PAGODA

Carolyn Talcott
SRI International
clt@csl.sri.com

October 26, 2005

1 About PAGODA

PAGODA (Policy And GOal based Distributed Autonomy) is a modular architecture for design of (partially) autonomous systems. A PAGODA node (agent) interacts with its environment by sensing and affecting, driven by goals to achieve and constrained by policies. A PAGODA system is a collection of PAGODA nodes cooperating to achieve some mutual goal. The PAGODA architecture was inspired by study of architectures developed for autonomous space systems, especially the MDS architecture [DRRS00] and its precursors [MPPW98].

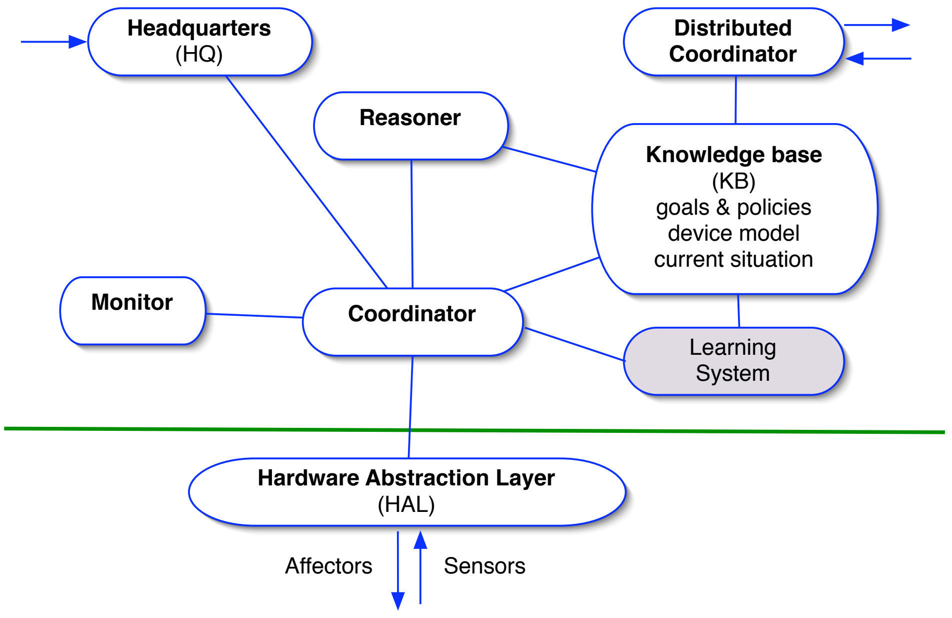

Figure 1: PAGODA node architecture

Figure 1 shows the principal components of a PAGODA node: a knowledge base (KB), a reasoner (R), a monitor (M), a 'hardware' abstraction layer (HAL), and a coordinator (C). There is also a component for communication with a system operator (HQ for headquarters), and a component responsible for distributed coordiation (DC).

The knowledgebase (KB) is the centerpiece. It contains knowledge that is shared and updated by the remaining components. This knowledge inlcudes a wide range of information:

- Goals, what the node or system is trying to achieve. A goal could be a very high-level goal such as carrying out a scientific experiment or mapping a region, or lower level goals that correspond to actions that can be carried out.

- Policies, rules that constrain the actions / interactions a a node or system is allowed to do. A policy might reduce the number of choices for setting parameters, computed based on the device model, for example based on importance of different goals. Another policy could determine trade-offs between speed and power usage. Other policies might constrain what information is exchanged amongst PAGODA nodes, or other agents, and how often.

- A device model that specifies the parameters that can be set (knobs) and read (sensors) and their relationships. It should also specify how values sensed at different nodes can be combined to determine non-local system properties, and the relationships of such local and global measured properties to higher level goals. For example turning on a particular security protocol can be used to achieve security goals guaranteed by that protocol.

- An environment model, representing relevant features of the environment that the node might find itself in. For a mobile node this could include terrain information or building maps.

- The current state, which includes values of variables determined by sensor readings and deduced from actions and information collected from other nodes. It also includes 'situation' information such as the stage in a complex task/mission or progress towards achieving a goal.

The HAL component is an interface to the sensors and affectors. It handles parameter setting and sensor reading requests. In a real system the HAL might map requests to a format that is understood by the actual hardware, or even to a lower level abstraction layer. The intent is that these interactions should obey the 'physics' specified by the device model, but the node needs to be prepared for things to go wrong—some hardware component breaks, the environment is different than expected, it is being operated outside the expected operational mode, and so on.

The HQ component in the other node interface to the external world. HQ transmits new goals and policies from operators/administrators and generally provides a way of driving a PAGODA system.

The job of the reasoner component is to determines proper parameter settings either in response to new goals, starting a new stage of a current goal, or unexpected sensor values, indicating that adjustments need to be made. The reasoner uses information from the KB as basis for its deductions: the device model, the goals and policies, and the current state. When new parameter settings are determined, the reasoner also provides justifications such as what sensor values and/or what relationships from the device model were used to infer the new settings. This can be used for diagnostics if things don't go as expected. The reasoner also specifies sensors that should be monitored and conditions on sensor reading under which the reasoner to be alerted to take corrective action.

The monitor component receives monitoring tasks from the reasoner, reads and evaluates specified sensors, and sends alerts to the reasoner when sensor readings are not within specified limits.

The coordiator component (C) mediates external interactions (with the HAL and with HQ). Thus it can control order and frequency of interactions, and take care of keeping a history of interactions (as specified by policy).

All components can query and update the KB. It behaves like an encapsulated object, handling one request at a time.

The distributed coordination component (DC) is responsible for coordination amongst a group of PAGODA nodes. It may request or transmit information from the KB or negotiate tasks and policies in order to achieve some overall goal.

Future developments include adding a learning component (L) which monitors actions and effects, evaluates them against the model and goals, and determines which strategies are better for given objectives or situations. It may also proactively propose actions to test their effects under certain conditions.

A formal executable specification of PAGODA architecture has been developed in the Maude language [CDE$^+$03a, CDE$^+$03b] and instantiated with a very abstract device model of a radio to test the ideas. Goals are treated as soft constraints on subsets of senor readings. The relationships between affectors (knobs) and sensor readings and between sensor readings and goals are formalized as constraint semi-rings, which provides a clean mathematical basis for solving soft-constraints [BMR97, NFM$^+$05]. The PAGODA node specification was composed with a specification of a radio, MadRad, that simulates actual radio hardware/software including random, unusual and faulty behavior and test scenario developed to illustrate possibly system behaviors.

Ideas for specifying the distributed coordinator are being developed based on a logical approach to distributed monitoring [SVAR04] and a distributed AI approach to distrubuted problem solving [ML04a, ML04b].

References

[BMR97] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, 1997.

[CDE$^+$03a] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and C. Talcott. Maude 2.0 Manual, 2003. `http://maude.cs.uiuc.edu`.

[CDE$^+$03b] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott. The Maude 2.0 system. In Robert Nieuwenhuis, editor, *Rewriting Tech-*

niques and Applications (RTA 2003), volume 2706 of *Lecture Notes in Computer Science*, pages 76–87. Springer-Verlag, 2003.

[DRRS00] D. Dvorak, R. Rasmussen, G. Reeves, and A. Sacks. Software Architecture Themes In JPL's Mission Data System. In *IEEE Aerospace Conference, USA*, 2000.

[ML04a] Roger Mailler and Victor Lesser. Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 438–445. IEEE Computer Society, 2004.

[ML04b] Roger Mailler and Victor Lesser. Using Cooperative Mediation to Solve Distributed Constraint Satisfaction Problems. In *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, volume 1, pages 446–453, New York, 2004. IEEE Computer Society.

[MPPW98] N. Muscetolla, P. Pandurang, B. Pell, and B. Williams. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, 103((1–2)):5–48, 1998.

[NFM$^+$05] R. De Nicola, G. Ferrari, U. Montanari, R. Pugliese, and E. Tuosto. A process calculus for qos-aware applications. In *Seventh International Conference on Coordination Models and Languages*, 2005.

[SVAR04] K. Sen, A. Vardhan, G. Agha, and G. Rosu. Efficient decentralized monitoring of safety in distributed systems. In *Proceedings of 26th International Conference on Software Engineering (ICSE'04)*, pages 418–427. IEEE, May 2004.