OzCare: A Workflow Automation System for Care Plans

Wenke Lee, Gail E. Kaiser
Department of Computer Science
Columbia University
New York, New York

Paul D. Clayton, Eric H. Sherman
Department of Medical Informatics
Columbia University College of Physicians and Surgeons
New York, New York

An automated environment for implementing and monitoring care plans and practice guidelines is very important to the reduction of hospital costs and optimization of medical care. The goal of our research effort is to design a general system architecture that facilitates the implementation of (potentially) numerous care plans. Our approach is unique in that we apply the principles and technologies of Oz, a multi-user collaborative workflow system that has been used as a software engineering environment framework, to hospital care planning. We utilize not only the workflow modeling and execution facilities of Oz, but also its open-system architecture to interface it with the World Wide Web, the Medical Logic Module server, and other components of the clinical information system. Our initial proof-of-concept system, OzCare, is constructed on top of the existing Oz system. Through several experiments in which we used this system to implement some Columbia-Presbyterian Medical Center care plans, we demonstrated that our system is capable and flexible for care plan automation.

INTRODUCTION

A *care plan* is a specified process for caring for a particular illness; it outlines specific procedures and protocols to be carried out at specific points of an illness or recuperation. Care plans exist most commonly in manual forms, where health care providers are expected to adhere to the textual lists of recommended practices through “check off” compliance. Studies have shown that adherence to the manual plans reduces lengths of hospital stays by 30% and cost of care by 20-25%[3]. It is widely believed that computer-automated care plans will result in more substantial savings in healthcare costs.

Care plan automation is similar to workflow automation because a care plan is a special form of workflow -- it describes the steps of delivering care to patients. However care plans have some unique characteristics. First of all, there are thousands of existing standard care plans and the number is still growing. More significantly, when each care plan is used in practice, it is to be applied to a specific patient and every patient can have individual circumstances that make the usual standard “one-size-fits-all” plan inappropriate. While in other domains staff might be required to stick to policies, even at the cost of individual creativity, to guarantee a high degree of quality assurance, a physician should not be forced to follow standard care plans under “unusual” situations.

Although a number of care plans have been implemented using only Medical Logic Modules (MLMs)[6]at Columbia Presbyterian Medical Center (CPMC), we discovered that MLMs alone are insufficient for care plan management because MLMs are low level, in that each MLM makes only a single medical decision such as generating an alert, making a diagnosis, etc. A care plan normally consists of a partially ordered set of care elements (clinical tasks). Although each care element can be implemented using one (or more) MLM(s), higher level control mechanisms are needed to manage the current status/progress of the entire care plan, the dynamic rescheduling of care elements with respect to the care plan due to resource constraints, and the dynamic restructuring of a care plan due to tailoring to the individual patient's current status. These higher level functions can readily be provided by a workflow management system.

Oz[1] is a collaborative workflow management system developed at the Programming Systems Laboratory (PSL) at Columbia University. Although Oz was originally designed to support software engineering processes, it has a general-purpose rule-based workflow modeling language and a programmable workflow engine, which make Oz applicable to other application domains. Therefore the motivation of our cooperative research (between PSL and CPMC) is to use Oz as the workflow management system that provides the high level control functions required by a hospital care plan system.

In the sections that follow, we first discuss the workflow management requirements of a care plan automation system. Then we provide brief background information on Oz, describe the design and implementation of OzCare, and conclude with discussion and future directions.

WORKFLOW MANAGEMENT

As we discussed in the previous section, a care plan system needs a workflow management component. Evidently the workflow management system should have a general approach to modeling and managing care plans, given the sheer numbers of potential care plans to be installed. It must also be specific enough to give precise guidelines for each patient, based on patient data from other components in the clinical information system. In the following, we examine care plans from several different perspectives to illustrate the requirements of a care plan workflow system.

From the data management point of view, a care plan consists of a set of care elements, each of which are medical care events, or tasks. Each (instance of a) task must be associated with a patient's medical record number (MRN), which is the key used to retrieve all demographic and medical information about the patient. Each task is assigned a group of care givers (physicians, nurses, therapists, etc.).

From the medical care point of view, each care element has a medical activity (also mapped to task), and its start time and stop time. The start time is used to schedule the task and the stop time is the deadline for completion of the task. When assigning and scheduling tasks, resources such as available physicians, hospital beds, operating rooms, etc. must be considered. Before a task can be carried out, some preconditions must be satisfied. These can be categorized into logistic conditions (e.g., the task has been assigned and the care giver is available) and medical conditions (e.g., the patient's vital signs have been stable for the past 24 hours). The evaluation of the medical preconditions normally requires invocation of the MLMs associated with the task. Each task, when finished, also has post-conditions used in determining which task(s) to carry out next. Again, MLMs may be used to obtain "expert system" medical advice to evaluate the current state of the care plan.

From the hospital administration point of view, the workflow system should monitor the progress of care plans. It should generate reminders for pending tasks as well as alarms for past due (not yet completed) tasks. The care plan system must be an integral part of a clinical information system; for example, it needs to communicate with the patient database, event monitors and MLMs using the standard Health Level 7 (HL7) format for information exchange[4]. Moreover, it should also have support for collaborative workflow management so that care plans can be carried out over a healthcare network.

Oz BACKGROUND

Oz provides a rule-based workflow modeling language. A rule generally corresponds to a workflow step, and specifies the step's name; input data and bindings of additional data from queries to the organization-specific database; a condition that must be satisfied before initiating the activity to be performed during the step; the activity, which is normally the invocation of a program or script in the host system; and a set of effects, of which assert the actual results of completing the activity.

The workflow model of Oz is different from that of a typical commercial workflow system: instead of representing tasks as a graph with strictly specified execution order, Oz represents a task as a rule, allowing any ordering consistent with the conditions and effects of rules. The workflow engine enforces that rule conditions are satisfied, and can automate sequences of workflow steps via forward and backward chaining. A user can specify any task (an entry point) he/she desires to perform, and Oz does the backward chaining (goal-driven) to automate satisfaction of prerequisites and forward chaining (event-driven) to automate fulfillment of consequences. The chaining of rules corresponds to the sequence (execution order) of workflow steps and is purely dynamic because it depends on user selection (of the entry rule) and the state of the data (that are evaluated in rule conditions and effects). The Oz workflow model is therefore much more powerful and flexible.

Oz supports object-oriented data definition (schema) and query languages. A class specifies one or more super-classes, primitive attributes (integers, strings, time stamps, etc.), file attributes (path names to files in an intentionally opaque "hidden file system"), composite attributes in an aggregation hierarchy, and reference attributes allowing arbitrary 1-to-N relations among objects. Commercial off-the shelf (COTS) tools and other external programs are interfaced to Oz through shell script envelopes[5]. A return code from the envelope determines which of the several rule effects is asserted.

Oz employs a client/server architecture. Clients provide the user interface and invoke the envelopes. Servers context-switch among multiple clients, and include the workflow engine, object management, and transaction management for concurrency control and recovery. An Oz environment consists of one or more servers (sites), each with its own workflow model, data schema, object base and tools. Clients are always connected to one "local" server, and may also open and close connections to "remote" servers. Servers communicate among themselves to coordinate workflow steps that involve data and/or users from multiple sites.

DESIGN AND IMPLEMENTATION

We studied some manual care plans, Physical Therapy and Occupational Therapy for open knee surgery, and a care plan for high cholesterol treatment, that are actually used at CPMC. We then developed a prototype system, based on Oz, to automate these care plans. OzCare has been tested in simulated environments, but not in actual medical trials. The principles and techniques behind this prototype are general and can be applied to manage many other care plans.

From the workflow modeling perspective, since a care plan specifies a set of care elements (workflow steps), it is tempting to represent care plans with an Oz workflow model so that each care element is one or more Oz rules. But there are many different care plans and most importantly, every patient may use a care plan that is a variant of the standard one. Mapping a care plan to an Oz workflow model would mean one workflow for each care plan for each patient, resulting in a (potentially) enormous and extremely dynamic rule base. Since Oz workflow models are normally long-lived and relatively static, this approach is not practical.

Instead, we first defined a general data model that captures the operational environment of care plans. Under this model, a care plan is composed of a set of care elements, where each care element is an activity that needs to be carried out by the system; see figure 1. A standard care plan is a "template" object that is instantiated (copied) and stored as first class object for each admitted patient. A patient specific care plan is the result of modifications (by using OzCare add or delete command to add or delete care elements) to the instantiated standard care plan.

```
CARE_ELEMENT :: superclass
                CLINICAL_EVENT;
 care_giver: set_of link HEALTHWORKER;
 MRN : string;
 start_time : time;
 stop_time : time;
 state : string;
 pre_mlm : MLM;
 activity : string;
 post_mlm : MLM;
 ...
end

PATIENT_CARE_PLAN :: superclass
                     ENTITY;
 name : string;
 care_element : set_of CARE_ELEMENT;
 start_date : time;
 ...
end
```

Figure 1: Definition of Care Element and Care Plan

Once the data model was in place, we identified the set of general policies and workflow management functions that control the execution of the care elements. These were then modeled as Oz rules. Figure 2 is an example of such a rule. *Report* is invoked when a care giver reports the completion of a care element. Note that the rule activity, *report_element*, is implemented as a tool envelope. Within this envelope, the MLM associated with this care element is invoked to report the completion of the task.

```
# name and parameters
Report [?care_element : CARE_ELEMENT,
 ?report : LITERAL, ?vital_signs : LITERAL]:
 # binding
 (and (exists MLM ?mlm suchthat
    (member [?care_element.post_mlm ?mlm]))
    (forall HEALTHWORKER ?worker suchthat
```

```
    (linkto [?care_element.care_giver
             ?worker]))
  (exists PATIENT ?patient suchthat
    (?patient.MRN = ?care_element.MRN))):
 # condition
 (and (?care_element.status <> Reported)
      (?care_element.status = checkout))
 # activity
 {CarePlan_tool report_element
     ?care_element.report ?report
     ?care_element.Name ?care_element.MRN
     ?mlm.name ?mlm.input ?vital_signs
   return ?report_status ?mlm_output}
 # effects
 (and (?care_element.update_time=CurrentTime)
      (?mlm.output = ?mlm_output)
      (link [?patient.track_record
             ?care_element])
      (link [?worker.track_record
             ?care_element]));
```

Figure 2: The Report Rule

It is important to see that *Report* can be applied to all care elements since it takes a care element object as its input argument. These execution rules (e.g., *Report*) are used by the Oz workflow engine to interpret and execute the care elements that are stored as first class objects. Modifications to care plans can therefore be achieved by simply changing objects in the object base (at anytime) instead of redefining the work flow model (the rules).

Besides the data model and workflow model, there are some other care plan-specific system requirements that we had to address. The first is the notion of timed event -- an event has to start and end at or by certain times. We defined in the data model an *EVENT_QUEUE* and an *ALARM_QUEUE*. Care elements are put into the *EVENT_QUEUE* based on their *start_time*, and *ALARM_QUEUE* based on their *stop_time*. Care elements in the *EVENT_QUEUE* are those that have not been "checked out" by the responsible care givers, whereas care elements in the *ALARM_QUEUE* are those that have not been "reported" yet. A UNIX timer program runs once every day to generate a "to-do" list for care givers based on the care elements in the *EVENT_QUEUE*. Another UNIX timer program runs once every hour to send reminders to the care givers for their care elements in the *EVENT_QUEUE*, and "alarms" (e.g., beeper messages) for their care elements in the *ALARM_QUEUE*. These time periods were only selected for demonstration purposes. Significantly shorter intervals would be needed in practice.

A critical requirement for OzCare is that it must interface with the CPMC MLM server because the clinical events associated with individual care elements are normally implemented as MLMs. The communication between OzCare and the MLM server can be either synchronous or asynchronous. Figure 3 depicts the architecture of the OzCare system.

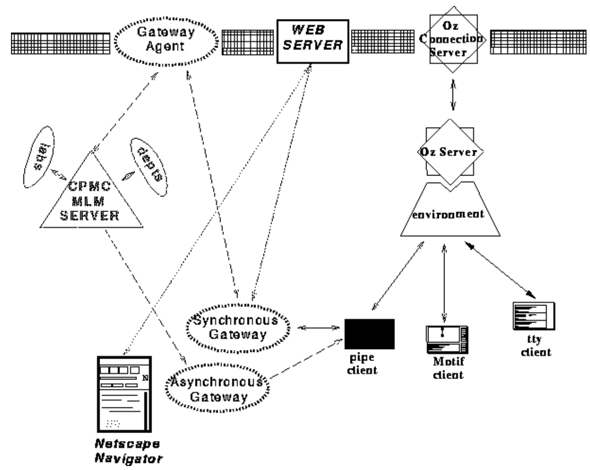

Figure 3: OzCare System Architecture

In the synchronous case, an MLM is called as a result of executing the activity of an Oz rule, e.g., *Report*. This kind of MLM normally triggers a clinical event or reports the completion of a clinical event. A gateway agent runs at a CPMC workstation and communicates with remote gateways through sockets. When an Oz rule activity invokes an MLM, the regular Oz client runs the synchronous gateway, which in turn sends the "run MLM" request (e.g., "run *new_rx*"), to the gateway agent in CPMC. The synchronous gateway then waits for MLM output. When the gateway agent gets the request, it invokes a "runmlm" command to execute the MLM in the CPMC MLM server, and sends the output back to the synchronous gateway of Oz. On receiving the MLM output, the synchronous gateway invokes the appropriate Oz rules (by sending the requests to the Oz client, which in turn communicates with the Oz server) according to the actions specified in the output. For example, an MLM output "delete:maintenance_rx:313:delay 24 hours", will result in an Oz query to find the care element "maintenance_rx" that belongs to the patient whose MRN value is 313, and is scheduled 24 hours from now. Once such an instance is found, the Oz rule *Delete* is invoked.

Asynchronous communication is also needed because there always can be "unexpected" clinical events that may cause the care plan to be altered. For example, a just-reported lab result may indicate that the patient is no longer suitable for the current care plan. Since such clinical events are implemented using MLMs, OzCare needs to "listen" to the MLM outputs that are of interest. Each MLM can have a list of email recipients (e.g., doctor@hospital) so that the MLM output is sent to the physicians in charge. For OzCare, a dedicated email account "ozmlm" can be set up, and its email address can be registered as one of the email recipients for certain MLMs. Further, an asynchronous gateway is used to process the incoming emails sent to "ozmlm".

Another important system requirement for care plan management, and in fact for healthcare applications in general, is that the system be integrated into the World Wide Web[2]. WWW is used for easy access to information sites and for collaboration among institutions. In OzCare system, users interact via the standard forms mediums using a regular Web browser. A set of scripts enable the Web server to interpret user inputs and to send appropriate object base queries and rule invocations to the Oz client. For performance considerations, the Oz client uses a pair of pipes (UNIX FIFO inter-process communication streams) for input and output.

DISCUSSION

The unique contribution of our work is that we have successful "separation of concerns", with MLMs implementing the low level medical knowledge intensive clinical events and decisions, and Oz implementing the high level operational workflow management.

A number of weaknesses of Oz also come to light in our implementation of OzCare. The most serious one is the lack of timing support. In the workflow modeling language, there is no way to specify that a rule, for example *check_alarm_queue,* be run at a frequency, for example, every 5 minutes. The workflow engine does not have a built-in timing control either. Thus in order to schedule a rule execution at certain time, a (separate) UNIX process has to be set up to send the request to Oz at the specified time.

FUTURE WORK

The next steps would be to first add timing support to Oz. Also needed is the support of viewing and analysis of workflow progress. An interesting experiment would be to apply Oz's workflow interoperability and geographical distribution features to support healthcare networks across multiple hospitals, clinics, etc., with perhaps different care plans (procedures and protocols), to collaboratively provide the highest quality patient care at the lowest cost.

Acknowledgments

This work is supported in part by the New York State Science and Technology Foundation Center for Advanced Technology in High Performance Computing and Communications in Healthcare. Steve Luxenberg, James J. Cimino, George Hripcsak, Robert Jenders, and Justin Starren at CPMC provided crucial direction and input to this research. We would also like to thank Steve Linde, Peter Skopp, and Issy Ben-Shaul for their work on the CGI-bins and alternative Oz interfaces for OzCare, and Jeff Do and Richard Baldwin for their helps with demo screens.

References

1. Isreal Ben-Shaul and Gail E. Kaiser. *A Paradigm for Decentralized Process Modeling*. Kluwer Academic Publishers, Boston, 1995.
2. James J. Cimino, Socrates A. Socratous, and Paul D. Clayton. Internet as clinical information system: Application development using the world wide web. *Journal of the American Medical Informatics Association*, 2(5):273-284, 1995.
3. Paul D. Clayton and George Hripcsak. Decision support in healthcare. *International Journal of Bio-Medical Computing*, 39:59-66, 1995.
4. Paul D. Clayton, Robert Sideli, and Soumitra Sengupta. Open architecture and integrated information at Columbia-Presbyterian Medical Center. *Clinical Computing*, 9(5):297-303, 1992.
5. Mark A. Gisi and Gail E. Kaiser. Extending a tool integration language. In Mark Dowson, editor, *1st International Conference on the Software Process: Manufacturing Complex Systems*, pages 218-227, Redondo Beach CA, October 1991. IEEE Computer Society Press.
6. ASTM Subcommittee E31.15 on Medical Knowledge Representation. Standard specification for definition and sharing modular health knowledge base (Arden Syntax for Medical Logic Systems). Pages 1460-92, April 1992.