SAPHARI

SAFE AND AUTONOMOUS PHYSICAL HUMAN-AWARE ROBOT INTERACTION

Project funded by the European Community's 7th Framework Programme (FP7-ICT-2011-7)

Grant Agreement ICT–287513

Deliverable D6.4.1

Interface framework for multimodal safe interaction

Deliverable due date: 1 November 2013	**Actual submission date:** 01 February 2014
Start date of project: 1 November 2011	**Duration:** 48 months
Lead beneficiary for this deliverable: DLR	**Revision:** Final

Nature: P	**Dissemination Level:** CO
R = Report	PU = Public
P = Prototype	PP = Restricted to other programme participants (including the Commission Services)
D = Demonstrator	RE = Restricted to a group specified by the consortium (including the Commission Services)
O = Other	CO = Confidential, only for members of the consortium (including the Commission Services)

www.saphari.eu

Introduction

In Milestone 6 – "Conceptual Design of Interface Framework" DLR proposed an Interface Framework which allows for advanced control of multimodal interaction modalities in human-robot collaboration scenarios. This deliverable shows the results of the efforts that emerged from that concept.

The goal was to allow intuitive, flexible and efficient use of a robot also for users who are not, or at varying level of expertise, familiar with robot programming and/or interaction. For quickly analyzing various interaction modalities, we also aimed for quick integration of new features and devices, so they can be easily bundled, compared and/or combined for further analysis. In particular, our framework allows the simple and seamless integration of different interaction channels into new tasks, i.e. it provides multimodal communication as part of the programming paradigm and not only consider the general benefit of certain modalities regarding for human-robot interaction (HRI).

Table of contents

Introduction 1

1. System Overview 3

2. System Components 3
   - Robot 3
   - Human Observer 4
   - Interaction Feedback Planner 4
   - Programming 7

3. Video 8
   - Part 1 – Input examples 8
   - Part 2 - Output examples 9
   - Part 3 - Combination example 9
   - Part 4 - Combination example - BRIO 10

4. Conclusion 10

1. System Overview

DLR's Interface framework consists of the components Robot, Human Observer, Interaction Feedback Planner and the integrated programming environment (IDE) "Bubbles". All components communicate via the open-source robot-middleware ROS. Additionally, we model the interacting humans as another component of the system and consider them as the central entity of our system. There are two distinct roles, as which a human can be involved: as a user interacting directly with the robot or as a programmer interacting with the robot on a more advanced, but still intuitive level of abstraction. In Fig. 1 the general data-flow between the entities is depicted.

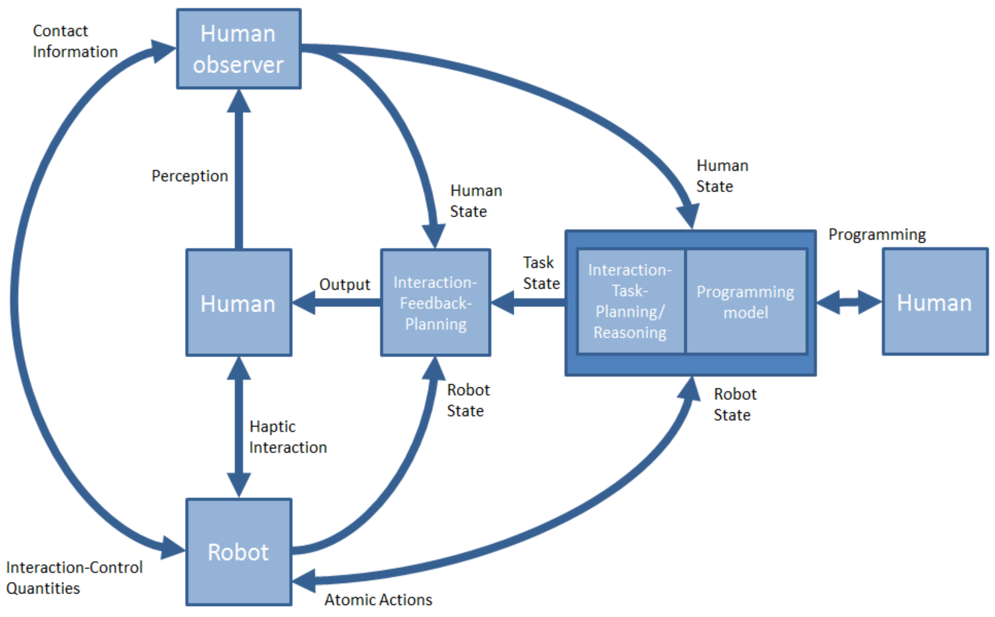


Fig. 1: The data flow diagram of the Interaction Framework

By this architecture a flexible and independent framework is realized whereby any component can run without the direct need of the others. That allows for developing a robot-task and interaction capabilities separately and fast adjustment of the behavior of one without the need of changing the other.

2. System Components

Robot

The Robot-component includes the manipulator hardware and its controlling unit RCU (Robot Controlling Unit). The RCU has several possibilities of interfacing. However, in this context mainly the action-interface (which is compatible to ROS action_libs) is of importance. It allows sending actions/commands from the IDE to the robot and offers a vast variety of important internal state parameters, as e.g. collision-state, velocity, brake-state, etc. in return.

Human Observer

To collect and process human related data in the system, the observation and interpretation unit “Human Observer” has been implemented. This collects all available information from sensors and software nodes that provide HRI relevant data. Such data would e.g. be skeletal tracking, gesture recognition, voice recognition, human intention models, contact states, contact forces, or human-attentiveness. In the current implementation, skeleton tracking from a Microsoft Kinect, the gesture recognition of partner TUM, several distance metrics (e.g. hand to end-effector distance, out of/in workspace), the ID of the currently interacting user and an estimated hand-over position where the robot could hand-over items from/to the human are collected and processed into an abstract “Human State”. The node has been developed to allow fast and easy integration of new sources of information.

All detectable humans are represented as agent systems. In case of multiple users being present, the human-observer chooses the currently active interacting user based on contextual and, if available, individual information. The simplest way for this decision is to select the person which is spatially closest to the robot. A more advanced choice could for e.g. include human attentiveness, identity, and interaction history. In addition to the abstract Human State representation, the Human Observer provides also safety-relevant information at fastest possible rates. Real-time collision avoidance algorithms, may receive the positions of human segments such that the robot is able to promptly react to this labeled proximity information and avoid collisions accordingly.

Interaction Feedback Planner

The Interaction Feedback Planner (IFP) acts as a highly flexible structure which function it is to map Human, Task and Robot State to an adequate, sensible and interpretable human perceivable output on different levels of abstraction and on a multitude of available communication channels. Figure 2 depicts its interface and examples of attached communication channels. The IFP contains a set of rules, which determine how a specific state or message is presented to the human. Therewith, the feedback to the human could be decoupled from the other components in the system. Especially, getting a strict decoupling from the task implementation is of importance. This has two advantages: On the one hand it allows for a holistic interaction-“look and feel” which does not depend on the currently running task, but is more or less identical every time the system is used. On the other hand this system allows very fast exchanging and adapting of the used interaction modalities. This can be used for systematical analyses on the usefulness of certain human-robot-interaction schemes.

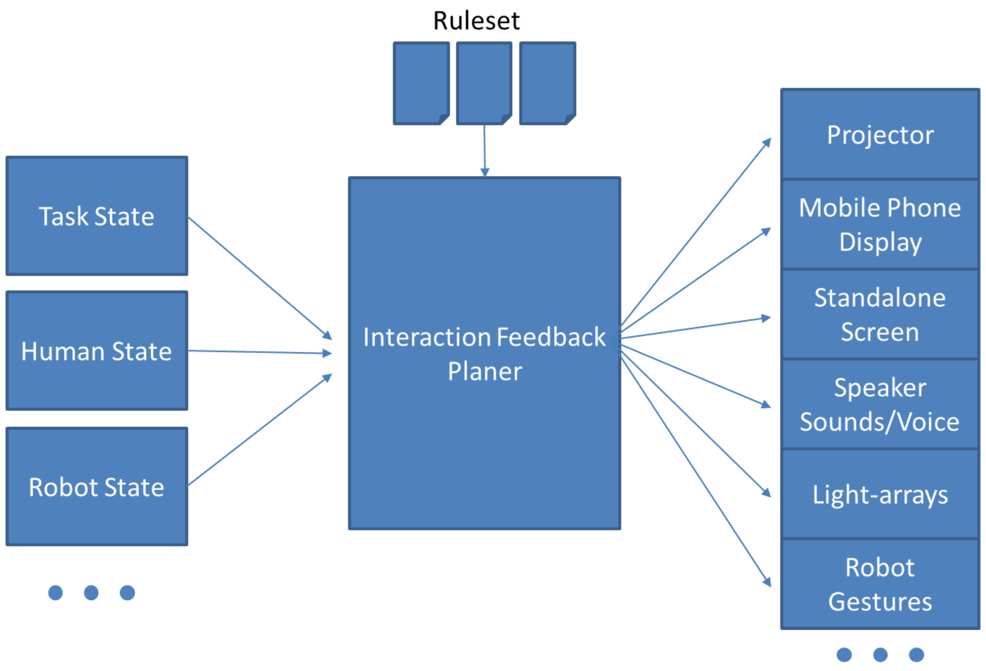


Fig. 2: The Interfaces of the IFP: examples of input elements, output devices and rule sets defining the relation between input and output

The IFP consists of four layers which provide, input, output and logical interconnection of input and output as well as a detailed description of feedback dependency on logical states, priorities and device-specific parameterization of expression or classes of expressions (Fig. 3). The logical interconnection of input and output and the final feedback are described by XML-files and are thereby flexible to modify and can be exchanged very fast. The description of IO-interconnection is realized by conditions and consequences that are combined in a rule. All rules and consequences together describe the behavior of the IFP.

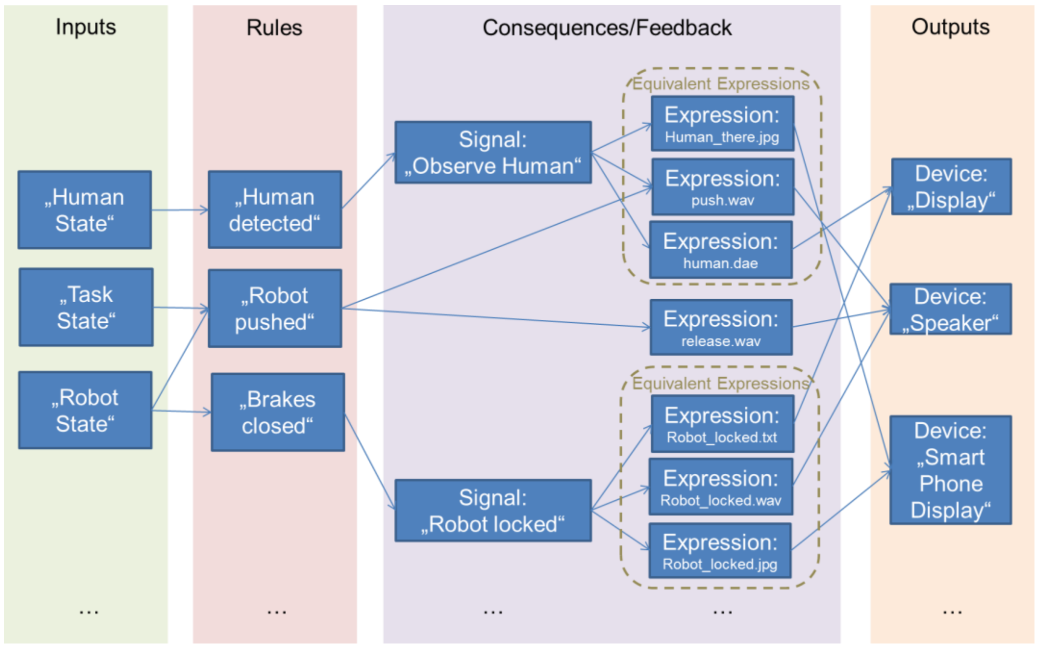
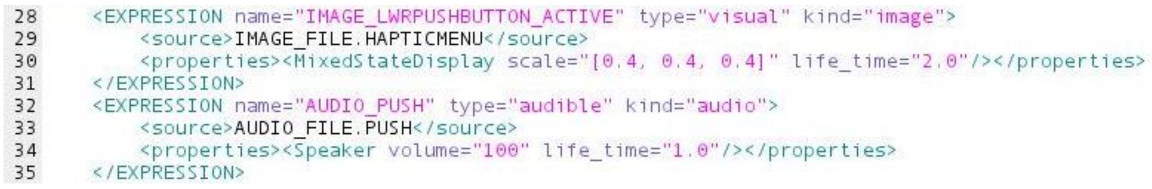


Fig. 3: Example structure of the interconnection of the two outer layers by the inner layers (rules and consequences).

In the following the definitions of expressions, signals and rules as well as samples of XML-implementations are given. **Expressions** represent a single atomic message and its parameters for a specific device, e.g. an image, a string, or an audio-file.

The XML definition of expressions looks like this:


```
<EXPRESSION name="IMAGE_LWRPUSHBUTTON_ACTIVE" type="visual" kind="image">
    <source>IMAGE_FILE.HAPTICMENU</source>
    <properties><MixedStateDisplay scale="[0.4, 0.4, 0.4]" life_time="2.0"/></properties>
</EXPRESSION>
<EXPRESSION name="AUDIO_PUSH" type="audible" kind="audio">
    <source>AUDIO_FILE.PUSH</source>
    <properties><Speaker volume="100" life_time="1.0"/></properties>
</EXPRESSION>
```

In the code above, the expressions used for the human-robot haptic feedback in the attached video are described. An image with its scaling and life time properties for the “MixedStateDisplay”-device (Top-screen) can be seen as well as the audio feedback for pushing the robot with volume and life time properties for the “Speaker” device.

The **signal** is a construct to combine expressions transporting an equivalent message to the user. This can be used if there are multiple ways of getting information to the user (e.g. printing a text on a screen, speaking the same text via voice-synthesis or displaying of an iconic presentation of the message). The selection of which channel to use can be influenced by priorities, available output-devices as well as complex decision strategies like preference/expertise of an identified user.

Example of signal definition in XML:

```
<SIGNAL name="POSE_3">
    <EXPRESSION name="TEXT_POSE_3"/>
    <EXPRESSION name="ANDROID_TEXT_POSE_3"/>
    <EXPRESSION name="AUDIO_POSE_3"/>
</SIGNAL>
<SIGNAL name="ROBOT_LOCKED">
    <EXPRESSION name="TEXT_ROBOT_LOCKED"/>
    <EXPRESSION name="ANDROID_IMAGE_ROBOT_LOCKED"/>
</SIGNAL>
```

In the code above, the signal definition for “ROBOT_LOCKED” and the “POSE_3” are described. The expressions combined to one signal are transporting an equivalent message to the user via different output channels.

Rules in the IFP are used to connect informations on the input-channels with expressions or signals on the output-channels via conditions. The rule also determines if the output is activated at the beginning (rising) or the end (falling) of the Boolean value of the condition or as long as the condition is true (holding).

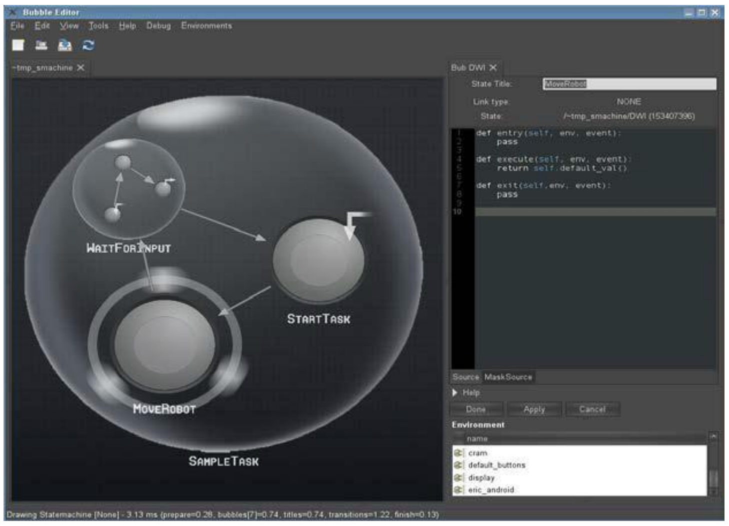
Example of rule definition in XML:

```
<RULE condition='"LWRPushButton_incontact" in TASK_STATE'>
    <EXPRESSION name="AUDIO_PUSH" trigger="rising"/>
    <EXPRESSION name="AUDIO_RELEASE" trigger="falling"/>
</RULE>
<RULE condition='"LWRPushButton_active" in TASK_STATE'>
    <EXPRESSION name="IMAGE_LWRPUSHBUTTON_ACTIVE" trigger="holding"/>
</RULE>
<RULE condition="any([HUMAN_STATE.states[h_id].state == HUMAN_STATE.IW for h_id in range(10)])">
    <EXPRESSION name="TEXT_HUMAN_IN_WORKSPACE" trigger="holding"/>
</RULE>
```

In the code-snippet above definition of rules can be seen. As long as the element “LWRPushButton_active” is in the “TASK_STATE” the condition of the second rule will be true and the expression assigned with a trigger “holding” is expressed. The first rule gives an example how the rising and falling of a rule-condition can be used to generate feedback. In this case the sound while pushing and releasing the haptic-gesture “LWRPushButton” is described, which is an important cue for the user in physical interaction with the robot. With this the human knows when he pushed hard enough for the robot to detect the haptic-gesture and he can decrease the force. This example turned out to make haptic interaction far more intuitive, robust and also faster than without sound feedback.

Programming

The programming environment developed at DLR is a python based IDE called “Bubbles” (see Fig. 4). With Bubbles the user can create so called skills by writing Python-scripts and combining existing skills to complex tasks. The aim of this was to provide a well-designed programming interface which imposes all relevant constraints to ensure both, simplicity and powerful programming models. Therefore, a simple to learn visual programming and more powerful scripting was combined. This approach gives trained users the freedom to program the system on all relevant levels of abstraction, and at the same time allows intuitive visual creation of new tasks also for untrained users. Functionalities to manage programs, generate, reuse and reconfigure skills and connect to external components are provided. In Bubbles states can be modified or created dynamically while the program is running. Hereby, programming becomes a real interaction with the system and this allows for faster programing and shorter evaluation cycles. Regarding the integration into the rest of the Interface Framework, Bubbles publishes the hierarchical path of currently active states as Task State.



State Title: MoveRobot
Link type: NONE
State: /~tmp_smachine/DWI (153407396)

```
def entry(self, env, event):
    pass

def execute(self, env, event):
    return self.default_val()

def exit(self,env, event):
    pass
```

Environment

name
cram
default_buttons
display
eric_android

Drawing Statemachine [None] - 3.13 ms (prepare=0.28, bubbles[7]=0.74, titles=0.74, transitions=1.22, finish=0.13)

Fig. 4: The intuitive and dynamic IDE "Bubbles"

3. Video

Part 1 – Input examples

In the first part of the video several examples of input modalities of the Interface Framework are presented. First the dynamic programing of a very simple task in an intuitive drag'n'drop manner is shown (Fig. 5 left). Here the Bubbles IDE is used to move the robot to an initial position and continues to move between three positions in a loop. It can be seen that the IFP immediately shows the state of the task (1-2-3) on the Top-screen without additional implementation effort (Fig. 5 right). After that a human enters the workspace and is observed by the Human Observer (Fig. 6 left). The Human State (In-Workspace) is then immediately presented on the screen to ensure the human that he is recognized by the system. The human then gets closer to the robot, stops it by touching and afterwards activates the emergency brakes. This change of the Robot State is again presented by the IFP (Fig. 6 right).

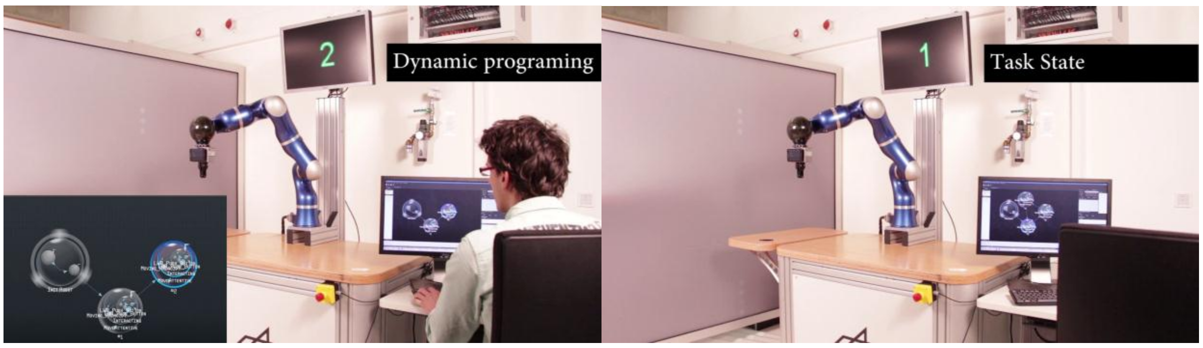


Fig. 5: Programing a task in «Bubbles« (left). Top-screen expressing Task State (left and right)

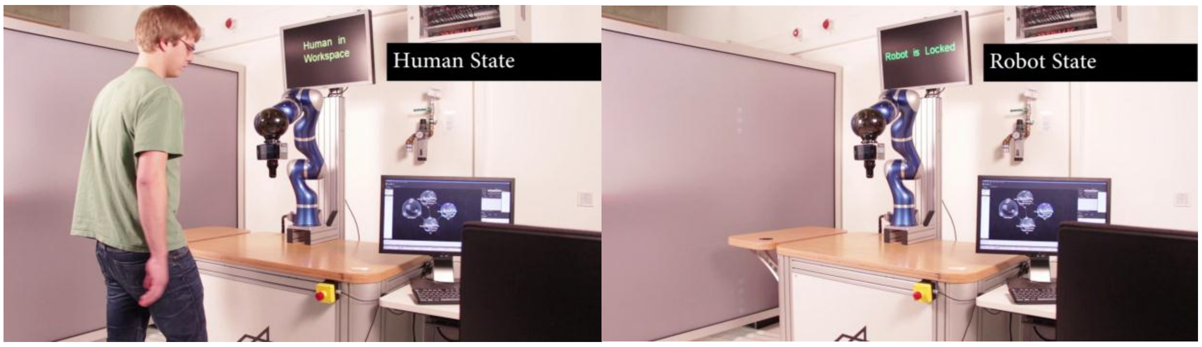


Fig. 6: Top-screen expressing «Human-In-Workspace» (left) or «Robot-Locked» (right)

Part 2 - Output examples

In this part, several possible output modalities are shown. The system executes the same "3-position-loop" Bubbles-program as before. The IFP is running with a different rule set loaded, so the output channel changes for each of the Task States. The state "moving to position 1" is presented on the Top-screen (Fig. 7 left), "moving to position 2" utilizes a voice output on the speakers of the system while "moving to position 3" is presented on the robot attached screen (Fig. 7 right).

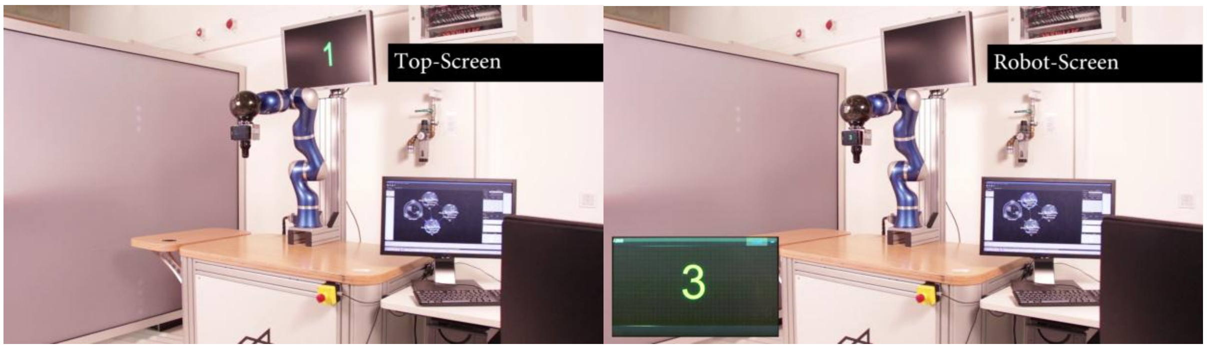


Fig. 7: Displaying information about the Task State on the Top-screen (left) or the Robot-screen (right)

Part 3 - Combination example

This part shows a more complex IFP-rule set. There are three rules loaded:

1. Human State (In-Workspace) on Top-screen
2. Task State & Robot State (Interaction-possibilities) on Robot-screen
3. Task State (positions) on all output channels

These rules are evaluated with a decreasing priority, so that if no other event occurs the 3rd rule uses all channels to present the position of the robot. If one of the other rules is activated it overrides rule 3, which leads to showing the "human in workspace" information on the Top-screen and the "interaction" information on the Robot-screen (Fig. 8).

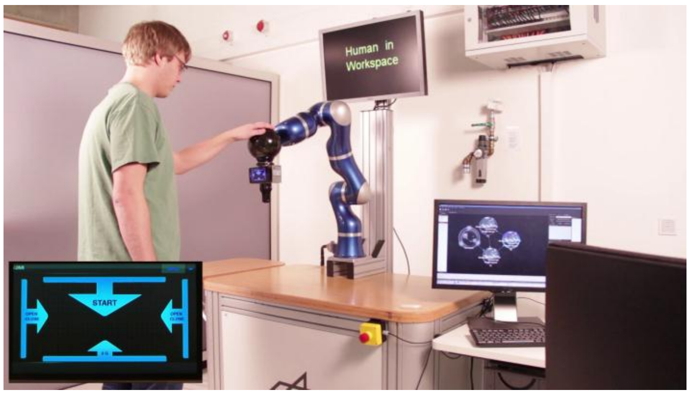


Fig. 8: Combination of multiple input and output channels to achieve sophisticated multimodal interaction

Part 4 - Combination example - BRIO

The last part of the video shows combined interaction modalities in the BRIO-demonstration. The Interaction Framework is used to facilitate rich and intuitive interaction by touch, robot-attached screens, virtual fixtures and sound in a complex human-robot joint task (Fig. 9).

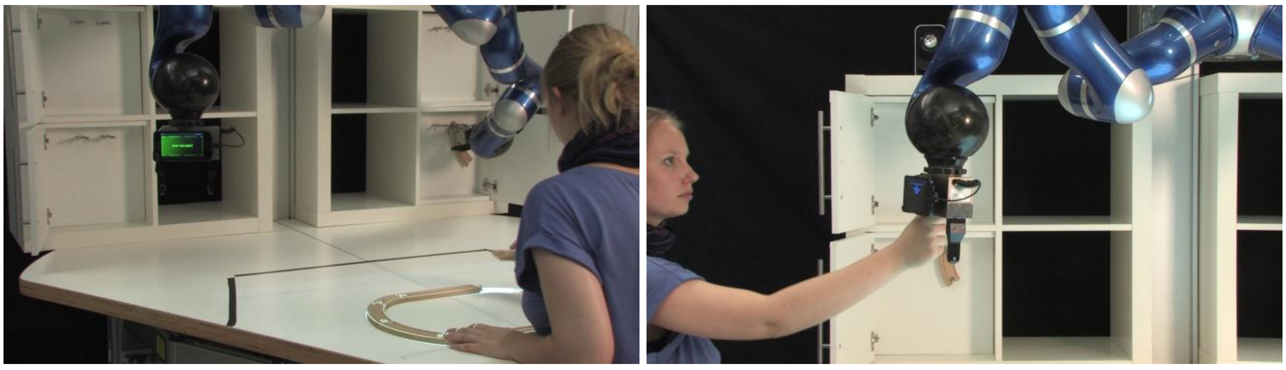


Fig. 9: Interaction examples from BRIO task

4. Conclusion

DLR implemented the Interface Framework proposed in MS6. Therefore a prototype version of the interaction feedback planner (IFP) was implemented. The DLR-IDE “Bubbles” was developed further with focus on stability, intuitiveness and expanded to communicate its internal Task State, which is needed as input for the IFP. Also the already existing Human Observer component was enhanced to meet the requirements of the target system.