A Model-Based Specification and Analysis Architecture for Real-Time Robotics Systems

D. L. Wigand*1, P. Mohammadi2, E. M. Hoffman3, J. Steil2, N. Tsagarakis3, and S. Wrede1

1 Bielefeld University, Germany, 2 TU Braunschweig, 3 Istituto Italiano di Tecnologia, Italy


Enrico Mingo, Istituto Italiano di Tecnologia (IIT), 2018
Complex scenarios, great amount of components
Torque control loops, require Real-Time (1 kHz)
Transparent deployment (sim/real)
Support for heterogenous robots
Formalization
CoSiMA's Execution Architecture in a Nutshell

CoSiMA build on top of 3 well-established technologies:

- JetBrains’ MPS as modeling environment
- Orocos RTT as real-time execution environment
- Gazebo as physics simulation environment
CoSiMA’s Execution Architecture in a Nutshell

Non RT Environment

Real-Time Environment
CoSiMA‘s Execution Architecture in a Nutshell

- Generic robot simulation interface
  fully configurable through URDF/SRDF
CoSiMA’s Execution Architecture in a Nutshell

- Generic robot simulation interface fully configurable through URDF/SRDF
- Robot Simulation (Gazebo) runs within the Orocos environment [https://github.com/corlab/rtt-gazebo-embedded](https://github.com/corlab/rtt-gazebo-embedded) forked from [kuka-isir/rtt_gazebo_embedded](https://github.com/kuka-isir/rtt_gazebo_embedded)
CoSiMA’s Execution Architecture in a Nutshell

- **Generic robot simulation interface** fully configurable through URDF/SRDF
- **Robot Simulation (Gazebo)** runs within the Orocos environment

Clock synchronization between Gazebo and Orocos

https://github.com/corlab/rtt-gazebo-clock-plugin
forked from ahoarau/rtt_gazebo/tree/master/rtt_gazebo_system
CoSiMA’s Execution Architecture in a Nutshell

- Generic robot simulation interface fully configurable through URDF/SRDF
- Robot Simulation (Gazebo) runs within the Orocos environment
- Clock synchronization between Gazebo and Orocos
- RT-safe communication between RT and Non-RT environment
Why is Timing Specification and Introspection important?

Zero Moment Point-based walk on a straight line.

Enrico Mingo, IIT, COGIMON Robot, https://youtu.be/n2lGsjskf3A
Debugging faulty (timing) behavior is not a trivial task

- Multiple components
- Parallel execution
- Different frequencies and priorities
- Hard to debug
- Trial and error
- Mostly no full specification
CoSiMA’s Sample-Based Timing Introspection

- Collection of Execution Traces for analysis of the dynamic behavior
- No violation of the real-time constraints
- Introduce as little overhead to computation as possible
- Implementation with extended RTT::TaskContext class; minimal effort for the component developer
CoSiMA’s Timing Specification

Currently, the specification is done using Precedence Task Graphs combined with execution time information on the component level (such as WCET). As well as sense-react chains, which e.g., define the shortest control which keeps the robots behavior stable and safe.
LIVE DEMO
CoSiMA’s Timing Analysis Tool

- Faulty misalignment of “com” component detected. “ik” is operating on old data, since “com” is sending its output to the next cycle.

- “com”, “base”, and “ik” are properly aligned. “ik” receives all the necessary data from the current cycle.
Conclusion

- Timing is important for diverse robotics applications
  - Bipedal walking, force-based interaction, etc.

- Explicit modeling of timing behavior is often neglected
  - Hidden in components
  - Not completely known

- CoSiMA is a step towards model-based design, simulation and analysis of real-time robot systems

- It’s open source! [http://cogimon.github.io](http://cogimon.github.io)
Outlook

- Modeling the timing specification in MPS
- Integrating the Timing Analyzer as MPS-plugin
- Generating verified scheduling and deployment configurations
Thank you!

QUESTIONS?

Dennis Leroy Wigand
dwigand@techfak.uni-bielefeld.de

Visit http://cogimon.github.io for CoSiMA
Visit http://cogimon.eu for project details

Special thanks to all partners that contributed to this work!