The Artemis Rover as an Example for Model Based Engineering in Space Robotics
Jakob Schwender, Thomas M. Roehr, Stefan Haase, Malte Wirkus, Marc Manz, Sascha Arnold and Janosch Machowinski

presented by Thomas M. Roehr

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de
The Spacebot Cup

- Development of an autonomous mobile manipulation system within 8 months

- Task:
  - unknown exploration area (21x21.5m²) containing three target objects
  - find and collect two objects
  - find main object and assemble all

- Constraints:
  - remote operation allowed up to three times up to 5 minutes each
  - communication delay of 2s (one-way)
Development approach

• Top down
  ▪ high-level mission decomposition and identification of required capabilities
  ▪ distribution of tasks to specialized (sub-)teams
  ▪ maximization of component and library reuse

• Main development lines / (sub) teams
  ▪ hardware
  ▪ software
    ▶ navigation
    ▶ manipulation
    ▶ (exploration and) object detection
    ▶ integration
Artemis - Hardware

- Velodyne HDL-32E
- AVT Prosilica
- Xsens IMU MTi 10
- Cup Storage
- W-Lan Modem
- Intel Core i7 Computer
- Battery Storage
- Wheel Module
- Elastic Wheel
- Six DoF Arm
- Two Finger Gripper
Artemis - Software

• Application of Robot Construction Kit (Rock) as basis
  ▪ model-based
  ▪ component-based
  ▪ established workflows and infrastructure for efficiently
    ► embedding external library
    ► performing library and component updates
    ► managing network of components

• Rock allows to interface with ROS components (nodes)
  ▪ managing component networks can deal with Rock and ROS components
Artemis – Model-based

Workflow

- Embed library
- Create functionality (library)
- Code generation
- Component specification

Levels of functionality

- Actions
  - move_to

- Compositions
  - Navigation

- Components (Orocos RTT)
  - TrajectoryFollower
  - MotionController
  - CorridorNavigation
  - Velodyne
  - ...

- Drivers / Libraries (C/C++)
  - trajectory_follower
  - vfh_star
  - corridonavigation
  - base_types
  - boost
  - further operating system dependencies
  - velodyne
  -...

+ ruby glue code
Model-based Components

- Specification uses a domain specific language (DSL)
  - Orocos RTT as component model

```plaintext
name "message_producer"
using_library "message_driver"
import_types_from "message_driver/Message.hpp"

task_context "Task" do
  output_port "messages", "message_driver/Message"
  periodic(1.0)
end
```

- Specification is applicable to other component models, e.g. ROS Nodes

<table>
<thead>
<tr>
<th>Orocos</th>
<th>ROS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Context</td>
<td>Node</td>
</tr>
<tr>
<td>Port</td>
<td>Topic</td>
</tr>
<tr>
<td>Deployment</td>
<td>Launcher</td>
</tr>
</tbody>
</table>
Artemis - Overview

Hardware

Rock / ROS Bridge

ROS Nodes

Communications

Drivers

Rock Components

Hardware

Frame

Electronics

Chassis Sensors

Navigation Sensors

Navigation

Motion Control

Object Matching

Manipulator Arm

Trajectory Control

Manipulation

Object Manipulation

Management/Execution

Operations

Communications

Telemetry
Localization and Mapping

3d point cloud (velodyne)

Graph SLAM
Generalized ICP, g2o

odometry

Multi-Level Surface Maps
Object detection

Far field detection – rough localization

Near field detection – exact localization
Workflow

Optional: Create deployments

Design component networks

Offline validation

Online validation

Manage compositions (aka component networks)

- Spacebot::Compositions::TiltScan
  - Roles: laser_front

- Spacebot::Compositions::ScanFilter
  - Roles: range_finder

- LaserFilter::Task[laser_filter_front]
  - Roles: filter
  - Arguments:
    - conf: [default, laser_front]
    - orocos_name: laser_filter_front

- Hokuyo::Task[hokuyo_front]
  - Roles: range_finder
  - Arguments:
    - conf: [default, hokuyo_front]
    - driver_dev: device
    - Sensors: [Hokuyo, as = hokuyo_front]
    - orocos_name: hokuyo_front

- Spacebot::Compositions::ServoDispatcher
  - Roles: transformer_laser_tilt_base_back, transformer_laser_tilt_base_front, laser_tilt_front

- SpacebotServoDispatcher::Task[dynamixel_dispatcher]
  - Roles: dispatcher
  - Arguments:
    - conf: [default, dynamixel_base]
    - orocos_name: dynamixel_dispatcher

- ServoDynaimxel::Task[dynamixel_base]
  - Roles: servo
  - Arguments:
    - conf: [default, dynamixel_base]
    - orocos_name: dynamixel_base
    - DevActuators: Dynamixel
Summary

• Artemis served to validate the current state of our model-based development approach
  - It showed to us that we made a good step towards a ‘less painful’ integration process for robotics

• Our robotic systems will become more complex

• Managing complexity will be our main challenge

• Rock is open-source:
  http://rock-robotics.org
Video footage

• Artemis in motion
Applications are welcome! We’re currently looking for motivated researchers

www.dfki.de/robotics
Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de