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 heterogeneous backgrounds and skills
RobMoSys Composition Structures

- Communication
- System Component Architecture
- Behaviour
- Component Definition
- Deployment

Multiple Stakeholders and Concerns

- Component and Architectural Design
- Validation and Verification
- Code Generation and Deployment

Modelling Views
- Abstractions

Implementation
- Design
- Safety Assessment
- Validation & Verification
- Quality Management
- Implementation
- Component Release

Tools:
- Papyrus4Robotics
- RobMoSys

Website:
https://robmosys.eu/
Ecosystem Tiers and Roles

Tier 1
- Defines the composition-structure, structures the ecosystem
- Example: RobMoSy/a consortium

Tier 2
- Structures the domains within robotics
- Example: the manipulation domain

Tier 3
- These are the users of the ecosystem, it is about providing and using content
- Example: SMEs providing specific solutions as component or building concrete systems

Roles on Composition Tiers:

Activities (in the composition-workflow):
- Definition of Composition-Structures
- Create Service Definitions
- Component Development
- System Integration

Artifacts (models):
- Communication Pattern
- Data Structure
- Service Definition
- Component
- System Configuration (Robot Application)

(List is not complete)
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

CEA realization of RobMoSys approach
https://wiki.eclipse.org/Papyrus/customizations/robotics
Roles and interactions

Reverse
youtu.be/fmAWeplIHzd0

Component
Definition

Skill
Definition

Model task and skills
https://bit.ly/33gCjLw

Tier 2

Tier 3

Service
Definition

Component
Definition

System
Build

Task
Modeling

Service
Designer

Component
Developer

System
Builder

Behavior
Developer

Skills
Designer

- Code-generation
- Deployment
- Risk assessment

Programmer

Safety
Engineer

Create ROS2 Build files
youtu.be/LZxr66gtHgA

Generate component code (AMCL)
youtu.be/M2J9XxWcgEl

Deploy robotic behavior
youtu.be/Mim0cAr1WCs

Link Hazards and Risks with Tasks
youtu.be/fNbgmT0NQYc?t=163

Cohesive CDT integration
youtu.be/P61COtOuUm0
Services and components

Ros2 data types and services (components’ data flows / configuration&coordination)
Services and components

Tier 2

Component Definition

Tier 3

Component Development phase
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

Tier 2

Skill Definition

Tier 3

Task Modeling

Behavior Development phase

Commissariat à l’énergie atomique et aux énergies alternatives
M. MORELLI
09 juillet 2021
Skill code-generation
Generation of build and launch files

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 launch configurations / console
Task execution

**Coordinator**

*Sequencing Layer*

*Skill Layer*

- *Info query*
- *Result*
- *Config & activ.*

*master interf.*

*slave interf.*

**Coordinated**

Run-time architecture

- **SWC**: Navigation
  - with manual binding of Services to algorithms
  - *bt_sequencer* (interpretes the BT)

- **BT action node realizations**

- **Coordination Services**

- **nav2_behavior_tree**

- **ROS2**

| Supporting modules (from standard ROS2) |
| Supporting modules (customized from standard ROS2) |
| Generated modules |
- 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

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/
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

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

<table>
<thead>
<tr>
<th></th>
<th>Body files</th>
<th>Header files</th>
<th>Service files</th>
<th>Build files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Generated</td>
<td>Manual</td>
<td>Generated</td>
<td>Generated</td>
</tr>
<tr>
<td>HMI</td>
<td>166</td>
<td>28</td>
<td>225</td>
<td>10</td>
</tr>
<tr>
<td>Skill Server</td>
<td>422</td>
<td>192</td>
<td>381</td>
<td>45</td>
</tr>
</tbody>
</table>

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 (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