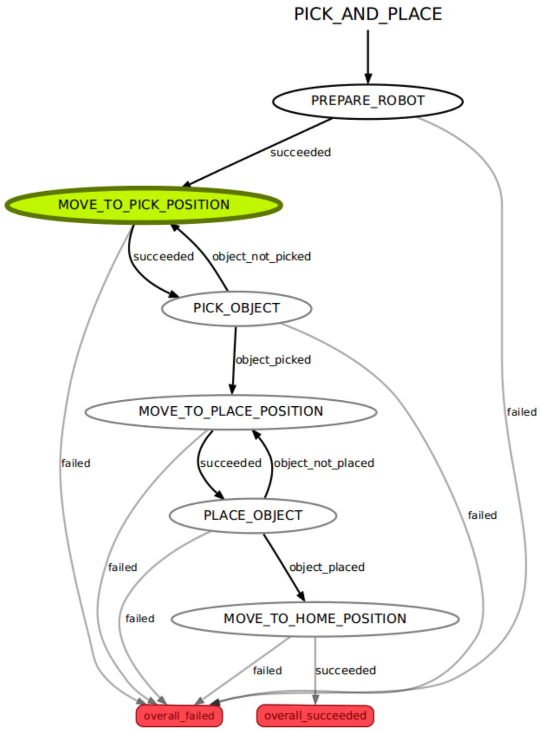
Application Development with ROS

Dipl.-Ing. Florian Weißhardt

Technology Seminar – ROS in Industrial Applications

ROS.org



Fraunhofer IPA

Application Development with ROS
Goals

- Learn about how to develop an application in ROS
  - System integration
  - Application setup
  - Task specific configuration
  - Application execution monitoring
- Learn about the hardware independence in ROS
  - Hardware independent application development
  - Developing an application in simulation
  - Transfer application from simulation to real hardware
  - Running same application on different hardware setups

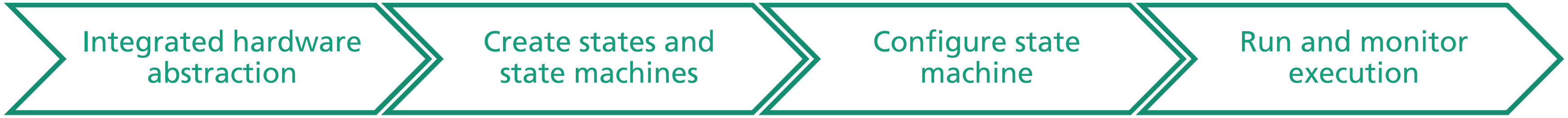
Application Development with ROS
A pick and place application

- Task: Pick an object from a pre-defined source location and place it on a pre-defined target location
- For creating such an application we need:
  - An manipulator with attached gripper (real hardware or equivalent in simulation)
  - Software interfaces to move the manipulator and open and close the gripper
  - Configuration to specify source and target locations
  - Coordinator component defining task execution

Application Development with ROS

A hardware independent pick and place application with ROS components

- ROS provides an hardware abstraction level through standardized ROS APIs (topics, services and actions)
- ROS provides a state-machine based task-level architecture for creating complex robot applications called SMACH
  - StateMachine, concurrent, sequence and iterator containers
  - Wrapper container for any ROS action
  - Configuration of states through ROS parameters
  - Runtime execution monitoring
  - Implemented in Python
- Process for using SMACH for task coordination:



Integrated hardware abstraction

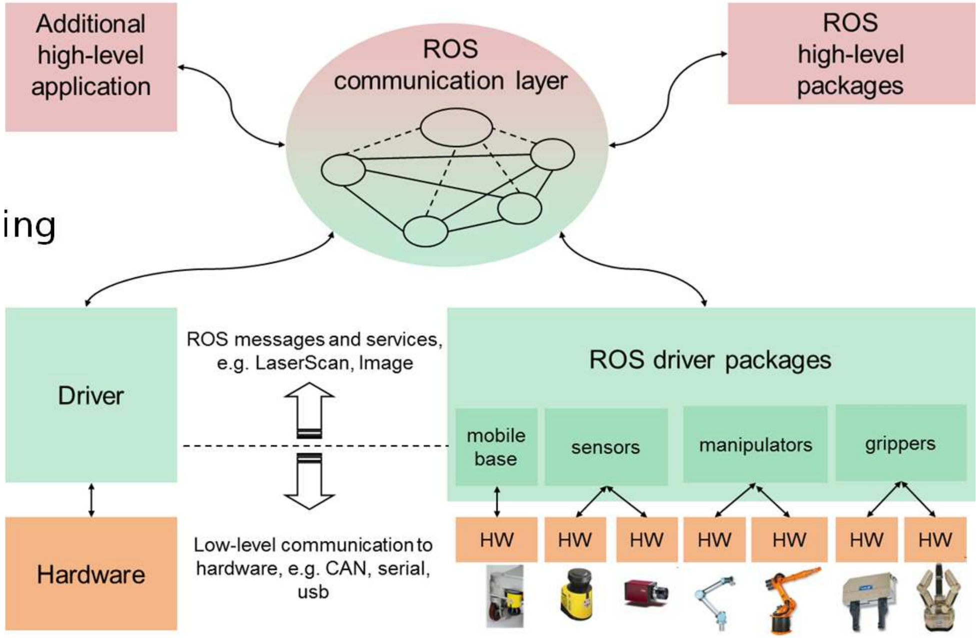
Create states and state machines

Configure state machine

Run and monitor execution

Application Development with ROS
Hardware abstraction

- Creating a system launch file including
  - All drivers (manipulator, gripper)
  - All high-level components (MoveIt!)
  - Offers standardized ROS interfaces (Topics, Services, Actions)
- Creating a application launch file including
  - Uploading parameters
  - Starting up application

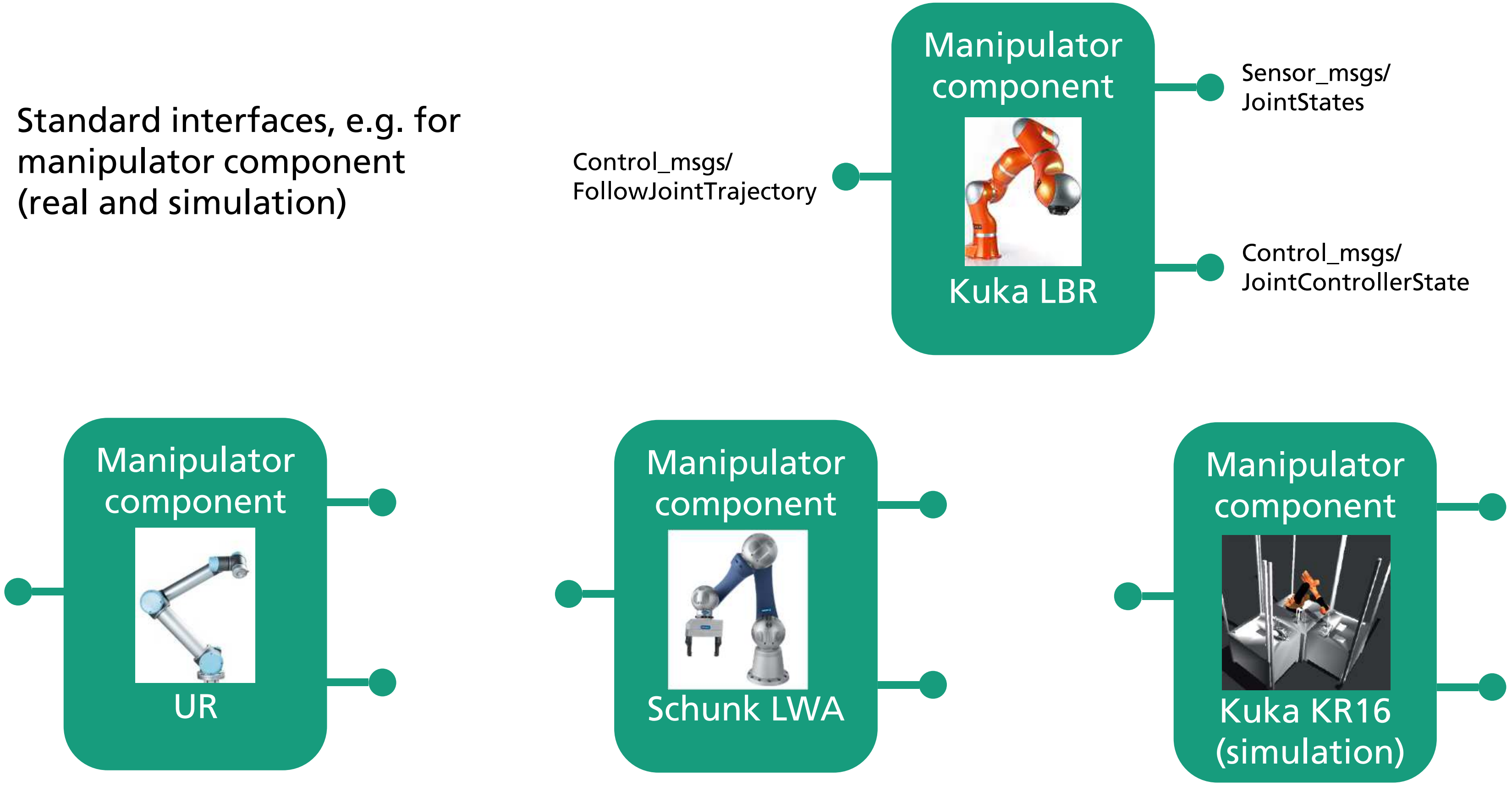

Additional high-level application
ROS communication layer
ROS high-level packages
Driver
ROS messages and services, e.g. LaserScan, Image
ROS driver packages
mobile base
sensors
manipulators
grippers
Hardware
Low-level communication to hardware, e.g. CAN, serial, usb
HW
HW
HW
HW
HW
HW
HW

Integrated hardware abstraction → Create states and state machines → Configure state machine → Run and monitor execution

Application Development with ROS

Hardware abstraction

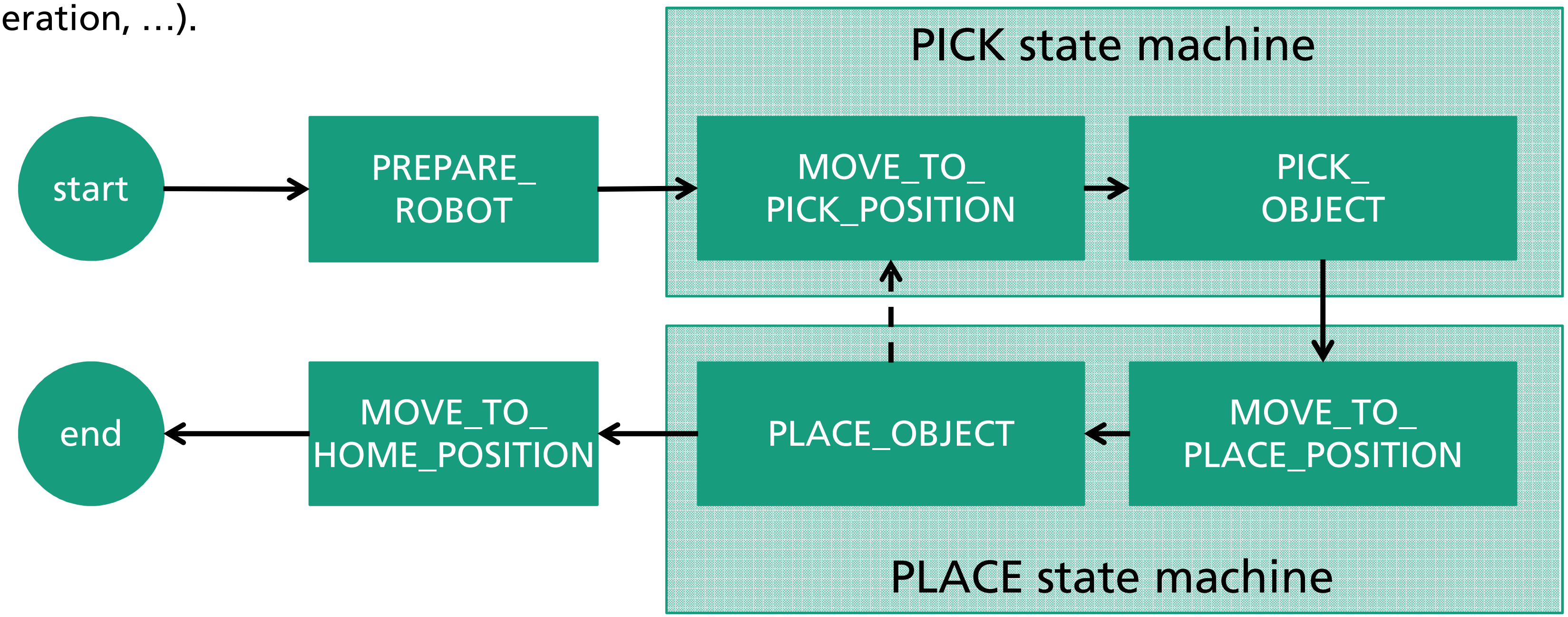
- Standard interfaces, e.g. for manipulator component (real and simulation)

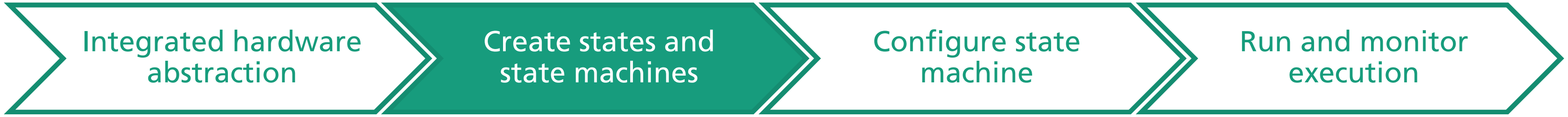


Integrated hardware abstraction → Create states and state machines → Configure state machine → Run and monitor execution

Application Development with ROS
Creating states and state machines

- Each state has transitions to link to following states .
- States are executed in containers. Containers define the execution behavior (e.g. concurrency, iteration, …).

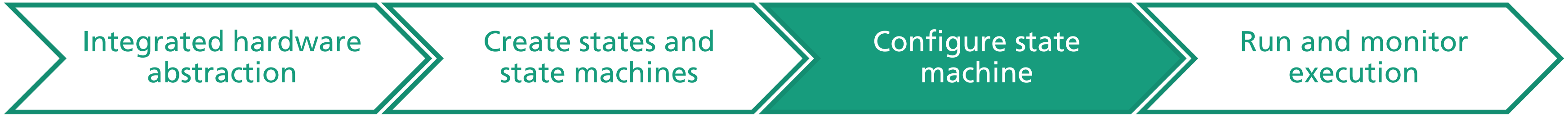




Integrated hardware abstraction
Create states and state machines
Configure state machine
Run and monitor execution

Application Development with ROS
Configuration

- Configuration needed for
  - Source location (Pose6D)
  - Target location (Pose6D)
- Configuration can be done through ROS parameter server
  - Uploading parameters in launch file
  - Uploading parameters through yaml file


Integrated hardware abstraction

Create states and state machines

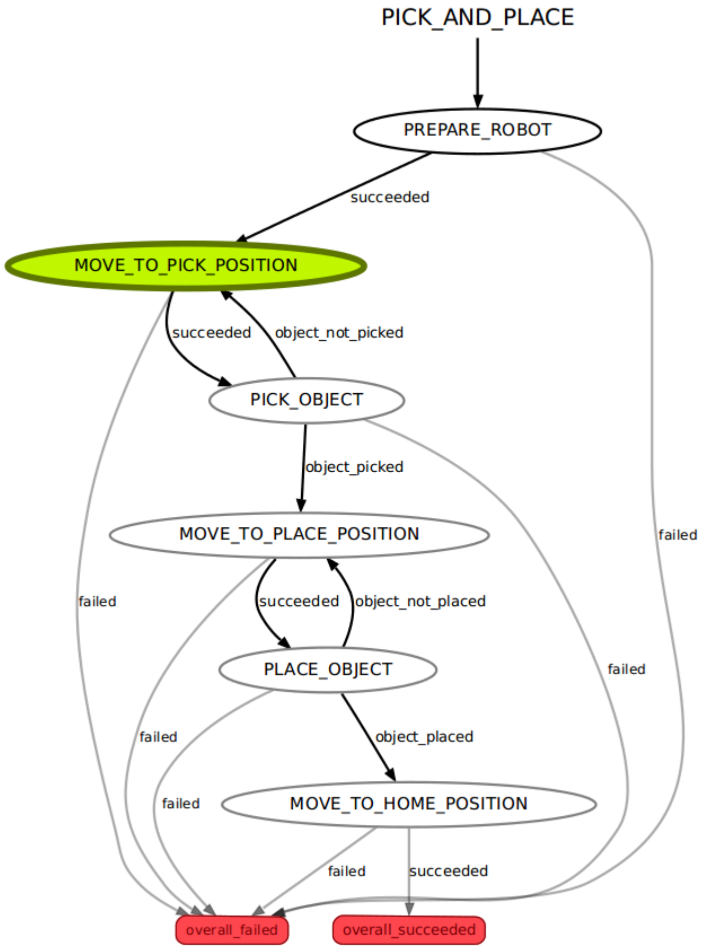
Configure state machine

Run and monitor execution

Application Development with ROS

Execution monitoring

- Running the application by starting the Python script for the state machine
- SMACH offers a graphical tool for
  - Visualizing states and transitions
  - Monitor current state of execution



Integrated hardware abstraction → Create states and state machines → Configure state machine → Run and monitor execution

Application Development with ROS
Summary

- ROS includes state-machine based task-level programming
  - SMACH: http://wiki.ros.org/smach
  - SMACH_viewer: http://wiki.ros.org/smach_viewer
- Separation of hardware driver layer, capability layer and application layer with hardware abstraction through standardized ROS interfaces
- More detailed documentation about standard interfaces can be found at
  - Sensor_msgs: http://wiki.ros.org/sensor_msgs
  - Geometry_msgs: http://wiki.ros.org/geometry_msgs
  - Trajectory_msgs: http://wiki.ros.org/trajectory_msgs
  - Control_msgs: http://wiki.ros.org/control_msgs

Application Development with ROS

Your ROS application expert

Dipl.–Ing. Florian Weißhardt

E-Mail: florian.weisshardt@ipa.fraunhofer.de

Phone: +49 711 970-1046