HUMBOLDT-UNIVERSITÄT ZU BERLIN

Common Platform Architecture

A Simple and Clean Architecture for Participation in SPL and Simulation 3D

AI Group, Institute of Informatics, Humboldt Universität zu Berlin
nao-team@informatik.hu-berlin.de
www.naoth.de

An appropriate architecture, i.e., framework, is the base of each successful heterogeneous software project. It enables a group of developers to work at the same project and to organize their solutions. From this point of view, the artificial intelligence and/or robotics related research projects are usually more complicated, since the actual result of the project is often not clear. In particular, a strong organization of the software is necessary if the project is involved in education. In this project we developed a flexible, multi platform architecture aiming to approach the needs of those research groups.

Requirements

- the target of the project is not precisely defined
- different (concurrent) implementations for the same problem
- changing team members with different level of education
- volunteer team members (students)
- different operation systems
- cooperation between many researchers (spatially separated)
- limited computational resources of the robot
- real time requirements

Design principles

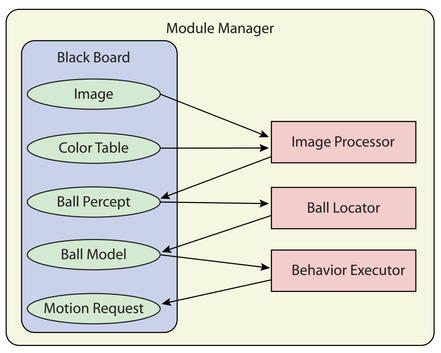
modularity: *the particular solutions for different (or the same) problems shouldn't affect each other and be easily exchangeable;*

as fast (small, simple) as possible: *the program should run in real-time on the robot and the framework should be simple enough to be maintained by future generation of students;*

easy to use and easy to test: *students with basic programming knowledge should be able to implement and test their algorithms;*

multi-platforms: *the resulting program should run on different platforms (e.g., simulation, Nao, etc.) and on different operation systems (Windows, Linux, etc.);*

transparency: *it should be possible to inspect the state of the program at any time during the runtime (e.g., which data is accessed by which module);*

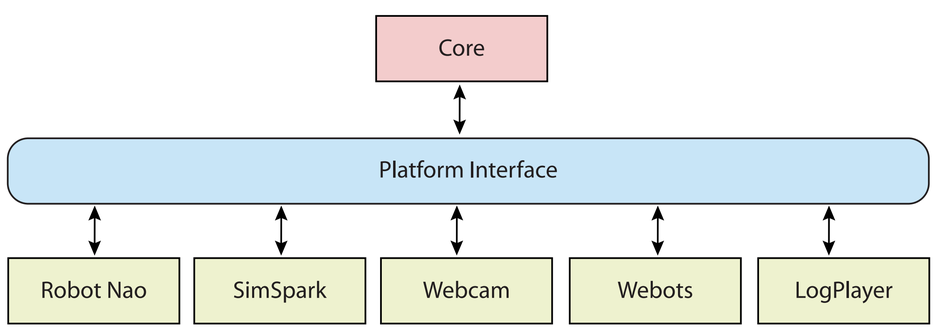


Module Framework

A blackboard architecture allows for an efficient data flow and exchangeable modules. It is always clear which module is reading or writing which data (*transparency*). The access is realized by references, so there is only minimal overhead (*efficiency*). Parallel solutions and experimental code can be dynamically switched on and off (*independence*).

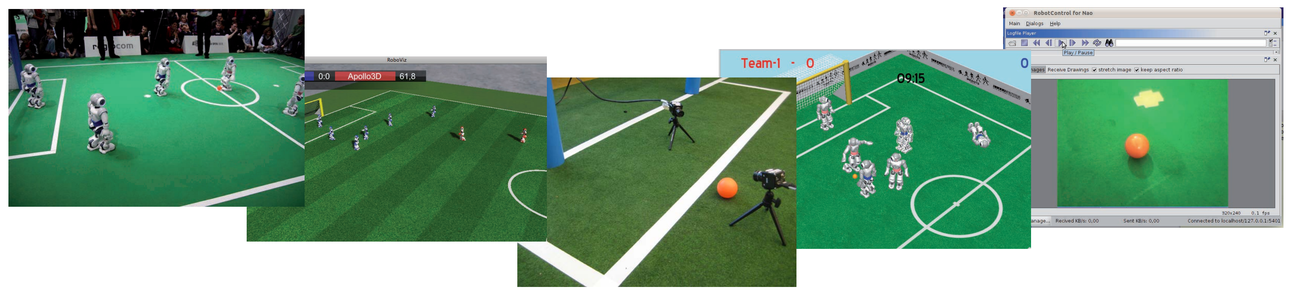
Platform Interface

A unified platform interface allows the usage of the same codebase at several robotic platforms, e.g., a real *Nao* robot, *Webots* simulator, *Simspark* simulator, Logplayer. This interface is transparent for the *Core* containing the actual algorithms, and thus it becomes independent of the particular platform used.



High Level Runtime Debug

Flexible debug architecture allows an easy implementation of runtime debugging concepts. A call-back architecture allows for an implementation of various high level debugging concepts. Some of the implemented examples are:
debug requests (activating or deactivating code parts), *modify* (allowing a modification of a variable), *stopwatches*, *drawings* (allows visualization in 2D/3D).



Communication

A generic communication infrastructure allows to transfer the information between the robots and debugging / monitoring tools. A particular implementation of the transport layer is dependent on the actual platform.

Logging

Record the state of the agent, e.g., images, and replay them with using the LogSimulator. Thereby, all data on the blackboard can be recoded and replayed in the LogSimulator selectively.

Testing

Testing Infrastructure which allows to implement and run automated tests has been implemented based on Google-Testing framework. Thereby, tests for single functions as well as for whole modules can be realized.

http://www.naoth.de