Adding Episodic-Like Memory To An Event-Based System

Dennis Stachowicz and Nick Hawes and Geert-Jan M. Kruijff

Abstract— **In this paper we describe how the use of event-based middleware allowed us to extend an existing intelligent robot system with functionality related to episodic-like memory.**

Index Terms— **episodic-like memory, event-based software, intelligent robotics**

I. INTRODUCTION

When developing intelligent robots there are many desirable qualities you may look for in your software middleware, e.g. speed, robustness, debugging tools, hardware support etc. When working on scientific projects over an extended period of time, a particularly desirable feature is for your middleware to allow easy extension to, and experimentation with, an existing system. This is crucial when a large part of your software engineering work is concerned with exploring possible designs for new architectural mechanisms and the trade-offs they provide, as opposed to delivering a single finished system. The flexible nature of event-based systems (EBSs), means they are ideal for this kind of exploratory software engineering. In this paper we use a case-study from a cognitive robotics project to demonstrate how EBS middleware allowed us to extend an existing integrated system with only minimal alterations to the initial codebase.

We start the rest of the paper by presenting CAST, the EBS middleware we used, and follow this with some background on systems we have previously built with it, including the Explorer, the system which forms the basis of the case-study (see Section III). The case-study involves the addition of *episodic-like memory* (ELM) to the Explorer. In Section IV we describe what ELM is, how we implemented an ELM system in CAST, and how we integrated this with the Explorer. Finally, in Section V we discuss the properties of CAST (and thus the event-based system paradigm) which made our approach feasible.

II. CAST

The CoSy Architecture Schema Toolkit[1] (CAST) [1] is an open-source component-based middleware toolkit designed to support the design and development of information-processing architectures for intelligent systems. Components in a CAST system do not exchange information directly, but instead share data via multiple *working memories* (WMs). CAST systems are decomposed into *subarchitectures* (SAs), where each SA is a group of components directly connected to a single WM. Components in a SA gain fast access to their local WM, whereas access to other WMs may be mediated by additional communication or translation mechanisms. Components can either add data to, overwrite data on, or delete or read data from WMs, which in turn generate *change events* (CEs) to describe these operations. Components are typically activated via callbacks which are configured to listen to appropriate fields within CEs (e.g. WM address, data type or access mode).

CAST's use of CEs mean that it can be classified as an event-based system (EBS) and thus falls within the remit of this workshop. The use of CEs combined with the data persistence provided by WMs give CAST the flexibility associated with the EBS paradigm whilst minimising communication overheads (as object data is only transmitted when required, not with every event). CAST can be most closely compared to the framework used by the information-driven integration work of Wrede et al. (e.g. [2]).

III. THE EXPLORER SYSTEM

CAST has been used to develop a number of intelligent robot systems with a variety of competences [3], [4][2]. CAST systems are typically developed incrementally, with new components (or whole SAs) being added to build on the results of existing components. As all information is shared by default on WM, such extensions incur a low (design and implementation) overhead. We have shown previously how this feature of CAST has allowed us to extend a system with addition modal SAs [3]. In this short paper we outline how we have also been able to add a new modality-general functionality (episodic-like memory) to an existing system in a similar way. The existing system is the Explorer, a mobile robot capable of building conceptual maps of space, and interacting with, and performing tasks for, a human [4]. The Explorer runs on an ActivMedia PeopleBot and is composed of six CAST SAs: object-based visual search, navigation, conceptual mapping, communication, cross-modal binding, and planning. This combination of competences allows the Explorer to extend SLAM maps with conceptual representations; search for specified objects; and interpret and generate references to objects and spatial representations, all within a unified planning and cross-modal fusion framework.

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

D. Stachowicz and G.J. M. Kruijff are with the Language Technology Lab, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany {Dennis.Stachowicz, gj}@dfki.de

N. Hawes is with the Intelligent Robotics Lab, School of Computer Science, University of Birmingham, UK n.a.hawes@cs.bham.ac.uk

[1]Available for download from http://www.cs.bham.ac.uk/go/cast

[2]See http://cognitivesystems.org and http://cogx.eu for videos and further publications.

IV. ELM IN THE EXPLORER

Although a robot running the Explorer system can show a wide range of behaviour and acquire semantic knowledge (e.g. in conceptual mapping tasks), it lacks the ability to represent, store and retrieve information about past events. The capability to remember past experiences in their original spatio-temporal context has been termed *episodic memory* [5] or *episodic-like memory*[3] (ELM) [6]. Humans often and naturally use ELM, e.g. when they talk about their experiences, or when they search to find a lost object ("Where was I when I last had it? What did I do then?").

The ELM system we have developed enables a robot to represent, store, retrieve and consequently leverage experiences from its past. It consists of a system-independent ELM library (providing the core functionality) which is integrated with the Explorer using a CAST ELM interface (C-ELM). Experiences are represented in the ELM library by data structures called *events* (not to be confused with the change events above!) which integrate information about what the robot experiences, and where and when these experiences occurred. Such events can be either *atomic* in nature (e.g. when an object is detected instantaneously) or *complex* (i.e. consist of several other (sub-)events, e.g. a social interaction). Complex events are constructed using event recognition modules. Events are stored in a relational database with spatial extensions. After insertion into the database, events can be retrieved using an under-specified event structure as a cue. Cues can, for example, contain an event type, a time interval, an area, or a combination of these. Related systems in robotics include ISAC's episodic memory system [7] and EPIROME [8].

The C-ELM integration layer is based on a relatively simple approach: the three main ELM processes (insertion into, and retrieval from, the memory store, and complex event recognition) are wrapped into three corresponding CAST components. Communication between these components and other CAST components (i.e. those which create or retrieve events) is performed via C-ELM datatypes on WM. C-ELM is used in the Explorer through the addition of the C-ELM SA (which contains the core ELM processes) and a number of system-specific *monitor components*. The monitor components listen for particular CEs, then translate the changed information into C-ELM events which are written to the C-ELM SA where they are stored in the event database. This is the core processing model used in the Explorer C-ELM. For a simple example of this approach, consider how we record events concerning the Explorer's position: the component `TopologicalPositionMonitor` is added to the navigation SA, where it listens for CEs describing overwrites to the Explorer's topological position struct on navigation WM (this struct describes which room the Explorer is currently in). When such CEs are received it loads the position information from WM and translates it into a C-ELM event which is written to C-ELM WM and stored. Event recognition is then used to construct complex events from these and other events.

[3]The notion of episodic-like memory is very similar to that of episodic memory but does not rely on purely subjective concepts, and is instead restricted to features accessible by behavioural experimentation and observation.

V. ADVANTAGES OF AN EVENT-BASED ARCHITECTURE

Although the above functionality could be added to any software architecture, the use of CAST provides a number of advantages. First, because the components in the Explorer already share information by default (data persists on WM until actively deleted), no code in the original system had to be altered to provide access to data for the new C-ELM monitoring components. This fact, plus the loose coupling provided by the event-based nature of CAST, means that the Explorer can now be run in its original instantiation, or with the addition of ELM, with only a change to the file which specifies which components to run: a useful ability when developing and debugging. Finally, the use of CAST means that an extension to either the Explorer (e.g. the addition of a new SA) or to C-ELM (e.g. the addition of new ELM functionality) can happen incrementally, and with no change necessarily required in the other system.

VI. CONCLUSION

In this paper we have described an implementation of episodic-like memory and how we integrated this implementation into the Explorer, an existing intelligent robotic system. This integration was made feasible because the Explorer was implemented using CAST, an example of event-based software. The loose-coupling, and thus flexibility, provided by the EBS paradigm, coupled with the information sharing enforced by CAST, meant that this exploration in the space of architectures for an intelligent system was easier to perform than with more traditional middleware packages.

REFERENCES

[1] N. Hawes, M. Zillich, and J. Wyatt, "BALT & CAST: Middleware for cognitive robotics," in *Proceedings of IEEE RO-MAN 2007*, August 2007, pp. 998 – 1003.

[2] S. Wrede, "An information-driven architecture for cognitive systems research," Ph.D. dissertation, Bielefeld University, 2008.

[3] N. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G.-J. Kruijff, M. Brenner, G. Berginc, and D. Skočaj, "Towards an integrated robot with multiple cognitive functions," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2008)*, R. C. Holte and A. Howe, Eds. Vancouver, Canada: AAAI Press, July 2007, pp. 1548 – 1553.

[4] N. Hawes, H. Zender, K. Sjöö, M. Brenner, G.-J. M. Kruijff, and P. Jensfelt, "Planning and acting with an integrated sense of space," in *Proceedings of the 1st International Workshop on Hybrid Control of Autonomous Systems – Integrating Learning, Deliberation and Reactive Control (HYCAS)*, Pasadena, CA, USA, July 2009, pp. 25–32.

[5] E. Tulving, "Episodic memory: from mind to brain," *Annual Review of Psychology*, vol. 53, pp. 1–25, 2002.

[6] N. S. Clayton, T. J. Bussey, and A. Dickinson, "Can animals recall the past and plan for the future?" *Nature Reviews Neuroscience*, vol. 4, no. 8, pp. 685–691, Aug 2003.

[7] W. Dodd and R. Gutierrez, "The role of episodic memory and emotion in a cognitive robot," in *Proceedings of 14th Annual IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), Nashville, TN, August 13-15, 2005*, 2005, pp. 692–697.

[8] S. Jockel, D. Westhoff, and J. Zhang, "Epirome - a novel framework to investigate high-level episodic robot memory," in *Proc. IEEE International Conference on Robotics and Biomimetics ROBIO 2007*, 2007, pp. 1075–1080.