Commercial Software Tools for Intelligent Autonomous Systems

S M Veres, N K Lincoln * J P Adolfsson, L Molnar **

* *University of Southampton, Engineering, Highfield, SO17 1BJ, UK, email: s.m.veres@soton.ac.uk*
** *SysBrain Ltd, No 10, SO19 9TE, UK*
*** *University of Liverpool, Computer Science, Liverpool, L69 7ZF, UK*

Date: 20.02.2012

Abstract: This article identifies some of the commercial software that can potentially be examined, or relied upon for their techniques, within a new 4 year EPSRC project "Reconfigurable Autonomy" to be undertaken between Liverpool, Southampton and Surrey Universities. Although such projects strive to produce new techniques of various kinds, the software reviewed here could also influence/shape, or help to integrate the algorithmic outcome of, all 8 projects awarded wihtin the EPSRC *Autonomous and Intelligent Systems* programme. To avoid mis-representation of technololgies provided by the software producer companies listed, most of this review is based on using quotes from original product descriptions.

Keywords: Autonomous systems, intelligent systems, physical agents, adaptive control, hybrid system abstractions, rational agents, agent programming languages, decision and control.

1. PROJECT BACKGROUND

The EPSRC projects EP/J011770, EP/J011843, and EP/J011916, running between 30.3.2012 and 29.3.2016, at the universities of Liverpool, Southampton and Surrey have a shared *summary*:

> "As computational and engineering applications become more sophisticated, the need for autonomous systems that can act intelligently without direct human intervention increases. Yet the autonomous control at the heart of many such systems is often *ad-hoc* and *opaque*. Since the cost of failure in critical systems is high, a more reliable, understandable and consistent approach is needed. Thus, in this project we aim to provide a rational agent architecture that controls autonomous decision-making, is re-usable and generic, and can be configured for many different autonomous platforms. In partnership with the industrial collaborators we aim to show how such *reconfigurable autonomy* can be achieved in relevant applications."

Given this summary, the project is expected to provide agent shells for applications that can be easily configured (and reconfigured) for various applications, and as such it aims to reduce development costs to companies while delivering higher performance and integrity systems.

2. COMMERCIAL SOFTWARE REVIEWS

This section reviews some of the relevant commercial software from the past that may not be new, but which provides as comprehensive view as was possible given resources available to the authors at the time of writing.

⋆ The authors are grateful to Michael Fisher and Yang Gao for their comments to improve the paper. Sponsored by SysBrain Ltd, Reg. England and Wales 04583971, Email:j.p.adolfsson@sysbrain.com

2.1 SRI's Procedural Reasoning Systems (PRS)

According to their own description (Myers, 1993) SRI's procedural reasoning system (PRS-CL):

> PRS software was developed "for representing and using an expert's procedural knowledge for accomplishing goals and tasks. Procedural knowledge amounts to descriptions of collections of structured actions for use in specific situations. PRS-CL supports the definition of real-time, continuously-active, intelligent systems that make use of procedural knowledge, such as diagnostic programs and system controllers. PRS-CL's architecture consists of (1) a database containing current facts and beliefs, (2) a set of goals to be achieved, (3) a set of plans or procedures describing how certain sequences of conditional tests and actions may be performed to achieve certain goals or to react to certain situations, and (4) an interpreter that manipulates these components to select and execute appropriate plans for achieving the system's goals."

A theoretical background to PRS has been laid down by (Georgeff and Lansky, 1987).

2.2 SciSys Ltd, UK

SciSys Ltd advertises three software system capabilities: *MMOPS*; *SARA*; and *OverSeer*.

- MMOPS - According to their own description, (SciSys-Web, Feb 2012), their Mars Mission On Board Planning System (MMOPS) "was developed to serve

the needs of remote planetary landers and mobile rovers. These platforms are typically commanded with a plan of tasks or actions prepared by operators on the ground. However these plans are prepared without a clear real-time picture of the actual state of the remote platform. Furthermore, operators have no opportunity to change the plans in real-time to meet mission goals. MMOPS uses stripped down, state of the art planning technology to ensure that uploaded plans can be managed in order to meet the original goals as state changes. It can change plans provided by human operators safely using models of how the platform will evolve given its current condition. Crucially, it provides an automated look-ahead capability to determine which course of action will achieve the mission goals with minimum use of resources. This ensures optimal platform utilisation. As new and potentially unforeseen opportunities arise, MMOPS can work out how to safely inject additional actions into the original plan in order to take advantage of the evolving operational situation."

- SARA is a "configurable computer vision based application which detects targets of interest in the scene and suggests additional actions in order to increase the relevance of the data required with respect to mission goals. SARA was initially developed to serve the needs of planetary rovers which require some degree of intelligence in order to determine when to take high-resolution images of science targets without human intervention. It can identify regions of interest in both wide angle and narrow angle images and provide an assessment of the basic visual and structural characteristics of objects in a scene. Although initially developed for space science applications it has also been adapted for terrestrial surveillance. It is fully compatible with the MMOPS mission management application and will provide requests to this application for re-planning in order to take advantage of opportunistic events detected by the Science Assessment element of SARA. The application is built from a number of discrete, automated blocks and can be configured or evolved based on the operational demands of any particular domain and application." (SciSys-Web, Feb 2012)
- OverSeer is, by their own description (SciSys-Web, Feb 2012), a "complete platform for accessing and controlling any type of robot. It provides many of the components needed to build interfaces and to integrate with wider operational systems. It is built using a number of components that can be adjusted, added to or combined to develop, test and implement robotics applications OverSeer is made up of software components that communicate using the CORBA standard. They are made up of two main types: *Servers* that interface to the hardware elements, for example to drive the wheels, retrieve information from the GPS or move the camera pan-tilt mechanism. *Clients* that provide an interface to these servers, drawing from a library of interface types, these give the user control of the platform and access to the data being collected.

  These clients can provide direct manual control or interface to autonomy applications run remotely or on board the platform. The interfaces presented by these clients can be combined or 'docked' into the OverSeer application or run as discrete windows within the operating system. Clients can be combined with other software elements to make seamless mission control systems tailored to particular operational needs or more intelligent autonomy applications. SciSys has successfully integrated OverSeer components into a range of environments, from complex control applications to smartphones such as the iPhone."

2.3 Agent Oriented Software Pty, AUS

Agent Oriented Software (AOS) has five products that they describe as follows (AOS-Web, Feb 2012).

- C-BDI implements "the BDI (Belief/Desire/Intention) model in the C programming language. C-BDI is a complementary product to JACK. In utilising the C language, it has the potential to be included as part of a certificated onboard system, including a sense & avoid capability. It comprises (1) a BDI agent platform written in C, (2) a focus on autonomous and semi-autonomous, mission-critical aerospace & defence systems, (3) is designed for use in real-time domains, and (4) is packaged as a set of libraries in ANSI/ISO C (C90)."
- The Intelligent Prognostic Health Manager (iPHM), developed by AOS, is "an autonomous system that takes prognostics to the next level of capability. It combines existing fault diagnosis techniques with Estimated Time to Failure under various conditions and provides recommended courses of action - for example to a military pilot, ship's engineer or an onboard Mission Management System. Utilising iPHM, autonomous systems can intelligently react and accommodate failures, allowing them to maximise their remaining capability."
- ISR Broker - supports "the changing task profiles within a mission, future military fighter, strike and surveillance aircraft will need to continuously obtain updated external sensor data from the Global Information Grid (GIG) network. To meet this requirement AOS is undertaking the development of a Reactive ISR Information Broker (ISR Broker) for the on-board interfacing to aircraft mission systems. AOS is utilising its Intelligent Agent autonomous software technology to form the core of the system. ISR Broker can also be utilised for other NCW platforms such as UAVs, Command & Control Centres, Naval Platforms and Land Vehicles."
- IMAPS — Progress in network-enabled warfare capability has reasulted in improvement in near real-time situational awareness that requires frequent re-tasking of both manned aircraft and UAVs. In this context re-planning the mission tactics in detail can be the bottleneck in the overall re-planning cycle. As AOS describes "the current generation of flight planning systems are designed to generate a single flight plan in full detail, but not to rapidly generate 'what if' alternative tactical mission scenarios. IMAPS is being developed by Agent Oriented Software to meet this new military requirement, providing mission planning staff with a comprehensive decision

support aid thereby allowing them rapidly to asses new mission taskings, taking into account:
- military/political constraints, such as Rules of Engagement;
- aircraft capabilities, weapons load and fuel trade-offs;
- equipment status, search radar / counter measures availability;
- nature of the battlefield environment, location of enemy defences and friendly support aircraft.

IMAPS is based upon the JACK Intelligent Agent Toolset, including the JACKSim graphical simulation environment."

- Intelligent Virtual Forces (IVF). Human trainees interact with "virtual forces play the roles of other pilots and ground controllers. For example, in our cockpit simulator environment, the human pilot talks on the radio and *Virtual Forces*, playing roles of pilots and controllers, responds with speech. JACK Intelligent Agents and JACK Teams, linked with speech recognition provide:
  (1) ability to converse with simulated entities
  (2) high-level graphical representation of human behaviour (goals, plans, beliefs)
  (3) flexible representation of teams and command, control and communications (C3) structures

  JBenefits is unique to Intelligent Virtual Forces:

  (JACK intelligent agents provide realistic, complex human behaviours and tactics:

  - reduced personnel requirements, lower staffing costs (human/agent substitution)

  - consistency of training experience, without false training caused by predictable scripted entities

  - pilots can practice in large scenarios - more realistic training experience

  - supports deployable trainer, where human operators are not available

  - easy to incorporate new doctrine and tactics - ensures training environment representative of current operational environment

  - graphical behaviour representation - supports V&V by operational personnel

  - JACK agents can exhibit human variability."
- JACK Autonomous Environment in a Box (JAE-Box) "provides a decision-making capability on board unmanned vehicles (eg, UAVs), flexible manufacturing systems, and on-board PHM applications. Based on Agent Oriented Software's well-proven JACK Intelligent Agents software, JAE-Box is suited to the time-critical operational environments. Early customers for this approach include DSTO Australia, QinetiQ Limited UK, and Monterey Technologies (under contract to the US Army).

  JAE-Box is a ruggedised single-board computer for installation in vehicles or systems, designed for the rigours of operational use, and is connected by a serial link to the vehicle or system control system (eg, UAV autopilot).

  Benefits include:
  (1) autonomous operations in rapidly changing environments where the vehicle must adapt its behaviour to achieve its mission;
  (2) easy high-level programming of intelligent behaviours;
  (3) ability to monitor vehicle's intentions during operation, keeping human operator *in the loop*; and
  (4) straightforward integration with control systems."

2.4 SysBrain Ltd, UK

The above listed product descriptions speak about optimised planning/replanning and PRS/BDI agent architectures that can be deployed in some valuable applications. They do not consider how complex intelligence will be efficiently created in any convincing way, apart from saying that a "user friendly programming environment" that is easy to program will be used. Also they do not mention, or appear to offer, any systematic development for:

(1) how high complexity "machine knowledge" will be represented in a form that it can be shared with human being operating these machines;
(2) how new autonomous system development will reach efficiencies of magnitudes of greater orders by developing technical publications for machines instead of programming; and
(3) how meta-level agent knowledge will be gradually improved.

SysBrain Ltd's recent developments offer answers to these questions while in fact adopting some of the programming paradigms of AOS, PRS, etc, and are capable of incorporating the advanced planning techniques of SciSys.

- sEnglish Publisher (sEP) (SysBrain-Web, Feb 2012b,F), by the company's own description "... is a tool to create documentation in PDF or HTML formats that can include 'executable' sEnglish sections. Each executable sEnglish section is a sequence of English sentences that describes a procedure or method. The author can set the detail in which a procedure is described. Best practice is to describe in English everything that makes the new contribution by the author comprehensively described. Standard or previously published algorithms can be referred to and placed into a zip file that the 'Reader Tool' of sEP automatically extracts into a project. Automatic extraction of sEnglish sections from published PDF and HTML papers into an sEnglish project that is, by design, identical to the project of the author is part of sEP. The sEnglish sections in a PDF or HTML publication can also be read by suitably programmed rational agents called sysbrains also offered by SysBrain Ltd under the CAT package. As opposed to alternative solutions to executable papers, there are no annotations used in sEnglish sections — what you see in printout or in the PDF file on screen is the whole relevant information that the machine will also interpret. sEP is the world's first publication system for both humans and machines simultaneously. sEnglish Publisher is implemented on Eclipse as a perspective alongside LaTeX and uses MATLAB MCR to execute MATLAB code compiled from sEnglish texts. LaTeX can be used on the same Eclipse installation alongside editing an sEnglish project to make preparing or run the demos of an executable conveniently."

- Cognitive Agents Toolbox (CAT) (SysBrain-Web, Feb 2012b) is, by the company's own description "... a toolbox that includes Rational Agent Editing under Jason+sEnglish, Jason+, and the MCMAS (Model Checker for Multiagent Systems) as an extension of sEnglish Publisher. CAT is a toolbox that includes components as extensions of sEnglish Publisher:
  (1) The Jason+sEnglish to Jason+ compiler and the Jason+sEnglish to ISPL compiler for formal modelling and verification of agent operations under MCMAS. The MCMAS formal verification system can be run on the same Eclipse alongside an sEnglish project defining the concept, modelling structures, relationships and reasoning of an agent. Reasoning of an agent includes English sentences based expressions of (1) Initial beliefs (2) Initial Goals (3) Declaration of perception processes (4) Generic behaviour rules at high abstraction levels (5) Plans to execute intentions (6) Plans to achieve goals.
  (2) The Agent Executive Toolbox (AET) for MATLAB. This is a library of M-files and Simulink blocks with examples of their use to complement the Jason+ system with external calls for signal processing and control. There is socket based based communication between Jason+sEnglish based reasoning and processes for sensing and control under MATLAB. The high level code can be compiled into executables of reasoning, sensing and action taking for embedded applications.
  (3) All the prototype Jason+sEnglish and AET processes can be compiled in CAT into a multithreaded C++ application for RTOS platforms that enables their implementation on small devices as welll as on large very complex systems."

3. CONCLUSIONS

This review has presented sample software platforms available commercially that can be used to host system capabilities in terms of sensing, control action, learning of skills, Bayesian learning of situational awareness, etc. The following table is a summary of features.

S/W	Replanning	Rational	Knowledge-based
MOPS	+	-/+	-
SARA	+	-/+	-
C-BDI	+	+	-
iPMH	+	+	-
IVF	+	+	-
MOPS	+	+	-
JAE-Box	+	+	-
- sEP	+	+	+
CAT	+	+	+

Table 1. Comparison table of commercial software for autonomy

Table 1 is based on what information was available from product descriptions. The selected comparisons have been based on three important features for developers of autonomous system:

(F1) *RT Replanning.* Goal achieving capability by real-time planning and replanning. It appears that SciSys products are the strongest in this feature. It is worth noting, however, that planning and replanning is integral part of the two BDI agents by AOS and SysBrain.

(F2) *Rational agency* is something that is not explicit in SciSys products but planning sequences are available for plans prepared and their execution. BDI agents of AOS and SysBrain make behaviour rules and logic for problem solving part of their solution. SysBrain's sEnglish based representation of plans and rules means that explanations of why an agent made a particular decision can be directly generated and communicated to human operators.

(F3) *Knowledge Centric.* The most striking difference is that relative to AOS and SciSys, SysBrain makes explicit usage of human concepts part of the core of agent reasoning. In the other two packages there can be only an indirect, natural-language-interface-based solution to sharing of concepts between human operators. For SciSys and AOS this is not a central issue but possibly obtainable by extensions of their software.

One may argue that there is no need to be "knowledge centric" in agent programming as human knowledged can be represented in abstract traditional Agent Oriented Programming (AOP). The variables used in AOP do need however to be remembered and understood by a large team of engineers. Misunderstandings often lead to bugs.

For easy sharing of functional knowledge between engineers in a team, the advantages of using of natural language programming in sEnglish also appears in that sections in technical papers can be published in sEnglish in so called "executable papers". These type of executable papers can be read by suitable agents produced under CAT. Hence learning by reading publicaitons becomes important for artificial autonomous systems. Old hardware can be used for as long as the agent can autonomously renew its knowledge and skills.

Finally, while we have examined some of the commercal software relevant to autonomy, it is important to note that there are many *academic* activities in this direction, often providing agent programming tools and environments in either *open source* or *GPL* form. Notable examples include Jason Bordini et al. (2007); JADE F.Bellifemine et al. (2007); and 3APL Dastani et al. (2010):

- As an agent orinted programming language the current version of Jason has the following features by their authors Bordini& Hübner: "(1) Strong negation: as is well known in the ALP community, close-world assumption is not ideal for open systems where uncertainty cannot be avoided; it helps the modelling of such applications if we are able to refer to things agents believe to be true, believe to be false, or are ignorant about.
  (2) Handling of plan failures: because of the dynamic nature of typical multi-agent environments, plans can fail to achieve the goals they were written to achieve; one important aspect of reactive planning systems is that the particular choice of the specific plan to achieve a goal is left for as late as possible so as to consider the latest information the agent might

have, but of course plans can still fail. Jason has a particular form of plan failure handling mechanism consisting of plans that are triggered by such failure, giving the programmer the chance to act so as to undo the effects of any action already done before the plan failed, if necessary, and then adopting the goal (that was not achieved) again, if the conditions are appropriate.

(3) Speech-act based communication: the philosophical foundation for all the work on inter-agent communication is speech-act theory; because mental attitudes which are classically used to give semantics for speech-act based communication are formally defined for AgentSpeak we can give precise semantics for how agents interpret the basic illocutionary forces, and this has been implemented in Jason. An interesting extension (Note that annotations as used here do not increase the expressive power of the language but are an elegant notation, making the belief base much more readable) of the language is that beliefs can have "annotations" which can be useful for application-specific tasks, but there is one standard annotations that is done automatically by Jason, which is on the source of each particular belief. There are essentially three different types of sources of information: percepts (i.e., information obtained by sensing the environment), inter-agent communication (i.e., information obtained from other agents), and "mental notes" (i.e., beliefs added by the agent itself which can facilitate various programming tasks).

(4) Plan annotations: in the same way that beliefs can have annotations, programmers can add annotations to plan labels, which can be used by elaborate (e.g., using decision-theoretic techniques) selection functions. Selection functions are user-defined functions which are used by the interpreter, including which plan should be given preference in case various different plans happen to be considered applicable for a particular event. "

- The Jade agent programming language, again by the author's F.Bellifemine et al. (2007) specficiation: "The goal of JADE is to simplify the development of multi-agent systems while ensuring standard compliance through a comprehensive set of system services and agents in compliance with the FIPA specifications: naming service and yellow-page service, message transport and parsing service, and a library of FIPA interaction protocols ready to be used. The JADE Agent Platform complies with FIPA specifications and includes all those mandatory components that manage the platform, that is the ACC, the AMS, and the DF. All agent communication is performed through message passing, where FIPA ACL is the language to represent messages. The agent platform can be distributed on several hosts. Only one Java application, and therefore only one Java Virtual Machine (JVM), is executed on each host. Each JVM is basically a container of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host.

  The communication architecture offers flexible and efficient messaging, where JADE creates and manages a queue of incoming ACL messages, private to each agent; agents can access their queue via a combination of several modes: blocking, polling, timeout and pattern matching based. The full FIPA communication model has been implemented and its components have been clearly distincted and fully integrated: interaction protocols, envelope, ACL, content languages, encoding schemes, ontologies and, finally, transport protocols. The transport mechanism, in particular, is like a chameleon because it adapts to each situation, by transparently choosing the best available protocol. Java RMI, event-notification, HTTP, and IIOP are currently used, but more protocols can be easily added via the MTP and IMTP JADE interfaces. Most of the interaction protocols defined by FIPA are already available and can be instantiated after defining the application-dependent behaviour of each state of the protocol. SL and agent management ontology have been implemented already, as well as the support for user-defined content languages and ontologies that can be implemented, registered with agents, and automatically used by the framework.

  Basically, agents are implemented as one thread per agent, but agents often need to execute parallel tasks. Further to the multi-thread solution, offered directly by the JAVA language, JADE supports also scheduling of cooperative behaviours, where JADE schedules these tasks in a light and effective way. The run-time includes also some ready to use behaviours for the most common tasks in agent programming, such as FIPA interaction protocols, waking under a certain condition, and structuring complex tasks as aggregations of simpler ones. Among the others, JADE offers also a so-called JessBehaviour that allows full integration with JESS, where JADE provides the shell of the agent and guarantees (where possible) the FIPA compliance, while JESS is the engine of the agent that performs all the necessary reasoning. One of the examples shows integration between JADE, JESS and Protege.

  The agent platform provides a Graphical User Interface (GUI) for the remote management, monitoring and controlling of the status of agents, allowing, for example, to stop and restart agents. The GUI allows also to create and start the execution of an agent on a remote host, provided that an agent container is already running. The GUI allows also to control other remote FIPA-compliant agent platforms."
- 3APL is a programming language for implementing cognitive agents Dastani et al. (2010). The 3APL project is "carried out at Utrecht University (Institute of Information and Computing Sciences, Intelligent Systems Group) and is partially supported by the Dutch Research Council NWO. The work is closely related to other ongoing research projects such as epistemic and dynamic logics within our group, and AgentSpeak(L) and Golog/Congolog projects outside our group. Although the aim of the AgentSpeak(L) project, carried out at the Australian Artificial Intelligence Institute, was similar to that of 3APL, only a limited set of cognitive concepts can be implemented by their proposed programming language. The aim

of Golog/Congolog, under way at Toronto University and Aachen University of Technology Dastani et al. (2012), is to develop a programming language to implement the high level control of cognitive robots in dynamic and unpredictable environments. While 3APL provides programming constructs to implement a large set of cognitive concepts, Golog/Congolog concentrates on programming constructs, such as sensing and planning, which are essential for dynamic and unpredictable environments.

The 3APL project 3APL-Web (2012) has developed a programming language for implementing cognitive agents with beliefs, observations, actions, goals, communication, and reasoning rules. In particular, agents' observations and beliefs can be implemented in 3APL by a subset of first-order predicate language (prolog-like facts and rules). The actions are implemented as triples consisting of action name together with pre- and post-conditions. The pre- and post-conditions of actions are belief formulae indicating the condition under which the action can be performed and the effect of the action after it is performed, respectively. The goals that can be implemented in 3APL are procedural or to-do goals, which can be implemented by expressions of an imperative language. These expressions are formed by applying constructs such as sequence, test, conditional choice, and recursion to actions and belief formulae. Communication can be implemented by the pre-defined send-message action. Finally, reasoning rules can be used to implement the generation of goals, the revision of actions that are blocked, the revision of goals that are not achievable, the ptimization of goals, etc. "

REFERENCES

3APL-Web (2012). Official 3APL Website: An Abstract Agent Programming Language. www.cs.uu.nl/3apl/.

AOS-Web (Feb 2012). Autonomous Systems Development Software. http://www.agent-software.com.

Bordini, R.H., Hübner, J.F., and Wooldridge, M. (2007). *Programming Multi-agent Systems in AgentSpeak Using Jason*. Wiley.

Dastani, M., F.Dignum, and Meyer, J. (2012). Official 3APL Website: An Abstract Agent Programming Language. www.ercim.eu/publication/Ercim_News/enw53/dastani.html.

Dastani, M., van Riemsdijk, M.B., and Meyer, J. (2010). Programming multi-agent systems in 3APL. In *Multi-Agent Programming: Languages, Platforms and Applications*, chapter 2, 39–67. Springer.

F.Bellifemine, G.Caire, and D.Greenwood (2007). *Developing multi-agent systems with JADE*. John Wiley and Sons, LTD.

Georgeff, M.P. and Lansky, A.L. (1987). Reactive reasoning and planning. In K. Forbus and H. Shrobe (eds.), *6th National Conference on Artificial Intelligence (AAAI-87)*, volume 2, 677–682. AAAI Press, Seattle, WA, USA. Www.aaai.org/Papers/AAAI/1987/AAAI87-121.pdf.

Myers, K.L. (1993). *User's Guide for the Procedural Reasoning System*. SRI International Artificial Intelligence Center, Menlo Park, CA.

SciSys-Web (Feb 2012). Software for Autonomous Systems Development. www.scisys.co.uk/what-we-do/autonomy-robotis.aspx.

SysBrain-Web (Feb 2012a). Cognitive Agent Toolbox. http://cognitive-agent-toolbox.com.

SysBrain-Web (Feb 2012b). sEnglish Publisher. http://www.senglish.org.