Künstl Intell (2014) 28:199–207
DOI 10.1007/s13218-014-0308-1

RESEARCH PROJECT

Reconfigurable Autonomy

Louise A. Dennis · Michael Fisher · Jonathan M. Aitken · Sandor M. Veres · Yang Gao · Affan Shaukat · Guy Burroughes

Published online: 30 July 2014
© The Author(s) 2014. This article is published with open access at Springerlink.com

Abstract This position paper describes ongoing work at the Universities of Liverpool, Sheffield and Surrey in the UK on developing hybrid agent architectures for controlling autonomous systems, and specifically for ensuring that agent-controlled dynamic reconfiguration is viable. The work outlined here forms part of the *Reconfigurable Autonomy* research project.

Keywords Hybrid systems · Autonomous vehicles · Intelligent agents · Robotics

The *Reconfigurable Autonomy* research project is supported within the UK by EPSRC under Grants EP/J011770, EP/J011843, and EP/J011916, together with industrial collaborators who form part of the *Autonomous and Intelligent Systems Partnership*.

L. A. Dennis · M. Fisher
Department of Computer Science, University of Liverpool, Liverpool, UK
e-mail: l.a.dennis@liverpool.ac.uk

M. Fisher
e-mail: mfisher@liverpool.ac.uk

J. M. Aitken · S. M. Veres (✉)
Department of Automatic Control & Systems Engineering, University of Sheffield, Sheffield, UK
e-mail: s.veres@sheffield.ac.uk

J. M. Aitken
e-mail: jonathan.aitken@sheffield.ac.uk

Y. Gao · A. Shaukat · G. Burroughes
Surrey Space Centre, University of Surrey, Guildford, UK
e-mail: yang.gao@surrey.ac.uk

A. Shaukat
e-mail: a.shaukat@surrey.ac.uk

G. Burroughes
e-mail: g.burroughes@surrey.ac.uk

1 Autonomous Systems

Autonomy is the ability of an entity to make its own decisions and to act on its own, both without external intervention. *Autonomous Systems* is a term covering a broad class of systems that *decide for themselves* what to do and when to do it, in particular without direct *human* intervention.

The popularity of this class of systems is increasing rapidly. But why? And why now? Let us consider some of the justifications for turning to autonomy. Autonomous systems are particularly being developed in the following scenarios:

- When systems must work in *dangerous* environments where humans cannot be nearby, and so humans cannot assess the possibilities easily and quickly;
- Similarly, systems that must work in *remote* environments where direct human control is infeasible;
- Situations where systems need to react much more *quickly* than humans can possibly achieve;
- Scenarios where, while human control may be possible, there are just *too many* autonomous entities active for any one human to keep track of; or (increasingly);
- Where it is *cheaper* to use autonomous systems rather than involving a human pilot/driver/controller!

Given these motivations, aspects of autonomy are now appearing across a wide variety of practical systems, from autonomous vehicles, such as driver-less cars and unmanned air vehicles, through to robotics, both in the home and in wider society (Fig. 1). Less obviously, autonomy is an implicit part of feedback-control in engineering and sensor-based systems, such as those seen in pervasive, autonomic, and ubiquitous computing [15].

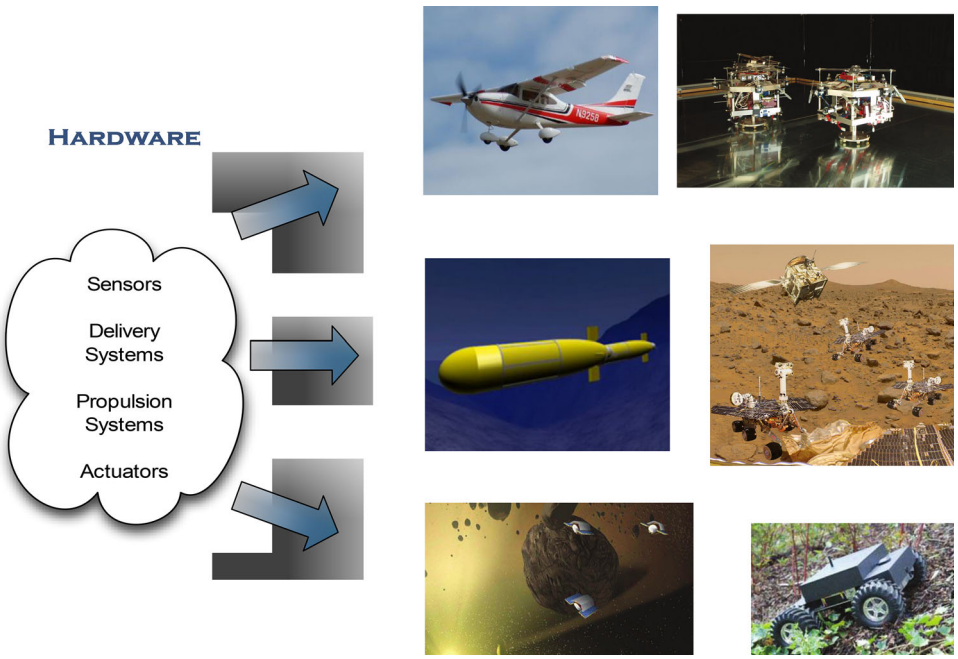


Fig. 1 Autonomy appears in many distinct practical systems

Many deployed autonomous systems comprise complex and intricate controllers, typically involving open and closed loop control systems, neural networks, or genetic algorithms.

While such controllers can be efficient for known dynamics of the environment and may be able to cope with the continuous nature of real-world interactions, such architectures are often quite hard to make work properly when there are frequent dynamical changes due to a varying environment. In such cases some situational awareness-based switching of controllers is needed. If stability can be guaranteed then an alternative is to use learning controllers.

On the other hand the operation of "successful" robust controllers for vehicles with possibly changing dynamics due to electro-mechanical and environmental variations can often be unclear (i.e. they are *opaque*). Since they are hard to understand, reliability is difficult to assess and both update and improvement can be tortuous. Even when such systems are modelled, for example, in terms of large sets of differential equations, these representations can also be opaque, difficult to analyze, and problematic to refine.

2 Assessing Autonomy

There are problems inherent in analyzing an autonomous system that operates feedback controllers and makes decisions to form a "hybrid system", i.e. a system with both continuous and discrete states. Even if traditional hybrid system analysis approaches were feasible then there is yet another problem. The key *new* aspect that complex autonomy brings is that an autonomous system must make decisions rather than a human controller making them. Thus, it is vital to be able to assess not only what a system does, but what decisions were taken, and why they were taken [2]. Extracting these decisions from complex control systems can be *extremely* difficult. It turns out that what we also need, and something that highlights the difference between an adaptive/automatic system and an autonomous one, is: (1) explicit *reasons* for making decisions one way or the other; (2) setting the system's own *feedback/feed-forward control targets* based on some form of reasoning, modelling and prediction. After all, a system can make decisions but we cannot assess whether it is reliable unless we know how it comes to these decisions. Only by understanding the underlying process can we truly trust the decisions that the system has taken [8].

2.1 Hybrid Agent Architectures

Developers of autonomous systems across many domains (air, space, underwater, robotics, software, etc) are coming to the realisation that we need a system architecture that captures, clearly and precisely, what it *does* (i.e. what series of actions it undertakes), what *choices* it made that led to these actions, and *why* it made one choice rather than another. This might seem quite ambitious but, fortunately, many decades of research on *rational agents* provide

theories, models, and implementations of exactly such entities [24].

What is an Agent? The 'agent' concept has been found to be a very useful abstraction for autonomous behaviour within complex, dynamic systems. Agents make decisions independently from their environment and, typically: work in environments that are both *dynamic* and *unpredictable*; can potentially *learn/evolve* new behaviour; act under varying *real-time constraints*; and are part of an *open system* (i.e. no fixed topology) and therefore have no central control. In general, they must be capable of *flexible autonomous action* [24].

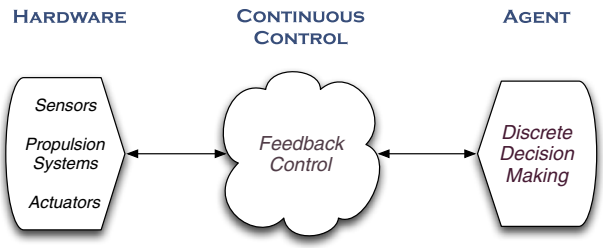
What is a Rational Agent? But the 'agent' concept is still not enough! Systems controlled by neural networks, genetic algorithms, control systems, etc, can act autonomously yet, as we suggested above, the reasons for their actions are often *opaque*. Consequently, such systems are often hard to develop, control and analyze and so the concept of a "rational agent" has become popular. This is an agent which *must have explicit* ***reasons*** *for making the choices it does, and should be able to explain these if necessary.* Such agents are often programmed and analyzed by describing their *goals* and *knowledge*, and how these change over time. Typically, rational agents can adapt their autonomous behaviour to cater for the dynamic aspects of their environment, requirements and knowledge.

Example: Spacecraft Landing Imagine a rational agent controlling a spacecraft that is attempting to land on a planet. The agent has: *control of dynamic activity*, e.g. velocity vector, attitude, etc; *information*, e.g. 'knowledge'/'belief' about the planet terrain, target landing sites, etc; and *motivations*, e.g. 'goals' such as to land soon, to remain aloft until safe to land, etc. The rational agent must dynamically

- Assess, and possibly revise, the information held,
- Generate new motivations or revise current ones, and
- Decide what to do, i.e. *deliberate* over its motivations and information.

2.2 Hybrid Agent Architectures

Thus, many contemporary autonomous systems (less so in the field of *Robotics*, where such architectures are relatively rare) employ a *hybrid* agent architecture, with control, neural, and genetic sub-systems being over-seen by a *rational agent*. These agents are high-level decision-makers and, crucially, their *rational* nature means that they should have (and be able to explain) reasons for making the choices they do:



Such hybrid agent architectures are increasingly common across many applications, from software autonomy to autonomous vehicles.

3 Towards Genericity

There have been several attempts to produce agent-based architectures for specific autonomous systems. We have developed *agent-based satellite formation flying* [1, 16, 17], *agent-based UAVs* [23], *agent-based underwater autonomy* (with Imperial College) [10], *agent-based reconfigurable mission planning* [12], essentially using similar styles of hybrid agent architectures. Our aim in the current *Reconfigurable Autonomy* project is to answer the question:

> *can a common "autonomy architecture" be provided and be configured for* different *autonomous systems?*

If so, then such an approach would be

- Easier to develop (through re-use of code/architecture),
- More reliable (since the particular core would be deeply analyzed and refined over several configurations), and
- Easier to deploy on new platforms.

So, we are generalizing and extending our hybrid agent architectures from [1, 16, 17, 23] to provide the basis for this. The two key aspects of this architecture will be that

(a) It is both rational and hybrid, allowing for reliable decision making in realistic environments, and
(b) It is extensible with the potential interfaces to the environment being precisely described within a "plug and play" interface library of sensors/controllers/actuators.

Consequently, our current work involves

1. Defining a core, open-source architecture, based on that introduced within [1, 16, 17, 23], but also incorporating key elements of reconfigurable mission planning from [12–14];

2. Investigating how a library of "environmental interfaces" can be incorporated into this architecture, with typical elements within the library corresponding to special control systems, networks interfaces, or infrastructure; and
3. In collaboration with a number of industrial partners, showing how this generic architecture can be instantiated, demonstrating a selection of *real* autonomous systems, and evaluating the *practical* efficacy of the instantiated architecture.

4 Towards Reconfigurability

An important aspect of our generic, hybrid, agent-based architecture is the separated, modular nature and the potential for *reconfigurability*. Basic reconfigurability concerns configuring the generic architecture for specific hardware, specific applications, and specific control systems. While such configuration might occur at start-up, can we change this configuration, or *reconfigure* it, during execution? The modular nature of our architecture can allow for a range of reconfigurability and, below, we identify some of the possibilities.

4.1 Reconfiguration due to Hardware

If the autonomous system loses some hardware (e.g. a thruster) then can the agent dynamically reconfigure the control systems to still achieve its mission with one less thruster? And can the same be done for any hardware, e.g. wheel, arm, sensor, etc?

But what if a new element of hardware is *added* rather than removed? Clearly control that takes this (for example, an extra thruster) into account must be organised by the agent.

Similarly, the agent itself might need to modify (or reconfigure) its high-level goal/plan selection to take into account the restricted/new possibilities. For example, with an additional sensor, the agent might be able to select or plan more complex missions.

4.2 Reconfiguration due to Control

Change in the controller need not be due to change to the hardware. Updates may be caused by many factors, for example to correct errors detected during runtime, newly found controllers that offer superior performance metrics or onboard errors necessitating reconfiguration.

Each of these cases necessitates a transition between controllers that must be managed by the agent. This technique is commonly known as Plug & Play Control [18]; therefore the controllers can be described as polymorphic, as they change during operation. These polymorphic controllers adjust their rules, parameters and sensor sources to suit present conditions including failures. It is the management of this polymorphism by the agent through a formal process of understanding of physical systems, representing topological constructs and appropriate analysis and synthesis to understand dynamics that will drive the change [11]. Naturally these changes must also be reflected where they impact upon high-level goals/plans.

The Robot Operating System (ROS), provides a natural architecture that offers the potential for re-configuration, by using a flexible, graph based, structure where nodes can be activated and incorporated or removed quickly. By representing a control system within this structure then components can easily be substituted to allow a reconfigurable ROS-based control system [5].

4.3 Reconfiguration due to Agent

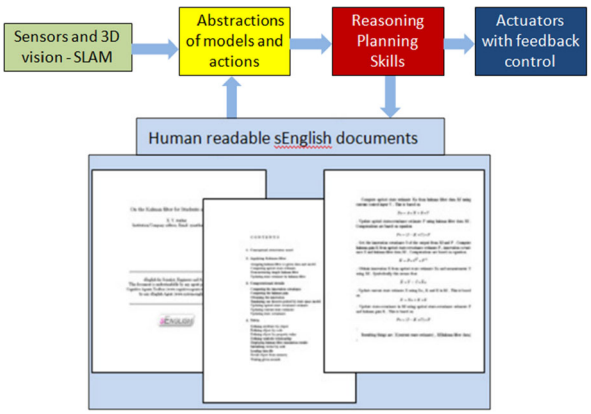
Another view of reconfigurability is where the hardware and control aspects of the system remain the same, yet the agent itself reconfigures high-level elements, such as its goals, its plans, its knowledge, and potentially its strategies. In addition, the modular nature of the architecture above allows for different (often improved) *planning*, *learning*, *coordination*, *etc*, modules to be incorporated.

Thus, our aim is to provide a generic architecture that also fits in with the above reconfigurability. Then, can we devise modular components, such as alternative planning and learning components, that will fit into this architecture (Note that we are not essentially devising new planning/learning systems, just packaging existing approaches to allow them to be seamlessly used in the generic architecture).

4.4 Agent Deciding on Reconfigurability

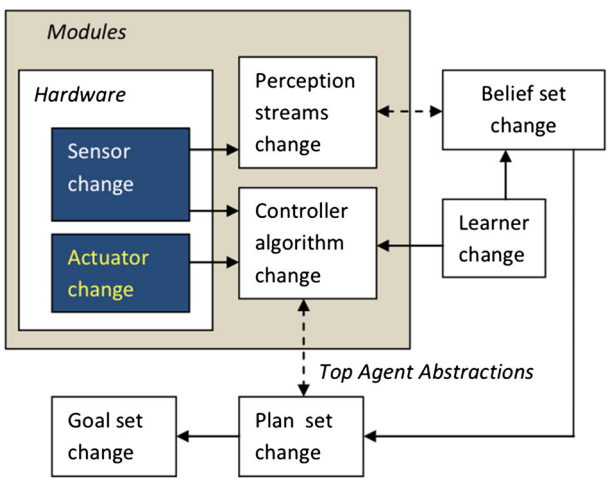
Before changing one component, be it hardware, control system, planning module, or decision strategy, the agent itself must know what it is doing. In particular what the current component offers, what the new component offers, what the new component requires, how it interacts with other components, etc. In addition, the agent must dynamically decide on the filtering/abstraction of information. One direction we are looking at is to use a high-level ontology to describe such components that can be used within complex engineering systems, for example "sEnglish" [21, 22]. This enables the designer to use natural language to specify the system, enabling then to focus on the problem rather than learning a new semantic programming language.

Using sEnglish. sEnglish has been used to describe control systems, and has a semi-formal semantics. We are developing sEnglish as a clear and coherent cross-architecture mechanism for describing components. Current research asks: *how effective is reconfiguration, based on sEnglish descriptions* and, given a reconfiguration strategy, *how can we refine/reconfigure the abstraction processes used?*



4.5 Nature of Reconfigurability Relations

Hardware, modules and high-level concepts may have dependency relationships such as illustrated in the following diagram. These dependencies can be seen in the interactions of the underlying components.



Here a continuous arrow means that an entry in the block of the start of the arrow has a *specification* that it provides to the entries in the block where the arrow ends. On the other hand the end block entries provide *capability requirements* for the entries in the starting block. However, the two-way dashed arrows mean that the higher level concepts are merely abstractions of the detailed modules. The data for hardware, module, plan, perception and learning objects are described via a high-level ontology and the classes defined are used in sentences that activate reconfiguration operations. For example, in sEnglish the sentences used are illustrated as follows:

```
Find controller C for 'steering' of vehicle
    with dynamics D using sensor set S and
    actuator set A for requirements Cr.

Find sensor set A and actuator set A for
    'steering' of vehicle with dynamics D
    and control requirement Cr.

Find learning method Lm for controller C
    with requirements Cr.

The "power is too low" is measured by
    "power level is less than ~10Q"

"Turning right" is executed by "Turn right
    with bank angle ~10 and radius ~20m."

Goal "going into orbit" can use plan "Go
    into orbit around asteroid K"
```

The meaning of each of these sentences is then described by further sEnglish sentences, in terms of ontologically defined concepts, until basic sentences have simple meanings and associated high-level code.

4.6 Real Applications

Thus, we aim to develop, refine, and assess the reconfigurability of a generic agent architecture for autonomous systems. Perhaps most importantly, we wish to adapt and apply the approach in various different application areas. This will allow us to refine the ideas/architecture, evaluate its effectiveness, and produce interesting and useful demonstrators. In this we are collaborating with real *industrial* users who form the *Autonomous and Intelligent Systems Partnership* within the UK. Working with these companies, covering such a range of different autonomous systems, and very distinct application areas, provides an ideal opportunity for testing/evaluating our approach and, if

Fig. 2 Sample of autonomous vehicles available in our labs

successful, producing flexible and reliable practical autonomous systems. Thus, a key part of our work is to collaborate with industrial partners to see how practical demonstrators can be built based on this architecture.

There are a range of possible applications, ranging across the autonomous systems field, including planetary rovers, autonomous satellites, robots exploring nuclear sites, UAVs for civilian applications, autonomous industrial processes, autonomous vehicles, etc. The industrial partners will have first hand access to leading research in these areas and will allow us to access practical autonomous systems.

4.7 University Demonstrators

While we rely on industrial collaborators for "industry strength" autonomous systems, we also have a range of demonstrators at our university labs. These will also provide initial testbeds for our techniques; see Fig. 2.

5 Related Work

To our knowledge, the scope of the problem outlined in this position paper has not received substantial attention to date. However, there exist a few examples that pose solutions for a subset of the problem. Some important topics of research that are covered by the current research endeavour (but not limited to) are *Autonomic Control Loops* [6], *Autonomic Computing* [26], *Self-CHOP*[1], and *Self-Organising Component Based Software* [25], which are all intrinsically linked. These paradigms act to modularise autonomy software, while maintaining system-wide autonomous capabilities. They provide good examples and a starting point for this project.

IBM proposed the Monitor, Analyse, Plan, Execute, Knowledge (MAPE-K), loop to provide an architecture for introducing autonomic computing [6]. This is computing that mimics the ability of the human body to regulate itself. A system is broken down in a similar way to a traditional control system, where the software itself, the aspect managed, forms the plant. Sensors record behaviour within the environment, for example network or memory usage, and effectors make adjustment to the environment to meet given goals, for example adding servers to increase capacity. Sensors feed information into the MAPE-K loop. Effectors act on decisions taken by an autonomic manager which contains the MAPE-K loop. The MAPE-K loop holds knowledge about the operation of the system, encoded as rules that it can use to alter behaviour. Monitoring processes, watch the state of the world using sensor information, which is *analysed* in conjunction with *knowledge* of the system before new *plans* can be made to be pushed back to effectors by *executing* code.

For example the ABLE toolkit [7] has been implemented as a multi-agent system in Java to monitor web servers. Valetto and Kaiser [19] have worked on a system of retrofitting autonomic computing onto legacy hardware [20]. This work focuses on where data should come from to allow the adaptation process to happen, specifically it focuses on sensors and execution of plans. One area this work does not consider is how adaptation planning will be conducted.

[1] Self-CHOP = The four self-properties: Self-Configuration, Self-Healing, Self-Optimisation and Self-Protection

In July 2004, CNES and ONERA started upon a common research program on autonomy for space systems. The product was Autonomy Generic Architecture, Tests and Application (AGATA) [3, 4]. AGATA is a autonomous architecture focused on the issues of maintaining a high level of autonomy while attempting to incorporate genericity, modularity, and autonomicity (in particular self-organisation). The modules are built on the basis of a common pattern, and connected together to form an architecture. Each module controls a part of the system, and is built upon a sense/decide/act pattern. Modules maintain their own knowledge and controls based upon an internal UML description. The architecture is overseen by the generic control module, which acts to connect and configure modules to complete the architecture overall goal, based upon the UML descriptions and the communication request of the modules. AGATA also provides methods for validation and verification, and fault detection, isolation and recovery for a generic modular autonomous architecture. AGATA demonstrates on-line reactive and deliberative reconfiguration method for autonomous software, which can deal with complex decentralised architectures. However, there exists no method to deal with hardware; the internal description of modules lacks expressiveness; and the system has a limited ability to diagnose compound faults that arise from modules interacting.

Another architecture (herein known as RDA) that aims for self-reconfiguration of autonomous modular software is proposed in [9]. The modules in RDA are described by UML components diagrams and Datalog, similar to the method of AGATA. However, unlike AGATA the modules can have a very generic definition and all their action and interconnection are determined by a centralised controller. The controller acts in a monitor, analyse, plan, and reconfigure loop, and thus is broken into three sub-controllers: the "Monitor" which is in charge of collecting, filtering, and normalising events and logs; the "Diagnoser" which identifies failures and discovers root causes; and the "Reconfigurator" which selects, plans and deploys "compensation actions" in response to failures. The reconfiguration is then achieved using requirement engineering techniques. However, this system lacks the ability to deal with hardware, is limited by its ability to describe and monitor modules, and is limited by its monolithic and deliberative nature.

6 Summary

While the need for autonomous systems that can act intelligently without direct human intervention is increasing, traditional adaptive control regimes are often *ad-hoc* and *opaque* to deep analysis. Furthermore they do not take into account the vital new aspect of truly autonomous systems, namely having *explicit* high-level justifications for the autonomous choices made.

In practical autonomous systems, such as autonomous vehicles, the idea of having an agent-based hybrid architecture is gaining traction. This separates out the high-level autonomous decision-making from continuous control components, not only making the decision strategy verifiable [2], but allowing a modular approach to the architecture. Once components can be added or removed, then the high-level rational agent must be able to reason about the requirements and consequences of such changes; this provides high-level description and analysis of *reconfigurability*. As well as reasoning about the decisions taken behind any reconfiguration it should be able to effectively communicate these decisions back to the system operators. This closes the loop around the design process but provides an important view to remove any opacity within the reconfiguration process.

In our ongoing research, described in this position paper, we aim to provide an *open-source rational agent architecture* that controls autonomous decision-making, is re-usable and generic (and so can be configured for many different autonomous platforms); whose agent core is potentially verifiable; and that is dynamically *reconfigurable*—not only mission goals, but also capabilities, control sub-systems, or hardware can be removed/added at run time.

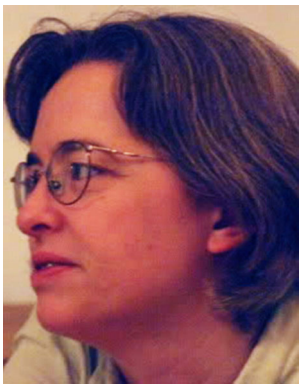
Project web page: http://www.csc.liv.ac.uk/RAIS

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

1. Dennis LA, Fisher M, Lisitsa A, Lincoln N, Veres SM (2010) Satellite control using rational agent programming. IEEE Intell Syst 25(3):92–97
2. Fisher M, Dennis LA, Webster M (2013) Verifying autonomous systems. ACM Commun 56(9):84–93
3. Verfaillie G, Charmeau M (2006) A generic modular architecture for the control of an autonomous spacecraft. In: Proceedings of the 5th international workshop on planning and scheduling for space (IWPSS), 2006
4. Charmeau M, Pouly J, Bensena E, Lemaitre M (2008) Testing spacecraft autonomy with AGATA. In: Proceedings of the 9th international symposium on artificial intelligence, robotics and automation in space (ISAIRAS), 2008
5. Aitken JM, Veres SM, Judge M (2014) Adaptation of system configuration under the robot operating system. In: Proceedings of the 19th world congress of the international federation of automatic control (IFAC), 2014
6. IBM (2003) An architectural report for autonomic computing. IBM Technical Report

7. Bigus JP, Schlosnagle DA, Pilgrim JR, Mills WN III, Diao Y (2002) ABLE: a toolkit for building multiagent autonomic systems. IBM Syst J 41(3):350–371
8. Muir B (1987) Trust between humans and machines, and the design of decision aids. Int J Man Mach Stud 27(5–6):527–539
9. Dalpaiz F, Giogini P, Myopoulos J (2009) An architecture for requirements-driven self-reconfiguration. In: Proceedings of the advanced information, systems engineering, 2009
10. Lomuscio A, Ezekiel J, Molnar L, Veres S (2011) Verifying fault tolerance and self-diagnosability of an autonomous underwater vehicle. In: Proceedings of the international conference on artificial intelligence (IJCAI), pp 1659–1664, 2011
11. Ippolito C, Joo S, Al-Ali K, Yeh YH (2008) Polymorphic control reconfiguration in an autonomous UAV with UGV collaboration. In: Proceedings of the IEEE aerospace conference, pp 1–14, 2008
12. Kandiyil R, Gao Y (2012) A generic domain configurable planner using HTN for autonomous multi-agent space system. In: Proceedings of the international symposium on artificial intelligence, robotics and automation in space (i-Sairas), 2012
13. Delfa Victora J, Fratini S, Policella N, Gao Y, Von Stryk O (2013) QuijoteExpress—a novel APSI planning system for future space robotic missions. In: Proceedings of the ESA workshop on advanced space technologies for robotics and automation (ASTRA), 2013
14. Delfa Victora J, Policella N, Gao Y, Von Stryk O (2012) Design concepts for a new temporal planning paradigm. In: Proceedings of the international conference on automated planning and scheduling (ICAPS), Workshop on planning and scheduling with timelines, 2012
15. Konur S, Fisher M, Dobson S, Knox S (2014) Formal verification of a pervasive messaging system. Form Asp Comput 26(4):677–694
16. Lincoln N, Veres SM, Dennis LA, Fisher M, Lisitsa A (2010) An agent based framework for adaptive control and decision making of autonomous vehicles. In: Proceedings of the IFAC workshop on adaptation and learning in control and signal processing (ALCOSP), 2010
17. Lincoln N, Veres SM, Dennis LA, Fisher M, Lisitsa A (2013) Autonomous asteroid exploration by rational agents. IEEE Comput Intell 8(4):25–38
18. Stoustrup J (May 2009) Plug & play control: control technology towards new challenges. Eur J Control 15(3–4):311–330
19. Valetto G, Kaiser G (2003) Using process technology to control and coordinate software adaptation. In: Proceedings of the 25th international conference on software engineering (ICSE), 2003
20. Parekh J, Kaiser G, Gross P, Valetto G (2006) Retrofitting autonomic capabilities onto legacy systems. J Clust Comput 9(2):141–159
21. Veres SM (2008) Natural language programming of agents and robotic devices. SysBrain, 2008
22. Veres SM, Molnar L (2010) Documents for intelligent agents in english. In: Proceedings of IASTED conference on artificial intelligence applications, Innsbruck, Austria. IASTED, 2010
23. Webster M, Fisher M, Cameron N, Jump M (2014) Generating certification evidence for autonomous unmanned aircraft using model checking and simulation. J Aerosp Inf Syst 11(5):258–279
24. Wooldridge M (2002) An introduction to multiagent systems. John Wiley & Sons, NY
25. Sterritt R, Hinchey M (2010) SPAACE IV: self-properties for an autonomous & autonomic computing environment—part IV a newish hope. In: Proceedings of the 7th IEEE international conference and workshops on engineering of autonomic and autonomous systems (EASe), 2010
26. Murch R (2004) Autonomic computing. IBM Press, Englewood Cliffs



Louise A. Dennis is a Postdoctoral Research Fellow in the Department of Computer Science at Liverpool. She is an expert in agent programming languages, formal verification of computational systems and automated reasoning.

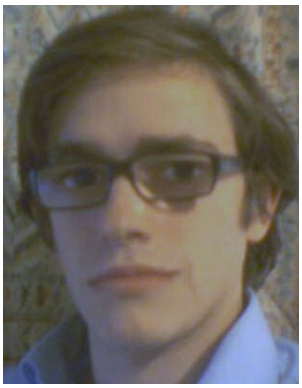
Michael Fisher is Director of the *Centre for Autonomous Systems Technology* and a Professor of Computer Science at the University of Liverpool. He is internationally recognized for research across formal logics, automated verification, and autonomous systems.

Jonathan M. Aitken is a Postdoctoral Research Fellow in Automatic Control and Systems Engineering at Sheffield. He is recognized for his work on autonomous network reconfiguration, analysis of distributed systems and system identification for control.

Sandor M. Veres is a Professor and Head of Sheffield's *Autonomous and Robotic Systems Research Group*. He is internationally recognized for research across mobile robotics including formation flight, agent supervised feedback control systems, architectures and programming of reasoning based intelligent agents.

Yang Gao is a Professor of Space Autonomous Systems and Head of the Surrey Space Centre's *STAR Lab*. She specializes in computational intelligence, robotic vision and biomimetics with applications to space systems and robots alike.

Affan Shaukat is a Postdoctoral Research Fellow in the Surrey Space Centre's *STAR Lab*. He specializes in robotic vision and machine learning.



Guy Burroughes is a PhD student in the Surrey Space Centre's *STAR Lab*. His PhD research focuses on reconfigurable software architecture design, validation and verification.