

1

Overview of Oracle Workflow

In this section, I'll be having a quick look at why you should use Workflow, the system architecture of Oracle Workflow (what the main components are, what are they used for, and how do they hang together), as well as possible alternatives to using Workflow. This section will be pretty "dry", so feel free to skip this section and dive straight into Getting Started where we begin some Workflow development and you can start to play with the product.

Objectives

By the end of this chapter, you should be able to:

- Understand why you should use Oracle Workflow;
- Understand and explain the concept of a Workflow-driven process;
- Define and describe the components of an Oracle Workflow system;
- Explain the concept of an event-driven workflow;
- Discuss the benefits and weaknesses of alternatives to workflow.

Why Use Workflow?

Becoming an e-Business

When I first joined Oracle, the marketing tag-line was "Enabling the Information Age". I seem to remember it becoming "Enabling the Information Age Through Networked Computing", but that might have just been a vicious rumour that I heard. Some time after that, though, I guess the Marketing department realised that the "Information Age" was pretty much here, and the tagline changed to "The Internet Changes Everything".

Over the past five to ten years or so, that tagline has become more and more relevant - the internet has changed (and is still changing) the way that companies do business, and use IT to support their business processes. Historically, IT has perhaps played more of a background supporting role in the business, but with the importance of the Internet the role of IT has grown. Implementing new systems that support business processes has become more common, and providing an engine which can drive through and automate business processes is a key thing.

A workflow engine is one of the key components in helping to streamline business processes, which in turn plays a critical role in transforming a business into an e-business. Using Oracle Workflow, which is supplied with the Oracle database and Applications Suite can help to streamline business processes both within and between enterprises.

When I first started developing with Workflow, I had problems trying to get my head round what exactly a "business process" was. Having spent much of my time as a technical developer writing procedures and the like, the idea of a "process" seemed a little alien to me. Businesses will have a large number of processes, but typically they are not written, formal processes. Instead they tend to rely on a large amount of knowledge

that is held by a limited number of employees, which makes attempting to automate (or even document, implement and improve) the process harder to develop. Additionally, it is usual for people to have a tendency to view the process from their own perspective only – it may be the case that some people have complete knowledge of the end-to-end process, but this is the exception rather than the rule.

A Business Process Example

One of the examples that I always use when talking about business processes with workflow is the idea of getting an expense claim approved - most people that I talk to know the frustration of trying to get some money back from their employers!

Here's a typical example of what each person involved in the process might know:

- Employee
 - I have to submit my expenses to my manager for approval.
 - I have to supply details of my expense claim and some receipts.
 - I need to comply with the Expenses Policy (if we have one!).
 - At some stage, I will get paid.
- Manager
 - I will receive some expenses from my staff.
 - I should check them over, and either send them back to the employee, or on to Accounts, who will action them.
- Accounts Staff
 - I will receive some approved expenses to action.
 - I should check the receipts and then either pay in full or short-pay the claim.

This may be a simplistic view of what staff know about the Expenses process, but there are some glaring omissions in the process:

- Who is responsible for checking the receipts match the claim?
- Who is responsible for informing the employee that the claim is to be paid?
- How does the employee ensure that their expenses are paid in full?
- Who does the employee need to contact to find out where their expense claim is in the system?

When starting to look at ways to move business processes into being Workflow-driven, there needs to be a certain amount of analysis undertaken before starting developing – without understanding what the end-to-end process currently is (and what it should become once it is implemented in the system) developing an appropriate Workflow to replicate the process will be nearly impossible. This analysis should include everyone who is involved with the process, in order to accurately model the process correctly.

Workflow-Driven Processes

One of the most important things that moving to a workflow-driven model is that there is a fundamental shift away from what I would refer to as “managing transactions” towards managing the whole business process.

The majority of actions that a business system performs tend to focus on each transaction as an individual entity. For example, when creating a purchase order line, a system typically focuses only on the order line transaction. Code within the system can interact with the record, either by selecting, updating or deleting the data. When moving towards managing the whole process, the focus moves from being on the individual transactions onto the “bigger picture”, looking at the collection of transactions which make up the whole process. Within order management, for example, this focus moves from being on the individual transactions which create an order or line, onto the series of transactions which need to take place in order to move from an order line being created through to the order being shipped.

Using a workflow engine to assist you in this process means that it becomes easier to model and adapt. Oracle Workflow can be used to

- Define business policies
- Implement the business policies
- Monitor the process activities, which in turn
 - Allows you to continuously build improvements into the process
 - Streamline the processes
 - Adapt the business processes as the business changes
- Co-ordinate the flow of work, between people and systems
- Provide an audit trail of the process

Using a workflow engine to model the business process, as opposed to (for example) using a series of PL/SQL modules, also provides a clear visual picture of what the process is. Since you have this visual image of what the process is doing, and you can see which activities in the process are time-consuming, it means that the business process can be altered to help reduce potential bottlenecks. All this can happen without the need for changing the business system underneath, which in turn gives a reduced cost of ownership and maintenance.

One of the key things that Oracle Workflow gives you is that at no stage does the tool put any artificial constraints on either the process or the business as a whole - pretty much anything is possible!!

A Business Process Example Revisited

There are two camps that we should consider when looking at the expenses process - the employee trying to get their money back; and the accounts employees. Taking the accounts department first, their main priority should be to be writing the company expenses policy, enforcing it, and (hopefully!) save the company some money in the process. The main focus of their job should not be spent on entering expenses for people and spent on the phone to irate employees explaining where their claim is in the process.

Now let's actually think about it in the real world. From my experience the majority of the time that people spend in the accounts department is spent answering questions like:

- Has my expense claim been approved?
- Is my claim waiting approval? Who's it waiting for?
- Have you checked the receipts?
- When will I get my money?
- Why was I short-paid?

If the process for submitting an expense claim was built into a Workflow, then the majority of these questions could be answered by the employee themselves - have a look at the workflow diagram for the process, and you'll see where it is, who it's been approved by, what the payment schedule date is, and so forth.

Of course, there are situations where using workflow is not a good idea, and later in this chapter we'll look at some of the alternatives to using workflow.

Components of Oracle Workflow

Oracle Workflow resides in four separate areas, and is made up of a number of components, as shown in Figure 1-1.

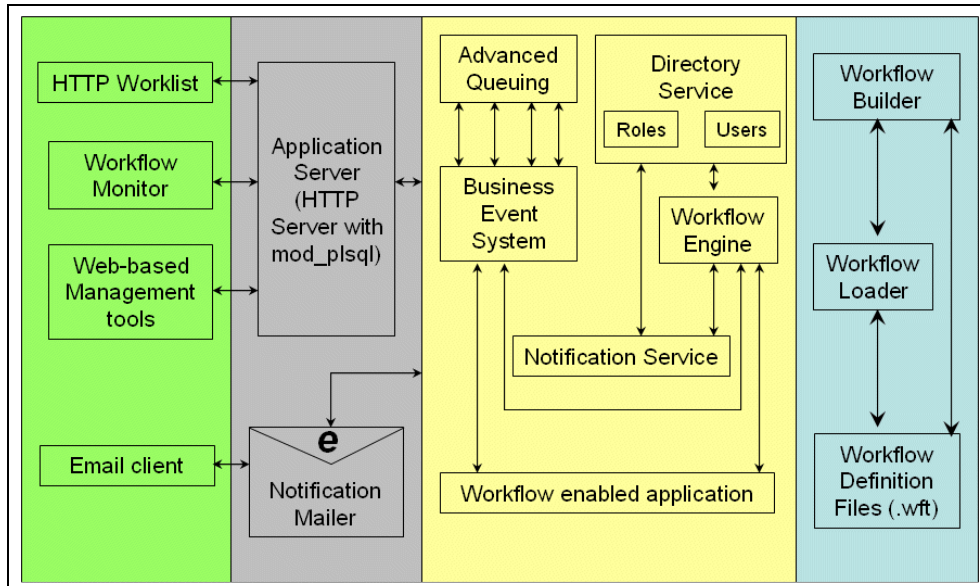


Figure 1-1. Oracle Workflow Architecture

Working from right to left in the diagram, the areas that workflow uses are

- Development Client
- Oracle database server
- Application server

- End-user client

Development Client

This is where you'll be doing a lot of the work - building your workflows. Oracle Workflow Builder is a program which runs on Windows 95 and above - and only on Windows. If you use Linux, then there are some good Windows emulators out there which you can use to run the builder - if you can't get it working then you'll need a Windows PC just to run the builder on. Some years ago, there was a plan to introduce a tool which would have been called something like "Workflow Builder Light" and the standard builder "Workflow Builder Professional". Builder Light was intended to be used for some really basic processes, and was going to have an HTML front end - having worked with some similar tools in this area, I'm pretty glad that this idea seems to have fallen by the wayside. Unless the process you are building is pretty simple, trying to define a graphical business process in what is essentially a text-based language is incredibly difficult and frustrating.

I'd like to think that the future development of Workflow builder will take it closer to the (increasingly inappropriately named) JDeveloper. There was a Workflow plug-in at one stage for JDev, which was described as "Beta-slushy" (i.e. almost frozen but not quite!) but I haven't seen a final version of it - perhaps once the Java implementation of the Workflow engine is more prominent then this move will take place.

Once you have developed a workflow in the builder, this is saved to a text file which contains the workflow definition. This .wft file is will be loaded into the database, and there are two different ways to do this. Firstly, from the Workflow builder you can choose File >> Save and supply a database connection. Secondly, you can run a command line utility which will load the definition for you.

Whilst there is nothing strictly wrong with using one or the other, this is my advice on when to use each one:

- When working in a development environment, use the Workflow Builder to save the file to the database.
- When deploying to a controlled environment (e. g. production), use the command line utility.

If you are delivering a new workflow (or a corrected one after bug-fixing etc.) to a controlled environment, this piece of code will (or should!) have been through a testing procedure. Once the workflow has been tested, if it is deployed through the workflow builder tool, then there is a possibility that someone might make a change before it gets saved to the database - if the only way to deploy a workflow to that environment is through a command line, then this can't happen. This is the main reason that I advocate deploying to the database via the command line instead of via Workflow Builder.

Which ever method you choose to deploy the workflow, you should ALWAYS ensure that when developing the workflow you are working from the flat-file version and not connected to the database. I'll come back to this idea later in the book when I talk about saving the definition, but I can't emphasize it enough - work from flat file, not from the database.

Oracle Database Server

The “guts” of the Workflow system resides within an Oracle database, and Workflow should work with any database later than Oracle release 8, although there are some caveats here.

Workflow 2.6 introduced the Business Event System (which I'll describe in a couple of paragraphs) and under the covers increased the number of Oracle Advanced Queues within the system. Whilst Oracle AQ was available with a release 8 database, I really wouldn't recommend trying to use AQ prior to 8i as it was with the 8. 1. 5 release of the database that queuing technology became more robust. Additionally, within the 8i database, you need to be running Enterprise Edition in order to create custom queues.

As with all Oracle products, the certification matrix should be checked before determining which versions of the software you need to use.

Workflow Engine

This is the part of the system where the majority of the processing will take place. The term “engine” is a little misleading - the way that it is implemented within the database is as a PL/SQL package which is called by each workflow instance to communicate changes in state. The “engine” then looks at the workflow definition, which is stored within the database in a collection of tables, to determine what to do next. Since each workflow process is merely a series of activities which are connected together, and the outcome of each activity determines the next activity to perform, the primary responsibility of the Workflow Engine is to control the process from start to finish, determining when to branch out or fan-in processes. So, although the Workflow Engine is responsible for doing the work, and driving the process onwards, while there is no work to be done there is no performance overhead to consider. Comparing the Workflow Engine to an engine in a car, for example, is appropriate – while there is something to do, then the engine drives the work forwards. However, when there is no work for the Workflow Engine to perform, sitting idle does not consume any “fuel”.

Within Workflow, there is also a concept of a “foreground engine” and a “background engine”. The background engine (again not really an engine!) is another piece of PL/SQL code which checks the workflow tables and identifies any activities which require processing, for example activities which have been deferred or have timed-out. The reason for referring to this as the background engine is because it needs to be configured to execute at a regular interval, rather than running immediately when an application communicates with the workflow engine. When an activity is identified by the background engine, a separate call is made to the foreground engine API to pick up the activity and continue processing it.

When configuring a background engine, the frequency will need to be considered. For example, if you have a process where an activity requires a response within 1 minute, you need to ensure that there is a background engine that is running frequently enough to identify the activities. Since the background engine can be processor intensive, within the same database it is possible to run multiple background engines (either with different or identical parameters) which will share the workload between them.

Determining and configuring background engines is more of a system administration task - as a developer there is no need to code for whether the activity is running via a foreground or background engine, or even to check that the background engine is running successfully. If a particular process is dependent on a background engine running, then

this should be highlighted within the design or installation documentation, but it is reasonably safe to assume that the background engine will run as normal once it has been scheduled.

Business Event System

The Business Event System (BES) was introduced as part of Workflow 2.6, in order to make it easier to develop workflows which respond to different firing conditions either to start or continue a process. Prior to release 2.6, it was possible to develop a workaround, but BES made the whole design, build and maintenance of an Event-driven workflow significantly easier.

Note on Event-driven Workflows

When I first started talking about workflows being Event driven, this was one of the hardest initial points to get across – when we get to Chapter **XX**, which deals with the Business Event System in detail, this should become clearer.

Workflows can be used for a number of different purposes, and depending on what the primary function of the workflow is there are different categories that I would use to describe the workflow processes.

A **non-event-driven** workflow is one that performs a series of automated tasks with no intervention from either people or other systems. The workflow will progress from one step to the next until the process completes. Don't believe that just because the process does not have any intervention from other systems or people that it is simplistic in any way.

An **event-driven** workflow is one which involves something happening to communicate with the process. This could be, for example, a response to an email notification from a user, or a message being communicated with the workflow process from an external system.

More detailed examples are provided in Appendix A.

The main component of the Business Event System is the Event Manager. The manager allows you to register events which are significant to your business, and to define subscriptions which identify what happens when the event occurs. The subscription can communicate with a Workflow process, call a piece of custom code, or raise further business events with further processing. Additionally, the event manager allows you to define multiple subscriptions for the same business event, and to have subscriptions which fire synchronously (i.e. immediately), or defer to the background for processing at a later date (asynchronously).

Business events can be raised from different sources - either locally via PL/SQL or Java, or externally by a message arriving on an Oracle AQ which has been registered with the event system. If the event is raised locally, then the synchronous subscription processing will be executed in the same transaction as the code which raised the event. Any deferred local subscriptions, and all external subscriptions, will only be fired when a job which is looking to process the messages is run. This job, whilst similar in nature to the Background Engine described earlier, is a different job which is scheduled in a different

manner. Scheduling and configuring processing of external messages and deferred subscriptions is covered in chapter **XX**.

The other main component of BES is Event Activities. The event activities are the method of modeling receiving, sending and raising business events within a Workflow process. By adding event activities within a process, the workflow can wait for a business event subscription to fire, raise further business events which in turn will trigger more processing, or send a business event from one system to another. Within the process that we will be building as part of the book, we will look at using events in a workflow process in some detail.

The first workflow project that I was involved with was concerned with trying to build an event driven workflow using Workflow release 2.5 (i.e. pre-BES). The next project that I worked on was one of the first implementations of Workflow 2.6 worldwide, again developing event driven workflows. Having had to build both variants, I know which one I would definitely prefer, and given the choice I will always include business events within my workflows in order to either initiate the workflow or to communicate changes of state between workflow processes. When we look further at developing event-driven workflows, we will look at how it can be done *without* using BES.

BES introduces a simple way of creating flexible workflow initiation, and when we look at how to start workflows I'll cover this in more detail.

One thing that I will say at this stage is that I'm a real business event "evangelist" – if I can do something via the business event system, I will do. Hopefully by the end of the book I'll have at least explained the importance of building event driven workflows, if I don't manage to "convert" you to building workflows that way.

Application Server

Within the Application Server area of Oracle Workflow, there are two main components - the notification mailer and the HTTP server.

Notification Mailer

The notification mailer is the part of Oracle Workflow which sends notifications outside of the database via email. The mailer handles both inbound and outbound communication, allowing non-database users to provide input into the business process – one of the key benefits that Oracle Workflow delivers. This inbound communication via email is particularly important in expanding the process outside of the organization to include, for example, trading partners.

Depending on the version of Workflow which you are running, the notification mailer uses differing technologies. Version 2.6.3 and later of workflow uses a notification mailer which is written in Java, and supporting the JavaMail standard that was introduced by Sun. The Java mailer is easier to maintain and support, as well as to configure, since this is managed through a web-based interface. The Java mailer also provides connectivity to SMTP (Simple Mail Transport Protocol) compliant mail systems. Earlier versions of the notification mailer run as a C program, and is configured via files which are placed on the application server. The versions of the notification mailer prior to release 2.6.3 rely on sending emails via either UNIX sendmail or a MAPI (Messaging Application Programming Interface) compliant system. One issue that using earlier versions of the notification mailer is that some email client tools are not capable of

handling Workflow generated emails which they perceive as attempting to violate a security flaw.

HTTP Server

The application server component of Oracle Workflow is required in order to generate the web-based interface which allows end-users to monitor and manage workflow processes and the workflow system. For use with Workflow, the Application Server only needs to provide an HTTP server which supports mod_PL/SQL. The mod_PL/SQL component is used to build the pages which are displayed to the user, and so it is possible to use different application servers than Oracle Application Server. However, Oracle Application Server is the only application server which Oracle will certify for use with Workflow. Additionally, it is possible to deploy workflow processes without an Application Server, but since this removes the standard management and monitoring tools, this is an unlikely scenario to consider.

End-User Client

The final component that makes up Oracle Workflow is the End-User Client. Essentially, this is what the users are going to be seeing and using, and will determine how users interact with the workflow system.

Email Client and HTTP Worklist

These two components compliment each other, so I'll deal with them both together. The two elements perform pretty much the same function – they allow users to see what notifications have been sent to them, either requiring a response or just for information only.

The HTTP Worklist is a web-based interface which allows users to read and action their notifications via the internet. Depending on each notification configuration, users can reassign the notification to other users for action, close the notification or respond to the notification.

The Email client is the one component of Oracle Workflow which isn't actually part of the system. Depending on global- or user-preferences, notifications can be sent to users via email. The Notification Mailer component described above will send the notification via email, and then the email client that the user is using will display the notification to the user.

For interacting with notifications, my recommendation would be to ALWAYS use the HTTP Worklist, rather than relying on users to use email to interact with the database. When we look at notification in detail, the reason for this recommendation should become clear.

Workflow Monitor

The Workflow Monitor is again provided by the Application server, and allows users or administrators to view workflow processes in more detail. The monitor has two main components – an HTML-based section of pages and a Java Applet. The HTML pages provide details about the process, including what steps have been executed, how long each step took, what the result was, and (where appropriate) which user performed the

action. The Java Applet is used to provide a view of the process in graphical form. Again, this allows the user to see what steps the workflow took, providing the same graphical view that the Workflow Builder provides. If the user is an administrator, they can also manipulate the workflow data, including forcing the process to retry steps, skip steps completely, or force processing down a particular route. Since this is very powerful manipulation of the workflow data, it is important to ensure that your workflow system administrator access is restricted.

Web-based Management Tools

Depending on how the database is being managed, you can either get a lot by the way of management tools, or you can be left surprisingly short!

If the system is being managed using Oracle Enterprise Manager (OEM), the “Oracle Workflow Manager” can be used to provide detailed analysis of the system, including graphical diagrams which provide an overview of the status of the system.

If you are not using OEM to manage your environment, then the “management tools” that are provided out of the box are pretty limited. Within the Workflow Home, there is a collection of useful and powerful SQL scripts which can be used to manage your system, but apart from the scripts there are no other tools to manage the system.

Chapter **XX** will provide guidance on managing your workflow environment, using both OEM and the SQL scripts provided.

Alternatives to Using Workflow

It might seem a little strange to be including a section on *not* using Workflow within this book, but I think that it's important to recognize that there are different ways of performing similar tasks.

1. Using multiple concurrent programs.

Through the use of multiple concurrent programs, it is possible to construct a workflow process. Since it is possible to trigger different programs from within a concurrent program or request set, including determining whether to wait for the job to complete or not, you could use a sequence of concurrent programs to develop a pseudo-workflow solution.

Whilst it is possible to use multiple concurrent programs to perform some degree of workflow functionality, this should only really be considered for a limited implementation. Given that concurrent programs are used within an Oracle Applications environment, which will include Oracle Workflow as standard, the preferred tool for developing processes should really be developed using Workflow instead.

2. Using custom code.

In a similar way as developing your own workflow engine using concurrent programs above, it is quite possible to use custom PL/SQL or Java to model your business process. Depending on the complexity of the process that you are

developing, it maybe easier and quicker to write some code directly rather than embedding the code within a Workflow.

For example, if all that the process needs to do is to wait for a trigger event to occur, run some code and then loop back until the trigger event occurs again, then this could be easily implemented using PL/SQL or Java. However, if there is some complicated business logic which needs to be performed, particularly if there needs to be two-way communication between the system and the users, then there is a very strong case to be made for using the Workflow engine rather than re-inventing the wheel.

When I first ran a training course for Workflow, one of the attendees was convinced that there was nothing that Workflow could do which could not be done by writing custom code. I would like to think that some years later we were both right to a certain degree – for the simple “process” he was looking to implement then there was no need for a Workflow engine, but for anything more complicated then attempting to write it in PL/SQL would be significantly more complicated than moving it into an Oracle Workflow. Since he is now working as an Oracle Workflow developer, I like to think that the benefits of using workflow became clear!

3. Using Scripting.

Oracle Scripting can be used to guide an end-user through a series of questions and actions based on the previous responses. This is of great benefit when there is a definite list of questions that need a large amount of user-interaction. However, if there is a more limited list of user responses required, and a greater amount of processing logic, then Scripting becomes less viable and Workflow should be given greater consideration.

4. Using UTL_SMTP.

If the requirement is merely to send an email to a given user (or groups of users), and the email does not require a response, then the standard database package UTL_SMTP can (and should) be considered. One key advantage that UTL_SMTP offers over using a workflow to do the same job is that the database package provides some important functionality that is missing from Oracle Workflow – the ability to send a “cc” or “bcc” to an email, which some business functions require. Additionally, you can use UTL_SMTP to send emails to people without needing to store all their details within a directory service, which if the user-base is relatively fixed and small can be an advantage over maintaining the directory service that Workflow uses.

Using multiple concurrent programs or PL/SQL code to simulate a workflow engine can work in some simple processes, or can be used in conjunction with Workflow if there is already a volume of code written which will do the job. However, if the process has a reasonable amount of complexity to it, then I would advocate using workflow – it is significantly easier to change the order of processing in a workflow diagram than having to re-write a multi-lined PL/SQL program! One analogy that a colleague of mine came up with was “re-inventing a wobbly wheel”, i.e. attempting to re-invent the wheel, but making it worse!