A flexible framework for task-oriented programming of service robots

D. Westhoff, University of Hamburg, **T. Scherer**, University of Bielefeld, **H. Stanek**, genRob.com, Sindelfingen, **J. Zhang**, University of Hamburg, **A. Knoll**, TU Munich

Abstract

In this paper we present a technology that establishes a framework for task-oriented programming of mobile robot systems. The framework allows writing distributed control or monitoring programs for easy adoption of robots to specific tasks. It enables the programmer to send programs referred to as Roblets® to a Roblet®-server running on the robot. Contrary to other distributed system frameworks these Roblets® consist of data and code. The Roblet®-server executes the Roblets® with well-defined behaviour, even in case of malfunctions. The framework hides all network details from the programmer, so that writing programs on a local computer is similar to working directly with the remote robot. This decreases the development time of programs controlling the robots. The framework is implemented in Java™ and tested with two service robots.

1. Introduction

Novel applications in robotics involve distributed systems and employ inhomogeneous hardware environments. Internet technology enables geographically spread groups of scientists to cooperate like in the mars pathfinder mission [2]. Consequently a framework for distributed robot applications must abstract from dedicated hardware. The objective of such a middleware is to shorten the time to install and develop complex distributed applications. This kind of framework relieves the development of systems as summarized in [4].

The classical approach when sharing resources through a network is to utilize a client-server architecture. The servers admit abstraction of hardware. Clients are used e.g. for monitoring, planning and controlling of operation as well as interaction with the user.

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 gestures. 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.

The framework proposed in this paper hides the network and the utilized hardware. Nevertheless more experienced developers are granted access to the details and parameters of the employed hardware. The concept of the presented framework is implemented in Java™ and tested with two service robot systems.

Section 2 in detail discusses the various problems that have to be taken into account when designing a framework for task-level robot programming. Section 3 presents our new concept that solves many of these problems. The concept is based on a technology referred to as Roblets®, which are programs a client sends to a server on a robot. Additionally, reasons for the implementation in Java™ are given. Section 4 motivates the need for the framework by analysing the two exemplary service robots shown in figure 4. Finally section 5 summarises the merits and demerits of the Roblet® technology.

2. Design aspects for mobile service robot applications

Traditionally, when developing service robot applications, the programmer chooses a dedicated system architecture to couple the diverse hardware systems like mobile platforms, manipulators or sensors. The computers used in an arbitrary scenario run different operating systems. The following subsections discuss some of the problems of service robot systems.

2.1 Mobile service robots

The main contradiction a programmer of a service robot has to deal with is the trade-off between easy operability and full flexible utilization.

Easy operability means the system should be easily programmable by non-experts. To allow this, current research investigates man-machine interfaces for natural instruction and communication. Unfortunately, most of these interfaces are only available as prototypes. Robust implementations of these techniques will not be available for some years to come.

In the meantime one compromise is to use script languages as in [6]. These languages are easy to learn and reduce the amount of knowledge needed to operate the robot. They abstract low-level details of the robot control and provide mid-level functionality. This mid-level functionality is then used to implement high-level task-oriented applications.

On the other hand script languages have serious limitations. They only allow alteration of certain parameters and sequences up to a certain point. Modifications or new tasks can only be implemented if they do not require more functionality than offered by the script language. Anything that is not possible within the functionality of the mid-level script language is therefore not possible at the task-oriented level. Such modifications would require access to low level details, but are intentionally hidden by the script languages.
As a general conclusion a framework for mobile robots should provide an easy-to-learn high-level interface for non-expert personnel, while also providing access to low-level details. This allows adding functionality that is missing in the high-level interface.

2.2 Distributed inhomogeneous systems

Since miscellaneous hardware is used, many operating systems can be present in a service robot scenario. The various systems are coupled via a network to communicate as shown in figure 1, for example by using the Common Object Request Broker Architecture (CORBA) from the Object Management Group (OMG) [5]. An Interface Definition Language (IDL) provides a unique description of the available interfaces between the systems. The interfaces are implemented for each system individually.

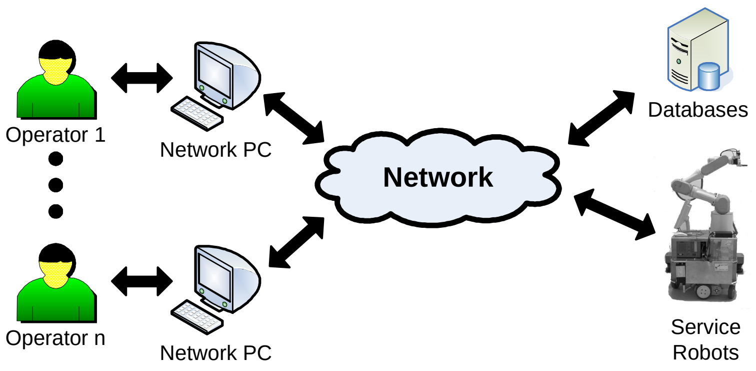

Figure 1: General scenario of a telerobotic system including clients, servers and robots.

For example, [3] present a distributed RT-CORBA system for low-level teleoperation of a robot. A server runs on the robot and answers commands sent to it from the clients to move the robot or read sensor information. The asynchronous method invocation (AMI) feature of RT-CORBA is used to try and ensure real-time behaviour [1]. The explicit purpose of such a teleoperation system is to transmit time-critical data over a network. However, only a minority of applications explicitly requires the server and the client to be running on separate machines. On the contrary, other applications may require a closer coupling to sensor information than can be achieved by any network. These applications will benefit if they are executed directly on the robot. Since only the bare minimum of data is transmitted network bandwidth is saved, which is important for wireless networks used with mobile robots. Although the above can be achieved with current technology, it requires a direct access to the robot and the possibility to restart the robot application each time a minor detail has to be changed, thus complicating the development and debugging from a local machine.

2.3 Task-oriented programming

Many service robot systems are developed for a specific task like mail delivery, hospital service or laboratory services. In order to keep the software maintainable the low-level details of the system are hidden by a hardware abstraction layer (HAL). The task-level programs implementing the service are then based on this HAL. Using the robot for a different service often can not be done by simply writing a new task-level program, but requires additional low-level changes as well. In existing systems this cannot be done while the robot operates.

A framework that allows easy task-oriented programming should provide the high-level functionality as well as access to parts of the low-level architecture because otherwise adding new functionality to perform new or even only slightly changed tasks is not possible.

3. Roblets®

Based on the above considerations we propose a framework to build task-level applications for mobile service robots that solves the mentioned problems. The concept of the framework is realised with Java™.

3.1 Basic Roblet® technology

The basic principle of the framework is to send Roblets® to a server on a remote robot. The server then executes the Roblets® with well-defined behaviour in case of malfunctions. The simplest possible setup has one user application, one Roblet® and one server application called Roblet®-server as shown in figure 2. 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. In addition a Roblet® can be terminated by a client application or the Roblet®-server. After a Roblet® terminates, the Roblet®-server resets itself to a well defined state.

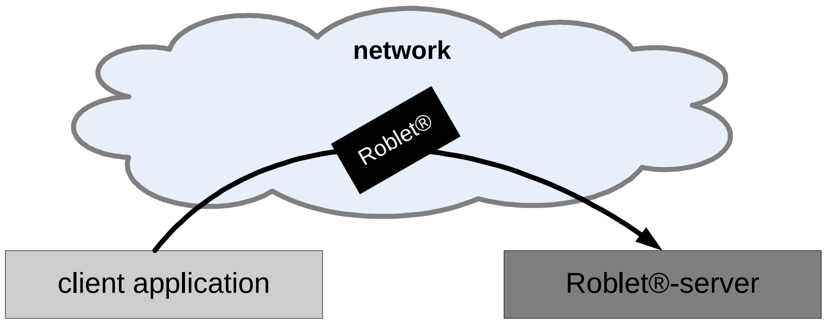

Figure 2: A client application sends Roblets® to a Roblet®-server for remote execution.

3.2 Implementation of the Roblet® framework

The above concept is implemented using Java™ and following a strict object-orientated approach. Java™ allows running the same binary program on various operating systems and hardware platforms. This is based on the concept of the Java™ Virtual Machine (JVM) available for many operating systems. The bytecode a Java™ program is compiled to can be interpreted by every JVM independently of the operating system.

Roblets® are transferred to the servers using Remote Method Invocation (RMI) available in Java™. If the execution of a Roblet® finishes, a single object containing results of the execution can be returned. In the case of an execution error an exception is returned. The fact that this exception occurred on a remote machine as well as many other network aspects are hidden from the client by RMI.

Often, software for various hardware devices already exists. In many cases a Roblet®-server is therefore implemented as a wrapper around existing software programs or libraries using the Java™ Native Interface (JNI). In general, this means that only higher-level functionality is accessible through the Roblet®-server. This is shown in figure 3. To increase flexibility, lower-level functions and parameters can additionally be made available where possible.

Contrary to CORBA and its IDL the implementation in Java™ does not need a separation between the interface definition and the code of the implementation. In Java™ the use of Java™ interfaces and classes is sufficient.

Jini™ provides a directory service that lets a client automatically find the available servers.

The Roblet® technology is comparable to Java™ applets, but has some important differences. While applets are sent to a client from the server, Roblets® are sent to the server from a client. Applets always run in a specific environment which abstracts a graphical user interface. In contrast, Roblet®-servers will differ from each other depending on the underlying hardware as will be explained in the following sections.

3.3 Roblet®-server

A Roblet®-server provides units which constitute the functionality of the server. A Roblet® queries a server about its units. Special units are written for special hardware and more generic units are defined for similar hardware, e.g. distance measurement sensors.

Only the definitions of the units must be known to the developer of a Roblet®. The developer of a Roblet®-server has to implement the unit according to these definitions.

The scope of Roblet®-servers can vary over a wide range: A Roblet®-server can abstract special hardware and provide only a few special units or may be built to represent a complete mobile service robot and its functionality through many different units.

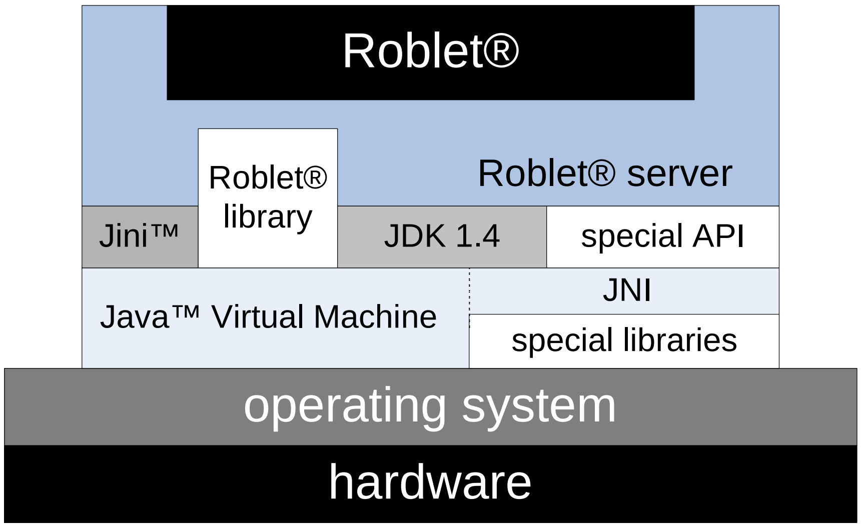

Figure 3: System architecture for a Roblet® server.

3.4 Extended Features

Besides the basic features of the Roblet® framework more advanced features can be used for the distributed control of service robots on task level.

Simply returning a single object upon Roblet® termination may be insufficient to build a more complex distributed system. Roblets® can establish additional connections using any network communication technology available in Java™ like RMI or TCP/IP.

Furthermore, a Roblet®-server can run multiple Roblets® in parallel. Additionally, a Roblet® can be made persistent and continue to run when the sending application terminates. Moreover, in scenarios where the robot is not directly accessible like in hazardous environments, operators can remotely change its assignment to a new task. In order to achieve that a new Roblet® has to be sent to adjust the robot to the altered situation.

4. Applications

The Java™ implementation of the framework has been tested with the two mobile robot systems shown in figure 4. The first system is a mobile service robot at the University of Bielefeld built to automate the sample management in biotechnological cell cultivations [6]. The script language used in this system allows minor changes like that of the position of devices to be operated or the motions required to do this. It also allows adding new devices if they can be operated with motion primitives that already exist. Modifications that go beyond what the script language supports are not possible, but require the server program itself to be changed and restarted. Therefore parts of the control system for the mobile platform are encapsulated by a Roblet®-server. Currently a Roblet®-based system is used to monitor and evaluate the mobile platform while moving in the laboratory. In the future the monitoring of the manipulator of the service robot and a substitution for the script language will be added. The second robot used at the University of Hamburg is very similar to the above mentioned system [7] exept that it uses more sensors. One of the sensors made available via Roblets®-servers are digital cameras. Nevertheless, the adoption of the existing Roblet® applications was easy and only minor changes were needed. This proves the feasibility of the developed approach.

5. Conclusion

The proposed framework has proved to be useful for many reasons. Its implementation is simple and takes advantage of different technologies available in Java™ like Jini™, RMI or JNI. The platform-independence of Java™ eliminates the need to port applications to different platforms which can be found in distributed robotics.

The functionality of the provided services can be expanded easily by defining new specific units. In addition, Roblets® extend the interfaces to each application individually. Already existing applications do not need to be modified if the new units are not needed.

The execution of the Roblet® on the remote system can reduce the transferred data which 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.

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 the analysis of the framework.

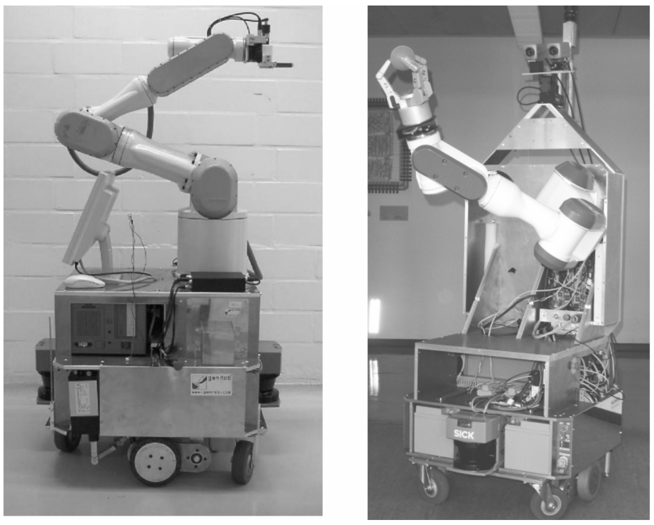

Figure 4: Mobile service robots of the Institute of Cell Culture Technology at the University of Bielefeld (left) and of the Group Technical Aspects of Multimodal Systems at the University of Hamburg (right).

References

[1] Amoretti, M.; Bottazzi, S.; Reggiani, M.; Caselli, S.: *„Evaluation of Data Distribution Techniques in a CORBA-based Telerobotic System“*, Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS 2003, pp. 1100-1105), Las Vegas, Nevada, 2003.

[2] Backes, P. G.; Tso, K. S.; Tharp, G. K.: *„Mars Pathfinder Mission Internet-Based Operations Using WITS“*, Proceedings of the 1998 IEEE International Conference on Robotics & Automation (ICRA 1998, pp. 284-291), Leuven, Belgium, 1998.

[3] Bottazzi, S.;Caselli, S.; Reggiani, M.; Amoretti, M.: *„A Software Framework based on Real-Time CORBA for Telerobotic Systems“*, Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS 2002, pp. 3011-3017), EPFL, Lausanne, Switzerland, 2002.

[4] Goldberg, K.; Siegwart, R., edtitors: *„Beyond Webcams: an Introduction to Online Robots“*, The MIT Press, 2001.

[5] Object Management Group: *„Common Object Request Broker Architecture (CORBA/IIOP)“*, http://www.omg.org/technology/documents/corba_spec_catalog.htm

[6] Scherer, T.; Poggendorf, I.; Schneider, A.; Westhoff, D.; Zhang, J.; Lütkemeyer, J.; Lehmann, J.; Knoll, A.: *„A Service Robot for Automating the Sample Managment in Biotechnological Cell Cultivations“*, Proceedings of the IEEE Converence on Emerging Technologies and Factory Automation (ETFA 2003, Vol. 2, p. 383-390), Lisbon, Portugal, 2003.

[7] University of Hamburg, Faculty of Informatics, AB Technical Aspects of Multimodal Systems: http://tams-www.informatik.uni-hamburg.de