Using ABLE to Bring Planning to Business Applications

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

Joseph P. Bigus and **Donald A. Schlosnagle**
Email: {bigus, daschlos}@us.ibm.com
IBM T.J. Watson Research Center
P.O. Box 704, Yorktown Heights,
New York, 10598, USA.

Problem: The success of planning in an application depends as much on the planning techniques used as on the way it is embedded into the runtime. The former represents a knowledge engineering challenge since the formulation of the planning problem (i.e., goal and initial states, actions, plan evaluation metric) should be meaningful in the domain assuming that an appropriate planning technique would be used. The latter refers to development of tools and methodologies to integrate off-line reasoning with runtime so that the most important and useful planning scenarios get efficiently solved.

Planning and execution is even today tightly tied to the specific context of the individual application which can have its own idiosyncracies. This lack of domain independent planning and execution infrastructure makes it hard to understand the role of planning in an application (are the planning needs really special or was the implementation ad hoc?), slows future upgradation to planning advancements and inhibits solution reuse. Business applications today, however, are built along architectures that allow componentization of building blocks, large-scale reuse and easy upgradation/maintenance. Hence, a domain independent planning and execution framework is needed for widely applying planning in business applications. This is what we seek to address and demonstrate in IBM Agent Building and Learning Environment (ABLE) toolkit(Bigus et al 2002).

As an example of business application, we will consider the problem of Automated System Recovery of web applications that are running behind a website. In this scenario, the web applications should be able to automatically self-configure and self-heal in response to runtime exigencies to keep the website available. As a very simple problem instance, let the website run on two high-end machines. The website uses two applications (e.g., servlets) that can run on any of the two machines provided an application server (e.g., WebSphere Application Server) is running on that machine. The applications access a database server (e.g., DB2) and a directory server (e.g., SecureWay), which may run on any machine. The initial state has the two machines running and all the software servers installed. The goal is to have both the applications running over time. If any software server or machine were to fail, the system should be able to infer and initiate actions to migrate the computation in a way that the web applications continue to run. A typical business website runs tens of applications on similarly large number of machines and follows a multi-tiered architecture integrating components from differents part of the firm's business - Products, Accounting and Finance, Supply Chain, Customer Relationship Management, etc.

To our knowledge, ABLE is the first publicly available toolkit that provides general purpose planning and execution support. It is being used to solve a variety of planning applications in IBM including the self-management/autonomic computing scenarios.

Solution Approach: We provide a domain independent *planning and execution environment* in ABLE. ABLE[1] is a toolkit for building multiagent autonomic systems. It provides a lightweight Java agent framework, a comprehensive JavaBeans library of intelligent software components, a set of rule development and test tools, and an agent platform. ABLE supports various type of rules (e.g., If-then-else, Fuzzy rules, Prolog rules) and their corresponding rule engines. A developer can build a composite JavaBean (called an *agent*) by mixing different types of rules and embed the resulting component in an application.

The ABLE Rule Language (ARL) is a rule-based programming language that provides tight integration with Java objects and the ability to externalize business logic using simple business rules or more complex inferencing rules. ABLE provides a set of rule engines ranging from lightweight procedural scripting to medium weight forward and backward chaining, up to the power and speed of advanced AI inferencing techniques. ARL also supports templates to allow customization of rulesets by non-technical users.

We have extended ABLE with a new type of rule that we call planning rules, and it is compliant with the planning community's Planning Domain Description Language (PDDL). Since PDDL comes in various flavors, i.e. levels, the planning rules cannot be tied for processing to any specific planning engine. Therefore, we have developed a general planning framework called Planner4J (Srivastava 2004) comprising of a set of common interfaces[2], and any planner

[1]Available at http://www.alphaworks.ibm.com/tech/able.

[2]It also has utility functions and reference planner implementations.

that is compliant with it can be used to process the planning rules, provided it can handle the corresponding level of expressivity (e.g., PDDL level). The planning framework also provides a common infrastructure so that a variety of planners are available to the developer and new ones can be easily built by reusing much of the existing code. As proof of concept, we have implemented a PDDL1 classical planner and a limited metric planner[3].

The planning ARL consists of specifications about predicates, variables, expressions, inputs and outputs. The important rule blocks are `init()` to initialize the domain information, `process()` to initialize problem information and invoke the planner, and `postProcess()` to cleanup. Control parameters are provided to specify the Objects in the planning problem, and the initial and goal states. In the planning rule block (called `doPlanning()` in the example), each action in the domain is specified as a rule. The planning rule semantics are equivalent to the Planning Domain Definition Language (PDDL) action and very similar in syntax.

The Planning engines incorporated in ABLE process Assertion and Planning rules. This engine makes use of predicates to represent states and uses Java class hierarchies to reason about types. The planning rule semantics are equivalent to the Planning Domain Definition Language (PDDL) action syntax. Control parameters are provided to specify the Objects in the planning problem, and the initial and goal states.

The processing sequence is to:

1. Process all Assertion rules in their declaration order.
2. Build a planning problem specified by the initial condition predicates.
3. Select and fire planning rules (actions) until the goal state is reached.
4. Return a plan solution, a sequence of actions with parameters to operationalize the plan.

The actions in the generated plan are now executed in the given order and basic exception handling is supported to handle runtime errors. See details in the full paper(Srivastava et al 2004). Since actions to carry out a plan are always domain specific, one must provide these actions as methods in a Java class that one writes. The methods must be public, static, return a boolean indicating whether the action worked (true) or not (false), and correspond one-for-one to the planning rules in the ARL file. The ABLE distribution contains a working example of how a plan can be dynamically synthesized and executed.

Related work: The planning area has seen a rush of applications recently. There is also a wide variety of planners available, e.g., LPG[4], Sapa(Do & Kambhampati 2001). However, the success of planning in an application depends as much on the planning techniques used as on the way it is encoded and embedded into the runtime. Tools for building intelligent agents are few and those supporting domain-independent planning and execution are fewer. Soar[5] is a well known architecture in academia to build intelligent agents. It provides tools to build agents in multiple languages and provides support for knowledge representation and inferencing. ABLE is a Java-based toolkit customized for building autonomic and business applications incorporating a range of rules.

[3]The ABLE 2.0.1 version on Alphaworks contains only the classical planner.

[4]http://zeus.ing.unibs.it/lpg/

For planning, GIPO (Graphical Interface for Planning with Objects) is an experimental GUI and tools environment for building planning domain models(McCluskey et al 2003). It can serve a complementary role with ABLE of helping the user analyze the business environment so that a planning model can be built using the ARL representation.

Discussion: The key benefits of using the planning-enabled ABLE are:

- It provides the applications with a common planning and execution platform to embed, test and evolve with state-of-the art planners.
- It supports arbitrary customization of an action's execution-time behavior using Java methods. Furthermore, the action set can be modified in the dynamic environment and a new planning problem posed quite easily.
- It contains a planning framework to develop new planners by reusing existing components.
- The existing range of learning beans, rule types and data filters can be used to build complex planning agents.

In future, we intend to extend the planning and execution capabilities in two directions. On the planning front, we want to improve the current implemetation with better heuristics and tighter ABLE and Planner4J integration, provide newer types of planners (e.g., HTN) and incorporate external planners e.g., Sapa. On the execution front, we want to include fine-grained plan monitoring and execution support so that partially-executed, world-altering actions can be taken into account during replanning.

References

Bigus, J., Schlosnagle, D., Pilgrim, J., Mills, W., and Diao, Y. 2002. ABLE: A Toolkit for Building Multiagent Autonomic Systems. *IBM Systems Journal, Vol. 41, No. 3.*

Do, B., and Kambhampati, S. 2001. Sapa: A Domain-Independent Heuristic Metric Temporal Planner. *Proc. European Conference on Planning.*

McCluskey, T.L., Liu,L., and Simpson, R. 2003. GIPO II: HTN Planning in a Tool-supported Knowledge Engineering Environment. *Proc. Intl Conf. on Automated Planning and Scheduling (ICAPS).*

Srivastava, B., Bigus, J., Schlosnagle, D. 2004. Bringing Planning to Autonomic Applications with ABLE. *To Appear in Proc. IEEE International Conference on Autonomic Computing (ICAC-04), New York, USA.*

Srivastava, B. 2004. A Software Framework for Applying Planning Techniques. *IBM Technical Report RI04001.* Available at http://domino.watson.ibm.com/library/CyberDig.nsf/Home.

[5]http://www.eecs.umich.edu/ soar/