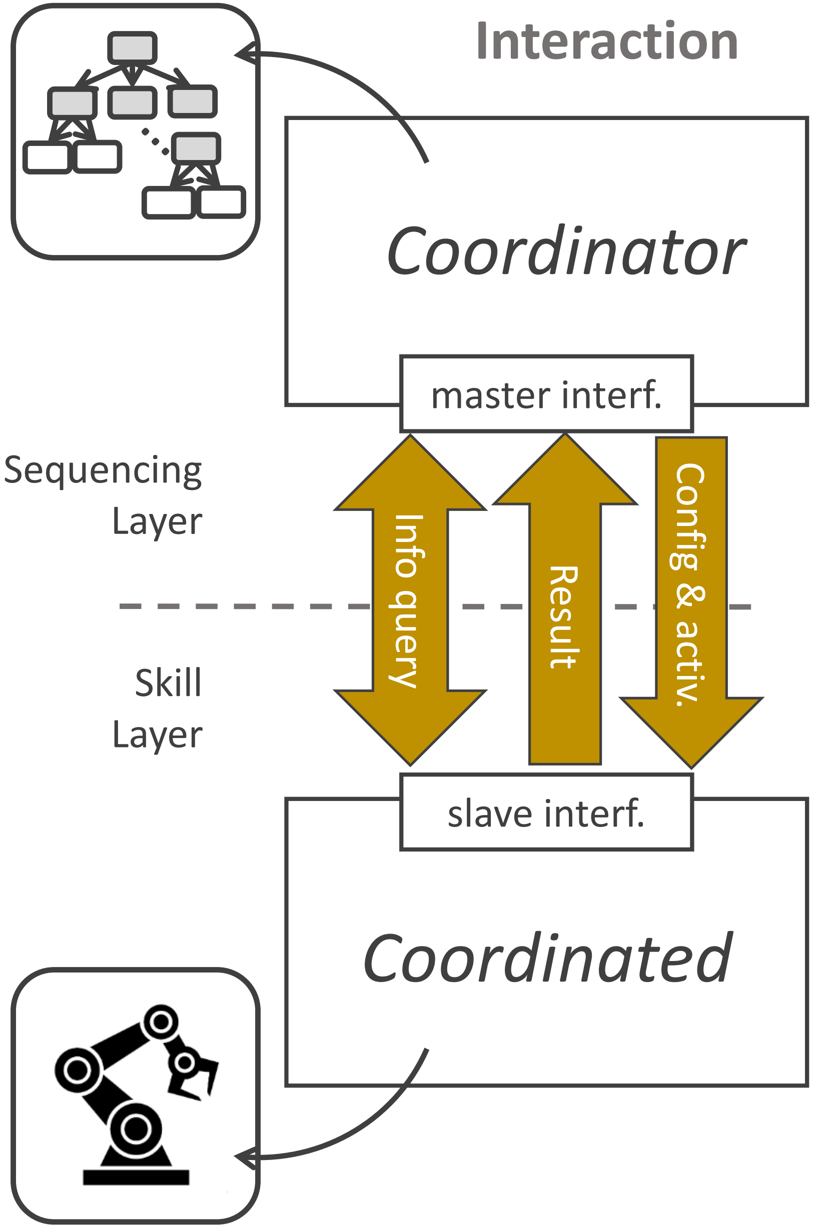
- **package.xml and CMakelists.txt**
  configure Eclipse CDT to use colcon
- **launch script with re-mappings according to components' composition**
  launch scripts are also generated that activate components automatically
- **YAML files for parameter configuration**
  default value overridden per instance

## Eclipse integration

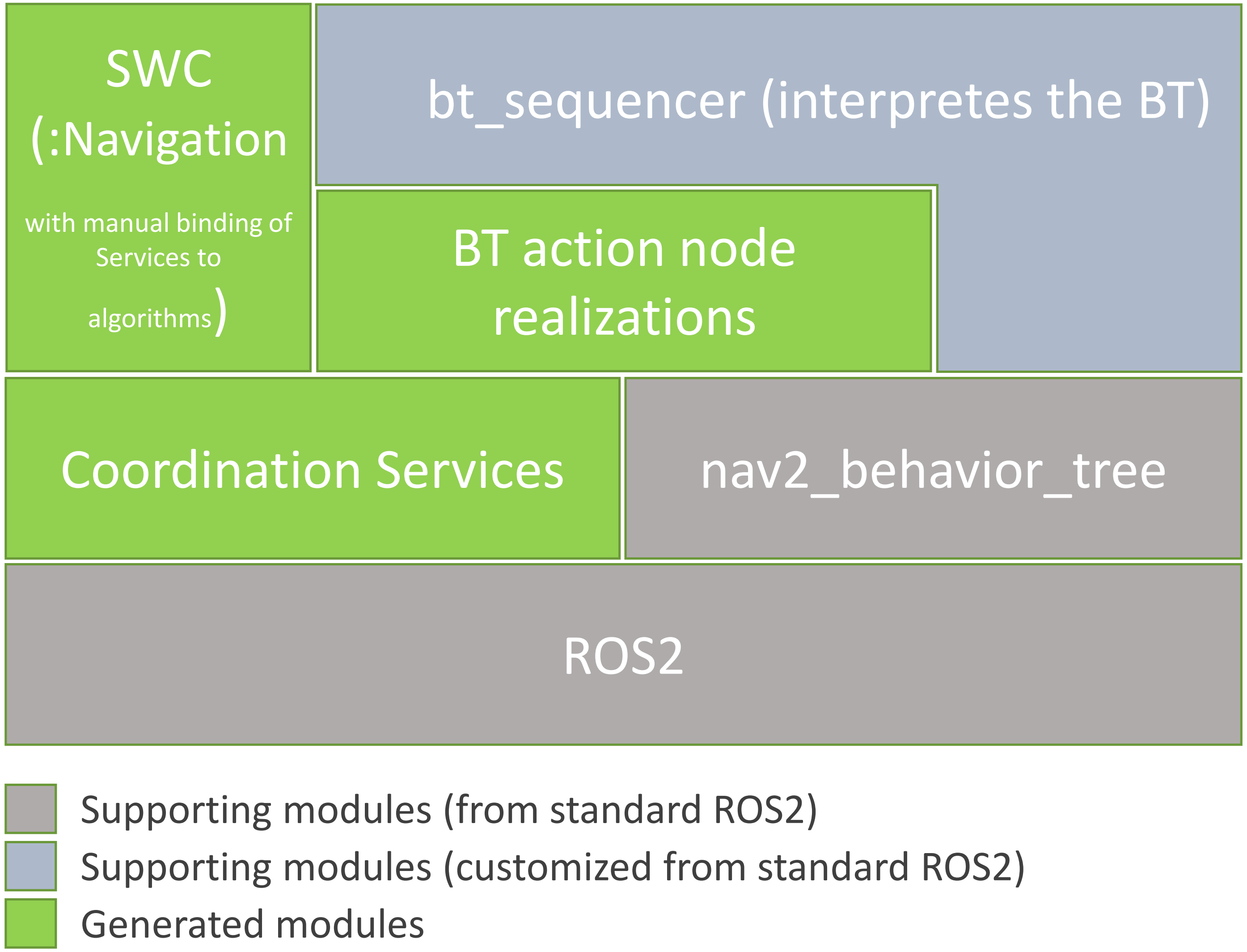
- **Can launch and debug (CDT debugger) a component from Eclipse**
  https://youtu.be/kWkUpKcJq48
- **Use of Eclipse lauch configurations / console**



## Interaction




## Run-time architecture

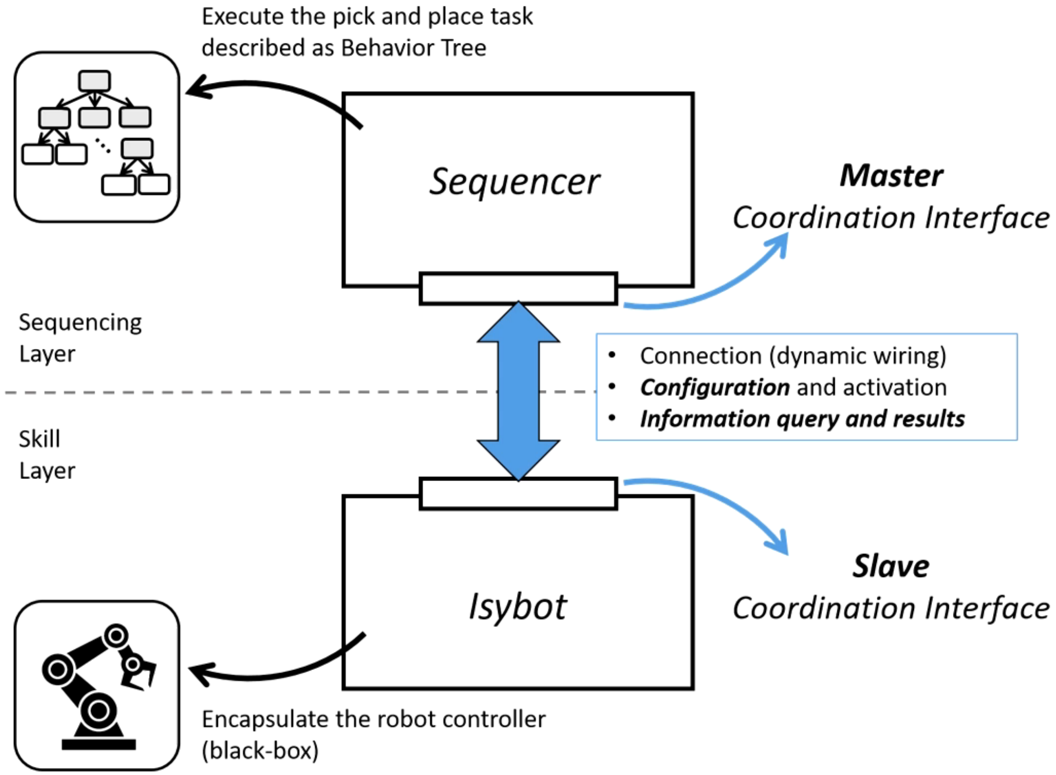


# Examples

- Context: simulation of the collaborative robot arm Isybot performing a pick and place task

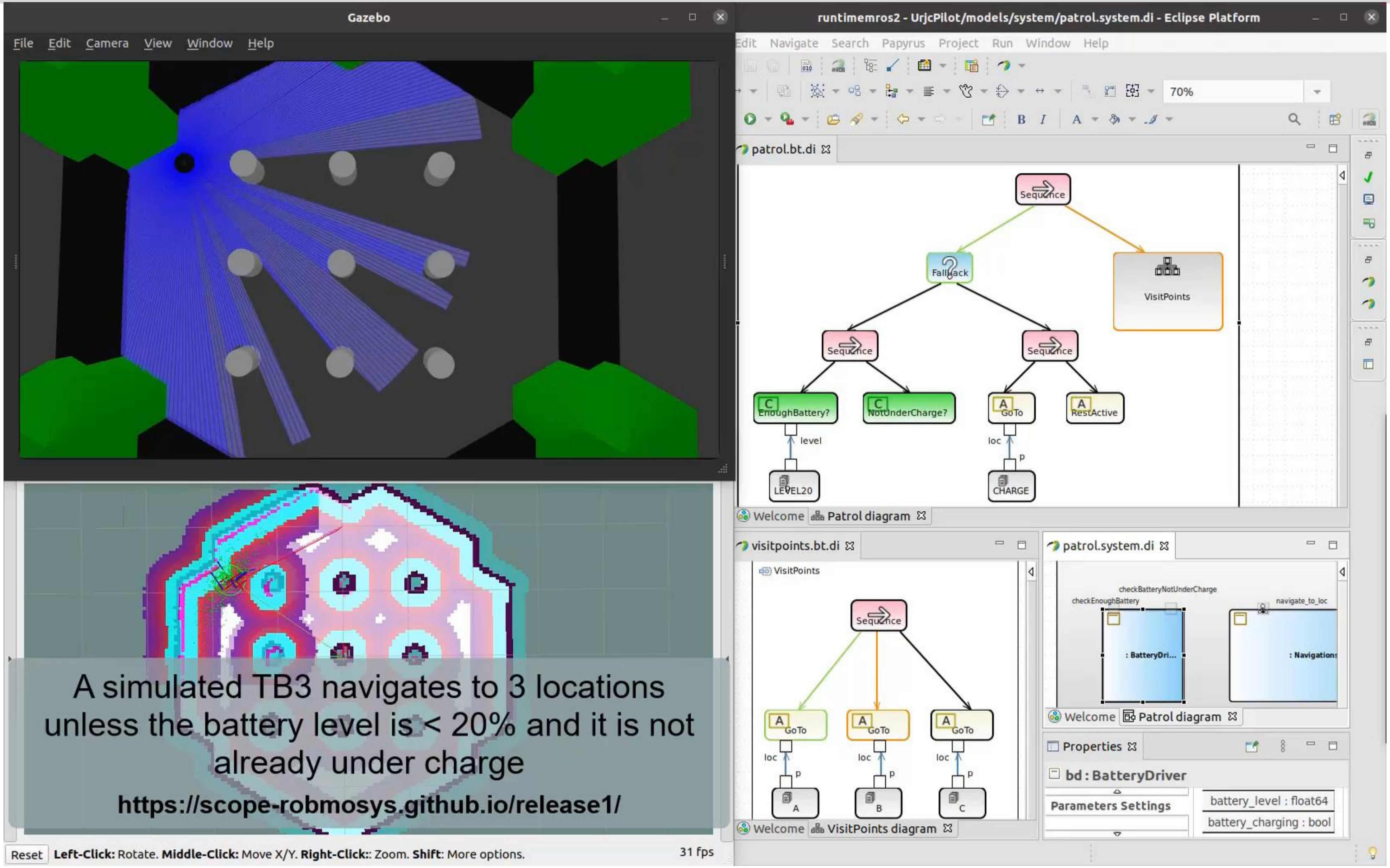
- System composed of 2 components -- the **Sequencer** executes the **behavior tree** specification of pick -and-place task
by opportune **configuration and activation of** the **Isybot component**

- The **coordination interface** conforms to the "Architectural Pattern for Component Coordination" [1] introduced in RobMoSys and is ***generated from models*** *of the behavioral and system specification*




[1] https://robmosys.eu/wiki/general_principles:architectural_patterns:component-coordination

# Examples


Gazebo
runtimemros2 - UrjcPilot/models/system/patrol.system.di - Eclipse Platform
A simulated TB3 navigates to 3 locations unless the battery level is < 20% and it is not already under charge
https://scope-robmosys.github.io/release1/
Sequence
Fallback
VisitPoints
EnoughBattery?
NotUnderCharge?
GoTo
RestActive
LEVEL20
CHARGE
bd : BatteryDriver
battery_level : float64
battery_charging : bool

## Goals

- **Reduce the effort spent in programming**
  component development effort and system integration effort (by generating artefacts from the models).
- **Improve the quality and safety of the obtained system**
  higher consistency due to generation, improved safety via HARA integration

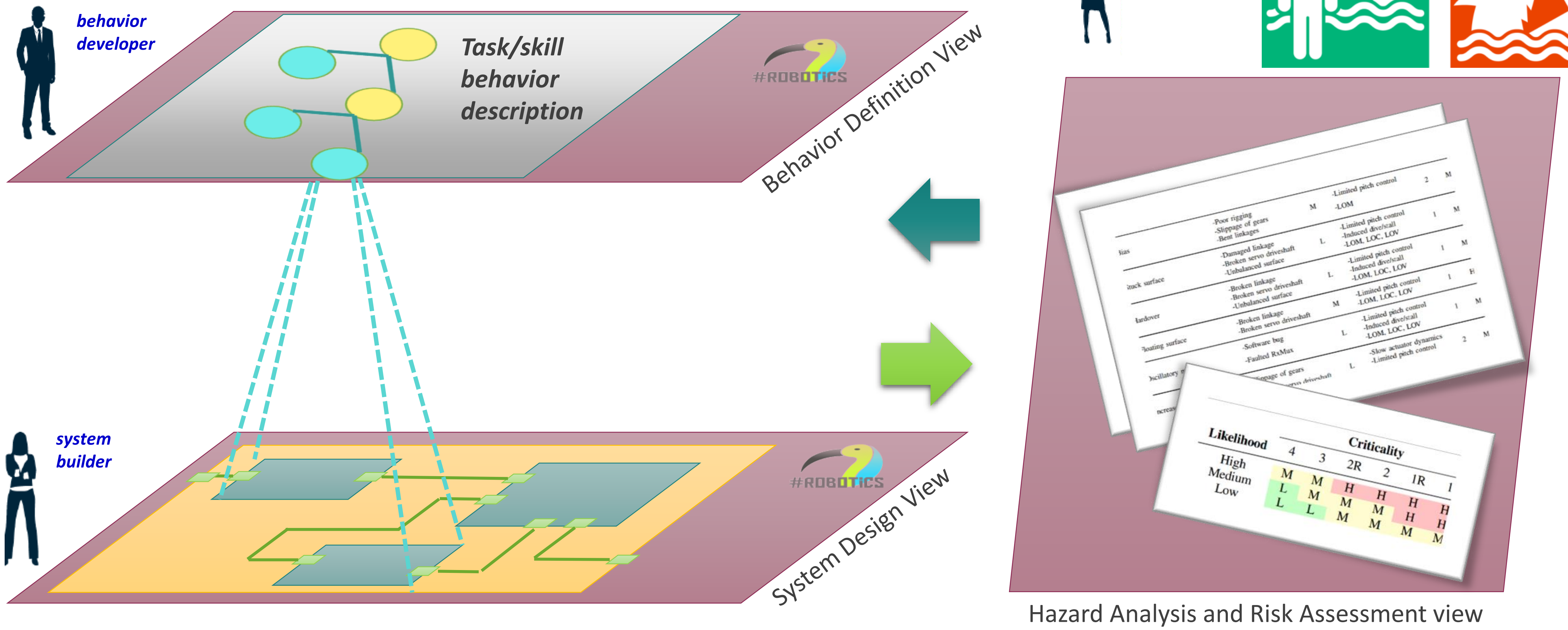
$$M_{\text{CEA-M3}} = \frac{1}{|C|} \sum_{C} \frac{\text{Generated LoC} - \text{LoC due to modeling effort}}{\text{Total LoC}}$$

| | Body files | | Header files | Service files | Build files | |
|---|---|---|---|---|---|---|
| | *Generated* | *Manual* | *Generated* | *Generated* | *Generated* | *Manual* |
| HMI | 166 | 28 | 225 | 10 | 190 | 7 |
| Skill Server | 422 | 192 | 381 | 45 | 199 | 7 |

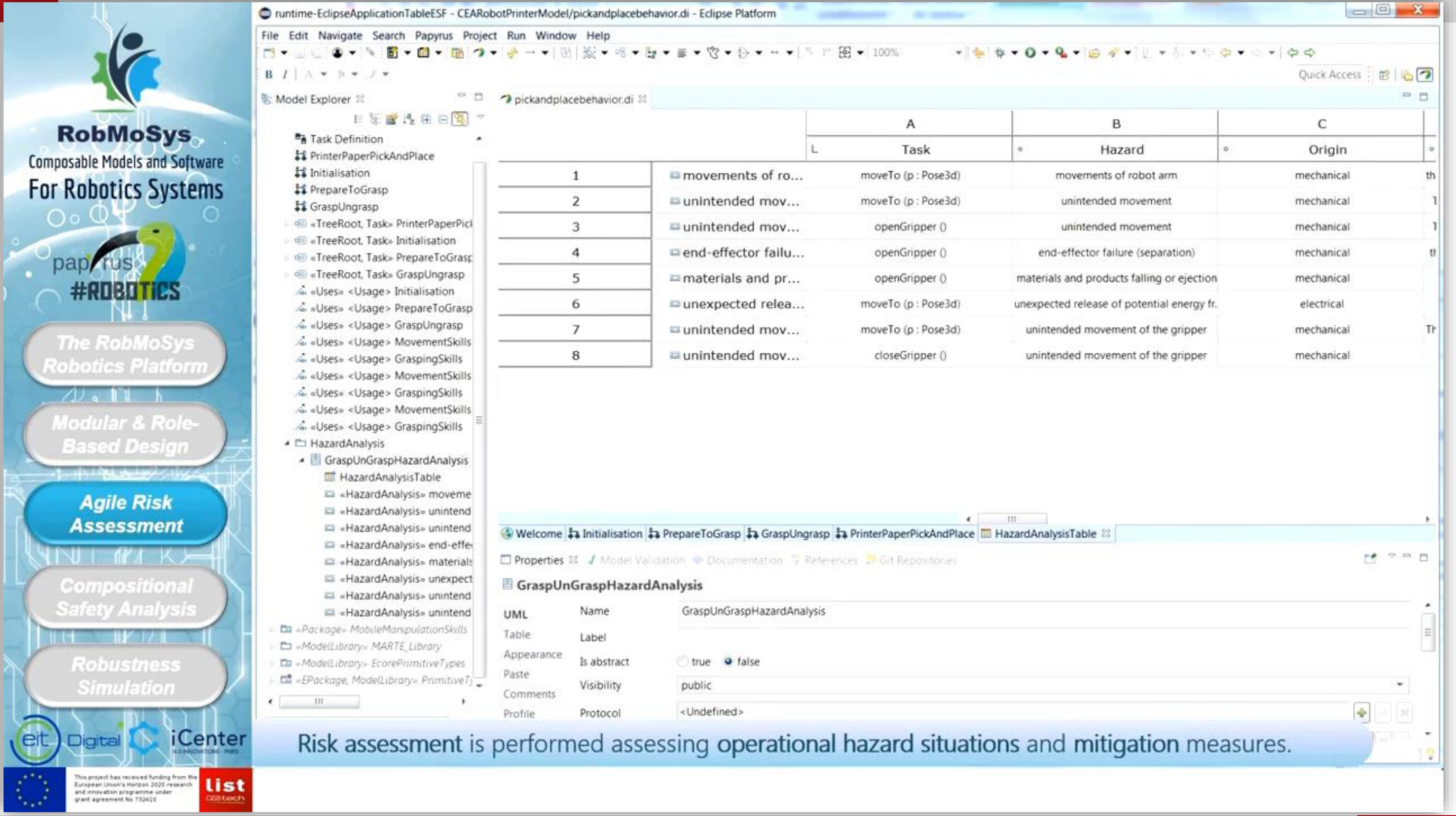
Table 5.2: Generated and manually added LoC for the C++ header/body files, service and build files of components `AdaptiveHMI` and `RobotSkillServer`.

Reduction btw 35% and 67,5%
(depending on the user background and expertise in programming)

# Task-based Hazard Analysis and Risk Assessment

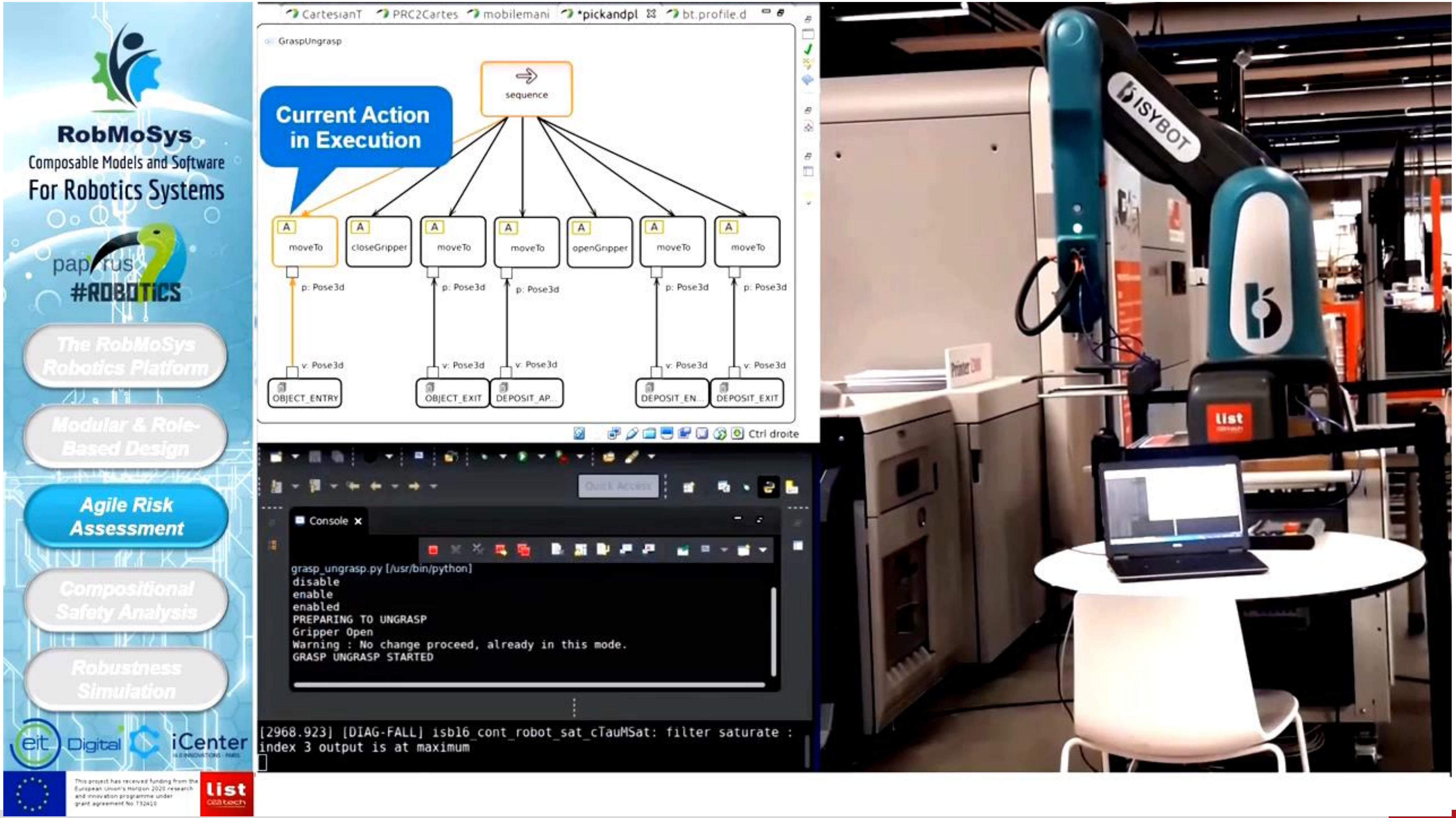




- **Task-Based HARA is performed following ISO 10218-2:2011.**
  For each action in the behavior tree, we list all the relevant hazards and compute their risk index.
  The risk analysis table structure is extracted from ISO/TR 14121-2:2007.

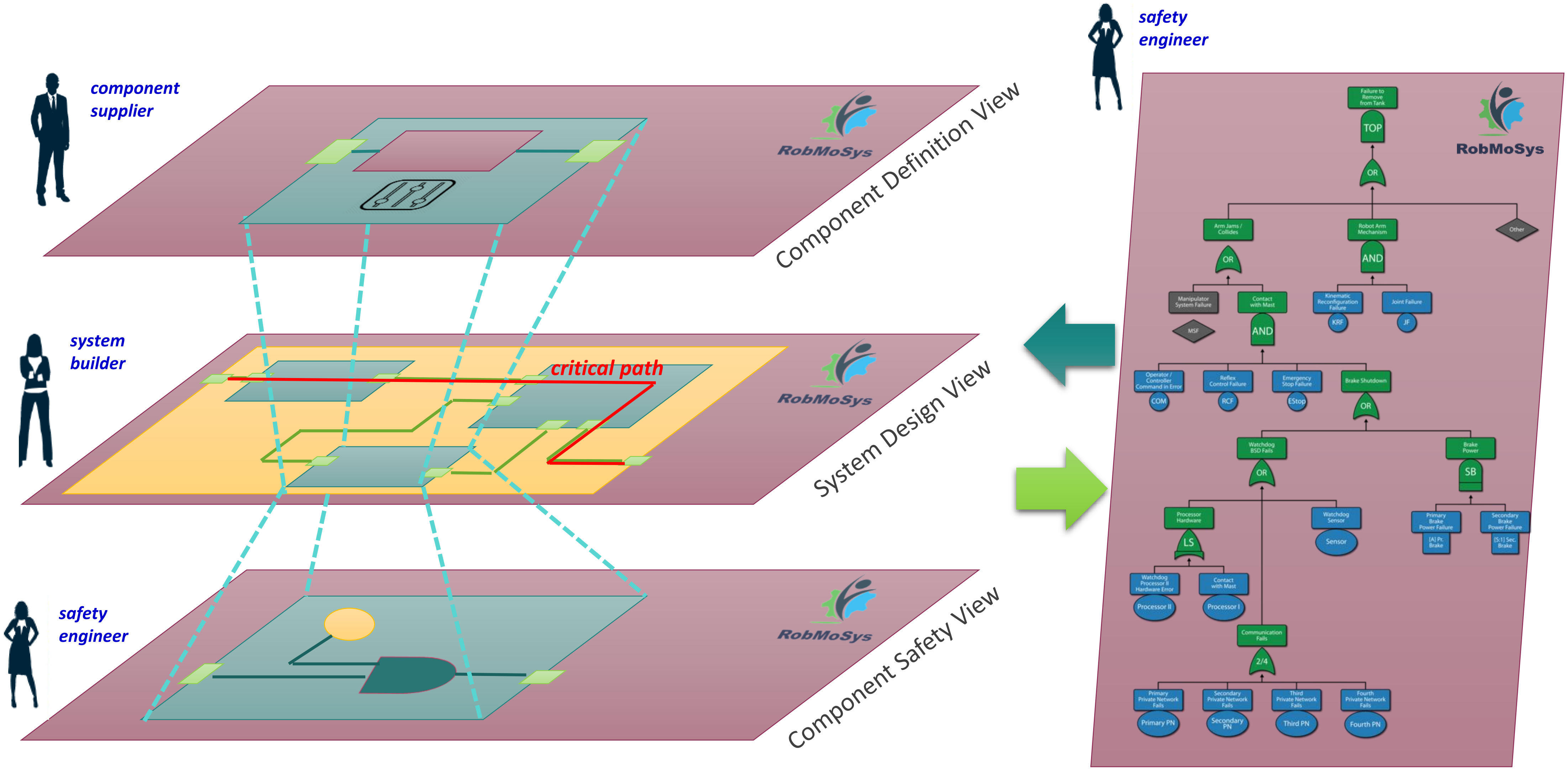



Risk assessment is performed assessing operational hazard situations and mitigation measures.

# Deployment of the final solution

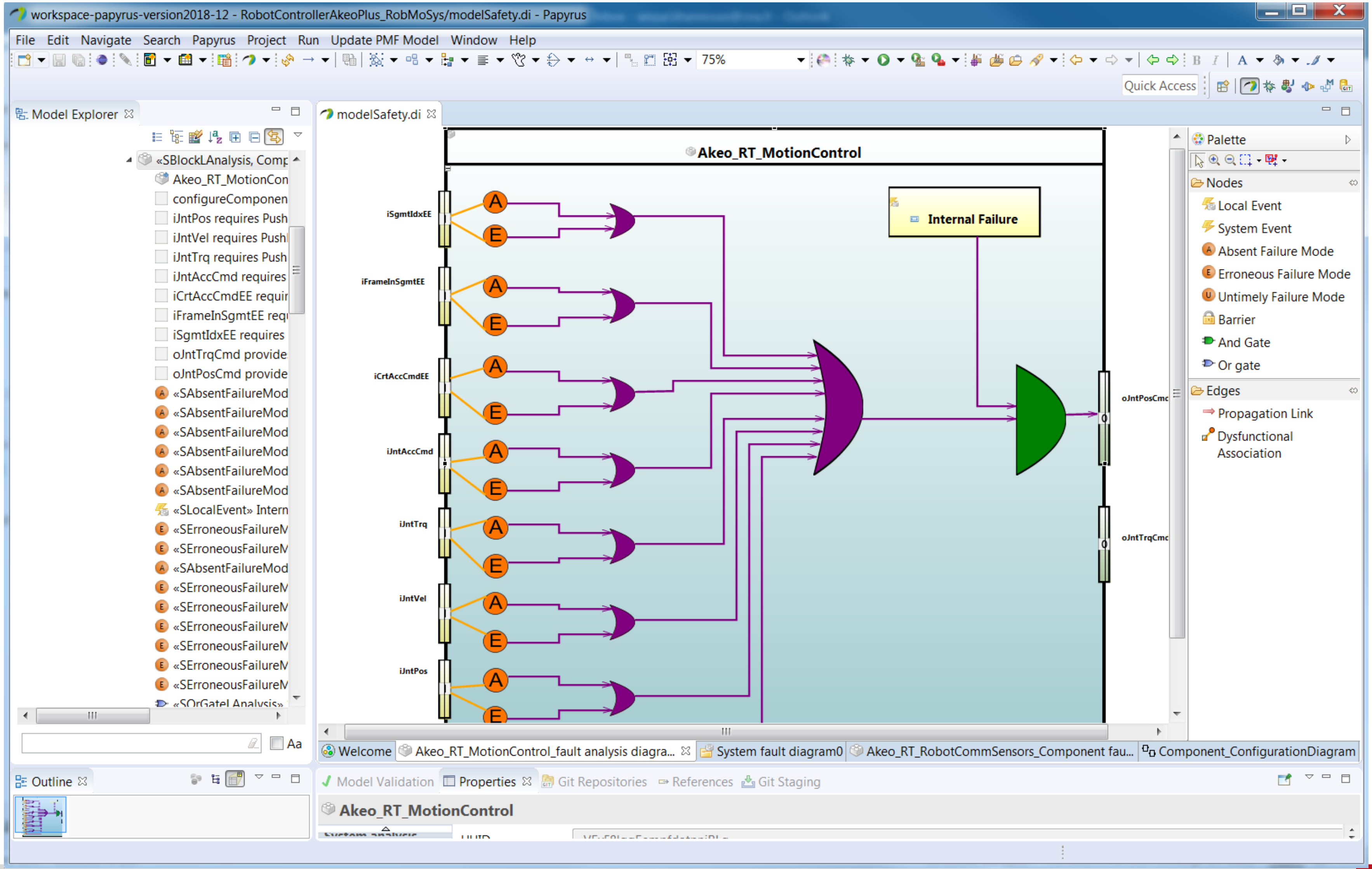


# Compositional Safety Analysis



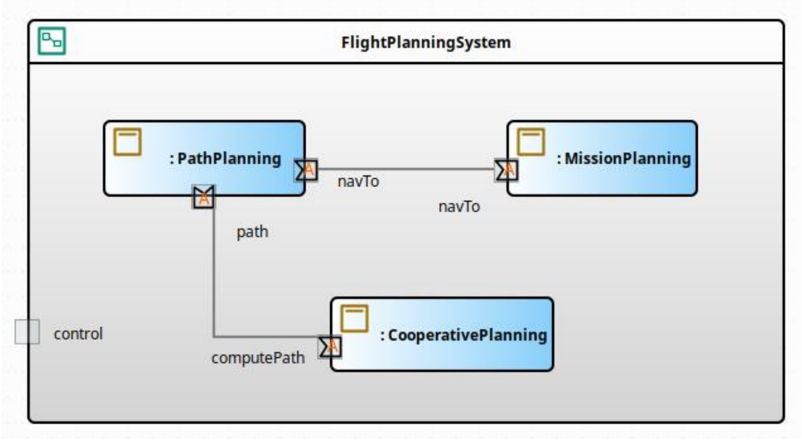

Fault Tree Analysis (FTA) View

# Compositional Safety Analysis (demo)



# Outlook roadmap

- P4R used internally in several (European) projects
  ⇨Continues to be developed actively
- Support for composite components to model whole-part relationship
- Modeling and deployment of ROS2-based automated planning solutions
- Real-time support, model of computation & communication (MoCC)
  - ROS2 and pyCPA support




2 Planning System