Modular state-based behavior control for safe human-robot interaction: A lightweight control architecture for a lightweight robot

Sven Parusel, Sami Haddadin, and Alin Albu-Schäffer

***Abstract*— In this paper we present a novel control architecture for realizing human-friendly behaviors and intuitive state based programming. The design implements strategies that take advantage of sophisticated soft-robotics features for providing reactive, robust, and safe robot actions in dynamic environments. Quick access to the various functionalities of the robot enables the user to develop flexible hybrid state automata for programming robot behaviors. The real-time robot control takes care of all safety critical aspects and provides reactive reflexes that directly respond to external stimuli.**

I. INTRODUCTION

In recent years robots have gained various new capabilities in soft-robotics control and motion generation. This progress makes it possible to physically interact, share a common workspace, and even directly collaborate with humans.

An interesting first work in this respect is e.g. given in [1], where a design of safety monitors for multifunctional robotic systems is considered. In [2] a reflex based approach for robotic arms is introduced. High-level commands that base on low-level reflexes are shown and e.g. used, for collision avoidance behaviors. In [3], [4] a human-friendly control scheme is presented, which uses gravity compensation to prevent human injury after collisions. To make human robot interaction more safe, [5] proposed an algorithm, which makes robot behavior predictable for a human operator. In [6] environmental, as well as human behavior and reaction observation is presented in order to improve the safety of human robot-interaction. [7] introduces a high-level robot programming approach, which uses sensory input to acchieve local autonomy.

Due to the diversity and complexity of these features and their sheer number it is non-trivial to design, implement and switch them consistently whithin complex robot tasks under the premise of ensuring safety to the human user **and** task execution. Although there is currently major effort in realizing safe robot control algorithms, the human safety on the control architecture side was not yet treated to a similar extent. For that reason we developed a control architecture, which contains and consistently combines a wide set of strategies for human safety and friendliness. The system provides easy and consistent access for robot task programming based on hybrid state machines.

In the line of cited research, the goal of our control architecture for human-friendly real-time robot control is to work as a lightweight platform for developing human-friendly robot tasks[1]. Our solution intends to relieve the high-level programming especially from the need to take care of the increasingly complex low-level safety control. Thus, we designed an encapsulated low-level control framework, which provides well defined interfaces at an intuitive yet, powerful level of abstraction to the task programmer. We support our results by various applications that demonstrate the approach on the DLR Lightweight Robot III (LWR-III). Creating a task for a robot is possible through abstract commands without detailed knowledge of low-level core parts of robot control. Especially the safety modalities of the robot are arranged and implemented in a very robust and non-accessible manner. Furthermore, it is possible to fuse all relevant information that is provided by the internal robot state or additional exteroceptive sensors to achieve complex behavior also on task level.

In order to enable such a design we develop a stable and modular robot control core that provides a locally autonomous fault tolerant behavior as far as the particular situation allows this. It is accomplished that errors in task programming or network communication do not necessarily lead to dangerous situations for humans or the robot.

For the presented concept we demand strict modularity through all levels of the system in order to allow faster development of core components and for reducing integration faults. These include especially controllers, motion generators, or safety methods. Furthermore, we show that our approach is able to support also the control of multi-robot setups.

The paper is organized as follows. Section II introduces and classifies the available methods for control and motion generation. Furthermore, we introduce our concept for operational and reflex behaviors for human-friendly operation and outline the concept of functional modes. Section III describes the developed system architecture, which introduces especially the systematic treatment of safety in a behavior based manner. Section IV outlines the graphical state based programming concepts that we designed particularly for being able to implement human-friendly behavior also on higher levels of abstraction. Finally, Sec. V concludes the paper.

II. BEHAVIORS AND FUNCTIONAL MODES

To create adequate behavior during human presence, the robot needs to be able to take multiple sources of internal and external (environment and human) information into account and act accordingly. Only full consideration of such knowledge enables the robot to work in a safe and yet efficient way. In order to equip the robot with robust behavior all levels of motion and behavioral control need to be able to respond in a flexible manner to unforeseen inputs and events. The basic motion generation and control schemes have to form an effective basis for implementing locally robust behavior so the next levels of abstraction can reliably built on that. For this we have developed numerous methods

S. Parusel, S. Haddadin and A. Albu-Schäffer are with German Aerospace Center, P.O. Box 1116, D-82230 Wessling, Germany sven.parusel@dlr.de, sami.haddadin@dlr.de

[1]The design we present here is the follow up and extension of the work done in [8].

for soft-robotics control and online trajectory generation with collision avoidance, which are described hereafter.

A. Methods

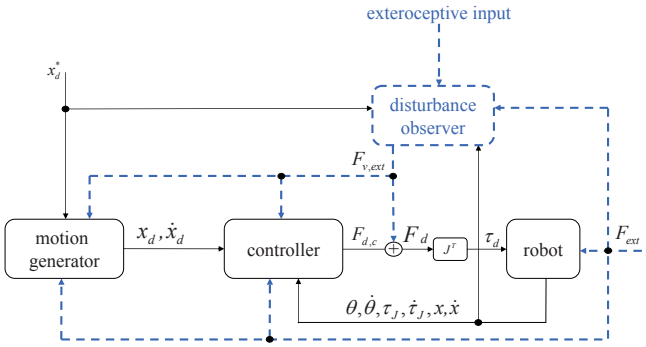

Fig. 1. Using disturbance inputs at different motion control levels for realizing effective disturbance response.

Figure 1 depicts a classical Operational space control loop consisting of

1) the motion generator for providing the desired reference motion in terms of generalized Operational space coordinates $\mathbf{x}_d$ as a general function of time,
2) the controller that provides the desired Operational space force, which is then transformed via the Jacobian transpose to desired torque commands, and
3) the physical robot that transforms the desired torque command via a low-level motor torque loop into motor torques that generate the respective robot motion.

A reactive disturbance response[2] can be fourfold:

1) Physical forces act on the robot and inherently produce a dynamic response of the robot.
2) The controller can implement a purely passive disturbance response (the measurement of external forces is not directly incorporated) with respect to external forces, or actively react to them (a classical example is inertia shaping). A combination of both is of course possible as well.
3) Furthermore, generalized virtual forces, generated e.g. by repulsive potentials, can act directly as a motor command input and add a respective behavior on the controller.
4) The motion generator provides motion commands that directly take into account the presence of physical or generalized virtual forces, leading e.g. to a collision retraction or reactive collision maneuvers.

Usually, disturbance reaction schemes act isolated in the sense that the particular response is exclusively carried out by a single scheme. However, in order to provide more sophisticated and situation dependent behavior, it is important to equip a robot with the capability to react on multiple levels of abstraction simultaneously. For this we developed various control and motion generation schemes over the last years that are able to process various sensorial inputs as described in the following.

1) Control algorithms: For the LWR-III there are numerous controllers available for both, Operational and joint space control. They include position control (C.1), torque control with gravity compensation (C.2), impedance control (C.3), and admittance control (C.4), see Tab. II-A.1. The different controllers enable us to feed different disturbance signals as physical or virtual forces[3].

2) Motion generation: Apart from different standard motion generators for operational and joint space (M.2), we developed several reactive motion generators as well. For example a reactive attractor-based algorithm (M.3) that is able to circumvent virtual object representations in real-time and at the same time retracts from external forces [14]. A particular method included in every motion generator is time scaling (M.4) [11]. For this a residual generator (generalized disturbance observer) monitors human presence in the workcell, proximity to the robot, and external as well as virtual forces acting on the robot. These quantities are used to scale the desired motion back and forth such that the geometric properties of the commanded trajectory are preserved, while collision avoidance and retraction is performed in a strictly task consistent manner (by scaling the time increment of the time generator). This enables the robot to slow down, stop the motion, and drive back along the desired path in a continuous and real-time manner. II-A.2.

From now on we denote the according space (joint or Cartesian) of a motion generator or controller by *.J* for joint space or *.C* for Cartesian space (or more generally Operational space).

3) Environmental disturbances: In order to incorporate external and virtual forces explicitly into the motion generation and control schemes, an accurate estimation of forces acting on the robot during physical contact and the geometric properties of the environment are needed. The according methods are introduced hereafter.

a) Estimation of external forces: The internal joint torque sensors in each joint of the LWR III and the available accurate dynamics model can be used to obtain detailed information about contacts that occurred between a robot and its environment [12], [11]. The measured torques are used in a nonlinear disturbance observer that estimates the

Controller	Virt. forces	Phys. forces	Reference
Joint position control (C.1)	×	×	[9]
Cartesian position control (C.1)	×	×	[9]
Torque control (C.2)	✓	✓	[10]
Joint impedance control (C.3)	✓	✓	[9]
Cartesian impedance control (C.3)	✓	✓	[9]
Joint admittance control (C.4)	✓	✓	[11]
Cartesian admittance control (C.4)	✓	✓	-

TABLE I
CONTROLLER AND LOW-LEVEL DISTURBANCE INPUT.

Motion generation	Virt. forces	Phys. forces	Reference
Stop (M.1)	×	×	[12], [11]
Non-reactive (M.2)	×	×	[12], [11]
Attractor based (M.3)	✓	✓	[13]
Trajectory scaling (M.4)	✓	✓	[12], [11]
Admittance based (M.5)	✓	✓	[12], [11]

TABLE II
MOTION GENERATOR AND LOW-LEVEL DISTURBANCE INPUT.

[2]Please note that we do refer to hard real-time reaction and not adaptation of via points that are e.g. provided by a global motion planner.

[3]Please note that we assume any generalized virtual disturbance signal to be a virtual force, i.e. every sensory input is somehow transformed into the force domain.

external torques and forces along the entire structure of the manipulator. Furthermore, a force-torque sensor can be attached to the wrist.

b) Virtual environment forces: For preventing the robot to collide with it's surrounding there are several strategies implemented, which work for static (e.g. table) and/or dynamic (e.g. human) parts of the environment. These are directly integrated into the low-level torque control and reactive motion schemes[4].

1) Virtual walls with adjustable compliance protect the static environment and the robot from colliding with it. These are also applicable in real-time and may vary with time.
2) So called virtual traps, which apply virtual forces to the robot in the sense of potential fields. They confine the manipulator similarly to virtual fixtures on specified spots or planes. They can be used for example to suspend the gravity compensated robot in order to avoid uncontrolled collisions. Thus, intuitive release strategies are available.
3) Advanced virtual geometric environment models can also be used to generate virtual forces at a slower rate (or even remotely). Such algorithms can be fed by dynamical data of objects or persons, which are located in the robot's vicinity. To gather data of dynamic objects any kind of additional external detection can of course be used. In our laboratory setup we utilize e.g. optical tracking via passive markers attached to the object, or various image processing approaches. A further possibility is to generate virtual objects at run-time from the input given by the external force estimation. This enables the robot to "remember" collisions and avoid them in its future course. This is realized by performing tactile exploration and then use the gathered information for collision avoidance.

Due to the non-trivial interaction between the aforementioned methods, a thorough concept for their combination and use is absolutely crucial. For unification of methods we define behaviors that consist of the relevant control and motion algorithms.

B. Operational and reflex reaction concept

In general, we design robot actions such that they are defined as

- **Operational behaviors:** a formal high-level parametrization of the robot capabilities that defines its particular motion, control, and safety properties. This fully determines the nominal motion control and disturbance response of a robot.
- **Reflexes behaviors:** a formal parametrization of a real-time reflex behavior of a robot that is associated with real-time activation signals. This represents either the indication of a certain stimulus or a fault[5]. Reflexes override the currently active behavior and execute a low-level strategy.

The formal definition of behaviors, as well as operational and reflex behaviors is described in the following.

[4]On http: www.safe-robots.com several videos show the particular schemes implemented on the LWR-III.

[5]Stimuli are general perception inputs, whereas faults are detected either by processed stimuli (observation of external torques, proximity information, ...) or general system malfunctions, as e.g. communication collapse or run-time violations.

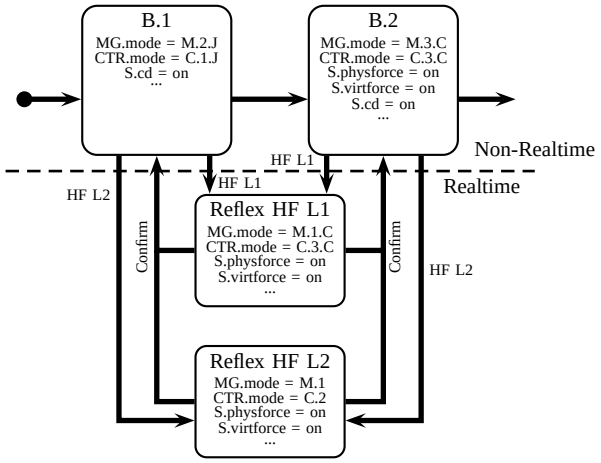

Fig. 2. A basic example depicting the combination of operational and reflex behaviors during a two-behavior nominal task. Besides the motion generator $mg \in MG$ and the controller $ctr \in CTR$ the important safety set parameters $\mathbf{s} \in S$ are depicted for each behavior. CD denotes the activation of collision detection and switching VirtForce/PhysForce to *on* enables the incorporation of virtual/physical forces.

1) Behaviors: A behavior $\mathbf{b} \in \mathcal{B}$ is defined as a $6-$tupel that is an element of

$$\mathcal{B} = (MG \times P_{MG} \times CTR \times P_{CTR} \times PR \times S), \quad (1)$$

where MG is the set of motion generators, P_{MG} the parameter set of MG, CTR the set of possible controllers, P_{CTR} the parameter set of CTR, and S the set of safety feature activation and parameterization. PR is the priority set. Therefore, each element $\mathbf{b}$ consists of a motion generator $mg \in MG$, the associated parameter vector $\mathbf{p}_{MG} \in P_{MG}$, a controller $ctr \in CTR$, the associated parameter vector $\mathbf{p}_{CTR} \in P_{CTR}$, a priority $\mathbf{pr} \in PR$ and a set of safety features $\mathbf{s} \in S$. Such a safety set contains activation signals and parameters for all defined safety features. It e.g. determines whether physical and/or virtual forces shall affect the activated motion generator and/or controller while the behavior is active. $\mathcal{B}_v$ is the subset of $\mathcal{B}$ including all valid combinations of mg,$ctrl$, and $\mathbf{s}$, while $\mathcal{B}_{inv} \in \mathcal{B}$ are the invalid combinations. An obvious example of a non-valid behavior would be a combination of a Cartesian motion generator and a joint controller. In nominal condition the according behavior is defined within $\mathcal{B}_v \subset \mathcal{B}$.

2) Operational behaviors: Operational behaviors $\mathbf{b}_{Op} \in \mathcal{B}_{Op} \subset \mathcal{B}_v$ are defined in the so called *operational space* (usually the non-realtime part of the system). They are used for the higher-level task composition and rely on data coming from sensors, actuators, and other real-time information sources. Operational behaviors close the loop via the low-level space.

In order to be able to rapidly (i.e. in real-time) respond to external or internal stimuli, a robot should be able to activate certain pre-defined behaviors within the fastest available control loop. We call this low-level space the *reflex space* (usually the real-time system) and the corresponding behaviors reflex behaviors. These behaviors act exclusively within the reflex space and are only enabled from the operational space, i.e. no loop closing takes place.

3) Reflex behaviors: Real-time reflex behaviors $\mathbf{r} \in \mathcal{R} \equiv \mathcal{R}_S \cup \mathcal{R}_F \subset \mathcal{B}_v$ build the set $\mathcal{R}$ of all conservative and task

abandoned behaviors. $\mathcal{R}$ is the union of $\mathcal{R}_S$ and $\mathcal{R}_F$, the stimuli and fault reflexes. They are used to appropriately react to the physical state of the system, e.g. during collisions. Useful examples for reflex strategies in case of collisions could be stopping the motion generation (M.1) and switch to torque control (C.2) for minimizing contact forces after contact. Another possible reaction is the use of the robot emergency brake[6].

Next, we introduce so called functional modes, which are directly related to certain subsets of $\mathcal{B}_v$

C. Functional modes

We generally distinguish between four major functional modes of the robot potentially working in human vicinity [15]:

1) **Autonomous task execution:** Autonomous mode in human absence
2) **Human-friendly mode:** Autonomous mode in human presence
3) **Collaborative mode:** Cooperation with human in the loop
4) **Fault and reflex reaction mode:** Safe fault behavior with and without human in the loop

In the **first** one the robot is autonomously fulfilling its given task without considering the human presence. The task is carried out under certain optimality criteria in order to increase productivity. The possible behaviors in this mode are $\mathcal{B}_{am} \subset \mathcal{B}_v$. The task has of course to be designed such that it does not collide with the cooperative mode. In the **second** mode the corresponding set of behaviors is $\mathcal{B}_{hf} \subset \mathcal{B}_v$. The set $\mathcal{B}_{hf}$ contains only behaviors, which are safe for the human. In the second and third mode we need a meaningful partition in task space, which subdivides the given workspace of the robot into regions of interaction and human-friendly behavior. In the **third** mode the interaction zones, in particular interaction tasks, are carried out. They have to be specified or generated for fulfilling a common desired goal, involving a synergy of human and robot capabilities in an efficient manner. These two modes are forming an integrative interaction concept, which allows seamless switching between each other. The **fourth** mode is defining the fault reaction reflexes, which take care of the appropriate and safe state dependent fault reaction of the robot. Here only reflexes, $r \in \mathcal{R}$, can be used.

To sum up following relations hold for the distinct sets of behaviors:

$$\mathcal{B} = \mathcal{B}_v \cup \mathcal{B}_{inv} \tag{2}$$
$$\mathcal{B}_v \equiv \mathcal{R} \cup \mathcal{B}_{am} \cup \mathcal{B}_{hf} \equiv \mathcal{R} \cup \mathcal{B}_{op} \tag{3}$$
$$\mathcal{B}_{am} \not\equiv \mathcal{B}_{hf} \not\equiv \mathcal{R} \tag{4}$$
$$\mathcal{B}_{am} \cap \mathcal{B}_{hf} \not\equiv \{\} \tag{5}$$
$$\mathcal{B}_{am} \cap \mathcal{R} \not\equiv \{\} \tag{6}$$
$$\mathcal{B}_{hf} \cap \mathcal{R} \not\equiv \{\} \tag{7}$$
$$\mathcal{R} \equiv \mathcal{R}_S \cup \mathcal{R}_F \tag{8}$$

The activation of reflexes is discussed hereafter.

[6]Of course also complex reaction chains are possible. However, they have to be designed carefully and should not be accessible to the task programmer.

1) Activation signals and fault severity stages: To achieve an adequate reaction with respect to activation signals or the severity of a fault we use six distinct layers. A different reflex reaction can be assigned to each of them. If a fault occurs, the behavior, which equates the activation/fault level is activated and retained until a confirmation was received. The layers are separated by thresholds on physical or virtual forces (they represent also external stimuli), communication faults and low-level robot states.

1) Human friendliness mode 1 (HF L1)
2) Human friendliness mode 2 (HF L2)
3) Fault tolerance mode 1 (FT L1)
4) Fault tolerance mode 2 (FT L2)
5) Emergency mode 1(E L1)
6) Emergency mode 2(E L2)

The human friendly mods HF Li are activated in case of minor activation signals or faults as e.g. slight contact. The fault tolerance modes FT Li are associated to faults as hard collisions. The emergency layer is activated only if a severe fault as a very hard collision occurs. The external torque threshold τ_{ext}^{EL1} is close to the robots maximum joint torque limits τ_{max}. The reflexes for FT L1 to E L1 is specified in advance to protect human and robot from being harmed. The reflexes in the HF layers can be chosen freely to fit the current state.

2) Concrete implementation: E L1 is a low-level fault, which directly enables the robot emergency breaks (e.g. triggered by the emergency stop button). If a very hard collisions occurs, which leads to an E L1 event, the system automatically stops the robot with its emergency brakes and exits the running task. Also the reaction strategy for FT L1 and FT L2 are hard coded in the reflex space (real-time) layer and cannot be changed by the user. FT L2 leads to a controlled software stop without using brakes (M.1). If FT L1 occurs, the robot is automatically set to gravity compensation mode (C.1). The threshold for these layers is fixed to 95%, 80% and 50% of τ_{max}. For HF L1, HF L2 and FT L1 the threshold can be set according to the task. To achieve a fast collision reaction it is necessary that the decision and the strategy itself run in the fast real-time loop of the system.

Figure 2 depicts an example with two operational behaviors $B.1$ (Non reactive joint motion generator (M.2.J) with joint position control (C.1.J)) and $B.2$ (attractor based Cartesian motion (M.3.C) with Cartesian impedance control (C.3.C)), a reflex behavior on HF L1 $ReflexHFL1$ (stopping the motion (M.1) and switch to Cartesian impedance (C.3.C)). In addition, we define for harder contact $ReflexHFL2$ on HF L2 (stopping motion (M.1) and switch to torque control (gravity compensation) (C.2.C)).

3) Selecting the functional mode: Up to now, industrial settings are usually simple sequences of tasks whose execution order is static and sometimes some binary branching is possible. In our concept, on the other hand, flexible jumps within execution are an integrated part and do not need some special treatment as they act autonomously. In order to optimally combine human and robot capabilities the robot must be able to quickly adapt to the human intention during task execution for both achieving a safe way of interaction and high productivity. Thus, the measured human state is the driving transition between the proposed functional modes.

Estimating the human state is a wide topic of research and has been treated in some recent work [16]. However, for selecting the functional modes, the more relevant infor-

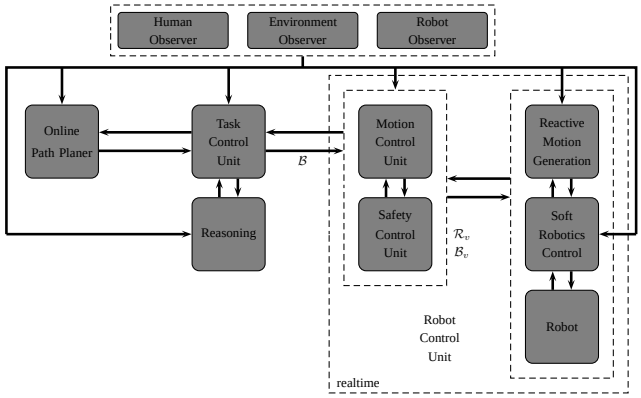

Fig. 3. Overview of the LWR-III control architecture for human-friendly behavior. The Task Unit is responsible for sending behaviors and reflexes to the Safety Unit, which checks them for validity and distributes them to the corresponding parts of the RCU.

mation is the physical state the human currently occupies. Thus, a clear set of sufficient behaviors and reflexes can be selected and activated, which leads to very robust and reliable behavior:

- **oP**: out of perception
- **iP**: in perception
- **iCM**: in collaborative mode
- **iHF**: in human-friendly mode

This information is primary used to switch between the different functional modes, since it indicates the nominal task and furthermore, it specifies how to react during certain faults. In Sec. III-C we describe this on the detailed low-level architecture.

III. SYSTEM ARCHITECTURE

To enable efficient robot task development with the aforementioned behaviors together with the functional modes, a well structured architecture is needed, which automatically handles consistent activation and switching of the system state. In this section we present our control architecture and discuss its core elements[7].

The basic structure of this architecture is depicted in Fig. 3. It shows the four central entities for robot control:

1) Task control unit (TCU)
2) Robot control unit (RCU)
   a) Safety control unit (SCU)
   b) Motion control unit (MCU)

The first two units serve as the general interface to the robot and communicate with each other via asynchronous protocols [8]. The TCU is the general state based control entity for gathering non-real-time data and providing the correct nominal behavior changes on an abstract level to the RCU. The RCU runs in the same clock rate as the robot, assigning control, motion generation, and safety methods, i.e. the concrete behaviors. Furthermore, it interprets and validates the selection of behaviors from the TCU, while preventing incorrect combinations with respect to the actual functional mode. The SCU serves as an underlying safety layer below the RCU, which combines all low-level safety behaviors and activates them consistently. The Motion Control Unit, which is supervised by the SCU, is responsible for appropriately changing the control and motion behavior of the robot. SCU and MCU are both designed as hybrid state machines. The SCU supervises information from various sources and routes all control sinks. It decides state dependently whether environmental sensor input is used and which behaviors for environment and robot protection are activated.

A. Modular Design

The modularity of the system begins with the separation of TCU and RCU, see Fig. 4. This has several advantages. On the one hand, the obvious performance benefit of separating real-time and non-real-time. The non-real-time parts, which are not necessary for real-time robot motion control and safety can be handled by a standard non-real-time operation system. On the other hand, developing a task is more efficient, since the complex real-time parts as e.g. controllers, robot dynamics calculation, or robot simulation are located in a static real-time core. This design allows online detachment and re-attachment of the TCU. Furthermore, we also demand strict separation of methods within TCU and RCU, which significantly accelerates the development and integration of new methods into the system.

Next, we discuss TCU, RCU, and the connection of external processing units.

B. Task Control Unit

The interface for the application programmer is the TCU, see Fig. 4(right). A task can be implemented and tested in a graphical hybrid state machine implementation. In this state machine the entire relevant system data is gathered and logically organized. This can then be used for implementing complex decision structures as operational behaviors. Furthermore, the reflex behaviors are activated according to the task specification. The run-time demands on the TCU are very relaxed compared to the RCU and due to the event based decision structure no hard real-time requirements have to be fulfilled.

C. Robot Control Unit

The Robot Control Unit (Fig. 4(left)) handles all parts of the system, which have to run in real-time (Reflex Space). The main elements of the RCU are

- controllers,
- motion generators,
- safety components,
- external components,
- state machine (SCU and MCU), and
- robot interface/ robot simulation.

The RCU state machine consists of the SCU and MCU. They are responsible for consistent activation of features and selecting different motion generators and controllers[9]. Developing a robot task in the TCU, one has not to care about timing problems or valid motion generator - controller combinations and their correct switching. As already mentioned, the behavior reflexes can be set up by the task implementation in the TCU. Afterwards the collision detection,

[7] For brevity we skip the communication design.

[8] DLR agile Robot Development communication (aRDnet) [17] is used.

[9] A particularly important aspect is that we can seamlessly switch between the real robot and and an integrated full dynamics simulation during runtime, which significantly speeds up the development process of new motion schemes and controllers.

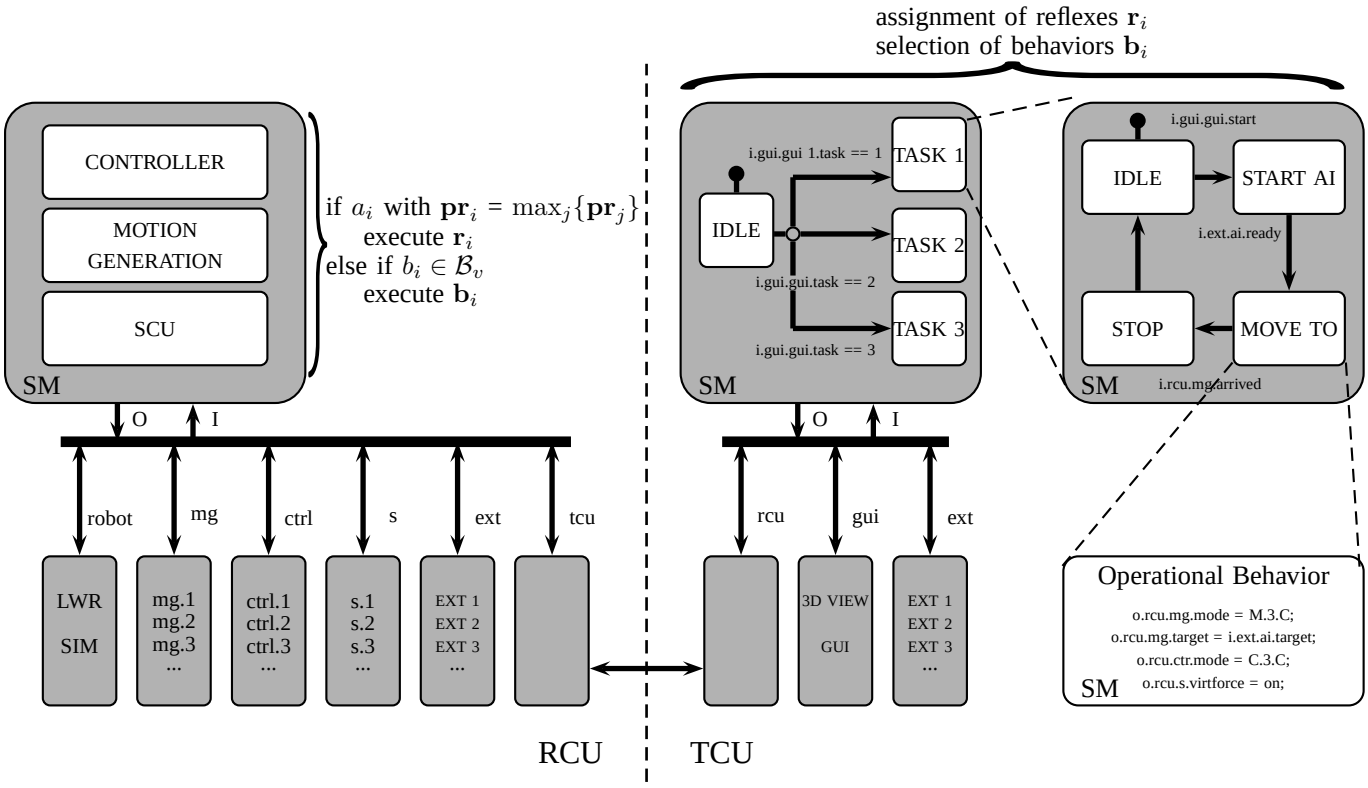

Fig. 4. The communication pathways between Robot Control Unit and Task Control Unit characterized by a complete bus system. The upper part of the TCU depicts the hierarchical graphical state based programming of high-level tasks. The label SM in a block indicates that is implemented as a state machine.

decision, and behavior switch are completely handled in the fast real-time loop. Figure 5 depicts a functional overview of the RCU. In the upper part all methods described in Sec. II and their interrelation are shown. The motion generation, controller and the robot interface, respectively robot simulation, are placed in the center. The small black frames around these blocks describe their particular use for the existing safety features. Dotted borders indicate that a block generates information that is used for safety decisions in the SCU. Blocks with dashed borders are effect conditioned with respect to this information. If a block has a solid line it works as both, a source and sink of safety information. The wide borders depict by which state machine this block is affected. Underneath, the SCU and MCU are shown.

D. External Components

For incorporating external sensing, reasoning, or controllers, both RCU and TCU provide a generic interface to connect stand-alone applications to the system[10]. The type of external application is not specific (e.g. path planner, image processors, or artificial intelligences). Therefore, they are generally denoted as external components. External components that require real-time data are directly connected to the RCU, while non time-critical components are attached to the TCU[11].

IV. GRAPHICAL STATE-BASED PROGRAMMING

An important factor in robot programming is to provide an environment that minimizes the chance of programming errors for the task programmer. A convenient way to do this is graphical state based programming. The programmer arranges well defined complex states that consist of elementary operational behaviors, reflexes behavior setups and corresponding commands hierarchically. Furthermore, information coming from the RCU or external components can be used for making local decisions (hybrid transition graphs).

In the following we discuss the concrete implementation of the example given in Fig. 2.

A. Programming example

Figure 6 depicts a concrete simple implementation example to showcase our approach. The task is equivalent to the conceptual scheme drawn in Fig. 2. However, HF L1 reflexes are omitted for sake of clarity. The nominal operational behavior is to move between two positions A and B. For this motion (operational behavior states b1 and b2) a reactive behavior is selected, which enables collision avoidance (of the human) and contact retraction based on proximity and physical forces. In case the contact forces exceed the perdefined HF L2 threshold (HF L2 is the corresponding activation signal a1), the collision reflex r1 is activated. This causes the robot to switch to torque control with gravity compensation and enables a virtual trap. After confirming the collision by intuitive physical interaction (a2) the task continues with the behavior prior to the collision (b2).

B. Applications

Several applications were developed at DLR that base on the concept of this paper. Figure 7 depicts a few of them. The first picture shows the recent Braingate experiment

[10]This is done via aRDnet, which allows direct access from the state machine accross heterogeneous computer networks.

[11]Of course it is possible to connect to both at the same time as well.

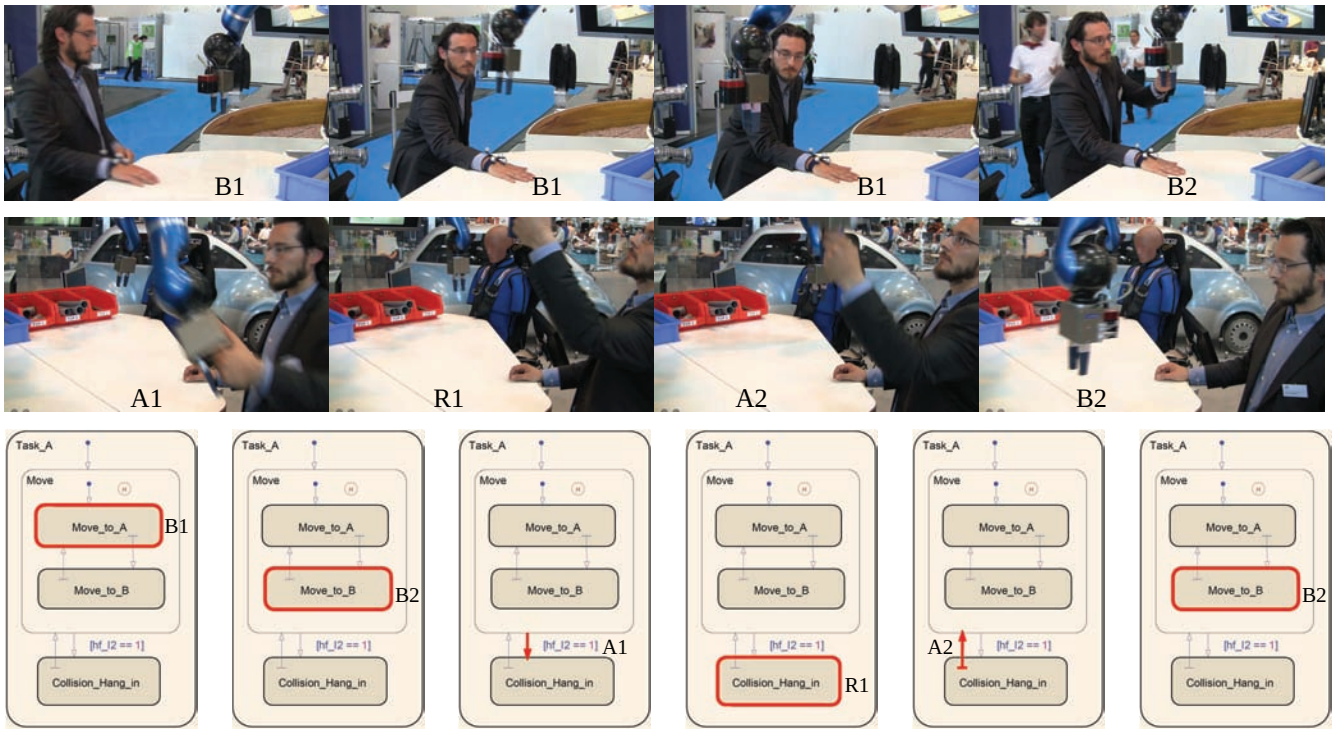

Fig. 6. Implementation example of a simple human-friendly task.

Fig. 7. Several example setups using the here described concept are shown. (from left to right) LWR-III controlled via Brain-Machine Interface, SAPHARI setup, billard playing, and EMG-controlled robot.

[18], where the LWR-III is continuously controlled via a Brain-Machine-Interface. The decoded neural data is used to command the robot, while several safety related behaviors are activated. The second image depicts the SAPHARI setup, which is an experimental multi-robot setup for evaluating safe and autonomous physical Human-Robot Interaction. At the recent trade fair AUTOMATICA 2010 the various complex interaction capabilites of the state based controlled robots were showcasd for different applications as e.g. interactive bin-picking[12]. The LWR-III billiard experiment [19] (third picture) is an application in which an autonomous task is combined with human-robot interaction modalities. The fourth picture shows an EMG-controlled LWR-III. Please note that numerous videos of realized applications can be found at http://www.safe-robots.com

[12]Please note that for the SAPHARI setup our concept is fully applied in a modular sense to the multi-robot setup.

V. CONCLUSION

In this paper we introduced a novel concept for formulating complex behaviors tailored to the needs of safe and autonomous pHRI. We extend the classical view on behaviors by introducing the associated safety behavior of the robot and distribute it in the sense that they are either operational or reflex behaviors. Furthermore, we proposed a light-weight modular robot control architecture, which builds this formulation and incorporates on a variety of methods for safe and human-friendly robot control. This architecture provides a graphical state based task programming interface and allows to implement sophisticated and safe robot behavior in

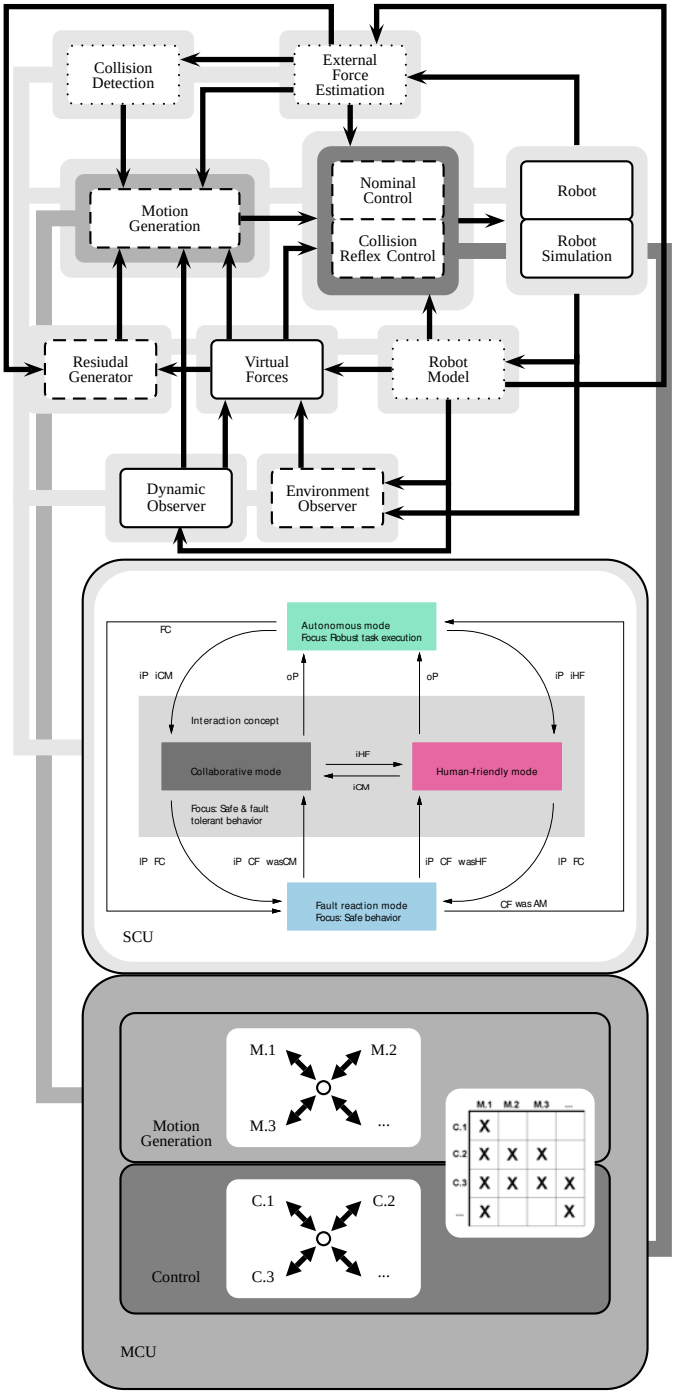

Fig. 5. Overview Robot Control Unit (RCU). In the upper part the available methods are shown. The thin borders indicate the type of the particular scheme with respect to their safety role: source (dotted line), sink (dashed line) or both (solid line). In the middle part the Safety Control Unit (SCU) is shown, which switches depending on the human information (FC = Fault Condition, CF = Confirm, oP = out of Perception, iP = in Perception, iCM = in Collaborative Mode, iHF = in Human Friendly Mode) between its operational modes. The Motion Control Unit (MCU) manages the consistent control of motion and control schemes.

complex human-robot interaction scenarios as showcased for numerous applications.

REFERENCES

[1] J. Guiochet, D. Powell, E. Baudin, and J.-P. Blanquart, "Online safety monitoring using safety modes," *6th IARP/IEEE-RAS/EURON Workshop on Technical Challenges for Dependable Robots in Human Environments*, 2008.

[2] T. S. Wikman, M. S. Branicky, and W. S. Newman, "Reflex control for robot system preservation, reliability and autonomy," *Comput. Electr. Eng.*, vol. 20, no. 5, pp. 391–407, 1994.

[3] Y. Matsumoto, J. Heinzmann, and A. Zelinsky, "The essential components of human-friendly robot systems," in *in Int. Conference on Field and Service Robotics*, 1999, pp. 43–51.

[4] J. Heinzmann, A. Zelinsky, and E. Zelinsky, "The safe control of human-friendly robots," in *Proc. of the 1999 IEEE/RSJ Int. Conf. on Intelligent Robots and System*, 1999, pp. 1020–1025.

[5] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila, "Task planning for human-robot interaction," in *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence.* New York, NY, USA: ACM, 2005, pp. 81–85.

[6] D. Kulic, "Pre-collision safety strategies for human-robot interaction," *Autonomous Robots*, vol. 22, pp. 149–164(16), February 2007.

[7] B. Brunner, K. Arbter, and G. Hirzinger, "Task directed programming of sensor based robots," in *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, vol. 2, sep. 1994, pp. 1080 –1087 vol.2.

[8] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmuller, A. Albu-Schäffer, and G. Hirzinger, "Towards the robotic co-worker," in *International Symposium on Robotics Research (ISRR2007), Lausanne, Switzerland*, 2009.

[9] A. Albu-Schäffer, C. Ott, and G. Hirzinger, "A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots," *The Int. J. of Robotics Research*, vol. 26, pp. 23–39, 2007.

[10] C. Ott, A. Albu-Schäffer, A. Kugi, S. Stramigioli, and G. Hirzinger, "A passivity based cartesian impedance controller for flexible joint robots - Part I: Torque feedback and gravity compensation," in *Int. Conf. on Robotics and Automation (ICRA2004), New Orlenas, USA*, 2004, pp. 2666–2672.

[11] S. Haddadin, A. Albu-Schäffer, A. D. Luca, and G. Hirzinger, "Collision detection & reaction: A contribution to safe physical human-robot interaction," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2008), Nice, France*, 2008, pp. 3356–3363.

[12] A. De Luca, A. Albu-Schäffer, S. Haddadin, and G. Hirzinger, "Collision Detection and Safe Reaction with the DLR-III Lightweight Manipulator Arm," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2006), Beijing, China*, 2006, pp. 1623–1630.

[13] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger, "Real-time reactive motion generation based on variable attractor dynamics and shaped velocities," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2010), Taipeh, Taiwan*, 2010.

[14] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, and A. Albu-Schäffer, "Reactive motion generation based on variable attractor dynamics and shaped velocities," in *Conference on Intelligent RObots and Systems (IROS2010), Taipei, Taiwan*.

[15] S. Haddadin, M. Suppa, S. Fuchs, A. Albu-Schäffer, and G. Hirzinger, "Towards the robotic co-worker," in *14th International Symposium on Robotics Research, Luzern, Switzerland (August/September 2009)*.

[16] D. Kulic and E. A. Croft, "Affective State Estimation for Human-Robot Interaction," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 991–1000, 2007.

[17] G. Hirzinger and B. Bäuml, "Agile robot development (ard): A pragmatic approach to robotic software," in *IROS*, 2006, pp. 3741–3748.

[18] V. Jörn, S. Haddadin, J. D. Simeral, S. D. Stavisky, D. Bacher, L. R. Hochberg, J. P. Donoghue, and P. van der Smagt, "Continuous control of the dlr light-weight robot iii by a human with tetraplegia using the braingate2 neural interface system," in *accepted at: International Symposium on Experimental Robotics (ISER2010)*, 2010.

[19] S. Parusel, "Playing billard with an anthropomorphic robot arm," Hochschule Kempten, Tech. Rep., Juli 2009. [Online]. Available: http://elib.dlr.de/61668/