Class 9

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
- Describe organizational or managerial aspects of a successful process implementation and differentiate those for unsuccessful process implementations

Readings

Business process automation is also called workflow management and a software system that provides such capabilities is called workflow manageent system (WfMS). The first article for this class provides a life-cycle model for business process automation. The second article for this class examines the general capabilities and architecture of such a system.

Miers, D.: Best practice BPM. *ACM Queue*, March 2006.

This article presents a life-cycle model for business process management. However, Miers' article is primarily focused on building or configuring the supporting information systems. In fact, he frequently talks about the capabilites of BPM suites. These are software application that provide support for the BPM life-cycle that he enivisons in the paper.

This focus on software development evident in the first paragraph where BPM (business process management) is characterized as a way to develop information systems or software applications. As the previous readings showed, there is a lot more to BPM than just the information technology aspect. This different focus is also reflected in his suggestion that organizations should be able to go around the life-cycle within 3 months. While this may be possible when only the information system is considered, it should be clear from the previous readings that organizational change management and implementation will take considerably longer. Only small changes to an existing process that is already supported by an existing information system appear to be feasible within a short timeframe of 3 or 6 months.

The remainder of the article takes a distinctly technical perspective, when it talks about process models, flow diagrams, etc. to understand a process. We will see more of this in some form or another in the rest of the course. Similar, in his “Design to Deployment” section, Miers deals primarily with aspects of software application design and configuration, such as user interface, integration with legacy applications, capturing of appropriate data for process metrics, and being able to control the running process.

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

- Miers suggests that “Creating the 'definitive' requirements specification is a waste of time. ... Indeed, the whole notion of a definitive requirements specification becomes irrelevant in a BPM project.” (pg. 43). This runs counter to much current wisdom where any development or implementation should begin with a complete set of well-defined requirements. Why does Miers propose this, and what are the advantages and disadvantages of his proposal?
- Why do you think Miers suggests that “rather than attempting to transform the as-is model ... build a new set of models” (pg. 44)?

- Concerning changes during development, Miers suggests that “if the proposed change has a dramatic impact, then identify it for a iter iteration” (pg. 45) Does this not run the danger that the developed process will not be useful for the businees, if it requires drastic changes? What (else) can be done to address this issue?

de Jong, P.: Going with the Flow. ACM Queue, March 2006.

While the readings define most terms they use, here just as an introduction some of the frequently used concepts. A workflow is an *executable* business process. This means that it is possible to automatically execute (“run”) and control this process by a WfMS. Not all business processes are like this: The article for this class show that a business process needs to be specified in great detail and with much precision. A workflow *instance* is one particular execution of such a business process. For example, there might a process (and workflow) for sales-to-cash, that begins when customers order products and ends when the products are paid for. The general definition of this process is a workflow. When customers A, B, and C each submit an order, the WfMS creates three separate *instances* of the workflow, one to handle each order. These can then run independently of each other. The *activities* in a workflow are then instantiated as *tasks* within the workflow instance. E.g. the general activity “perform credit check for customer” becomes the task “perform credit check for customer A” in the first workflow instance.

A data object is a piece of information or data (e.g. an order to be processed) that needs to be worked on, e.g. created, checked, updated, authorized, etc. Data objects are assigned to activities as inputs or outputs. For example, the activity “perform credit check for customer” may require the customer order (with total order amount and the customer number) as input, and produces an approved customer order (to which credit approval has been added) as output, which can then be used in some other activity. The WfMS keeps track of all the data objects used by all workflow instances by storing them in a database (“making them *persistent*”).

The activities of a workflow typically require resources (e.g. human beings or physical resources such as machines) although it is not necessary that the resources be completely specified. Instead, usually it is sufficient to specify the type of resource. When speaking about a type of human resource, we use the term *role*. Different employees can perform the same role and the same employee may be able to perform multiple roles. For example, Jane, Jill and John may all be able to drive the forklift to pick stock in the warehouse (role A), but, in addition, John can also act as warehouse manager and sign off on inventory moves (role B). One speaks of John filling a role or of John being a role instance.

A WfMS controls all active workflow instances and determines, based on the rules in the workflow, the next set of tasks (there may be multiple tasks that can be performed at the same time, i.e. in parallel). It then looks at the resource requirements for these tasks (e.g. the roles), and assigns the tasks to role instances (e.g. human beings). There are multiple ways of routing, e.g. assignment based on some criteria, offering of tasks, etc. The work items that are inputs to the assigned or chosen tasks are then routed to the *queue* for this role instance. Think of this queue as a combination of to-do list and inbox. It provides a list of all the tasks together with the work items that the tasks need. The WfMS has to make sure that sufficient resources are available to handle all tasks, that resources are utilized fairly or evenly and that tasks do not wait unnecessarily for resources to become available.

The de Jong (2006) article is an introduction to WfMS and provides an overview of four aspects of a WfMS, the messages, the work items, the business rules, and the flowcharts. When discussing flowcharts, which describe the workflows, de Jong uses the Business Process Modeling Notation (BPMN) as an example. You can safely skip the details of this description. The YAWL language used in

this course is different (although based largely on the same concepts), and a later class will compare the two in detail.

de Jong uses the terms “workflow runtime” for what is called the WfMS “engine” and used “workflow design” to mean the “workflow”, i.e. the description or specification of the executable business process. He uses the term “work item” for what is called “task” above, i.e. an individual activity in a workflow instance.

A typical WfMS saves information about everything that happens (each task execution, each resource assignment, changes made to data, decisions made, etc.) in a log file. This is called the “trace” of a workflow instance execution. This information can be fed as input into the WfMS when simulating a new workflow design.

In addition to the notion of work items introduced above, de Jong adds the idea of messages. Work items can be attached to messages. Thus, messages are used to send work items from task to task (and resource to resource, as assigned to the tasks).

Serialization/Deserialization and hydrating/dehydrating simply refers to the fact that internally, the WfMS will save away some of the currently running, but inactive, workflows in a database and retrieve them later, when they become active again.

A transaction is a series of tasks that must either all complete successfully or not at all. This means that in case one of the tasks fails, the actions of the previous (successful) tasks must be undone (“rolled back”). Furthermore, the results of the series of tasks is not visible to other tasks until they have all completed successfully. A good example the transfer of money between two accounts. Assume that each account has a balance of $500 for a total of $1000. First, $100 is taken out of one account and then $100 is put into another. The money transfer is OK only if both actions take place. If the second one fails (e.g. there is insufficient funds, or no authorization), the first one will have to be undone, otherwise the accounts would not balance anymore. If someone enquired about the total money in the accounts, the intermediate balance ($900 after the first action) should not be visible to others.

Finally, de Jong mentions the problem of asynchronous task execution by separate systems and the implications of this for the WfMS system. An example might be the checking and updating of inventory as a work item or task. While the WfMS is in charge of executing (and scheduling) this task, it will actually be performed e.g. by the inventory management system. The WfMS has no control over the inventory management system and how long it might take to complete this task.

While reading this article (and in your reflection) you might want to think about the following:

- Do you agree with de Jong's assessment that workflow management extends the computerization into the natural and open parts of an organization?
- Why do you think there is little support for transactions in most workflow systems? Do transactions provide an advantage over explicitly handling exceptions the way de Jong describes it? In what situations might there be an advantage?
- Is it a good idea (or, under what conditions might it be a good idea) to allow changes to a workflow while it executes? Who should make those changes?
- What types of exceptions can you imagine for tasks or work items?

Review Questions

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

- Explain Mier's (2006) life-cycle model for implementing business processes.
- Define and explain the following terms
  - workflow management system,
  - workflow,
  - work item,
  - role,
  - exception and exception handling,
  - transaction,
  - serialization and deserialization,
  - trace and tracing,
  - choreography
- Explain the three ways in which a business rule can be triggered
- Explain the three purposes of process models in BPM?