Class 13

Learning Objectives

- Understand the purpose of workflow management systems and their relationship to business process management
- Be able to describe in general terms the architecture of a workflow management system

Textbook Chapter 1

Before reading the first chapter, you should read the Preface (page v). While this may seem unusual, the preface is a good introduction to the book and the goals it tries to achieve. This course shares some of those goals.

Chapter 1 provides an overview of the history of different BPM approaches and positions YAWL within this history as a remedy to the lack of standardization of industry-led approaches to BPM and workflow management. In particular, the chapter suggests that standards efforts may have failed by lack of precise definitions, the interests of software vendors, consultants, and other parties. As a result of this, the workflow pattern initiative was begun to identify sets of recurring patterns in workflow specifications. Originally, these patterns were used to evaluate BPM modeling languages and assess their expressiveness and suitability for modeling these patterns. Subsequently, the YAWL language was developed, based on the mathematical formalism of workflow nets, to directly support these patterns. Workflow patterns in turn take Petri nets as their foundational formalism. The book (not necessarily in this chapter) provides extensive detail on the mathematics of workflow nets and Petri nets. *This course is not concerned with these mathematical foundations and details and you can skip these in whatever chapter they appear*.

Section 1.7 finally introduces a concrete example of a business process. At this point, do not be concerned with the details of the graphical language, this will discussed in detail in the next class. Simply note that there are two pieces of software involved, the graphical editor with which to model processes ("build time"), and the runtime environment, which is accessed through a web-browser and which executes the process model ("run time"). You should not the general architecture of the YAWL system, shown in Figure 1.8. Other workflow management systems are similar in principle, although the precise architecture might differ somewhat.

As you read this chapter, you may want to reflect on the following questions:

- The textbook mentions the advantage of a WfMS over specialized software whereby changes to the process logic can be made simply by changing the process specification, rather than requiring changes of a piece of software (pg. 4). Is this always idea? What possible pitfalls or disadvantages could this have?
- The book makes a strong argument against existing standards such as BPMN or BPEL and claims there are not well-defined, leading to interoperability problems. You may wish to spend some time on internet research and see whether this is really the case.
- The YAWL system is heavliy componentized (see Figure 1.8). What are some of the advantages of separating the system into different components? Are there disadvantages as well?

Textbook Chapter 7

Chapter 7 of the textbook continues the look at the internal architecture of the YAWL system (and workflow management systems in general) at a more detailed level. There is probably too much detail in this description, and, in Section 7.1, you can safely ignore all the technical acronyms about REST, JDOM, Xerces, Saxon, Xquery, Servlets etc. However, Figure 7.1 and the general description of the architecture in Section 7.2 is important to understand how YAWL works. A case here is a workflow instance, i.e. a particular workflow that is executing. For example, a case might be the specific workflow instance that is handling the processing of purchase order 123. Case data refers to the data objects associated with such a case. For example, this would be all the information about purchase order 123, e.g. the material and amounts to purchase, the date required, etc.

The concept of a work item is defined precisely, and you should note the definition of *enabled*: A work item is enabled when the task/activity in the workflow is reached. At this point, it could be worked on, though it may have to wait awhile for resources to become available, e.g. a human worker selecting the work item from her inbox or queue. Thus, the time when a work item is *enabled* is not necessarily the time at which it becomes *active* (is being worked on).

Section 7.3 presents the YAWL components or services in detail. It is sufficient that you understand *in general terms* what each component does, how it does it, and how it relates to other components. For example, you do not need to know each of the interfaces of the engine, but you should have a rough idea of the purpose of the engine and how other services make use of it (this being primarily the set of notifications that the engine creates and that services can subscribe to receive). Similarly, you should have an understanding of the editor, the resource service (and its sub-services), the worklet service (specifically the exception service), web service invoker service and the custom services. Many of these will be used in the following classes and in the assignments you will do with YAWL.

You may wish to use Exercise 4 at the end of Chapter 7 as a reflection exercise.

Review Questions

After this class, you should be able to answer the following review questions:

- Name three other (previous) approaches to business process modelling than YAWL
- Explain and differentiate the concepts “build-time” and “run-time”
- Draw a diagram identifying the main components of the YAWL software and explain their purpose.
- Describe the three layers of the YAWL system architecture.
- Describe the path of manual work items through the YAWL services.
- Exercises 1, 2, and 3 at the end of Chapter 1
- Describe the events that the YAWL engine provides to other services
- Describe the four sub-services of the resource engine.
- Describe the purpose of the worklet service for exception handling
- Describe 5 of the 7 possible types of exceptions in YAWL