Defence Science Journal, Vol. 60, No. 1, January 2010, pp. 15-22

REVIEW PAPER

Robotic Architectures

Mbali Mtshali[*] and Andries Engelbrecht[**]

[*]*Council for Scientific and Industrial Research, Pretoria, South Africa*
E-mail: mmtshali@csir.co.za
[**]*University of Pretoria, Pretoria, South Africa*
E-mail: engel@cs.up.ac.za

ABSTRACT

In the development of mobile robotic systems, a robotic architecture plays a crucial role in interconnecting all the sub-systems and controlling the system. The design of robotic architectures for mobile autonomous robots is a challenging and complex task. With a number of existing architectures and tools to choose from, a review of the existing robotic architecture is essential. This paper surveys the different paradigms in robotic architectures. A classification of the existing robotic architectures and comparison of different proposals attributes and properties have been carried out. The paper also provides a view on the current state of designing robot architectures. It also proposes a conceptual model of a generalised robotic architecture for mobile autonomous robots.

Keywords: Mobile robotic systems, autonomous robots, robotic architectures, robotics, mobile robots

1. INTRODUCTION

Though the research in robotics has been in existence for a number of years yet there are still many challenges in the field[1]. This is because robotic systems are complex to build and difficult to maintain. Moreover, as developers move away from stationary robots, the area of mobile robots has received much attention in the research community. The term mobile robot describes a robotic system able to carry out tasks in different places and consisting of a platform moved by locomotive elements[2]. When mobile robot systems operate in the real-world environment without any form of external control for expected periods of time, these are described as being autonomous. Mobile autonomous robots thus have mobile and autonomous capabilities to carry out required tasks with minimal user control. These robots may consist of a number of sensors, actuators, global positioning system (GPS), and processing units.

The underlying framework which interconnects the entire robotic system (hardware and software) is referred to as a robotic architecture. The robotic architecture is defined as the discipline devoted to the design of highly specific and individual robots from a collection of common software building blocks[3]. Architecture for a robotic system can also define how sensors, actuators, and the physical platform interact with each other. Due to non availability of uniform standards in robotics, there is no commonly accepted robot architecture. This has led to researchers being forced to develop their own robotic architectures. Researchers develop architectures for their own robot's specific need which may not be applicable to other robotic systems. This is a waste of time and causes re-invention of the wheel[2]. Another challenge can be the complexity and insufficient information on the design and description of the architecture which causes difficulties to robot developers when adopting that particular architecture.

2. MOBILE ROBOTIC ARCHITECTURE

Mobile autonomous robots can be defined as complex systems designed to be intelligent and mobile with some level of autonomy. Mobile autonomous robots are required to perform different tasks concurrently and asynchronously, thus, adding immensely to system complexity. This leads to an appropriate structure of how all sub-systems interact. This structure in robotics is referred to as a robotic architecture.

A robotic architecture is the discipline devoted to the design of highly specific and individual robots from a collection of common software building blocks[3]. The architecture of a robotic system can also define how sensors, actuators, and the physical platform interact with each other. The architecture thus represents an abstraction of a system and should be described by the sub-components that support the system's design and description of the architecture which causes difficulties to robot developers when adopting that particular architecture.

3. MOBILE ROBOTIC ARCHITECTURE DESIGN PROPERTIES

The design used in development of a MRA depends on the robotic system design requirements. One needs to know what properties an architecture for autonomous mobile

Received 22 September 2009

robots should have. Some design properties are[3]:

- *Reliability:* is concerned with a system's ability to check if the architecture can achieve its designated tasks (possibly with some performance reduction in time, cost, or accuracy) even when performance is degraded. Reliability can be evaluated by manipulating function (e.g. sensor/actuator failure) and measuring the effect on task achievement.
- *Generalisation and adaptability:* refer to the ability of a system to act appropriately (not necessarily optimally) in situations that it has not encountered before. Generalization in machine learning is assessed by separating training, testing and demonstration performance in new situations. Generalization and adaptability allow a system to refine the current task and its behavior according to the current goal and execution of the task.
- *Modularity***:** refers to the generic design of a system to consist of easily interchangeable modules/components. Components of a modular system can be replaceable.
- *Autonomy:* refers to a system's ability to carry out its own actions independently with least, minimal, or no user involvement.
- *Robustness***:** is concerned with the system's ability to handle imperfect inputs and unexpected events such as sensor failures.
- *Extensibility***:** is concerned with the system's ability to expand a system and the level of effort required to implement the expansion. Extensions can be through the addition of new functionality or modification of existing functionality.
- *Reactivity:* is concerned with the automatic response of a robotic system to situations.
- *Run time flexibility:* refers to the system's ability to be adjusted or reconfigured during execution and how easily adaptation and learning can be introduced.

The above design properties provide guidelines in evaluating the performance of the system. Generally, the design depends on what the requirements are. In some situations, surgical robots for example, generalisation may be more important than extensibility.

4. CLASSIFICATIONS OF ROBOTIC ARCHITECTURE

In classification of robotic architectures, primitives used involve the more crucial actions performed by a robot to achieve tasks. In general, robots sense the environments using sensors. Then they plan the next step and act by following the planned action. Thus, the accepted robotic architecture primitives are sense, plan, and act. These are associations of how sensory data is processed and propagated through the system. Sense refers to the sensory or perceptual system that a robot uses to perceive the world around it. Plan refers to a planner and usually a complex planner that uses some sophisticated problem-solving technique. Act refers to actuators with which the robot can act upon its environment. This approach is referred to as a sense-plan-act (SPA) approach[4]. Robotic architectures are classified based on their SPA approach. The three main classifications, namely a function-based architecture (deliberative), a behaviour-based architecture (reactive), and a hybrid architecture.

4.1 Function-based Architecture

The function-based architecture (Fig. 1), is one of the first approaches which was the dominant paradigm in the early days of artificial intelligence (AI) robotics where the focus was on robot planning and higher-level reasoning[2]. These are also referred to as deliberative architectures. Function-based architectures follow a top-down SPA approach. The emphasis of function-based architectures is on constructing a detailed environmental model and then to carefully plan the functions to be carried out. The sensing module is used to translate sensor data into an internal world model. Based on this internal model, and given a goal, the planner generates a series of actions to be undertaken. The act module then takes the plan and sends actions to the robot actuators. The process is iterative and there is a feedback mechanism from act to sense.

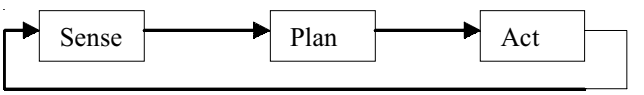

Figure 1. Function-based architecture.

The benefit of the function-based architecture is its ability to utilise past experience and knowledge to accomplish a task. To accomplish a task, a robot needs to form a representation of the world. It uses a process of inference to derive the new representations about the world, and use these new representations to deduce what to do[5].

A major challenge with this architecture is that, in dynamic environments, the robot can't maintain coupling (interaction) with its environment. This is due to its world model representation which is not instantaneously maintained. Consequently, the robot often loses the correlation to the environment and needs to carefully observe the environment again to find the correlation and re-plan. In addition, because of intensive planning, response is slow and latency becomes variable, thus making this approach unsuitable for dynamic environments.

4.2 Behaviour-based Architecture

A shift from the traditional function-based approach is the behaviour-based (reactive) systems. Inputs to actuators in a behaviour-based architecture, as depicted in Fig. 2, are direct outputs from a sensor and there is no planning involved. behaviour-based systems exhibit various behaviours, some of which are emergent. These systems are characterised by direct coupling between sensors and actuators, and minimal computation. In this type of architecture, the robot components include behaviour modules, and the feedback control mechanism for behaviours closely connects the robot to the real-world.

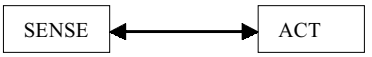

Figure 2. Behavior-based architecture.

Behaviour-based architectures are more suitable to robots that perform in dynamic environments as these react well to changes in the environment[6]. However, the main challenge of the behaviour-based architecture is that it tends to lose its appeal as environmental task complexity increases. The more the world changes during execution, the more the resulting value of any previously generated plans decreases. This result in any representational knowledge stored ahead of time or gathered during execution becoming more unstable[1]. There is also a challenge of selecting the proper behaviours for robustness and efficiency in accomplishing goals. Again, as complexity within the environment increases, so does the number of Behaviours that a robot may exhibit. This makes the prediction of ultimate behaviour difficult. Another challenge is the low level of intelligence as there is no planning.

Examples of behaviour-based architectures include the well-known subsumption architecture by Brooks[7], behavioural architecture for robot tasks (BART), the distributed architecture for mobile navigations (DAMN)[9], and a hierarchical architecture for behaviour-based robots[10].

There is always a question of how the system should arbitrate/coordinate/cooperate its behaviours and actions? Behaviour-based architectures uses a coordination mechanism based on domain-specific metrics to select the appropriate behaviour. Different behaviours are coordinated to have one behaviour/command sent to the actuators. Coordination mechanisms avoid conflicts between two or more active behaviours. The two types of coordination methods used are competitive and cooperative coordination[3]. In competitive coordination (Fig. 3), behaviours compete and only one behaviour is selected and activated.

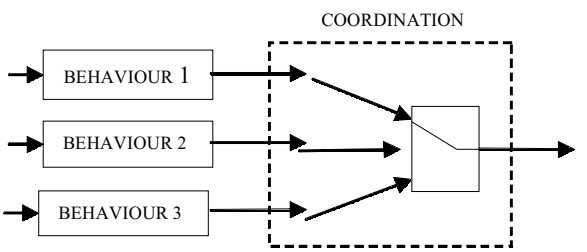

Figure 3. Competitive coordination.

Behaviour arbitration requires that a coordination function serving as an arbiter selects a single behavioural response[1]. Behaviours can be selected based on priority with the use of suppression and inhibition. Suppression overrides the normal input signal from being transmitted. Inhibition prevents a signal from being transmitted along a behavioural module's wire from reaching the actuators. Brook's subsumption architecture[2] adopts this approach. Another method of competitive-based approach is using action-selection coordination. For this more democratic method, behaviours generate votes for actions, and the action with most votes is chosen. The DAMN architecture[9], amongst others, adopts this method. The advantages of using competitive coordination include modularity, robustness, and tuning time. Demerits include performance, development time, and complexity; these problems are caused by the mechanisms used to select the optimal behaviour amongst the various competing behaviours.

Cooperative coordination (Fig. 4) differs from competitive coordination in a sense that the different behaviours are all considered and fused together. This type of coordination method provides the ability to concurrently use the output of more than one behaviour at a time. The main issue is the combination mechanism for all behaviour output to one fused behaviour. The resultant behaviour from the vector summation is then normalised. Advantages of using such approach include increased performance, development time, and simplicity. However, cooperative coordination is disadvantageous due to no modularity, no robustness, and increased tuning time.

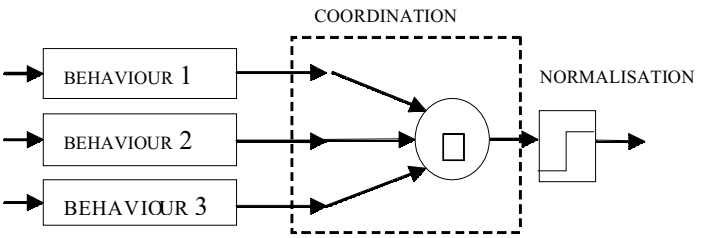

Figure 4. Cooperative coordination.

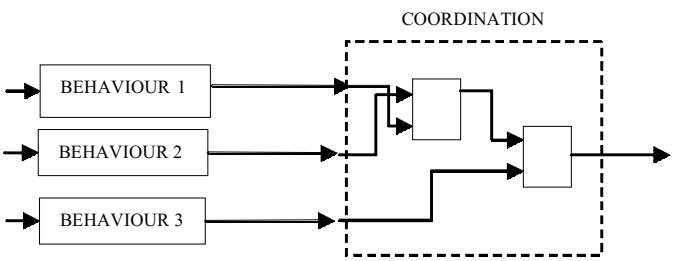

Figure 5. Hybrid coordination.

Hybrid coordination then proposed a combination of competitive and cooperative coordinations to overcome their demerits[33]. Hybrid coordinations is illustrated in Fig. 5. As shown in Fig. 5, two behaviours are initially fused together. The third behaviour is the behaviour that results by fusing behaviours 1 and 2.

Adopting the hybrid-based coordination approach may increase the system's latency due to multi-processing of different behaviours.

4.3 Hybrid Architectures

Hybrid architectures (Fig. 6) were developed to address the problems of the function-based and behaviour-based

architectures and to blend their qualities, such as retaining of planning in some way as well as direct coupling. Hybrid architectures have become the leading architectural paradigm as these combine the qualities of both architectures. A hybrid-based architecture enables the architecture of autonomous robots to have both decision-making and reactive capabilities.

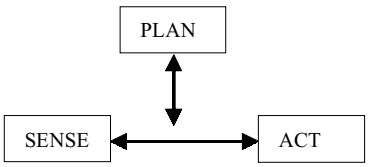

Figure 6. Hybrid architecture paradigm.

Hybrid architectures are more suitable for complex environments, which can be both static and dynamic for a while. There are many variations of architectures in this category, depending on the design details, and how these are combined[1].

Research work on hybrid architectures includes ATLANTIS[6], layering using data persistence[10] (LUDAP), servo-subsumption-symbolic (SSS) architecture[11], task control architecture[12] (TCA), autonomous robot architecture[13] (AuRa), DD&P[14], 3-Tiered[15], RHINO[16], SOMASS[17], CIRCA[18], and SAPHIRA [19]. Most of these architectures have proven to be successful but they are still not robust enough for a range of tasks. As the hybrid architectures also inherit the weaknesses (such as slow response) of both functional and behavioural architectures, a difficult task is to find a reasonable balance between the function-based and behaviour-based architectures.

An interesting approach closely related to hybrid architecture, presented by Lagland[19], *et al.* is the integration of two distinct architectures to produce a dual-architecture rather than developing a single hybrid architecture that has qualities of the function-based and behaviour-based architectures.

One of the most prominent works in hybrid architecture is the development of the coupled layer architecture for robotic autonomy[5] (CLARAty) which was designed by NASA for space robots in 1995. Research on mobile robots was expanded to support Mars exploration rovers and this led to an interest to develop a reusable robotic framework. This then led to CLARAty development by collaborative effort among JPL, Carnegie Mellon, NASA Ames Research Centre, and joined later by the University of Minnesota. CLARAty is a two-layered architecture with decision and functional layers. The functional layer is an interface to all system hardware and the hardware capabilities, including nested logical groupings. The decision layer breaks down high level goals into smaller objectives. The objectives are arranged in known constraints and system states for the appropriate capabilities of the functional layer to access these. CLARAty has successfully contributed to the field of MRAs. However, here are still many challenges in developing reusable software. These challenges and the steps towards reusable robotic software are discussed by Nesnas[20], *et al.* The software is available for download[21].

Another prominent robot architecture is 4-D/RCS,[22] which is a reference model architecture for intelligent unmanned ground vehicles. The 4-D/RCS architecture is derived from the real-time control system (RCS) architecture and the German VaMoRs 4-D approach to dynamic machine vision. The 4-D/RCS architecture provides a theoretical foundation for designing, engineering, integration, and testing intelligent systems software for unmanned vehicle systems. The 4-D/RCS architecture is hierarchical and is composed of a common node structure at each level.

A comparison and summary of the key features of function-based, behaviour-based, and hybrid architectures are given in Table 1.

5. LAYERED APPROACH TO MOBILE ROBOTIC ARCHITECTURES

To improve the efficiency of the design of robotic architectures for robots to perform complex tasks in dynamic environments, tasks should be assigned to separate layers[4]. The three-layer architecture approach has become a commonality in robot control architectures[23]. However, there is no fixed limit to the number of layers. The layers differ between architectures as well as the mechanisms for communicating state information and coordinating activity. The top layer adopts a more goal oriented view and plans over a longer scope using information received from the sensory data. The upper layer is deliberative whilst the middle layer is reactive. The lower layer provides fast, short horizon decisions to facilitate quickly executed actions based on sensory data. The lower layer controls the robot architecture components, namely the sensors and actuators.

Table 1. Comparisons and summary of function-based, behaviour-based and hybrid-architectures

Function-based	Behaviour based	Hybrid
- deliberative	- reactive	- deliberative and reactive
- intense computational requirements	- minimal computational requirements	- intense computational requirements
- slow response	- quick response	- slow response
- world representation	- no world representation	- world representation
- suitable for static environments	- suitable for dynamic environments	- suitable for complex environments
- high level of intelligence	- low level of intelligence	- high level of intelligence

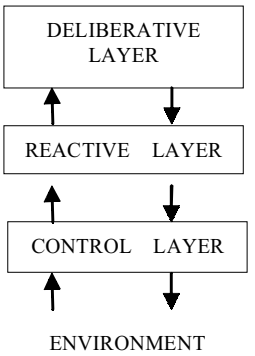

Figure 7. Three-layered architecture.

The hybrid (sense-plan-act) architecture adopts thethree layer approach. For example, the Aura architecture[13] has the following layers: a hierarchical component (deliberative), reactive component and execution (control) layers. The hierarchical component includes a mission planner, spatial reasoner, and plan sequencer. The hierarchical component is concerned with establishing high level goals for the robot and specifying the constraints within which it must operate. The reactive component specifies and instantiates behaviours which are sent for execution. Other architectures which adopt the three-layered architecture include 3T[24] .

6. CENTRALISED VS DISTRIBUTED ARCHITECTURES

In a robotic system, it is also crucial to choose between a centralised approach or a distributed approach. A centralised system consists of one sub-system which has overall responsibility for control. The communication between the sub-systems is via the controller sub-system. The centralised approach has the ability to coordinate multiple goals and constraints in complex environments. However, if a controller sub-system fails, the whole system fails. Thus, a purely centralised architecture is clearly not appropriate for a real-time system where the environment is dynamic or uncertain. This then resulted in the design of distributed architectures.

A distributed system consists of multiple sub-systems that control and communicate with each other by message passing. Distributed architectures offer reactivity to dynamic environments. Flexibility and robustness are also increased. However, distributed architectures cannot perform very well in complex environments. Moreover, the interaction between the system and its environment becomes less predictable and more difficult to understand and modify. Communication amongst modules also becomes a challenge in distributed systems. The designer needs to choose the desirable communication mechanism. The subsumption architecture[18] has direct communication between modules and this provides the system designer with a high degree of control over the operation of the system, which may be desirable when modules are engineered to interact.

A comparison between the centralised and distributed approaches is shown in Table 2.

Table 2. Centralised vs Distributed architecture

Centralised	Distributed
Advantages: - coordination of multiple goals and constraints in complex environments	*Advantages*: - reactivity, flexibility and robustness
Disadvantages: - not appropriate for dynamic environments with uncertainty - can be subject to terrible failure if the controller subsystem fails	*Disadvantages:* - not appropriate for complex environments

7. PROGRAMMING AND DEVELOPMENT TOOLS

Development of mobile autonomous robots is still an adhoc activity and suffers from the lack of standard paradigms. There is still a challenge[25] of obtaining models that encompass full autonomy in robot system design. Many successful development tools are mostly based on trial and error due to a number of issues with the development of robots. These challenges may include the complex tasks a robot has to perform. Environmental changes (static or dynamic environments) also contribute to challenges in robotic system development.

The development of a system begins with gathering system requirements and performing system design. For system design and documentation of the system's requirements the unified modelling language (UML)[32] is mostly used because of its widespread tool support and is an industry standard for specifying software systems. Other modelling languages used in robotic systems include EXPRESS (data-modelling language)[36] and behaviour trees (formal graphical modelling language)[37].

Communication amongst the robot system components is also crucial. The software system that implements communication amongst system components is referred to as middleware. There are two basic approaches to system's communication: client-server and publish-subscribe. In client-server communication, components interact directly with other components. There are several existing client-server-based middleware solutions, for example ORCA[38] and Go[39]. Protocols based on the client-server approach are Corba[40] and remote procedure call protocol (RPC)[41]. The client-server approach allows distributed data communication with no central module to distribute data. A disadvantage to distributed data communication is the high overhead introduced, especially if many components need the same information.

The publish-subscribe mode of communication publishes data for components to subscribe to. Only the subscribers receive published data to reduce distribution of data to the entire system. Examples of middleware based on publish-subscribe include Player and GenoM. The popular protocols of publish-subscribe are the inter-process communication (IPC) and data distribution service (DDS). The publish-

subscribe mechanism is very simple to use and has low overhead. However, publish-subscribe often uses a single central server, proving a single point of failure.

The choice of selecting the best tools and mechanisms to use depends on the system requirements and the robotic system design specifications. These choices can be the most important and constraining of many decisions a robot architecture designer has to make. In many cases, most debugging time in robot architecture development have to do with communication between components. A useful comparative review of robot programming languages has been done by Pembeci and Hager[27].

8. A GENERALISED ARCHITECTURE FOR MOBILE AUTONOMOUS ROBOTS

The proposed generalised architecture for mobile autonomous robots is illustrated in Fig. 8. This is only a conceptual model which has not yet been tested and analysed but a proposal for mobile autonomous robots. The model is a three layered, hybrid-based architecture supported by the design properties discussed in Section 3.

The proposed architecture has both deliberative and behavioural layers. The deliberative layer will ensure good planning mechanisms and high level reason are carried out by the robotic system. The control layer represents the hardware units that controls the robot's sensing and actuation. The behavioural layer will ensure that the robot reacts quickly to unknown environments or situations by creating behaviours. The architecture includes the following major components- a world model, planner, learning component, and the command generator. World model defines the external environment with which the robot interacts. World model can represent dynamic and static environments. In a static environment, the environment does not change whilst the dynamic world model represents an environment with the ability of changing, for example, moving obstacles.

The planner is the processor unit which interacts with the world model and decides whether any planning is necessary, otherwise it reactively responds by creating behaviours. If planning is required, the planner derives a plan in terms of sequential tasks the system has to undertake to achieve a goal.

Behaviours are coordinated using a hybrid coordinator. The hybrid coordinator uses both competitive and cooperative mechanisms to combine the different behaviours. The command generator creates commands for the robot's actuator to perform. The learning component introduces new knowledge (facts, behaviours, rules, etc.) into the system. Several learning mechanisms that can be used for learning include reinforcement learning, learning by imitation, and neural networks.

The proposed architecture is to adopt a distributed approach which offers reactivity, flexibility, and robustness. The distributed approach is suitable for dynamic environments with uncertainty which is-what mobile autonomous robots are subjected to a client-server-based communication mechanism will be applied.

Overall, the proposed model promises to contribute significantly in offering a reliable, modular, adaptable, and modular robotic architecture. A more detailed architecture

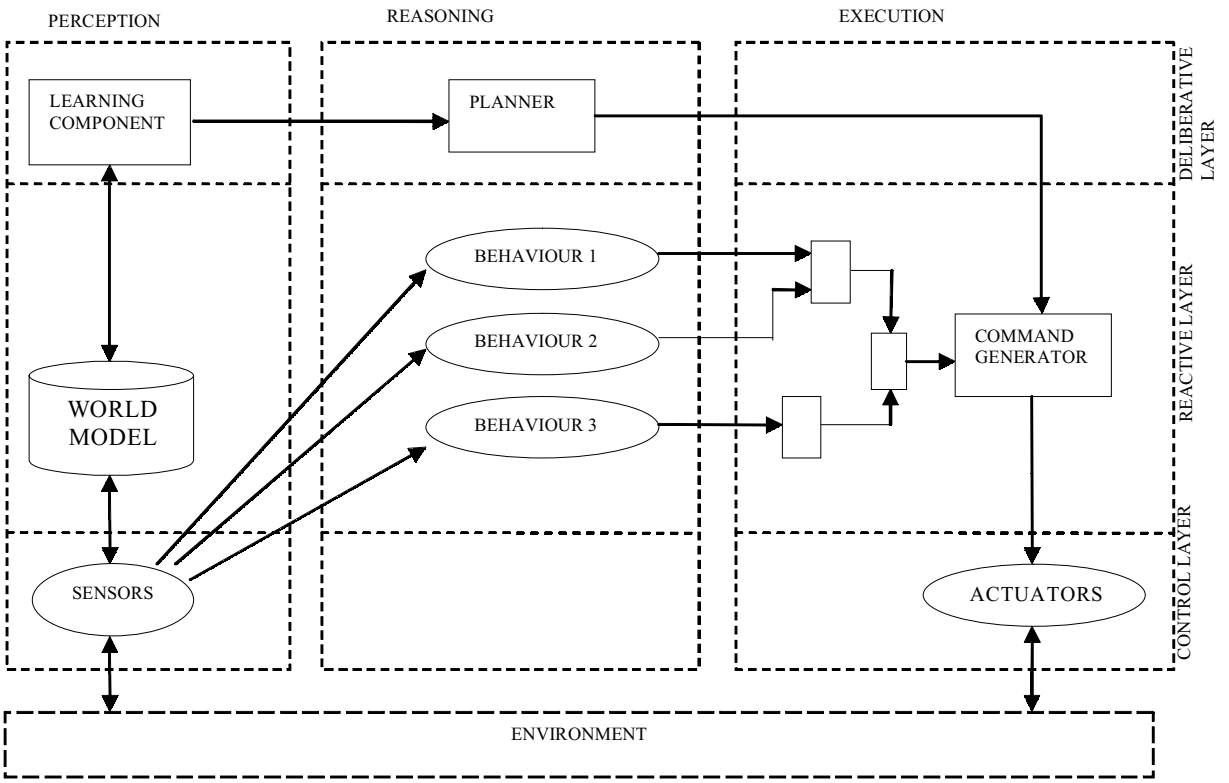

Figure 8. Conceptual hybrid robotic architecture.

functionality will be published in future. The proposed architecture aims at addressing development challenges by offering a robustness to improve fault-tolerant capabilities. Implementing learning should also be possible.

9. CONCLUSIONS

Research in robotics is shifting away from just static robotic systems. The new focus is now on designing mobile the system has to undertake to achieve a goal. The learning component introduces new knowledge (facts, behaviours, rules, etc.) into the system. Several learning mechanisms include reinforcement learning, learning by imitation, and neural networks.

Autonomous robot systems will function with minimal user control. An important framework for integrating the software and hardware components of a robot system is a robotic architecture. Designing and developing a robotic architecture is necessary in the development of a robotic system. However, as no standard reference architecture exists in literature, it becomes complex and difficult to develop one from scratch or to use existing architectures available in literature. A comparison and classification of mobile robotic architectures is thus useful in assisting the architecture developer in looking at different robotic architecture design properties.

REFERENCES

1. Murphy, R. Introduction to AI robotics. 2000.
2. Garcia, E.; Jimenez, M.; De Santos P. & Armada, M. The evolution of robotics research. *IEEE Robotics Automation Mag.*, 2007.
3. Arkin, R. Behaviour-based robotics. MIT Press, 1998.
4. Gat, E.; Bonnasso, R.; Murphy, R. & Press, A. On three-layered architectures. *Artifi. Intelli. Mobile Robots,* 1997.
5. Volpe, R.; Nesnas, I.; Estlin, T.; Muts, D.; Petras, R. & Das, H. The CLARAty architecture for robotic autonomy. *In* IEEE Proceedings of Aerospace Conference, 2001.
6. Gat, E. Integrating planning and reacting in a heterogenous asynchronous architecture for controlling real-world mobile robots. *In* 10th National Conference on Artificial Intelligence (AAAI). 1992.
7. Brooks, R. A robust layered control system for a mobile robot. *IEEE J. Robotics Automation,* 1986, **1**(RA-2).
8. Kahn, P. Specification and control of Behavioural robot programs. *In* Proceedings of the AIAA Conference on Computer in Aerospace VI, Wakefield, MA, 1991.
9. Rosenbalt, J. DAMN: A distributed architecture for mobile navigation. *J. Experi. Theo. Artifi. Intelli.*, 1997, **9**(2-3). *In* Symposium on Lessons Learned for Implemented Software Architecture for Physical Agents, CA , 1995.
10. Nicolescu, M. & Mataric, M. A hierachical architecture for behavaior-based robots. *In* Proceedings First International Joint Conference on Autonomous Agents and Multi-Agenet Systems, 2002.
11. Sheshagiri, M. & Marie, desJardins. Data persistence: A design principle for hybrid robot control architectures. *In* International Conference on Knowledge Based Computer Systems, Mumbai, India, 2002.
12. Connell, J. SSS: A hybrid architecture applied to robot navigation. *In* Proceedings of the IEEE Conference on Robotics and Automation (ICRA-92), 1992. pp. 271-2724.
13. Simmons, R. Structured control for autonomous robots. *IEEE Trans. Robotics Automation*, 1994.
14. Arkin R & Balch T, AuRA: Principles and Practise in Review. *J. Experi. Theo. Artifi. Intelli.*, 1997, **9**(2-3), 175–89.
15. Joachim, H. & Frank, S. DD&P: Concurrency in the DD&P robot control architecture. *In* Proceedings of International NAISO Congress on Information Science Innovations, 2001. pp. 1079-085.
16. Bonasso, P.; Firby, R.; Gat, E.; Kortenkamp, D.; Miller, D.; & Slack, M. 3-tiered, experiences with an architecture for intelligent, reactive agents. *J. Experi. Theo. Artifi. Intelli.*, 1997, **9**(1-2).
17. Buhmann, J.; Burgard, W. A.B.; Cremers, A.; Fox, D.; Hofmann, T.; Schneider, F.; Strikos, J. & Thrun, S. RHINO: The mobile robot RHINO. *AI Magazine*, 1995, **16**(2), 31–38.
18. Malcolm, C. SOMASS: The SOMASS system: A hybrid symbolic and behaviour-based system to plan and execute assemblies by robot. *In* Proceedings of AISB Conference, Sheffield, April 1995.
19. Musliner D.; Durfee E. & Shin K. CIRCA: World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence*, 1995.
20. Konolige, K.; Myers, K. & Ruspini, E. The SAPHIRA architecture: A design for autonomy. *J. Experi. Theo. Artifi. Intelli.*, 1997.
21. Langland, B.; Jansky, O.; Byrd, J. & Pettus, R. Integration of dissimilar control architectures for mobile robot applications. *J. Robotic Syst.,* 1997, **14**(4).
22. Nesnas, I. & *et al.* CLARAty: Challenges and steps toward reusable robotic software. *Int. J. Adv. Robotic Syst.,* **3**, 2006.
23. http://claraty.jpl.nasa.gov/man/overview/index.php
24. Albus, J. 4D/RCS reference model architecture for unmanned ground vehicles. *In* Proceedings of the SPIE 16th Annual International Symposium on Aerospace/ Defence Sensing, Simulation and Controls, Orland, FL, 2002.
25. Brooks, R. A robust layered control system for a mobile robot. *IEEE J. Robotics Automation,* 1986, **2**(1).
26. Brooks, R.A. How to build complete creatures rather than isolated cognitive simulators. Architectures for Intelligence, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991. pp. 225-39.
27. Pembeci, I. & Hager, G. A comparative review of robot programming languages, 2001
28. Biggs, G. & MacDonald, B. A Survey of Robot Programming Systems.
29. Bonasso, R.; Firby, R.; Gat, E.; Kortenkamp, D.; Miller, D. & Slack, M. Experiences with an architecture for

intelligent, reactive agents. *J Experi. Theo. Artifi. Intelli.,* 1997.
30. Shakhimardanov, A. & Prassler, E. Comparitive evalution of robotic software integration systems: A case study. *In* I/RSJ Interntional Conference on Intelligent Robots and Systems, San Diego, 2007.
31. Bensalem, S., *et al.* Designing automous robots. *IEEE Robotics Automation Mag.*, 2009.
32. http://www.uml.org/
33. Carreras, M.; Yuh, J. & Batlle, J. A hybrid methodology for RL-based Behaviour coordination in a target following mission with an AUV, OCEANS, 2001, MTS/IEEE Conference and Exhibition, **4**, 2001.
34. Dickmanns, E. & *et al.* The seeing passenger car VaMoRsP. *In* International Symposium on Intelligent Vehicles'94, Paris, 1994.
35. Albus, J. The NIST real-time control system (RCS): An Application Survey*, In* Proceedings of AAI 1995 Spring Symposium Series, Stanford University, CA, USA, 1995.
36. Schenck, Douglas A. & Wilson, Peter R. Information Modeling the Express Way, Oxford University Press, 1993.
37. Dromey, R. From requirements to design: Formalizing the key steps, Keynote Address, SEFM, pages 2–11. *IEEE Computer Society*, 2003.
38. Brooks, A.; Kaupp, T.; Makarenko, A.; Williams, S & Orebaeck. Towards component-based robotics. *In* Proceedings of the IEEE/RSJ Intternational Conference on Intelligent Robots and Systems (IROS), 2005.
39. Baum W and Parlitz, Go execution framework for robotics applications–description and manual, 2005.
40. http://www.corba.org
41. Srinivasan R, RPC: Remote procedure call protocol specification version 2. RFC 1831. August 1995.

Contributors

Ms Mbali Mtshali obtained her Masters degree in Computer Science from the University of Zululand, South Africa. She is a PhD student in the Department of Computer Science at the University of Pretoria, South Africa. She is a researcher at the Mobile Intelligent Autonomous System Unit at the Council for Scientific and Industrial Research (CSIR), South Africa. Her research interest is software architecture for robotic.

Dr Andries Engelbrecht received his Masters and PhD degrees in Computer Science from the University of Stellenbosch, South Africa, in 1994 and 1999, respectively. He is a Professor in Computer Science at the University of Pretoria, and serves as Head of the Department. He also holds the position of South African Research Chair in Artificial Intelligence, and leads the Computational Intelligence Research Group at the University of Pretoria, consisting of 40 Masters and PhD students. His research interests include swarm intelligence, evolutionary computation, artificial neural networks, artificial immune systems, and the application of these computational intelligence paradigms to data mining, games, bioinformatics, finance, and difficult optimization problems.