DISTRIBUTED APPLICATIONS FOR ROBOTIC SYSTEMS USING ROBLET®-TECHNOLOGIE

Daniel Westhoff
University of Hamburg
Germany

Hagen Stanek
genRob.com
Germany

Jianwei Zhang
University of Hamburg
Germany

Speaker: Daniel Westhoff, University of Hamburg, Germany, westhoff@informatik.uni-hamburg.de

Topic: Robot Programming

Keywords: service robots, distributed systems, Roblet®-Technology, software development

Abstract

In this paper we present the application of a technology that establishes a framework for task-oriented programming of distributed robotic systems. The framework allows writing distributed control or monitoring programs and enables the programmer to send programs referred to as Roblets® to Roblet®-servers running on different computers within a local area network. At the University of Hamburg the framework is used to create a distributed system to control a service robot which can accomplish several tasks in an office environment. For this purpose, the system combines the control of the service robot, sensor information of the robot's sensors and stationary sensors in the environment as well as the results of tracking algorithms, path planning algorithms and other algorithms. Several examples are given to illustrate the usability of the framework. They result is a complex distributed system which is kept maintainable by consistent application of the Roblet®-Technology.

1. Introduction

The field of service robotics has seen a lot of advances over the last years, but still lacks usability and robustness. We think that one reason for this is the absence of a unifying software architecture that handles the miscellaneous challenges which the software engineers encounter. These challenges vary from the development of distributed applications to the handling of the diversities of the different hardware platforms present in service robotics.

In future, task-level programming of service robots will be done by non-expert personnel. Modern robotic research analyses new interface technologies for the interaction between humans and robot systems. The goal of these research activities is the creation of interfaces that allow people to communicate with machines in a natural way, integrating different modalities like speech or gesture. Robust implementations of these techniques will only be available in a few years. Therefore, for this paper non-expert personnel is assumed to be familiar with programming in general, but not required to have special knowledge about robotics and networking.

Consequently a framework for distributed robot applications must abstract from dedicated hardware and networking. The objective of such a middleware must be to shorten the time to install and develop complex distributed applications. In [1] and [2] we presented such a framework referred to as Roblet®-Framework and introduced Roblet®-Technology as a solution to the problems that one has to deal with when distributed systems are applied in robotics. A main feature of the framework is the ability to integrate existing solutions to specific robotic problems. We will show that it is possible to encapsulate libraries for motion control for manipulators as well as for mobile robots. A variety of hardware devices which are present on modern service robots can be integrated into the architecture. A layer of abstraction generalizes the access to these devices. Thus, existing applications can be transferred to other robotic systems without changes.

As mentioned before, the proposed framework hides the network and the used hardware. Nevertheless more experienced developers are granted access to the details and parameters of the employed hardware. The concept of the framework is implemented in Java™ and was tested on different service robot systems.

For this paper the Roblet®-Framework is applied to the service robot of the TAMS Institute at the University of Hamburg. The **TA**MS-**S**ervice-**R**obot (TASER) is built from standard components. It is a mobile robot with two manipulators and a multi-modal man-machine interface. Figure 1 shows TASER and presents its main components. The robot operates during normal workdays in the office environment of the TAMS Institute. It is used as an experimental platform to point out the usability of the Roblet®-Technology.

In the last two years, several distributed applications were developed during research, diploma and student projects. Especially the student projects show that less skilled programmers are enabled to build complex systems in limited time when they use the Roblet®-Framework. Thus, the participants of the projects could concentrate on the specific robotic and algorithmic problems rather then learning nothing but how to program basic networking communication with all the error handling needed.

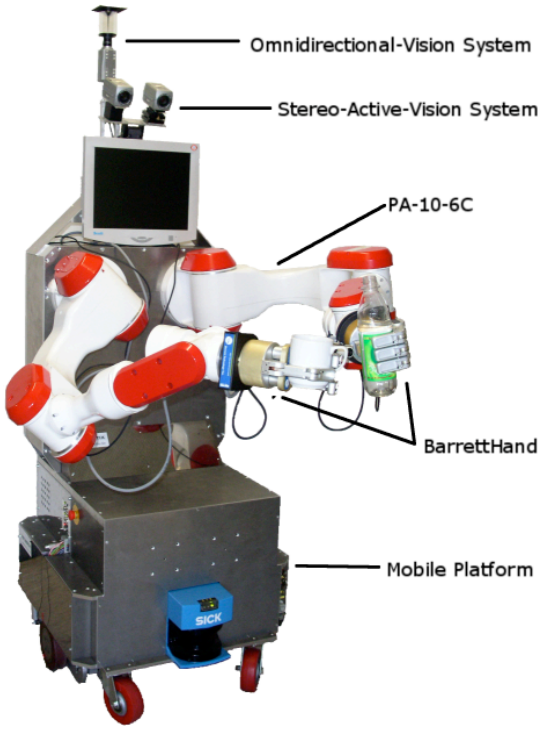

Figure 1: The multimodal service robot TASER of the TAMS Institute at the University of Hamburg.

The remainder of this paper is organized as follows: In section 2 an overview of existing software architectures in robotics is given. Section 3 gives a description of the hardware of TASER and shows how the resulting problems occur in service robotics in general. The next section introduces the software architecture and how the hardware is encapsulated by the proposed framework. In Section 5 some preliminary experimental applications with our robot are presented. Three applications using Roblet®-Technology are explained in more detail. Section 6 gives a conclusion and an outlook on future work.

2. Related Research

This section gives an overview of existing software architectures for service robots. Recently, a workshop during the 2004 conference on Intelligent Robots and Systems (IROS) tried to list the various research activities in the field of robotic middleware [3]. In the following, some of these activities are discussed. Besides, further related research projects are presented.

The *OROCOS* project started in 2000 as a free software project due to the lack of reliable commercial robot control software [4]. It is divided into two decoupled sub-projects: *Open Real-time Control Services* and *Open Robot Control Software*. The first one is a real-time software framework for applications for machine control. The second one is a set of libraries and an application framework including generic functionality mainly for manipulators. Support of mobile robots is still in its early stages. In 2004 the *Orca* project emerged from the *OROCOS* project [5]. It adopts a component-based software engineering approach using *Ice*[1] for communication and the description of interfaces. The project's goals are to enable and to simplify software reuse and to provide a generic repository of components. The use of different middleware packages for inter-component communication is extensively discussed on the project's home page.[2] In Addition to writing custom middleware, the use of CORBA and XML-based technologies is compared to Ice. Orca is available for various operating systems and compiles natively.

[6] introduces the *Player/Stage* project, a client-server framework to enable research in robot and sensor systems. It provides a network interface to a variety of robot and sensor hardware and to multi-robot simulators. Multiple concurrent client connections to the servers are allowed. Client applications connect over TCP sockets. The project's server software and the simulators are limited to Unix-like operating systems.

In [7] *MARIE* is presented, a design tool for mobile and autonomous robot applications. It is mainly implemented in C++ and it uses the *ADAPTIVE Communication Environment* (ACE)[3] for communication and process management.

---

[1] http://www.zeroc.com
[2] http://orca-robotics.sourceforge.net
[3] http://www.cs.wustl.edu/~schmidt/ACE.html

Figure 2: The biotechnological laboratory at the University of Bielefeld.

In 2002 *Evolution Robotics* introduced the *Evolution Robotics Software Platform* (ERSP) for mobile robots [8]. It is a behavior-based, modular and extensible software available for Linux and Windows systems. The main components that are included are vision, navigation and interaction.
In [9] a service robot for a biotechnological pilot laboratory is presented. The mobile platform of this robot is equal to parts of our robot TASER. A seven degrees-of-freedom arm is mounted on top of the mobile platform. The system is designed to take samples from a sampling device, handle a centrifuge, a fridge and other biotechnological equipment and fulfill the complete process of sample management. It relieves the laboratory personnel of monotonous time-consuming tasks. Nevertheless it operates in a standard laboratory with standard equipment. An easy-to-use script language is proposed to define high-level work sequences. The scripts are parsed by the robot's control software and the robot fulfills the defined task. This encourages the idea of simplifying the programming of robots but lacks the flexibility of a widespread programming language including network programming for distributed systems.
The Roblet®-Framework emerged partly as a side-project of [9]. It was used to control, monitor and evaluate the mobile platform. Because of this, parts of the sample management can be accomplished and controlled by using Roblet®-Technology. The Roblet®-Framework was integrated into the existing application step by step and can be used in addition to the script-based control. Figure *2* shows the biotechnological laboratory. In figure *3* the architecture of the automation system is illustrated.

3. Robot-Hardware for Multi-Modal Interaction and Service Tasks

A lot of service robots which are described in literature have in common that they are built from special hardware. Partly, the hardware is designed for certain robot systems only.
One of the particular objectives while building TASER was to assemble it mainly from off-the-shelf hardware. This guarantees that most of the solutions developed for this robot can be transferred to other systems. Two further advantages are that damaged parts are easily exchangeable and for most parts of the hardware existing programs and libraries are used. The main hardware parts of the system are:

- a mobile base including the power supply,
- two robot arms with six degrees of freedom,
- two three-finger robotic hands for manipulation,
- a stereo vision system including a pan-tilt unit,
- an omnidirectional vision system,
- a monitor with loudspeakers for interaction and
- one Pentium IV computer for control.

The service robot is powered by eight lead-acid batteries which supply a main power of *48 Volt* with the total power of *3.84 kWh*. This guarantees an overall independent working time of approximately eight hours. The onboard computer contains additional interface hardware, such as: a high-speed serial interface, an ARCNET interface, a CAN interface, a frame grabber and a wireless network adapter. As operating system of the computer a standard Linux derivate is used.

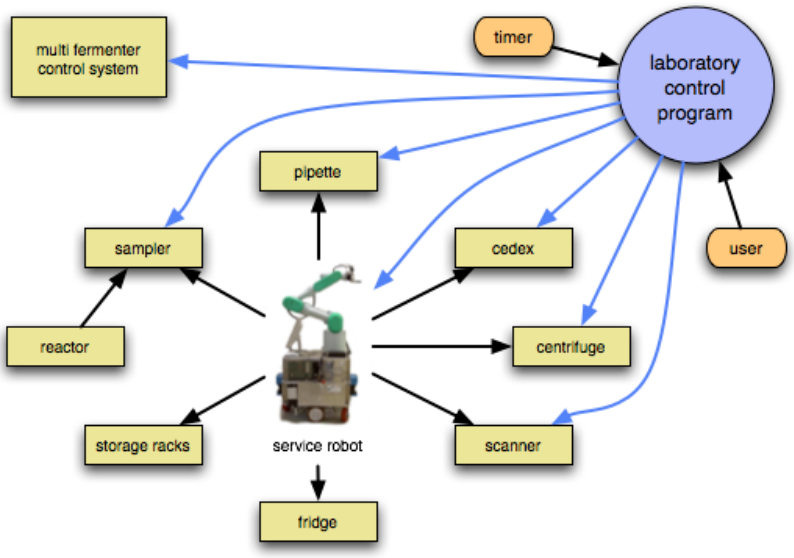

Figure 3: The architecture of the automation system for the laboratory in [9].*The control of the motion sequences of the mobile robot and the path planning between the different biotechnological devices is implemented with the Roblet®-Framework.*

The software architecture introduced in section 4 will encapsulate the different components of TASER's hardware. Therefore, these components are analyzed in detail in the following subsections. Their prerequisites, limitations and particularities constitute the foundations of many design decisions for the software architecture. They are presented here since they exemplarily show difficulties which are similarly met by other service robotic platforms.

3.1 Mobile Platform

The mobile platform of TASER is a modified MP-L-655 from NEOBOTIX[4] with a special extension to mount two robot arms. The platform is equipped with a differential drive with integrated wheel encoders, two laser-range finders and a gyroscope for navigation. The parts are accessed via a *Controller Area Network* (CAN). The mobile platform is controlled by a C/C++ application developed in [9] and enhanced for TASER. It provides a TCP/IP interface to trigger motion commands. Motion commands allow safely moving the robot forward or backward over a given distance and to rotate the robot to a given orientation. Additionally, pose information from the localization algorithm or measurements from the sensors can be requested. The localization algorithm incorporates an extended Kalman filter and reaches an accuracy of ±1 *cm* in position and ±1° in orientation.

3.2 Manipulator System

The manipulator system of TASER currently consists of two *Mitsubishi Heavy Industries PA10-6C* robot arms. A *BH-262 BarrettHand* is attached as a tool to each arm. With this design the system attains a humanlike workspace and silhouette.
The PA10-6C has six degrees of freedom (DoF) and a kinematical length which is similar to the length of a human arm. The manipulator is capable of carrying a payload up to *10 kg* although its weight is only about *38 kg*.[5] It is accessed through an ARCNET interface and controlled by the *Robot Control C Library* (RCCL). RCCL was developed by [10]. [9] extended RCCL to control the PA10 robot series.
The BarretHand is a three-finger robot hand with 8 DoF. The inner joints of each finger are coupled to the outer by a *TorqueSwitch*™ similar to a human hand [11]. Two fingers are linked by a spread joint. A serial interface and the controllers for the four motors are integrated in the housing of the hand. A self-programmed C++-class library controls the BarrettHand. A real-time control cycle is integrated in the library to make the input from the force sensors of the hand available to other applications. Another self-programmed library encapsulates the arm control and the BarrettHand control. It allows a coupling of the force-sensor measurements of the hand with the arm control. This way, the arm motions can be force-controlled and supervised. The library provides high-level access to the manipulator system. High-level functionality includes Cartesian motions of the manipulators and a series of different grasps with the hands. Figure *4* illustrates three of the available grasps. Additionally, some predefined complex motions and work sequences are available.
A micro-head camera is mounted on each palm of the BarrettHand. The camera is used as an input device for visual servoing, to fine-position the arm and hand when operations are performed.

---

[4] http://www.neobotix.de
[5] This is the weight when running the arm at a power of *100 Volts*. We use *48 Volts* so the payload might be less.

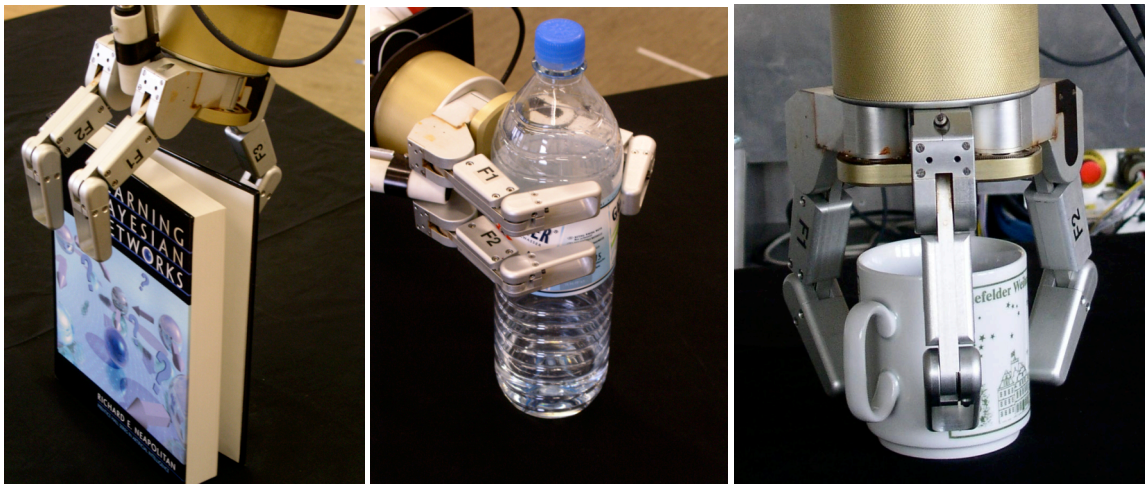

Figure 4: *The images show different grasps that can be accomplished by the BarrettHand. These grasp are available as high-level functions through a library encapsulating the control software of the BarrettHand.*

3.3 Interaction

The robot is equipped with a versatile multimodal interface for human-robot interaction which uses video, audio and laser range data gathered by the robot's active vision system, omnidirectional cameras, microphones and laser-range finders. The passive setup does not interfere with the environment besides the robot itself. There is no need for special hardware that is cumbersome to set up and use like data gloves or magnetic field sensors. The interface is intended to make the interaction between the robot and humans simpler and more intuitive.

The main sensors for the interaction are several camera systems mounted on the robot. Beside the micro-head cameras on the hands, the system has a stereo-camera system mounted on a pan-tilt unit. The third vision system of the robot is an IEEE1394-camera that is combined with a hyperboloidal mirror. This combination forms the omnidirectional vision system.

4. Software Architecture

In the previous section we introduced some of the capabilities of TASER's different subsystems. In this section we propose a novel software architecture to ease the development of high-level programs combining the functionality of robotic subsystems.

4.1 Roblets®

The basics of the proposed framework are realized with Java™ and use Roblet®-Technology, a concept firstly introduced in [1]. Roblet®-Technology is a client-server architecture where clients can send parts of themselves, referred to as Roblets®, to a server. The server, referred to as Roblet®-server, then executes the Roblets® with well-defined behavior in case of malfunctions. Notice that not only data is transmitted between the client and server but complete executable programs. This can be compared to Java™ Applets but with the difference that Roblets® are not downloaded but sent. Complex setups can consist of multiple client applications and Roblet®-servers. A Roblet® terminates if the execution of its code finishes normally or throws an exception. Exceptions are sent back to the client application. In addition, a Roblet® can be terminated by a client application remotely or by the Roblet®-server directly. After a Roblet® terminates, the Roblet®-server resets itself to a well-defined state.

Roblet®-Technology is applicable to all kinds of distributed systems but it has several features that make its integration into robotic applications useful. In general, high-level applications in service robotics are mostly distributed systems. Besides one or multiple mobile robots, there are visualization- and control applications that run on workstations in a local area network. Sometimes there is no direct access to the robot systems via keyboard, mouse and monitor but only through a wireless network. Roblet®-Technology introduces the possibility to develop, compile and execute an application on one workstation. When the application is executed it will send parts of itself to available servers and spread in the local network. Roblets® may send parts of themselves to other servers as well. The network communication is hidden from the programmer by the Roblet® library, which simplifies the overall development. That means, the network is transparent and developing distributed applications based on Roblet®-Technology is like developing one application for one workstation.

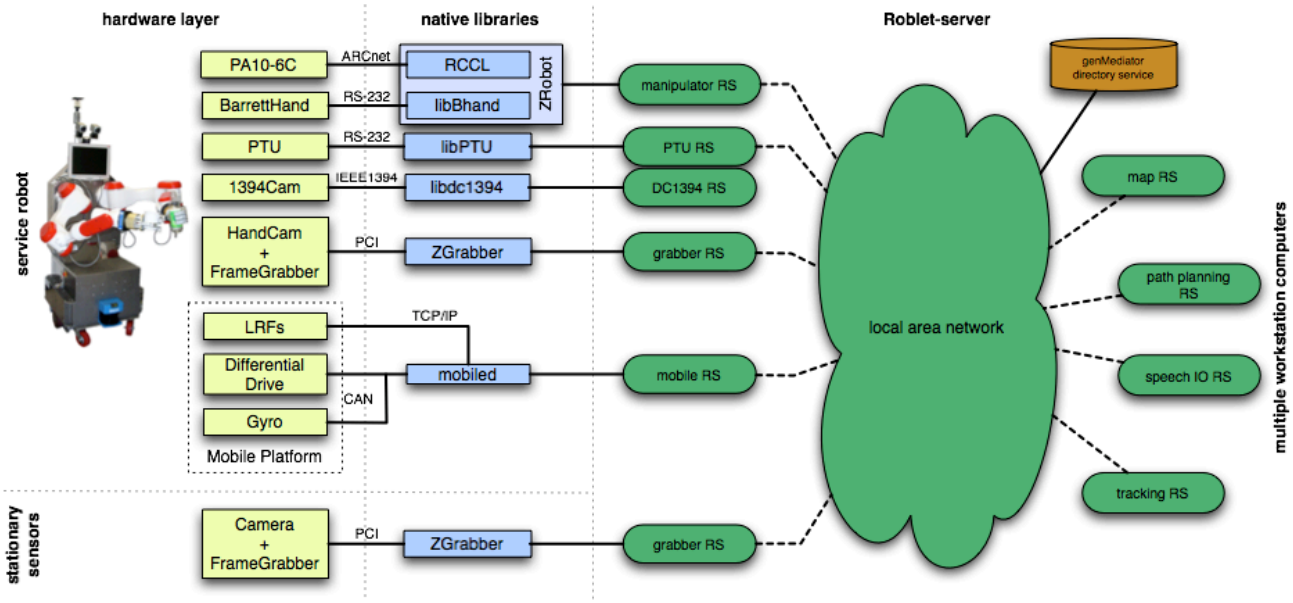

Figure 5: *The software architecture of the presented robotic system: Roblet®-servers (RS) are used to provide a hardware abstraction layer. Generalization is realized by this hardware abstraction. Distributed applications are independent of the particular hardware of the robot system. Some Roblet®-servers and connections are left out for clarity.*

Access to the remote servers is encapsulated in a client library, reducing the execution of a Roblet® on the remote system to one method call.

4.2 Modules

For robotic applications we propose *modules* to extend the basic Roblet®-server provided by the Roblet®-Framework. Figure *6* shows a chart of the structure of a Roblet®-server. A module is loaded when the Roblet® server is started. It is meant to encapsulate a class of similar functionality. For the robot TASER we developed several modules: One module merges the functionality of the mobile platform, a second module wraps the manipulator system including the robot arms and the hands. There are modules for the different vision systems, the pan-tilt unit, a speech module and other parts of the interaction subsystem. Figure *5* gives an overview of the main parts of the current software architecture for TASER. The system incorporates several smaller Roblet®-servers and multiple client applications not shown in the figure for clarity. Notice that the map server and the path planning server do not run on the robot's control computer but on a workstation in the local network. This allows the integration of information gathered
by multiple robots. For example, in the case of dynamic map adjustment this relieves the robot's onboard computer of some computationally expensive tasks which need no real-time capabilities.

4.3 Units

Modules are further divided in units. Units are Java™ interfaces that are implemented within the modules. Units make up the hardware abstraction layer in our framework. For example, a module encapsulates the localization subsystem of a mobile robot and a Roblet® wants to query the current *pose* estimate[6] of the robot. Then the module would implement a unit which defines a method to get the pose. On another robot there may be another localization system encapsulated by another module. However, if the module implements the same unit, the same Roblet® can be executed on both robots and works without changes. Nonetheless, special features of a subsystem are made available to Roblets® if module-specific units, e.g. to change special parameters of a subsystem, are implemented. Therefore a Roblet® has only access to units, it does not know anything about a module and a module's implementation of the interface. The whole concept is strictly object-oriented.
By introducing units, the framework is able to generalize access to similar classes of subsystems without loosing access to their special features. Additionally, units introduce a possibility of versioning to the system. If new features are integrated into a module then new units will be introduced. As long as older units are still available, all client applications and their Roblets® using these old units still work. This has proven to be of great use since complex applications often consist of dozens of client applications and Roblet®-servers developed and maintained by different programmers. A transition to new units can be accomplished step by step for each client application.

[6] A *pose* is the triple of 2D position coordinates and the robot's orientation.

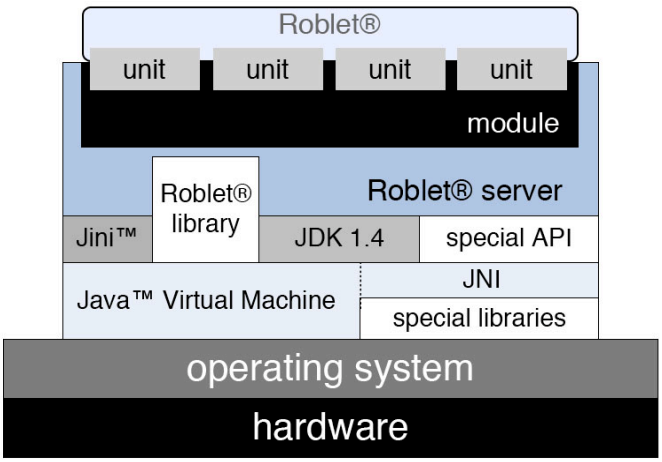

Figure 6: *The chart shows the structure of a Roblet®-server and how it hides the hardware from a Roblet®.*

4.4 Platform Independence

There were several reasons to use Java™ to implement the concept of Roblet®-Technology: First of all, Java™ virtual machines and compilers are available for a variety of different platforms from embedded systems over PDAs to workstation computers. All of these different systems can be found in the field of robotics. Since Java™ source code is compiled into bytecode, the programs can be compiled on any of these systems and executed on another system without change. Besides, Java™ provides a vast standard library available on all of these systems. The standard libraries include techniques for network communication like *Remote Method Invocation* (*RMI)* or *Jini™* used within the Roblet® framework.[7] These well-tested libraries ensure reliable operation of the framework since they are used in millions of internet applications as well.

In contrast, using other programming languages like C/C++ would require the compilation of the source code for each target machine. Additional libraries, e.g. CORBA, Ice or ACE, are required for network communication, which demand additional knowledge of the programmer. Further on, these libraries may sometimes be only available for a subset of systems present in a robotic scenario. In future, the .NET framework from Microsoft may become an alternative to Java™ since it also compiles source code into a bytecode first. Nonetheless, to date .NET is only available for Windows platforms. The open-source projects implementing .NET for other platforms do not provide full support yet.

Since Java™ has no real-time capabilities, programs written within a Roblet® are not intended to contain real-time control loops. There exists a specification for a real-time Java™ virtual machine but at present no implementation [12]. The Roblet® framework allows less skilled programmers to design and develop robotic applications without in-depth knowledge about the used subsystems. First experiences using the Roblet®-Framework in lectures for graduate students have proved this.

5. Applications in Service Robotics

In this section we will describe three preliminary experimental applications which emphasize the capabilities of the proposed architecture. The applications control TASER when it accomplishes high-level tasks using a combination of its various components. They have been developed in the course of several different research projects to evaluate the functionality of the underlying libraries which control the deployed hardware.

5.1 Object Manipulation

The first example is a combination of localization, planning of paths, object manipulation and interaction where the robot is instructed to operate a light switch. An operator chooses one of the light switches in the laboratory of the TAMS Institute and commands the robot via speech orders or an interactive dialog to operate it.

The application uses various Roblet®-servers shown in figure *5*. One client application running on a workstation computer in the laboratory controls the sequence of actions. Therefore, multiple Roblets® are sent to the Roblet®-servers used in this scenario and RMI connections to the Roblets® are established. Via these connections the client program gathers sensor input and starts actions.

---

[7] Jini™ is used by the directory service *genMediator* shown in figure *5*. Roblet-servers register at the directory service when they are started and are removed if they are shut down. Client applications are automatically informed about added or removed Roblet-servers in the local area network.

Figure 7: *Left: TASER is operating a double-switch. Right: TASER is instructed to grasp a cup by speech and gesture.*

First, a Roblet® on the Roblet®-server for the speech IO informs the client application that a light switch is to be operated by the robot. Then the position of the light switch which is stored as a point of interest in a map is requested from the map server. This Roblet®-server encapsulates a database in which map elements like obstacles and points of interest within the robot's working area are stored. Multiple applications can get, alter or add map elements of the database concurrently through this Roblet®-server. The server coordinates the access to the database if multiple Roblets® run in parallel. After the Cartesian position of the light switch was determined, the client application sends a Roblet® to the path planning server to get a path to the light switch. The path planning server has its own connection to a Roblet® on the map server and uses the current map information for its path planning algorithm. After a path is calculated, the client program commands the robot to drive to the light switch on the given path and maneuver the robot to a suitable position so that one robot arm can reach the switch. Obstacles on the path to the switch are autonomously avoided by the robot. The final position of the robot arm relative to the switch is obtained from a method call to the arm-operations library which is provided by the corresponding Roblet®-server. By solving the kinematical chain, the robot computes a position and orientation suitable to operate the switch. After this position has been reached by the robot arm, a Roblet® tries to fine-position the manipulator in front of the light switch with the hand camera by visual servoing. Therefore, the Roblet®-server on the robot has a module which provides units to access the hand camera as well as a module with units to move the manipulator. A unit of the hand camera's module provides positioning errors calculated on the observed images. When the arm is centered in front of the switch, an approach move is made by the arm which is force-controlled by sensor input of the BarrettHand. The sensors of the hand are precise enough to stop the movement of the arm when the finger touches the switch. In the final step a finger operates the switch. By using a final movement of this finger, even switches like the double-switch shown in the left image of figure *7* can be operated independently.
Some of the units can only be accessed by one Roblet® at a time. This guarantees that the whole sequence of actions presented above cannot be interrupted by other client applications. After the sequence has finished these units are released and other client programs with different Roblets® can take over control and use the robot for different tasks.

5.2 Object Grasping and Transport

The second example is given by the task of object grasping and transport. The user can advise the robot to fetch and carry objects lying on a table via an interactive dialog.
Each source of information about humans interacting with the robot is encapsulated into its own Roblet®-server and can thereby be employed by Roblets®. Besides the Roblet®-server for speech IO, the active stereo head, the omnidirectional vision system and the laser range finders are encapsulated. The developer of the client program chooses which sources he wants to incorporate into his application.
After the robot was instructed which object it has to fetch, the robot plans its path to the object using the Roblet®-server for path planning. After reaching a position suitable for object grasping, the robot tries to identify the object by means of object recognition. In case of ambiguities the interaction system is used with other Roblet®-servers to resolve the situation. For example, if the robot cannot distinguish objects on the table, it uses the active vision system to recognize pointing gestures and gaze to resolve the position the manipulator of the robot is intended to move to. Such a situation is shown in the right image of figure *7*. Additionally, the user can teach the robot new grasping motions and grasps [13][14].

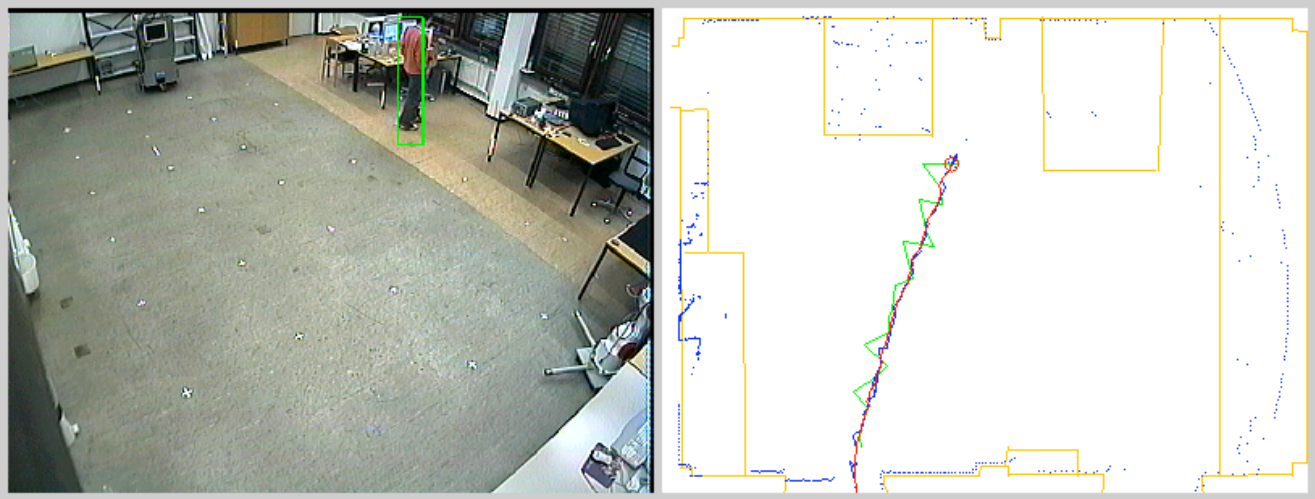

Figure 8: Multimodal tracking: The left image shows the current image of the stationary camera in the laboratory of the TAMS institute. The tracked person is marked by the green box. The right image shows the tracking result of the individual Roblet®-servers, where the green line represents the results of the camera-based tracking and the blue line is the result of the laser-based tracking. The red line indicates the tracked path after the fusion of the results of the single sensor tracking. The additional blue dots in the right image indicate the current measurements of the laser range finders.

When the object is successfully recognized, the robot selects a suitable grasp for the object from an internal grasp database and executes it. After grasping the object the robot moves the manipulator back into a safe transporting position. If the transporting position has influence on the security outline around the robot, this outline is modified and a path to where the object is to be placed will be calculated based on the new outline. Thus, the outline information is sent to the path planning server and the server takes the new outline into account for the next path planning. When the final position has been reached the robot will set down the grasped object and is available for new tasks again.

5.3 People Tracking

A third example uses the laser range finders of the robot and a stationary camera in the laboratory to track people. Each sensor is encapsulated by its own Roblet®-server. A tracking server was developed which establishes a connection to the servers for the sensors via two Roblets®. Each Roblet® tracks people using the data provided by the underlying sensor. The position information about the currently tracked people as well as the uncertainty about the position estimate is sent to the tracking server. The tracking server fuses the tracking results provided by the two Roblets®. Figure *8* shows some tracking results. The amount of data sent over the network is reduced if only the tracking results are transmitted instead of the raw sensor data.

The tracking server provides the fused tracking results to other client applications. In addition, motion patterns are generalized to describe paths often used by people in the office environment of the TAMS institute. A graph is used to describe these generalized paths. An exemplary graph generated after a few people were tracked is shown in figure 9. Furthermore, the tracking server provides trajectories which predict the motions of tracked people based on the generalized motion graph. Therefore, the probability that people walk along a segment of the generalized graph is learned. The predicted trajectory is used for path planning, e. g. to avoid paths where people are moving or to intercept people to hand over an object.

6. Conclusion

The presented software architecture reveals a possibility to build high-level applications for service robots using standard components of the robot as well as specialized hardware. Its implementation and extension is simple and takes advantages of different technologies in Java™ like Jini™, RMI and JNI. The platform-independence of Java™ reduces the effort to port applications to different platforms which can be found in distributed robotics.

All examples in section 5 incorporate several different Roblets®. In the first two cases the programmer only had to concentrate on the overall work sequence, a still not trivial task. The complex control of the individual actions accomplished by the service robot was only triggered through the various Roblet®-servers. The subsystems hidden by the servers executed the control autonomously. Therefore, the Roblet®-framework introduces a separation in the development of service robotic applications. On the one hand, there are close-to-hardware programmers providing high-level functionality through Roblet®-servers. On the other hand, there are task-oriented programmers using multiple Roblet®-server to create complex distributed applications for service robotics.

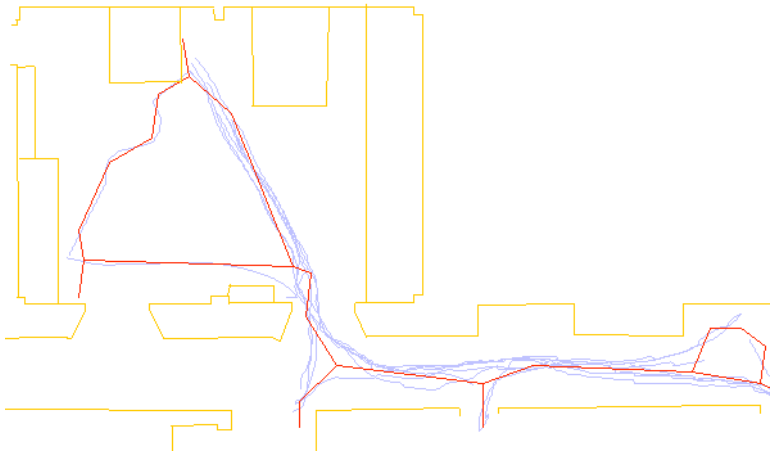

Figure 9: The blue lines are the tracked paths which are determined by multi-sensor fusion. The red graph represents a generalized motion pattern describing frequently used paths.

The third example extended the setup of Roblet®-servers. Since the tracking algorithms where incorporated within Roblets®, the existing servers where not changed. This technique allows developing novel algorithms while the overall system stays the same and remains usable by other applications. When the tracking algorithms are tested and stable they can be incorporated in the module and their results become accessible through new units. Another way would be to make a Roblet® itself into a unit by the Roblet®-server. Then other Roblets® on the same server could access the new unit while the Roblet® is present on the server. This would allow extending the functionality provided by a server over the network without the need to restart the server. The software of the Roblet®-server stays small which mostly results in better maintainability.

This emphasizes that with modules and units a hardware abstraction layer is introduced that can be easily extended. It provides facilities for generalization as well as specialization. In addition, versioning can be introduced by defining new units to change or extend existing programming interfaces to the underlying hardware. For this reason, already existing client applications do not need to be modified while the modules still provide the older units.

A software architecture for service robots based upon the Roblet®-Technology is a powerful medium for robots. The feature of running client programs as a distributed software offers the possibility to run algorithms which need great computation power on different machines which provide this power. The execution of Roblets® on remote systems can also reduce the transferred data as it was shown in the third example. This is important if only limited bandwidth is available like in wireless networks.

The focus of the current work clearly is on task-level programming and monitoring for the all-day operation of robots. This high-level programming of service robots relieves the developer of special knowledge about robots. If the developer is familiar with Java™ programming it is easy to build applications using a service robot's capabilities. Several successful student projects already confirmed this thesis. The experimental scenarios of section 5 give some insight into the variety of possible robotic applications based on the Roblet®-Framework. They show that Roblet®-Technology divides the whole complex distributed system into small maintainable parts.

The next step will be to implement further software to improve the usability of the robot system and create a toolbox of reusable program parts. In this step the variety of the high-level functions like object grasping and multimodal interaction will be increased. First results for this are shown in [13] and [14]. Furthermore, the possibilities of autonomous navigation and map building will be extended.

Finally, the presented framework is not limited to robotics. The technology is usable for many other distributed scenarios. This will be investigated in future work and will allow an in-depth analysis of the framework.

References

[1] D. Westhoff, H. Stanek, T. Scherer, J. Zhang, A. Knoll: *A flexible framework for task-oriented programming of service robots*, Robotik 2004, VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, VDI-Berichte (ISBN 3-18-091841-1), Munich, Germany, 2004.

[2] D. Westhoff, J. Zhang, H. Stanek, T. Scherer, A. Knoll: *Mobile Manipulatoren und ihre aufgabenorientierte Programmierung*, atp - Automatisierungstechnische Praxis 10/2004, Oldenbourg Industrieverlag GmbH, Munich, Germany, 2004. ISSN 0178-2320

[3] *Workshop on Robot Middleware towards Standards*, International Conference on Intelligent Robots and System (IROS'04), Sendai, Japan, 2004, http://www.is.aist.go.jp/rt/events/20040928IROS.html.

[4] H. Bruyninckx: *Open robot control software: the OROCOS project*, Proceedings of the IEEE 2001 International Conference on Robotics and Automation (ICRA'01), volume 3, pages 2523–28, Seoul, Korea, 2001, http://www.orocos.org.

[5] A. Brooks, T. Kaupp, A. Makarenko, A. Orebäck, S. Williams: *Towards Comonent-Based Robotics*, Proceedings ot the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05), Alberta, Canada, 2005.

[6] B.P. Gerkey, R.T. Vaughn, K. Stoy, A. Howard, G.S. Sukhatme, M.J. Mataric: *Most Valuable Player: A Robot Device Server for Distributed Control*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01), pages 1226–1231, Wailea, Hawaii, 2001.

[7] C. Cote, D. Letourneau, F. Michaud, J.-M. Valin, Y. Brousseau, C. Raievsky, M. Lemay, V. Tran: *Code Reusability Tools for Programming Mobile Robots*, Proceedings of the 2004 IEEE/RSJ International Conference on Intellige Robots and Systems (IROS'04), pages 1820–1825, Senda, Japan, 2004.

[8] N. Karlsson, M.E. Munich, L. Goncalves, J. Ostrowski, E. Di Bernado, P. Pirjanian: *Core Tehnologies for Service Robotics*, Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04), Senda, Japan, 2004.

[9] T. Scherer: *A mobile service robot for automisation of sample taking and sample management in a biotechnological pilot laboratory*, University of Bielefeld, Ph.D. Thesis, 2005, http://bieson.ub.uni-bielefeld.de/volltexte/2005/775/.

[10] V. Hayward, J. Lloyd: *RCCL User's Guide*, McGill University, Montreal, Quebec, Canada, 1984.

[11] W.T. Townsend: *The BarretHand Grasper - programmably flexible part handling and assembly*, Industral Robot: An international Journal, Vol. 27, Nr. 3, pp.181-188, 2000.

[12] The Real-Time Java™ Expert Group: *The Real-Time Specification for Java* (RTSJ), 2002, http://rtsj.dev.java.net.

[13] M. Hüser, T. Baier, J. Zhang*: Learning of demonstrated Grasping Skills by stereoscopic tracking of human hand configuration*, To Appear, IEEE International Conference on Robotics and Automation, Orlando, Florida, May 2006.

[14] T. Baier , M. Hüser, D. Westhoff, J. Zhang: *Multimodal Learning of demonstrated grasping skills for flexibly handling grasped objects*. Robotik 2006, VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, Munich, Germany, May 2006

Bibliography

Daniel Westhoff received the diploma degree from the Faculty of Technology at the University of Bielefeld, Germany, in June 2002. He worked as a scientific assistant in the group "Technical Computer Science" of the Faculty of Technology till September 2002. Since October 2002 he is a scientific assistant at the group "Technical Aspects of Multimodal Systems (TAMS)" at the Department of Computer Science at the University of Hamburg, Germany. Currently he is working on his PhD thesis on robust multimodal self-localization for mobile robots.

Hagen Stanek is the initiator of the *genRob-project* and designer of the Roblet®-Framework. His fields of activity include architecture and software development for complex distributed systems with a focus on mobile robotics. He is treasurer of the *Java Users Group Stuttgart e. V.* and active in the Java™ environment. As a member of the Fraunhofer IPA and at University he developed a six-legged walking robot and has long experiences in the field of autonomous robots.

Jianwei Zhang is a full professor and director of the Institute of Technical Aspects of Multimodal Systems, Department of Computer Science, University of Hamburg, Germany. He received his Bachelor (1986) and Master degree (1989) from the Department of Computer Science of Tsinghua University, China, and his PhD (1994) from the Department of Computer Science, University of Karlsruhe, Germany. His research interests include multimodal information processing, robot learning, service robots and human-robot communication. In these areas he has published over 100 journal and conference papers, five book chapters and two research monographs. He leads numerous basic research and application projects, including the EU basic research programs and the Collaborative Research Centre supported by the German Research Council. He has received several awards including the IEEE ROMAN Best Paper Award 2002. He is the coordinator of the Hamburg-Tsinghua International Research Training Group CINACS.