Programming Autonomous Robots Using Agent Programming Languages

(Doctoral Consortium)

Pouyan Ziafati
SnT, University of Luxembourg
Intelligent Systems Group, Utrecht University
pouyan.ziafati@uni.lu

ABSTRACT

This research aims at identifying and addressing BDI-based agent programming languages requirements for programming autonomous robots. Four requirements are discussed and the current state of this research in addressing these requirements is presented. The requirements are: 1- Built-in support for integration with existing robotic frameworks such as ROS[1], 2- Real-time reactivity to events, 3- Management of heterogeneous sensory data and reasoning on complex events, and 4- Representation of complex plans and coordination of the parallel execution of plans.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Languages and structures, Intelligent agents*; D.2.1 [**SOFTWARE ENGINEERING**]: Requirements/ Specifications—*Languages, Tools*

General Terms

Languages

Keywords

Robotic Agent Languages, Agent Programming Languages, Autonomous Robotics

1. INTRODUCTION

To achieve complex goals in dynamic environments, robots need to be empowered with deliberative behavior. One of the suitable architectures for implementing deliberative behavior is the BDI architecture [5]. Various agent programming languages (APLs) such as 2APL [2] have been developed to facilitate the implementation of BDI architecture. However, the application domains of these languages have been mainly limited to cognitive software agents.

The aim of this research is to address necessary requirements to facilitate the use of BDI-based APLs for implementing robotic control systems in a modular and systematic way. Exploring a home-care NAO service robot application scenario [7], the problems of plan execution control, sensory data management and real-time reactivity were faced as basics when applying current APLs in autonomous robotics. Section 2 presents various requirements related to these problems that APLs should address to be more suitable for programming autonomous robots. Section 3 describes the current state of this research in addressing such requirements. Section 4 concludes the paper and presents future work.

[1]http://www.ros.org/wiki/

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

2. REQUIREMENTS FOR PROGRAMMING AUTONOMOUS ROBOTS

Studying the problem of programming autonomous robots and looking into current robotic architectures and tools such as the ones presented in [4], [3] and [6] shows that current APLs lack support for different aspects of programming autonomous robots [7]. Our aim is to enable such support for BDI-based APLs in general and 2APL agent programming language in particular.

One of such requirements is integration with existing robotic frameworks. This can encourage the use of APLs by robotic community and facilitate their use for rapid prototyping and development of autonomous robots.

The second requirement is providing support for the development of sensory management components (SMCs) that process and manage heterogeneous sensory information. Such components should enable a unified representation of sensory data and domain knowledge as well as reasoning on high-level events (i.e. situations). The sensory information managed and processed by SMCs should be accessible by a BDI-based control component in a symbolic form and through both querying and receiving of events.

Another requirement is extending plan representation and execution capabilities of current APLs. Required capabilities include providing support for governing the execution of plans by sequential, temporal and priority orderings, and based on different internal conditions and external events, representing and handling conflicts in parallel execution of plans, and monitoring and handling plan execution failures.

The last requirement is real-time reactivity to events. Based on the current trends in robotic control architectures [3, 6], a suitable approach for addressing such requirement seems to be supporting the development of distributed real-time BDI-based control systems. This requires a specific version of an agent programming language dedicated to development of real-time control components. The semantics and implementation of such a version should guarantee safe and bounded-time computations to enable analysis and guaranteeing required real-time properties of the control component. In addition, a dedicated architecture and runtime environment is required to support the real-time coordination and communication of different control components of a robot. We envision an architecture consisting of a distributed set of BDI-based control components with different functionalities (e.g. deliberative, reactive, plan failure handling) which can share beliefs and goals, and other robotic software components including sensory components described above. These components can have different real-time requirements. Some of them should run in real-time and guarantee bounded reaction and response time to events.

3. RESULTS

This section describes the current result of this research in tackling the problems of integration with robotic frameworks and sensory data processing and management.

3.1 Interface to Robotic Frameworks

To address the requirement of integration with existing robotic frameworks, we have developed an environment interface for 2APL to facilitate its integration with ROS. The interface enables the communication of 2APL with ROS components using ROS communication mechanisms. We have used ROS to provide basic robotic capabilities such as face recognition, voice recognition and a number of high-level actions such as stand-up(), turn-neck(O) and walk-to(X,Y) for our NAO robots. Using 2APL and ROS, we have developed a demo application in which different movements of a NAO robot can be controlled by voice. Furthermore, NAO can remember faces and whenever a user greets NAO, NAO greets the user by his/her name if it recognizes the user's face,.

3.2 Event-Processing Software Library

To address the problem of sensory data processing and management, we have developed a software library to support the implementation of sensory management components for an autonomous robot [8]. The software library integrates the Etalis event processing language [1] to facilitate the implementation of event processing tasks such as filtering, transformation and detecting complex patterns of events. Etalis is an open-source, expressive and efficient event-processing language with a formal semantics.

The software library supports the necessary interaction mechanisms between a robot's software components (e.g. a control component developed in an APL) and SMCs. These mechanisms allow a (control) component to subscribe at run-time to SMCs for its events of interest, to receive the events of interest asynchronously, to maintain necessary histories of events in SMCs and to query the histories on-demand.

4. CONCLUSION

The paper presents the current state of our research in identifying and addressing BDI-based APLs requirements for autonomous robot programming. The focus of the research is on the application of BDI-based APLs in service robot application scenarios such as the one presented in [7]. However, the set of requirements and solutions identified so far are similarly applicable to a wider area of autonomous robot programming including plan-based robot programming languages and knowledge-based robotic architectures.

To support the application of APLs in a broad area of autonomous robot programming, the research methodology adopted is to identify the necessary requirements and suitable solutions based on an analysis of a prototypical service robot application scenario [7] on the one hand, and based on an extensive survey of the related literature on the other hand. Our survey includes a large number of plan-based robot programming languages and knowledge-based and cognitive robotic architectures.

The Future work is to further develop the presented event-processing library to support the implementation of active memories for APLs and to address the other requirements. The end result of this research project will be a set of reusable software libraries facilitating the application of BDI-based APLs in autonomous robot programming. The result should be validated by the implementation of a prototype application scenario for one or more NAO and possibly other robots using the extended version of 2APL.

Acknowledgement

Pouyan Ziafati is supported by the National Research Fund (FNR), Luxembourg.

5. REFERENCES

[1] A. Darko. Event processing and stream reasoning with ETALIS. PhD thesis, Karlsruher Institute of Technology, 09 Nov 2011.

[2] M. Dastani. 2APL: a practical agent programming language. Autonomous Agents and Multi-Agent Systems, Volume 16 , Pages 214-248, 2008.

[3] R. Passama and D. Andreu. ContrACT: a software environment for developing control architecture. 6th National Conference on Control Architectures of Robots, 16 p, 2011

[4] K. Rajan, et. el. Onboard Adaptive Control of AUVs using Automated Planning and Execution, Intnl Symposium on Unmanned Untethered Submersible Technology (UUST), Durham, NH, August 2009.

[5] A. Rao and and M. Georgeff. BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pages 312-319, June 1995.

[6] V. Verma, et. el. Survey of command execution systems for NASA spacecraft and robots. In Plan Execution: A Reality Check Workshop at the International Conference on Automated Planning and Scheduling (ICAPS), 2005

[7] P. Ziafati, et al. Agent Programming Languages Requirements for Programming Cognitive Robots. Proceedings of the Tenth International Workshop on Programming Multi-Agent Systems, ProMAS @ AAMAS, Pages 39-54, 2012.

[8] P. Ziafati, et al. Event-processing in autonomous robot programming. Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, May, 2013, Saint Paul, Minnesota, USA.