Beyond Reactive Planning:
Self Adaptive Software and Self Modeling Software in Predictive Deliberation Management

Topics

Authors: Jack Lenahan, Mike Nash, and Phil Charles
POC: Jack Lenahan
Organization: Office of the Chief Engineer
Space and Naval Warfare Systems Command
Charleston, S.C.
Address: P.O. Box 190022
N. Charleston, South Carolina: 29419
Phone: 843-218-6080
Email: John.Lenahan@Navy.mil
Introduction

- The purpose of this paper is to examine an approach to planning which extends beyond the traditional reactive planning state space. We present the following hypothesis: predictive deliberation management using self adapting and self modeling software will be required to provide mission planning adjustments after the start of a mission. This to support the requirement for dynamically adaptive planning.

- Self adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible.

- Self modeling systems construct their own abstractions as a basis of computational intelligence.

- In order to provide a proper process context for the evolution of software toward this level of autonomy, and in alignment with the proposed planning maturity models, we put forth a concept of a NCW C2 Software Maturity Model. This new C2 software maturity model will take software beyond the service oriented paradigm into a new era of software designing its own replacements or modifications in order to satisfy new command and control requirements.
In this paper we explore the possibility of expanding the definition of predictive deliberation to include computing the time necessary for permitting self adaptation to solve planning problems after the plan execution begins. We discuss several agent architectures to accomplish this based upon the notion of agent reputation. Through self modeling, an “honest agent” will be able to self evaluate whether or not it was capable of producing the changes required to successfully adjust a plan. If the original un-adapted agent believed that it could make the necessary changes to the plan, it would simply “adapt the plan” to the new requirements. If the honest agent did not believe that it had the planning capability required, it would adapt itself.

We advance the following approach to self-adaptation:

- Determine the new capabilities required and then find the new planning capabilities required by conducting a “reputation based search” on the GIG. After discovering the required components and services, the agent will then bind to them thus creating a new version of itself capable of solving the new planning problems in the time allocated by the “predictive deliberator function”.
- If the merging of capabilities results in a set of integrated agents under the control of a manager agent rather than a single large or complex agent, we have dubbed this configuration a “Mogul”.

The reputation mechanism we propose is a modified aggregation operator necessary to perform dynamic reputation scoring possibly from many sources concerning the same agent or collection of agents acting as a mogul.

Finally we propose a C2 agent planning software capability maturity model which would hopefully provide a roadmap for the creation of this level of capability.
Self Modeling Systems

- Self modeling systems construct their own abstractions as a basis of computational intelligence
- The literature describes “Self modeling systems” as systems capable of constructing their own goals rather than simply attempting to carryout goals given to it exclusively from the outside
- Reactive agents are too slow to operate effectively in complex environments
- We need anticipatory agents with models of their own processes so they can identify anomalies in their behavior or in their environment’s behavior.
- Suppose a planning agent fails to successfully plan for a particular task or set of tasks, it is the goal of this type of autonomous system to maintain rules which say “I can do better than this and I need to improve myself by adapting because my current self model does not demonstrate sufficient capability to perform a particular task successfully”
- Please note that this can be enhanced by continuous synthetic stimulation, adaptation, and testing when the agent is not engaged in a mission. The agent can then anticipate future capability needs for itself and include these in its self model. Thus, the agent can begin the process of self adaptation without the pressure of time constraints.
Self Adapting Systems

• Self adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible\(^1\).
Predictive Deliberation and Self Adaptation

• Predictive deliberation attempts to predict in advance how many remaining decisions there are to make, how much time there is to make a particular decision and how long it will take a particular planning agent to make that decision.

• In addition we are modifying this definition to state that predictive deliberation must determine if an agent has the time to ‘self modify’ or adapt itself given a new planning requirement.
• If we assume the perspective of a planning agent in this context, we may have the following conversation with ourselves as planning agents: “I may need to evaluate my own planning competency rating or reputation since my planning reputation does not match the planning skills or planning competency required by the change in the mission. But if I had two additional components inside me, I could accomplish the planning task in half the required time.” In other words, extend the predictive deliberation management functions to ask the following question: “what capabilities would an agent need to perform this task in time and is it possible to reconfigure (adapt) the existing planning agents in time to complete the superior plan?”
How does the agent self adapt to new planning requirements?

• First it asks itself if it is competent to perform the new planning task requirements
• Second if the answer to the above question is no, it can take 2 approaches:
  – ‘Outsource the Planning Work’ work to other agents on a reputation bidding basis
  – Discover other agents through a reputation based search and bind to them forming a larger version of itself
Am I good enough at Planning?

- And if my (agent) planning skills do not match the planning tasks to be accomplished in the time predicted, do I have enough time to reconfigure myself into a better planner? (Predictive deliberation with self adaptation of the agents). By requesting services from other agents, the incomplete agent could integrate the components within its own code body or establish a collaborative planning capability with other agents with the newly required skills.

- Several schemes are available:
  - The component centric work of Karsai et.al involves a technique which includes a supervisory layer and a run time infrastructure which does not use CORBA style interfaces.
  - The next type of agent planning model which I have termed a Mogul, is based upon Minsky’s work and in effect becomes a “society of planning agents”, each agent providing a specific planning capability invoked as required by a supervisory agent or Mogul. It is the Mogul’s task to engage in contract and QoS negotiations with the other agents required for a particular task and to establish the agent based communications and data strategy required to be successful with such an architecture. The predictive deliberator in this case must have predicted that sufficient time exists for agent architecture creation on the fly, contract negotiations, and architectural testing.
  - The “Huge Local Component Library” concept usually is applied to traditional software applications. Everything in the library is put in the include file along with an enormous set of “hard wired rules” which invoke the different modules upon the demand of some supervisory component, recompiling only the required modules at run time.
  - Probabilistic agent architectures which, given a short time to reconfigure by the predictive deliberator, compute a probabilistic model of which other components or agents will be likely satisfy a given set of new requirements and then construct the probabilistic model under the assumption that the new configuration will actually generate an acceptable plan and then execute the plan at run time with no testing.
  - Workflow or BPEL based architectures, which may consist of agents, components, web services, or any combination of these structures. Architectures of this type are less inviting since they usually rely only on web service descriptions (UDDI level descriptions), or component descriptions or other types of Meta-data which do not support the rather detailed level of data modeling required to perform dynamic reconfigurability. The other issue with this approach is that the BPEL and Workflow engines are usually human centric and therefore are incompatible with dynamic reconfigurability.
How do you get there from a software perspective?

Planning Maturity Model

degree of situational understanding

Shared Understanding

Widespread Information Sharing

Controlled/Restricted Information Sharing

planning approaches

Traditional Collaborative Edge Dynamically Adaptive

0 1 2 3 4 5
Planning versus software maturity

Please note that for this discussion edge planning requires that decision rights be moved further down the traditional chain of command such that edge planners may execute their plans with a minimum of organizational overhead. Thus, planning agent reputations will need to emerge which will enable growing confidence in the delegation of decision rights to the edge.
Reputation Model

- Reputation
  - Individual Reputation
  - Group Reputation
    - Direct Reputation
      - Interaction Derived Reputation
      - Observed Reputation
    - Indirect Reputation
      - Prior Derived Reputation
      - Group Derived Reputation
      - Propagated Reputation
Conclusion

• In order to achieve the goals of dynamically adaptive planning and predictive deliberation, we believe that a movement must be made towards software that is able to reconfigure itself on the fly.

• Our research indicates that self modeling and self adapting agents, which can earn a reputation, offer the best approach to solving this problem.

• Our research and that of others also indicates that the use of peer to peer agent communications offers a much more flexible methodology in terms of orchestrating mission sequences than does a BPEL or Workflow engine which requires human intervention.