A Library for Event-Processing and Adaptable Component Interactions in Autonomous Robot Software

Pouyan Ziafati, Holger Voos, Leendert van der Torre
SnT, University of Luxembourg
Email: {Pouyan.Ziafati, Holger.Voos, leon.vandertorre},@uni.lu

Mehdi Dastani, John-Jules Meyer
Intelligent Systems Group, Utrecht University
Email: {M.M.Dastani, J.J.C.Meyer}@uu.nl

***Abstract*—A light-weight framework-independent software library is introduced to facilitate a modular and systematic development of sensory management components for an autonomous robot. Such components are used to implement complex event-processing tasks such as content-based filtering, integration and transformation of sensory data. In addition, they can be used as mediators to provide a number of high-level interaction mechanisms among a robot's software components. To this end, they enable components with subscription to their events of interest, asynchronous reception of events, maintaining necessary histories of events and querying of the histories at runtime.**

I. Introduction

The software of a service robot situated and interacting in a natural human environment needs to integrate a large number of components to provide the robot with various perception, actuation and cognitive capabilities. The integration of such components is hard due to the inherent heterogeneity in the data representation and communication styles of the components and their parallel and asynchronous executions.

State of the art robotic frameworks such as ROS[1] facilitate such integration by providing extendible data structures and different interaction styles such as service-based and publish-subscribe communication mechanisms. The next step which could improve the robotic software engineering experience in and among robotic frameworks is to identify, document and support the implementation of architectural design patterns beyond the current component interaction patterns [2], [4], [3]. This may include high-level component structures, interfaces, communication patterns and corresponding development tools to increase the usability, reuasability, observability and comparability in the development of common robotic software engineering tasks.

To this end, this paper introduces a sensory management software library [7] which provides an efficient high-level language with a formal semantics to support the implementation of sensory management components (SMC) for an autonomous robot. SMCs are used for the unified representation of sensory events, detecting complex patterns of events, and as mediators to support a number of high-level content-based interaction mechanisms among the other components. The library supports these functionalities by integrating and extending the Etalis event-processing system [1] with so called dynamic subscription, history management and on-demand query mechanisms. The semantics and implementation of the library are described in details in [7]. This paper presents an informal overview of the library and our current research on its further development. The rest of the paper is organized as follows. Sections 2 and 3 present its support of event-processing and component interactions respectively. Section 4 presents its integration with ROS. Sections 5 concludes the paper and presents future work.

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

II. Event-Processing

The inputs of a SMC are streams of events. In our approach, an event is a data item represented as a Prolog fact with an occurrence time interval. For atomic events where occurrence times are single time points, the start and end of their occurrence time intervals are the same time points.

Example 1. The face of Antonio recognized with the reliability of 70% in an image taken at the system time 28 could be represented as $face(\text{“}Antonio\text{”}, 70)^{\langle 28,28\rangle}$ and the continuous recognition of Antonio from the time 28 to 49 could be represented as $observed(\text{“}Antonio\text{”})^{\langle 28,49\rangle}$. Please note that the $face(\text{“}Antonio\text{”}, 70)$ event is generated after the time 28 due to the face recognizer processing, however it is time stamped with the time of taking the picture in which the Antonio face was recognized.

The event-processing tasks performed by a SMC are implemented using Etalis event rules. Each event rule specifies an event to be detected based on a pattern of the occurrences of other events, and the SMC knowledge base represented as a Prolog program. Etalis provides an expressive language for specifying complex event patterns allowing the representation of various temporal relationships between the occurrence time intervals of events and also non-occurrence of an event between the occurrences of two other events.

The provided event-processing support facilitates a modular and efficient implementation of a wide variety of sensory data processing tasks in SMCs. These include filtering, detection of complex patterns and integration and transformation of data which are common in robotic applications [3], [6].

Example 2. $objRec(O, Pos)$ events specify recognized objects and their relative positions to the camera. $camTF(CTF)$ and $baseTF(BTF)$ events respectively specify the camera position relative to the base and the base position relative to the world over time. These events are received by a SMC asynchronously. Whenever an object is recognized and it's of the type “soft drink”, the event rule 1 in the SMC[2] calculates and outputs its absolute positions by generating a new event of the form $softDrinkRec(APos)$. A drink's absolute position is calculated based on camera and base positions at the time the drink was recognized using the $equals$ operator which requires two events to have happened at the same time. Please note how, for example, ontological knowledge (knowledge of soft and hard drinks) is seamlessly used in the event rule.

$$softDrinkRec(APos) \leftarrow objRec(O, Pos)\ equals\ camTF(CTF)\ equals\ baseTF(BTF)\ where\ (softDrink(O),\ APos = BTF \times CTF \times Pos). \quad (1)$$

Example 3. Rule 2 detects a face as a reliable one if within 5 seconds it is recognized 3 times with more than 60% reliability in the increasing order.

$$relFace(F) < - (face(F, R1)\ seq\ face(F, R2)\ seq\ face(F, R3)).5sec\ where(R3 > R2 > R1 > 60). \quad (2)$$

A SMC receives events asynchronously in an order which does not necessarily correspond to their occurrence times. Regardless of this, temporal relations in event rules are evaluated based on event occurrence times and not the order of their arrival to SMCs. Another point worth mentioning about Etalis is its garbage collection mechanisms pruning events which can no longer contribute in detecting any desirable event.

III. Component Interactions

SMCs as mediators between components enable two high-level interaction mechanisms among them. By the first one, components can subscribe themselves or the other components to certain events of interest at runtime. For example when the robot is having the goal of serving Antonio a drink, the control component can subscribe itself (using its unique id) to a SMC which processes $face(F, R)$ events for events of the reliable recognition of Antonio by sending the request $register(ID, s\_win^{\langle now,+\infty,<relFace(F),F=“Antonio''>\rangle})$ specifying its interest for such events from now on. Later, it can unsubscribe from those events. Another example is when the robot is seeing two moving objects and receives an order from the user to follow the $obj1$ by its head. To this end, the control component can subscribe the gaze control component to events of the $obj1$ position over time. This helps to avoid overloading the control component (with heavy cognitive capabilities) with a large number of events unrelated to its operational context.

The second interaction mechanism is provided by enabling components to specify at runtime certain histories of events of interest to be kept and maintained by SMCs and to query the histories on-demand. A history of events of a certain type and content can be maintained using one of the following policies: 1) the N most recent instances of such events, 2) such events with occurrence time interval within the last l seconds, or 3) such events of which occurrence times are within a certain period of time. The management of history using such policies in SMC supports to deal with the mismatch in communication styles of components [3]. For example a component which processes events of a certain type with a lower frequency than the frequency at which those events are generated can ask a SMC to keep the N most recent instances of such events to access them on demand.

[2]For the sake of readability, the syntax of event rules in this paper might be slightly different than the actual Etalis syntax

IV. Integration with ROS

The sensory management library is implemented in Prolog and accessed through a Java interface. The only essential requirement of a framework to integrate the library is supporting a publish-subscribe communication mechanism. Although the on-demand query mechanism of the library is naturally implemented on top of a service-based communication mechanism, in an asynchronous setting such interaction could be simulated by a proper token passing among the interacting components. We have developed a prototype integrating the library in ROS. It provides a user-friendly interface (XML configuration file) to subscribe a SMC to ROS topics as its input event streams. ROS messages received on the subscribed topics are automatically converted to the corresponding event formats to be consumed by the SMC. ROS components can subscribe to, and query the events processed and managed by SMCs at runtime. The communication between SMCs and other ROS components are realized using ROS communication mechanisms.

V. Conclusion

A light-weight software library is introduced which facilitates a systematic implementation of sensory management components for an autonomous robot. The library integrates the efficient language of Etalis to support the implementation of complex event-processing tasks and extends it with dynamic subscription and on-demand query mechanisms to support a number of high-level inter-component interaction mechanisms.

We are currently preparing a setup to test our approach in a knowledge-processing and interacting application scenario for the NAO robot, validating it in terms of usability, reusability and efficiency. The future work is to extend its functionalities with other types of history management and querying to support the implementation of active memories similar to [5].

References

[1] Darko Anicic. Event Processing and Stream Reasoning with ETALIS. *PhD Thesis, Karlsruher Institute of Technology*, 2011.

[2] Nick Hawes. Building for the Future: Architectures for the Next Generation of Intelligent Robots. *Proceedings of a Symposium held in Honour of Aaron Sloman*, 2011.

[3] I Lütkebohle. Facilitating re-use by design: A filtering, transformation, and selection architecture for robotic software systems. *Software Development and Integration in Robotics conference*, (section III), 2009.

[4] I Lütkebohle et al. Generic middleware support for coordinating robot software components: The Task-State-Pattern. *Journal of Software Engineering for Robotics (JOSER)*, 2(1):20–39.

[5] Sebastian Wrede and Marc Hanheide. An active memory as a model for information fusion. *.... on Information Fusion*, 2004.

[6] Pouyan 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.

[7] Pouyan Ziafati et al. Event-Processing in Autonomous Robot Programming. *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, 2013.