AI Planning in Portal-based Workflow Management Systems

Michelle Cheatham, Collaborative Technologies, AFRL, Wright-Patterson AFB, OH, michelle.cheatham@wpafb.af.mil
Michael T. Cox, Computer Science Department, Wright State University, Dayton, OH, mcox@cs.wright.edu

Abstract — *Workflow management systems (WfMS) allow multiple agents to work towards achieving a common goal by facilitating communication between them. This paper discusses the distinctive characteristics of portal-based WfMS and considers the utility of using techniques employed in other WfMS environments in this domain. Specifically, the idea of constructing workflows by applying artificial intelligence planning techniques to a user-specified goal is explored.*

1. INTRODUCTION

The term *workflow* has been surfacing in many different contexts recently. In the nineties, workflow referred to a business process, and much effort was devoted to business process re-engineering to improve organizational efficiency. This type of workflow consists primarily of human components and is relatively static over time [3] [11]. An example is the series of steps a customer service department goes through when an item is returned. More recently, workflow has been used to describe a sequence of services executed on a computing grid. These workflows primarily involve software components and are often applied to problems involving scientific simulations [7]. They are dynamically assembled from available components to fit the problem at hand.

For the purposes of this discussion, we will consider a workflow management system (WfMS) to be a method of enabling communication between multiple agents (also called operators, services, or nodes) in order to achieve a specified goal. These agents may be humans, hardware, or software. The type of workflow management system that will be the focus of this paper is that found within web-based portal frameworks. Though there are a range of different WfMS that fall into this category, they all lie somewhere along a spectrum between traditional and grid-based workflows. This paper will discuss the characteristics of portal-based WfMS and examine the utility of applying concepts currently being considered for use in other types of WfMS to portal-based systems. Specifically, the idea of using user-specified goals and artificial intelligence (AI) planning techniques as a way to construct workflows is tested through a prototype portlet.

The remainder of the paper is organized as follows: Section 2 presents an argument on the merits of using AI planning techniques to aid in the creation of workflows; Section 3 discusses the characteristics of portal-based WfMS and their implications on AI planning strategies; a preliminary implementation of a planning-based workflow generator in a commercial portal environment is shown in Section 4; conclusions and future work are covered in Section 5.

2. BENEFITS OF AI PLANNING IN WFMS

Portals revolve around users. However, despite the advertising claims of many companies, current workflow systems within portals require a software developer to construct the workflows. These WfMS require a user to specify *how* a goal is accomplished instead of simply *what* needs to be achieved. In order for a workflow to be created, a user must have a detailed knowledge of every operator within the system, including its pre-conditions, inputs, outputs, and post-conditions. The user must also be proficient in the use of the middleware required to chain the operators together. AI planning techniques can be used to remove this burden from the user by partially automating the workflow generation process [1]. This entails significant up-front development costs because all operators within the system, both human and software, have to be described in terms of a planning language. However, this initial development time is more than offset by the potential ability of users to dynamically create their own workflows without the need to wait for a developer to become available.

While enabling users to create workflows on their own is the primary benefit of automating workflow creation, there are other advantages as well. For instance, it is easy to generate a new workflow if an operator becomes unavailable, resulting in a more fault tolerant system. The planner simply needs to be run again with the same goal, and new operators will be chosen to replace the inoperative one. This is in contrast to current systems, where workflows are created and saved for later use instead of being generated on-demand. In that case, each workflow that uses an operator that has become unavailable must be hand-edited to use a substitute. Similarly, if new operators are introduced into the system, a planner can immediately begin using them in workflows, while a standard WfMS would again require review of all existing workflows and

manual editing of those that could benefit from the newly available operator.

Another advantage of using AI planning in the workflow creation process is that the knowledge captured by describing all of the agents in the system in terms of their pre- and post-condition states can be used in aspects of the portal beyond the workflow management system. In particular, representing the human elements of the system using a planning language opens the possibility of adjusting the user's view of the portal based on her current goals. For instance, if the current user is the actor in the operator *ApprovePlan*, and that operator is the current node in the workflow, then the user's main portal screen could be made to show the plan being approved, along with other pertinent information.

3. OPPORTUNITIES AND CHALLENGES IN PORTAL-BASED WFMS

The unique nature of portal-based workflow management systems makes it worthwhile to examine how some of the problems uncovered by other research regarding AI planning in WfMS apply within a portal framework. While some characteristics of portal-based WfMS make AI planning integration easier, some issues uncovered through research with other types of WfMS are problems in a portal framework as well.

One of the primary concerns when using standard AI planners to solve real-world problems is that the size of the search space may overwhelm the planner, resulting in an inability to generate plans in an acceptable amount of time. Current research suggests a variety of possibilities to deal with this issue: codifying business rules to guide the search process [15], using templates or a plan library as a starting point [3], or taking a mixed-initiative approach [8]. While using these ideas in a portal-based WfMS is possible, the nature of some portal systems limits the severity of this problem. Many portals are based on enterprises or communities of interest, which are organized around a single topic [9] [13]. Workflows created in these environments will consist of operators specific to this topic or from a limited set of generic operators. This implies that the search space is generally small for this type of portal framework.

Another issue when using AI planning in workflow management systems is the language needed to describe the workflow operators. There are competing requirements in this area: end users naturally think in the vocabulary of the problem domain, and it is best if the language allows them to specify their goals in these terms; however, the performance of workflows generated by AI planners is limited by how much information the language provides about the relationships, capabilities, and trade-offs of available operators [7]. Some research in this area is based on building two separate ontologies to describe problems: one for the domain-specific concepts and another for the planning concepts [3]. However, this level of expressiveness may not be needed in portal-based WfMS. As mentioned earlier, the number of different operators is often relatively small, and these operators usually exist within the same limited domain. Therefore, the language used to describe the operators' pre-conditions and effects need not be as complex as that used with other types of WfMS. Another relevant feature of portal-based workflow management systems is that they have not been designed with inter-operability in mind [15]. Even if this were not the case, it will be some time before most organizations will consider adopting workflow operators not under their control. While this reluctance obviously limits the available workflow operators and therefore the types of problems that can be solved by the organization's employees (or members) by creating workflows, it also reduces the demands placed upon the planning language. For example, the planning language does not need to be expressive enough to describe operator characteristics related to quality of service and trust concerns, because these issues are not critical if all of the operators are under the organization's control and can be dealt with internally by the organization if a problem develops.

Finally, unlike most grid computing environments, portal-based WfMS typically do not use a peer-to-peer architecture. Instead, the portal server acts as a focal point for facilitating communication among software applications participating in various workflows. The implication of this is that it is not necessary to provide dynamic discovery mechanisms for workflow nodes within a portal-based WfMS.

A common problem in all types of WfMS, portal-based or otherwise, is that oftentimes a workflow cannot be completely specified at design time because later actions may have complex dependencies on information gathered during the execution of earlier actions. There are several different strategies for handling this complexity. If the dependencies between early and later nodes are understood at design time and the only information that is missing is the actual outcome of the earlier nodes, then a contingency planner can be used. This type of planner can generate plans containing operators that may not actually be used during execution, due to some operators having uncertain effects [11]. Another tactic is to interleave the planning and execution stages. In this case, workflows may initially contain some nodes that are high-level placeholders that are refined as execution progresses [7]. This method places more demands on both the planner and the workflow management system. The planner has more operators to consider because there may be multiple synonymous operators at varying levels of abstraction, and the WfMS must pace the workflow execution to allow the planner time to fully specify nodes before the workflow reaches them. A third alternative, proposed in [15], is to use business rules as a basis for "transformational rules," which would specify how a workflow should be adapted in light of new

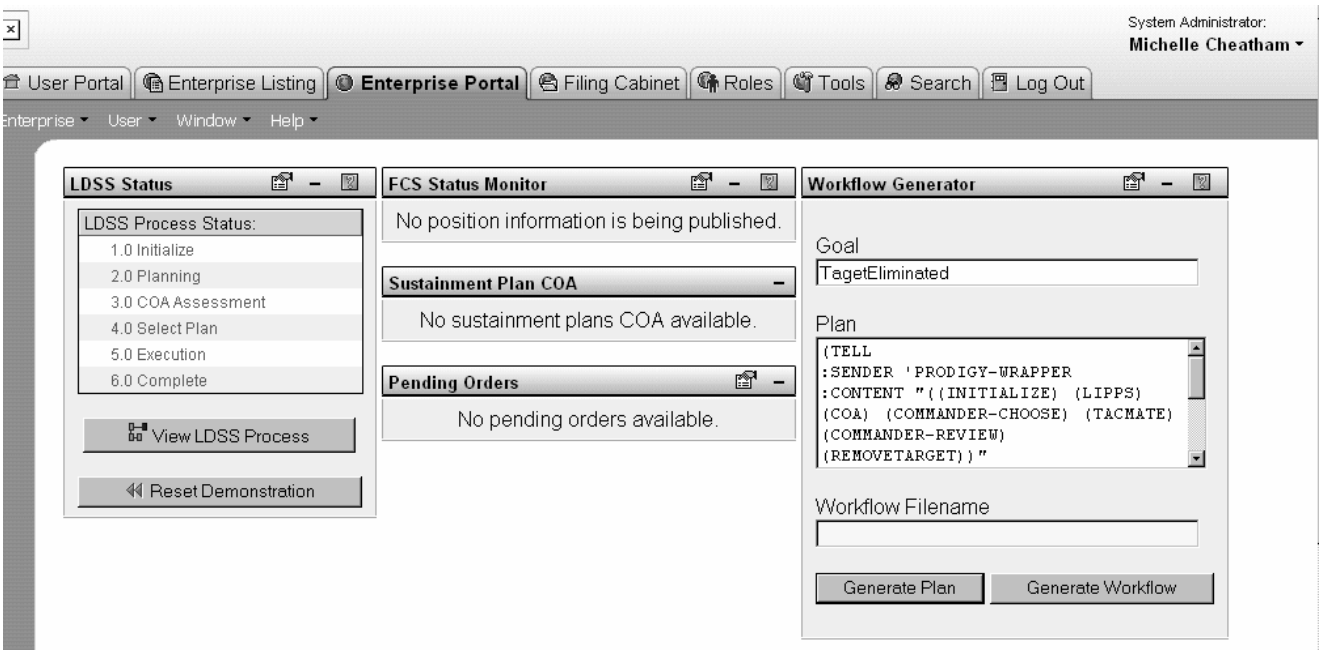

Figure 1 – Combat Decision Support System portal

information gained during execution. This method would be challenging for many organizations to implement because business rules often exist as tacit knowledge within the company and are not easily codified.

Access control is another issue that must be handled in all types of WfMS, including those that are portal-based. In grid-based workflow management systems, access control is very complex due to the nature of the grid – it is likely that services will be made available by a multitude of different organizations. Access privileges may depend not only on the group or role a given user has, but also on resource usage policies between organizations. These policies may change suddenly – possibly even in the middle of workflow execution – and the WfMS must be able to compensate [7]. Fortunately, the situation in a portal-based WfMS is simplified somewhat by the likelihood of all operators being provided by a single organization. Access privileges will still need to be controlled through user and group attributes, however. These attributes could also be used by the workflow system to prioritize requests.

The greatest challenge regarding incorporating AI planning techniques into existing portal workflow management systems is that most existing software was written using an object oriented paradigm, while workflow operators need to follow a service oriented approach. Object oriented programming has been extremely popular for more than a decade. One of the main ideas of this methodology is creating programs out of loosely coupled components, or objects. Operators within a workflow need to be closer to services than objects, however. Both are loosely coupled, but services encompass complete business functions that are meant to be reused in configurations not thought of when the services were originally developed [12]. Creating the proper services when starting from monolithic legacy systems, even when these systems are object oriented, is not always an easy or straightforward task. Moving to a service oriented architecture (SOA) requires identifying which business functions should be exposed as services, determining the proper interfaces for these services, and finding the underlying code necessary to implement them. Because services represent complete business functions, the code to implement them may need to be integrated from pieces in several different applications [12].

4. PRELIMINARY IMPLEMENTATION

In order to gather first-hand experience using AI planning techniques in a web-based portal environment, we have used PRODIGY [2] [14], a state space planner, to implement a workflow generation portlet within the KnowledgeKinetics™ framework. This portlet is a proof of concept; a more robust implementation will be part of our future work in this area.

KnowledgeKinetics™ [9] is a collaboration framework developed and commercialized by Ball Aerospace and the Air Force Research Laboratory Collaborative Technology and Applications Branch. The collaboration framework is meant to allow geographically distributed teams to collaborate on projects and decision support ranging from product design to research. The workflow system within KnowledgeKinetics™ supports both human and software operators. KnowledgeKinetics™ is based on the J2EE platform; software operators may be written in any programming language, but Java wrappers must be created for them to function within the WfMS. The human operators are integrated into the portal framework and can

monitor the user's interactions with entities inside the portal. For example, human operators include actions such as a user filling out a form, approving/choosing an option, uploading a document. When a developer creates a workflow, she first checks to see that all necessary operators are available. If not, additional applications are integrated into the system. Once all of the required operators are available, they are dragged into place using the workflow integrated development environment (IDE), along with process control nodes such as conditional branches, loops, and parallel series. Nodes in the workflow are connected by joining the outputs of some to the inputs of others.

KnowledgeKinetics™ exemplifies many of the characteristics of portal-based workflow management systems discussed previously. The system supports both human and software operators. Workflows in the system are a blend between static and dynamic: some workflows represent standard business processes that seldom change, such as travel expense approval; others are more dynamic in nature, such as those created to chain together simulation tools to do what-if analyses. Software developers are required to create all but the simplest workflows due to the knowledge required about each of the available operators and the need to write scripts that act as "glue" by passing information between some workflow nodes. In addition, the KnowledgeKinetics™ server acts as a broker between all of the agents in the system. Finally, all existing KnowledgeKinetics™ workflows use agents belonging to an individual organization.

There are many different AI planners available, see [11]. PRODIGY, a domain-independent state space planning tool, was chosen for this implementation. PRODIGY has a partial order planning mode – in addition to finding a sequence of operators to achieve a given goal, it is also capable of recognizing when some operators can be executed in parallel.

The prototype workflow generation tool we have implemented has been applied to a prototype Combat Decision Support System (CDSS). This portal was developed several years ago as a proof of concept demonstration of the kind of assistance that a sophisticated web-based portal could provide to the military with respect to command and control operations. The CDSS portal serves as a focal point for a commander monitoring a battle. There are portlets available to plan a battle, simulate the plan, issue orders, monitor assets, and watch the battle unfold. Workflow nodes to support these activities, as well as standard KnowledgeKinetics™ operators, such as sending a notification message to a user, getting a user to approve a proposal, and tasking a user to fill out an online form, also exist within CDSS. Choosing the CDSS portal as our implementation target allowed us to examine the issues arising from attempting to retrofit an existing system to take advantage of AI planning techniques.

In order to use AI planning to create workflows within the KnowledgeKinetics™ framework, additional information is required in the workflow node representation. Workflow nodes currently consist of the following information: type, name, inputs, outputs, and action (either a script or a method name). In addition to these fields, the pre- and post-conditions of each node must also be stored. An example of the new node representation is shown in Figure 2. The operator in the example is a software tool that analyzes a set of alternative courses of action. This operator takes as input a set of potential plans and returns a risk analysis of each one. The *Resource Name* and *Resource Key* fields indicate which software agent provides this action. The *preconds* and *effects* sections indicate to the planner that this operator can be applied only after a set of plans have been created and will result in each plan within the set being evaluated. For a more thorough discussion of the PRODIGY section of the operator definition, see [2].

```
(OPERATOR COA
; Type | Resource Activity
; Name | COA
; Attributes
; Resource Name | COA Assessment
; Resource Key |
AgentProxyHome.MyCommunity.1089638838578
; Inputs
; plans | java.util.Vector
; Outputs
; risk | java.util.Hashtable
  (params <planset>)
  (preconds
    ((<planset> SETOFPLANS))
      (forall ((<plan> (and PLAN
        (gen-from-pred (memberOf <plan>
                        <planset>)))))
      (created <plan>))
  )
  (effects
    ()
    ((add (evaluated <planset>))
  )
```

Figure 2 – Operator representation

In our system, the same file contains the information required by both the planner and the workflow engine for all operators within an enterprise. This alleviates some consistency issues and simplifies the creation and use of a domain language that describes all of the available operators. We are currently developing a portlet to assist software developers in creating new operators and adding the required information to this file. In many current systems, including KnowledgeKinetics™, new operators are added in the same IDE used to create workflows. This can potentially cause problems because it creates a temptation for developers to create "glue" nodes that are tightly coupled to other nodes in the workflow or that are useful only in the workflow they are currently creating. A separate interface to create new nodes independent of any specific workflow will help to emphasize the ideal of developing independent services that are generic enough to be used in many different circumstances.

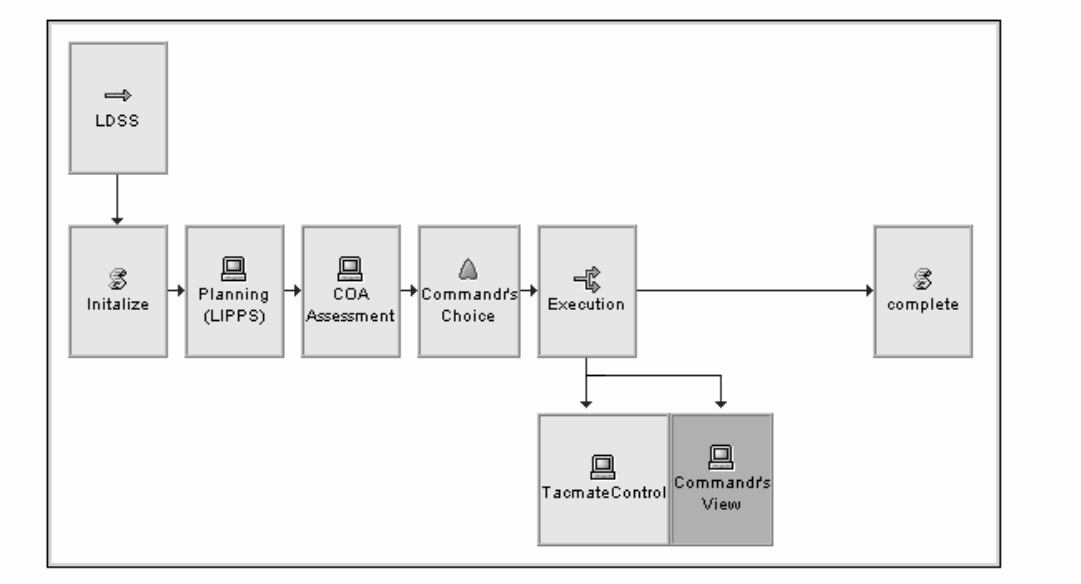

Figure 3 – Ready-to-execute workflow

In order to create a new workflow, an end user first logs into the Combat Decision Support System portal. We have created a new portlet that is viewable on the main screen of the enterprise (Figure 1). The user enters the goal of the workflow into the text box at the top of the portlet. An integrated help system provides a dynamic list of all possible goals within the enterprise, based on the currently available workflow operators. Complex goals can be made by joining individual goals with boolean operators. Once the goal has been entered, the user clicks on the *Generate Plan* button, which causes the portlet to communicate with the PRODIGY server via Prodigy/Agent [5] [6]. Prodigy/Agent is a Java-lisp interface that allows Java-based clients to communicate via KQML [10] messages with the PRODIGY server in order to establish goals and generate plans. An example goal and the resulting plan can be seen in Figure 1. Once a suitable plan has been generated, the user creates the workflow by clicking on the *Generate Workflow* button in the lower right corner of the portlet. The user is then taken to a screen containing the ready-to-execute workflow, which is shown in Figure 3.

Overall, using an AI planning tool to facilitate workflow creation within the KnowledgeKinetics™ framework was a relatively straightforward task. The manner in which the workflow nodes were described had to be changed in order to incorporate the pre- and post-conditions of the operator, but this information (which is used by both the workflow generator and the PRODIGY planner) is stored in a single file and is therefore not difficult to maintain. As discussed previously, the language used to describe the operators was relatively simple. Despite the overall complexity of the Combat Decision Support System domain, the number of operators was not large enough to make generating the plan a time-intensive task. By far the most time consuming part of the process was rewriting some of the operators within the system in order to decouple them enough to exist as independent services, without the need for scripts to glue them together.

5. CONCLUSIONS AND FUTURE WORK

This paper illustrates how the unique characteristics of portal-based workflow management systems – a combination of human and software agents, the limited scope of domains, a centralized architecture, and agents located within the boundaries of the organization – influence the use of AI planning techniques to facilitate workflow generation as a means of enabling multiple agents to work together to achieve the user's goal.

Much of the research into applying AI planning to this type of problem stems from the area of grid-based services. While some of the problems encountered on grids are not relevant in a portal environment, many remain important issues. Future work shall include examining the potential of less centralized architectures for portal-based workflow management systems that would allow for more complex communication between various agents. Using ontologies and other semantic web standards to allow a more diverse collection of agents to work with each other and with a large set of data resources could also be explored. The utility of creating agents within workflow management systems that are more goal-centric could be considered, along with the possibility of integrating them by analyzing goal and subgoal relationships rather than by using domain specific knowledge and constraints.

As Curbera points out in [4], it may be some time before grid-based service oriented computing comes into its own. In the meantime, our research has shown that some of the same ideas can be used to improve portal-based workflow management systems today.

REFERENCES

[1] Allen, J. F., Hendler, J. & Tate, A. (Eds.). *Readings in planning*, San Francisco: Morgan Kaufmann, 1990.

[2] Carbonell, J. G. et al. "Prodigy4.0: The Manual and Tutorial," technical report CMU-CS-92-150. Computer Science Department., Carnegie Mellon University, 1992.

[3] Chung, P.W.H. et al, "Knowledge-based process management – an approach to handling the adaptive workflow," Knowledge-Based Systems, 16, 149-160, 2003.

[4] Curbera, Francisco et al, "The Next Step in Web Services," Communications of the ACM, Vol. 46, No. 10, October 2003.

[5] Cox, M. T., Edwin, G., Balasubramanian, K., & Elahi, M. "Multiagent goal transformation and mixed-initiative planning using Prodigy/Agent," in *Proceedings of the 4th International Multiconference on Systemics, Cybernetics and Informatics*, Vol. 7 p. 1-6, 2001.

[6] Elahi, M. M. "A distributed planning approach using multiagent goal transformations," masters dissertation, Wright State University, Computer Science and Engineering Department, Dayton, OH, 2003.

[7] Gil, Yolanda et al, "Artificial Intelligence and Grids: Workflow Planning and Beyond," IEEE Intelligent Systems, 26-33, 2004.

[8] Kim, Jihie et al, "An Intelligent Assistant for Interactive Workflow Composition," ACM IUI '04, January 13-16, 2004.

[9] KnowledgeKinetics™ homepage, https://k2.knowledgekinetics.info

[10] Knowledge Query and Manipulation Language (KQML) specification, http://www.cs.umbc.edu/kqml/kqmlspec/spec.html

[11] Moreno, M.D.R. and Kearney, P. "Integrating AI planning techniques with workflow management system," Knowledge-Based Systems, 15, 285-291, 2002.

[12] Papazoglou, Mike P, "Service-Oriented Computing: Concepts, Characteristics, and Directions," Proceedings of the Fourth International Conference on Web Information Systems Engineering, IEEE, 2003.

[13] Smith, Reid G. and Farquhar, Adam, "The Road Ahead for Knowledge Management: An AI Perspective," AI Magazine, 17-40, Winter 2000.

[14] Veloso, M. M., et al. "Integrating planning and learning: The PRODIGY architecture," Journal of Theoretical and Experimental Artificial Intelligence. 7(1): 81-120, 1995.

[15] Yang, Jian et al, "A Rule Based Approach to the Service Composition Life-Cycle," Proceedings of the Fourth International Conference on Web Information Systems Engineering, IEEE, 2003.

[16] Yang, Jian, "Web Service Componentization," Communications of the ACM, Vol. 46, No. 10, 35-40, October 2003.