A USE CASE IN MODEL-BASED ROBOT DEVELOPMENT USING AADL AND ROS

Gianluca Bardaro
Andrea Semprebon
Matteo Matteucci
THE ROBOTIC PLATFORM

On board computer

Joystick

Encoders

Lasers
APPLICATION

Core functionalities

Human in the loop
- Joystick
- Teleoperation
- Fully manual
- Assisted
- Autonomous

App oriented interaction
Brain computer interface
High level task coordinator
THE MODEL
LIFE CYCLE

Two steps node initialization
1. Generic node setup
2. User-defined

Asynchronous loop
Check for termination and errors

Two steps node shutdown
1. Generic tear down
2. User-defined

User-extendable error management
CODE GENERATION: EXISTING ROS NODES

- Node model
- Create launch file
- Remap topics
- Assign parameters

Eventually hierarchical

Defined only as an interface

Avoid dead connections

ASN.1 or YAML?
CODE GENERATION: CUSTOM NODES

- Set parameters
- Initialize variables
- Not the best solution

- Parse ASN.1
- Only supported types
- Define ROS elements

- Node model
- Detect threads
- Bind methods

- Collect dependencies
- Create headers

- Create launch files
- Create build files
- Almost ready to build
- Message types from ports
- Specified as properties

- Only supported types
- Specified as properties

- Not the best solution
Call `ratp_lib` API instead of a publisher

`std::function` to bind a callback-equivalent

`ratp` thread skipped by the code generator
ARCHITECTURE COMPARISON
ARCHITECTURE COMPARISON
AT RUNTIME
This work is the answer to a practical problem:

“Simplify the development of our mobile platforms”

“Push re-usability from components to designs”

What we learnt:

“Modeling is (relatively) easy”

“Code generation is (definitely) hard”