The Case for Automated Planning in Autonomic Computing

Biplav Srivastava
IBM India Research Laboratory
Block 1, IIT Delhi, Hauz Khas,
New Delhi 110016, India.
Email: sbiplav@in.ibm.com

Subbarao Kambhampati*
Dept. of Computer Sc. & Engg.
Arizona State University
Tempe, AZ 85287, USA.
Email: rao@asu.edu

Introduction: The objective of this paper is two fold: (i) to motivate early adopters of AC, who are using procedural policies, and to consider automated planning and goal-type policies by highlighting their potential for self-management, and (ii) to foreground some of the challenges and research problems raised in adapting automated planning techniques to AC applications.

The vision of Autonomic Computing (AC)[3] is to improve manageability of complex IT systems by making them self-configuring, self-healing, self-optimizing and self-protecting. This would require that the behavior of system elements are monitored and analyzed, and the performance is used to plan and execute suitable actions to take or keep the system in desirable states. Policy is a popular term in industry to refer to any declarative specification of behavior that is desired from a software system (e.g., agent) and the behavior is enforced by a policy engine. A procedural policy, i.e., Event-Condition-Action (ECA) statement, describes desired behavior and exhaustively lists necessary actions to meet them under all conditions. During runtime, a policy engine will verify the conditions and take the stipulated action. A goal-type policy lists the system's expected behavior (e.g., goal state) and it is left to the policy engine to deliberate and determine what actions need to be taken to ensure the satisfaction of goals. *Automated Planning provides the policy engines for goal-type policies.* We will discuss the issues involved in adapting the automated planning techniques to autonomic computing.

Planning needs of Autonomic Computing: Planning is a critical component of the Autonomic Computing (AC) vision [3] where in the behavior of system elements are monitored and analyzed, and the performance is used to plan and execute suitable actions to take or keep the system in desirable states. In the AC vision[3], four aspects of self-management have been identified. We discuss the role of planning in these aspects.

Self-configuration: deals with installation, configuration and integration of IT systems. The installation procedures work by gathering information about the host environment, figuring out the dependencies among needed tasks and also optimizing performance measures, and finally executing the tasks to realize the changes. Information about host system is increasingly getting standardized along structured formats but the executable tasks can be ad-hoc scripts. Humans want to be closely involved in key decisions during execution.

Self-healing: deals with determination of problematic situations and recovering from them. It requires the system to reason with how activities can be performed, how diagnostic information is produced and how new changes can be affected with minimal cost and maximum benefit. The specification of actions could be known at some level of granularity.

Self-optimizing: deals with improving the performance of running systems by leveraging alternative opportunities. The system would monitor its performance and based on its changing environment, could initiate new changes (e.g., resource re-provisioning).

Self-protecting: deals with monitoring the environment for threats and responding to them. It is related to self-optimizing aspect but with the difference that the situation needs time-bound response and lead to cascading effect. Humans want to be closely involved based on the seriousness of the situation.

Automated Planning Technology: Automated planning is a very wide-ranging discipline characterized by how the environment, the agent's goal and its model of the world are represented. Various types of planners can return sequential plans, conditional plans or a generalized state-action mapping specifying what action to take in any state during exe-

*Research supported in part by an IBM Faculty Award. A longer version of this paper is available at http://rakaposhi.eas.asu.edu/icac-05-longversion.pdf

Proceedings of the Second International Conference on Autonomic Computing (ICAC'05)
0-7695-2276-9/05 $ 20.00 **IEEE**

cution (hence procedural policies), that are optimized with respect to a defined metric [1]. In terms of performance, planning has seen an upsurge in the last 6-7 years with new planners that are orders of magnitude faster than before and are able to scale this performance to complex domains, e.g., metric, temporal and exogenous constraints.

At its simplest, a planning problem PP is a 4-tuple $\langle P, I, G, A \rangle$ where P is the set of predicates, I ($\subseteq P$) is the complete description of the initial state, G ($\subseteq P$) is the partial description of the goal state, and A is the set of executable (primitive) actions. A specification of an action consists of preconditions ($A_i^{pre} \subseteq P$) and postconditions ($A_i^{post} \subseteq P$). A plan for PP is an action sequence S, such that if S is executed in I, the resulting state of the world would contain G. A planner finds a plan by efficiently searching in the space of possible states configurations or action orderings (plans). The problem becomes more complex when the initial state is not completely known (i.e., can be one of many possible states), and/or the actions have non-deterministic effects. Both of these necessitate search in the space of "belief states" (where a belief state is a set of states), resulting in "contingency plans" (which are tree-plans that branch using sensing actions). Contingency plans can be efficiently generated by automated and semi-automated techniques. They cover a significant part of the plan representation offered by general workflow standards such as BPEL4WS[1] (which offer sequence, conditional, parallel, non-deterministic and loop constructs).

Challenges for Planning: Despite the obvious connections between what is offered by the existing planning techniques and what is needed in autonomic computing, successful integration involves overcoming several "impedance mismatch" issues. Based on our experience, we believe that these mismatches include: (a) The plans may be needed even when only a partially complete/correct domain model is available. (b) Plans may be partly manually generated and/or modified upon analysis. (c) Automated plan generation is important but plans could also be obtained by users or domain-dependent methods. (d) Over time, there would be a repository of previously generated and executed plans. They have to be considered while selecting existing or generating new plans.

Perhaps the most important of these is that correct and complete domain theories that are assumed as given by most existing planning techniques are often not available. The following table summarizes the level to which information about the initial state (I), goals (G), action specification (A), existing plans (S) and domain constraints (Constraints) is expected to be available in the different AC scenarios. In self-configuring situation, actions may be scripts whose pre- and post-condition information may not be known and there may be no plans available *a priori*. In self-healing scenario, A is expected so that alternative plans could be explored. In self-optimizing and self-protecting scenarios, a plan would be available for the running system but the goal specification will be more clearly defined for the latter.

Type	I	G	A	S	Constraints
Self-configuring	Yes	Yes	-	-	Yes
Self-healing	Yes	Yes	Yes	-	Yes
Self-optimizing	-	-	-	Yes	Yes
Self-protecting	-	Yes	-	Yes	Yes

The lack of complete domain theories poses several difficulties in both plan synthesis and management. The main technical challenge is that the planner is no longer able to "certify" the correctness of the plan it generated. Techniques are needed to consolidate the domain knowledge the planner has into a form that can be used to partially verify the plan correctness (and commence repairs when it is found to be flawed even with respect to its partial model). Another alternative is to develop automated and semi-authomated (e.g. annotation) techniques that learn the domain model [5].

Current Status: Despite the challenges, existing planning methods can be applied effectively in AC. CHAMPS[2] is an example of domain-dependent planner for AC self-configuration scenarios. ABLE, implemented with a domain-independent planner, is an example of a system that can be used across multiple AC scenarios[4]. Planning can also be used for managing procedural policies - while creating new policies, validating properties with existing policies, updating policies - based on whether a set of policies could be composed. We believe that more effective applications of planning techniques to autonomic computing can be achieved by tackling the challenges outlined in the previous section.

References

[1] Ghallab, M., Nau, D. and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

[2] Keller, A., Hellerstein, J., Wolf, J., Wu, K. and Krishnan, V. 2004. The CHAMPS System: Change Management with Planning and Scheduling. *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2004)*.

[3] Kephart, J. and Chess, D. 2003. The Vision of Autonomic Computing. *IEEE Computer, Vol. 36, No. 1, pp 41-50.*

[4] Srivastava, B., Bigus, J., and Schlosnagle, D. 2004. Bringing Planning to Autonomic Applications with ABLE. *Proc. IEEE 1st Intl. Conf. on Autonomic Computing, New York, USA.*

[5] Srivastava, B., Vanhatalo, J., and Koehler, J. 2005. Managing the Life Cycle of Plans. *Proc. of Innovative Applications of AI.*

[1] *http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/*

Proceedings of the Second International Conference on Autonomic Computing (ICAC'05)
0-7695-2276-9/05 $ 20.00 **IEEE**