Memory-based Software Integration for Development in Autonomous Robotics

Thorsten Peter SPEXARD, Frederic H. K. SIEPMANN and Gerhard SAGERER

Applied Computer Science, Bielefeld University,
Universitätstraße 25, 33615 Bielefeld, Germany;
E-mail: {tspexard, fsiepman, sagerer}@techfak.uni-bielefeld.de

Abstract. Focusing the development of non-industrial robotics in the last decade the growing impact of service and entertainment robots for daily life has emerged from pure science fiction to a serious scientific subject. But still many questions in how to solve everyday tasks like laying the table or even “simpler” detecting objects in unstructured areas with varying lighting conditions are unsolved. Hence the strong need to evaluate and exchange different approaches and abilities of multiple robotic demonstrators under real world conditions is also a crucial aspect in the development of system architectures. In this paper an architecture will be described providing strong support for simple exchange and integration of new robot abilities.

Keywords. autonomous robotics, human-robot interaction, cognitive architecture, system integration

1. Introduction

In combination with the increasing computational power both the amount of abilities for robots and the complexity of single abilities grew. Besides the impressive progress in the skills of robots and the concerning algorithms behind them, another issue becomes more and more important: Operating in human-oriented surroundings. This requires the combination of multiple different robot abilities. Based on our previous experience in building integrated robotic systems we present an advancement of our former System Infrastructure. This enhanced approach increases both the flexibility in integrating and changing components, as well as to efficiently combine information given from separate modules. After a short discussion on alternative proposals for integrated systems in Section 2, the robotic demonstrator acting in a human centered environment is presented in Section 3. Subsequently the new architecture with its benefits is discussed in Section 4. In Section 5 we present numbers for the decreased system complexity and practical experiences before closing with a brief summary.

2. Related Work

In research a broad variety of different architectures have been developed. Using classical AI strategies Nilsson [1] propagated the deliberative robot control. Based on a world model and a given goal plans are generated to achieve the goal. If the environment changes the planning has to be repeated. This is almost always too time consuming for
embedded systems in dynamic human-centered environments. Therefore, reactive architectures [2] should overcome this burden, but the complexity of the tasks to be executed is strongly limited. Thus the combination of both deliberative and reactive components in a hybrid architecture [3] should enable a robot to generate plans for complex tasks and to react dynamically to a changing surrounding. Motivated by biological models behavior based architectures propose systems with separate abilities driven by reactive or deliberative components. These abilities can be seen as separate behaviors like following a human and avoiding obstacles. Hence behaviors may provide contradictory results like in this following task an arbitration mechanism has to select the final action.

We propose a system that is more deterministic than a weighted arbitration and therefore easier to design on the one hand, on the other hand it is sufficient dynamic to allow even online changes in the behavior modules and arbitration without redesigning source code or configuration files offline. Additionally the approach described in this paper is not depending on globally equal interface definitions as these often result in overhead and a less dynamic integration. This concerns especially behaviors, which are either exchanged with partners of different institutes or were not planned at the start of a project as their necessity emerged during experiments.

3. Applications in the Real World

Instead of artificial experiments under laboratory conditions an apartment is continuously rented to test our demonstrator BIRON (Bielefeld Robot Companion) under real world conditions including small doorways, sticky grounds, and uncontrolled lighting. BIRON is equipped with several sensors that allow an assessment of the current situation as a basis for interaction. Its onboard computational equipment in combination with a wireless LAN notebook is sufficient to achieve a system running completely in real time which is necessary for HRI in the Home-Tour-Scenario. The scenario envisions a newly purchased robot being introduced to its new working area – usually an apartment – by an unexperienced user. Due to the huge variety of application areas especially in home environments only a small set of pre-programmed knowledge is useful. The main knowledge such as maps or objects in the new environment has to be obtained online during interaction with a person [4].

To succeed in this task a modular approach was chosen to easily add and exchange abilities. More than 30 asynchronous modules are running in parallel using an XML based Communication Framework (XCF, [5]). The communication was mainly realized in 1:n XML publish/subscribe data streams and remote method invocations (see Fig. 1(a)). The synchronization of events like switching the robot control from the person tracking and attention module (PTA) [6] to object tracking was realized by a finite state machine implementation (ESV) using the information sent via the data streams as transitions between its internal states. For each transition modules like the dialog system (DLG) [7] were reconfigured according to their new task. DLG, ESV, PTA, and Hardware Control (HWC) build a core system for basic HRI which is extended with modules for localization, navigation, gesture recognition and object attention.

With the growing number of modules the number of stream connections between them became a limitation: A module getting information from a data stream needed the data stream and therefore the stream generating module already running before being started itself. Thus modules depending on each other caused restarts of the whole system...
if a 1st-level module exited unexpectedly during development. Although this problem was caught by exception handling, the system could get into an inconsistent state when a module broke and was restarted, as during the reconnection process of the data streams information was lost in the meantime. The sender was not aware whether the information was successfully received or not. Furthermore the combination of information from different modules needed to either add a combining module with even more stream connections or the adaptation of an existing module, implementing worldknowledge concerning the software which could be adapted by source code manipulation only.

4. Integration

To overcome these limitations we propose that as less knowledge as possible concerning the robotic system is incorporated in the single modules depending not on the operation of another module but only on incoming data. This was realized by the active memory concept [8]. This system wide memory is able to hold all information of the modules and structure them semantically with regard to the expiration of validity:

**Short term memory** This memory contains information with a short duration of validity. A forgetting process removes all entries in the memory which are not updated within a certain period of time, e.g., person positions.

**Scene memory** As a representation of the current scene this memory keeps all information that are valid until changed or removed by a module. The memory contains, e.g., information about the current room the robot is in.

**Long term memory** This memory stores information which shall be kept permanently. This includes system parameters for BIRON’s behavior as well as, e.g., object descriptions or maps.

Instead of stream connections a module registers database triggers which specify the kind of information the module needs by XPath and a database event. It is activated automatically when the information is available, independently from the information source and its current state. This reduces the communication complexity (see Fig. 1(b)) and encapsulates modules in terms of reducing system knowledge in all modules, increasing the extensibility and adaptivity of the system. Alternatively, remote method invocation (RMI) is used when an affirmation for information exchange is required. For continuous raw data (e.g., laser data) the primary streaming approach was kept as partial drop outs of these data do not effect the system and memory flooding is avoided.

As a consequence of the decreased system knowledge within the modules a more powerful coordination module replaced the former ESV. The active control memory interface ACMI takes the current memory state with all its information into account. It observes the system by registering to the different memories to derive information like the current system state (e.g., following someone, recognizing new rooms) from combining stored information. ACMI is also capable of manipulating the current memory content by inserting, removing or replacing information. It takes responsibility for behavior arbitration and RMI for crucial information exchange, like the information which module gains control of the hardware in a certain situation. The necessary parameters are part of the ACMI configuration and represented in XML. The mayor and as simple as powerful advantage is that this configuration is also stored in the memory. Thus it can be modified during runtime either explicitly by a developer or by a software module. With ACMI
(a) Communication using mainly streams. (b) Advanced communication with memory.

Figure 1. Comparing former and memory based communication dependencies: The number of inter module connections was decreased by more than 70% from 30 to 8, now using streams for low level sensor data, only.

triggering its own configuration information the system is enabled to adapt e.g. its arbitration routine if a preprogrammed strategy does not succeed. An example of the use of triggers and adaptive memory use is provided by the HWC.

If the content of the memories indicates the necessity for a change in the hardware controlling module, ACMI uses an RMI on the HWC to send the name of the module, which is to control the hardware next. The RMI enables the system to estimate when the configuration is concluded and to propagate this back to the memory as well as the result of the HWC configuration. Both, the name of the controlling module, and the necessary memory content are defined in the ACMI configuration and thus stored in the memory, too. Using a standard prefix with the module name as suffix an XPath is generated. The XPath defines the location in the memories where the hardware commands of the controlling module are inserted. Subsequently a trigger is registered to newly inserted information under the given XPath and a potential former trigger is removed. If new information is inserted into the memory matching the specified XPath HWC actuates the hardware with the extracted information. This method proposed here is introduced as memory pointing comparable to programming languages not the information itself but where to find it is exchanged. The benefit of memory pointing is that it is flexible and extendible and no source code adaptation is needed to integrate new modules that need to control the hardware, which was necessary with the prior stream-based implementation. In addition memory pointing is robust towards missing or misbehaving modules as with missing information the system just idles.

5. Benefits

Regarding the core system, inter module connections were reduced by more than 70% from 30 to 8 streams as demonstrated in Figure 1. This reduces the number of possible communication failures as well as it decouples the modules. They now depend on the active memory module only, to interact with the whole system. Previously 8 layers of software modules had to be started causing modules in a higher layer to become unstable if a lower layer module broke, now the number of layers was reduced to 4. Taking into
account that the first two layers are used to set up the communication framework the dependency depth of robot operating modules is finally reduced from 6 to 2. This results in modules restartable during development with only minimal influence to the system. Besides the theoretical value of communication complexity the authors present a first test concerning both the robustness of the system but also the benefit of the architecture for integration of new modules. A module for localization from a project partner of a different institute was integrated. Providing a sample source file for memory access methods and exchanging an XML-interface definition the whole practical integration was done within one day. During the integration the localization module had to be restarted several times. In contrast to previous approaches the remaining system did not need to be restarted but continued working stable, saving a huge amount of time. After the integration a first Home-Tour was performed. The robot worked robust and in real time for half a day limited only by battery life.

6. Summary

We presented a new architectural approach for autonomous mobile robots operating in natural, human centered environments. For this approach the usage of data stream was changed to a biologically inspired memory system consisting of short term, scene, and longterm memory. The memory approach allows the robotic system to store high level information as gestures or objects and combine the information with global access for each component. Moreover the new approach strongly supports the idea of loose coupling by reducing both the component dependencies on each other and the direct data stream connectivity by 70% The applicability of the presented work was proven by implementation of a core system and testing the demonstrator by a Home Tour in the apartment with a module for location learning from a different institute. The system ran stable during both the integration and interaction for hours.

References


1Olaf Booij, Intelligent Systems Lab Amsterdam, University of Amsterdam