The Challenge of Real-Time Robotics Behavior: An Applied Research Perspective

Christian Schlegel

Computer Science Department
University of Applied Sciences Ulm

http://www.hs-ulm.de/schlegel

Technik
Informatik & Medien
Hochschule Ulm
University of Applied Sciences

Introduction and Motivation

- engineering the software development process in robotics is one of the basic necessities towards developing the mass market of service robotic systems

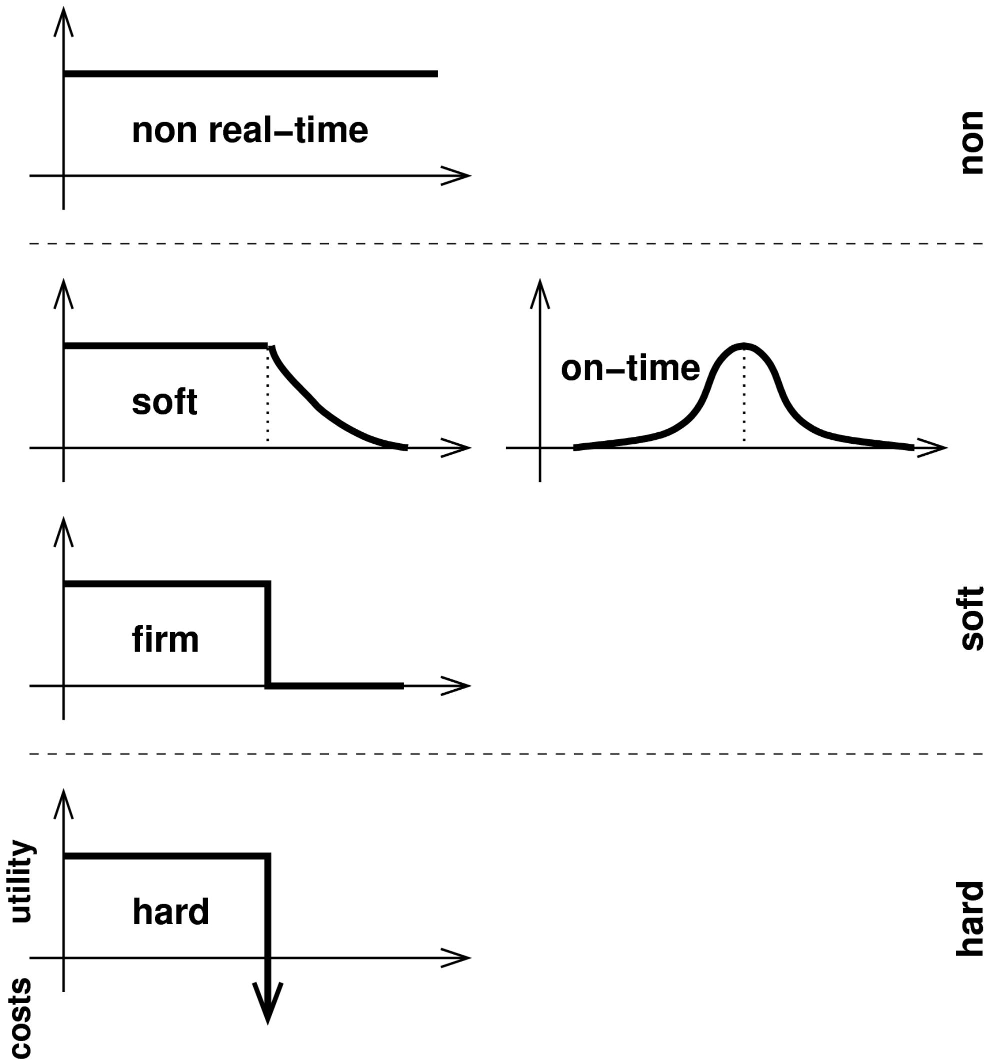
- the obvious gap between current systems and the required level of robustness and reliability can only be closed by
  - ***not any longer ignoring real-time aspects (or in more general: introducing resource-awareness)***
  - ***making the step from code-driven systems to model-driven systems***

=> overcome hand-crafted single-unit systems

=> models are the basis for system verification and validation tools

=> ***time*** affects everything in a system design and can NOT just be added afterwards

Hochschule Ulm



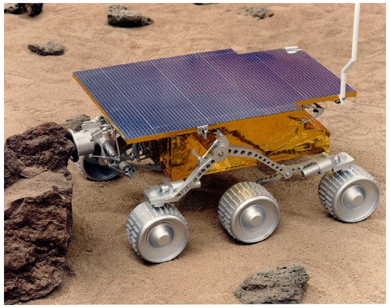
What is a real-time system ?

- real-time systems are defined as those systems in which the correctness of the system not only depends on the logical correctness of the computation but also on the time at which the result is produced.

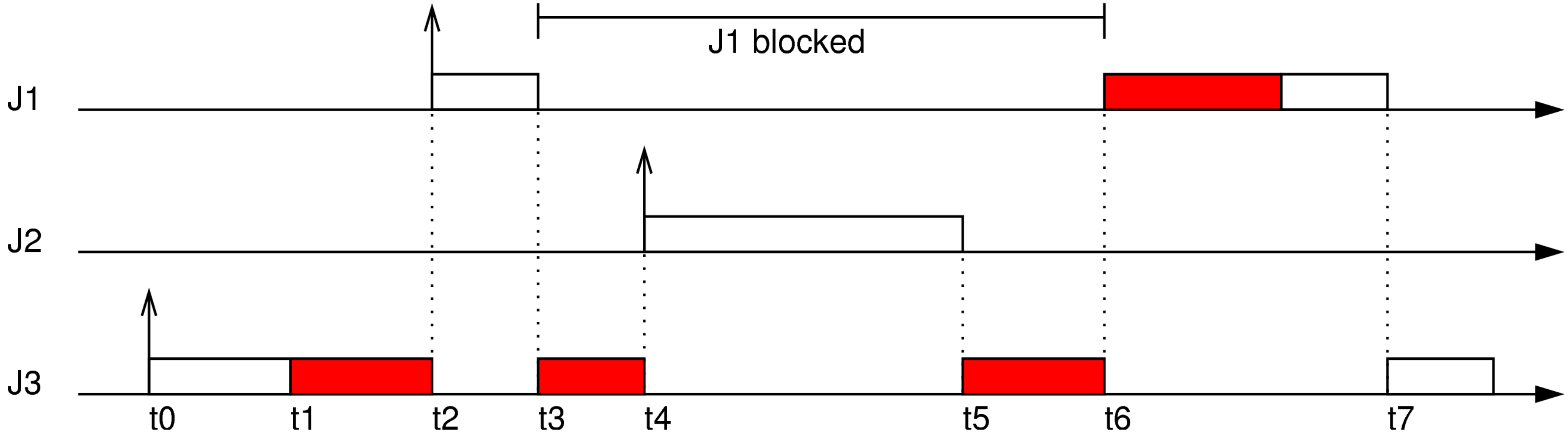
Basic Scheduling Principles and Effects

- Rate Monotonic Scheduling
  - Schedulability Tests
- Priority Inversion
  - Priority Inheritance
  - Priority Ceiling
- Hybrid Scheduling
  - Polling Server
- ...

Algorithm (Liu and Layland, 1973):
Assign the priority of each task according to its period, so that the shorter the period the higher the priority.

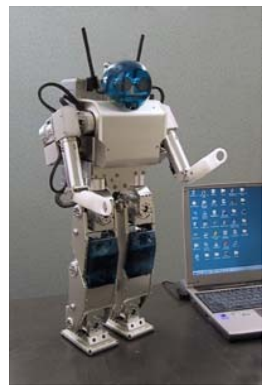


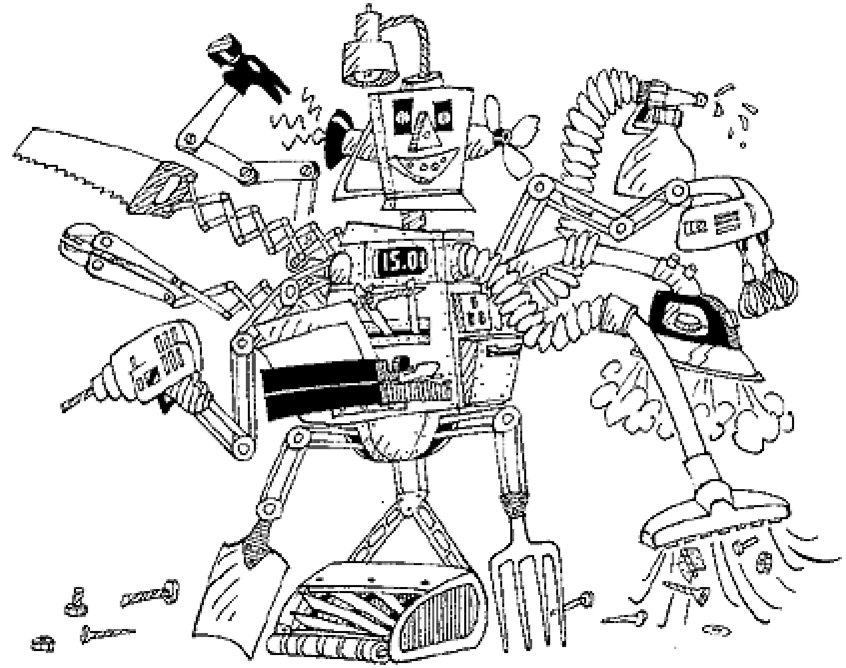
- ***priority driven***
- ***clock driven***



The Robotics Domain

- Manipulators / humanoids / ...
  - control applications
  - compliant motion
  - ...
- Wheeled platforms
  - obstacle avoidance
  - ...
- Flying platforms
  - airborn: fail-safe state ?





- mobile base with manipulator ?
- dynamic environment ?
- ...

Hochschule Ulm

How we so far addressed real-time ?

Pose Labels

- pose labels serve as global frame of reference
- add pose labels to all data objects
- add coordinate system reference counter
- pose labels are available system-wide and are updated periodically
- allows delayed integration of sensor data

=> soft real-time (be careful: real time data is aging !)

Principle of Locality

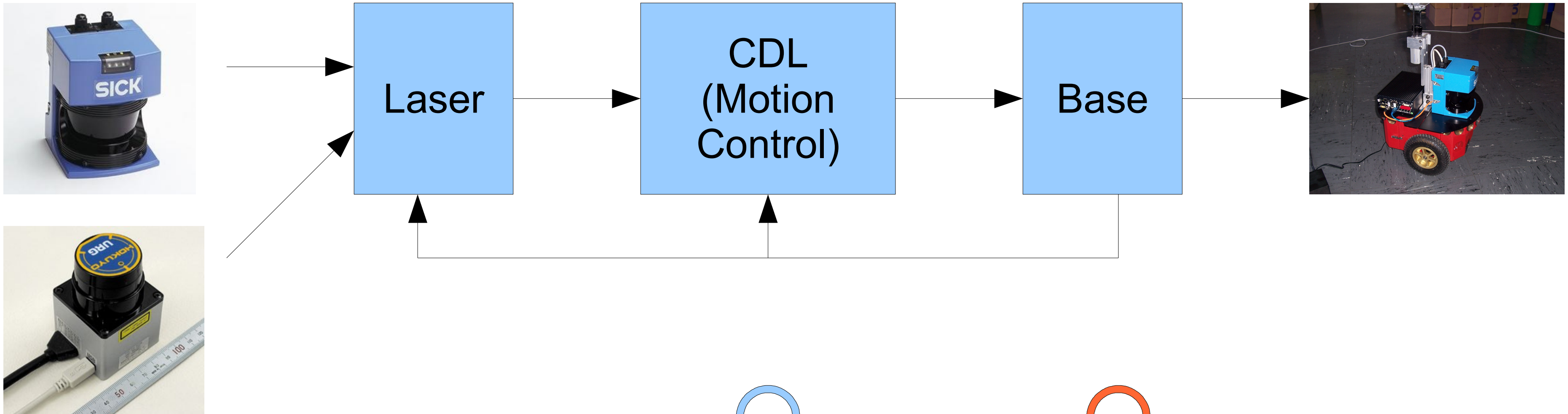
- to achieve decoupling each service takes care of itself
- a service requestor has to monitor whether it gets a response in time
- a service provider has to monitor whether it rejects further requests

Publish/Subscribe

- state-based view maps easily onto hard real-time communication systems
- load / update rate can be verified as soon as deployment is known

=> request / response interaction ?

How we so far addressed real-time ?



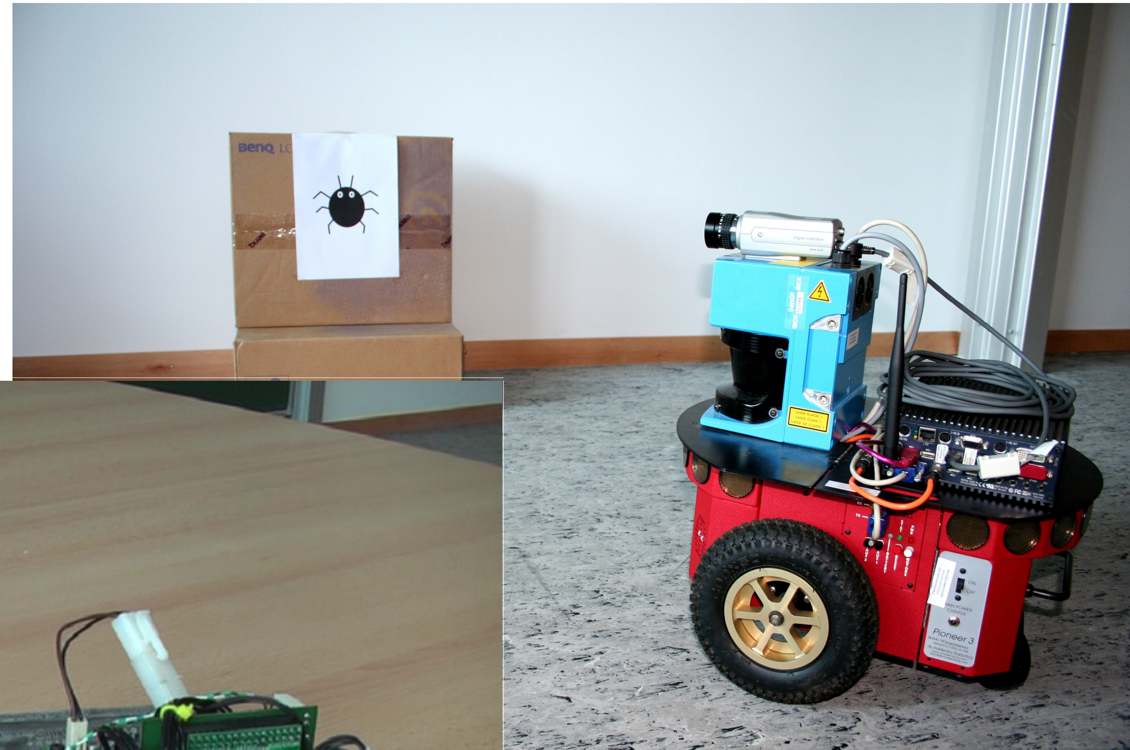
- watchdog checks whether robot state and laser range scan is in time
- reduce speed in case of delayed updates
- works well

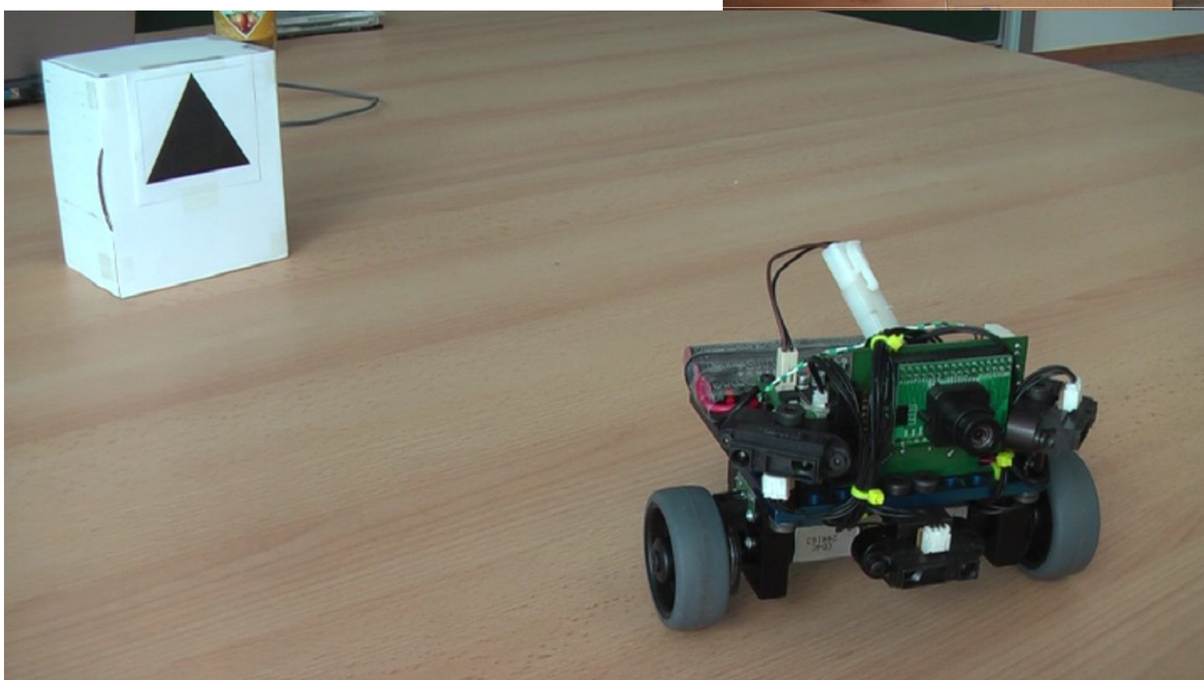
- watchdog checks whether v/w is in time
- reduce speed in case of delayed updates
- works bad

Hochschule Ulm

Model-Driven Software Design

- search for pattern
- drive towards pattern
- stop at pattern

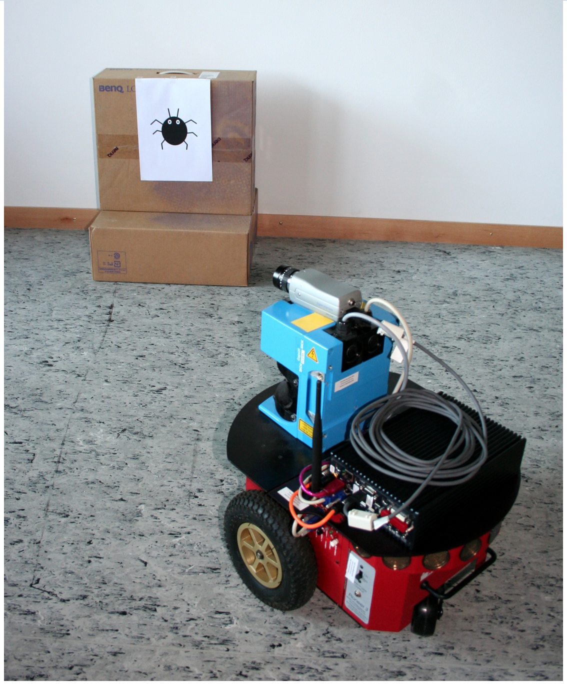


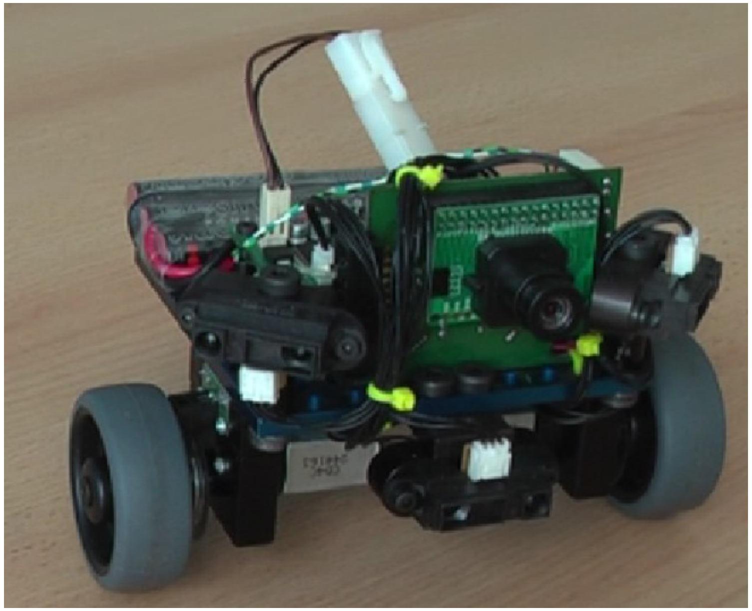


Model-Driven Software Design

QFIX

- ATMEL Atmega128 / 16 Mhz
- 4K RAM / 128K flash
- freeRTOS
- tinySMARTSOFT (based on single process space)



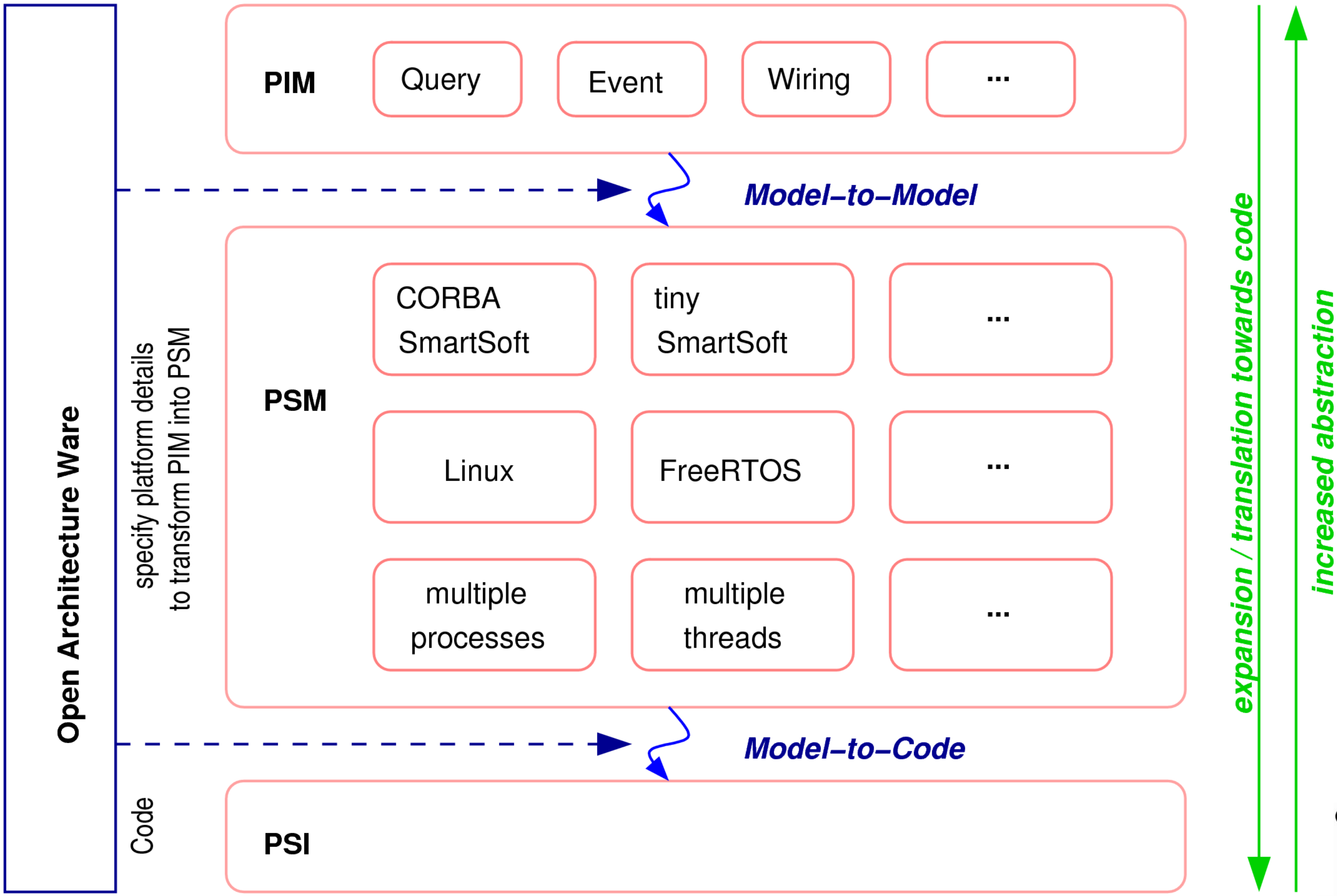


P3DX

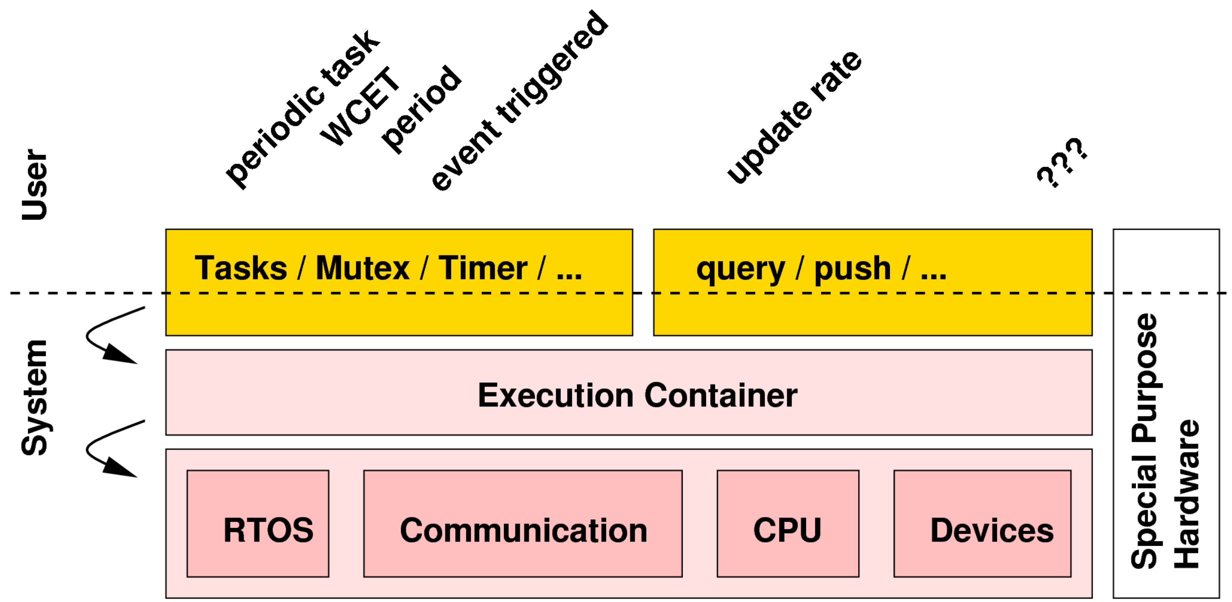
- SMARTSOFT
  (based on ACE/TAO Corba)
- Pentium M 1.5 GHz
- Player / Stage
- Linux

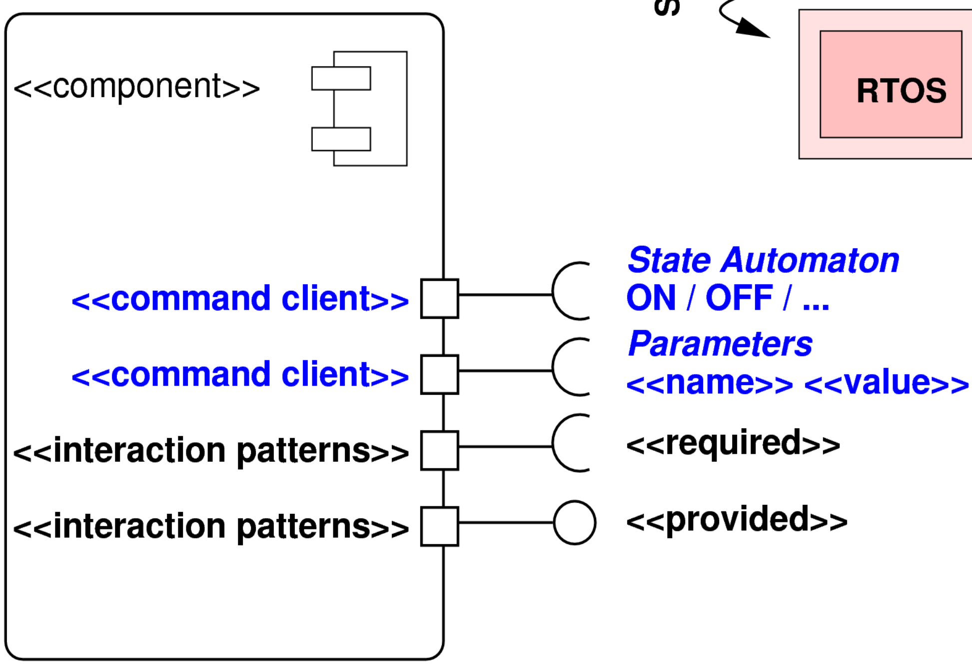
Hochschule Ulm

Model-Driven Software Design



Model-Driven Software Design

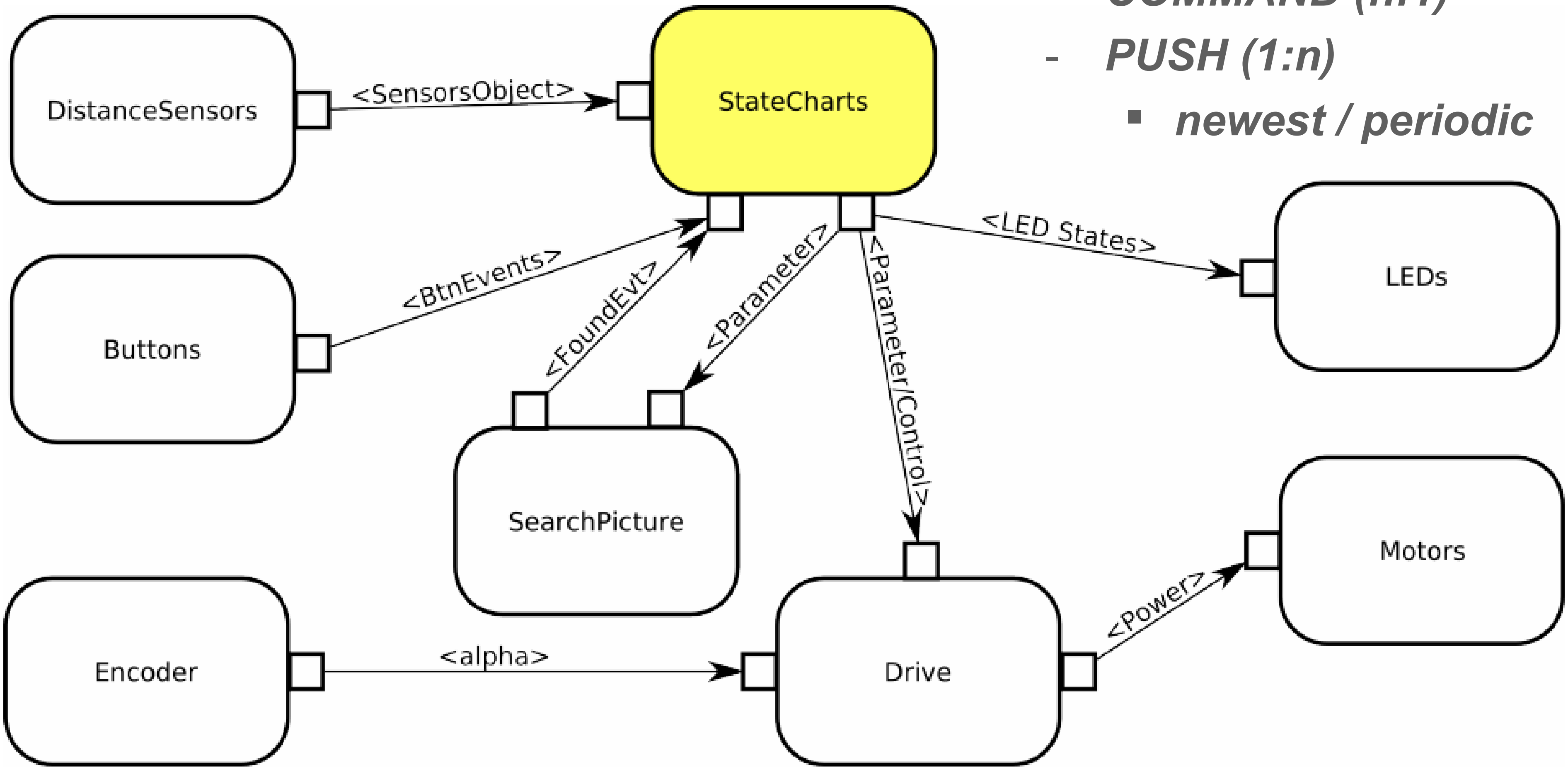




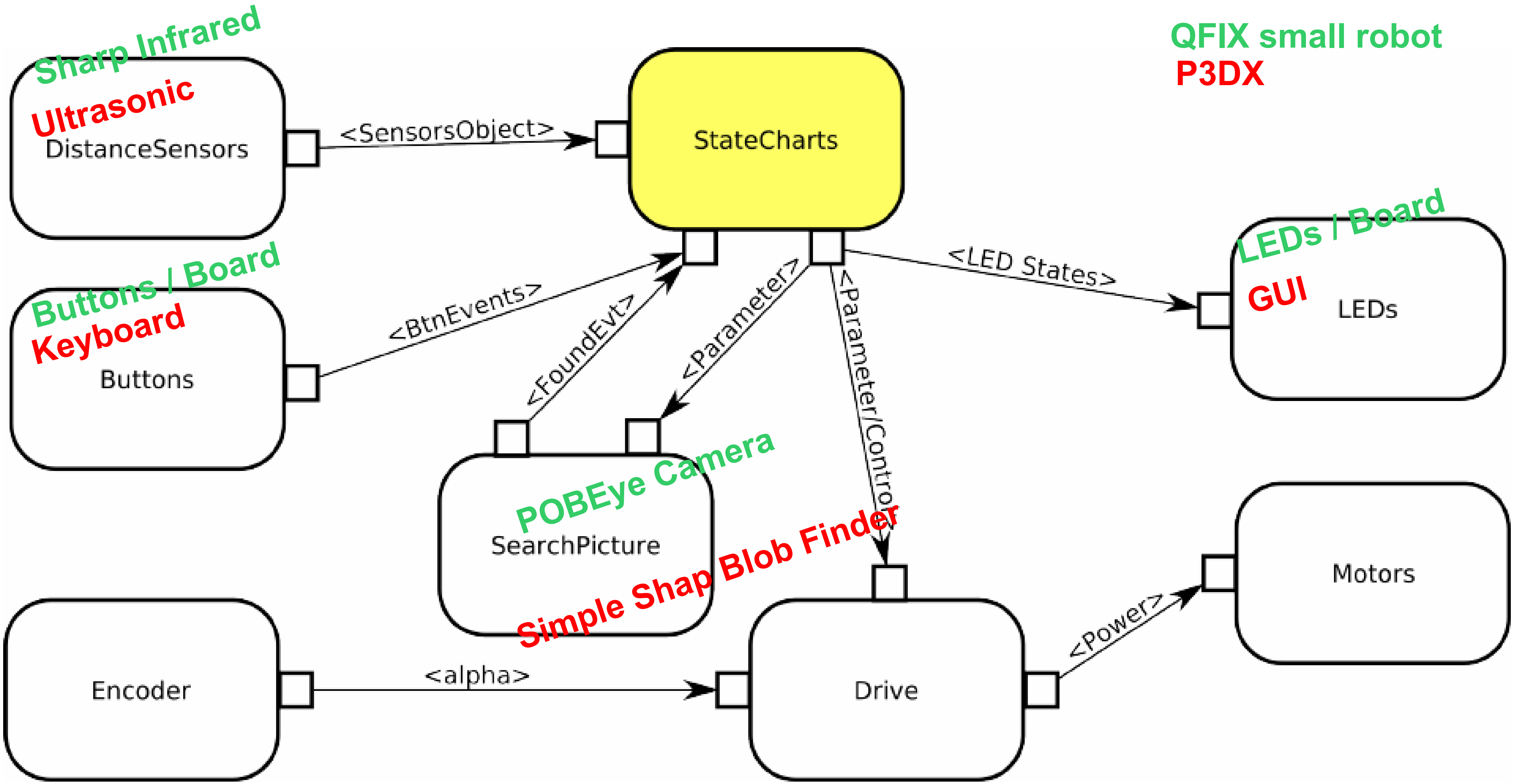
- *scheduling*
- *interrupt handling*
- *remote communication / event propagation*
- *resource control (anytime, QoS)*
- *component management facade*
- *safety watchdog / error handling*
- *...*

Model-Driven Software Design

- *COMMAND (n:1)*
- *PUSH (1:n)*
  - *newest / periodic*



Model-Driven Software Design



Model Driven Software Design: Current Work

- intellectual property
- solutions
- available as models and code blocks or libraries

=> depends on generic platform model only

Modeling Tool

EMF

GMF

- generic knowledge
- best practices
- reusable design patterns

=> coded in templates of code generator

Code Generator

xPand

SMARTSOFT

Cyclic Executive

Frameworks

generated Code

Component Architectures and Realtime

... and where is real-time ?

- tasks have periods / WCET
  - separation of scheduling parameters from task implementation
- publish/subscribe between components
  - communication guarantees
  - data flow driven
  - decoupling via globally shared state variables
- communication patterns (TO BE DONE)
  - service requests with local timeouts (decoupling)
  - QoS parameters in requests

Conclusion and Roadmap

Gap between lab prototypes and service robots that naturally populate our environment can be overcome only by addressing two fundamental issues:

- ***do not further on neglect real-time requirements***
  - mandatory towards robust, reliable, safe systems
  - take advantage of progress made in the DRE community
  - address service level agreements (QoS) at all levels, in particular at interface level between components

  - ***this requires components / middleware systems / frameworks to support resource-awareness***

Conclusion and Roadmap

What might be a starting point ?

- ***put the development process on a more abstract level***
  - use (any kind of) models in the development process to decouple robotics knowledge and problem solutions from rapidly changing (hype) implementational technologies
  - head towards code generation / composing and deployment tools / validation tools based on models

  - ***extract specific best practices, structures and requirements to build up a repository of robotics design patterns***
  - ***these could serve as basis for migrating from code-driven to model-driven design / development / validation in robotics***