IMPLEMENTING ORACLE WORKFLOW

Derek Mathieson
Developer
CERN - European Laboratory for Particle Physics, Geneva, Switzerland

Summary

CERN (see [CERN]) is the world's largest physics research centre. Currently there are around 5,000 people working at the CERN site located on the border of France and Switzerland near Geneva along with another 4,000 working remotely at institutes situated all around the globe. CERN is currently working on the construction of our newest scientific instrument called the Large Hadron Collider (LHC); the construction alone of this 27-kilometre particle accelerator will not complete until 2005. Like many businesses in the current economic climate CERN is expected to continue growing, yet staff numbers are planned to fall in the coming years. In essence, do *more* with *less*. In an environment such as this, it is critical that the administration is as efficient as possible. One of the ways that administrative procedures are streamlined is by the use of an organisation-wide workflow system.

Electronic Documents

The CERN workflow system (named EDH for Electronic Document Handling) is the most heavily used in-house administrative application at CERN. We have around 4,500 active users and process over 120,000 *work-items* every year. A *work-item* is the term used by the Workflow Management Coalition (see [WFMC96]); this corresponds to an EDH *document* within our application. EDH currently supports thirteen documents such as Internal Purchase Orders, Claims for Overtime, or Requests for Annual Leave.

Creating Documents

Documents are created either with our old Client-Server application (see [DEJONGHE93]) or via our new web-based interface, for example Figure 1 shows a typical Internal Purchase Order (Demande Achat Interne or DAI in French) being created.

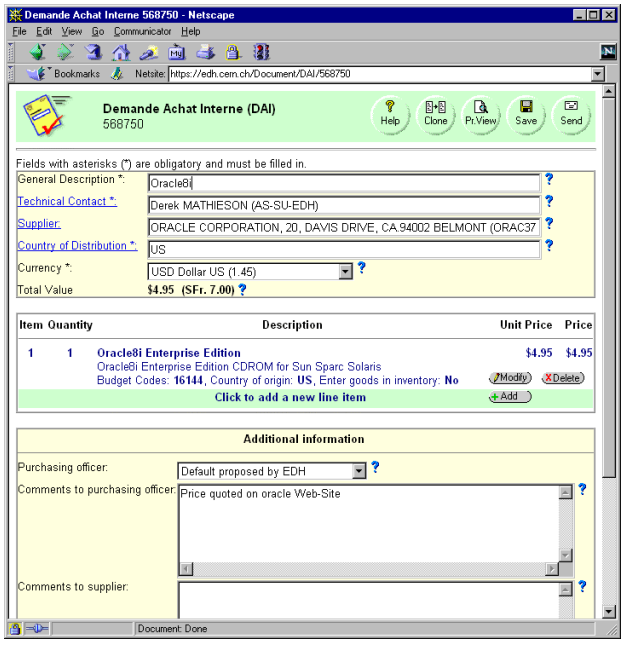

Figure 1: Creating a Purchase Order

Once a document has been created, clicking the Send button initiates the Workflow Process.

Workflow at CERN

Workflow at CERN involves the forwarding of an electronic document to the people that are required to approve it. Approval of a document involves the user entering their *authorisation password* in a specially encrypted field in the document. Our own auditors and all of the external institutes that wish to use EDH have accepted this authorisation password as equivalent to a paper signature.

One of the complexities of the situation at CERN is the enormous number of special workflows that we require. The routing of financial documents has to take into consideration not only CERN's own financial rules, but also the requirements of other institutes for whom CERN spends money. Additionally different working practices within the organisation are also supported which means that currently there are around 270 different paths that can be chosen depending on the type of document, who created it, and whose money is being spent.

Within Oracle Workflow Builder we have defined how each document should be routed. For example for a DAI the workflow process is defined as shown in Figure 2.

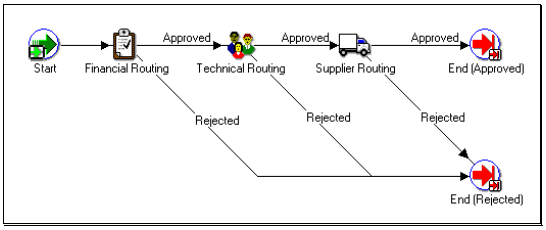

Figure 2: Purchase Order Routing

This may look simple, but each of the steps in the diagram above itself expands to another sub-process such as that shown in Figure 3 where the CERN Standard Financial Routing is defined.

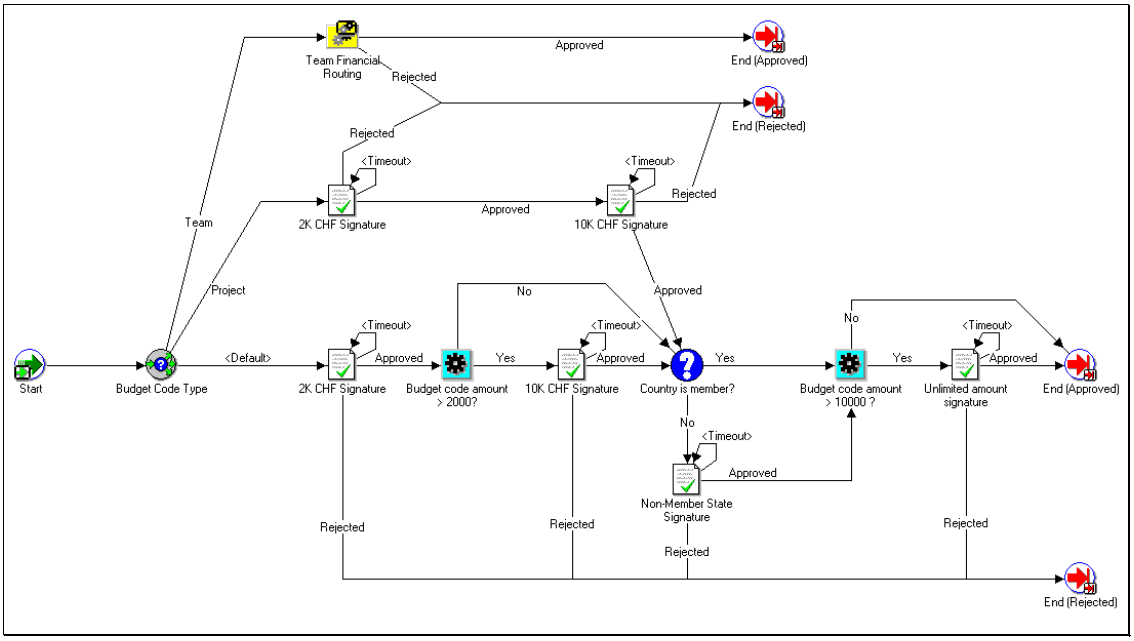

Figure 3: Standard Financial Routing

Signature Rights Database

In order that the workflow system is able to decide who has the right to sign for a particular step in the workflow process we have a special database of signature rights for every workflow action. Most of the

data in this signature rights database is derived automatically from our corporate databases. The organisation structure from our human resources database is merged with data from our financial system to obtain a database of what accounts someone can spend from and to what limit.

Figure 4 shows the basic arrangement of the organisation. CERN is divided into four *Sectors*, each of which is split into three or four *Divisions*, which are in turn split into *Groups*, which are further split into *Sections*[†].

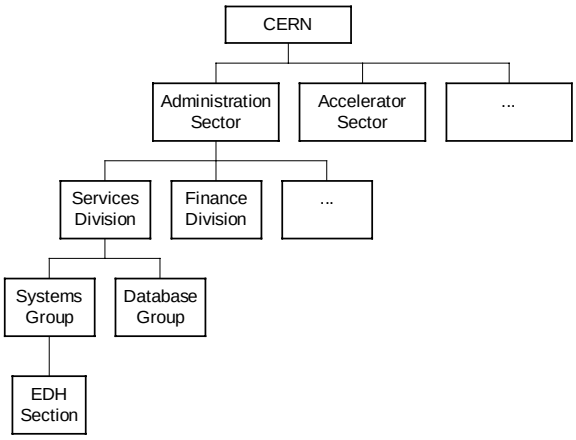

Figure 4: Simplified Structure of CERN

In the Administration Division we define that a Group Leader can spend up to 10,000 Francs on any account in their group, and that a section leader can spend up to 2,000 Francs on any account in their section. This means that either the Group Leader or the Section Leader can sign a document for 500 Francs, however the signature rights are automatically *prioritised*, such that the section leader will be selected first. The Group Leader would be asked to sign only if the Section Leader was unavailable (for example, on absent due to training, vacation, etc.).

In a typical routing definition there are many *signature* actions, for example most financial documents require at least one signature to authorise payment. These actions appear on the diagram like the example in Figure 5. When the workflow reaches this step in the routing a special stored procedure is called which determines who has the right to sign for the expenditure according to our signature right database. The procedure then uses the human resources database to select the first person that has the right to sign and is not absent.

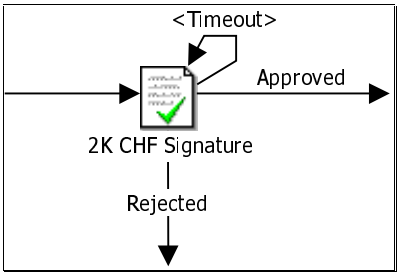

Figure 5: A typical Signature Action

Workflow Re-Engineering

Workflow is not new to CERN; in fact since 1992 we have been using a workflow system that was developed completely by our own engineers. The core system was implemented in C with the process definitions stored in an Oracle Database. During its lifetime the system has gradually grown, new special cases have been added, additions to the process definitions were made directly to the database using SQL. Finally the size and complexity of the system caused it to become virtually impossible to maintain.

[†] Some *Sub-Sections* also exist at CERN.

We investigated many commercial packages and commissioned an external consultancy firm to review the market to try to find a replacement for our existing system. Unfortunately we could find no product that could meet our particular requirements, and finally decided to start work on a new in-house development of a new workflow system.

At that time there was a general move towards the use of Java in the development of EDH software, so we decided to implement our new workflow engine in Java.

Oracle Workflow

About three months into the development of our new workflow engine Oracle Workflow was released. When we looked at the product we were very interested in its open architecture. Most of the work that we had done until then was writing Java versions of our Common Business Objects (see [CBO99]), something that would still be useful if we adopted Oracle Workflow.

Graphical Design

One feature of Oracle Workflow that really convinced us to use the product was the graphical process design tool - Oracle Builder. Graphical visualisation was one aspect that was not addressed in our original system. Over the years we realised that such a capability was very important to the success of a workflow system. Many calls to our help desk were simply asking for explanations of the routing decisions made by the workflow engine and without any visualisation tools it would frequently involve an engineer with SQL*Plus in order to find the answer.

Oracle Builder is a Microsoft Windows Application that provides a drag-and-drop environment for designing workflow processes. In Oracle Workflow terminology a workflow process is called a process activity. A process activity consists of a single start point, a series of interconnected function or process activities and one or more end points. A function activity is a step where Oracle Workflow calls a PL/SQL stored procedure and its returned value can be used to determine which branch in the workflow should be chosen. One useful property of this arrangement is that workflow processes can include other workflow processes, as a sub-workflow. This allows us to define common sub-processes that can be shared by several processes. For example, we have defined a *Standard Financial Routing* that is common to all documents involving a financial transaction.

In our current system (which is not yet complete) we have defined around 120 function activities and 80 process activities.

Error Processes

Another area in which our existing system was particularly weak was in error handling. Originally the system was built such that if, during the execution of the routing process an error occurred, for example there was no one available that could be assigned the document, an electronic mail would be sent to one of the developers and the document would get 'stuck'. Nightly batch jobs would try to re-process any stuck documents in case a person that could be assigned the document became available, but usually manual intervention was required to resolve the problem. In the early day of the project this was acceptable, but latterly the amount of time being spent by maintenance engineers trying to resolve these problems had become unacceptable. Oracle Workflow solves this problem by introducing *error processes*. These processes are initiated automatically when a problem occurs in a workflow process, and allow us to introduce a more distributed maintenance structure. When a problem occurs within a customised routing, the person responsible automatically is assigned the document, it is then their responsibility to resolve the problem and either allow the document to proceed or reject it.

The Migration

After selecting Oracle Workflow we now had to undergo the process of migration. By using the primitive tools that we had already developed in order to help solve routing questions, we extracted a textual description of all of the process definitions from our existing system. Then started the task of introducing equivalent routings into Oracle Workflow.

An Opportunity to Simplify

We saw this migration as an opportunity to look at the processes that had been introduced over the years. We tried to establish if the routings were still relevant and to standardise them as much as possible, extracting *common working practices* and defining them as shared sub-processes. Once we had most of the process definitions complete we then took them back to the divisions[†] and asked them to verify that it was what they wanted. We took an active role in these discussions, pointing out non-standard practices and highlighting areas that could be simplified. This exercise was very useful; not only for our work, but also for the divisions themselves since it gave them an opportunity to see what workflow processes were taking place in their division. They were aware that, as developers of the system, we were ideally suited to see the organisation-wide view of the problems and to make suggestions for simplification.

Customisations

One of the essential benefits of Oracle Workflow is its open architecture. This enabled us to integrate the product with our existing Java code relatively easily.

We use Java throughout the EDH application from *servlets* (see [JSA98]) to implement the user interface, to the underlying Common Business Objects modelled after Enterprise JavaBeans (see [EJB98]). This was our first obstacle that could prevent us from using Oracle Workflow. Oracle Workflow is extended via a well-defined application programmer's interface to PL/SQL stored-procedures and although Java stored-procedures will be *available soon*™, they are not available in our existing Oracle 7.3.3 database. To solve this problem we developed the *JavaBridge*.

JavaBridge

The JavaBridge provides the ability of a PL/SQL stored-procedure to make a call to a Java function. This allows Oracle Workflow to make calls to our Java code while executing a workflow process. It is implemented in a similar way to a remote procedure call system, consisting of two parts, a small pre-processor program (written in Java) and a multithreaded Java server.

The Stub Generator

The pre-processor program takes as input a Java class-file containing the implementation of a function that we wish to use in Oracle Workflow and produces a PL/SQL *stub* file. The PL/SQL stub file contains an implementation of each of the functions that were in the input class file. The implementation in the stub file simply passes each of its arguments through a DBMS_PIPE and then returns the response that comes back from the DBMS_PIPE.

The Java Bridge Server

The Java Bridge Server connects to the other end of the DBMS_PIPE and waits for messages from the PL/SQL stubs. When a message arrives, the name of the function, the argument list is extracted from it, and then a call is made to the named function. Finally, the return value from the function call is re-packaged and sent back through the DBMS_PIPE to the waiting PL/SQL stub. Figure 6 illustrates the operation.

[†] This could not be done before introducing the processes in Oracle Workflow since we had no way of visualising the process definitions.

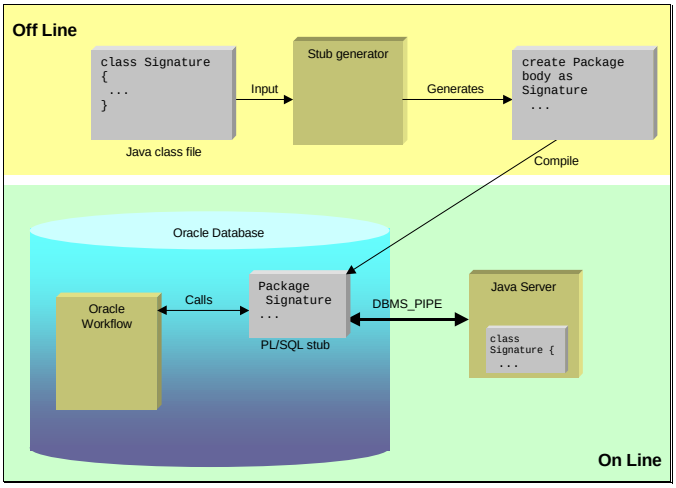

Figure 6: The Java Bridge

We see the use of the JavaBridge as a temporary solution until Java stored-procedures are available in our database.

Java Interface Builder

The Java Bridge allows calls from PL/SQL to Java, naturally therefore we needed a mechanism of calling back to PL/SQL from Java, for example using one of the Oracle Workflow APIs to query the state of a routing. Java already has this capability via JDBC (see JDBC), but in order to simplify the structure of our Java code we wrote a small Java application that would do some of the work for us. It reads a PL/SQL specification file and generates a Java class that exposes the same API as the PL/SQL package using the necessary JDBC statements. Figure 7 illustrates its operation.

Original PL/SQL Procedure

```
procedure CreateProcess
    (itemtype in varchar2,
     itemkey in varchar2,
     process in varchar2);
```

Generated Java Code

```
static public void
   CreateProcess (String itemtype,
                  String itemkey,
                  String process) {
  // JDBC Statements
}
```

Figure 7: Example of Java Interface Builder Output

Multiple Concurrent Processes

One property of CERN internal purchase orders is that the expense can be split over more that one account code. In fact, it is common that for large purchases several divisions will share the expense. In cases like these the document must be routed in parallel according to the rules specified by for each account code involved. There was no easy way to support this capability in the standard Oracle Workflow product (since the number of budget codes can vary from one to over 20 in some cases), therefore we introduced a special function activity which dynamically creates several sub-processes, then waits for them to complete. If all sub processes complete with a status of *Approved* then the function returns with a status of *Approved*. If any sub-process returns with a status of *Error*, or *Rejected*, all other sub processes are terminated, and the function returns with a status of *Error*, or *Rejected* respectively.

Interface to CERN Databases

For a workflow system to be effective it is vital that it is capable of integrating with the existing environment. In effect the workflow engine is the piece that links the other applications together. At CERN we have four main types of application that interfaces with the workflow engine.

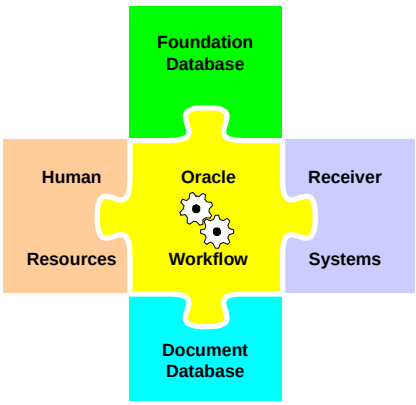

Figure 8: Oracle Workflow Dependencies

Interface to Foundation Database

The CERN foundation database contains the description of the structure of the organisation, office locations, supplier details, projects, account codes, and of signature rights. This database is interrogated by the Java function activities when determining, for example, which division owns a particular account code, or who has the right to approve annual leave for a particular person.

Interface to Human Resources

The HR database contains assignment information as well as details of leave; it is used by the workflow system when determining, for example, the supervisor of an employee, or if the person is on leave at the time.

Interface to Document Database

The interface to the Document Database operates in both directions. The workflow system needs to query the properties of a document, such as who created it, how much is it for, etc., and the Document Database is updated with information about the current state of the document (approved, rejected, etc.).

Interface to Receiver Systems

The Receiver systems are the final destination of fully approved documents. This could be the purchasing system for an internal purchase order, or the Human Resources database for an Overtime claim. A document is transferred via a specially written transfer program that is activated as the last stage of a workflow process.

Problems

During the implementation of Oracle Workflow at CERN we encountered several obstacles that we needed to overcome

Transaction isolation

Our biggest problem was caused by our choice of Java as the language in which we implemented the logic of the function activities. During the operation of oracle workflow calls are made to the function activities that make use of the JavaBridge. The use of the DBMS_PIPE means that the Java running at the other end is in a different transaction than that of the workflow engine. The result of this is that the Java functions cannot *view* the current state of the document, since the workflow engine has not yet committed the transaction. Most functions do not depend on the current state of the document, and therefore this does not pose a problem, but some functions do. We *solved* this by implementing a special message that the Java can pass back down the DBMS_PIPE. This message causes the workflow engine to commit its transaction. We realise that there are some risks with this approach, since normally the workflow engine relies on the ability to roll back to a previous state in the event of an error, in practice, however, we have not found any serious drawbacks with this solution.

Bug-lets

We also found one or two bugs where PL/SQL variables were not defined to be a sufficient size to hold values from certain Oracle defined columns, luckily using the supplied source code allowed us to fix the problem ourselves (we will be sending a bug report).

One serious problem was the fact that we have around 4,500 users defined in our database. The supplied applet tried to show a list of all users when an administrator attempted to reassign a document to another user. With the list being so long the user interface became unusable. Again, Oracle provided us with enough source code to enable us to fix this problem ourselves (although we have been told that this problem has been fixed in the next release of Oracle Workflow).

Users Perspective

We tried to make the introduction of Oracle Workflow as transparent as possible, we did this to allow us a seamless transition from the old system to the new, and if there were problems we could quickly go back! We started by running the new system in parallel with the notification system disabled. This allowed us to compare the new system with the old, verifying that the new routing decisions were comparable to the old. Once we were satisfied that the new system was operating correctly the routing could then be silently switched over.

This *stealth production* approach means that we were unable to make use of some of the functionality that is provided by oracle workflow.

The In-Tray (Work List)

We already had the capability of presenting a user with a list of documents to sign, etc. Therefore we did not need the equivalent functionality built in to Oracle Workflow.

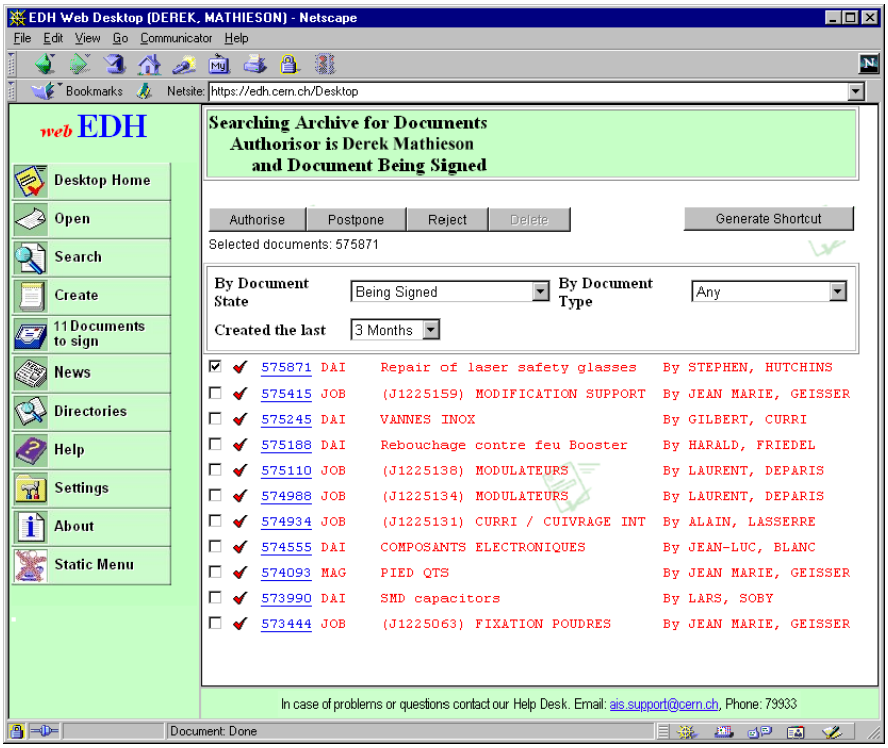

Figure 9: The EDH In-Tray

Applet

One feature that many of our users have asked for is the ability to view a diagram of the route taken by a document. Oracle Workflow provides such a facility via a special Java Applet. The Java Applet seemed to be a useful addition to EDH but we felt that it would need some changes before we thought it could be

used at CERN. We were fortunate that most of the source files of the applet are supplied by Oracle, which enabled us to begin a parallel sub-project to modify the applet.

Aspects of the Applet the we felt needed changing were:

Problem: The size of the viewable area of the diagram was fixed. Some of our diagrams were quite large and a small window made it difficult to understand.

Solution: The applet was changed to open a new, resizable, window when the user clicked a 'View Diagram' button on documents web page.

Problem: The Applet did not allow a user to Zoom In to sub-routings that were created dynamically by function activities.

Solution: This was the biggest change to the applet. Function activities that had created sub-processes were specially marked so that the zoom function worked. When a user attempted to zoom in to one of these activities, where there was more than one sub-process (for example, in the case of many accounts sharing the cost of an order) a dialogue box is presented to the user asking them to select one of the sub-processes.

Problem: The Applet provided the capability to re-assign, block, skip actions, etc., yet our existing application already had this capability via a different interface. We felt that providing the same functionality via two different interfaces could be confusing.

Solution: We removed these functions from the applet. This also solved the problem that the applet did not operate well when there was a large number of users defined (we have around 4,500 users).

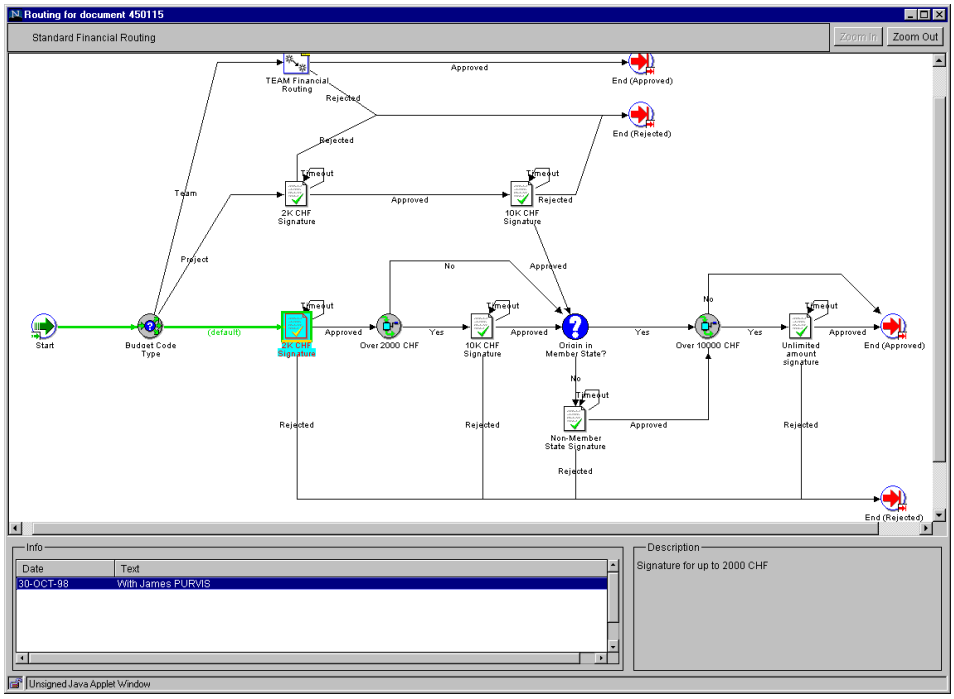

Figure 10: Example of Modified Applet

In the current implementation we have still not included the applet in our system as we currently have a policy of removing client-side Java from EDH due to stability problems (primarily on Macintosh). In addition, the use of the Applet requires us to run Oracle Web Server which, at the moment, we are not doing. In the future we hope to provide the modified applet to our users.

Document Routing Information

When a document is assigned to a user, either for signature, or simply for information, the document status is updated. Figure 11 shows an example of the information that is displayed on a typical personnel document (a Leave Request).

Document Status	
15.03.1999 17:46	Approved by creator MATHIESON
15.03.1999 17:47	With PURVIS J. awaiting approval (as Supervisor)
16.03.1999 09:16	Approved by PURVIS
16.03.1999 09:18	With ZORICA V. awaiting approval (as Admin Officer)
16.03.1999 12:17	Approved by ZORICA
16.03.1999 12:19	Transferred into ORACLE-HR

Figure 11: Document Status Information

As the document moves around between the signatories this information is maintained as a permanent record of all of the actions that were performed on the document. This information is available to the creator of the document so that they can follow the progress of their document. For some types of document status information is added to the document even after it has completed its workflow and been transferred to the receiver system. For example, in Figure 12 the creator of the purchase order can see information regarding their request as it is ordered, received, invoiced and finally paid.

Document Status	
22.02.1999 11:25	Approved by creator HERZOG
22.02.1999 11:46	With MANDICA J. awaiting approval (as BH)
22.02.1999 15:56	Approved by MANDICA
22.02.1999 15:56	Transferred to Purchasing Service
23.02.1999 14:30	Sent to buyer LARA
25.02.1999 17:00	EDH document generated order CA 1155019(1) SWIP50
25.02.1999 17:00	View order at: http://edh.cern.ch/Info/Order/CA/1155019
26.02.1999 08:00	DAI is being processed by LARA CRISTINA
04.03.1999 08:00	Order being received
09.03.1999 08:00	Order being invoiced

Figure 12: Status Information on a Purchase Order

Time-outs

Once a document is assigned to a person for signature, they have a limited time (normally three working days) in which to act upon it. They may postpone a decision on it, which effectively extends the deadline by 6 months, or they may approve or reject it. If the time-out expires the time-out branch is taken which usually is a loop back to the same signature activity. The signature activity is written in such a way that it will ask the next person that has the right to approve the action. If the signature function is unable to find another person, the process is marked as *In Error* and the Error workflow is started.

No Resign rule

Generally, no person is asked to sign a document more than once. When a person approves a document **all** of the signature rights that they have are applied to the document. If, later in the workflow, another signature right is required which is possessed by one of the people that have already signed then the activity is immediately marked as *complete* with a status of *Approved*. In some cases this is not the required behaviour, since some documents may be modified during the workflow process. To solve this problem certain signature actions have a special flag that causes them to ignore previous signatures.

Experience

One surprising aspect of the introduction of Oracle Workflow is that our Admin users now feel it is easy to introduce new modifications to the standard routings. This is somewhat contrary to our efforts to reduce the complexity and proliferation of custom routings. We currently keep control of this by requiring routing changes to be approved by a central 'Procedures Office', this is currently required because we have not given write access to the database to anyone outside of the EDH team. In future we hope to relax this restriction and allow our EDH Admin users to maintain their routing definitions themselves, although there are still questions about the guarantee of minimum standards.

One important reason for minimising the number of steps in a workflow where each step only gives approval, without modification to the document is the atmosphere of what might be called *distributed responsibility*. If an authoriser knows that there will always be someone else to sign after them, they feel the need for checking the details of the document is less important (since the next person will check it). The problem is of course, that if a supervisor sees that three other people already approved the document they are less inclined to check it! By reducing the number of approval steps to the minimum, problems like this are eliminated.

The Future

Currently we only use a fraction of the capabilities of Oracle Workflow. In the future we plan to use more of the product such as the ability to sign documents via e-mail by replying to the notification message. Oracle Workflow also supports notification rules, where users can define automatic actions that are performed when a notification arrives.

Conclusion

Although we are still in the early stages of the introduction of Oracle Workflow, results so far are encouraging. While Oracle workflow is not as complex as some other workflow products, its power comes from the fact that it is easily extendible. It is almost impossible to find a ready-made workflow product that can be integrated into an existing environment without customisation; Oracle Workflow's open architecture makes it possible. It has many excellent features such as the graphical design and visualisation tool and in the future, we hope that its support for Java will improve such that we will no longer need our *JavaBridge*.

Finally, it is important to note that with such a large number of users and a complex organisation structure accurate financial and personnel databases are vital to the success of this project.

Contact Information

For more information regarding our work on Oracle Workflow I can be contacted via e-mail at: Derek.Mathieson@cern.ch

---

References

[CBO99] .....Jonsson Per Gunnar – Common Business Object Presentation
http://edh.cern.ch/doc/CBO/index.html

[CERN].....CERN - European Laboratory for Particle Physics, Geneva, Switzerland
http://www.cern.ch

[DEJONGHE93]...De Jonghe, Jurgen - *The Paperless Organisation?* -
EOUG Oracle User Forum, Vienna 1993
http://edh.cern.ch/doc/Vienna

[EJB98].....JavaSoft - *Enterprise Java Beans Specification* -
http://www.javasoft.com/products/ejb/index.html

[JSA98].....JavaSoft - *Java™ Servlet API* -
http://www.javasoft.com/products/servlet/index.html

[WFMC96] .....Workflow Management Coalition - *Terminology & Glossary* -
http://www.aiim.org/wfmc/standards/docs/glossary.pdf