Autonomic System Architecture: An Automated Planning Perspective

Falilat Jimoh, Lukáš Chrpa, Mauro Vallati

Autonomic System Architecture: an Automated Planning Perspective

Falilat Jimoh, Lukáš Chrpa, and Mauro Vallati

School of Computing and Engineering
University of Huddersfield
{Falilat.Jimoh,l.chrpa,m.vallati}@hud.ac.uk

Abstract. Control systems embodying artificial intelligence (AI) techniques tend to be "reactive" rather than "deliberative" in many application areas. There arises a need for systems that can sense, interpret and deliberate with their actions and goals to be achieved, taking into consideration continuous changes in state, required service level and environmental constraints. The requirement of such systems is that they can plan and act effectively after such deliberation, so that behaviourally they appear self-aware. In this paper, we focus on designing a generic architecture for autonomic systems which is inspired by the Human Autonomic Nervous System. Our architecture consists of four main components which are discussed in the context of the Urban Traffic Control Domain. We also highlight the role of AI planning in enabling self-management property of autonomic systems. We believe that creating a generic architecture that enables control systems to automatically reason with knowledge of their environment and their controls, in order to generate plans and schedules to manage themselves, would be a significant step forward in the field of autonomic systems.

Keywords: automated planning, urban traffic control, autonomic systems

1 Introduction

Autonomic systems (AS) are required to have an ability to learn process patterns from the past and adopt, discard or generate new plans to improve the process control. The ability to identify the task is the most important aspect of any AS element, this enables AS to select the appropriate action when healing, optimising, configuring or protecting itself. Advanced control systems should be able to reason with their surrounding environment and take decision with respect to their current situation and their desired service level. This could be achieved by embedding situational awareness into them and given the ability to generates necessary plans to solve their problem themselves with little or no human intervention - we believe this is the key to embody autonomic properties in systems. Our autonomic system architecture is inspired by the functionality of the Human Autonomic Nervous System (HANS) that handles complexity and

uncertainty with the aim to realise computing systems and applications capable of managing themselves with minimum human intervention.

The need for planning and execution frameworks has increased interests in designing and developing system architectures which use state-of-the-art plan generation techniques, plan execution, monitoring and recovery in order to address complex tasks in real-world environments [1]. An example of such an architecture is T-Rex (Teleo-Reactive EXecutive): a goal oriented system architecture with embedded automated planning for on-board planning and execution for autonomous underwater vehicles to enhance ocean science [2, 3]. Another recent planning architecture, PELEA (Planning and Execution LEarning Architecture), is a flexible modular architecture that incorporates sensing, planning, executing, monitoring, replanning and even learning from past experiences [4]. The list of system architectures incorporating automated planning is, of course, longer, however, many architectures are designed as domain-specific.

In our previous work, we introduced the problem of self-management of a road traffic network as a temporal planning problem in order to effectively navigate cars throughout a road network. We demonstrated the feasibility of such a concept and discuss our evaluation in order to identify strengths and weaknesses of our approach and point to some promising directions of future research [5, 6]. In this paper, we propose an architecture inspired by Human Autonomic Nervous System (HANS) which embodies AI planning in order to enable autonomic properties such as self-management. This architecture is explained on the example of Urban Traffic Control domain.

2 Urban Traffic Control

The existing urban traffic control (UTC) approaches are still not completely optimal during unforeseen situations such as road incidents when changes in traffic are requested in a short time interval [7, 8]. This increases the need for autonomy in urban traffic control [9–11]. To create such a platform, an AS system needs to be able to consider the factors affecting the situation at hand: the road network, the state of traffic flows, the road capacity limit, accessibility or availability of roads within the network etc. All these factors will be peculiar to the particular set of circumstances causing the problem [6]. Hence, there is a need for a system that can reason with the capabilities of the control assets, and the situation parameters as sensed by road sensors and generate a set of actions or decisions that can be taken to alleviate the situation. Therefore, we need systems that can plan and act effectively in order to restore an unexpected road traffic situation into a normal order. A significant step towards this is exploiting Automated Planning techniques which can reason about unforeseen situations in the road network and come up with plans (sequences of actions) achieving a desired traffic situation.

2.1 Role of AI Planning in Urban Traffic Control

The field of *Artificial Intelligence Planning*, or AI Planning, has evidenced a significant advancement in planning techniques, which has led to development of efficient planning systems [12–15] that can input expressive models of applications. The existence of these general planning tools has motivated engineers in designing and developing complex application models which closely approximate real world problems. Thus, it is now possible to deploy deliberative reasoning to real-time control applications [16–18]. Consequently, AI Planning now has a growing role in realisation of autonomic in control systems and this architecture is a leap towards the realisation of such goal. The main difference between traditional and our autonomic control architecture is depicted in Figure 1. Traditionally, a control loop consists of three steps: sense, interpret and act [19]. In other words, data are gathered from the environment with the use of sensors, the system interprets information from these sensors as the state of the environment. The system acts by taking necessary actions which is feedback into the system in other to keep the environment in desirable state. Introducing deliberation in the control loop allows the system to reason and generate effective plans in order to achieve desirable goals. Enabling deliberative reasoning in UTC systems is important because of its ability to handle unforeseen situations which has not been previously learnt nor hard-coded into a UTC. This helps to reduce traffic congestion and carbon emissions.

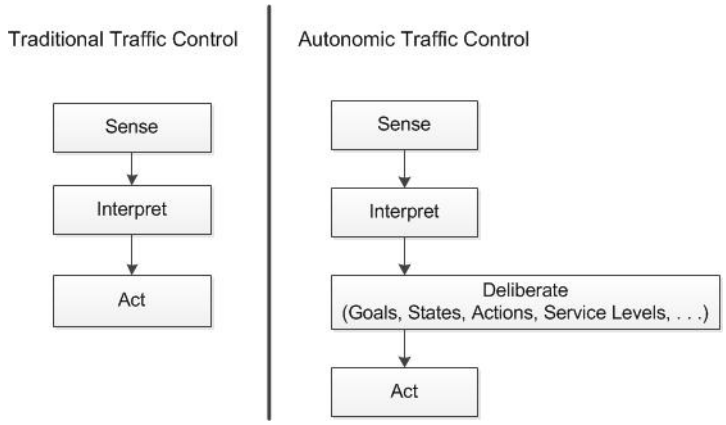

Fig. 1. Illustration of Autonomic System Architecture in Urban Traffic Control

3 Automated Planning

AI Planning deals with the problem of finding a totally or partially ordered sequence of actions whose execution leads from a particular initial state to a

state in which a goal condition is satisfied [20]. Actions in plans are ordered in such a way that executability of every action is guaranteed [21]. Hence, an agent is able to transform the environment from an initial state into a desired goal state [22, 23]. A planning problem thus involves deciding "what" actions to do, and "when" to do them [24, 25].

In general, state space of AI planning tasks can be defined as a state-transition system specified by 4-tuple (S, A, E, γ) where:

- S is a set of states
- A is a set of actions
- E is a set of events
- $\gamma : S \times (A \cup E) \rightarrow 2^S$ is a transition function

Actions and events can modify the environment (a state is changed after an action or event is triggered). However, the difference between them is that actions are executed by the agent while events are triggered regardless of agent's intentions. Application of an action a (or an event) in some state s results in a state in $\gamma(s, a)$ ($\gamma(s, a)$ contains a set of states). It refers to non-deterministic effects of actions (events).

Classical planning is the simplest form of AI planning in which the set of events E is empty and the transition function γ is deterministic (i.e. $\gamma : S \times A \rightarrow S$). Hence, the environment is static and fully observable. Also, in classical planning actions have instantaneous effects.

Temporal planning extends classical planning by considering time. Actions have durative effects which means that executing an action takes some time and effects of the action are not instantaneous. Temporal planning, in fact, combines classical planning and scheduling.

Conformant planning considers partially observable environments and actions with non-deterministic effects. Therefore, the transition function γ is non-deterministic (i.e. $\gamma : S \times A \rightarrow 2^S$). The set of events E is also empty.

For describing classical and temporal planning domain and problem models, we can use PDDL [26], a language which is supported by the most of planning engines. For describing conformant planning domain and problem models, we can use PPDDL [27] (an extension of PDDL) or RDDL [28], languages which are supported by many conformant planning engines. Our architecture can generally support all the above kinds of planning.

4 Autonomic Computing Paradigm

We use the term "autonomic system" rather than autonomic computing to emphasise the idea that we are dealing with a heterogeneous system containing hardware and software. Sensors and effectors are the main component of this type of autonomic system architecture [29]. AS needs sensors to sense the environment and executes actions through effectors. In most cases, a control loop is created: the system processes information retrieved from the sensors in order to be aware of its effect and its environment; it takes necessary decisions using its

existing knowledge from its domain, generates effective plans and executes those plans using effectors. Autonomic systems typically execute a cycle of monitoring, analysing, planning and execution [30].

System architecture elements are self-managed by monitoring, behaviour analysing and the response is used to plan and execute actions that move or keep the system in desired state. Overall self-management of a system is about doing self-assessment, protection, healing, optimisation, maintenance and other overlapping terms [29]. It is important to stress that these properties are interwoven. The existence of one might require the existence of the others to effectively operate. For instance, a self-optimisation process might not be complete until the system is able to self-configure itself. Likewise, an aspect of a proactive self-healing is an ability of the system to self-protect itself. Any system that is meant to satisfy the above objectives will need to have the following attributes:

- Self-awareness of both internal and external features, processes resources, and constrains
- Self-monitoring it existing state and processes and
- Self-adjustment and control of itself to the desirable/required state
- Heterogeneity across numerous hardware and software architectures

4.1 Case Study of Human Autonomic Nervous System

Human breathing, heartbeat, temperature, immune system, repair mechanisms are all to a great extent controlled by our body without our conscious management. For instance, when we are anxious, frightened, ill or injured, all our bodily functions evolves to react appropriately. The autonomic nervous system has all the organs of the body connected to the body central nervous system which takes decisions in order to optimise the effective functioning of other organs in the body. The sensory cells, senses the state of each organs in relation to the dynamically changing internal and external environment. This information is sent to the brain for interpretation and decision making. The execution of relevant action is sent back to the organs via the linking nerves in real time. This process is repeated trillions of time in seconds within the human body system.

5 System Architecture

Our system architecture is divided into four main blocks. Description of each block is given in the subsequent subsections, however, due to space constraints we cannot go into very details. The entire architecture comprises of a declarative description of the system under study; the individual components that make up such system; the dynamic environment that can influence the performance of such system ; its sensing and controlling capabilities and it ability to reasoning and deliberate.

Our system architecture consists of four main blocks:

- The System Description Block (SD)

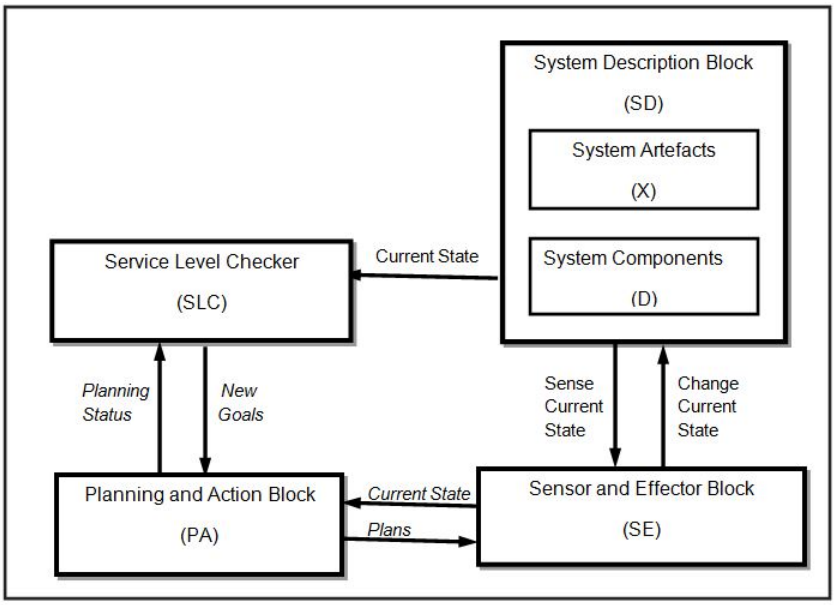

Fig. 2. Diagram of our autonomic system architecture illustrating the relationship between the four main blocks

- The Sensor and Effector Block (SE)
- The Service Level Checker (SLC)
- The Planning and Action Block (PA)

5.1 The System Description Block (SD)

The system description block stores the information about the environment. Formally, the system description block is a triple $\langle D, X, S \rangle$ where

- D is a finite set of system components
- X is a finite set of artefacts
- S is a set of states, where each state $s_i \in S$ is a set of assignments $f_k(y_l) = v_m$ where $y_l \in D \cup X$ and v_m is a value of y_l

System components are parts of the system which are fully controlled by the system. In human body, components are, for example, legs, kidney, hands, lungs etc. Artefacts refer to external objects which are in the environment where the system operates. In analogy to human body, artefacts are objects the human body can interact with. States of the system capture states of all the components and artefacts which are considered. For example, a stomach can be empty or full, water in a glass can be cold or hot etc.

In Urban Traffic Control (UTC) domain, for instance, SD consists of a declarative description of the road network of an area that need to be controlled while

optimising traffic flow pattern. The components are all the objects that can be controlled by the system, for example, traffic control signal heads, variable message signs(VMS) and traffic cameras. The artefacts, objects which cannot be directly controlled by the system such as roads layout and infrastructures.

5.2 The Sensor and Effector Block (SE)

A human body monitors a state of its components (organs), where one can feel pain, hunger, cold etc., and states of artefacts (external objects) with sensors such as skin, eyes, nose etc. SE is responsible for sensing in order to determine a current state of the system (environment) which is stored in SD. Different sensors can provide an information about states of the system components and artefacts. Since the outputs of the sensors might not directly correspond with the representation of states. Formally, we can define a function $\eta : O \rightarrow S$ which can transform a vector of outputs of different sensors (O is the set of all such possible vectors) to a state of the system (S is a set of all possible states of the system). For example, in the UTC domain, sensing can be done by road sensors and CCTV cameras.

SE is also responsible for executing plans (sequences of actions) given by the Planning and Action block in Figure 2. Executing actions is done via system components which indirectly affect artefacts as well. In the UTC domain, a plan to change a traffic light from red to green would cause vehicles to start moving through the intersection. Since we have to consider uncertainty, after executing a sequence of actions, the sensors sense the current state, and if the current state is different from the expected one, the information is propagated to the Planning and Action block which provides a new plan.

5.3 The Service Level Checker (SLC)

The service level checker, as the name implies, repeatedly checks the service level of the entire system to see if the system is in a 'good condition'. It gets the current state of the system from SE and compare it with the 'ideal' service level. If the current state is far from being 'ideal', then the system should act in order to recover its state to a 'good condition'. In analogy to a human body, if it is infected by some virus, the body cells detect this anomalies and start acting against the virus. Formally, we can define an error function $\epsilon : S \rightarrow \mathbb{R}_0^+$ which determines how far the current state of the system is from being 'ideal'. $\epsilon(s) = 0$ means that s is an 'ideal' state. If a value of the error function in the current state is greater than a given threshold, then SLC generates goals (desired states of a given components and/or artefacts) and passes them to the Planning and Actions block (see below).

An example of this in UTC domain can be seen in an accident scene at a junction. The 'ideal' service level for such a junction is to maximise traffic flow from the junction to neighbouring routes. The present state of the system shows that the traffic at that junction is now static with road sensors indicating static vehicles. This situation is not 'ideal', the error function in such a state is

high, hence, new goals are generated by SLC to re-route incoming traffic flowing towards such intersection. It is also possible to alter the error function by an external system controller. For instance, an autonomic UTC system can also accept inputs from the traffic controller (Motoring Agencies). This means that the system behaviour can be altered by the user if needed. This gives the user a superior autonomy over the entire autonomic system. For instance, despite the human body system having the capacity to fight virus (as mentioned before), an expert (doctor) still may have control, so he can, for instance, prescribe anesthetic to reduce the way the body response to pains. This anesthetic act on the brain or peripheral nervous system in other to suppress responses to sensory stimulation. Our architecture is thus designed to have autonomy over it entire component and artefacts and robust enough to accommodate sovereign external user control.

5.4 The Planning and Action Block (PA)

PA can be understood as a core of deliberative reasoning and decision making in the system. This is 'the brain' of our architecture. PA receives the current state of the system from SE, and goals from SLC. Then produces a plan which is then passed to SE for execution. However, it is well known from planning and execution systems [31] that producing a plan given a planning problem does not guarantee its successful execution. As indicated before SE, which is responsible for plan execution, verifies whether executing an action provided a desired outcome. If the outcome is different (from some reason), then this outcome is passed back to PA which re-plans. It is well known that even classical planning is intractable (PSPACE-complete). Therefore, PA might not guarantee generating plans in a reasonable time. Especially, in a dynamic environment a system must be able to react quickly to avoid imminent danger. Therefore, SLC generates goals which are easy to achieve when the time is crucial.

In analogy to human beings, if hungry they, for instance, have to plan how to obtain food (go shopping). This might not be a very time critical task such as if one is standing on the road and there is a car going fast against him he has to act quickly to avoid a (fatal) collision.

In the UTC domain, if some road is congested, then the traffic is navigated through alternative routes. Hence, PA is responsible for producing a plan which may consist of showing information about such alternative routes on variable message signs and optimising signal heads towards such route.

6 Conclusion

In this work, we have describe our vision of self-management at the architectural level, where autonomy is the ability of the system components to configure their interaction in order for the entire system to achieve a set service level. We created an architecture that mimics Human Autonomic Nervous System (HANS). Our

architecture is basically made of four main blocks which are discussed in the context of the Urban Traffic Control Domain. We describe the functionality of each block, highlighting their relationship with one another. We also highlight the role of AI planning in enabling self-management property in an autonomic systems architecture. We believe that, creating a generic architecture that enables control systems to automatically reason with knowledge of their environment and their controls, in order to generate plans and schedules to manage themselves, would be a significant step forward in the field of autonomic systems.

In a real-world scenario where data such as road queues are uploaded in real-time from road sensors with traffic signals connected to a planner, we believe that our approach can optimise road traffic with little or no human intervention. In future we plan to embed our architecture into a road traffic simulation platform. We also plan to provide a deeper evaluation of our approach, especially to compare it with traditional traffic control methods, and assess the effort and challenges required to embody such technology within a real-world environment. Other aspects of improvement would include: incorporating learning into our architecture which will store plans for the purpose of re-using valid plans and save the cost of re-planing at every instance.

References

1. Myers, K.L.: Towards a framework for continuous planning and execution. In: Proc. of the AAAI Fall Symposium on Distributed Continual Planning. (1998)
2. McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R., McEwen, R.: A Deliberative Architecture for AUV Control. In: Intnl. Conf. on Robotics and Automation (ICRA). (2008)
3. Pinto, J., Sousa, J., Py, F., Rajan, K.: Experiments with Deliberative Planning on Autonomous Underwater Vehicles. In: IROS Workshop on Robotics for Environmental Modeling. (2012)
4. Jimènez, S., Fernández, F., Borrajo, D.: Integrating planning, execution, and learning to improve plan execution. Computational Intelligence **29**(1) (2013) 1–36
5. Shah, M., McCluskey, T., Gregory, P., Jimoh, F.: Modelling road traffic incident management problems for automated planning. Control in Transportation Systems **13**(1) (2012) 138–143
6. Jimoh, F., Chrpa, L., Gregory, P., McCluskey, T.: Enabling autonomic properties in road transport system. In: The 30th Workshop of the UK Planning And Scheduling Special Interest Group, PlanSIG. (2012)
7. Roozemond, D.A.: Using intelligent agents for pro-active, real-time urban intersection control. European Journal of Operational Research **131**(2) (2001) 293–301
8. De Oliveira, D., Bazzan, A.L.C.: Multiagent learning on traffic lights control: Effects of using shared information. In: MultiAgent Systems for Traffic and Transportation Engineering. (2009) 307–322
9. Yang, Z., Chen, X., Tang, Y., Sun, J.: Intelligent cooperation control of urban traffic networks. In: Proceedings of the International Conference on Machine Learning and Cybernetics. (2005) 1482–1486
10. Bazzan, A.L.: A distributed approach for coordination of traffic signal agents. Autonomous Agents and Multi-Agent Systems **10**(1) (January 2005) 131–164

11. Dusparic, I., Cahill, V.: Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. ACM Trans. Auton. Adapt. Syst. **7**(1) (May 2012) 1–25
12. Gerevini, A., Saetti, A., Serina, I.: An approach to temporal planning and scheduling in domains with predictable exogenous events. Journal of Artificial Intelligence Research (JAIR) **25** (2006) 187–231
13. Coles, A.I., Fox, M., Long, D., Smith, A.J.: Planning with problems requiring temporal coordination. In: Proc. 23rd AAAI Conf. on Artificial Intelligence. (2008)
14. Penna, G.D., Intrigila, B., Magazzeni, D., Mercorio, F.: Upmurphi: a tool for universal planning on pddl+ problems. In: Proc. 19th Int. Conf. on Automated Planning and Scheduling (ICAPS). (2009) 19–23
15. Eyerich, P., Mattmüller, R., Röger, G.: Using the Context-enhanced Additive Heuristic for Temporal and Numeric Planning. In: Proc. 19th Int. Conf. on Automated Planning and Scheduling (ICAPS). (2009)
16. Löhr, J., Eyerich, P., Keller, T., Nebel, B.: A planning based framework for controlling hybrid systems. In: Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS). (2012)
17. Ferber, D.F.: On modeling the tactical planning of oil pipeline networks. In: Proc. 18th Int. Conf. on Automated Planning and Scheduling (ICAPS). (2012)
18. Burns, E., Benton, J., Ruml, W., Yoon, S., Do, M.: Anticipatory on-line planning. In: International Conference on Automated Planning and Scheduling. (2012)
19. Gopal, M.: Control systems: principles and design. McGraw-Hill, London (2008)
20. Nau, D., ¡, M., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
21. Fox, M., Long, D.: PDDL2.1: An extension of PDDL for expressing temporal planning domains. Journal of Artificial Intelligence Research (JAIR) **20** (2003) 61–124
22. Gupta, S.K., Nau, D.S., Regli, W.C.: IMACS: A case study in real-world planning. IEEE Expert and Intelligent Systems **13**(3) (1998) 49–60
23. Garrido, A., Onaindia, E., Barber, F.: A temporal planning system for time-optimal planning. In: Progress in AI. Volume 2258 of LNCS. (2001)
24. Gerevini, A., Haslum, P., Long, D., Saetti, A., Dimopoulos, Y.: Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. Artificial Intelligence **173**(5-6) (2009) 619–668
25. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research (JAIR) **14** (2001) 253–302
26. McDermott D. et al.: PDDL–the planning domain definition language. Technical report, Available at: `www.cs.yale.edu/homes/dvm` (1998)
27. Younes, H.L.S., Littman, M.L., Weissman, D., Asmuth, J.: The First Probabilistic Track of the International Planning Competition. Journal of Artitificial Intelligence Research (JAIR) **24** (2005) 851–887
28. Sanner, S.: Relational dynamic influence diagram language (RDDL): Language description. http://users.cecs.anu.edu.au/ ssanner/IPPC_2011/RDDL.pdf (2010)
29. Ganek, A., Corbi, T.: The dawning of the autonomic computing era. IBM Systems Journal **42**(1) 5–5 00188670 Copyright International Business Machines Corporation 2003 2003.
30. Lightstone, S.: Seven software engineering principles for autonomic computing development. ISSE **3**(1) (2007) 71–74
31. Fox, M., Long, D., Magazzeni, D.: Plan-based policy-learning for autonomous feature tracking. In: Proc. of Int. Conf. on Automated Planning and Scheduling (ICAPS). (2012)