SWIM: An AI-based System for Organisation Management

Pauline M. Berry
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
berry@ai.sri.com

Brian Drabble
CIRL
University of Oregan
1269 University of Oregon
Eugene, OR 97403
drabble@cirl.uoregon.edu

Abstract

This paper reports on the development of the Smart Workflow for ISR Management (SWIM) system which has been designed to enable complex systems, business and software, to be controlled within the workflow management paradigm. The system exploits recent advances within the AI community in reactive control, scheduling and continuous execution. SWIM extends the workflow paradigm to respond to the dynamic and uncertain environments by viewing the control processes themselves as dynamic evolving entities. SWIM is being applied to the domain of Information Surveillance and Reconnaissance (ISR), a highly reactive domain where continual and complex requirements for information acquisition, analysis and distribution must be satisfied within a temporal and resource constrained setting. Similar problems occur in the space domain in the development of missions, science experiments, payload check out, etc. The SWIM system comprises two main components: process manager and the dynamic execution order scheduling system. Details of both of these components are provided in the paper.

Introduction

Many applications in the space domain require the close cooperation of a number of individuals and groups e.g. assembly, integration and test, mission management, science experiment planning, etc. These applications can be categorized as a number of agents (human and/or software) working together on a common task. The agents themselves can be within the same group or spread over different continents and time zones. For example, the components of a spacecraft may come from different contractors, e.g. Rockwell, ESA, etc, and their design, development and integration is a complex coordination task. In addition the agents can be working on more than one task, i.e, resources shared across multiple missions, e.g. Galileo, Voyager, etc. SWIM is designed to work at this coordination level and not at the level of controlling values, motors and engines. Coordinating and tasking a group of agents raises a number of key research questions:

- How to identify the best agent for a task? For example, which of my three telescope schedulers is the best for the task?
- If no agent(s) can be found for a task how can it be divided into sub-tasks for which agents can be found? For example, in house testing is no longer available so what is the process to sub-contract the work?
- How is information communicated between agents, when and what should be sent? For example, if the power specification for a probe changes which agents should be notified?
- How can the workload on individual agent be monitored and reallocated with minimum disruption? For example, if the solar panels are three days late which agents should be reassigned?

As described above one of the key questions is dealing with the dynamics of coordinating cooperating and distributed agents. A task can fail for a number of reasons and the reason can be as simple as a missed deadline e.g. the solar panels will be three days late, through to the complete re-tasking of a spacecraft program e.g. we are changing from solar panels to nuclear fusion as the power source. In most cases the failures can be solved by inserting a few new tasks and modifying and/or deleting others. However, these new tasks may cause further knock on effects within the network with some agents becoming overloaded and other sitting idle. This process of task assignment and balancing is often referred to in the literature as process or workflow management. Many domains of interest to the workflow community are characterized by

ever-changing requirements and dynamic environments. However, traditional workflow systems provide only limited reactivity and flexibility. Within the AI community, work on reactive control has led to the exploration of techniques for intelligent process management to meet the requirements of adaptivity for dynamic and unpredictable environments.

This paper describes a revolutionary approach to workflow management using advanced AI planning, scheduling, and reactive control techniques. The system described has been built to manage the highly dynamic processes involved in the intelligence, surveillance and reconnaissance (ISR) domain. The Smart Workflow for ISR Management (SWIM) system comprises two main components: process manager and the Dynamic Execution Order Scheduler (DEOS). The function of the process manager is to select and instantiate processes to deal with *events* occurring in the domain. The function of the DEOS is to develop a coherent schedule for the each of the separate processes identified by the process manager. The schedule assigns resources to each of the activities and tries to maximize the execution window for each activity. Details of the process manager and the DEOS system are provided later in the paper.

The system has been evaluated in the ISR domain to develop plans at the process level, i.e the development of the overall ISR plan, e.g. the development of information needs, collection track generation, etc. This would equate to the process of developing a mission from its inception, through to spacecraft assembly, verification and launch. Changes can occur in many places which require activities being removed, added and in some case modified, e.g. the decision to sub-contract out a component. The first version of the SWIM system was successfully demonstrated to the ISR community in August 1999. The DEOS has been separately evaluated in the air campaign planning domain to generate schedules at the process level, i.e. the development of the overall strategy and at the the lowest level to assign weapon/aircraft to tasks [Drabble 1999].

Motivation

This section provides an overview of the ISR process and shows the links between the problems faced by ISR planner and those faced by mission planners and ground staff.

Intelligence planners are faced with the task of coordinating multiple Information Surveillance and Reconnaissance (ISR) assets to provide as much information about the battlefield as possible and thus

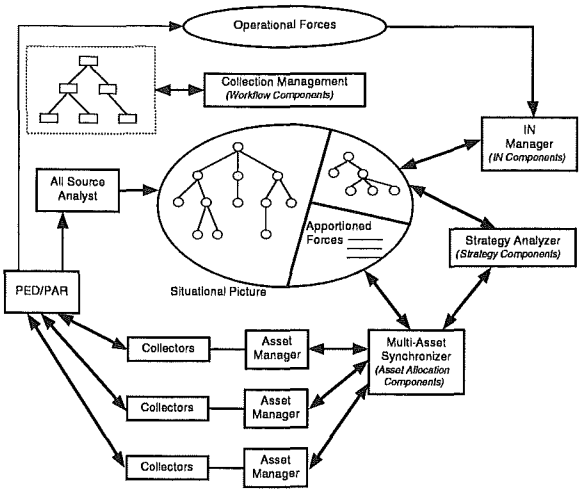

Figure 1: Overview of CPED/PAR Process

increase the effectiveness of fielded forces. The ISR process is driven by Information Needs (INs) which range in complexity from obtaining a complete picture of the electrical power grid of Bosnia to recognizing whether a tank is at a location. Before assets can be tasked with collecting the IN a complex planning process needs to executed. This includes checking if the requested IN is already in a database, is tasked to platform, prioritizing it with other INs, analyzing its overall information gain, etc. A similar process exists in the mission planning domain e.g. a request for an image of the North Atlantic. The INs are divided into different collection tasks depending on time, resources needed, importance, resolution, platform capabilities, risk, etc. The ISR collection (C) tasks cannot be considered in isolation as they are inherently related to processing, exploitation, dissemination (PED) tasks and processing, analysis and reporting (PAR) tasks. This is similar to the case in the space domain where the designers of one component need to be aware of the capabilities and needs of other component developers. The scenario being explored by the AIM project involves the fusion of CPED/PAR tasks as shown in Figure 1.

The development of the SWIM system is funded by the DARPA Advanced ISR Management (AIM) program, which will provide ISR planners with tools for more effective and efficient management of the assets and information in the system. It will address the complete ISR management problem from the strategic development of objectives to the management of individual assets. From a SWIM perspective all these tools (human and/or software) are viewed

as agents that are capable of adding value to the emerging ISR plan. For these agents to work efficiently, they must be organized and managed; that is, a workflow plan that describes how the problem will be solved and identifies the agents necessary to support it must be developed. This will involve the development of intelligent workflow techniques that identify when an agent should be tasked and which tasks are appropriate for the agent to solve. The use of intelligent workflow management together with information discovery and integration are the keys to success in this domain.

ISR Process Modeling Methodology

The AIM process defines the ways in which ISR plans are developed and communicated between agents. Although this planning process involves activities and their coordination, they are described in terms of the activities that take place in the planning process itself (such as plan, analyze, review) rather than containing activities that relate to military effects (such as photograph, intercept). The activities are described using a set of "process verbs" indicating the actions performed at the process level e.g, plan, analyze, review and the process products that are created, modified, used, and authorized within the action. Examples of process products are the documents, reports, orders, letters, and communications (formal or informal) and these are viewed as resources within SWIM. Authority relationships and other conditions can also be modeled and used as an extension of the basic mechanism.

Details of the development of the verb and process product models can be found in [Berry & Drabble 1999]. This framework is general enough to be applicable across various different military domains e.g. logistics, and air campaign planning, as well as many complex business processes and distributed software applications. The *verb/noun(s)/qualifier(s)* (VNQ) model was adopted from a previous modeling exercise that used a similar model to capture air campaign planning processes [Drabble, Lydiard, & Tate 1998]. The SWIM models are encoded in the ACT representation [Myers 1993], which can be directly executed by the Procedural Reasoning System (PRS). It is hierarchical and provides a rich scheme for both the representation of normative processes and the derivation of new processes based on AI reasoning and planning.

The SWIM System

The Smart Workflow for ISR Management SWIM system is a multi-agent framework for performing and managing complex tasks in dynamic and uncertain environments. It provides taskability (i.e., the ability to formulate and execute processes to achieve assigned tasks at different levels of abstraction) and reactivity (i.e., the ability to adapt behavior based on changes in the operating environment). Tasks may be served by different processes and SWIM can select appropriate processes based on context and adapt processes to reflect new environmental or task features. Tasks may also involve long-term commitments that require look-ahead analysis; for this reason, generative planning technology will eventually be used to compose new plans from libraries of process building blocks *(operator templates)*.

SWIM leverages many of the reactive control capabilities from CPEF [Myers 1998], augmenting them with advanced resource allocation, capacity analysis, and scheduling capabilities. CPEF is a novel continuous planning and execution framework embracing the philosophy that plans are dynamic, open-ended artifacts that must evolve in response to an ever-changing environment. Plans are updated in response to new information and requirements in a timely fashion to ensure that they remain viable and relevant, and replaced by alternatives when they are not. SWIM similarly embraces this philosophy, drawing a parallel between plans and workflow processes.

The Procedural Reasoning System (PRS) [Myers 1997], a hierarchical reactive control system, is used as both an executor for workflow processes, and a high-level controller for the overall system. Advanced techniques to effectively schedule tasks onto processing entities are drawn from recent work on the Squeaky Wheel Optimizer (SWO) [Joslin & Clements 1998] and effectively integrated with the process enactment using novel representations that allow process activities to "breath" within temporal windows.

SWIM supports both *direct* models of execution, e.g. actions performed by the system itself, and *indirect* models of execution for which the system supervises execution by a collection of distributed, e.g. a subcontractor for the solar panels. The indirect model of execution is essential for domains where direct control of processing entities is impossible, including many classes of WFM problems. SWIM employs a *procedure library* (encoded in the ACT representation language [Myers 1993]) providing a seamless

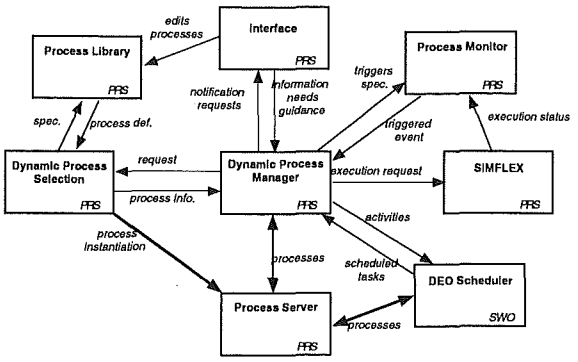

Figure 2: Functional Overview of SWIM

transition between workflow processes (also represented in ACT) and the internal control processes. Elements of the library span multiple abstraction levels and are usable for both process generation and execution, thus supporting smooth transitions between the two capabilities. Process expansion and generation can proceed to arbitrary levels of refinement, with the executor applying additional procedures at runtime to refine tasks to executable activities. Process scheduling and execution operate asynchronously, in a loosely coupled fashion, with agents communicating domain knowledge, activities, requests, and situation information as required. Monitors are automatically created from enacted processes and their constituent activities based on content and context. Future work will define a wide range of monitors, which will be used to support the repair of both specific and generalized types of failure. To date, SWIM has not implemented repair mechanisms but will be building on minimal-perturbation dependency structure methods that have been extended to accommodate generalized failures in CPEF and heuristic techniques.

The SWIM Architecture

The SWIM architecture, shown in Figure 2, is compatible with the WFMC's architectural definition but has distinctive characteristics derived from the field of AI.

Dynamic Process Manager, the hub of SWIM, directing its overall behavior and responsible for managing the instantiation of processes, their dynamic decomposition, their consolidation, the release of activities to the DEO scheduler, the creation of a monitoring plan, and recovery from failure.

Dynamic Process Selector, responsible for the selection, adaptation and evolution of processes. It responds to requests from the Dynamic Process Manager with the appropriate process definition at the required level of abstraction.

Process Library Server, maintains both template processes and process building blocks (or *operator templates*) from which new processes can be generated by the Dynamic Process Selector.

Interface, supports interactions between the user and SWIM and currently includes the ability to edit processes in the Process Library, input requests for information (information needs) and inform the user of critical events and activities.

DEO Scheduler, responsible for the allocation of activities to processing entities (*agents*) over time. It also reasons about windows of opportunity aiding the interaction with higher-level process reasoning.

Process Server, provides a store for multiple processes at a variety of levels of abstraction, e.g. those with scheduled activity information and those without.

Process Monitor, monitors the execution state of the activities and processes using trigger definitions generated automatically by the Dynamic Process Manager.

SIMFLEX (*SImulated FLexible Execution*) a PRS-based simulation environment developed to enable testing, evaluation and demonstration of the dynamic workflow management capabilities of SWIM, [Myers 1998].

Dynamic Process Management

A unique feature of SWIM is the inclusion of an advanced procedural-based reactive controller. The PRS-based controller is organized around an interpreter that runs a tight control loop of *sensing* to detect key changes in the environment e.g. the design, development and launch of Cassini, or sets of assigned tasks, *deliberation* to determine how to respond to sensed changes, and *acting* to execute relevant responses. This approach involves predefined procedure libraries describing processes that can be performed to achieve some goal, or that serve as appropriate responses to designated events (for example, [Firby 1994; Georgeff & Ingrand 1989; Myers 1996; Musliner, Durfee, & Shin 1993; Howe & Cohen 1991]). The bodies of these procedures employ rich operations and control constructs that provide a highly expressive method for representing activity. As such, procedural reactive control is particularly well suited for the *activity-based* paradigm for workflow, although it could readily accommodate artifact and communication-based models

through the introduction of appropriate ontological constructs into the basic process description languages.

DEO Scheduler

The basic concept behind a DEO is to generate schedules quickly and to update them on the fly as new requirements and changes occur. A DEO schedule uses an expressive formalism that breaks down the tasks into the constituent parts. These parts reflect a natural breakdown of the task from a user perspective. In the case of the ISR domain the task is broken down into 4 sub-tasks forming a structure called the PAER model:

- **P**lan: Time to plan the task, e.g. to time taken to identify which telescope scheduler should be used. Once a plan has been identified it is inserted in the slot for other tasks to examine and check. This allows other tasks to identify why the particular scheduler was chosen and the product(s) it will generate, i.e. the telescope schedule.
- **A**cquire: Time to acquire the information necessary to carry out the task, e.g. the observation requests, weather information, etc.
- **E**xecute: Time to carry out the alloted task, e.g. time take to generate the schedule.
- **R**eport: Time to file or report the results of the task, e.g. sending e-mail to the scientists, database update, etc.

Each task is represented by a task specification block (TSB) composed of the PAER sub-tasks. The TSB can "breath" as changes in the domain are reflected as changes in one of the TSB's sub-blocks. For example, if an agent chosen for a task develops a problem during it's **A**cquire phase e.g. the computer collating the observation requests crashes, then the **A**cquire sub task will expand. Alternatively, a second agent may be scheduled with the task inheriting the results from the failing action. An example of a partial PAER network is shown in Figure 3.

The more common reason for a TSB to change is due to a "knock on" effect from another TSB. For example, a change in the **R**eport block of one task may caused the **A**cquire block of a dependent block to expand. By creating a dynamic link between sub-blocks it becomes possible to quickly identify the impact of a change and to identify an appropriate set of repairs. Failures may also cause new TSBs to be added to the schedule to deal with repairs. The dependency links usually reflect an interchange of information between the tasks. The information

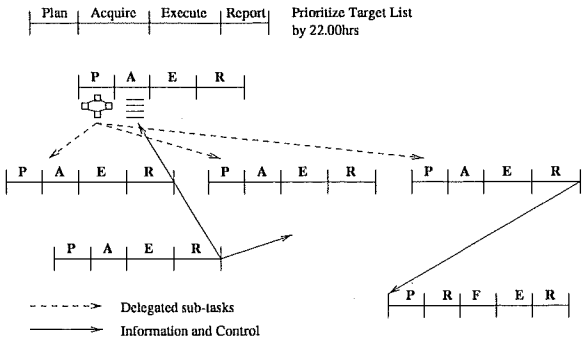

Figure 3: PAER tasking structure

takes the form of products from a task e,g. reports, orders, etc, which reflect new information created or updated by the task. Interactions between the different TSB's is handled by the squeakywheel algorithm which is described later in this section.

A TSB can be generated in response to decision made by other parts of the schedule, e.g. the use of a particular test chamber might dictate that certain additional components are needed. If the test chamber develops a fault and a different test chamber is used then the request for additional components can be removed. The scheduler dynamically launches a new TSB to deal with the chamber substitution and provides feedback to the user of the changes. This approach also has the advantage of hiding unnecessary information, e.g. the staff operating the replacement test chamber do not need to know the details of the failure, just the time at which the spacecraft should be available. By using the DEO model to break the scheduling problem into smaller pieces it allows large scheduling problems to become tractable and maintains the necessary dependencies between the subproblems. Similar task breakdown structures have been developed for other domains including, USAF mission planning and CD manufacturing.

Squeaky Wheel Optimization (SWO) The insight behind SWO is that in any real-world problem it is impossible to capture all associated constraints and that in most cases the context in which the constraints apply cannot be easily determined. SWO uses a priority queue to determine the order in which tasks should be released to a greedy scheduling algorithm. The priority queue is determined by how difficult the task is to deal with that is, the higher the task is in the queue the harder it is to handle it correctly and not by some external priority identified by the user. On each iteration of the algorithm, SWO quickly creates a schedule and

then examines it to identify the parts that were handled badly, for example, task was completed too late or by an unsatisfactory agent. Any task that "squeaks" is promoted up the priority queue, with the distance it is promoted determined by the the extent of the problem. The new priority queue is then used to generate another schedule that is analyzed for problems. This process continues until no significant improvement in the schedule is noted over several iterations. SWO is extremely fast with each iteration taking less than a few seconds, even for large problems.

Summary and Further Work

We have presented an overview of a system for dynamic workflow management. The SWIM system and its generic tasking and process models are applicable across a wide number of applications involving distributed agents (human and/or software) working cooperatively to solve a task. The SWIM system is work in progress and an initial version of the system has been successfully demonstrated in August 1999 to the ISR community. The initial demonstration tackled problems varying from lost assets to large scale re-plans, resource updates and changes in mission objectives. SWIM provides the foundation for workflow enabled reactive control that includes an agent-based architecture, rich modeling and representation of processes and their constituent actions, flexible integration of process instantiation, task allocation and execution, and highly reactive scheduling techniques. Additional experiments have shown that the DEOS scheduler (using a variation of the PAER task model) was able to generate and update schedules for 2500 tasks in less than 5 seconds. Further development of the SWIM system is planned with the main foci being the full integration of the DEOS scheduler, automatic process creation and evolution, advanced failure recovery and repair techniques, and the maintenance and use of the dynamic dependencies between tasks at different levels of abstraction.

In introducing a system such as SWIM there are a number of factors which need to be considered. One of the main concerns is the availability of adequate information on the tasks being performed and the capabilities of the agents being tasked. For example in the design phase it may not be case that a task's duration can be accurately specified. However, it is the case that the system can still perform well with such data, but an answer of between 1 and 60 days may not be useful to a program manager. The current simple models of agent capability need to take into account a number of more realistic concerns, e.g. the change over time between tasks, people's skills changing over time, the combining of skills among agents, etc. It is expected that these simplifying assumptions will be relaxed as more experiments are carried out.

References

Berry, P. M., and Drabble, B. 1999. The AIM Process Modeling Methodology. Technical report, AI Center, SRI International, Menlo Park, CA.

Drabble, B.; Lydiard, T. J.; and Tate, A. 1998. Workflow Support in the the Air Campaign Planning Process. In Myers, K., and Ferguson, G., eds., *Proceedings of the Workshop on Interactive and Collaborative Planningheld as part of Fourth International Conference on Artificial Intelligence Planning Systems (AIPS-98).*

Drabble, B. 1999. Task Decomposition Support to Reactive Scheduling. In *5th European Conference on Planning (ECP-99).* New York, NY: Springer Verlag Press.

Firby, R. J. 1994. Task Networks for Controlling Continuous Processes. In *Second International Conference on AI Planning Systems.* Menlo Park, CA: AAAI Press.

Georgeff, M. P., and Ingrand, F. F. 1989. Decision-Making in an Embedded Reasoning System. In *Eleventh International Joint Conference on AI.*

Hollingsworth, D. 1994. The Workflow Reference Model. Technical Report TC00-1003, Workflow Management Coalition.

Howe, A. E., and Cohen, P. R. 1991. Failure Recovery: A Model and Experiments. In *Nineth National Conference on Artificial Intelligence.*

Joslin, D. E., and Clements, D. P. 1998. Squeakywheel Optimization. In *Fifteenth National Conference on Artificiall Intelligence.* Menlo Park, CA: AAAI Press.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: A Cooperative Intelligent Real-time Control Architecture. *IEEE Transactions on Systems, Man, and Cybernetics* 23(6).

Myers, K. L. 1993. *The ACT Editor User's Guide.* AI Center, SRI International, Menlo Park, CA.

Myers, K. L. 1996. A Procedural Knowledge Approach to Task-Level Control. In *Third International Conference on AI Planning Systems.* AAAI Press.

Myers, K. L. 1997. *User's Guide for the Procedural Reasoning System.* AI Center, SRI International, Menlo Park, CA, version 1.8 edition.

Myers, K. L. 1998. Towards a Framework for Continuous Planning and Execution. In *Proceedings of the AAAI 1998 Fall Symposium on Distributed Continual Planning.* Menlo Park, CA: AAAI Press.