"Verification of Skills for Task Execution in Service-Robotics"

Overview

Service robots, or especially rehabilitation robots (e.g. FRIEND II), are designed to support disabled and elderly people to perform ADL-tasks (ADL = activities of daily living), when they are normally dependent upon care-personnel for 24 hours a day. The goal is to provide them with approximately 1.5 hours of complete autonomy. A large variety of robot-based tasks could be performed within this time span: Prepare drinks / meals and serve them to the user, tidy up, operate drawers, cupboards, switches, provide the user with reading or other entertaining materials.

For this purpose, a new framework for the semi-autonomous task execution with service robots has been developed at the IAT [Mar05, Mar05a] - its implementation and enhancements are in progress [Pre05]. In the following the basic idea of task specification and execution within the framework is illustrated:

Task-Specification and Execution on Base of Process-Structures

The elementary process structures are compiled as Petri-Nets directly so far. The Petri-Net corresponding to the depicted function block net is as follows:

The elementary process structures are compiled as Petri-Nets directly so far. The Petri-Net corresponding to the depicted function block net is as follows:
The execution of a skill in the skill layer, e.g. the skill “MoveAndAdjustGripperToObject()” is equivalent to the execution of a C++ method of the respective skill-server (here: ManipulatorSkillServer). The verification of the elementary process-structures already guarantees the

- exclusion of resource conflicts
- availability of data
- reachability of target state
- exclusion of modeling errors.

But from the viewpoint of this layer certain assumptions are made in consideration to the level of the C++ method representing a certain skill. So far, no automatic verification of the correctness of skill-programming is available.

Goal

Within this work a method for the verification of correct system behavior on skill-level shall be developed. This method is not limited but should include the verification of

1. correct usage of specified parameters
2. usage of only specified resources
3. correct return values
4. availability of data from the database accessed by the skill additionally to that one specified in the parameter list
5. timing and correct communication during cooperation with other skills
6. exception raising infrastructure
7. correct and complete application of conventions for skill programming, like
   a. sending of call-backs
b. non-excessive CPU power consumption in waiting loops

c. providing of upper-layer communication (ability to abort or re-
   parameterize a skill)

d. checking for availability of CORBA-proxies

e. logging

f. deletion of temporary data from the world model

In general it is distinguished between constructive and analytical quality assurance in software engineering [Eng03]. The first one can be achieved through automated generation of implementations from high-level models or a definition language of higher abstraction. The analytical methods include verification and validation. A variety of formal techniques is applied here (i.e. model checking, usage of Petri-nets or temporal logic, …).

First a literature research related to the existing methods of verification of autonomous system’s behavior is necessary. These key methods may be useful for the start of the research:

- Conventional or timed Petri-Nets
- Net Condition Event Systems, Signal/Net-Systems
- Temporal Logic, Computation Tree Logic, Model Checking

It shall be analyzed whether one of the available mathematical methods are useful for their application for verification purposes in this context here (e.g. to model behavior in time, communication or exception infrastructure).

From these starting points the method to fulfill the above stated requirements (or additional requirements?) shall be developed.

Procedure

- Become familiar with the framework by reading the listed literature / documentation.
- Develop the methods for verification, use the skill MoveAndAdjustGripperToObject() from the Manipulator Skill Server as sample skill to start with. It is furthermore suggested to start with verification steps like 1, 2, 3 and 7 for reasons of simplicity and introduction to the topics. It is important to develop flexible methods of verification that can be enhanced or exchanged in the future. This flexibility in verification can be best achieved with the help of formal specification of models or templates with the help of the above cited methods.
- Design the necessary classes (UML) and implement them. Apply your classes in testing scenarios and document the functional correctness of your implementations.
- Documentation of the project, presentation.

Hints

- Documentation has to take place in parallel to the project and is (beside regular meetings) an additional feedback to the supervisor.
- SW in the FRIEND-II-project has to be developed in C++ (therefore, good knowledge of object oriented programming in C++ is necessary [Str00]). The SW-design has to be done with the help of UML [UmlWP]. Furthermore, the doxygen-SW [DoxWP] is used during the design- and documentation phase.
- The IAT-programming guidelines [Mar02] have to be respected; otherwise the implementations are hardly useful in the FRIEND project.
Literature & Links


