

**SIXTH FRAMEWORK PROGRAMME
PRIORITY 2**



Information Society
Technologies

Specific Targeted Research Project

ROBOT@CWE

Advanced robotic systems in future collaborative working environments

Contract Number 034002

**Deliverable 3.8@M36:
Final report on software concepts for ROBOT@CWE**

Version	1.0
Status	Final version
Date	31-10-2009

Deliverable Administration & Summary					
Deliverable	No.	Name			
	D3.8	<i>Final report on software concepts for ROBOT@CWE</i>			
Workpackage	No.	Name			
	WP3	<i>Robotic – CWE interfaces and evaluation of impacts</i>			
Task	No.	Name	Description from DOW		
	T3.5	<i>Software support for ROBOT@CWE</i>	<p>ROBOT@WORK will adopt a service-oriented-architecture approach. Architectures are generally described in terms of components (computational elements), connectors (interaction elements), and their configurations. An architectural style further defines a vocabulary of component and connector types as well as a set of constraints on combining instances of those types in a software system.</p> <p>Selecting an appropriate architectural style is a key determinant of a software system's success. Software architectures provide design-level models and guidelines for composing software systems. However, to be useful in a development setting, these models and guidelines require support for implementation and evolution. Therefore, in the ROBOT@CWE project we will focus on the design, implementation, and empirical evaluation of techniques for supporting architecture-based software development using the concept of "Open Abstract Framework" (OAF). Several aspects of OAF-SOA development (service component-based system composition, explicit software connectors, architectural styles, upstream system analysis and simulation, and support for dynamism) make it a good fit for ROBOT@CWE demands.</p>		
Status	Final	Due	M36	Date	2009-10-29
Author(s)	Lorenzo Blasi, HP; Astrid Weiss, PLUS; Olivier Stasse, CNRS; Daniel J.Hernandez, UC3M				
Editor	Lorenzo Blasi, HP				
Comments					
Document workflow					
Draft version	Lorenzo Blasi, HP			Date	2009-07-24
First revision	Lorenzo Blasi, HP			Date	2009-10-12
Submitted	Abderrahmane Kheddar, CNRS			Date	2009-10-31
Document history					
Version	Date	Editor(s)		Description	

0.1	2009-07-24	L. Blasi	ToC and Organization of the work
0.2	2009-10-12	L. Blasi	Integrated contributions from PLUS, CNRS, UC3M, HP
1.0	2009-10-27	L. Blasi	Integrated comments provided by internal reviewers
1.0	2009-10-31	A. Kheddar	Reading, small typos correction, final checks

Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

TABLE OF CONTENTS

1.	Executive summary	6
2.	Iterative Development of the Human-Robot Interface (HRI)	7
2.1.	Iterative interface design	7
2.2.	First Expert Evaluation.....	8
2.3.	Second Expert Evaluation	17
2.4.	Description of the new HRI	26
3.	Interacting with HOAP-3 robot.....	29
3.1.	Second year demonstrator	29
3.2.	Final demonstrator: collaboration with Robot@Moon	31
4.	Controlling an avatar in a 3D Virtual World.....	35
4.1.	OpenSimulator	35
4.2.	Integration between OpenSimulator and HRI	37
5.	Interacting with HRP-2 robot	40
5.1.	Second year demonstrator	40
5.2.	Final demonstrator: collaboration with Robot@Construction-site	40
6.	Conclusions	45
7.	References	46
8.	Appendix	48
8.1.	Goal-setting / approaching an object.....	48
8.2.	Grabbing/dropping an object.....	49
8.3.	Strategy selection request.....	50
8.4.	Strategy indication.....	50
8.5.	Using an object.....	51

TABLE OF FIGURES

Figure 1 - First design mock-up which was evaluated by means of a heuristic evaluation	8
Figure 2 - Resulting design recommendations from the heuristic evaluation.....	15
Figure 3 - Interface which was evaluated in the second expert iteration	19
Figure 4 - HRI interface and main components	27
Figure 5 - a) the recreation of lunar scenario. b) HOAP-3 teleoperated through a corridor looking for the 'antenna'	29
Figure 6 - Global overview of the 2nd year demonstrator architecture	30
Figure 7 - Robot HOAP grasping the 'antenna' a) the robot computes grasping trajectories and receives a command by the operator. b) robot successfully picks up the 'antenna'	31
Figure 8 - 3rd year demonstrator. Operator teaching skill to robot (EPFL). HOAP-3 robot walking on the moon scenario (UC3M).....	31
Figure 9 - Global overview of the 3rd year demonstrator architecture	32
Figure 10 - HOAP-3 software server architecture	33
Figure 11 - Some screenshots from HP's OpenSimulator virtual world	36
Figure 12 - Creating an object with Hippo OpenSim Viewer.....	37
Figure 13 - Architecture for the HRI to OpenSimulator integration.....	38
Figure 14 - Generic REST command request/reply protocol.....	39
Figure 15 - Data and control flows of the software architecture.....	40
Figure 16 - Data and control flows of the final demonstrator.....	42
Figure 17 - The first version of an OpenSim environment for BSCW integration.....	43

1. Executive summary

This document reports on software created by partners of the project for supporting the cooperation between humans and robots. During the lifecycle of the project several software components have been created and integrated both for the purpose of adding specific functionalities to existing platforms and creating new ones. One of those components has been selected as the unifying thread of the whole deliverable: the HRI user interface created by HP, but most of the other important components created by other partners stand out in the discussion and are described in the deliverable. Another common thread is represented by collaboration, shown between humans and robots in both demonstrators and even between two robots, in the Robot@Moon final demonstrator. Explored software CWEs are BSCW, in collaboration with eCoSpace, the virtual world created in OpenSimulator and Skype.

The first section describes the HRI and reports the process and results of two usability evaluations, which greatly influenced its final graphical aspect and functionalities.

The second section reports about the demonstrators contributed by UC3M for the 2nd and 3rd year of the project. These demonstrators show the HOAP-3 robotic platform in a simulated space environment interacting and collaborating with a human operator through the HRI and with another HOAP-3 platform through a Shared Knowledge Database.

The third section introduces a simulation environment named OpenSimulator used both for testing the HRI and for collaborating with the HRP-2 robotic platform through the BSCW Collaboration Working Environment. In the OpenSimulator virtual world an avatar can be controlled through the HRI using the Robot Command Protocol already described in deliverable D3.3. The integration between OpenSimulator and BSCW allows assigning a task to the HRP-2 robot.

In the last section the various components integrated with the HRP-2 platform for the 2nd and 3rd year demonstrators are summarized. The final demonstrator is discussed in more detail in deliverable D4.1-3.

Two main WP3 objectives are addressed in this deliverable: “usability and user experience aspects” and “Software architecture support for HRI and ROBOT@CWE integration”. Usability is the key objective addressed in the first section (Iterative Development of the Human-Robot Interface), but also the work described in the third section (Controlling an avatar in a 3D virtual world) has been useful in supporting usability testing. The second objective, software architecture, is another common thread which is present in all sections of this document, from the HRI and RCP extensions described in the first section, to the OpenSimulator-HRI integration described in section three to the architectures for the two demonstrators described in sections two and four.

2. Iterative Development of the Human-Robot Interface (HRI)

One goal of the Robot@CWE project was to develop a Human-Robot Interface (HRI) that offers an intuitive usage for *naïve* users, because it should be used not only by experts but also by normal casual users. To reach this goal the software development for this interface was accompanied by an iterative design process. The following chapter presents this design process, which consisted of two expert-based evolutions. The final proof-of-concept of the iteratively designed HRI was finally tested in the final demonstrator evaluation study.

2.1. Iterative interface design

Iterative development can help to identify problems in the interface in a very early phase of the design, which is important because problems found in an advanced stage of the development can be time consuming and thereby very expensive to solve [1]. Furthermore, it is impossible to design a user interface without usability problems from the beginning, even for usability experts. Therefore, iterative design helps to improve an interface. At each stage of the interface design, a usability evaluation is conducted and identified, and usability problems are reported. These usability problems should be fixed in the next iteration of the interface, in order to improve usability.

2.1.1 Related Work

Different approaches have been tried in order to tele-control a robot. Using a joystick was one of the first approaches, as the majority of people were familiar with its use. In a study, Song et al. [2] reported that a joystick for controlling a robot is most efficient. Other devices like button interfaces are valuable when the task was specified to rotate a robot. The main problem using a joystick is that it is difficult to apply to any other robot platforms straightforwardly. Other input modalities researchers are still trying to improve are speech input (most natural way to communicate) and gesture control [3]. For instance, the HRP-2 robot is able to compute speech input, early shown by [4] but there are still problems occurring as the robot is not able to understand the speech command caused by bad pronunciation or by noise coming from the environment. Further details on these issues are presented in D2.8-2.9. To control a physical robot, Micire et al. [5] developed a multi touch interface. In their approach they tried to get the benefits of usability from a joystick and button device interface by putting them together. Some parts of the interface showed good usability, but one major problem was the lack of feedback during the interaction, so the user could not understand the general interaction paradigm. Other interfaces [6,7,8] were developed but needed excessive training in order to control them efficiently. All in all, the majority of the developed interfaces are difficult to use, even when controlling the basic movements of a robot. Therefore, there is a need of developing an interface where the usability problems are fixed before the interface is being released. To develop an easy and usable interface, an iterative design approach was chosen.

2.2. First Expert Evaluation

This subsection describes the evaluation of the first interface proposal, which was conducted by means of a heuristic evaluation.

2.2.1 First Prototypical Implementation

The first interface proposal which was evaluated by the heuristic evaluation is shown in Figure 1. The integrated robot camera view (1) is an important feature the HRI should enable, since, without it, exact navigation in an unknown environment would be impossible. The basic navigation (2-9) is placed at the right side of the HRI and should allow intuitive and easy use. Therefore the basic control functions supported by the interface are the following:

- Integrated robot camera view (1)
- Move the robot forward (2)
- Move the robot backward (3)
- Step the robot to the site (4,5)
- Decrease/increase speed of the robot (6,7)
- Turn the robot to the left/right side (8,9)
- Stop the robot (10)
- Goal pointing Map (11)¹
- Message log display (12)

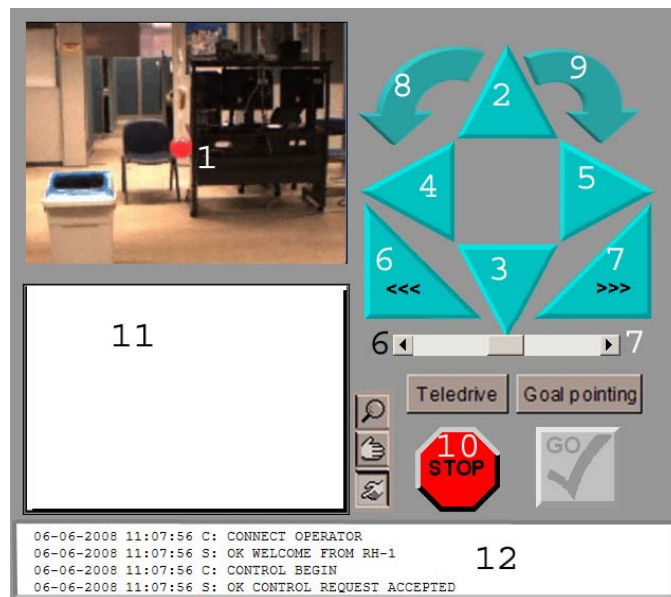


Figure 1 - First design mock-up which was evaluated by means of a heuristic evaluation

Through the message log display the user receives actual feedback about the command performing and how long it lasts. With that kind of feedback the user can determine when a command is finished. All in all, the HRI should be kept simple, in order to be efficient, easy to use, and portable to other systems.

¹ The function goal pointing was not part of the heuristic evaluation

2.2.2 Study Setting

To find usability problems of this first prototypical implementation a heuristic evaluation of the HRI was conducted. The evaluation of the interface was conducted by PLUS at the IT&S Center (University of Salzburg) in June 2008. For the evaluation five Human-Computer Interaction and interaction design experts (3 males and 2 females, age between 20 and 30) were invited, who were all familiar with the procedure of a heuristic evaluation in accordance to [9]. Each expert needed approximately one hour to conduct the usability problem identification.

2.2.2.1 *Instruments*

A heuristic evaluation is a so called “discount usability” technique which should be conducted with usability experts. Three to five expert evaluators are able to find about 75% of the usability problems of an interface and are therefore appropriate enough [9]. In order to categorize the identified problems a list of heuristics is provided, see

Table 2. Each expert evaluator inspects the given interface independently and reports identified usability problems violating these heurists. Then a list of all usability problems, found by the expert evaluators, is compiled and then sent back to the expert evaluators, in order to rate them between 0 - no error - to 4 – a very grave error, which is explained in detail in the following:

- 0 = no usability problem at all
- 1 = cosmetic usability problem, need to be fixed unless extra time is available
- 2 = minor usability problem, fixing should be given low priority
- 3 = major usability problem, important to fix giving high priority
- 4 = usability catastrophe, imperative to fix before the product can be released

The evaluation consisted of two scenarios and followed by of a qualitative interview. Each scenario comprised a stepwise presentation of necessary user actions and system responses. Table 1 shows the two scenarios in detail:

Table 1 Description of scenarios used for heuristic evaluation.

	Scenario I	Scenario II
Task	1) Move the robot by using the HRI 2) Increase the speed 3) Stop the robot	1) Take a look at the navigation and find out what you can do with it
Goal	Use the right buttons in order to move the robot to a desired location.	Find out which activities the navigation supports.
Context	You use the HRI for the first time. Move the robot forward; find out how to increase the speed of it and how to stop the robot.	You use the HRI for the first time and want to find out which possibilities the navigation offers to you.

Table 2 Heuristics from Nielsen [9] used for the Heuristic Evaluation of the remote control

1) Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2) Match between system and the real world	The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3) User control and freedom	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4) Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5) Error prevention	Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
6) Recognition rather than recall	Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7) Flexibility and efficiency of use	Accelerators—unseen by the novice user—may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8) Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9) Help users recognize, diagnose, and recover from errors	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10) Help and documentation	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

2.2.3 Procedure

The evaluation of the HRI started with an introduction of the interface followed by an explanation of the two scenarios, presented as power point slides. Then, the expert evaluators were walking through the scenarios step by step on a tablet PC, documenting all usability problems which could be identified in an excel sheet and classified them according to the

heuristics of Nielsen [9]. At the end of the second scenario an interview about general impressions and design recommendations was conducted. Finally, a list of all usability problems found was created and then sent to the expert evaluators again. The expert evaluators were asked to rate the listed problems on a scale between 0 (no usability problem) and 4 (usability catastrophe).

2.2.4 Results

All in all, 22 usability problems were identified and rated by the expert evaluators. Table 3 shows an overview of the distribution of the violated heuristics. The evaluators marked seven out of ten heuristics of the interface to be violated, whereas heuristic 1, 2, and 7 were violated most often. This can be interpreted as that the interface could not show the system status adequately (heuristic 1), that some parts of the interface could not match between system and the real world (heuristic 2), and that the use of the interface was not flexible and efficient enough (heuristic 7). Furthermore, 2 out of 22 usability problems were indicated as usability catastrophe (severity ranking of 4) and 15 out of 22 as major usability problems (severity ranking of 3). According to [9] problems indicated as 3 or 4 in a heuristic evaluation are imperative to fix.

Table 3 Heuristics used and number of violation per heuristic

Heuristics	Number of Violations
1) Visibility of system status	9
2) Match between system and the real world	9
3) User control and freedom	3
4) Consistency and standards	4
5) Error prevention	1
6) Recognition rather than recall	0
7) Flexibility and efficiency of use	11
8) Aesthetic and minimalist design	2
9) Help users recognize, diagnose, and recover from errors	0
10) Help and documentation	0

In the following the problems identified and rated as 3 or 4 by the expert evaluators are grouped into different problem areas and described in detail:

System Status

- When starting the interface, it is not clear which mode of the two available modes (“Teledrive” or “Joystick”) is selected (4 out of 5, mean expert rating 3.6, standard deviation 0.55).
- There is not enough information about the status of the system (4 out of 5, mean expert rating 3.4, standard deviation 0.89).
- No haptic feedback is possible (1 out of 5, mean expert rating 2.8, standard deviation 0.45)

Labeling

- One of the main problems is that the items are not clearly understandable (5 out of 5, mean expert rating 3.2, standard deviation 0.84). For instance the speed buttons are labeled with <<< (decrease speed) and >>> (speed up).

- Another problem caused by the insufficient labeling is the missing link between the speed buttons, and the slider (5 out of 5, mean expert rating 3.8, standard deviation 0.45).
- It is difficult to interpret the text in the message log display because it is cryptical, for instance “Set speed 80” (5 out of 5, mean expert rating 3.0, standard deviation 1.22).
- The functions of some control items of the interface are not clear, for instance, “Teledrive”, “Goalpointing” or arrow up and down (5 out of 5, mean expert rating 3.4, standard deviation 0.89).
- The buttons are not labeled (5 out of 5, mean expert rating 2.8, standard deviation 1.3).

Structural Problems

- The grouping of the buttons on the interface makes it difficult to understand which button belongs to which function (4 out of 5, mean expert rating 3.2, standard deviation 0.84).
- The map buttons are not very intuitive (4 out of 5, mean expert rating 2.6, standard deviation 1.14).
- The speed presentation and speed control are horizontally aligned; therefore they are difficult to interpret (1 out of 5, mean expert rating 2.6, standard deviation 0.89).

Navigational Problems

- The function of the movement buttons is not clear (2 out of 5, mean expert rating 3.2, standard deviation 1.1).
- The interface does not offer the manipulation of speed and direction at the same time (5 out of 5, mean expert rating 3.2, standard deviation 1.1).
- The buttons “Stop” and “Go” are misleading, as it is not clear what they should be used for (5 out of 5, mean expert rating 3.2, standard deviation 0.84).
- The integrated camera does not deliver enough content of the environment, so it is difficult to navigate. More environmental information could be required (2 out of 5, mean expert rating 2.8, standard deviation 0.84).

Size of the Interface Elements

- Some buttons of the interface are too small (3 out of 5, mean expert rating 3.2, standard deviation 0.84).
- A touch screen is not very suitable for a construction site (2 out of 5, mean expert rating 2.6, standard deviation 0.89).

This subsection presents the usability problems identified which could be classified in the rating scale from 1 and 2. These problems are not imperative to fix.

Minor Usability Problems:

- The 3D effects of the buttons are badly designed (1 out of 5, means expert rating 2.2, standard deviation 1.3).
- Due to the fact that two different modes are available there is no default mode visible at the system start (1 out of 5, mean expert rating 2.0, standard deviation 0.71).
- The use of the “Stop” button in the evaluated scenario is redundant (5 out of 5, mean expert rating 1.8, standard deviation 0.84).

- There is too much information in the message log display which should be reduced (4 out of 5, mean expert rating 1.8, standard deviation 1.1).

Cosmetic Usability Problem:

- If in the goal pointing mode it is not clear whether the user can choose between the “Screen” or “Joystick” (2 out of 5, mean expert rating 1.4, standard deviation 0.55).

2.2.5 Final Interview

After the expert evaluators went through the two scenarios a final interview was conducted. The purpose of the interview was to get the evaluators’ impressions of what was positive or negative about the interface, taking into account general structure, design and learnability. The results of the interview were direct and free answers of the expert evaluators.

General Impression of the HRI

- The expert evaluators complained about the interface, its obscurity, and its confusing design (4 out of 5).

Impressions of the Layout and Design

- The design of the interface is unstructured (4 out of 5).
- Repetitious placement of the slider for increasing the speed of the robot. A first impression of what the slider should be used for was not clear.
- The expert evaluators stated that the “Teledrive” and “Goalpointing” modes should be separated (2 out of 5).

Impression of the Clearness of the Design

- There is not enough feedback in the interface about the state of the system. Moreover, the user is confronted with too much information in the message log display (2 out of 5).

Impression of the Learnability

- The expert evaluators stated that the interface is not very intuitive and the user has to try out the different functions (3 out of 5).
- The learnability is supported due to the fact of the small number of buttons on the interface (2 out of 5).

2.2.6 Design Recommendations

This section presents grouped recommendations and detailed explanations about the new design proposal which was developed based on the previous results from the heuristic evaluation and the final interview. For the new design approach of the HRI, only the ratings of 3 and 4 of the identified usability problems from the heuristic evaluation were considered. The new interface approach is shown in Figure 2, where the proposal was split into three areas considering recommendations of problems in the system status, navigational problems and integrated camera view.

Recommendations for the System status ((1) on Figure 2)

- In order to get a clear feedback on the status of the robot, the information box contains relevant information about the connectivity, speed of the robot, and the amount of battery charge.
- Tabs are placed on top of the interface in order to distinguish easily between the different modes the interface could enable.

Recommendations for the Navigational Problems ((2) on Figure 2)

- Because of the visual difference between the two modes, the problem with the labeling (“Stop” and “Go”) could also be solved. Therefore, the stop button has been moved to the center of the navigation which makes it possible to stop the robot immediately.
- Due to the partition of the available modes, the problem of which button belongs to which function disappeared.
- This kind of navigation offers a direct manipulation of speed and direction at the same time, which was also a problem the expert evaluators reported about.

Recommendations for the Integrated Camera View ((3) on Figure 2)

- Important for the navigation of a robot in unknown environment is the view of the camera. Therefore, the screen and its range of view should be bigger in order to enable adequate navigation.
- Through sliders the camera of the robot can be manipulated up and down via the interface. Furthermore, the user is able to determine through the position of the sliders how far the head of the robot has been moved.

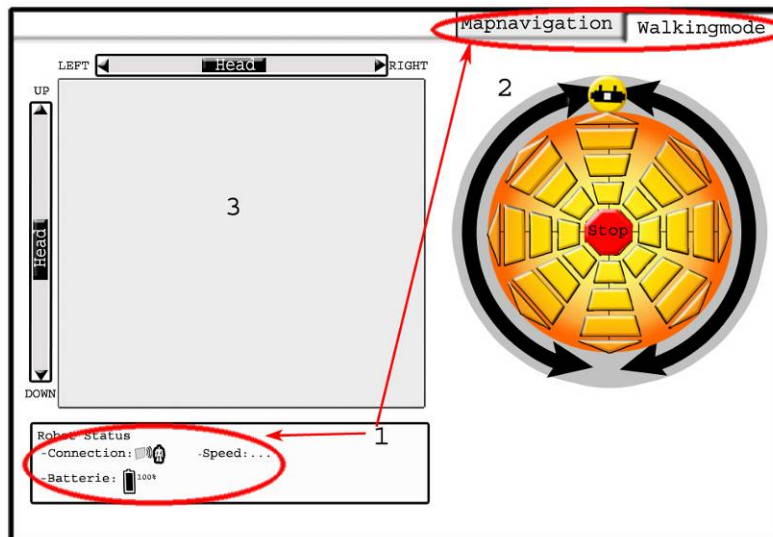


Figure 2 - Resulting design recommendations from the heuristic evaluation

2.2.7 Design implications

Based on the result of the usability study, a second prototype for the remote control of a humanoid robot, namely Fujitsu HOAP-3 robot, has been developed. The recommendations from the study should be confronted with the constraints imposed both by existing tools for the implementation of the user interface and the Robot Command Protocol (RCP) which allows controlling the robot.

The issues identified during the first phases of the prototype implementation are the following. First, the human-robot interface resulting from the heuristic evaluation (it may be called the “ideal human-robot interface”) includes some non-standard visual components hard to find in the classic GUI toolkits/frameworks, for example a circular slider controller. The second issue to cope with is the dynamic configurability of the controls, whose parameters depend on the abilities of the connected robot. For example, the circular component which controls both direction and speed of the robot should adapt its precision and range to the

rotation capability of the connected robot. Another issue is about asynchronous bidirectional communication. The human-robot interface should keep track of the robot's status and modify itself in order to avoid not permitted or dangerous command sequences. This behaviour requires that the HRI and its components should be as much configurable and flexible as possible.

The implemented HRI is shown in

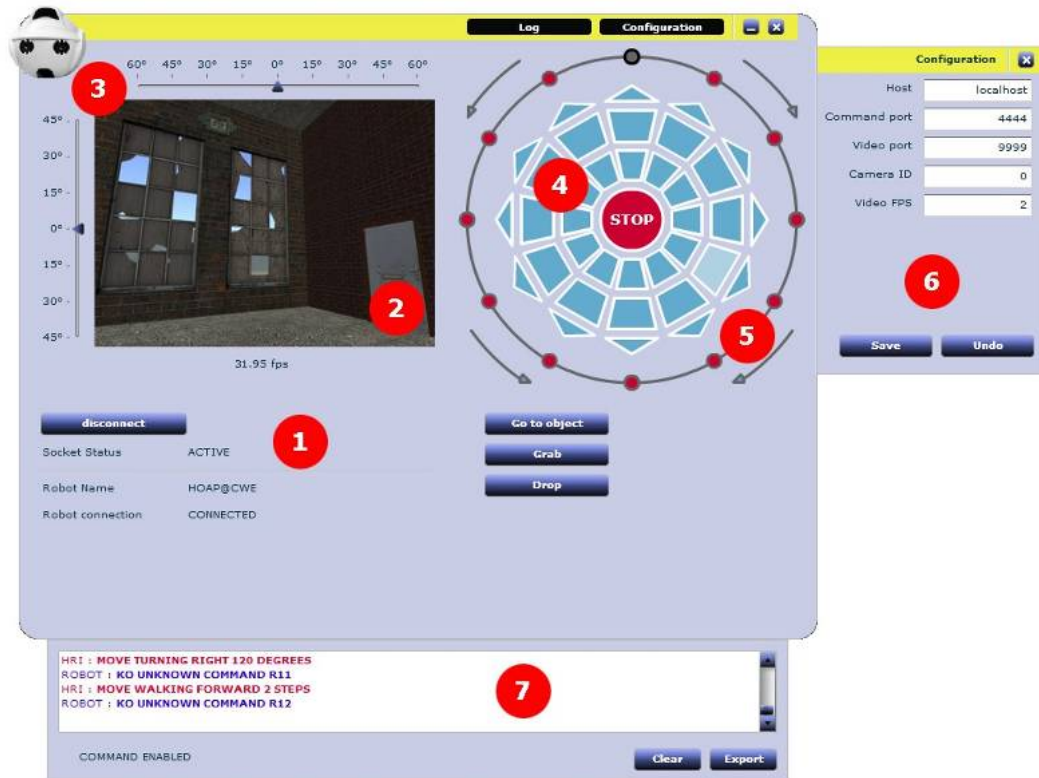
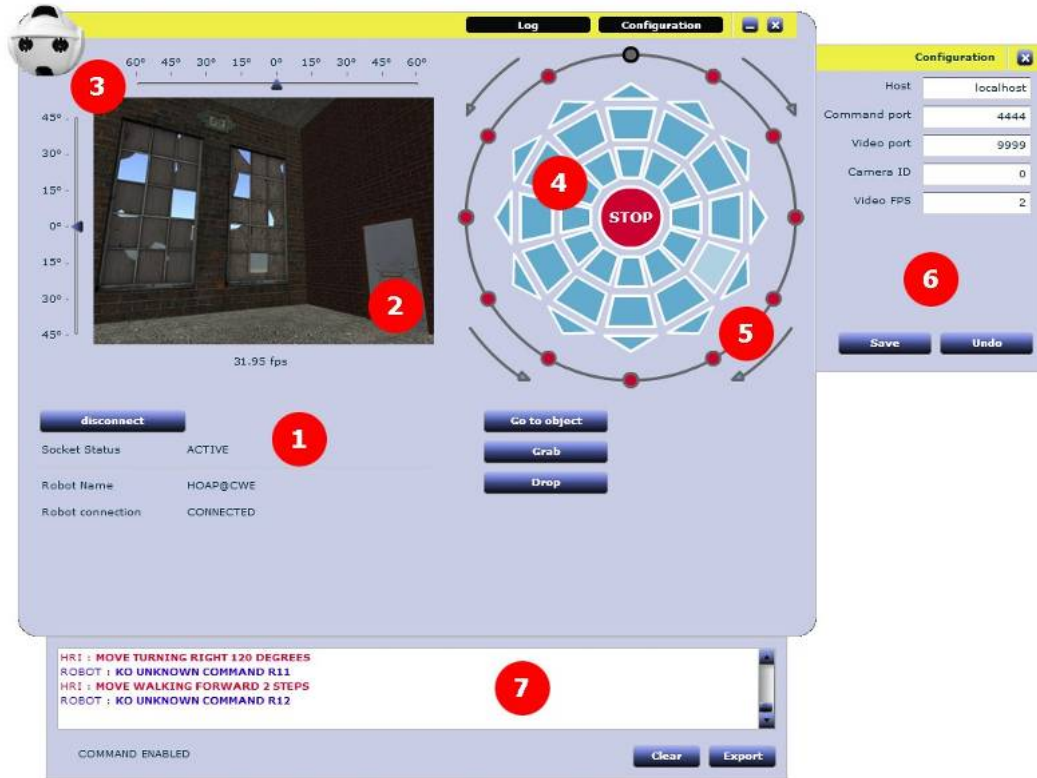


Figure 3. The implementation of the prototype followed an incremental approach based on a flexible plug-in-based architecture in which new user interface modules can be added to support new protocol commands. According to the recommendation the interface should have two tabs, one for each of the two modes “Goal pointing” and “Teledrive”. In the current prototype only the Teledrive mode has been implemented and consists of the following user interface modules. In the information box (1) the only available information is the robot's name and the status of the connection. The integrated robot camera view (2) shows the video stream coming from one of the cameras installed on the robot. Slide bars (3) for moving the robot's head are graduated and during slider movement a pop-up balloon indicates the exact amount of degrees of the rotation command. The speed and direction control (4) has been designed as a custom control with the possibility to enable only the range which is supported by the robot. The circular slider has been substituted with a circle of equispaced buttons (5), each of which commanding the robot to rotate about the corresponding angle. Configuration information (6) can be modified and saved through a separate window, which the operator can show and hide at will. The messaging control (7) is in a separate container, and the operator can choose to keep it open or to hide it.

2.3. Second Expert Evaluation

Based on the implementations of the results from the first evaluation, a second evaluation was set up



(Figure 3 shows a screenshot of that interface). Not all suggestions from the first iteration could be taken into account due to limitations of standard GUI components, for instance the circular slider component of

Figure 2 was then implemented as buttons

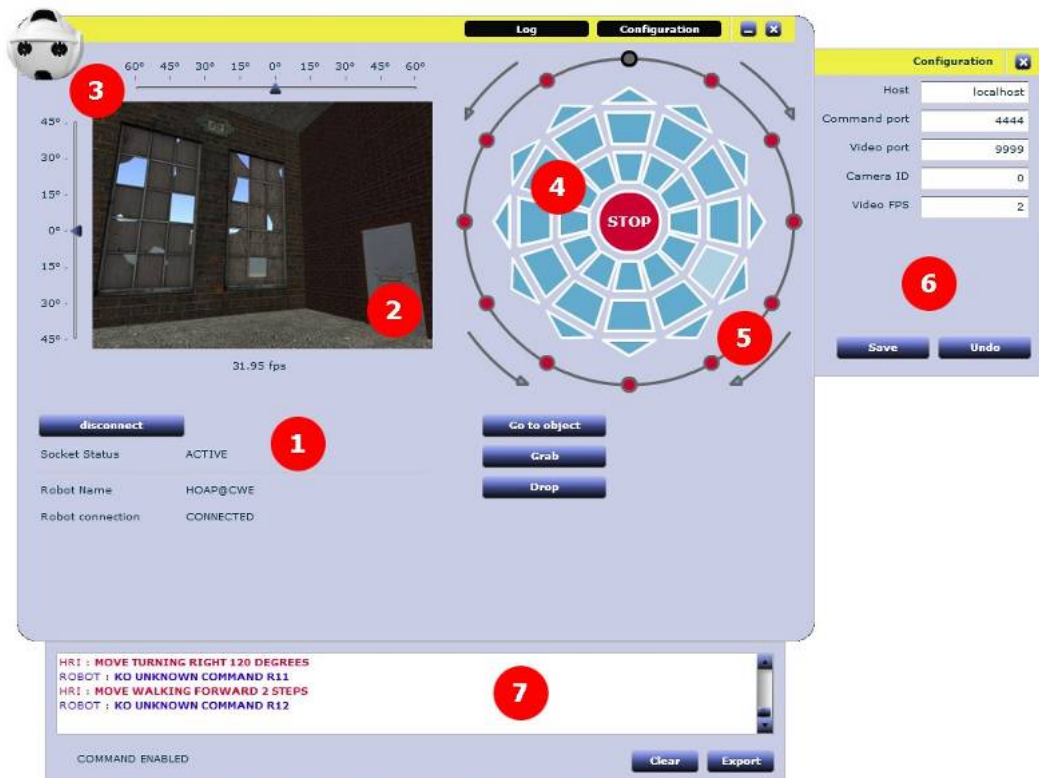


Figure 3 (5) around the speed and direction control

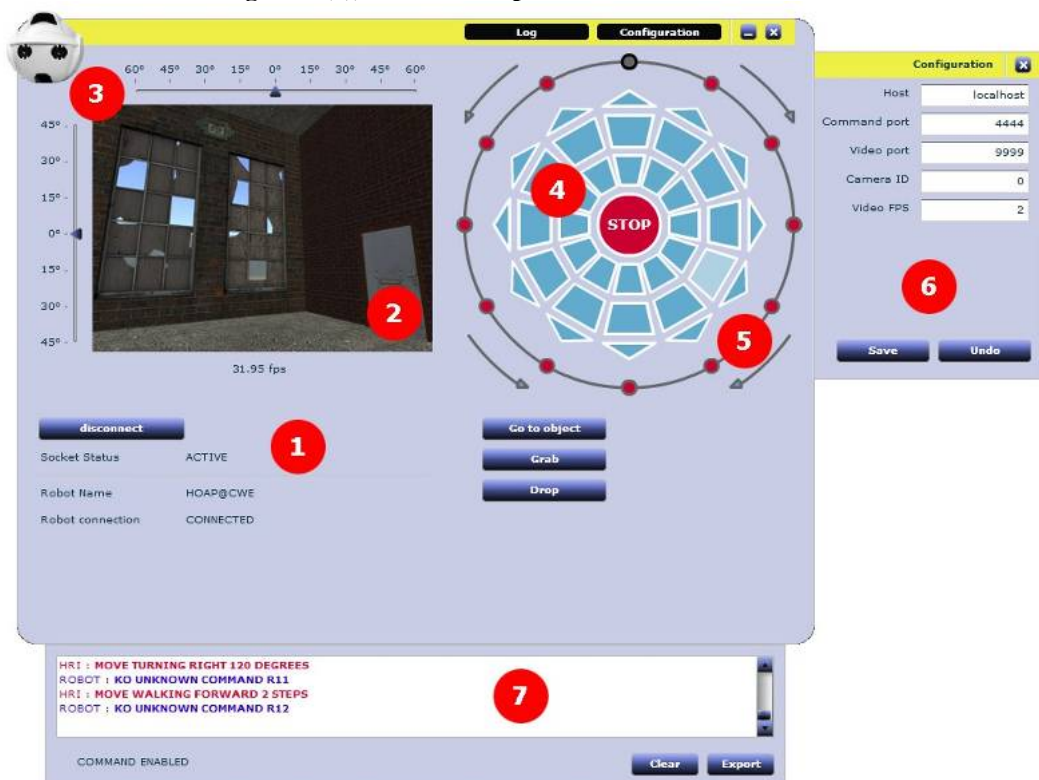


Figure 3 (4)). Furthermore, the size of the camera view could not be enhanced (its range of sight is quite small) due to bandwidth limitations or it is not supported by the robot itself. Further details of how the interface was implemented are described in sections 2.2.7 and 2.4.

2.3.1 Study Setting

The goal of the second evaluation study was to identify further usability problems of the iterated version of the HRI. Firstly, it has been performed a cognitive walkthrough with six experts of HCI and/or design. The reason was that the cognitive walkthrough method focuses on assessing the learnability and inductivity of an interface. Therefore the interface should be usable for novice users and should require a low cognitive load. However, the evaluation method was switched to the heuristic evaluation because the cognitive walkthrough only provides an overview of the particular tasks and not on the general design. Thus, to gather additional information and to provide better design recommendations, a heuristic evaluation was conducted. The results presented below involve insights from both methods. The cognitive walkthroughs and the heuristic evaluations were conducted between June and July 2009 at the ICT&S Center (University of Salzburg) by PLUS. The expert evaluation involved six experts (three females, three males, age between 20 and 30). Three of the experts took part in the cognitive walkthrough, and three of the experts conducted a heuristic evaluation.

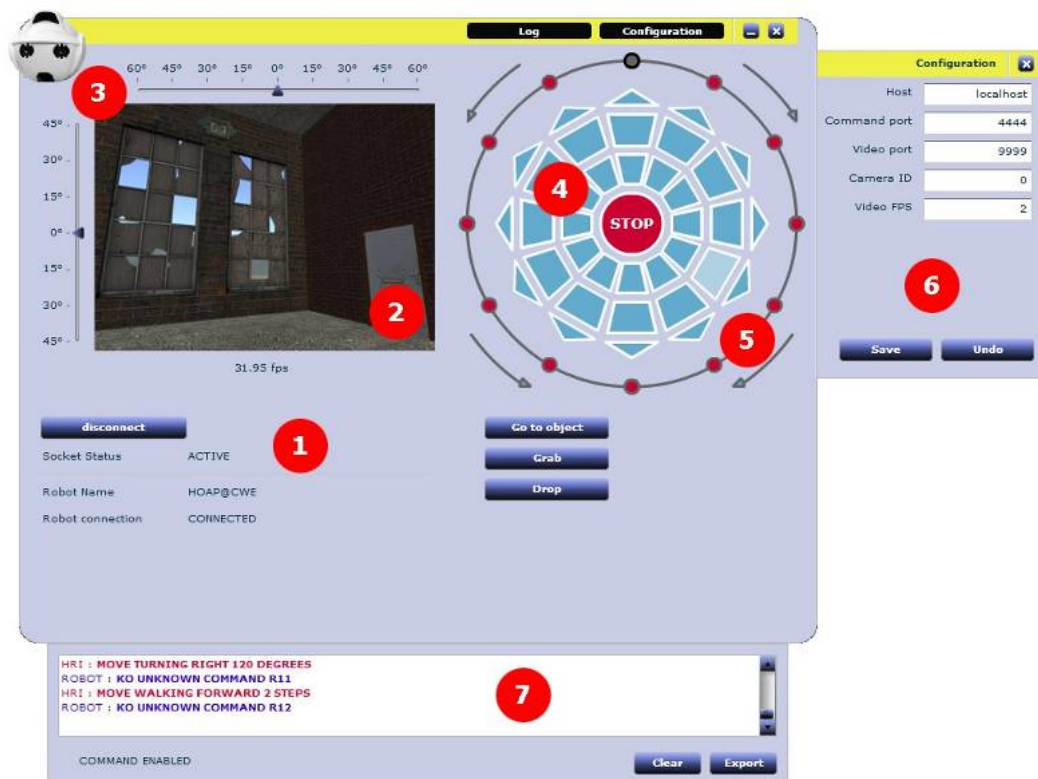


Figure 3 - Interface which was evaluated in the second expert iteration

2.3.2 Instruments

For the cognitive walkthrough, the experts (two male, one female) were presented a scenario which consisted of four tasks and an ideal, step-by-step solution for solving the task. After each step of a task, the expert was asked what he thinks the user will do and which problems could occur. The scenario was the following:

Imagine you are standing in a factory work floor. With the help of the HRI you should navigate a humanoid robot to the table with the red cube. Command the robot by means of the interface to take the cube and bring it back to your start position. The HRI offer a touch

screen for direct manipulation. Through tipping with the finger on that screen you can command the robot to perform specific actions.

The tasks of the given scenario the evaluators had to walk through were:

- 1) Move the robot to the table
- 2) Lift the cube
- 3) Carry the cube back to the starting point
- 4) Put the cube down to earth

The expert evaluators walked through each action sequence answering the following four questions:

- 1) Will the user be trying to achieve the right effect?
- 2) Will the user notice that the correct action is available?
- 3) Will the user associate the correct action with the desired effect?
- 4) If the correct action is performed, will the user see that progress is being made?

If one of the four questions is answered with “NO” then a usability problem is identified. Hereby the expert evaluator should recommend a solution which could solve the problem.

For the heuristic evaluation, one example task and three tasks had to be conducted by the experts (one male, two females). Therefore, each task was split up into single steps representing the ideal solution. After each step, the expert had to indicate which of the given heuristics (see

Table 2) were violated and which other problems may possibly occur. Suggestions for improvements could also be stated out here. The tasks given were the following:

- 1) Example task: Move the robot to the table
- 2) Lift the cube
- 3) Move the robot backwards
- 4) Put the cube down to earth

2.3.3 Procedure

All expert evaluations lasted between one and one and a half hour. After a short introduction the experts examined the tasks stepwise. In the cognitive walkthrough, the experts tried to answer the question of what a typical user would do and which problems he/she would have. At the end of the cognitive walkthrough, the evaluator stated general comments about the design of the HRI. In the heuristic evaluation, the experts tried to find out which heuristics were violated and to find suggestions for improvement.

2.3.4 Results

Through the cognitive walkthrough and the heuristic evaluation of the HRI 41 usability problems were identified whereas 33 problems were different, as 6 usability problems could be discovered in both evaluation methods. 19 usability problems were identified in the cognitive walkthrough and 22 usability problems in the heuristic evaluation. In the following the problems identified were grouped and described generally.

2.3.4.1 Cognitive Walkthrough

This section describes the four tasks and the actions belonging to the task in detail. Whereas below each action sequence a list of problems identified by the expert evaluators in the cognitive walkthrough are presented.

Task 1: Move the robot to the table

1. Step: Connect to socket

Problem: The main problem here is the wording “Connect to socket”. Users who are not familiar with that interface would not intend to push that button in order to connect the interface with the robot.

2. Step: Turn the robot around

In order to look for the red cube in the room, there are two possibilities. Some people will use the head in order to look because it is faster than turning the robot around. As the user has turned the head around and is able to see the red cube, the user has to turn the robot.

Problem: The first problem arising is that it is not possible to adjust the body of the robot to gazing direction of the head.

Problem: Secondly it is not possibly to set the head back to its 0-Position (vertically and horizontally) just with one action. The other possibility is to turn the robot with the circle navigation until the red cube is in the vision of the robot –but this possibility is not as fast as the previous one to orientate and to spot objects.

3. Step: go to the red cube

Problem: To move the robot to the red cube the user has to push the navigation buttons. Hereby different problems arise:

1. The user is not able to estimate the distance to the object;
2. The user cannot estimate the size of the steps with the interface. Excessive training of the interface with one robot must be done, but it could be difficult if the robot type and with the type the size of steps change;
3. The user will have problems when mapping the conceptual model of the navigation with his/her mental model. The reason is because the user's mental model is rather connected to the room and not static bound to navigation model of the robot. E.g. the user's mental model turns when the robot turns in a desired direction, but the navigations of the interface model keeps static. While looking at the coordinate system it looks like an absolute coordinate system, but in reality it is a relative coordinate system.
4. The third step symbol looks like a play symbol, which means when the user pushes this button he/she could think the robot moves as long as the user pushes the stop button. This function would not be bad but some security issues have to be solved before including it. For example, what happens with the robot, if the remote control freezes after the user had pushed that button? Will the robot continue walking or will it stop?

Task 2 Lift the cube

1. Step: push the grab button

Problem: The user will push that button. The following window which appears will be difficult to interpret. The user cannot command the robot to grab the object if the user does not know the id of the object.

2. Step: type in ID

Problem: No feedback appears if the wrong ID was typed in the text field. The only feedback is just available in the message log display. Furthermore, it could be possible that the user does not understand the connection between ID and grab.

3. Step: select the object

Problem: The user is not able to identify how the robot will move the arm.

Problem: It is not possible to cancel the current action.

Problem: As the robot has taken the red cube the user is not aware if the robot carries the cube.

Problem: Furthermore, the wording of the given strategy is not very good.

Task 3: Carry the cube back to the starting point

Problems: see Task 1: Move the robot to the table

Task 4: Drop the red cube

Problem: A problem arises when the window for the "Drop" action appears whereas the user could get confused whether he/she should type in the ID of the cube or the ID of the target object.

Problem: When the robot is not visible, the user has to remember in which hand the object is.

Problem: There is a lack of feedback when the object has been dropped. The user is not able to see the object. He/she has to take a step back in order to get an overview of the current situation.

Problem: Once an action has been selected, there is no possibility to cancel the action.

Problem: The wording of the available actions is not good, for example, what is object A and “Drop” could also mean to let the object fall down, could be confusing.

Problem: When the strategy window appears (drop and grab), the navigation is now enabled. Furthermore, all the other actions are also enabled, for example when clicking on the “Grab” button the previously opened strategy window is still there.

2.3.4.2 Heuristic Evaluation

During the heuristic evaluation 22 usability problems were identified, whereas Table 4 shows which heuristics and how often they have been violated. Furthermore, Table 4 shows that the heuristics 1, 2 and 7 with 23 violations, cover most of the 35 heuristic violations. The reason why the number of violation is higher than the found usability problems is, because some of the usability problems violate more than one heuristic. From the 22 usability problems identified one of them was rated as usability catastrophe (severity ranking of 4) whereas eleven were rated as major usability problem (severity ranking of 3).

Table 4 Heuristics used and number of violations

Heuristics	Number of violations
1) Visibility of system status	8
2) Match between system and the real world	8
3) User control and freedom	1
4) Consistency and standards	3
5) Error prevention	3
6) Recognition rather than recall	3
7) Flexibility and efficiency of use	7
8) Aesthetic and minimalist design	1
9) Help users recognize, diagnose, and recover from errors	0
10) Help and documentation	1

In the following a list of the rated usability problems found for the HRI are presented and grouped into the scale of the severity ranking:

Usability catastrophe

- There is no information available whether the robot has taken or dropped the red cube. (3 out of 3, mean expert rating 3.66, standard deviation 0.56).

Major usability problem

- It is not obvious the further away the buttons are from the center of the navigation the more steps the robot is doing, for instance the button in the inner circle makes one step, buttons at the outer circle, formed like an arrow, 3 steps (3 out of 3, mean expert rating 3.33, standard deviation 0.6).

- For the user it is difficult to understand turning in a direction and then moving. For instance when pushing the move back button, the robot turns 180 degrees and then moves. Then the problem arises, whether the robot moving backwards or forwards (3 out of 3, mean expert rating 3.33, standard deviation 1.2).
- It is not clear which number (ID) belongs to which object (3 out of 3, mean expert rating 3.0, standard deviation 1.0).
- It is not possible to make a step backwards instead the robot turns 180 degrees and then moves one step forward (2 out of 3, mean expert rating 3.0, standard deviation 1.0).
- When turning the robot it is not clear how far the robot has been turned (1 out of 3, mean expert rating 3.0, standard deviation 1.0).
- For turning the robot is no recognizable rule available (1 out of 3, mean expert rating 3.0, standard deviation 1.0).
- The user has to remember the ID of the object (3 out of 3, mean expert rating 2.66, standard deviation 1.5). Numbers are difficult to remember and often a source for human errors.
- There is no feedback available when the robot is performing a task. It is not clear whether the robot is in the middle of the action or if the action has already been completed (2 out of 3, mean expert rating 2.66, standard deviation 0.5).
- The input of the “ID” is not very intuitive. The appearing window recalls on a calculator but the user does not have to calculate something (1 out of 3, mean expert rating 2.66, standard deviation 0.6). For instance one of the evaluators asked, if she has to calculate the way to the cube.
- The buttons “Drop object” and “Use object” are available as well as when the robot is not carrying any object. When or does have the user to know that the robot can carry an additional object (1 out of 3, mean expert rating 2.66, standard deviation 0.6)?
- The user is not able to recognize the distance between the robot and objects in the vision of the robot (1 out of 3, mean expert rating 2.66, standard deviation 0.66).

Minor usability problems

- When grabbing an object, the user has additionally to push the select button, in order to start that action (1 out of 3, mean expert rating 2.33, standard deviation 0.6).
- It is difficult to move the head of the robot back to its 0-Position (horizontally and vertically) (1 out of 3, mean expert rating 2.33, standard deviation 0.6).
- It is not clear, which effect the command “grab object from above” will have, because there is no visualization of the grab skills available (1 out of 3, mean expert rating 2.33, standard deviation 1.2).
- At system start it is not clear whether the interface is connected to the robot or if it is not connected, because the message “No Video” is placed very prominently (2 out of 3, mean expert rating 2.0, standard deviation 1.0).
- If the user is not able to connect the interface with the robot, there is no help or feedback notification available what he/she could do (2 out of 3, mean expert rating 2.0, standard deviation 1.0).
- The wording of “Drag” and “Drop” is misleading. The user does not know which kind of input he/she can expect (2 out of 3, mean expert rating 1.66, standard deviation 0.66).

- The wording of “Drop Object” is misleading. For instance which object is meant when the interface suggests “Place Object A over Object B” (1 out of 3, mean expert rating 1.66, standard deviation 0.66).
- There is no possibility of an undo function, when the user has placed the object on the wrong place (1 out of 3, mean expert rating 1.66, standard deviation 0.6).
- It is not possible to automate repetitious tasks (1 out of 3, mean expert rating 1.66, standard deviation 0.6).

Cosmetic usability Problem

- The buttons for turning the robot around are too small (1 out of 3, mean expert rating 1.33, standard deviation 1.2).

General Recommendations

Analyzing the discovered problems it has been developed a number of recommendations that can help improving Human-Robot Interaction via the HRI. The recommendations are divided into categories matching different User-Interface guidelines [9].

Minimize the user’s memory load

- Typing in numbers is generally not recommendable because the user must keep in mind which and what kind of input is required. Let the user chose from a pool of suggestions the robot is able to recognize.
- Provide the ID for the object, for instance a number above the object in the camera view.
- Change of the naming of the button, e.g. “Connect to Robot” or if there are more robots available “Connect to <Robotname>”. This could also be done in the configuration field, instead of typing in the IP-Address of the robot. A pull down menu could provide the names of the different robots available in the network.
- The object which has been selected should be fat outlined. In order to avoid typing in some ID, a list of available objects should be provided.
- The robot should provide suitable strategies which the user only has to confirm or - if not satisfied - cancel. After the robot has taken the red cube make it visible in which arm the robot carries the object.
- Only suggest actions of the robot which are possible to perform. For instance show the user which arm carries which object. Hereby the user does not have to remember the ID of the object.

Flexibility

- Provide a button to reset the position of the head to its origin position. Furthermore if possible, provide a second button in order to adjust the body of the robot to its head.
- Remove unnecessary action steps, for instance select an action (grab) and then confirm the action with select.

User Control and Consistency

- In order to be able to estimate the distance of objects, provide a grid. For example one grid can be compared with the length of one step. Another way could be by displaying the amount of steps when looking at an object.
- The navigation should provide side steps and backward steps. The buttons for changing the directions are valuable.

- Make it possible to cancel the current action, e.g. with a “Cancel-Action”-Button, or enable the “Abort” button to cancel the current action, for instance grab an object.
- Disable the actions when a window is opened. Enable that window with a close button in order to close the window.
- Only enable actions of the robot which are possible, for instance, it would be better if the robot is not carrying any object, that the button “drop object” is disabled.
- The frame of the window for selecting a strategy, for instance, “grab object from above” is not the same as the previous one. Moreover, with the missing title (“Grab”) there is no information available of which action is being performed.
- Provide a map, so the user can see where the robot is. You can link the direction the robot is looking in the map to the speed and direction control. When the robot turns, then the speed and direction control is also turning. Furthermore, you can highlight the speed and direction control which button moves the robot forward.

User Language

- Clear wording is important in order to avoid misunderstanding. Maybe “Put down the object” could be suitable.
- Other wording of “Grab object”, maybe “Grab object with number” so the user knows when the next window appears, that he/she has to type in a number.

Other Design Issues

- As long as an action is being performed, provide a small visualization of the action. Such an animation could provide the necessary feedback that the user knows that the action is being carried out and, what is going on.
- The interface should provide the user with the best available solution. There should be no need to change the frame rate of the video stream for example.
- The video screen and its range of sight is quite small, it is difficult to navigate using such a small screen as only little of the environment can be seen.

2.4. Description of the new HRI

A lot of work has been done since the last D3.3 deliverable and a new working version of the HRI user interface has been produced and tested, keeping into account the recommendations from the first usability experts' evaluation (see section 2.2.6). In this section is described the user interface, its components, main functionalities, the improvements implemented after testing and the extensions to the Robot Command Protocol (RCP) designed and implemented for the last demonstration. Both the HRI and the RCP protocol have been described also in a paper presented at ICAR 2009 [21].

2.4.1 Graphical Interface and functionalities

The HRI user interface is shown in Figure 4. Its main functionalities are:

- Connect to one robot at a time via (Wireless) TCP/IP
- Display streaming video from the robot camera
- Drive robot's movements and speed
- Move robot's head (tilt and pan)
- Send high-level commands (tasks) to the robot

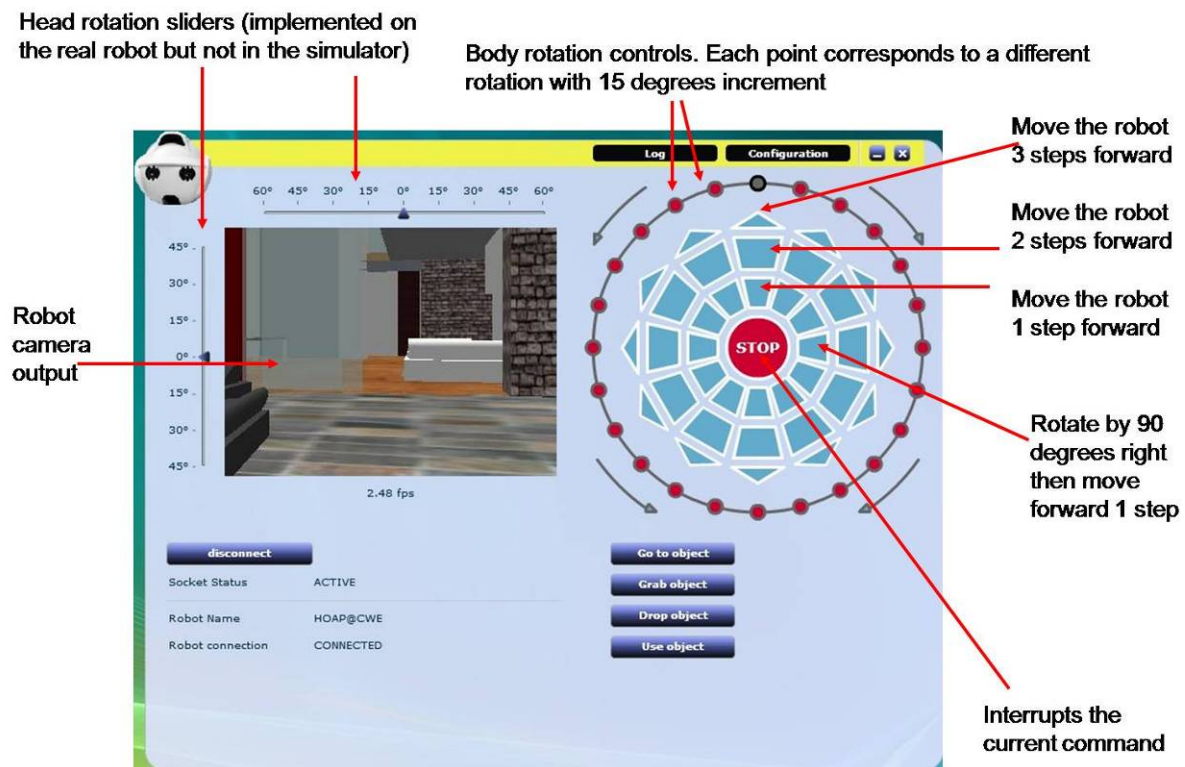


Figure 4 - HRI interface and main components

The main innovation in the HRI user interface is the graphical control component, shown in the upper right part of the window, which allows the operator to move the robot in several directions at different speeds, rotate it and stop it. The control, implemented in Flex as the rest of the HRI, is the result of a trade-off between the concept idea suggested by the first evaluation and the limitations imposed by GUI and robot technology. The main modification from the idea to the implementation was the change from continuous to discrete control for direction and speed.

Further functionalities of the HRI, not shown in Figure 4, are the log window, where all the commands sent to the robot and related responses can be seen and the configuration window, which allows the specification of robot's IP address and ports, desired streaming speed, video encoding and network latency. The installation of the HRI and of its prerequisite software (Adobe Flash + AIR) is also very easy: it's enough to click on the installation link in a browser and the installation wizard does all the work.

2.4.2 Extensions to the RCP protocol

The enhancements to the RCP protocol for the 2nd year demo allow the following user-robot interactions:

- user tells the robot “go there” (see also Goal-setting sub-protocol in D3.3)
- user tells the robot “grab object”
- user tells the robot “put down object”
- user tells the robot “use object”
- robot asks user “which strategy for this task?” (a list of strategies is proposed by the robot)

- user answers selecting one of the strategies

The commands added to the RCP protocol are the following:

- GOTO <location> / GOTO OBJECT(<object_id>)
- GRAB OBJECT(<object_id>)
- DROP OBJECT(<object_id>)
- USE OBJECT(<object_id>)
- SELECT STRATEGY FOR <cmd instance id> [<string>, <string> ... <string>]
- USE STRATEGY FOR <cmd instance id> <string>

A detailed description of the new commands can be found in the appendix to this document.

2.4.3 Improvements after testing

Testing of the HRI interface was first done in the lab by connecting to a server written in Java and simulating both command execution and video streaming. Without the need for usability experts, since the first tests some problems with early versions of the HRI were evident.

A first problem was with the use of network bandwidth by the video stream. The server in the HOAP-3 robot provides a video stream by sending uncompressed 320×240 YUV bitmaps of 112,5kB each. To have 25 frames per second (fps) the needed bandwidth is nearly 22Mb/sec, which is a lot for normal LANs and definitely unfeasible on the Internet. The result was an interface showing at most 1 or 2fps.

The HRI interface was improved to accept a video stream composed of compressed jpeg images. In this way a more acceptable 20fps on LAN has been reached, slowing to a still manageable 7fps over the Internet with an encrypted VPN between client and server.

Further testing of the HRI was done after setting up the OpenSim 3D virtual world with the possibility to control an avatar through the same command protocol used for the robot (more details on this in section 5).

Another problem which appeared evident when commanding avatar movements through the HRI was the low rotation granularity, fixed to 30 degrees in the first versions. With such coarse grained rotation it was very difficult to target an open door or climb stairs, thus the user experience was improved by doubling the rotation granularity to 15 degrees.

3. Interacting with HOAP-3 robot

The First use of the developed HRI was to control and teleoperate the HOAP-3 humanoid robot. The HRI allows for an operator to see the environment from the HOAP-3 cameras, as well as to control various movements of the robot and give orders for doing some tasks, e.g. “use object”.

A first teleoperation experiment with the HOAP-3 robot and the HRI were conducted for the second year demonstrator at UC3M. The experiment consisted on working collaboratively with a robot to perform a task by the teleoperation of the HOAP-3 robot. The human agent works collaboratively with the humanoid robot by supervising, controlling and helping in the decisions taken by the robot.

For the third year demonstrator and experiment featuring several technological sub-demonstrations is being prepared with the work of partners HP-EIC, EPFL, DRAGADOS and UC3M. Using the HRI an operator will control the HOAP-3 robot with HP tablet-PC interface. The Humanoid Robot and the Human Operator will work collaboratively in the construction of a “shelter”. The Operator will teach a robot the necessary skills to perform this task. Also the 3rd year demonstrator experiment would present Robot-to-Robot collaboration for the transference of the skill knowledge to a robot at a remote location. With the help of the HRI the Human Operator will supervise and monitor the proper performance of the task.

3.1. Second year demonstrator

For the second year demonstrator an experiment of teleoperation and collaborative work was conducted with the HOAP-3 humanoid robot at UC3M. A lunar scenario was built to simulate the operation of a robotic agent working in collaboration with a human in a space environment. The task to be performed consists of teleoperating the HOAP-3 robot, first walking through an enclosed hall and finding an object, in this case an ‘antenna’, then grasping the object and placing it in a different location [21].



**Figure 5 - a) the recreation of lunar scenario.
b) HOAP-3 teleoperated through a corridor looking for the ‘antenna’**

The HOAP-3 is a humanoid robot; its height is 60cm and the weight is 8.8kg. This robot is capable of handling small objects and provides vision recognition. With the HRI the human operator works collaboratively with the robot to complete the task. The HRI allows an operator to see the environment from the robot cameras, as well as to control various walking

and turning movements of the robot and give high level orders for doing some tasks, e.g. “grab object”.

3.1.1 2nd Year Demonstrator concept architecture

Figure 6 shows the architecture for the second year demonstrator. A human operator works with a humanoid robot in the realization of a task, picking of an object. The human operator supervises and monitors the robot performance of the task while the robot autonomously performs the task after receiving orders from the human operator to do so.

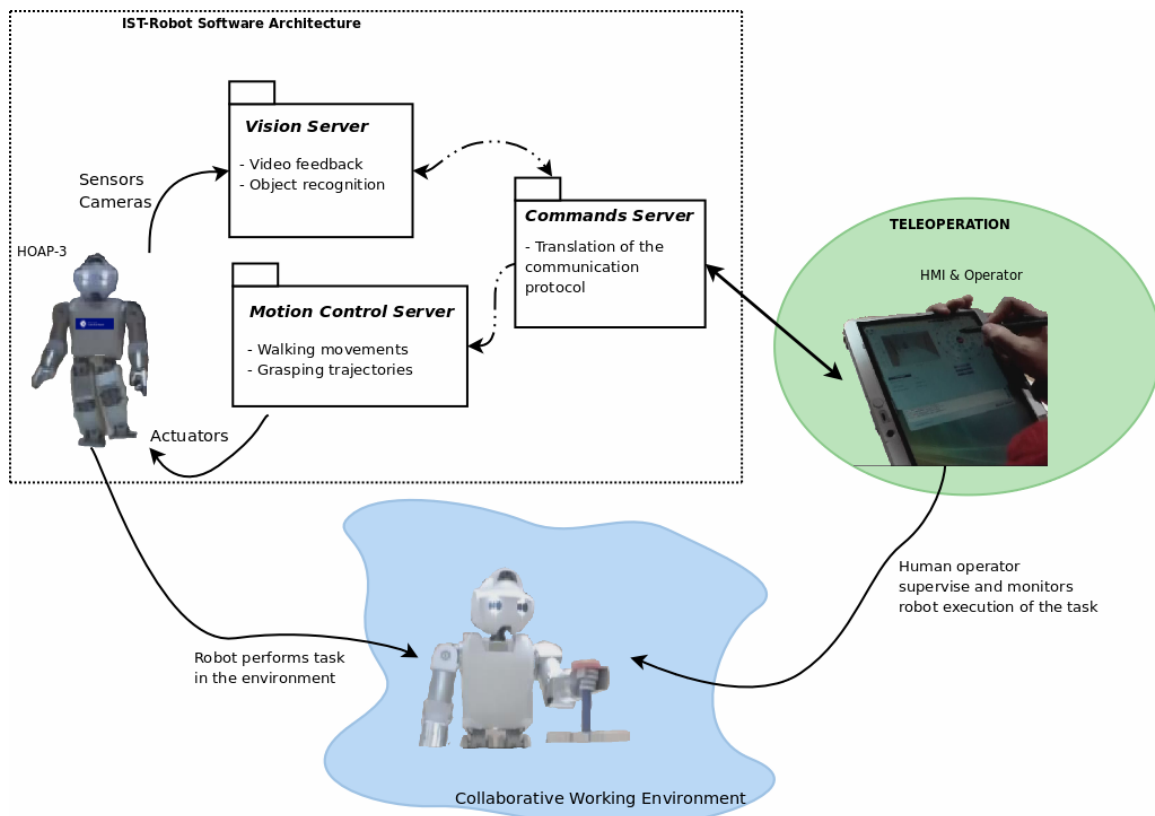


Figure 6 - Global overview of the 2nd year demonstrator architecture

There are two major software components for the Demonstrator architecture, the HRI and the HOAP-3 server. The HRI was described on Chapter 3. The HOAP-3 architecture will be explained in more detail on section 4.2.1. The robot software consists of three modules, Vision Server, Commands Server and Motion Control Server.

The Vision Server is in charge of vision services, it provides video feedback and visual cues of what the robot is seeing to user. It also gives distance and orientation parameter from the recognizable object to the Motion Control Server.

The Commands Server is in charge of the communications with the HRI. It forwards the instructions from the operator back to the other services and gives the adequate responses to the operator and the HRI.

The Motion Control Server is in charge of sending the appropriate commands to the actuators for the walking, turning and grasping motions.

3.1.2 Results and Issues

The 2nd Year Demonstration experiments were conducted on December 2008 in Universidad Carlos III de Madrid (UC3M). A scenario has been designed to simulate a lunar ambient. It consists on a long corridor surrounded by cliffs where the robot can walk and interact with the environment. The surface of the cliffs has been built with planes of polystyrene where it has been made holes to simulate craters.

This scenario allows the human operator to interact with the robot. Through teleoperation and using the HRI, human gives the humanoid the order to walk until it sees an object that simulates an antenna. Video feedback from the robot cameras indicates to the operator that the robot has located the ‘antenna’. Then, the robot approaches the object to a close enough distance so that it can grab it when requested by the human operator. Finally the robot computes the best trajectory for the grasping movement and performs accordingly to the operator decisions.

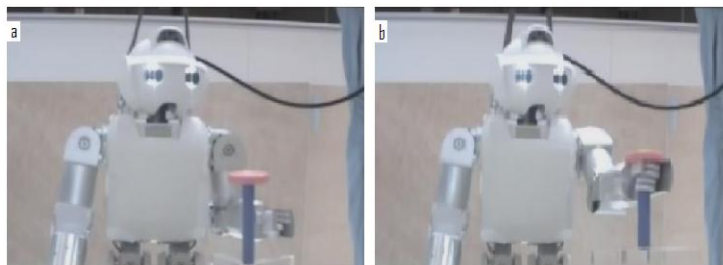


Figure 7 - Robot HOAP grasping the ‘antenna’ a) the robot computes grasping trajectories and receives a command by the operator. b) robot successfully picks up the ‘antenna’

3.2. Final demonstrator: collaboration with Robot@Moon

The overall story of the demonstrator is a robot at a remote space location, the moon, will perform autonomously a task of building a “lunar shelter”. The robot would not know how to perform this task at first and it will learn the task with the assistance of a Human-Robot team at earth. An expert operator at earth will teach a Robot the task and then Robot@Earth would communicate the learned task to Robot@Moon.



Figure 8 - 3rd year demonstrator. Operator teaching skill to robot (EPFL). HOAP-3 robot walking on the moon scenario (UC3M)

When necessary the human operator will teach a ‘mirror Robot’ at earth, with the same capabilities and functionalities that Robot@Moon, how to perform the Task. Using the robot programming by demonstration teaching and learning techniques, developed by EPFL, the operator will teach a Robot@Earth the skills needed to perform the task. Once the Robot@Earth has learned the task it will communicate through the Shared Knowledge Database the learned skill to the Robot@Moon. If the operator is not satisfied with the reproduction of the skill it can retrain the task until it is done in an appropriated way.

For the 3rd year demonstrator various agents would be working together to carry out the tasks. A humanoid Robot HOAP-3 located at the moon scenario build at UC3M will be performing the building “lunar shelter” task. A human operator located at EPFL in Lausanne will have two tasks: First, with the developed HRI interface the operator will send instruction for teleoperation and monitor the state of the HOAP robot at Moon. With the teaching and learning techniques developed by EPFL the operator will teach how to perform the task to a robot at EPFL. The humanoid Robot HOAP-3 at Lausanne (EPFL) will learn the task and transmit this knowledge to robot HOAP at moon (UC3M).

The experiment for the 3rd year demonstrator would have three major subtasks: autonomous walking, teaching of the skill task and execution of the new skill. These subtasks will be under the supervision and teleoperation of the human operator using the HRI. The autonomous walking subtask is handled by the HOAP-3 software server that will be presented in section 3.2.1. The teaching of the skill subtask utilizes EPFL robot programming by demonstration learning techniques already detailed on deliverable 3.1-2. The transference and the execution of the new learned skill is handled through the Shared Knowledge Database, described in section 3.2.2, and the HOAP-3 software server.

Figure 9 shows the architecture for the 3rd year demonstrator. The human operator monitors the state of Robot@Moon with the HRI and sends instructions for teleoperation to realize the tasks. HOAP-3 software server handles the robot movements and motions and maintains communication with the operator through the HRI.

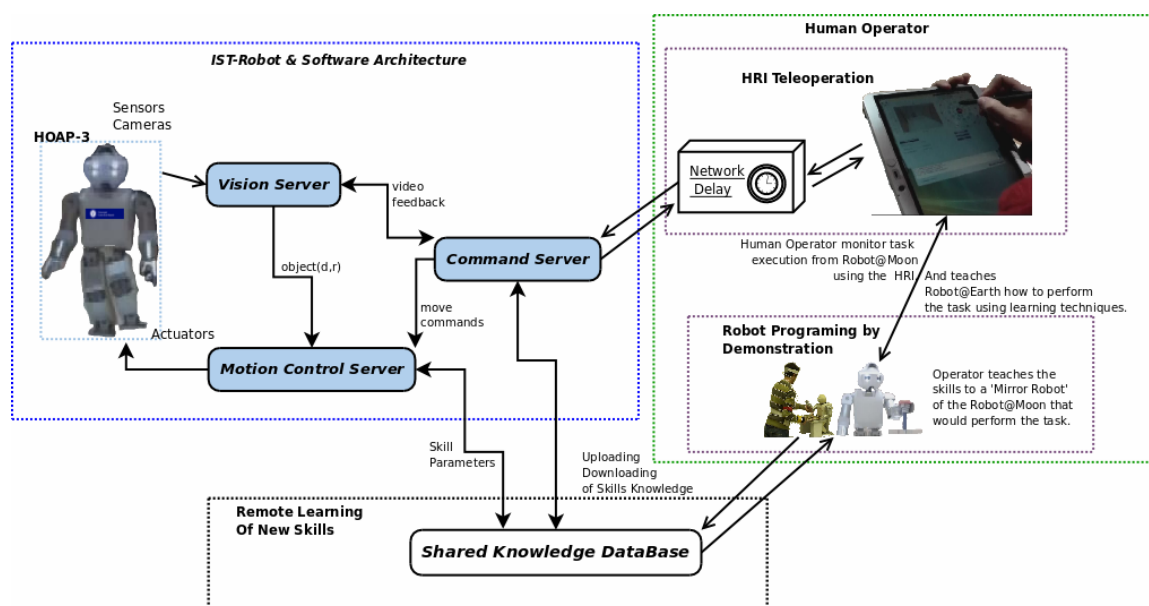


Figure 9 - Global overview of the 3rd year demonstrator architecture

3.2.1 HOAP-3 software server

The HOAP-3 software architecture is composed of three modules: A vision server for visual perception, a commands server to handle communication protocol with the HRI and the motion control server to control movement of the robot. In Figure 10 the architecture for HOAP-3 server is depicted.

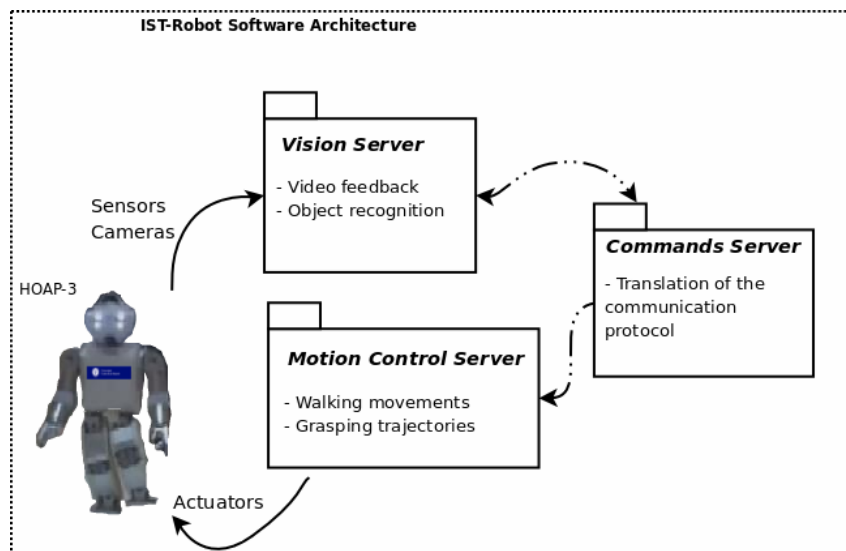


Figure 10 - HOAP-3 software server architecture

The command server is in charge of translating the communications with the outside world, through a Robot Command Protocol (RCP) [14], with the operator on the HRI to the other modules. Vision server is in charge of the vision capabilities of the robot. The motion control server is in charge of controlling walking and grasping movements of the robot. All HOAP-3 software server modules are program in C/C++ and run under RTLinux environment inside the HOAP-3 robot processor.

3.2.1.1 Vision server

The Vision Server module handles the vision services for the HOAP-3 Robot. It grabs images from the two robot cameras. The vision server provides the user with video feedback and visual cues, regarding what the robot is seeing. Object recognition capabilities are managed by the vision server. The vision server can detect and track objects in its environment. And it also gives distance and orientation parameter from the recognizable object to the Motion Control Server.

3.2.1.2 Command server

The Commands Server handles communications with the HRI. It is in charge of translating the Robot Command Protocol (RCP) [14] to forward the instructions from the operator back to the other services and gives the adequate responses to the operator and the HRI. RCP is a text-based protocol which has its roots in Unix protocols like SMTP or FTP. Each RCP command is a text string terminated by a newline character.

The commands server will receive and process all the requests that must be handled by the robot like capture video frame, move or grasp, or use the learned skill with the shared knowledge database.

3.2.1.3 Motion control server

The motion control server is in charge of controlling the joints' actuators and of sending the appropriate commands for the walking, turning and grasping motions. The motion control server receives orders from the command server of what motions must be performed; it also receives distance and orientation parameters of the tracked object from the vision server. The motion control server would generate the trajectories to walk towards and objects and grasp them.

Also the motion control server communicates with the shared knowledge database to obtain the learned skill parameters necessary for its reproduction.

3.2.2 Shared Knowledge Database (BSCW)

The shared knowledge database allows the two HOAP robots to transfer the skill learned knowledge. The shared knowledge database would hold common information learned by the robots HOAP. The learn information to be shared would primary concern object and task data. Object data contains necessary information for the recognition and identification of the object and any constraint relate to it: Tag, Color, Size, Shape, etc. The task data contains the skill parameters and necessary information to reconstruct the model of the skill for the task.

The shared knowledge database is an interface that allows robot-to-robot interaction. Both Robots are able to access the shared knowledge database and upload or download learned skill knowledge for the reproduction of the tasks. The shared knowledge database would communicate with the HOAP-3 server commands server module and motion control server module for the learning of parameter to reproduce new skills.

3.2.3 Delay module

In order to simulate the conditions and problems that arise in a real space communication applications, a controllable time delay module has been added between the HRI and the robot. This module permits to set a variable delay and to test the performance of the proposed teleoperated system in space environments. The utility acts as a proxy between the HRI and the robot, handling both video and command channels. Every time the utility receives some data it waits for a configurable time before transmitting it to the other side. The selected delay introduces a latency in the communication network to simulate the problems with communication that would be present when working collaboratively with a robot in such a remote location as space or the moon.

4. Controlling an avatar in a 3D Virtual World

The first need for a 3D simulation framework rose when HP was designing strategies for both testing and demonstrating the HRI user interface, from the basic consideration that HP Italy has no robotic platform to be driven by the HRI. Our requirement to have a 3D simulation environment that could be easily used also for demonstration purposes drove us to select a widely known environment, available in open source, named OpenSimulator. By using our HRI to drive an avatar in a 3D virtual world HP obtains the possibility to perform a robot-less HRI demo to its customers. The HRI-OpenSimulator integration has been very useful also for the second usability evaluation of the HRI (reported in section 2.3.). Access to the OpenSimulator server in HP has been made fully available to PLUS through a VPN and the HRI interface has been evaluated by PLUS experts connected to the OpenSimulator virtual world and driving an avatar in it. Moreover OpenSimulator has been integrated with BSCW and will be used in the final demonstrator Robot@Construction-site to create a task in BSCW and assign it to the HRP-2 robot for execution (see section 5.2.2).

The issues to be solved for providing such an HRI-OpenSimulator integration were: controlling an avatar from a program running outside the 3D environment, translating RCP commands to avatar commands and extracting a video stream showing the virtual world from the avatar's point-of-view.

4.1. OpenSimulator

OpenSimulator (www.opensimulator.org) is an open source 3D application server, and allows both creating and running a virtual world.

Wikipedia [10] says that *a virtual world is a computer-based simulated environment intended for its users to inhabit and interact via avatars. [...] Some, but not all, virtual worlds allow for multiple users. [...] The computer accesses a computer-simulated world and presents perceptual stimuli to the user, who in turn can manipulate elements of the modelled world and thus experiences telepresence to a certain degree. [...] The model world may simulate rules based on the real world or some hybrid fantasy world. Example rules are gravity, topography, locomotion, real-time actions, and communication.*

A virtual world, being a multi-user virtual environment, can also be seen as a kind of Collaborative Working Environment, where people can interact and collaborate to accomplish common tasks despite their physical distance. In the spirit of Robot@CWE the virtual world has been selected as a mean for humans and robots to collaborate.

OpenSimulator is released under a BSD license (some call it the “Apache of virtual worlds” [20]) and can be used to simulate a virtual world similar to the widely known and widespread Second Life™ environment (including client compatibility).

Virtual worlds have many application domains, such as gaming, social, psychological research and therapy, commercial, advertising, education and – what interests us – simulation. Physics simulation in OpenSimulator can be handled by several integrated third-party libraries, such as Open Dynamics Engine (www.ode.org). A physics engine, by simulating gravity, collisions... helps in making the virtual world more similar to the real one.

In the OpenSimulator virtual world, like in Second Life, any user can use built-in three-dimensional modelling tools to create geometrical objects and compose them to *build* structures and environments which can either represent only the result of one's fantasy or the copy of real environments. Some screenshots of structures and environments available in HP's OpenSimulator-based virtual world can be seen in Figure 11.



Figure 11 - Some screenshots from HP's OpenSimulator virtual world

An OpenSimulator virtual world can be accessed directly with a client which is compatible with the Second Life protocol, such as the open source Hippo OpenSim Viewer [11].

To create a virtual world in OpenSimulator and building 3D environments there are various tools available. The first and most available one is the building tool integrated with the Hippo Viewer, which allows creating 3D objects from within the virtual world itself. When you create a new virtual region, you start with a small, empty island in the middle of the sea. You then have to modify the terrain (mountains, plains, sea) and create objects (buildings, trees, etc.) to build the desired scenario (see, Figure 12).

Another option is to create a full scenario using a 3D modeller such as 3D Studio Max (3DS) and then use a tool to export the 3D model into a format which OpenSimulator can load. The most used tool for exporting a model from 3D Studio Max to OpenSimulator is a 3DS plug-in called Prim Composer [12].

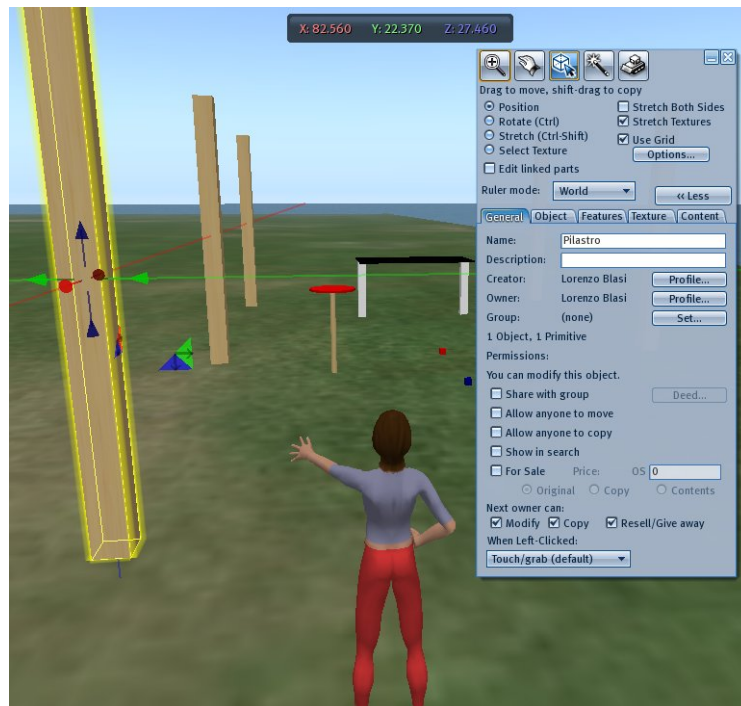


Figure 12 - Creating an object with Hippo OpenSim Viewer

4.2. Integration between OpenSimulator and HRI

Our main goal in exploiting OpenSimulator was to use an avatar as a substitute for a robot in HRI testing, usability evaluations and customer demonstrations.

The first issue to be solved for this was to find a way for controlling an avatar from a program running outside the 3D environment. To do this an OpenSimulator plug-in has been developed which exposes a REST HTTP interface. The current version of the interface exposes a subset of the OpenSimulator functionalities, mainly related to avatar movements and interactions with objects (grab, drop, use).

The second issue to be solved was about translating RCP commands to avatar commands. In the first stages of HRI development a Java RCP interpreter was created for testing the Command Protocol interactions between the HRI and the robot. To control the OpenSimulator avatar we extended the RCP interpreter created for testing to obtain a module which translates each command of the RCP protocol into the corresponding OpenSimulator command. This RCP interpreter thus, in the code for the execution of each command, calls the related REST functionality by issuing an HTTP GET request to the OpenSimulator plug-in.

The third issue was related to the video stream to be displayed by the HRI. When the HRI controls a robot this video stream comes directly from the robot camera. When otherwise the HRI is used to control an avatar in the virtual world the problem is to produce a video stream representing what the controlled avatar “sees” of the virtual world. This is a very complex problem in that it involves real-time rendering of all the objects, terrain and other avatars in the field-of-view. Luckily it’s the same problem solved by any virtual world viewer, so our solution was to encapsulate a viewer on a dedicated (virtual) machine and create video frames by capturing at regular intervals the screen of that VM.

The architecture of the components described above and used to interface the HRI with OpenSimulator is described in Figure 13.

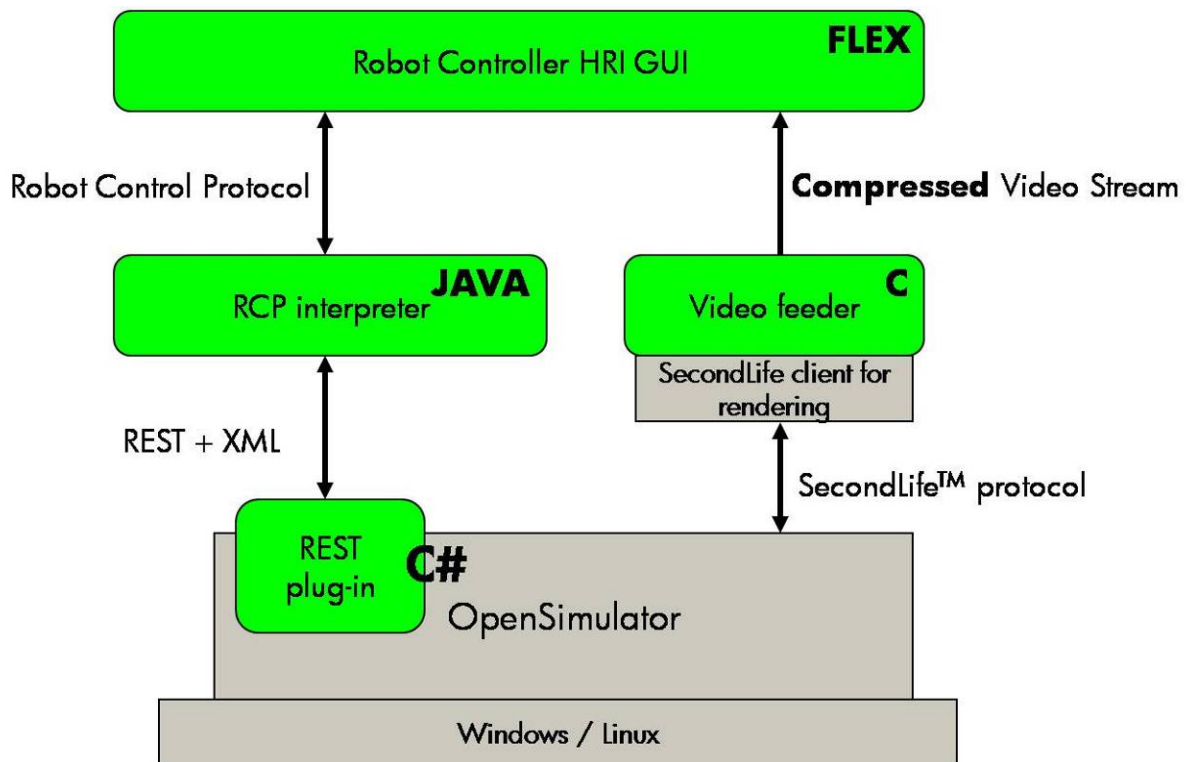


Figure 13 - Architecture for the HRI to OpenSimulator integration

The figure also shows the technology used to develop each component. The OpenSimulator code is written in C#, so the REST plug-in had to be written in the same language. The RCP interpreter was the evolution of the Java RCP simulator, while the video feeder is a C program invoked by a Windows shell script and the HRI itself is written using the Flex language and framework. It should be noted that C# is a technology born and mainly used on Windows platforms, but HP found also the possibility to deploy it on Linux by leveraging the Mono [13] open source .NET development framework.

Another issue to be solved in the translation from RCP commands to avatar commands was their difference in terms of invocation type: OpenSimulator commands are synchronous, while the RCP protocol is asynchronous. The problem was solved in the design of the REST protocol interactions common to all commands. The solution is depicted in Figure 14 and involves two HTTP REST request/response interactions for each synchronous command.

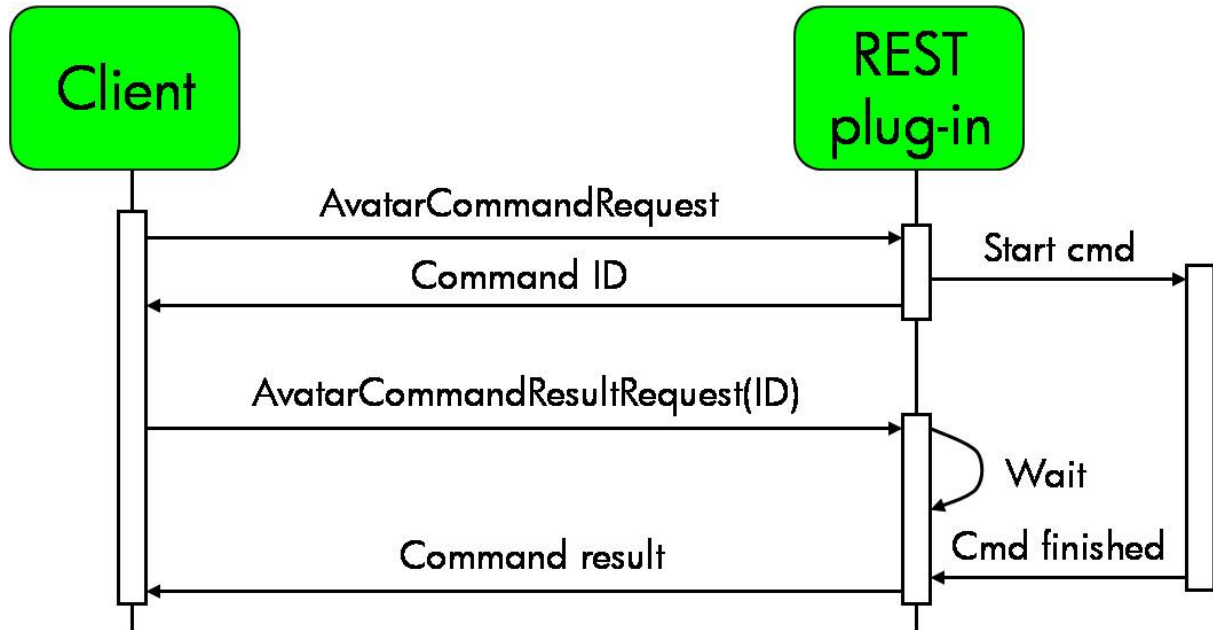


Figure 14 - Generic REST command request/reply protocol

The first request is an AvatarCommanRequest which identifies the avatar and sends the command to be executed, obtaining the relative command ID in the response. The second request is a (synchronous with timeout) AvatarCommandResultRequest(ID) which waits for the completion of the command with a given ID and returns the related result. Another possible request is the AvatarCommandStopRequest(ID) which interrupts the running command corresponding to the given ID.

5. Interacting with HRP-2 robot

5.1. Second year demonstrator

The second year demonstrator has been described in a journal paper in which technical details are provided [15] (which can be downloaded from www.robot-at-cwe.eu). The main aim of this demonstrator is to briefly show how it was possible to give high level mission to a robot through a collaborative working environment software platform. In this set-up the robot is able to go to one place automatically and send back a report as well as an image of the inspected location. The overall functional structure is described in Figure 15.

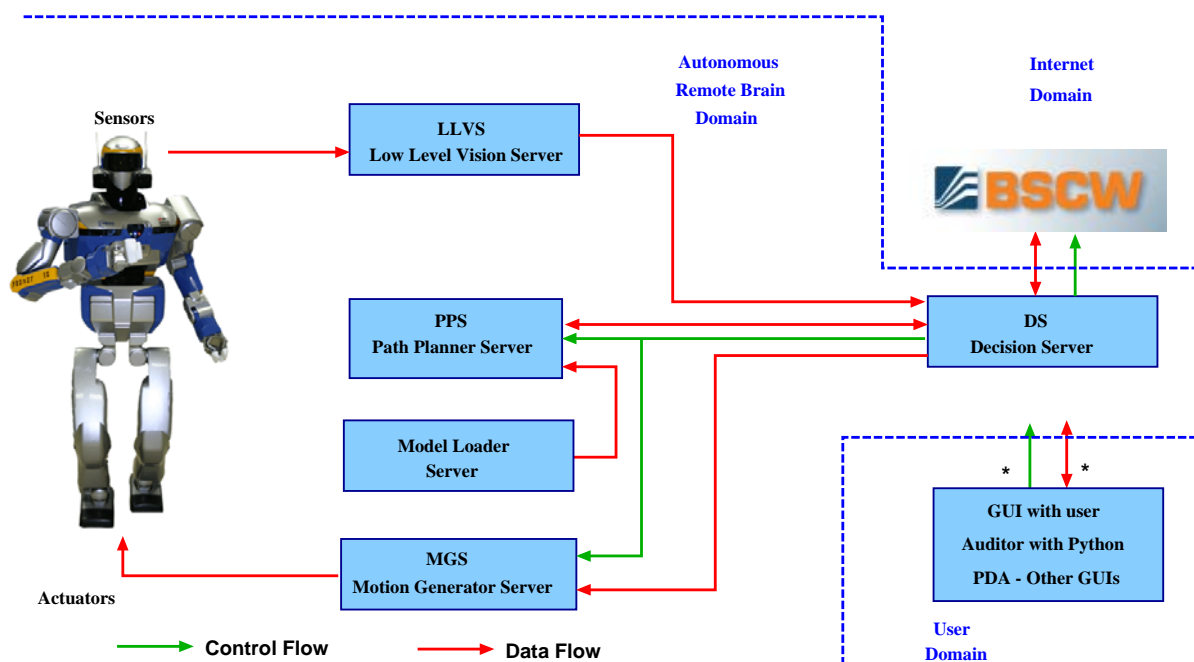


Figure 15 - Data and control flows of the software architecture

Each of these blocks represents a CORBA server which provides specific functionalities. The Low Level Vision Server provides specifically here the images send back to the user, it can also perform low-level vision processes. The Model Loader Server loads the model of the environment and the model of the robot for collision checking. The Path Planner plans series of steps to move from one location to the other. A compact representation of the robot (i.e. its bounding box) coupled with a non-holonomic mobile robot model allows fast planning resolution. The result of the motion planner is then sent to the Motion Generator Server. The Decision Server is implementing a Hierarchical Finite State Machine which controls the overall behaviour of the robot.

5.2. Final demonstrator: collaboration with Robot@Construction-site

The final demonstrator, detailed in deliverable D4.1-3, integrates three major differences with the one presented during the second year:

1. It includes two parts where a distant operator takes over the robot to perform a task. The first part is an extended range for tele-operation of HRP-2 considering its walking capabilities. The remote-operator has to make the robot grasp a remote object. In the second part, the remote-operator helps the local operator to handle an object through the HRP-2 humanoid robot.
2. There is an autonomous physical human-humanoid robot collaborative task which consists in moving an object from one location to another.
3. There is a lifting phase of the object to be moved using learning techniques.

From the software viewpoint the first point called for a specific network protocol having a direct access to the Motion Generation Server and to the cameras of the robot. The control over those accesses has been realized by a CORBA server acting as a proxy between the Vishard7 system located in Germany and the Humanoid Robot Software Toolkit.

The second point has mostly an impact inside the Motion Generator Server and more precisely on the structure of the walking pattern generator and of the inverse kinematics based control scheme. Indeed a key technology to achieve this behaviour was to have a very fast pattern generator included inside in versatile control architecture. Therefore, the internal architecture was extended to cope with modification of steps inside the receding horizon of the control involving stability. In order to test several kinds of controllers, namely the force control scheme for the wrists, the control of the HRP-2 head, and the controllers involved in balancing the robot, an architecture has been developed which allows on-line loading, instantiate, and prioritizing such controllers. The algorithm used for walking and the controllers involved in the process are presented in details in the paper accepted for IEEE Humanoids 2009 conference [16]. The software architecture called the Stack-Of-Tasks has been described in details in a paper presented at ICAR 2009 conference where a special session on Robot@CWE where organized by the coordinator [17].

Although it was initially planned to build a separate CORBA server for the learning part as described in deliverable D3.3, this has been changed. The learning algorithms have been organized as a library, and they are currently directly used at the control level inside the Stack of Tasks architecture.

Finally the initial software architecture proposed in deliverable D3.3 has been implemented, but not all of it is used inside the final demonstrator. The Visual Object Model and Visual Attention CORBA server are currently only used for object model building as described in papers [18] and [19].

The architecture used for the final demonstrator 2 is depicted in Figure 16. In addition to the previous modifications, it was added a connection to Skype in order for the HFSM to communicate its state. However the robot communicates only important transitions to help the user to prepare for the following actions to be performed. The protocol used for the communication with Linux is D-BUS, a light-weight CORBA like protocol for Desktop applications. The robot too runs in its computers Skype for audio communications between the remote and local operators.

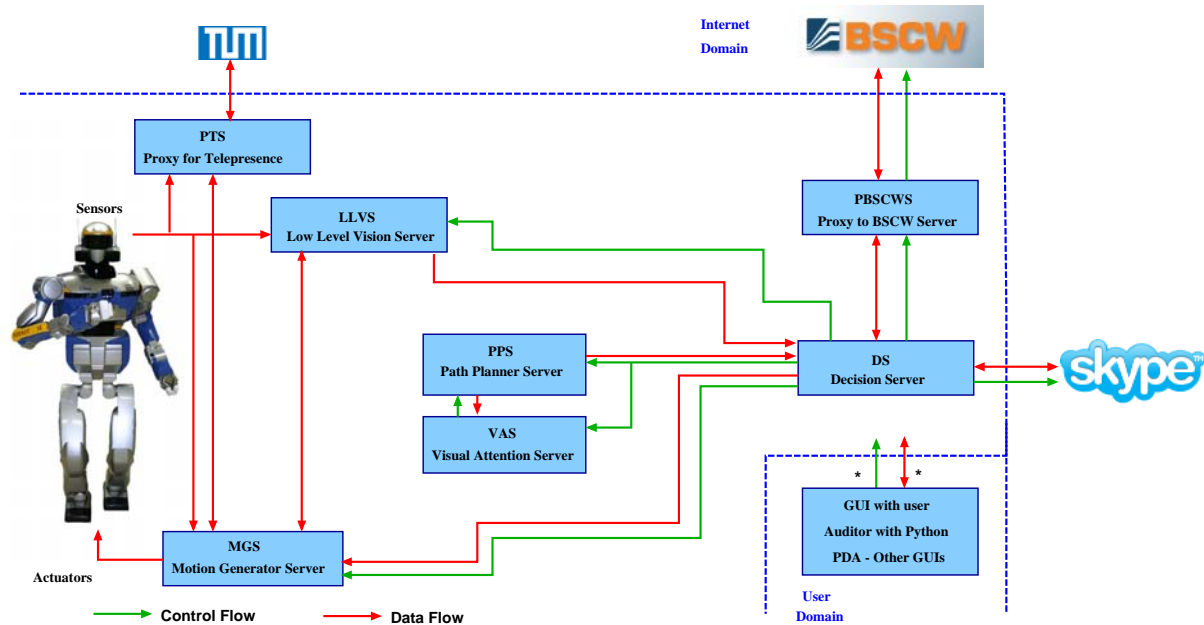


Figure 16 - Data and control flows of the final demonstrator.

5.2.1 BSCW

The interaction with BSCW is realized through a XML-RPC client which acts as proxy between the Decision Server and BSCW. This proxy uses a library implementing all the needed calls to the API defined for BSCW. The library has been provided to HP to implement in the OpenSimulator the call to create the task starting the overall behavior. In addition to this library which is only specific to BSCW, it has been defined an organisation of the shared environment for the user to assign tasks to the robot. The class of tasks that the robot can achieved are represented as templates in BSCW. The user can therefore instanciate from the appropriate folder the tasks he wants to be achieved by the robot, or by a team of operators including the robot. In a dedicated folder, the robot download the results of the execution.

5.2.2 3D Virtual World

BSCW is a very rich and powerful CWE provided by Fraunhofer Institute in Germany and used by the FP6 European project eCoSpace. The integration between OpenSimulator and BSCW was done with the support of the eCoSpace consortium, where HP participates.

One of the functionalities of the BSCW collaboration environment is related to the concept of task and allows one member of the CWE to create a task and assign it to another member for execution. Each task has a name, a description, a status, a list of participants and may have several input and output parameters.

In the second year demonstrator the HRP-2 robot was already integrated with BSCW so that it can periodically check its task list and execute the assigned tasks. The idea behind this final integration is to provide a simple way for creating a task in BSCW with position and orientation parameters corresponding to the desired robot's target position.

The environment in which the robot operates is modeled in the OpenSimulator virtual world, by replicating dimensions, size and proportions for both the room and each object housed within it. Figure 17 shows a view of the first environment created in OpenSimulator for testing this integration, with markers on the ground and a couple of tables which reproduce the real environment in AIST lab (prior to moving in a new building, no pilors any more).

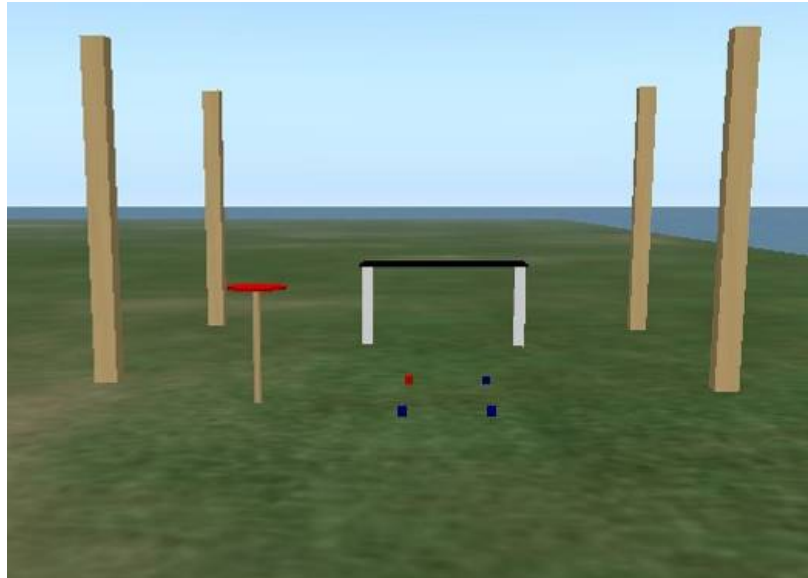


Figure 17 - The first version of an OpenSim environment for BSCW integration

The OpenSim-BSCW integration allows creating a task which directs the robot to move in a specific position and perform a given action. The robot's target position is specified using task input parameters. Values for those parameters are retrieved from the virtual world by moving the OpenSim avatar in a position which corresponds to the target position in the real world. Once the avatar is in the target position a command from the HRI triggers the creation of a task in BSCW. The C# code for OpenSim-BSCW integration is in the same REST plugin already described in section 4.2. and shown in Figure 13.

The creation of a task in BSCW is triggered by the RCP command USE OBJECT(9), where the object with ID=9 is predefined in the environment as a camera. The predefined strategy (i.e. method) for the camera object is "take photo" and when the HRI sends the command SELECT STRATEGY "take photo", the OpenSim plug-in activates the code which invokes the BSCW "create task" web service with the current avatar position and orientation as parameters.

5.2.3 Humanoid Robot Software Toolkit (HRST)

The Humanoid Robot Software Toolkit is a collection of 28 packages implementing the functionalities provided by the CORBA servers described in Figure 16. This software toolkit includes 3 main parts:

robotpkg (available at <http://www.laas.fr/~mallet/robotpkg>) which is a compilation framework and packaging system for building robotics software. This allows us to handle third-party open-source packages. Among those omniORB provides the CORBA implementation. OpenRTM provides an additional synchronization layer allowing independent componentization based on ports formalism and explicit scheduling classes. OpenHRP is providing an application framework which includes dynamic simulation, collision detection, and the client managing the control layer of HRP-2.

HPP standing for Humanoid Path Planner is a framework which includes several libraries to plan paths for the HRP-2 humanoid robot. It relies on several frameworks and the most noticeable is KineoWorks which provides basic path planning algorithms. In addition, this

framework includes a dynamic library, the pattern generator (6) and the Stack-Of-Stacks (3). A description of HPP is currently submitted to the Journal Of Software Engineer for Robotics. The additional packages which constitutes HRST in addition to robotpkg and HPP are specifically dedicated to interact with BSCW.

6. Conclusions

This deliverable has reported how the various software components developed by partners of the Robot@CWE project have designed, developed and coordinated to show how IST robots can be integrated into CWE environment for collaboration with humans in real work situations. The two resulting demonstrators target human-robot collaboration in realistic working environments such as those defined in deliverable D1.4: a space environment and a construction environment. Both demonstrators integrate technologies and components developed by different partners, thus representing the result of a collaboration effort towards the common project goal.

The resulting demonstrators have been described in this deliverable from a software components and integrator point-of-view. A more detailed description of each demonstrator can be found in the Deliverable D4.1-3.

7. References

- [1] J. Nielsen, (1993) - *Iterative User-Interface Design*. In Computer 26, 11 (Nov. 1993), 32-41. DOI= <http://dx.doi.org/10.1109/2.241424>
- [2] T. H. Song, J. H. Park, S. M. Chung, S. H. Hong, K. H. Kwon, S. Lee, and J. W. Jeon - *A study on usability of human-robot interaction using a mobile computer and a human interface device*, in MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services. New York, NY, USA: ACM, 2007, pp. 462–466.
- [3] J. Wachs, U. Kartoun, Y. Edan, and H. Stern - *Real-time hand gesture telerobotic system using the fuzzy c-means clustering algorithm*, in Proceedings of the 5th Biannual World Automation Congress, WAC 2002, vol. 13, 2002, pp. 403–409
- [4] T. Ayres, B. Nolan (2004) - *Voice activated command and control with java-enabled speech recognition over wifi*, in PPPJ '04: Proceedings of the 3rd international symposium on Principles and practice of programming in Java. Trinity College Dublin, 2004, pp. 114–119
- [5] M. Micire, J.L. Drury, B. Keyes, and H.A. Yanco - *Multi-touch interaction for robot control*, in IUI '09: Proceedings of the 13th international conference on Intelligent
- [6] H.M. Sadeghi, H. Bastani, and E. Azarnasab, - *IUTMicrobot: Robocup Rescue Robot League Competition Awardee Paper*, 2003. [Online]. Available: <http://www.ecerc.org/iutmicrobot>
- [7] M.W. Kadous, R.K. Sheh, and C. Sammut - *Effective user interface design for rescue robotics*, in HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction. New York, NY, USA: ACM, 2006, pp. 250–257.
- [8] L. Zalud (2003) - *RoBrno: RoboCup Rescue Robot League Competition Awardee Paper*.
- [9] J. Nielsen (1994) - *Usability inspection methods*. In Conference Companion on Human Factors in Computing Systems (Boston, Massachusetts, United States, April 24 - 28, 1994). C. Plaisant, Ed. CHI '94. ACM, New York, NY, 413-414
- [10] http://en.wikipedia.org/wiki/Virtual_world
- [11] <http://mjm-labs.com/viewer/>
- [12] <http://liferain.com/downloads/primcomposer/>
- [13] <http://www.mono-project.com/>

- [14] Robot@CWE, Deliverable D3.1, mostly in Annex II.
- [15] O. Stasse, R. Ruland, F. Lamiroux, A. Kheddar, K. Yokoi and W. Prinz, *Integration of humanoid robots in collaborative working environment: a case study on motion generation*. Intelligent Service Robotics : Springer Verlag, 2009, Vol. 2., no. 3, pp. 1861-2276.
- [16] O. Stasse, P. Evrard, N. Mansard and A. Kheddar, *Fast foot prints replanning and generation during walking in physical human-humanoid interaction*. IEEE, Int. Conf. on Humanoid Robotics., 2009. p. to appear.
- [17] N. Mansard, O. Stasse, P. Evrard and A. Kheddar, *A Versatile Generalized Inverted Kinematics Implementation for Collaborative Working Humanoid Robots: The Stack of Tasks*. International Conference on Advanced Robotics, 2009.
- [18] T. Foissotte, O. Stasse, A. Escande, P.-B. Wieber and A. Kheddar, *A Two-Steps Next-Best-View Algorithm for Autonomous 3D Object Modeling by a Humanoid Robot*. Intl. Conf. on Robotics and Automation, 2009. pp. 1159--1164.
- [19] T. Foissotte, O. Stasse, P. Evrard and A. Kheddar, *Using NEWUOA to drive the autonomous visual modeling of an object by a Humanoid Robot..s.l.* : Int. Conf. on Information and Automation (ICIA), 2009. pp. 78--83.
- [20] <http://ostatic.com/blog/an-introduction-to-opensim-the-apache-of-virtual-worlds>
- [21] P. Pierro, D. Hernández, M. González-Fierro, L. Blasi, A. Milani, C. Balaguer, *Humanoid teleoperation system for space environments*, The 14th International Conference on Advanced Robotics (ICAR 2009), Munich Germany, 22-26 June 2009.

8. Appendix

This appendix describes in more detail the commands added to the RCP protocol.

The new commands are the following:

- GOTO <location> / GOTO OBJECT(<object_id>)
- GRAB OBJECT(<object_id>)
- DROP OBJECT(<object_id>)
- USE OBJECT(<object_id>)
- SELECT STRATEGY FOR <cmd instance id> [<string>, <string> ... <string>]
- USE STRATEGY FOR <cmd instance id> <string>

8.1. Goal-setting / approaching an object

The “go there” command syntax is:

GOTO <location>

Where <location> can be

- <x> <y> coordinates referring either to a single point in the plane or to a cell in a grid
- the name of an object

To be able to specify the name of an object a name->coordinates translation must be done referring to a table configured in advance.

Note that if a concrete object, say a table, has a location identified by coordinates say (tx, ty), the robot cannot go to the same coordinates occupied by the object!

Thus “GOTO <location>” must be interpreted as “goto near location, possibly facing it”; but how far from the object is “near”?

To solve this doubt a new configuration parameter *stop_distance* has been defined, which indicates at which distance from the target object the robot must stop. It can be also interpreted as “distance from the target object where the object can be considered as reached”.

The *stop_distance* parameter will have for now a fixed value, which will be specified later. Semantics of GOTO <location> is that the robot stops at the given distance (*stop_distance*) from the target location, facing it.

The path from the current position to the target one for the demo will be decided autonomously by the robot.

What is needed/planned for the demo is to have the (always-on) robot’s object recognition feature which overlays on the object’s image in the video a “blob” with a numeric identifier, thus an object can be represented by a number.

To avoid confusion between GOTO <x> (wrong command because <y> is missing) and GOTO <object_id>, the decision is to mark the object_id number representing an object (the

number shown in the video overlay by the object recognition component) with the following syntax:

```
OBJECT(<object_id>)
```

Therefore the goal-setting command to be implemented for the demo becomes:

```
GOTO OBJECT(<object_id>)
```

Possible answers to this command from the robot to the HRI are:

```
OK COMMAND <cmd instance id> STARTED  
OK COMMAND <cmd instance id> COMPLETED  
OK COMMAND <cmd instance id> INTERRUPTEDBY <cmd instance id>  
KO UNKNOWN COMMAND <offending command>
```

8.2. Grabbing/dropping an object

The same syntax used to indicate an object for the “go there” command can be used for the “grab object” and “drop object” commands, which are defined as follows:

```
GRAB OBJECT(<object_id>)  
DROP OBJECT(<object_id>)
```

where <object_id> is the number² with which the robot’s object recognition component identifies the object.

Possible answers to this command from the robot to the HRI are:

```
OK COMMAND <cmd instance id> STARTED  
OK COMMAND <cmd instance id> COMPLETED  
OK COMMAND <cmd instance id> INTERRUPTEDBY <cmd instance id>  
KO UNKNOWN COMMAND <offending command>  
KO ERROR OUTFREACH  
OK COMMAND <cmd instance id> NEEDMOREINFO
```

Two new answers have been introduced for the GRAB command:

- “KO ERROR OUTFREACH” indicates to the client that the GRAB command has failed because the object is still too far from the robot to be grabbed; this can happen for example if the *stop_distance* parameter has been set to a value which is too high, e.g. higher than the robot’s arm length
- “OK COMMAND <cmd instance id> NEEDMOREINFO” indicates that the requested command can be performed using several alternative strategies and a human decision is needed to select the best strategy (see section **Erreur ! Source du renvoi introuvable.**)

² Uniqueness of this number with respect to different objects or to the same object at different times, even within the same session, is yet to be analyzed.

8.3. Strategy selection request

When more than one strategy is possible for executing a given task, the robot sends³ the following command to the HRI to request a human decision.

```
SELECT STRATEGY FOR <cmd instance id> [<string>, <string> ... <string>]
```

The strategy selection request has a reference to the command (<cmd instance id>) for which the execution alternatives have been identified.

Each possible alternative in the list is represented by a double-quote-limited string (e.g. "option X") describing it.

The same string will be displayed in the HRI for human user's selection.

A string is also quite general as any datatype can be represented through a string; hence the command may be reused in other situations, for example path selection for a GOTO command.

Possible answers to this command from the HRI to the robot are:

```
OK COMMAND <cmd instance id> RECEIVED  
KO UNKNOWN COMMAND <offending command>
```

A new answer has been introduced for the SELECT STRATEGY command. COMMAND RECEIVED is a simple acknowledge, indicating that the command has been received and understood and it was well formatted; otherwise the KO UNKNOWN COMMAND answer will be used.

8.4. Strategy indication

When the human user has selected one of the indicated strategies the HRI communicates this selection to the robot with the following command:

```
USE STRATEGY FOR <cmd instance id> <string>
```

The strategy selection indication has a reference to the command (<cmd instance id>) for which the execution alternatives have been proposed. It is the same command id referenced in the corresponding SELECT STRATEGY command.

The string indicating the selected strategy (e.g. "option 2") will be one of the strings describing the proposed alternatives and listed in the SELECT STRATEGY command.

When the strategy string is a complete and unambiguous description of a strategy it is acceptable that the indicated strategy is a new one wrt what has been listed in the SELECT STRATEGY command. This may happen for example, even if not in this demo, in case the SELECT/USE STRATEGY dialog is used to decide which path to use for a GOTO command.

³ This is not the first example of a command sent from the robot to the HRI, see the Positioning subprotocol description in D3.3 section 10.8.

Note that this command, coupled with the USE OBJECT and SELECT STRATEGY commands, allows for an easy object-oriented dynamic extension of the command protocol, which will be demonstrated in the last review demo. When the operator asks the robot to USE an OBJECT, the robot can answer with the list of methods / tasks it knows how to execute on the given object; then the SELECT STRATEGY command indicates which of the listed methods to execute on the previously referenced object. If the method / task is not yet known to the robot, the task can be learned and then selected without modifying the protocol.

Possible answers to this command from the robot to the HRI are:

```
OK COMMAND <cmd instance id> RECEIVED
KO UNKNOWN COMMAND <offending command>
KO UNKNOWN STRATEGY <string>
```

A new answer has been introduced for the USE STRATEGY command. KO UNKNOWN STRATEGY indicates either that the robot didn't understood the proposed strategy or that the strategy is not valid.

This KO answer concludes the strategy dialog and indicates that the original command (the command indicated by <cmd instance id> in the USE STRATEGY indication) will not be performed and it must be repeated, if needed, thus possibly starting a new strategy dialog.

After a successful USE STRATEGY command has been received and acknowledged, the robot will start executing the original command using the indicated strategy.

When starting and completing the execution of this original command the robot informs the HRI with the usual answers

```
OK COMMAND <cmd instance id> STARTED
OK COMMAND <cmd instance id> COMPLETED
```

where <cmd instance id> is referred to the original command (the same referenced by USE STRATEGY).

8.5. Using an object

Command syntax is:

```
USE OBJECT(<object_id>)
```

Where <object_id> has the same meaning as in the already defined GRAB OBJECT command.

Possible replies are:

```
OK COMMAND <cmd instance id> STARTED
OK COMMAND <cmd instance id> COMPLETED
OK COMMAND <cmd instance id> INTERRUPTEDBY <cmd instance id>
```

KO UNKNOWN COMMAND <offending command>
KO ERROR OUTFREACH
KO NO TASKS AVAILABLE
OK COMMAND <cmd instance id> NEEDMOREINFO

The “KO NO TASKS AVAILABLE” reply is sent when the object cannot be used for any task known by the robot. Other replies have the same meaning as in the GRAB OBJECT command. More precisely, when the robot knows at least one task that can be performed using the object it sends a NEEDMOREINFO reply. This is true even if only one task is available, because it allows the operator to confirm task execution (in case the task known by robot is not the one expected by the operator).

After sending a NEEDMOREINFO reply the robot sends a SELECT STRATEGY command listing available tasks as the strategies. This command works just like in the GRAB command. After the operator selects a task by means of the USE STRATEGY command, the robot starts executing it.

The following is an example dialog for the robot using an object. Each command is prefixed by “U” (user) when sent from the HRI to the robot, “R” when sent from the robot to the HRI.

U: USE OBJECT(1)
R: OK COMMAND U1 NEEDMOREINFO
R: SELECT STRATEGY FOR U1 [“task 1”, “task 2”, “task 3”]
U: OK COMMAND R1 RECEIVED
U: USE STRATEGY FOR U1 “task 1”
R: OK COMMAND U2 RECEIVED
R: OK COMMAND U1 STARTED
R: OK COMMAND U1 COMPLETED