Preprint of the paper which appeared in:
Robotics and Autonomous Systems, Vol. 58, No. 7, pp. 921-929, 2010.

Towards Cooperation of Heterogeneous, Autonomous Robots: A Case Study of Humanoid and Wheeled Robots ⋆

Jutta Kiener and Oskar von Stryk *

Technische Universität Darmstadt
Simulation, Systems Optimization and Robotics Group
Hochschulstr. 10, D-64289 Darmstadt, Germany

Abstract

In this paper a case study of cooperation of a strongly heterogeneous autonomous robot team, composed of a highly articulated humanoid robot and a wheeled robot with largely complementing and some redundant abilities is presented. By combining strongly heterogeneous robots the diversity of achievable tasks increases as the variety of sensing and motion abilities of the robot system is extended compared to a usually considered team of homogeneous robots. A number of methodologies and technologies required to achieve the long-term goal of cooperation of heterogeneous autonomous robots are discussed including modeling tasks and robot abilities, task assignment and redistribution, robot behavior modeling and programming, robot middleware and robot simulation. Example solutions and their application to the cooperation of autonomous wheeled and humanoid robots are presented in this case study. The scenario describes a tightly coupled cooperative task, where the humanoid robot and the wheeled robot track a moving ball, which is to be approached and kicked by the humanoid robot into a goal. The task can be fulfilled successfully by combining the abilities of both robots.

Key words:
Heterogeneous multi-robot team, autonomous humanoid robot, task distribution and allocation, behavior control, robot middleware, robot simulator

⋆ Revised and extended version of a paper presented at the IEEE International Conference on Intelligent Robots and Systems (San Diego, 2007).

* Corresponding author. *Email address:* stryk@sim.tu-darmstadt.de. *URL:* www.sim.tu-darmstadt.de

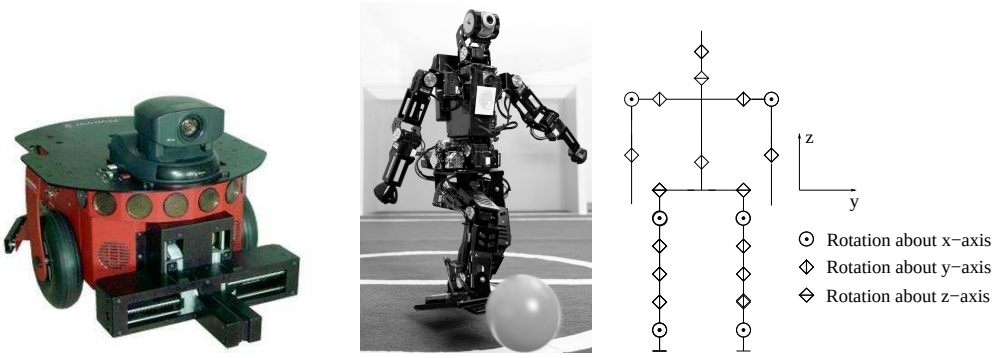
1 Introduction

With the growing importance of autonomous mobile robots in industrial and research applications the need to execute successfully challenging missions and tasks has also grown. To fulfill a large diversity of tasks with a sufficient reliability in the robot system, teams of robots are used instead of single robots with many different abilities. The majority of research in robot teams considers homogeneous robots, most of them based on wheeled locomotion. The investigated tasks differ in the complexity of structure and cooperation, starting from basic tasks as foraging [12] or exploration of an area without a specific cooperation [25] up to problems with high communication and synchronization demands, e.g., cooperative box pushing [18] or cooperative surveillance of an area [2,14] or soccer playing [9,35,36]. A classification of different stages of cooperation is given in [8].

Robots of a homogenous team are usually equipped with identical types of sensors and actuators which usually differ only slightly, e.g., because of different wear and tear. Therefore, the diversity of tasks which can be accomplished by a homogeneous robot team is quite limited. This drawback can be overcome in principle by a team of heterogeneous robots, each or several of them equipped with different sensing, perception, motion and onboard computing capabilities. Several applications have been investigated with robots, which differ only slightly in their capabilities. Although these robots are not fully identical, commonly they are still considered to form a homogeneous robot team [27]. Depending on the level of heterogeneity robots in a team are classified as weakly or strongly heterogeneous. An application with a strongly heterogeneous robot team has been developed, e.g., for aerial surveillance [24], where different robot types, a blimp, an airplane and a helicopter, cooperatively monitor a rural area for detecting forrest fires.

Another strong motivation for investigating cooperation of heterogeneous autonomous robot teams comes from the assumption that in one or two decades robot teams will usually consist of strongly heterogenous and not homogenous robots. Also many different autonomous robotic systems of different generations and capabilities will have to cooperate to achieve common tasks, presumably in an ambient intelligent environment.

Basic requisites for heterogeneous robot teams are complementary sensing, planning as well as motion and physical interaction abilities based on different hardware (e.g. sensors, actuators, computational units) and software modules. To ensure a large variety of different skills present in the robot team not only complementary but also redundant, competing abilities are required for different robots to achieve fault tolerance through sufficient redundancy in case of failures of single sense, plan or act abilities.



(a) Pioneer 2dx (b) Bruno (c) Kinematical structure of Bruno

Fig. 1. Strongly heterogeneous, autonomous robots used in the case study: Wheeled Pioneer 2dx robot and humanoid robot Bruno.

The paper is organized as follows. In Sect. 2 the robots used in the case study are presented. Sect. 3 discusses tasks and robot abilities and describes the mission scenario for the case study. Models of robot abilities are used for task assignment based on a utility function in Sect. 4. Also in Sect. 4 behavior modeling, programming and control are discussed. Sect. 5 focuses on enabling technologies, robot middleware and simulator, which are mandatory for the investigation of complex applications of heterogeneous robot teams. Results for the case study are presented in Sect. 6. Conclusions are drawn in Sect. 7.

2 Heterogeneous Robots Used in the Case Study

In this case study two strongly heterogeneous robots for indoor applications are investigated: a humanoid robot and a wheeled robot (Fig. 1).

The motion capabilities of the 55 cm tall humanoid robot Bruno (Fig. 1(b)) are based on 21 rotary joints actuated by servo motors (6 in each leg, 1 in the waist, 3 in each arm, and 2 in the neck, see Fig. 1(c)) which enable versatile walking, ball kicking and getting up abilities. The walking motions are inertially stabilized using gyroscopes attached to the robot's hip at a rate of 100 Hz. The maximum forward walking speed is 0.4 m/s. Internal sensors measure each joint angle. As only external sensors the robot uses two identical off-the-shelf CCD cameras but with different lenses. The articulated head camera offers a (horizontal) field of view of 45 deg and is used for the perception of small objects like a small ball. The second camera is attached to the robot's chest and is used to obtain a more peripheral view of the environment with a field of view of about 95 deg. The chest camera is not used in the present scenario. The humanoid robot has two onboard computers. For feedback control on the "reflex layer" a micro-controller board with a Renesas SH7145 32-bit proces-

sor running at 50MHz and 1 MByte of RAM which is programmed in C and is used for the planning and execution of humanoid leg and arm motions by coordination of multiple joints and for postural stability control. The "cognitive" computing layer includes the computations for robot vision, localization and behavior control as well as WLAN communication which are performed on an off-the-shelf Pocket PC with an Intel PXA272 processor with 520 MHz, 128 MB SDRAM, 64 MB Flash ROM and integrated power supply. The operating system is Windows Mobile 2003 CE. The two cameras are connected to this onboard computer via USB. The autonomous robot also carries the batteries for energy supply of the motors and the controller board. To enable fast and stable bipedal walking motions a careful overall lightweight design of the robot including all payload had to be made. For further information about the humanoid robot Bruno, model 2006-2007, the reader is refered to [9].

The Pioneer 2dx robot from MobileRobots is a widely used, differential drive platform with two driven wheels and one rear castor wheel. The locomotion abilities on a planar surface are versatile, stable and fast. For example, the robot can rotate on the spot as well as locomote straight ahead at a maximum speed of 1.6 m/s, which is reduced carrying an additional payload of up to 20 kg. Unlike the humanoid robot it cannot locomote on a diagonal path. The used version of the robot is equipped with a gripper with two degrees of freedom and a maximum opening of of 21.5 cm. The robot carries a standard laptop as additional computational unit, connected via RS232, replacing the built-in onboard computer by a faster processor, namely 1.6 GHz with 1 GB RAM under Windows XP, and WLAN communication ability. With the gripper the robot can lift up objects with a mass of up to 2 kg and carry at least 3.5 kg. The gripper can be extended by a seat for the humanoid robot. The power supply is given by two 9 V lead batteries. The robot is equipped with a sonar sensor ring consisting of 16 units operated at a rate of 25 Hz and a camera as external sensors. In this scenario the camera of the wheeled robot is not used to create a more heterogeneous robot team in combination with the humanoid robot.

3 Tasks and Robot Abilities

3.1 General Considerations

To achieve a mission's objective it must be decomposed in suitable tasks which can be assigned to individual robots for completion through sequential or parallel operation in time and space. The distribution of specific tasks as well as robot behavior control not only depend on the mission objective but also strongly depend on the individual robot's abilities. These consist mainly of

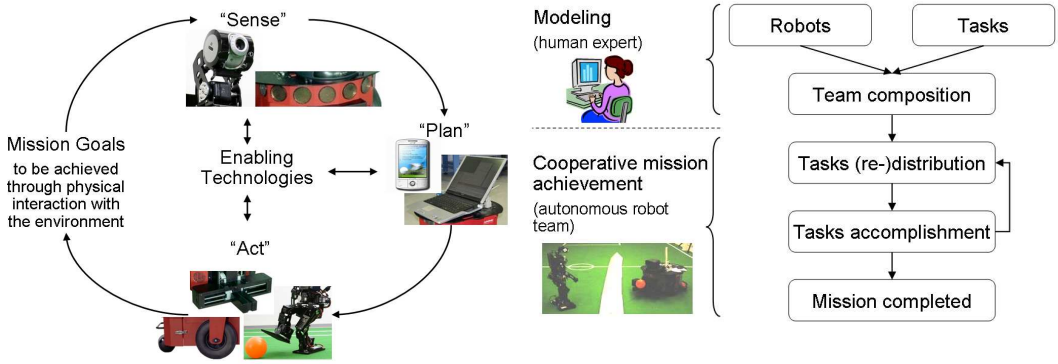


Fig. 2. Left: To achieve a mission's objective through physical interaction with the environment proper abilities in sensing and perception, onboard computing and planning as well as physical motion and interaction are required in a robot team. Right: Models provided by a human expert are needed as prerequisite for mission achievement through autonomous task allocation by the robot team.

abilities in the categories (Fig. 2 left) of (i) sensing and perception, (ii) physical motion and interaction (like locomotion or manipulation), (iii) onboard computing and planning and (iv) communication. The approach investigated in this paper for task modeling and distribution is based on models of the specific sense, plan and/or act abilities available in one or several robots which are required to achieve certain tasks. Therefore it differs from standard approaches and taxonomies for multi-robot task allocation like [11,14,25] which operate on more abstract levels that do not directly take into account specific sense, plan or act abilities of the robots.

The composition of the robot team including the allocation of proper robot abilities required to achieve certain tasks autonomously depends on the availability of proper robot hardware and software and is under the responsibility of human experts (Fig. 2 right). If the robot team consists of mainly complementary robot abilities then quite diverse tasks can be achieved in principle, but reliability in case of failure of one robot's abilities is low. If the robots have mainly similar, redundant and competing abilities then reliability in case of failure of one robot is higher but the diversity of potential tasks is much smaller. On the other hand having many robots and each with only a few, but quite diverse abilities is more reliable in case of failure of one robot than having only very few robots but with a multitude of diverse abilities.

3.2 Tasks and Robot Abilities in the Case Study

The mission scenario includes a close cooperation of autonomous mobile robots, represented by the humanoid and the wheeled robot (Sect. 2). The autonomous

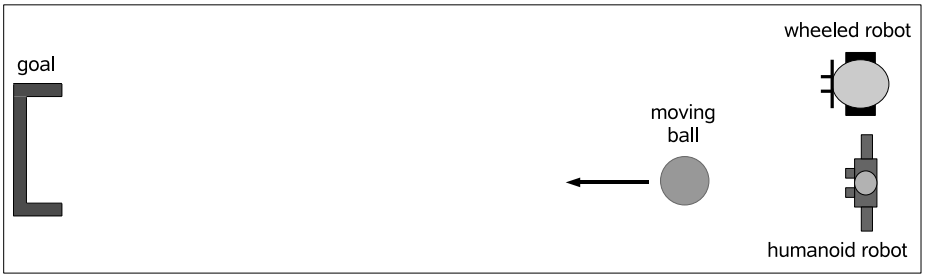


Fig. 3. Sketch of the mission scenario: A team of two robots has to find and follow an object (a ball) over a potentially long distance and finally to kick or push the ball into a goal.

robot team has to find and to track the moving ball, to reach it and to push it into a yellow goal (Fig. 3). Different abilities are required to fulfill the tasks needed to complete the mission: To find and track the ball for a possibly long distance, the robot team must be able to perceive and track the ball by a camera and must also be able to follow it sufficiently fast. To push the ball into a yellow goal pushing or kicking abilities are needed.

These tasks can be achieved better by combining the abilities of both robots than by one robot alone. Both robots offer competing and complementary abilities. Both have versatile locomotion skills, but with different maximum speeds and payloads. The wheeled robot can push the ball in principle but the humanoid robot can perform much stronger and better directed kicks. Both robots are equipped with computational units and wireless LAN for communication with each other as well as with other nodes. However, they have quite different perception abilities. Only the humanoid robot can perceive the ball and the yellow goal with its head camera. In some cases, it could be able to complete the mission itself, but in others its locomotion speed may be to small and its operation time may be too short to follow the moving ball. The payload abilities of the wheeled robot can be used to carry the humanoid robot at high speed while tracking and following the ball. Thus the scenario includes the possibility of a tight cooperation task, where the humanoid robot is carried by the wheeled robot and navigates the latter to follow the ball.

To achieve a possibly fast and reliable accomplishment of the mission, the required major tasks

- **Ball Finding and Following**: Searching for the ball, tracking the located ball, following it by robot navigation
- **Preparation for Kick and Ball Kicking**: Proper positioning towards the ball, ball kicking

are extended for a team of heterogeneous robots with different abilities in

perception, locomotion and payload, to

- **Boarding**: Boarding of one robot onto another robot (Fig. 8 (a) - (c))
- **Ball Finding and Following**: Searching for the ball and following it by the robot team where one robot may transport another (Fig. 8 (d))
- **Preparation for Kick and Ball Kicking**: Dismounting of the robots, positioning towards the ball, ball kicking (Fig. 8 (e) - (f)).

An optimal assignment of tasks to the robots must account for their different abilities. Furthermore, it is assumed that all tasks of this mission are executed in a tight cooperation of the robots, where the robots continuously communicate with each other by WLAN to exchange information using UDP, e.g. on currently perceived objects and robot behavior, for successful mission completion.

3.3 Models of Robot Abilities

The human experts are expected to provide models of the tasks as well as of the individual robot's sense, plan, and act abilities as prerequisites for autonomous (re-)distribution of tasks between the robots (Fig. 2 right). For the purpose of this paper, the ability of a robot r to perform a certain, basic task a is described by a parameter $c = c(r, a)$. The range of the (relative) characteristics is $0 \leq c(r, a) \leq 1$. The value of $c = 0$ describes that the robot is not capable of a specific basic task at all, e.g. to locomote well over a certain terrain or to perceive certain objects in the environment. On the other hand $c = 1$ denotes a robot "perfectly" capable of a task. The value of c can be based on a task specific metrics (like the average maximum speed over certain terrain) and on the relation between a robot and the robot best capable of this task. However, for the purpose of task distribution an "exact" determination of $c(r, a)$ is not required. A coarse approximation may be sufficient as long as the order of robot characteristics $c(r_i, a) \leq c(r_j, a)$ is representing the relation of certain abilities of two robots correctly.

The robots used in this case study (Sect. 2) offer both complementary and redundant abilities. Complementary capabilities enlarge the diversity of solvable tasks. In the case study several of the capabilities are complementary, e.g. object perception or transportation. Only the skills of locomotion, ball manipulation and communication are available on both robots but the first two with different properties. Therefore a failure of locomotion of one robot may compensated by the other.

Based on the qualitative rating of the robot abilities given in Table 1 a quantitative rating with weights $c(r, a)$ is applied to each ability a of a robot r as described above, see Table 2. In this case, the weights are based on expert

Table 1
Qualitative rating of complementary and competing abilities of humanoid and wheeled robots

robot type	locomotion	object perception	transportation	communication
wheeled	++	–	++	++
humanoid	+	++	–	++

Table 2
Quantitative rating of robot abilities

robot type	locomotion	object perception	transportation	communication
wheeled	0.7	0	0.9	1
humanoid	0.4	0.7	0	1

knowledge of the robots and a rule of thumb. The skill communication is assumed to be very well developed for both robots. Otherwise the envisioned tight cooperation would be very difficult to implement. It is important to note that all values $c(r, a)$ can in principle be updated online during the execution of the mission to account for occuring failures. If a robot ability a degrades or fails accordingly updated weights $c(r, a)$ can be taken into account in a redistribution of tasks between robots (cf. Sect. 4.1). Further data associated with each robot are a unique robot number and a IP address which can be extended by a team number in case of several robot teams.

4 Task Assignment and Behavior Control

4.1 Task Modeling and Assignment

The mission is decomposed into basic tasks, which can be assigned to one or several robots for execution (Sect. 3.1). This decomposition is organized by a human expert as it is usually done, see e.g. [18,19]. The tasks can be modeled as dependent tasks, which have a child and/or a parent task and are connected via time by them, or as independent tasks, which can be executed on their own. Dependent tasks are also used to model a cooperative parallel execution of several tasks. For implementation each task is modeled by

- a unique task ID,
- required robot abilities and cost for completing the task,
- any predecessor and/or successor tasks and
- the current state of execution (assigned/not assigned or executable/being

executed or executed/solved).

Basic tasks may be executed either sequentially, when no communication between robots during operation is required, or in parallel by several robots, dependent or independent of each other, which requires communication during execution between participating robots. Furthermore, lists of dependent and independent tasks with priorities are maintained during operation. Robustness against failures is achieved by

- communication to successor tasks about the current state of execution of a task,
- maximum time limits allowed for the execution of basic tasks and
- redistribution of tasks to robots if a predecessor task cannot be solved (fast enough) by a robot.

All tasks are classified based on the robot abilities, which are required for their completion. Each task demands one or more robot abilities a which are weighted with a relative factor $c(r, a)$. Each task receives a utility factor by which tasks more important for mission completion are ranked higher than others when they are assigned to robots.

Performing task assignments for multi-robot coordination based on utility or quality functions is not new, cf. e.g. [11,13,14,25]. However, unlike most previous approaches the utility function used in this paper is based on the specific sense/plan/act abilities of the robots required to achieve certain tasks (Sect. 3.3). For each task t_k a utility value

$$u(k,i) = \sum_{l=1,m} c(r_i, a_l)$$

is calculated where $r_i,\ i = 1, ..., n$, represents one of the n robots of the team, $a_l,\ l = 1, ..., m$, the abilities required to fulfill this task and $c(r_i, a_l) \in [0, 1]$ the characteristics of the ability a_l of the robot r_i (Sect. 3.3).

A task is assigned to the robot r_i, which is best qualified, i.e. currently has the highest utility value $u(k, i)$ for the task t_k. If several robots share the same utility value, then the robot r_{opt} with the lowest task load $load_i$ and the smallest robot number is chosen

$$r_{opt} : u(k, opt) = \max_{i=1,...n} load_i \cdot u(k, i) \quad \text{with} \quad load_i = \frac{(m_{all} - p_i)^2}{(m_{all})^2}$$

where m_{all} denotes the number of all tasks in the mission and p_i the number of tasks currently assigned to robot r_i.

Tasks are modeled with different states of execution and a maximum time allowed for execution. If a task has been assigned but not been solved after this time span, a redistribution of tasks is initiated. In this case it is likely that a task cannot be solved by the currently selected robot. Another, related approach for incorporating execution time for reallocation of tasks has been described in [26].

The tasks are distributed in a decentralized manner using a contract-net type negotiation approach [33] based on robot communication. The original contract-net protocol is modified in such a way that all tasks are rated by one robot first by computation of the utility values. Then the results are communicated to the next robot for rating. The last robot of the team receiving the tasks for rating communicates to the other team members the resulting distribution of tasks to robots. This approach ensures that tasks which can be executed in parallel are allocated to a robot before their initiation. The effort consists of $O(m \cdot n)$ communication steps in case of m tasks and n robots. If the number of robots is moderate, then for the number of tasks $m > n$ can be expected as more robots than tasks would be an usual situation.

4.2 Behavior Modeling, Programming and Control

Modeling, programming and control of complex behaviors for cooperative multi-robot applications are challenging tasks in the dynamic environments of many real-world problems. Besides different methodological approaches for behavior control like reactive or deliberative paradigms mature *technologies* are required for programming robot agent behaviors. These must be able to cope with necessary real-time requirements, only partial or noisy observability of the environment, and the unpredictability of dynamic environments. Technologies for programming and control of robot behavior should met further requirements (cf. [30]) including

- Modularity: Highly complex robot behavior can only be managed if it can be structured in a modular way. Modularity is also a prerequisite to enable several human experts to develop and program robot behavior simultaneously. Modularity also supports reusability of single modules of a complex robot behavior control for other robots or applications. It also enables the composition of complex robot behavior from more basic behavior modules.
- Portability: The technology for programming robot behavior should be independent from a specific robot plattform or application.
- Flexibility: There should be no restrictions on the type of behavior control which can be implemented, i.e. any type of reactive or deliberative, discrete or continuous behavior should be enabled.
- Usability: The programming of robot behaviors should be supported through

monitoring and debugging facilities.

A number of formal specification methods for programming robot and agent behaviors efficiently have been proposed and applied like the Behavior Language [4], the Reactive Plan Language [3], the Configuration Description Language [21], the Planning Domain Definition Language [22], the Task Description Language [32], COLBERT [17], Petri Net Plans [37] and the Extended Behavior Programming Language (XABSL) [20,30,31]. However, most of them do not meet all of the requirements mentioned above. Several of the mentioned specification methods are based on finite state machines or make use of them, e.g. [4,17,20,30].

In this paper XABSL [20,30] is applied for behavior modeling, programming and control of heterogeneous multi-robot teams. XABSL is based on hierarchical state machines, enables deliberative as well as reactive behavior control paradigms and was developed to meet the above mentioned requirements [30]. It consists of several components: a modular behavior architecture based on concurrent hierarchical, finite state machines, a specification language for describing hierarchical state machines, a compiler generating documentations and intermediate code to be parsed by the runtime system, and a C++ runtime library used to execute the behavior inside an agent software environment.

Common features between XABSL and Petri nets as an alternative formal approach for modeling robot behavior as described in [37] are hierarchical decomposition of complex behaviors, concurrent execution of partial behaviors, and support for multi-robot cooperation. In principle modeling of robot behavior with Petri nets (PNs) and hierarchical finite state machines (HSMs) have a similar expressiveness as also concurrent behavior execution is possible with XABSL. However, it seems that HSMs as used in XABSL are more intuitive than PNs becaus of more compact behavior descriptions. An advantage of PNs formalism is the possibility of the analysis and verification of certain formal properties of the specified behaviors (like reachability or liveness). Some formal verification (like reachability of states) is also possible with HSMs. Further details about a comparison between Petri Net Plans and XABSL can be found in [31]. In this context it should be noted that in [38] a graphical behavior modeling tool using PNs has been developed which automatically generates XABSL source code.

In XABSL the hierarchy of finite state machines consists of agents, options and basic behaviors. An *agent* represents the whole robot behavior, e.g. of one robot in the team. The *options* of this agent denote different sub-behaviors. The lowest level of the hierarchy is represented by *basic behaviors* by which different types of executable motion primitives or output signal can be implemented. The agent can be described by a directed, acyclic graph with options as nodes and one root option als designated initial node. Standardized vari-

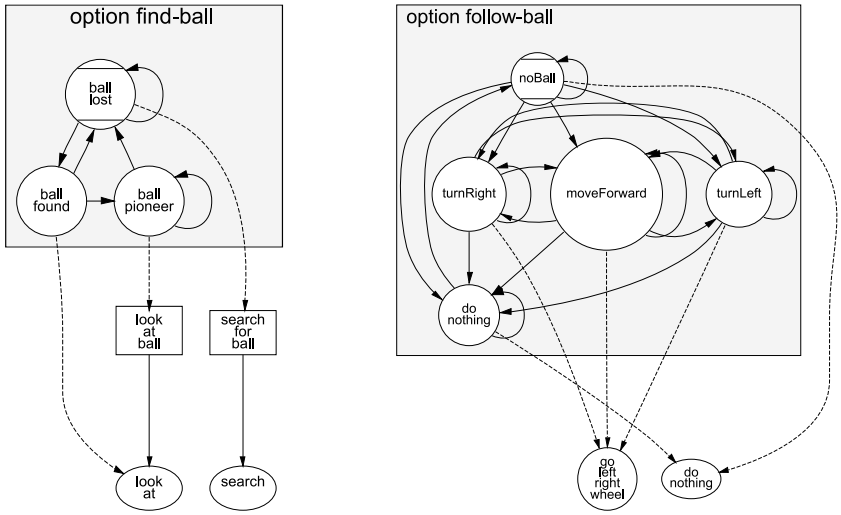


Fig. 4. Option graphs of robot behavior for the tasks Ball Finding and Following: humanoid robot (left) and wheeled robot (right).

ables (e.g. as integer or real variables), so-called symbols, are used in XABSL as input, output or internal variables and can be connected by mathematical, e.g. arithmetical or logical, operations. The options can be controlled by a decision tree through the values of the symboles which are set in functional modules, e.g. with the current distance of the robot to an object of interest. The current state of the HSM describing an agent is given by the so-called option activation tree of active options or basic behaviors starting from the root option. This state is updated in certain time steps depending, e.g. on the frequency of incoming new information from sensing and perception modules.

The XABSL source code of an option is written in a C++ like description language and contains decisions and transitions depending the current value of symbols and actions which are to be performed while a specific state is active and which is interpreted from a plattform independent runtime engine. XABSL also offers tools for editing, visualization, monitoring and debugging of behavior programs and is used by about a dozen research groups currently. It is available from www.xabsl.de.

4.3 Behavior Modeling for the Case Study

The options available in an agent representing a robot's behavior can be associated to specific tasks to be performed by the robots. The option graphs used for the behaviors of the humanoid and the wheeled robots are displayed in Figs. 5 and 4.

In the option *find-ball* in Fig. 4 left, executed by the humanoid robot, the robot searches for the ball. If a valid ball is recognized in the camera image, then

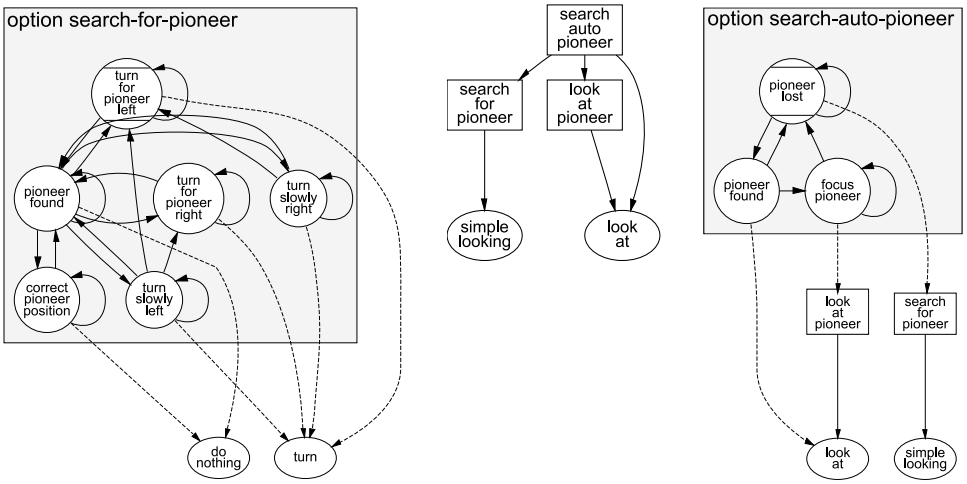


Fig. 5. Option graph of the humanoid robot behavior (left) for the boarding task, the agent for its head control (middle) and one option of the head control (right).

the position of the ball and a reliability which depends on the recognition quality in the image are communicated to the wheeled robot. The humanoid robot starts the option *look at ball*, which controls the head motors to keep the recognized ball in the middle of the camera image. When the ball is lost, the behavior calls the option *search for ball*, which executes a search path for the head camera by the neck joints. This search path is precalculated to cover the area in front of the robot, where the ball is likely to be. If this search is not successful specific search locomotions of the humanoid robot or the wheeled robot transporting the humanoid robot could be initiated. The transitions between options denoted by edges depend on the state of the current world model included in the current value of the symbols which in this case depend on the results of the ball recognition in image processing.

The option *follow-ball* in Fig. 4 right describes the behavior of the wheeled robot transporting the humanoid robot. It depends on the information about the current state of the world model, mainly the currently perceived ball position, communicated by the humanoid robot. The robot can move to the left or right with different turning angles depending on the current ball position resp. move forward. If no ball position is communicated to the wheeled robot within a certain time the wheeled robot stops. It only starts to move again, if the reliability of a communicated ball position is sufficiently high. The wheeled robot tries to keep the ball in a center position in front of it.

5 Enabling Technologies for Heterogeneous Robot Teams

For efficient development, operation and maintenance of heterogeneous autonomous robots in research and industry further enabling technologies are

becoming increasingly important, namely robot middleware and robot simulator. Their availability and further development are indispensable for the efficient investigation and realization of more complex and more reliable multi-robot applications in the future [6].

5.1 Robot Middleware

Modularity and reusability of hardware and software modules of heterogeneous, autonomous robots can only be achieved by a flexible and standardized robot middleware which ensures not only timely and consistent communication between the various hard- and software modules. A number of efforts have been made in the past and are still underway, e.g. Microsoft's Robotics Studio and and Willow Garage's Robot Operating System to name only two prominent, commercial activities. However, there are yet no solutions available that meet all requirements and are widely accepted.

In this paper the software framework RoboFrame [29] is applied which has been developed to address the special needs of heterogeneous teams of autonomous robots with a large variety of hardware and software components. Its main characteristics are platform independency, modularity and high efficiency and that it is also bundled with a library of common components for robot control software, which provides much more support to the robot programmer than a robot middleware alone.

RoboFrame has been designed as a framework that can also be applied to robots with only small-scale onboard computing abilities. It offers flexible communication mechanisms either based on messages (ring buffers) or shared memory (black boards) and supports a variety of operating systems like Linux, FreeBSD, Windows 2000/XP/CE 5. Main elements are modules, processes and connectors (as generalization of communication interfaces). *Modules* capsule functional components of a robot control software like image perception, localization, world model, behavior control, motion planning and generation. Instead of hard-wired interfaces between modules descriptive specification of in-/out-going data are used. *Processes* are the runtime environments for modules which can be executed asynchronously on a single or distributed on several onboard computers. Processes are implemented as threads of the operation system. Modules are executed thread-safe at variable or given execution times. The platform abstraction layer offers multi-threading and synchronization mechanisms as well as file and network functions and mathematical functions. A graphical user interfaces offers debugging capabilities and visualization of algorithm performance. More information about RoboFrame, which is available to interested researchers upon request, can be found at www.dribblers.de/roboframe.

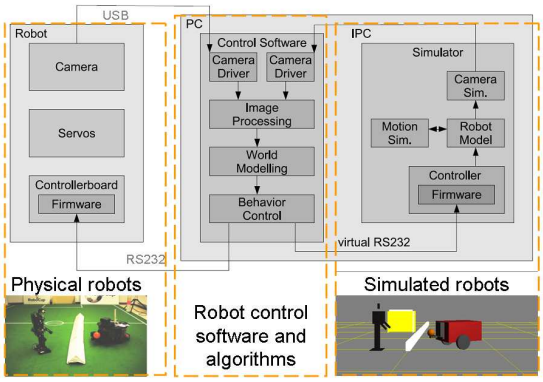


Fig. 6. Software-in-the-loop testing of robot control software using a robot simulator.

5.2 Robot Simulator

The development of behavior control software for teams of autonomous robots is a highly challenging task. Reasons for lack of performance as well as for failure are extremely difficult to analyze by experimental evaluation only, because an autonomous robot usually consists of a highly interacting set of different software and hardware modules. Therefore one of the most valuable tools supporting the development of control software is software-in-the-loop testing using simulation of a robot's sensing and/or motion system under real-time conditions (Fig. 6). The benefits of simulation are manifold and include testing of robot software under repeatable and controllable conditions and unlimited availability which is not possible with real robot hardware.

The general requirements on the simulator differ significantly depending on the scope of the simulation experiment, e.g. testing of localization algorithms depending on perceived environmental information and odometry or testing of behavior control algorithms with different levels of localization accuracy. High physical accuracy may be mandatory for some scenarios, e.g. for investigations in postural stability control of fast humanoid robot locomotion, but may be not important for other, e.g. testing of team coordination strategies. Furthermore, physics-based robot simulation may impact the real time performance of a simulator severly. If a simulation depends on external packages for physics simulation or other purposes, adjusting the accuracy and level of physical detail of the simulation is difficult if not impossible. One possible solution is to use different simulators for different purposes but this requires consistent interfaces to different simulators. This approach is used, e.g., with Gazebo [16] for 3D physics and Stage for 2D simulation, but is not practical for other simulators. There is also another tradeoff between accuracy and level of physical detail of the simulation and size of the robot team investigated.

A variety of 3D robot simulators exist. Most of them rely on external packages for physics simulation. Quite often the Open Dynamics Engine [34] is used,

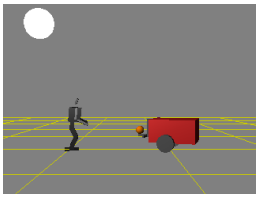
e.g. in Webots [23] and Gazebo [16]. Other packages used are PhysX [1] by NVIDIA (used in Microsoft Robotics Studio) or game engines like the Unreal Engine [7] (used in USARSim [5]). Most of the existing robot simulators are tuned for real-time computations and physical plausible but not necessarily accurate simulation results. For example, the level of physical detail in robot dynamics, e.g. multibody system dynamics, simulation as well as the numerical integration method of motion dynamics, e.g. fixed step size Euler versus variable step size higher order methods, which strongly influence the fidelity of simulation, can not be changed. Another drawback of existing robot simulators is that validation and calibration of robot simulation for rating and adaption of the simulation accuracy by a systematic comparison with data from robot experiments is not supported well.

To overcome these limitations the multi-robot simulation framework is being developed (MuRoSimF [10]). It allows the flexible and transparent exchange and combination of any of the algorithms used for the simulation of robot motion or sensing systems in a scenario with individual level of realism. Also different algorithms can be selected for different robots, e.g. a dynamics simulation for the humanoid robot's and a kinematic or point mass model for the wheeled robot's locomotion. Different level of details in the robot sensing system can, e.g., account for different distortions of camera images or for the effects in a data record of distance sensors from a laser scanner or a sonar ring resulting from motion of the robot during recording of data. Further information about MuRoSimF, which is available to interested researchers upon request, can be obtained from www.dribblers.de/murosimf.

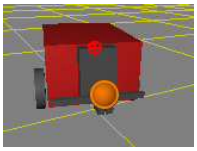
For the purpose of testing the control software of the hetereogeneous robot team in the case study, the wheeled robot is modeled with two actuated wheels and an articulated 2-axes gripper which is needed in other scenarios. The humanoid robot kinematics is modeled with 21 articulated joints (Fig. 1(c)). Its the head camera with simulation of focal length and distortion has been calibrated from real head camera images.

5.3 Multilevel Testing

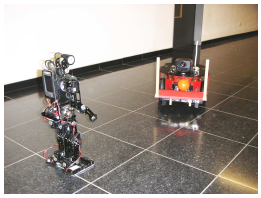
Conventional testing and debugging mechanisms used in software engineering to ensure that software is free of errors are applicable to robot control software only to a very limited extend because of large uncertainties in robot perception and motion, the high dimension of possible robot and environmental states and real-time requirements. However, it should be noted that a tailored robot middleware, like RoboFrame, integrated with a simulator, like MuRoSimF, enables multilevel testing strategies for robot control software. These include component tests, online and offline tests as well as software-in-the-loop tests in



(a) View of the beginning of the simulated task of the humanoid robot boarding onto the wheeled robot

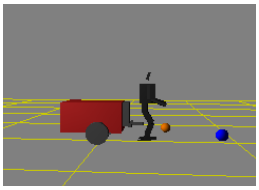


(b) Simulated image of the humanoid robot's head camera in the scene left

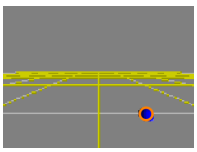


(c) View of an experiment of the humanoid robot starting to board onto the wheeled robot

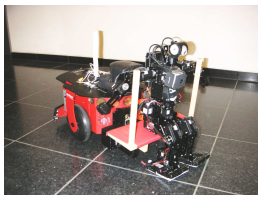
(d) Head camera image of the humanoid robot with recognized objects



(e) View of the simulated ball following task of the robot team



(f) Simulated head camera image of the humanoid robot while following the ball



(g) Experiment of the ball following task

(h) Head camera image of the humanoid robot with recognized ball

Fig. 7. Results from simulated and real experiments for the beginning of the boarding task (top row) and the ball finding and following task (bottom row) where the wheeled robot transports the humanoid robot which navigates both.

combination with real robot hardware or an adequate robot simulation [28].

6 Results

6.1 Simulation Results

Fig. 7 depicts scenes from the simulation of the tasks Boarding and Ball Following. In the first scene the humanoid robot recognizes the pose of the wheeled robot with a color-based perception of the red color of the wheeled robot and the orange color of a second ball used as marker on the wheeled robot. The wheeled robot turns based on the communicated pose until the orientation and distance is suitable for the humanoid robot to board onto it. The humanoid robot determines the distance to the wheeled robot with a size-based projection of the recognized orange marker. The recognition of the wheeled robot is robust enough to account for differences in the size of the recognized red area of the wheeled robot. The results of the simulated and real scenarios

depicted in Figs. 7 (a) and (c) as well as (e) and (f) match reasonably well. The simulated camera image in Figs.. 7 (b) and (f) as well as the real ones in Figs. (d) and (h) show the recognized objects of interest (the wheeled robot and the orange marker or the ball).

6.2 Runtime of Task Assignment

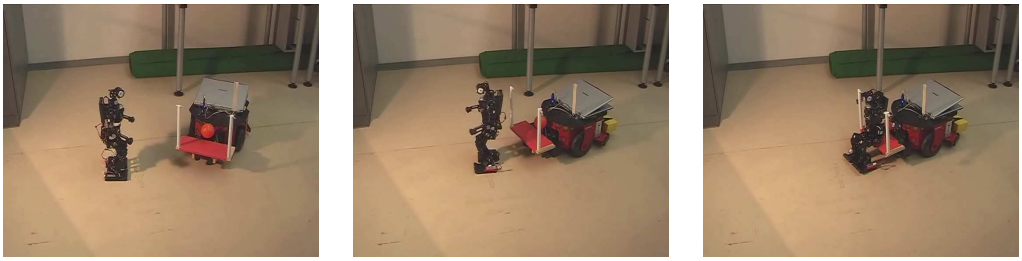
The runtime measurement for the task assignment described in Sect. 4.1 has been tested for two different types of tasks: Three dependent tasks with 10 subtasks each and 10 independent tasks. The measurement in Table 3 has been accomplished both for simulation on a Windows XP Laptop (1.6 GHz), which is also used as onboard computer for the wheeled robot, and the Pocket PC with Windows CE (512 MHz), the main onboard computer of the humanoid robot.

Table 3
Runtime measurements for a set of dependent and independent tasks, both executed in simulation and on the humanoid robot's Pocket PC

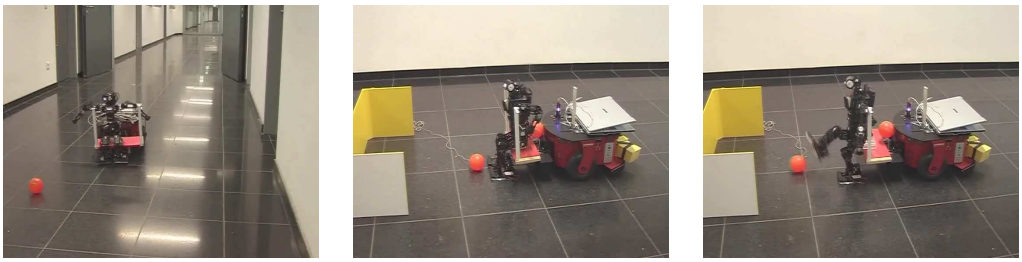
	dependent tasks		independent tasks	
Simulation	∅ 4 ms,	max. 31 ms	∅ 5 ms,	max. 15 ms
1 robot	∅ 13 ms,	max. 25 ms	∅ 5 ms,	max 11 ms
3 robots	∅ 14 ms,	max. 24 ms	∅ 11 ms,	max. 12 ms

6.3 Experiment of Mission Scenario

One of the several experiments performed has been documented in a video which is available online [15]. The whole mission takes about 3 min. After both robots have entered the scene it takes about 3 s for the humanoid robot to recognize and localize the wheeled robot. About 15 s later the robots have prepared for the beginning of the boarding task (Fig. 8(a)) which takes about 27 s (Fig. 8(c)). After the ball has been detected the robot team follows the ball which is moved by a human by the help of a cord attached to the ball more than 15 m through a hallway (Fig. 8(d)). For more than 40 s the humanoid robot navigates the wheeled robot to follow the ball. For test purposes, the ball is then moved out of the field of view of the humanoid robot by the human. The wheeled robot stops and the humanoid robot starts to search for the ball by purposely moving its neck joints. After about 8 s the ball is put back in front of the robots and moved again followed by the robot team. When the ball comes to rest close to a yellow goal (Fig. 8(e)) the humanoid robot dismounts from the wheeled robot, prepares for kicking and kicks the ball into the goal (Fig. 8(f)) to complete the mission.



(a) Communication about robot position (b) Autonomous boarding (c) Boarding completed



(d) Ball Following (e) Preparation for ball kicking (f) Kicking the ball

Fig. 8. Tasks required for mission achievement as performed in the experiment: The humanoid robot mounts onto the wheeled robot (upper row) and navigates the wheeled robot to follow the moving ball in a fast and reliable way and finally completes the mission by kicking the ball into a goal after dismounting from the wheeled robot (lower row).

7 Conclusions

A new mission scenario for a team of strongly heterogeneous, autonomous robots, a humanoid and a wheeled robot, requiring tight cooperation has been presented and successfully investigated. Several methodologies and technologies required to achieve the long-term goal of cooperation of truly heterogeneous autonomous robots have been discussed. Example solutions for task distribution based on a utility function rating robots sense/plan/act abilities required for task achievement, robot behavior modeling and programming using the extensible agent behavior specification language XABSL, robot middleware using the robot software framework RoboFrame and robot simulation using the multi-robot simulator framework MuRoSimF have been presented and applied successfully to the robots in the case study. The methods and technologies presented in this paper are not limited to this specific scenario, but aim at more general heterogeneous robot teams and missions.

References

[1] AGEIA PhysX website, http://www.ageia.com/physx/ (2007)

[2] M.A. Batalin, G.S. Sukhatme: Coverage, exploration and deplayment by a mobile robot and communication network, in: Intl. Workshop on Information Processing in Sensor Networks, Palo Alto, 376 - 391 (2003)

[3] M. Beetz, D. McDermott: Executing structured reactive plans, in: L. Pryor, S. Steel (eds.): Proc. AAAI Fall Symposium: Issues in Plan Execution, AAAI Technical Report FS-96-01 (1996).

[4] R.A. Brooks: The behavior language; user's guide, Technical Report AIM-1227, MIT Artificial Intelligence Lab (1990)

[5] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, C. Scrapper: USARSim: a robot simulator for research and education, in: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 1400-1405 (2007)

[6] S. Carpin, I. Noda, E. Pagello et al.(eds.): *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2008)*, Lecture Notes in Artificial Intelligence **5325**, Springer-Verlag (2008)

[7] Epic games, unreal engine, http://www.epicgames.com (2007)

[8] A. Farinelli, L. Iocchi, D. Nardi: Multi-robot systems: A classification focused on coordination, IEEE Transactions on System Man and Cybernetics Part B **34** (5), 2015-2028 (2004)

[9] M. Friedmann, J. Kiener, S. Petters, H. Sakamoto, D. Thomas, O. von Stryk: Versatile, high-quality motions and behavior control of a humanoid soccer robot, International Journal of Humanoid Robotics **5** (3), 417-436 (2008)

[10] M. Friedmann, K. Petersen, O. von Stryk: Simulation of multi-robot teams with flexible level of detail, in [6], 29-40 (2008)

[11] B.P. Gerkey, M.J. Mataric: A formal analysis and taxonomy of task allocation in multi-robot systems, International Journal of Robotics Research **23** (9), 939-954 (2004)

[12] D. Goldberg, M.J. Mataric: Coordinating mobile robot group behavior using a model of interaction dynamics, in: Intl. Conf. on Autonomous Agents, Seattle, Washington, 100-107 (1999)

[13] L. Iocchi, D. Nardi, M. Piaggio, A. Sgorbissa: Distributed coordination in heterogeneous multi-robot systems, Autonomous Robots **15** (2), 155-168 (2003)

[14] N. Kalra, D. Ferguson, A. Stentz: Hoplites: A marked-based framework for planned tight coordination in multirobot teams, in: Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), Barcelona, Spain, April 18-22, 1170-1177 (2005)

[15] J. Kiener, O. von Stryk: Cooperation of heterogenous, autonomous robots: A case study of humanoid and wheeled robots. Video, available online at http://www.dribblers.de/media.video.en.php (2007)

[16] N. Koenig, A. Howard: Gazebo - 3D multiple robot simulator with dynamics, website http://playerstage.sourceforge.net/gazebo/gazebo.html (2003)

[17] K. Konolige: COLBERT: A language for reactive control in Sapphira, in KI-97: Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence **1303**, Springer-Verlag, 31-52 (1997)

[18] C.R. Kube: Task modelling in collective robotics, Autonomous Robots **4** (1), 53-72 (1997)

[19] T. Längle, H. Wörn: Human-robot cooperation using multi-agent-systems, Journal of Intelligent and Robotic Systems **32**, 143159 (2001)

[20] M. Loetzsch, M. Risler, M. Jüngel: XABSL - A pragmatic approach to behavior engineering, in: Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Beijing, China, Oct. 9-15, 51245129 (2006)

[21] D. MacKenzie, R. Arkin, J. Cameron: Multiagent mission specification and execution, Autonomous Robots **4** (1), 29-52 (1997)

[22] D. McDermott: PDDL – The planning domain definition language, Technical report, Yale Univ. (1998)

[23] O. Michel: Cyberbotics ltd. – webots(tm): Professional mobile robot simulation, Intl. Journal of Advanced Robotic Systems **1** (1), 39-42 (2004)

[24] A. Ollero et al.: Multiple eyes in the skies: architecture and perception issues in the COMETS unmanned air vehicles project, IEEE Robotics and Automation Magazine **12** (2), 46-57 (2005)

[25] L.E. Parker: ALLIANCE: An architecture for fault-tolerant multi-robot cooperation, IEEE Transactions on Robotics and Automation **14** (2) 220-240 (1998)

[26] L.E. Parker: Lifelong adaptation in heterogeneous multi-robot teams: response to continual variation in individual robot performance, Autonomous Robots **8** (3), 239-267 (2000)

[27] L.E. Parker: Intelligence, reasoning, and knowledge in multi-vehicle systems: recent advances and current research challenges, in: Proc. 1st IFAC-Symposium on Multivehicle Systems, Salvador, Brazil, Oct. 2-3 (2005)

[28] S. Petters, D. Thomas, M. Friedmann, O. von Stryk: Multilevel testing of control software for teams of autonomous mobile robots, in [6], 183-194 (2008)

[29] S. Petters, D. Thomas, O. von Stryk: RoboFrame – A modular software framework for lightweight autonomous robots, in: Proc. Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware of the 2007 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Oct. 29, 2007

[30] M. Risler: Behavior control for single and multiple autonomous agents based on hierarchical finite state machines, Ph.D. Thesis, Technische Universität Darmstadt (May 2009).

[31] M. Risler, O. von Stryk: Formal behavior specification of multi-robot systems using hierarchical state machines in XABSL, in Proc. AAMAS'08 Workshop on Formal Models and Methods for Multi-Robot Systems, Estoril, Portugal (May 2008)

[32] R. Simmons, D. Apfelbaum: A task description language for robot control, in: Proc. IEEE Conf. on Intelligent Robots and Systems (IROS), Oct. 13-17 Oct., 1931-1937 (1998)

[33] R.G. Smith: The contract net protocol: High-level communication and control in a distributed problem solver, IEEE Transactions on Computers **29** (12), 1104 1113 (1980)

[34] R. Smith: ODE – Open Dynamics Engine, http://www.ode.org (2007)

[35] P. Stone, M. Veloso: Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork, Artificial Intelligence **110** (2), 241-273 (1999)

[36] H. Utz, F. Stulp, A. Mühlenfeld: Sharing belief in teams of heterogeneous robots, in: RoboCup 2004: Robot Soccer World Cup VIII, Lecture Notes in Computer Science **3276**, Springer-Verlag, 508-515 (2005)

[37] V.A. Ziparo, L. Iocchi, D. Nardi, P.F. Palamara, H. Costelha: PNP: A formal model for representation and execution of multi-robot plans, in: Proc. 7th Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008) 79-86 (May 2008)

[38] O. Zweigle, R. Lafrenz, T. Buchheim, U.-P. Käppeler, H. Rajaie, F. Schreiber, P.Levi: Cooperative agent behavior based on special interaction nets, in: T. Arai et al. (eds.): Intelligent Autonomous Systems 9 (IAS-9), Proc. 9th Intl. Conf. on Intelligent Autonomous Systems, Tokyo, March 7-9, 651-659 (2006)