Applications of Web-Based Workflow

Chuck Ames, Scott Burleigh, Steve Mitchell, Tom Huynh
Jet Propulsion Laboratory, California Institute of Technology
{chuck,scott,steve,tom}@tel.jpl.nasa.gov

Abstract

A web-based workflow system was developed at the Jet Propulsion Laboratory and several pilot applications were developed. We give an overview of the architecture and functionality of the workflow system, describe three pilot workflows that have been or are being deployed, and relate lessons learned.

1. Introduction

Workflow Management is becoming an increasingly important part of organizational information systems. A Web-based workflow management system called WWWorkflow was developed at the Jet Propulsion Laboratory (JPL) [1]. WWWorkflow was designed to provide a general-purpose process mediation service as an integral part of an organizational "Intranet". WWWorkflow's user interface consists of dynamically generated Web pages, email messages, and pager notifications, so no unique desktop applications are required in order for users to interact with the system.

A key design principle embodied in the WWWorkflow system is the careful separation of process mediation from product data management. This has been shown to be a useful distinction in other contexts [2]. Instead of building data management functions (e.g. forms processing, data vaulting, etc.) into WWWorkflow, we define interfaces to external systems that provide these functions. Thus WWWorkflow can be used to add workflow functionality to pre-existing data management systems.

WWWorkflow has been used to implement and deploy several workflow management applications at JPL and at the NASA Johnson Space Center (JSC). These applications vary in procedural complexity, size and distribution of user community, and volume of transactions. We learned much about process engineering and workflow implementation in general, and about the validity of the design of WWWorkflow in particular.

This paper is organized as follows: an overview of the architecture and design of the WWWorkflow system is given. Next, three case studies of specific applications of the WWWorkflow system are presented. We conclude with a discussion of lessons learned from these three applications.

2. WWWorkflow System Overview

The purpose of Workflow Management systems is to coordinate activities of people. Many existing Workflow Management systems provide very good support for process engineering but limited support for process execution, often requiring additional proprietary desktop applications which may have limited platform availability. Unfortunately, process engineering cannot be effective unless the results are accessible to the people who must carry out the processes. WWWorkflow provides a process mediation service using tools that most users already have and know how to use. Users interact with the system using their Web browser to check their "TODO" list and read Task Descriptions about assignments. In its "nag daemon" role, WWWorkflow uses email and pagers to notify people when they need to do something.

2.1 System Architecture

The WWWorkflow system architecture contains three main parts:

1. A Common Gateway Interface (CGI) [3] front end which allows users to interact with the system via a browser.
2. A workflow "engine" which sends and receives messages into and out of the workflow system.
3. A database which contains predefined workflow procedures, status information for procedures underway, and complete histories of previously executed procedures.

The user interface to the system is the user's web browser software. The web browser uses the Hyper Text Transfer Protocol (HTTP) [4] to send requests and receive documents from an HTTP daemon process. The HTTP daemon helps the user interact with the workflow system by calling the CGI program responsible for WWWorkflow activities. The CGI program communicates both with the workflow engine and with the workflow database to retrieve instructions for the user and to notify the workflow engine of activities the user has performed.

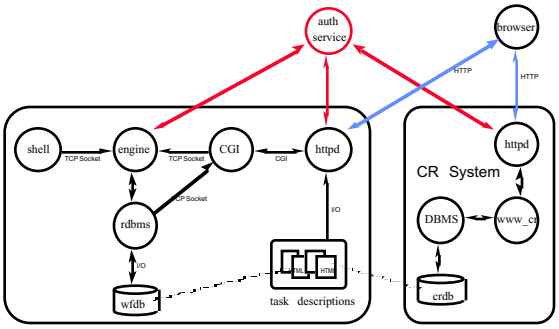

Figure 1: WWWorkflow system architecture

Figure 1 shows the system architecture as deployed in the Process Change Request system described in section 3.3. The system drives a document change request process used to approve changes to Space Station documentation. Shown in the diagram are both the workflow system and the document change request (CR) system. While WWWorkflow is used to embody and execute change request procedures, the system is carefully separated from the product data within the CR system itself. Also shown is the optional communication with existing enterprise authentication services. The architecture of WWWorkflow is consistent with the recommendations of the Workflow Management Coalition Reference Model [5].

2.2 Workflow Language 2.0

Workflow Language (WFL) is the low-level language in which commands are written to control the operation of the Workflow Daemon (server). In this respect the relationship between WFL and the workflow server is functionally analogous to the relationship between SQL and a relational database management server. That is, a procedure definition is not written in WFL, exactly; what is written in WFL is a set of instructions to the workflow server that enable it to record the procedure definition in the workflow database. Other WFL commands are used to instantiate procedure definitions, to note the completion of procedure steps (and thereby trigger activation of contingent procedure steps), and to edit existing procedures and procedure definitions.

Every procedure definition comprises one or more *steps*. Each step has an associated *precondition*, some statement that must be true before the step can begin (in any procedure instantiated from this definition); the precondition of a step is usually expressed in terms of the completion statuses of other steps. Each procedure may also have any number of arbitrarily declared *operands* whose values constitute the "memory" of the procedure, similar in function to the local variables of a computer program.

When a procedure definition is instantiated, i.e., a specific **initiative** governed by that definition is begun, the initiative is essentially a replica of the procedure definition; it has all the same steps and operands. Those steps of the initiative whose precondition is the predefined value *START*, or some Boolean expression that is True when *START* is true, are automatically begun as soon as the initiative has been recorded in the workflow database.

Steps may be either simple or compound in nature. A simple step is one whose initiation is expressed as a single *task* assigned to a single person or a single automated task (e.g., the execution of a script); a compound step is one whose initiation is expressed as multiple tasks, such as the assignment of the same work item to each member of a team. When a task is completed, the assignee uses WFL (usually by triggering a CGI script from a Web page) to notify the workflow daemon of the task's completion status. The daemon uses this information to update the workflow database and automatically initiate all steps whose precondition is true now that the affected step has been completed in the status indicated. The completion of a task may in fact re-initiate a step that was previously completed; that is, the daemon may trigger multiple *iterations* of the same step if required by the procedure definition.

WFL is a dialect of Tcl, the Tool Control Language [6] devised by John K. Ousterhout at the University of California at Berkeley. The WFL extension to Tcl comprises just six commands: **new, edit, delete, start, stop,** and **get**. Subsets of these commands can be applied to six general types of

workflow database objects: **procedures**(which are actually procedure definitions), **initiatives** (procedures instantiated from those definitions), **operands**, **steps**, **tasks**, and **statuses**.

Associated with each type of object are data attributes, such things as name, value (in the case of an operand), assignee and task description (in the case of a step), etc. The values of several of the attributes of a step may be "late bound", that is, they may explicitly be Tcl scripts but implicitly be the values returned by evaluating those scripts at the time the step is begun. The value of the "assignee" attribute of a step may be explicitly or implicitly a list of assignee names, making the step compound rather than simple. Built-in Tcl variables whose values are updated automatically in context-sensitive fashion are provided to simplify the writing of WFL scripts.

The **start** command is used only to start a new instance of a procedure definition. The **stop** command is used principally to note the completion status of a task. The **get** command simply returns the current value of an operand (which, like any other attribute, can be set by means of the **edit** command). Step preconditions can be arbitrarily complex Boolean expressions concerning the completion statuses of other steps.

Here is an example of a small WFL script that creates a simple procedure definition:

```
new procedure add_user
   do {new operand $Pr $In loginId}
   do {new operand $Pr $In userName}
   do {new step $Pr $In add_acct
   assignee Marla
   condition START
   }
do {new step $Pr $In tell_user
   assignee Darryl
   condition {status .add_acct Done}
   }
```

A complete discussion of WFL syntax and functionality is available in [7].

2.3 WWWorkflow User Interface

The critical component necessary to put WWWorkflow to work is the Web Gateway to the workflow engine and database. WWWorkflow uses Common Gateway Interface (CGI) technology to provide an interface to the users. The modules are written in Perl. It has been shown that CGI programs can provide effective user interfaces to complex systems. [8].

To interact with the system, the user directs a web browser to access the Uniform Resource Locator (URL) of the CGI program that will act as the user's interface to the organization's workflow. The CGI program interrogates the user for authentication and identification information. A main menu is presented to the user that contains options to interact with the user's current "todo" list, start previously defined procedures, and view the status or historical information of procedures. Todo lists have been show to be an effective way of communicating assignments to distributed workgroups [9].

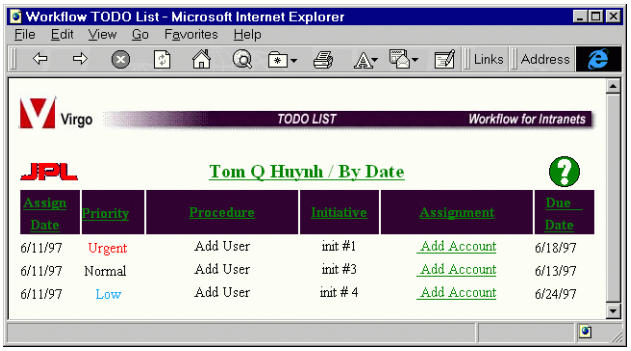

Figure 2: TODO list display

The user's todo (figure 2) list is the result of a query to the workflow database. The query finds all active instances of steps within active procedures for which the user is currently responsible.

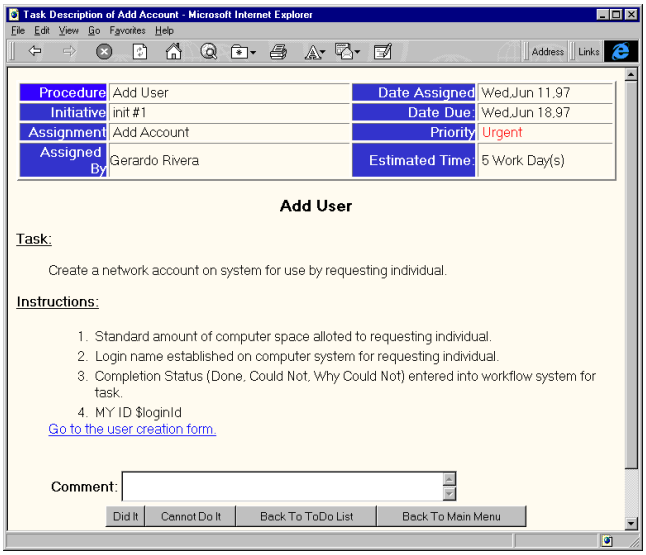

Figure 3: task description display

Once a user selects a task from the todo list, the user is presented with a task description of the step (figure 3). The task description is defined and documented with HTML files. These use hypertext links to help users begin working on a certain task. Files in the collection of task descriptions for a process may also serve as documentation or on-line procedure manuals.

At the bottom of each task description the user is shown one button for each of the prescribed completion statuses for the task at hand. The user selects one of these completion statuses when he has completed the task. The selection of a completion status button will cause the CGI to send a completion command to the workflow system for that particular task. This is one of the ways the workflow system is notified of user participation, and helps drive the workflow.

3. Case Studies

This section describes three cases where WWWorkflow has been successfully applied. All three cases required that WWWorkflow be integrated with other pre-existing data management systems. For each case, we describe the problem domain, the workflow procedure that was designed, and the resulting system architecture with reference to external data management systems that were integrated.

3.1 Case 1: Process Action Item

Small teams often operate by establishing an action item list. Action items are assigned by the team leader, carried out by team members, then closed by negotiation between team member and team leader.

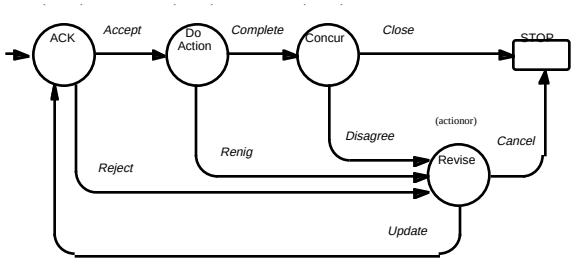

Figure 4: The "process action item" procedure

A WWWorkflow-based system was designed to establish team action item lists and track progress of individual action items [10]. WWWorkflow was integrated with an external action item database which provides forms for creating and editing action items, as well as generating summary reports. When a new action item is created, a new initiative of the Process Action Item (PAI) workflow procedure (figure 4) is started within WWWorkflow. WWWorkflow then initiates work and tracks progress according to this initiative. The initiative is complete when the Action Item is either closed or canceled by the Action Item list owner.

The PAI procedure is relatively simple, composed of 4 possible steps, with 8 possible paths. Each initiative is short lived, lasting from a few days to a few weeks. The Process Action Item application involved a small user community of 12-15 users distributed over a single site.

3.2 Case 2: Develop Command Sequence

JPL builds and operates unmanned planetary spacecraft. Part of the operations process is design and preparation of command sequences to be sent to spacecraft in flight. The application of the following command sequence workflow involved a medium-sized user community of 30-150 users distributed over multiple sites. The procedure is complex, involving 73 possible steps, with a high degree of parallelism. Each initiative is long lived—roughly 56 days.

The process of developing a sequence of commands to send to an unmanned planetary spacecraft has always been time-consuming and labor-intensive. Historically, most or all of the responsibility for ensuring that command errors don't damage the spacecraft or jeopardize its mission has rested with the mission operations teams rather than with the on-board software charged with executing the commands, due to severely constrained on-board computing capacity. The spacecraft are complex and expensive; they respond to hundreds of different commands that configure many different subsystems, managed by different organizations, often with conflicting operational goals; and recovering from some types of error may be difficult or physically impossible. Consequently command sequencing has always involved many stages of conference, negotiation, analysis, modeling, review, and approval.

Sequence planning for the Cassini mission to Saturn is a particularly demanding, yet critical, process because Cassini is the largest and most complex unmanned spacecraft ever built. WWWorkflow was used to characterize the Cassini

sequence planning process with an eye toward minimizing the delay between the completion of any step and initiation of the contingent steps, to give all participants as much productive time as possible within the 56-day window allotted to the planning of each sequence.

The procedure can be divided into three chained subprocedures: activity planning, sub-sequence generation, and sequence integration and validation. Each subprocedure requires that the work done by mission operations team members be smoothly integrated with automated work, such as the execution of automatic conflict detection and resolution tools and the archiving of team decisions in a mission database. WWWorkflow's built-in notion of parity between "automated" steps and steps assigned to people minimized the difficulty of this integration.

The activity planning subprocedure is relatively straightforward, with two provisos. First, determination of the availability of files needed to support the automated steps can proceed in parallel with the review of the Phase Update Package (PUP). Second, both review steps must be performed by all members of the Sequence Virtual Team, i.e., they are compound steps.

The sub-sequence generation sub-procedure is conceptually very simple, but implementing it using an earlier version of WFL was tedious. Support for compound steps in WFL 2.0 greatly simplifies the sub-procedure by collapsing many nearly identical sub-steps into a single compound step with assignee-specific variations (see Figure 6 for an illustration of this general model). The compound step reduces sub-sequence generation to only five steps, one of which is assigned with variations to each of the subsystem teams.

Sequence integration and validation is the most complex and time-consuming of the sub-procedures, but it too is simplified by applying a new feature in WFL 2.0: built-in support for iteration enables us to include the steps entailed in uplink package approval just once in the workflow but let them be executed twice, or even more often if time permits. Figure 5 illustrates this kind of simplification.

3.3 Case 3: Process Change Request

The Johnson Space Center (JSC) is leading development of the International Space Station, and Station Operations will be conducted from JSC. Development of Operations Procedures and other Operations-oriented documentation has begun with contributions from the entire Space Station development community. These contributions and proposed changes flow through a change request process.

The change request originates when contributions or modifications to documents are identified by authors and document reviewers using the document library system. The change request workflow is started through the document library with a CGI form interface. This interface appears seamless to the user of the document library—they see it as a part of the library system. The CGI interface is passed the appropriate document identification information from the library system, along with the changes requested or new contributions to be added. The CGI then starts a new change request workflow. Given the document identification, the newly initiated workflow can retrieve documents from the document library or aid users in retrieving documents.

The document change requests are routed through several review steps, each with potential outcomes such as "approve with modifications" and "disapprove." The workflow potentially iterates over review steps several times as modifications are made and re-reviewed.

In addition to several review/approve cycles, the workflow must coordinate the roles of reviewers and composition of teams constructed to review changes. For example, depending on the type of document to be changed, the teams constructed to review the document may differ from change request to change request. Additionally, several roles in the change request procedure (such as review board chair, or review manager) may need to be determined from the results of ad hoc database queries or input from users.

This application involves a large user community of more than 1,000 users distributed around the globe. The procedure is complex, involving approximately 60 steps. Each initiative lasts from one week to one month. This application involves interfaces to 3 external data management systems, including a relational database, a document vault, and an organizational model.

4. Observations and Lessons Learned

In this section, we relate specific insights achieved and solutions to problems found while implementing the applications described in the case studies in section 3. As a result of these experiences, several improvements were made to the workflow language, the workflow server, and the user interface. Also,

certain assumptions which drove the design of WWWorkflow were reinforced.

4.1 Iterations

Early in the development of the Process Action Item (PAI) workflow (section 3.1) it was clear that the execution of one or more steps within a procedure might need to be repeated several times. Specifically, the PAI workflow required that a collection of steps repeat themselves infinitely.

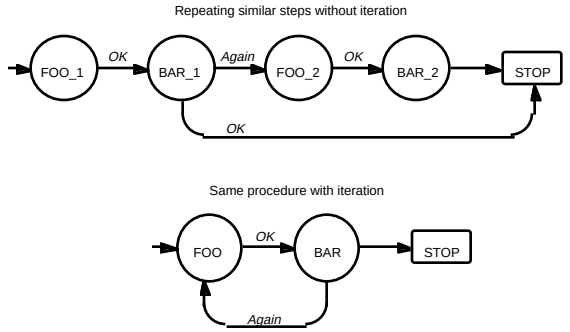

Figure 5: Introduction of "iteration" to WFL

This concept of "looping" was not originally supported in the workflow language design—once a step completed, it was finished forever. In real world business and engineering processes however, we see that a step must often be repeated until it is done correctly, or that iterations over a certain activity may be the desired flow of work. For this reason workflow language was modified to support looping by enabling steps to be re-initiated after they have previously completed. That is, the daemon may trigger multiple iterations of the same step if required by the procedure definition. See section 2.2.

4.2 Late Binding

The Develop Command Sequence (DCS, section 3.2) workflow showed a need for a late binding mechanism. Late binding allows the author of a workflow to defer the assignment of values of certain workflow attributes until the workflow is underway. For example, the DCS workflow introduced a concept called virtual teams—dynamic lists of roles, each role filled by individual workers when the sequence is underway, but are not known at the time the workflow is authored or even when the workflow is initially started. Use of these virtual teams allows the organization responsible for the uplink sequence to be more flexible in staffing. Additionally, this concept allows teams to change composition depending on the requirements of a given sequence. For example, an uplink sequence is the responsibility of a Sequence Virtual Team Leader (SVTL). Many of the steps in the uplink process are this individual's responsibility. However, the SVTL is not identified until the workflow is underway, and may change during the execution of the workflow. While the original workflow language design allowed procedures underway to be modified ad hoc, the coding was tedious and difficult to debug. The language was modified to allow the assignee (and other attributes) of a step to be directly determined from the evaluation of arbitrarily complex embedded Tcl commands at the instant that the attribute is being used. This late binding mechanism allows tremendous flexibility of workflow authoring and execution.

4.3 Compound Steps

Another lesson learned from both the DCS workflow project and the PAI process was the idea of compound steps (figure 5) and better support for the execution of sub-procedures. For example, when an uplink sequence is started, the SVT is composed of the SVTL and several Subsystem Engineers (SE).

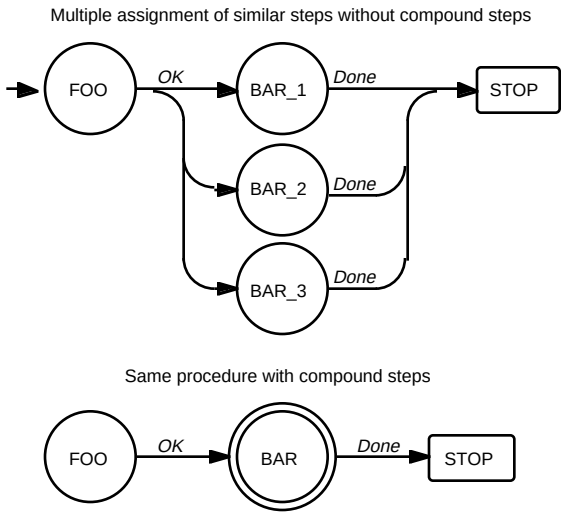

Figure 6: Introduction of compund steps to WFL

The list of SEs in the SVT depends on the subsystems that might be involved in the sequence. Difficulty in coding the workflow arose when sub-procedures (collections of steps) were assigned to

SEs. First, the list of subsystems involved in the sequence was not known until the workflow was underway and may change during the execution of the workflow. Second, the precondition on certain steps was the conjunction of the outcome of several sub-procedures—each assigned to a SE. Again, while the original workflow language design allowed a workflow underway to be modified ad hoc, the coding was tedious and difficult to debug. The language was modified to support compound steps by providing support for steps whose initiation is expressed as multiple tasks, such as the assignment of the same work item to each member of a team. See section 2.2. Compound steps, while arbitrarily complex, assert a single completion status—suitable for use as a term in the precondition of a following step. Combined with late binding, compound steps allow assignment of complex subprocedures to one or more individuals or virtual teams while preserving the logical flow of the workflow.

4.4 Integrating WWWorkflow with External Data Systems

WWWorkflow's architecture was designed to leverage a base of pre-existing information services within an organization. The system serves to not only remind people to perform certain tasks, but also to interact with external data systems.

Several points exist during the execution of a procedure in which the workflow system may need to use or modify information residing in external data systems. This interaction with external data systems may be required to aid the user in performing a certain task (for example, retrieving a database record or document), or the procedure may have been designed to interact with external data systems without user involvement.

To aid the user in the use of external data systems, the workflow system leverages its web based nature--specifically its web interface (see section 2.3). The user is aided in interaction with external data systems by carrying state information out of the workflow system and passing it on to external resources. Since the user interacts with WWWorkflow via a web browser, HTML documents are served up via an intermediate CGI gateway program which scans text for special 'variables' and replaces them with the values of certain operands for the currently active initiative.

For example, a workflow procedure to add a new employee started with a supplied operand of 'name' given a value of 'John Smith', the HTML document for a given step might contain the embedded string:

"Add $$name to the department phone list."

The text would then show up on the user's workspace frame as:

"Add John Smith to the department phone list."

In order to enable WWWorkflow itself to interact with external data systems, the workflow server was modified. An early design decision to use a dialect of Tcl for the workflow language and to embed a Tcl interpreter in the workflow server paid off when the need for interaction with external data systems was realized. By using Tcl 7.6 (which supports dynamic loading of language extensions) and loading Tcl extensions into the workflow server's Tcl interpreter, the server is easily extensible. For example, the start attribute of a procedure may be a Tcl script that uses an Oracle RDBMS Tcl extension to interact directly with a database when the procedure is started. Another example would be to write an assignee attribute of a step as a script that uses an X.500 Tcl extension to retrieve information from the enterprise's directory service.

4.5 Computer programs as a model for workflows

Workflow procedures, like computer programs, are plans; the principal difference is that workflow procedures are typically executed by people rather than by computers. Taking this analogy seriously enabled us to look for deficiencies in the design of the workflow database, which is where the representation of a procedure definition ultimately resides in this system: because programs are similar to workflows, we would expect the features of a good programming language to map into the design of the workflow database. The fact that complete computer programming languages need looping constructs alerted us to the need to enable iteration of steps; the usefulness of block structures in programming languages suggested the need for something analogous in the workflow database design -- support for sub-initiatives. This perspective will very likely continue to inform the ongoing development of WWWorkflow.

5. Summary

This paper has described applications of a workflow management system deployed with a World Wide Web interface. The architecture of the WWWorkflow system was presented, followed by a description of the latest version of the workflow language. Three case studies were presented; an action item processing procedure that helps team leaders coordinate the activities of team members, a spacecraft uplink sequence development procedure that supports the design and preparation of command sequences to be sent to spacecraft in flight, and finally a document change request processing system that manages contributions and proposed changes to space station procedures and other operations-oriented documentation. These case studies detail the respective problem domains, describe the procedure, and present a view of the system architecture in the context with other data systems. Finally, observations and lessons learned from each of the case studies were presented:

1. Procedures often require that steps or subprocesses iterate. Real world business or engineering processes often require that the same type of activity occur many times. This was revealed by the Process Action Item and Process Change Request projects.
2. Often much of the information needed to perform or even start a task is not known until the procedure is well underway. The need to use a late binding mechanism of procedure or step attributes to arbitrary commands was discovered when developing all three of the cases described in this paper.
3. Procedures often contain subprocedures or logical collections of steps which may need to be instantiated or completed singularly and in iterations. Development of the Process Action Item and Develop Command Sequence workflows were made more practical with the addition of support for compound steps and subprocedures in the workflow language.
4. Interfaces to external data systems need to be widely supported throughout the system architecture. For example, not only does the user need to be able to link to a database application from their task description screen, but the workflow server itself might need to communicate with the database in order to decide how the workflow should proceed or who might need to be assigned to a certain task. This was discovered while developing all three of the cases described in this paper.

6. Future Work

Work is currently underway to develop a graphical workflow authoring tool for the design and creation of WWWorkflow procedures. The tool is written entirely in the Java programming language and is intended to be used as an applet in a WWWorkflow user's web browser. The tool shares much of the Workflow Language 2.0 and WWWorkflow database object models—extending the objects to include attributes and methods for the graphical representation and manipulation of procedures.

Design is underway for a graphical workflow monitor. Likely to be based on the components being developed for the graphical editor, the monitor will allow users to visually observe the state and progress of procedures underway. As steps are completed and preconditions for following steps are met, users will be able to view the changing state of the workflow being monitored.

More support for integration with enterprise services is being developed. In addition to database services, current projects have revealed the need to communicate with such services as X.500 directories, DCE authentication and directory services, Kerberos authentication services, and potentially calendar and scheduling systems. Such support is being integrated with extensions to Perl CGI programs within the system and Tcl extensions to the workflow server language.

7. Acknowledgments

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

8. Availability

The URL for the documentation of the system is [http://workflow.jpl.nasa.gov]. Copies of the system may be obtained by agreement with the California Institute of Technology.

9. References

[1] Ames, C.; Burleigh, S.; Mitchell, S.; "WWWorkflow: World Wide Web based Workflow", in *Proceedings 30th Hawaii International Conference on System Sciences*, IEEE Computer Society, January, 1997

[2] F. Leymann, D. Roller, "Workflow-based Applications" IBM Systems Journal, Vol 36. No 1, 1997.

[3] "The Common Gateway Interface" [http://hoohoo.ncsa.uiuc.edu/cgi].

[4] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol—HTTP/1.0," RFC 1945, May 1996.

[5] Hollingsworth, D.; "The Workflow Reference Model", Workflow Management Coalition WFMC-TC-1003, 29-Nov-94, Version 1.1

[6] J. Ousterhout, *TCL and the Tk Toolkit*, Addison Wesley, 1994.

[7] S. Burleigh, "Work Flow Language User's Guide," [http://workflow/doc/wfl_guide/], February 1996.

[8] J. Rice, A. Farquhar, P. Piernot, & T. Gruber, "Lessons Learned Using the Web as an Application Interface," Knowledge Systems Laboratory, KSL-95-69, September 1995.

[9] Thomas Kreifelts, Elke Hinrichs, Gerd Wetzel, "Sharing ToDo Lists with a Distributed Task Manager" *Proceedings of the Third European Conference on Computer-Supported Cooperative Work,* 13-17 September, 1993, Milan, Italy

[10] Ames, C.; "Internet Action Item Tracker (InterAction) Project Page", [http://workflow.jpl.nasa.gov/virgo.html], May 1997.