WORKFLOW RESOURCE PATTERNS

Nick Russell[1], Arthur H.M. ter Hofstede[1], David Edmond[1]
[1] *Centre for Information Technology Innovation, Queensland University of Technology*
GPO Box 2434, Brisbane QLD 4001, Australia
{n.russell,a.terhofstede,d.edmond}@qut.edu.au

Wil M.P. van der Aalst[1,2]
[2] *Department of Technology Management, Eindhoven University of Technology*
GPO Box 513, NL-5600 MB Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract

Workflow systems seek to provide an implementation vehicle for complex, recurring business processes. Notwithstanding this common objective, there are a variety of distinct features offered by commercial workflow management systems. These differences result in significant variations in the ability of distinct tools to represent and implement the plethora of requirements that may arise in contemporary business processes. Many of these requirements recur quite frequently during the requirements analysis activity for workflow systems and abstractions of these requirements serve as a useful means of identifying the key components of workflow languages.

Previous work has identified a number of *Workflow Control Patterns* and *Workflow Data Patterns*, which characterize the range of control flow and data constructs that might be encountered when modelling and analysing workflows. In this paper, we describe a series of *Workflow Resource Patterns* that aim to capture the various ways in which resources are represented and utilized in workflows. By delineating these Patterns in a form that is independent of specific workflow technologies and modelling languages, we are able to provide a comprehensive treatment of the resource perspective and we subsequently use these Patterns as the basis for a detailed comparison of a number of commercially available workflow management systems and business process modelling languages.

Keywords: Patterns, Resource Modelling, Organisational Modelling, Workflow Systems, Business Process Modelling

1 Introduction

Over the last decade there has been increasing interest in *process-aware information systems*, i.e., systems that are used to support, control, and/or monitor business processes. Typical examples of systems that are driven by implicit or explicit process models are WorkFlow Management (WFM) systems, Enterprise Resource Planning (ERP) systems and Customer Relationship Management (CRM) systems. These systems can be configured to support specific business processes. Recently, several

languages have been proposed to support process-orientation in the context of web-services (cf. BPEL4WS, BPML, WSCI, etc.). The support of IBM, Microsoft, HP and SAP for a language like BPEL4WS (Business Process Execution Language for Web Services, [ACD+03]) reinforces the fact that process-awareness has become one of the cornerstones of information systems development. Existing languages and tools focus on control-flow and combine this focus with mature support for data in the form of XML and database technology. As a result, control-flow and data-flow are well-addressed by existing languages and systems. Unfortunately, *less attention has been devoted to the resource perspective*. This continues to be the case even with relatively recent advances such as BPEL4WS which does not provide any degree of direct support for resources in business processes based on web-services. Similarly, languages like XPDL [Wor02], the "Lingua Franca" proposed by the Workflow Management Coalition (WfMC), has a very simplistic view of the resource perspective and provides minimal support for modelling workers, organizations, work distribution mechanisms, etc. John Seely Brown (a former Chief Scientist at Xerox) succinctly captures the current predicament: "Processes don't do work, people do!". In other words, it is not sufficient to simply focus on control-flow and data issues when capturing business processes, the resources that enable them need to be considered as well.

In this paper, we focus on the resource perspective. The resource perspective centres on the modelling of resources and their interaction with a process-aware information system. Resources can be human (e.g., a worker) or non-human (e.g., plant and equipment), although our focus will be on human resources. Although process-aware information systems typically identify human resources, they know very little about them. For example, in a workflow system like Staffware a human resource is completely specified by the work queues (s)he can see. This does not do justice to the capabilities of the people using such systems. Staffware also does not leave a lot of "elbow room" for its users since the only thing users can do is to execute the work items in their work queues, i.e., people are treated as automatons and have little influence over the way work is distributed. The limitations of existing systems triggered the research presented in this paper. By identifying Resource Patterns and providing a critical analysis of existing workflow management systems we hope to encourage workflow researchers and workflow developers to improve the resource perspective in future offerings.

This work extends the *Workflow Patterns Initiative*[1] to include a detailed analysis of the resource perspective. Initially, the initiative focused on control-flow dependencies in workflow languages [AHKB03]. Later it was extended to include web-services composition languages [Aal03] and the data perspective [RHEA04]. It is interesting to note that this work has directly influenced tool selection processes, commercial and open-source workflow systems, and workflow standards (for details see our website www.workflowpatterns.com). This paper adopts an approach similar to that in [AHKB03] and [RHEA04] although in this case, the focus is on the resource perspective.

This paper proceeds as follows. Section 2 provides a taxonomy of the resource and workflow concepts utilised in this paper. Section 3 describes a series of Resource Patterns which have been identified in the context of workflow systems. Section

[1] See www.workflowpatterns.com for more information, i.e., animations, papers, tool evaluations, etc.

4 relates these Patterns to a number of commercial workflow systems. Section 5 discusses the results of these investigations and potential future research directions. Section 6 outlines material in related research areas and Section 7 concludes the paper. A number of appendices are included which explain the ratings achieved by each of the workflow products for specific Resource Patterns and outline the overall Patterns assessment criteria.

2 Resource characterisation

2.1 Resource modelling

For the purposes of this research, we consider a resource to be an entity that is capable of doing work. This is usually assigned to the resource in the form of work items, each of which describe an integral unit of work that the resource should undertake. A resource is classified as either *human* or *non-human* i.e. a resource that does not correspond to an actual person - e.g. plant and equipment.

A human resource is typically a member of an *organisation*. An organisation is a formal grouping of resources that undertake work items pertaining to a common set of business objectives. They usually have a specific *position* within that organisation and in general, most organisational characteristics that resources possess relate to the position(s) that they occupy rather than directly to the resource themselves.

As a consequence of their position(s), resources may have a number of associated *privileges*. They may also be a member of one or more *organisational units* which are permanent groups of human resources within the organisation that undertake work items relating to a common set of business objectives. Similarly they may also be a member of one or more *organisational teams*. These are similar to organisational units but not necessarily permanent in nature. Even less formal in nature is the notion of organisational *groups* which are often used to define groupings of resources with some common characteristic or cause e.g. social club members, fire-wardens etc.

Each resource is generally associated with a specific *branch* which defines a grouping of resources within the organisation at a specific physical location. Resources may also have a *level* which indicates their position within the organisational hierarchy. They may also belong to a *division* which defines a large scale grouping of resources within an organisation either along regional geographic or business purpose lines.

In terms of the organisational hierarchy, each resource may have a number of specific relationships with other resources. Their *direct report* is the resource to whom they are responsible for their work. Generally this is a more senior resource at a higher organisational level. Similarly, a resource may also have a number of *subordinates* for whom they are responsible and to which each of them report. Finally, a resource may also have a *delegate* which is an alternate human resource to which they assign work items previously allocated to them. This reassignment of work items may occur on a temporary or permanent basis.

A resource may have one or more associated *roles*. *Roles* serve as another grouping mechanism for human resources with similar job roles or responsibility levels e.g. managers, union delegates etc. Individual resources may also possess *capabilities* or attributes that further clarify their suitability for various kinds of work items. These may include qualifications and skills as well as other job-related or personal attributes such as specific responsibilities held or previous work experience. They

may also have *features* which further describe specific characteristics that they may possess that could be of interest when allocating work items.

Non-human resources may be *durable* or *consumable* in nature. A durable resource is one whose capacity to undertake work is unaffected by the amount of work that it has undertaken, whereas a consumable resource is one that is consumed (either partially or wholly) in the act of completing a work item. There is usually a *rate of consumption* or *capacity* associated with consumable resources indicating how much work they can actually undertake before being depleted and requiring further replenishment.

Each resource may have a *schedule* and *history* associated with them. These are essentially inverses of each other. A schedule is a list of work items that a resource is committed to undertaking at a specified future times where as a history or work log is a list of work items that a resource has completed (successfully or otherwise) at some time in the past.

The following meta-models depicts the major characteristics of a resource described above in the form of an Object Role Model (ORM) [Hal01] diagram. Figure 1 illustrate the concepts pertinent to non-human resources, where as Figure 2 captures the attributes relating to human resources. Figure 3 describes how complex attributes (e.g. features) are captured for resources.

Several commercial workflow systems have been examined in the context of this research. Most of these utilise an internal organisational model to identify resources and represent the relationships that exist between them. In all cases, the organisational model used employs a subset of the concepts identified in Figures 1 - 3.

- Staffware has a relatively simple model that denotes users (i.e. individual resources), groups and roles, and allows work to be assigned on the basis of these groupings. The use of roles is somewhat restrictive as each role can only be undertaken by a single user.

- WebSphere MQ Workflow provides a richer model that allows users to be described in a broader organisational context (e.g. organisational unit, branch, division to which they belong, who their manager is). It also supports roles and there can be a many – many correspondence between users and roles. Work items can be assigned to users based on various characteristics of the organisational model.

- FLOWer supports an organisational model that is exclusively role-based and is defined in terms of a role hierarchy. Correspondences are established between individual users or groups of users and roles. All work allocations are role-based.

- COSA provides an organisational model that embodies many of the human resource concepts from Figure 2. Users can be defined and organised into groups and hierarchies of groups are supported. Additionally both individual users and groups can be assigned roles and there is provision for the identification of group supervisors. Competencies can be identified for individual workflow users. Work items can be routed to users using any of these concepts.

- iPlanet has a minimal organisational model that allows for the identification of users and assignment of roles to users. There is also support for extended user profiles that allow attributes to be used in the allocation of work to users.

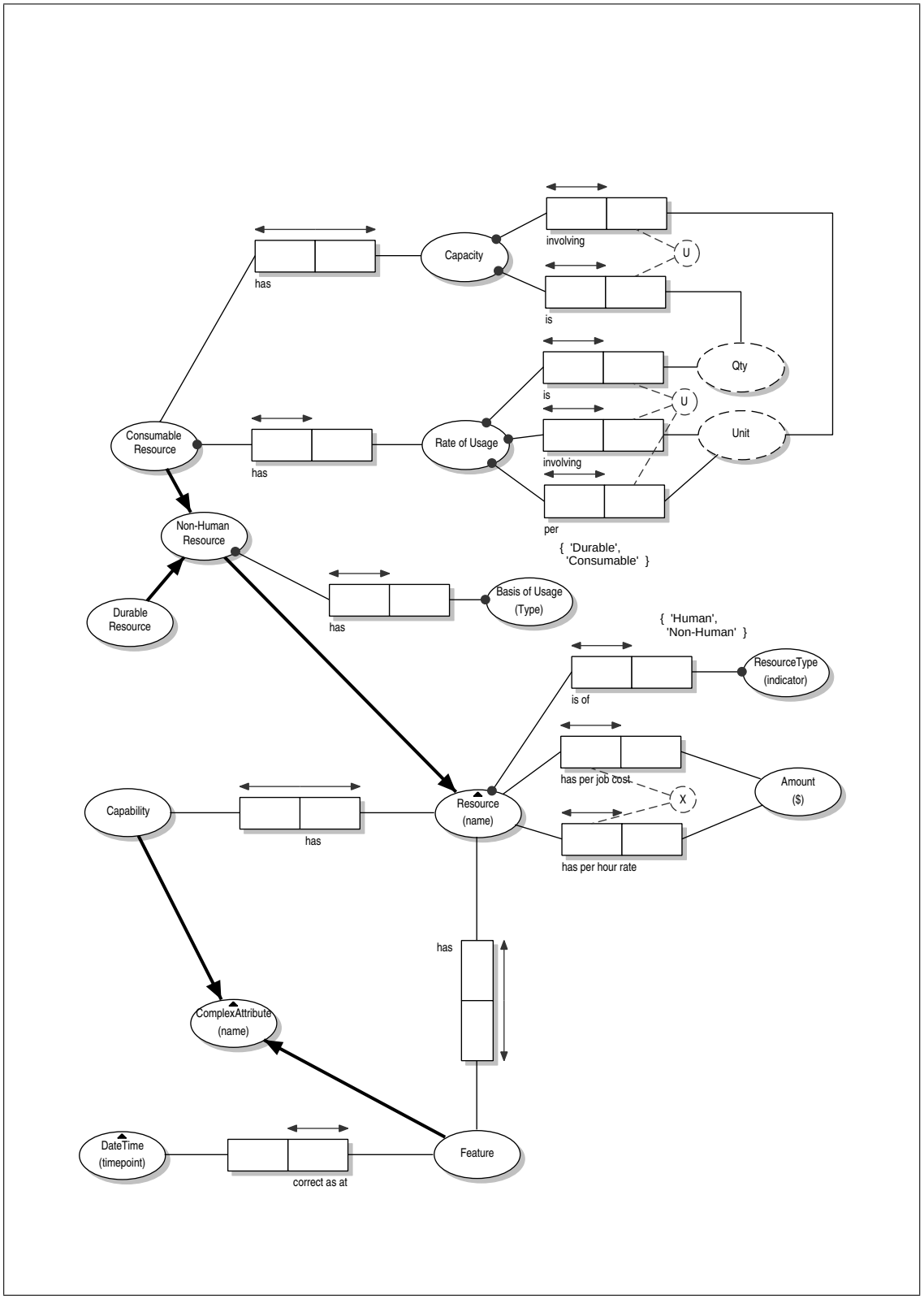

Figure 1: Object Role Diagram for Workflow Resources – Non-Human Resources

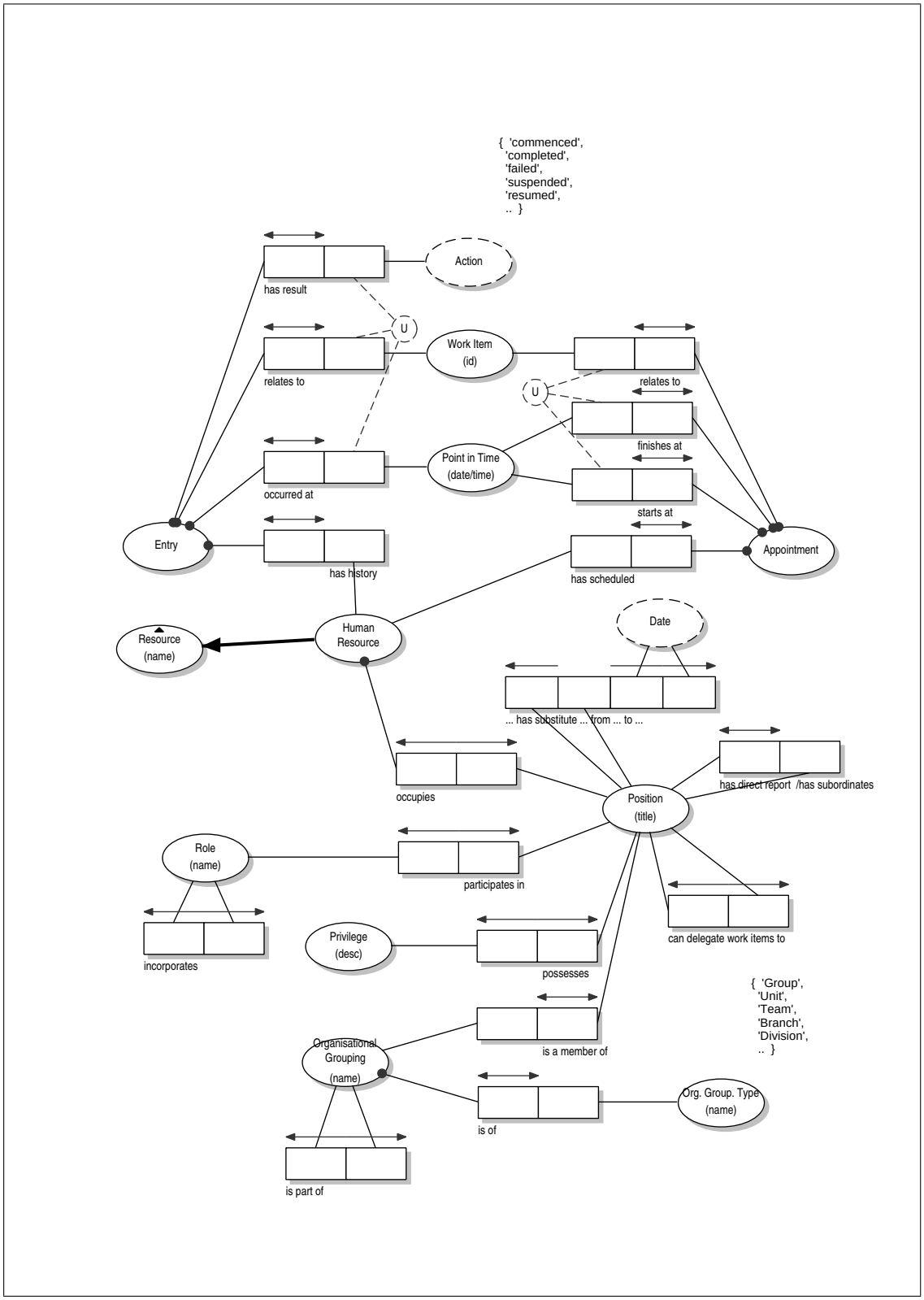

Figure 2: Object Role Diagram for Workflow Resources – Human Resources

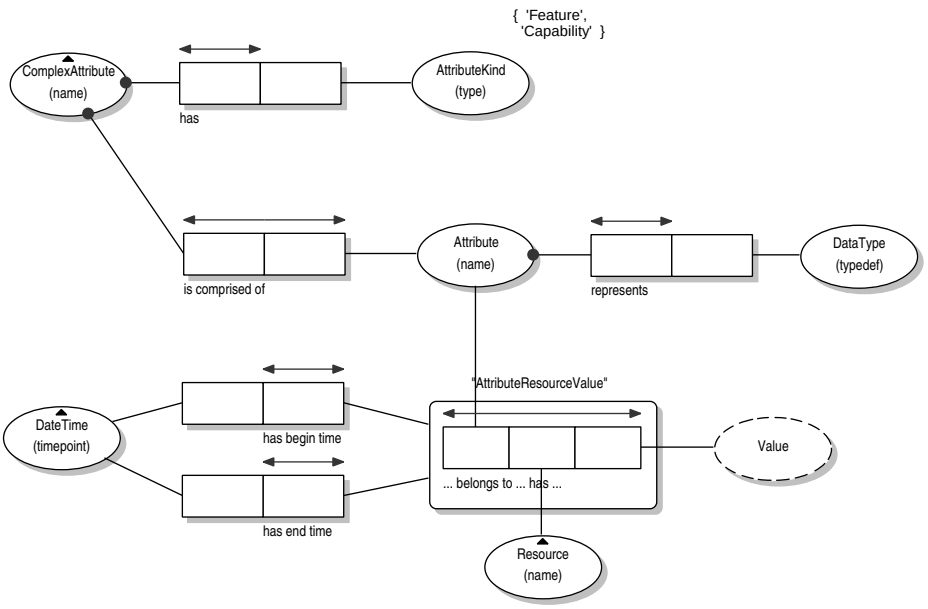

Figure 3: Object Role Diagram for Workflow Resources – Complex Attributes

2.2 Workflow structure

Before we describe the actual Resource Patterns in detail, we first present a standard set of definitions for the various components of a workflow system that we will utilise throughout this paper.

A *workflow* or *workflow model* is a description of a business process in sufficient detail that it is able to be directly executed by a *workflow management system*. A *workflow model* is composed of a number of *tasks* which are connected in the form of a directed graph. An executing instance of a workflow model is called a *case* or *process instance*. There may be multiple cases of a particular workflow model running simultaneously, however each of these is assumed to have an independent existence and they typically execute without reference to each other.

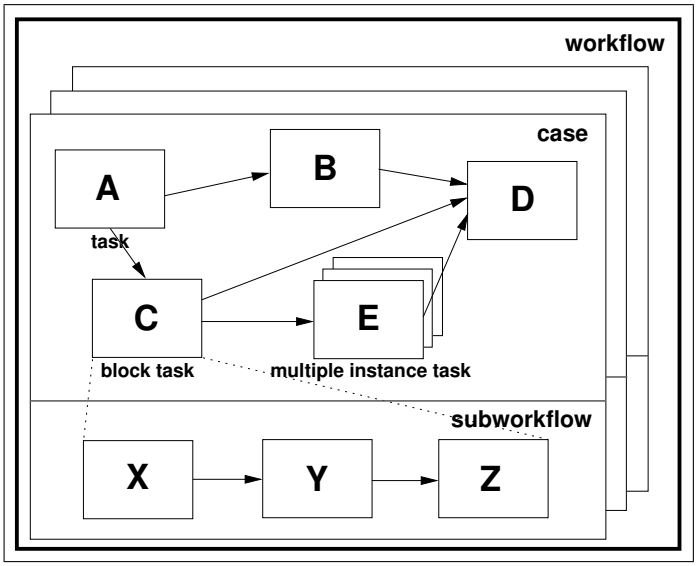

Figure 4: Components of a workflow

There is usually a unique first task and a unique final task in a workflow. These are the tasks that are first to run and last to run in a given workflow case.

A *task* corresponds to a single unit of work. Four distinct types of task are denoted: *atomic*, *block*, *multi-instance* and *multiple-instance block*. We use the generic term *components* of a workflow to refer to all of the tasks that comprise a given workflow model.

An *atomic task* is one which has a simple, self-contained definition (i.e. one that is not described in terms of other workflow tasks) and only one instance of the task executes when it is initiated.

A *block task* is a complex action which has its implementation described in terms of a *sub-workflow*. When a *block task* is started, it passes control to the first task(s) in its corresponding *sub-workflow*. This *sub-workflow* executes to completion and at its conclusion, it passes control back to the *block task*. E.g. block task C is defined in terms of the sub-workflow comprising tasks, X, Y and Z.

A *multiple-instance task* is a task that may have multiple distinct execution instances running concurrently within the same workflow case. Each of these instances

executes independently. Only when a nominated number of these instances have completed is the task following the multiple instance task initiated.

A *multiple-instance block task* is a combination of the two previous constructs and denotes a task that may have multiple distinct execution instances each of which is block structured in nature (i.e. has a corresponding *sub-workflow*).

The control flow between tasks occurs via the *control channel* which is indicated by a solid arrow between tasks.

Each invocation of a task that executes is termed a *work item*. Usually there is one work item initiated for each task in a given case however in the case of a *multiple-instance task*, there may be several associated work items that are created when the task is initiated. Similarly, where a task forms part of a loop, a distinct work item is created for each iteration.

In general a work item is directed to a resource for execution (although a resource is not required to undertake automatic tasks). There are a variety of ways by which this may be achieved which will be discussed subsequently.

A task may initiate one or several tasks when it completes (i.e. when a work item corresponding to it completes). This is illustrated by an arrow from the completing task to the task being initiated e.g. in Figure 4, task B is initiated when task A completes. This may also occur conditionally and where this is the case, the edge between tasks indicates the condition that must be satisfied for the subsequent task to be started.

2.3 Work distribution to resources

Of particular interest from a resource perspective is the manner in which work items are advertised and ultimately bound to specific resources for execution. Figure 5 illustrates the lifecycle of a work item in the form of a state transition diagram from the time that a work item is created through to final completion or failure. It can be seen that there are a series of potential states that comprise this process.

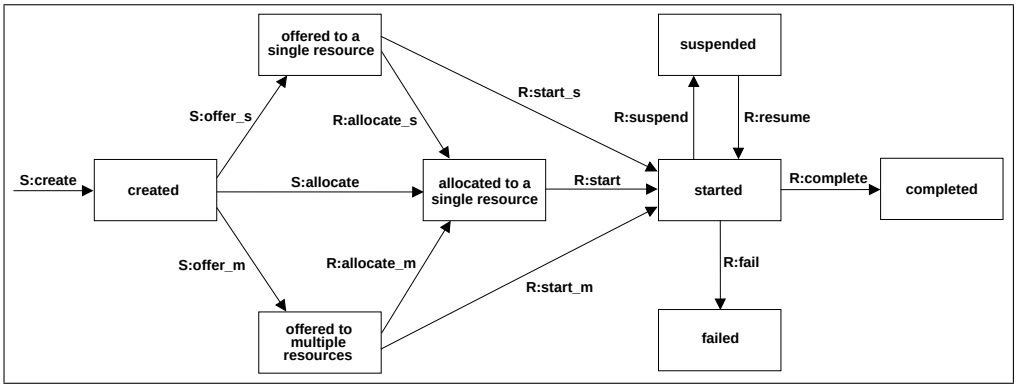

Figure 5: Work Item Lifecycle

Initially a work item comes into existence in the *created* state. This indicates that the preconditions required for its enablement have been satisfied and it is capable of being executed. At this point however, the work item has not be allocated to a resource for execution and there are a number of possible paths through these states

for individual work items. Each edge within this diagram is prefixed with either an S or an R indicating that the transition is initiated by the workflow system or resource respectively.

Transitions from the *created* state are typically initiated by the workflow system. They centre on the activity of making resources aware of work items that require execution. This may occur in one of three distinct ways denoted by the subsequent states. A work item may be *offered to a single resource* meaning that the workflow system informs exactly one resource about the availability of a work item. It may do this by sending a message to the resource or adding the work item to the list of available work items that the resource can view. Inherent in this is the notion of the system selecting a specific resource to which the work item should be advertised. This may occur in a variety of different ways – the process model may include specific directives about the identity of the resource to which a given work item should be directed or it may be based on more general requirements such as utilising the least busy, cheapest or most appropriately qualified resource. In each of these situations, there is the need to determine which resources are suitable and available to undertake the work item and then to rank them and select the most appropriate one.

An alternative to this course of action is indicated by the state *offered to multiple resources*, where the system informs multiple resources of the existence of a work item. Again the notion of resource selection applies, however in this case, the system informs all suitable resources of the work item. It does not attempt to identify which of them should undertake it.

The *allocated to a single resource* state denotes a work item which a specific resource has committed to executing at some time in the future. A work item may progress to this state either because the workflow system pre-emptively allocates newly created work items to a resource or because a resource volunteers to undertake a work item that has been offered.

Note that the work item lifecycle illustrated in Figure 5 assumes that a work item is undertaken by a single resource, thus there is not a state which corresponds to "allocated to multiple resources".

Subsequent states in the work distribution model are *started*, which indicates that a resource has commenced executing the work item, *suspended* which denotes that the resource has elected to cease execution of the work item for a period, but does intend to continue working on it at a later time, *failed* which identifies that the work item cannot be completed and that the resource will not work on it any further and *completed* which identifies a work item that has been successfully executed to completion.

The Patterns presented in this paper are intended to be language independent and do not assume a concrete syntax. In the absence of an agreed workflow model, the aim is to define them in a form that ensures they are applicable to the broadest possible range of workflow systems.

3 Resource Patterns

In this section, we discuss the range of Resource Patterns that have been identified in workflow systems. These are grouped into a series of specific categories depending on the specific focus of the individual Pattern.

3.1 Creation Patterns

Creation Patterns correspond to limitations on the manner in which a work item may be executed. They are specified at design time, usually in relation to a workflow task, and serve to restrict the range of resources that can undertake work items that correspond to the task. They also influence the manner in which a work item can be matched with a resource that is capable of undertaking it.

The essential rationale for Creation Patterns is that they provide a degree of clarity about how a work item should be handled after creation during the offering and allocation stages prior to it being executed. This ensures that the operation of a workflow process conforms with its intended design principles and operates as efficiently and deterministically as possible.

In terms of the work item lifecycle, Creation Patterns come into effect at the time a work item is created. This state transition occurs at the beginning of the work item lifetime and is illustrated by the bold arrow in Figure 6.

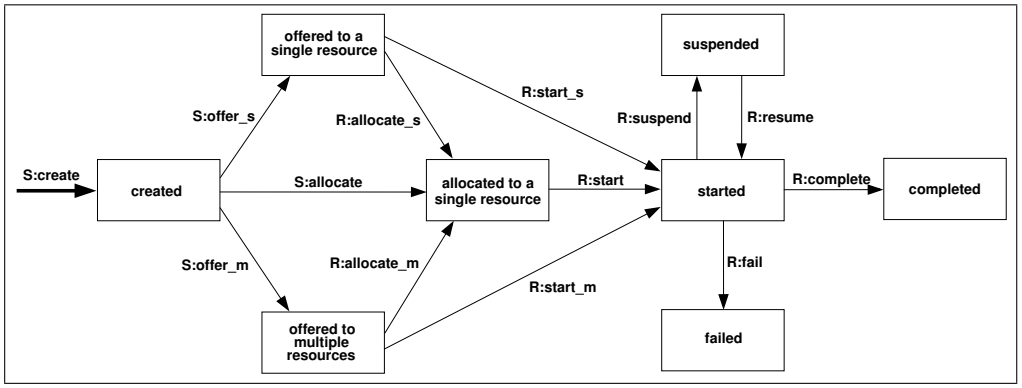

Figure 6: Creation Patterns

As Creation Patterns are specified at design time, they usually form part of the process model that defines a workflow process.

1. Pattern R-DA (Direct Allocation)

Description The ability to specify at design time the identity of the resource that will execute a task.

Example

- The *Fix Bentley* task must only be undertaken by *Fred*.

Related Patterns R-D (Delegation).

Motivation Direct allocation offers the ability for a workflow designer to precisely specify the identity of the resource to which instances of each task will be allocated at runtime. This is particularly useful where it is known that a task can only be effectively undertaken by a specific resource as it prevents the problem of unexpected or non-suitable resource allocations arising at runtime by ensuring work items are routed to specific resources, a feature that is particularly desirable for critical tasks.

Implementation Most workflow engines offer some form of support for direct allocation of tasks to specific resources. In most cases, the allocation is to a single

resource, however Staffware allows a work item to be allocated to a series of specific resources (achieved by specifying the names of multiple resources for potential allocation) and at runtime, the work item is routed to all of these resources and each of them is required to release it before the work item can be deemed to have finished and the case can progress.

Issues One of the main drawbacks of this approach to resource allocation is that it effectively defines a static binding of all work items associated with a task to a single resource. This removes much of the advantage associated with the use of workflow technology for managing work distribution as the workflow engine is offered little latitude for optimising the allocation of work items in this situation.

Solutions There is no real solution to this problem although the use of deadline and escalation mechanisms offer ways of ensuring that situations are detected where a specific resource becomes overloaded and cannot deal with its assigned workload in a reasonable timeframe.

2. Pattern R-RBA (Role-Based Allocation)

Description The ability to specify at design time that a task can only be executed by resources which correspond to a given role.

Example

- Instances of the *Approve Travel Requisition* task must be executed by a *Manager*.

Related Patterns R-RA (Authorisation), R-OR (Organisational Allocation).

Motivation Perhaps the most common approach to work item allocation within workflow systems, role-based allocation offers the means for the workflow engine to route work items to suitably qualified resources at run-time. The decision as to which resource actually receives a given work item is deferred until the moment at which it becomes "runnable" and requires a resource allocation in order for it to proceed. The advantage offered by role-based allocation (over other work item allocation schemes) is that roles can be defined for a given workflow process that define the various classes of available resources to undertake work items. Task definitions within the process model can nominate the specific role to which they should be routed, however the actual population of individual roles does not need to occur until run-time.

Implementation All of the workflow systems examined support role-based allocation in some form. Generally roles serve as groupings of resources with similar characteristics or authority and provides a means of decoupling the routing of a work item from that of resource management. The most restrictive approach to role definition occurs in Staffware where only one resource can be identified for each role although it is possible to specify multiple roles when defining the routing of a work item. WebSphere MQ allows multiple resources to be specified for each role and also multiple roles to be used when routing a work item. iPlanet supports roles in a similar way although the actual mechanism used for work item distribution takes the form of an expression which includes the various roles rather than simply listing the roles to which the work item will be forwarded. COSA also uses roles as a grouping mechanism for resources and allows them to be used as a routing mechanism for work items, however where a work item is routed to multiple resources, it appears on a

shared (group) work queue rather than being replicated on the work lists of individual resources. COSA provides support for explicitly representing quite complex organisational structures and work allocation mechanisms by allowing role, organisational and authorisation hierarchies to be distinctly modelled and drawn together where required in the distribution functions for individual work items. FLOWer supports multiple users per role and allows a user to play different roles in distinct cases. Roles serve as the main basis of work item distribution although resources have a reasonable degree of autonomy in selecting the work items (and cases) that they will undertake rather than having work items directly assigned to them.

Issues In some workflow systems, the concepts of roles and groups are relatively synonymous. Roles serve as an abstract grouping mechanism (i.e. not just for resources with similar characteristics or authority, but also for identification of organisational units e.g. teams, departments etc.) and provide a means of distributing work across a number of resources simultaneously. One difficulty that arises with this use of roles occurs where the intention is to offer a work item to several resources with the expectation that they will all work on it.

Solutions Staffware provides support for this style of work allocation. It operates in much the same way as role-based allocation with groups being identified within the workflow system consisting of several resources. Individual resources may belong to more than one group (unlike the situation with roles) and a task within the process model can be specified as requiring routing to a specific group at run-time. However the operation of group-based allocation differs from role-based allocation at run-time with a work item that is allocated to a group being visible to all of the resources in the group and not specifically (and privately) assigned to one of them during the allocation process. Group-based allocation is non-deterministic with respect to resources and the work item is ultimately allocated to the first resource in the group that commences work on it. From this point, none of the other resources in the group can execute it although it remains in the work queue of all of the resources until it has been completed.

None of the other workflow engines examined provide support for this approach to work allocation.

3. Pattern R-FBA (Deferred Allocation)

Description The ability to defer specifying the identity of the resource that will execute a task until runtime.

Example

- During execution of a case, instances of the *Assess Damage* task will be executed by the resource named in the *next_resource* field.

Related Patterns R-DA (Direct Allocation), R-RBA (Role-based Allocation)

Motivation Deferred allocation takes the notion of indirect resource allocation one step further and allows the workflow designer to defer the need to identify the resource for a specific task (or work items corresponding to the task) until run-time. One means of achieving this is to nominate a data field from which the identity of the resource to which a work item should be routed can be determined at runtime. The resource identity can be changed dynamically during workflow execution by updating

the value of the data field, thus varying the resource allocation of future work items which are contingent on it.

Implementation This approach to resource allocation is generally achieved by associating the field name which will contain the resource identity with the task at design time. In order to facilitate this, the field needs to be a data element within the scope of the task at runtime – usually a case level data element. It is possible that more than one data element (and hence more than one resource) could be taken into account when deciding on the allocation at runtime. Both Staffware and WebSphere MQ directly support this Pattern.

Issues Two significant issues arise in implementing this Pattern:

- Determining whether the value in the data field relates to a specific resource, group or role name. This determination is important as it varies the approach taken to resource allocation.
- Ensuring that the data field contains a valid resource name.

Solutions The first of these issues is usually addressed by ensuring that the names used for specific resources, groups and roles are disjoint. This means that a name cannot be used in more than one context and hence there is no potential for ambiguity at runtime.

The second issue is more problematic as a data element can potentially contain any value and there is no means of ensuring that it corresponds to an actual resource in the workflow or to specify the action to take when the resource name is invalid.

4. Pattern R-RA (Authorisation)

Description The ability to specify the range of resources that are authorised to execute a task.

Example

- Only the *Finance Director, Senior Loans Manager* and *Financial Accountant* are authorised to execute instances of the *Finalise Loan* task.

Related Patterns R-CBA (Capability-based Allocation), R-SOD (Separation of Duties)

Motivation Through the specification of authorisations on task definitions, it is possible to define a security framework over a workflow implementation that is independent of the way in which work items are actually routed at runtime. By defining authorisations on individual tasks, the range of resources that can access details of a work item or execute it can be restricted. This ensures that unexpected events that may arise during workflow execution (e.g. work item delegation by a resource or reallocation to another resource outside of the usual workflow operation) do not lead to unexpected resources being able to undertake work items.

Implementation COSA is the only workflow engine observed that implements the notion of task authorisation as a distinct concept to that of task distribution. It treats authorisation and distribution of tasks in the same way in the design time model and provides facilities for defining the resources, groups and roles that are authorised to execute a task and also those to which it can be allocated. FLOWer uses roles as the

main basis for case and work item distribution. Roles are organised as hierarchies and only resources that directly (or indirectly) possess a required role are able to view and execute a specific work item.

Issues The range of resources that are authorised to undertake a task may not correspond to those to which it could be assigned based on the current resource pool within the workflow system.

Solutions COSA provides a solution to this scenario as follows:

- Where a resource is allocated a work item that it is not authorised to execute, the work item will appear in its work list, but the resource cannot execute it. The resource can however reassign it to another resource that may be able to execute it.
- Where a resource is authorised to undertake a given task, but the task is not able to be distributed to the resource (i.e. the distribution rules for the task preclude it from being allocated to the resource), work items corresponding to the task will never appear in the work list for the resource but it is able to execute them if they are directly allocated to it by other resources.

5. Pattern R-SOD (Separation of Duties)

Description The ability to specify that two tasks must be allocated to different resources in a given workflow case.

Example

- Instances of the *Countersign cheque* task must be allocated to a different resource to that which executed the *Prepare cheque* task in a given workflow case.

Related Patterns R-CBA (Capability-based Allocation), R-RF (Retain Familiar), R-CH (Case Handling)

Motivation Separation of duties allows for the enforcement of audit controls within the execution of a workflow case. This ensures that a work item cannot be executed by the same resource that executed another work item within the same case.

Another use of this Pattern arises with workflow engines that support multiple task instances. In this situation, the degree of parallelism that can be achieved when a multiple instance task is executed can be maximised by specifying that as far as possible no two task instances can be executed by the same resource.

Implementation This Pattern can be implemented in a number of distinct ways:

- WebSphere MQ and FLOWer provide the ability to specify at task level, a link with another (preceding) task. At runtime, the work item corresponding to the task cannot be allocated to the same resource as that which undertook the previous instance of the work item corresponding to the linked task.
- iPlanet utilises the concepts of *linked activities* which allow the data elements of two distinct tasks to be shared and *evaluate methods* which define how the work items for a given task will be allocated to the various resources within the

workflow system. For a given task, a custom *evaluate method* is constructed which ensures it cannot be allocated to the same resource that undertook the (preceding) instance of the task to which it was linked.

- COSA allows the effect of separation of duties to be achieved through the use of access rights which restrict the resource which undertook the preceding work item from executing the latter.

Issues None identified.

Solutions N/A.

6. Pattern R-CH (Case Handling)

Description The ability to allocate the work items within a given workflow case to the same resource.

Example

- All tasks in a given case of the *Prepare defence* process are allocated to the same *Legal Advisor*.

Related Patterns R-RF (Retain Familiar)

Motivation Case handling is a specific approach to work distribution that is based on the premise that all work items in a given case are so closely related that they should all be undertaken by the same resource. The identification of the specific resource occurs when a case (or the first work item in a case) requires allocation.

Case handling may occur on either a "hard" or "soft" basis i.e. work items within a given case can be allocated exclusively to the same resource which must complete them all or alternatively it can serve as a guide to how work items within a given case should be routed with an initial resource being identified as having responsibility for all work items and subsequently delegating them to other resources or allowing them to nominate work items they would like to complete.

Implementation This approach to work distribution is not generally supported by workflow systems. Of those examined only FLOWer (which describes itself as a case handling system) provides direct support.

Issues None identified.

Solutions N/A.

7. Pattern R-RF (Retain Familiar)

Description Where several resources are available to undertake a work item, the ability to allocate a work item within a given workflow case to the same resource that undertook a preceding work item.

Example

- If there are several suitable resources available to undertake the *Prepare Match Report* work item, it should be allocated to the same resource that undertook the *Umpire Match* task in a given workflow case.

Related Patterns R-CH (Case Handling), R-CE (Chained Execution)

Motivation Allocating a work item to the same resource that undertook a previous work item is a common means of expediting a workflow case. As the resource is already aware of the details of the case, it saves familiarisation time at the commencement of the work item. Where the two work items are sequential, it also offers the opportunity for minimising switching time as the resource can commence the latter work item immediately on completion of the former.

This Pattern is a more flexible version of the Case Handling (R-CH) Pattern discussed earlier. It only comes into effect when there are multiple resources available to undertake a given work item and where this occurs, it favours the allocation of the work item to the resource that undertook a previous work item in the case. Unlike the Case Handling Pattern (which operates at case level), this Pattern applies at the work item level and comes into play when a work item is being allocated to a resource.

The Chained Execution Pattern is a specialised form of this Pattern designed to expedite the completion of a case by automatically starting subsequent work items once the preceding work item is complete.

Implementation Not surprisingly, this Pattern enjoys wider support than the Case Handling Pattern. WebSphere MQ allows individual work items to be allocated to the same resource that started another work item in a case or to the resource that started the case itself. FLOWer provides a facility in the design time workflow model to enforce that a task must be executed by the same resource as another specified task in the case. COSA does the same thing using a customised distribution algorithm for a specific work item that requires it to have the same executor as another work item in the case. Similarly iPlanet achieves the same result using the linked user concept which requires two work items to be executed by the same resource.

Issues None identified.

Solutions N/A.

8. Pattern R-CBA (Capability-based Allocation)

Description The ability to offer or allocate instances of a task to resources based on specific capabilities that they possess.

Example

- Instances of the *Airframe Examination* task should be allocated to an *Engineer* with an aeronautics degree, an Airbus in-service accreditation and more than 10 years experience in Airbus servicing.

Related Pattern R-RA (Authorisation)

Motivation Capability-based allocation provides a mechanism for offering or allocating work items to resources through the matching of specific requirements of work items with the capabilities of the potential range of resources that are available to undertake it. Capabilities are evaluated at run-time during work item offering or allocation. Depending on whether the work allocation strategy is push or pull-based, the actual allocation process can be initiated by the workflow system or the resource. In the former situation, the workflow system determined the most appropriate resource(s) to which a work item should be routed. In the latter, a resource initiates a search for an unallocated work item(s) which it is capable of undertaking.

Implementation Capability-based allocation is based on the specification of capabilities for individual resources. Capabilities generally take the form of attribute - value pairs. A dictionary of capabilities can be defined in which individual capabilities have a distinct name and the type and potential range of values that each capability may take can also be specified. Classes of resources then indicate which capabilities are relevant to (and recorded for) individual resource instances. Similarly, tasks can also have capabilities recorded for them.

The actual allocation process is often based on the specification of competency functions which are executed at runtime and determine how individual work items can be matched with suitable resources. These may be arbitrarily complex in nature depending on the range of capabilities that require matching between resources and work items and the approach that is taken to ranking the matches that are achieved in order to select the most appropriate resource to undertake a given work item. Both COSA and iPlanet implement capability-based allocation through the use of user-specified competency functions that form part of the process model. In both cases, the strategy is push-based.

The example shown below in Figure 7 illustrates capability-based allocation with the capability function matching a work item to a resource on the basis of both resource capabilities and work item attributes.

FLOWer uses case queries to determine which cases can be allocated to a specific resource. These can include data elements relating to both the case and the individual resource.

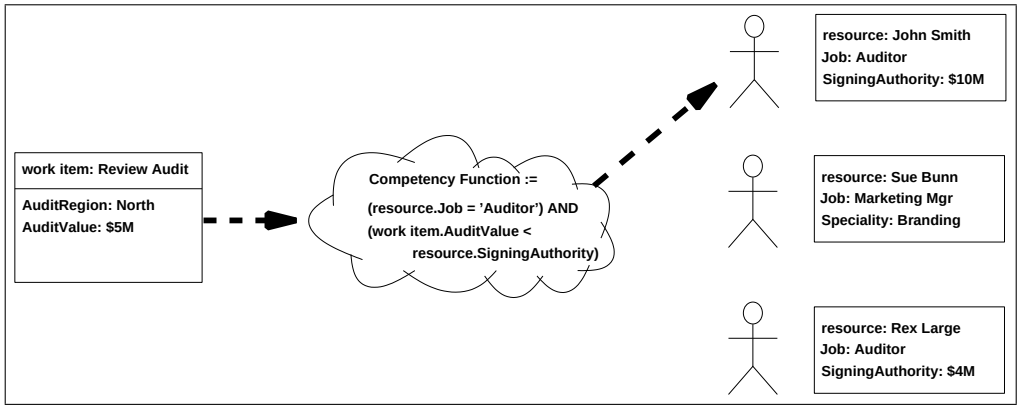

Figure 7: Capability-based Allocation

Issues 1. One consideration with push-based capability allocation is that it is possible for capability functions to identify more than one possible resource to which a work item may be assigned. Where this occurs, the work item may either be offered to multiple resources or assigned to one of the identified resources on a random basis. It is also possible for the capability function to return an empty set of possible resources.

2. In the case of pull-based capability allocation, it is possible for a resource to identify more than one work item that it is capable of undertaking.

Solutions 1. The first of these issues is not necessarily a problem although it may result in sub-optimal resource allocations. It can be avoided through more precise

definition of capability functions. As an example, if the intention of the competence function in Figure 7 was to allocate the task to a single auditor, then a ranking function (e.g. `minimum`) should be included in the competence function to ensure only a single resource is returned.

The second issue can be avoided by testing whether the competence function returns an empty set and if so, assigning a default value for the resource. COSA provides the `ifnull` operator for this purpose. iPlanet allows its evaluate methods to be arbitrarily complex to cater for situations such as this.

2. This should not generally result in difficulties. Under a pull-based allocation strategy, resources should anticipate the possible return of multiple work items. In some systems, it is possible for a resource to query matching work items without committing to executing them.

9. Pattern R-HBA (History-based Allocation)

Description The ability to offer or allocate work items to resources on the basis of their previous execution history.

Example

- Allocate the *Finalise heart bypass* task to the *Surgeon* who has successfully completed the most of these tasks.
- Allocate the *Core extraction* task to the *drill* that has the lowest utilisation over the past 3 months.

Related Pattern R-RF (Retain Familiar)

Motivation History-based allocation involves the use of information on the previous execution history of resources when determining which of them a work item should be offered or allocated to. This is an analogue to common human experience when determining who to distribute a specific work to which considers factors such as who has the most experience with this type of work item or who has had the least numbers of failures when tackling similar tasks.

Implementation None of the workflow engines examined provide direct support for history-based allocation, however for some of them it is possible to achieve some of the benefits of this approach by extending specific workflow models. There are essentially two methods of facilitating this:

- Extend the details maintained by individual resources on their work history and utilise this information when allocating work items.
- Extract details of work performance from the workflow log and incorporate this into the allocation process.

COSA provides facilities for the second method via customised distribution functions utilising the services of an external program to mine the workflow log. iPlanet is able to support both options using extended user profiles, modified task definitions to update user histories and customised distribution functions.

Issues The main difficulty with facilitating this allocation strategy is that it places an additional processing overhead on workflow execution in order to maintain user execution details in a format that can be used when distributing work items.

Solutions There is no immediate solution to this issue. Maintaining user execution profiles in a useful form requires additional processing to gather the required information and additional storage to maintain it. Where the allocation strategy is not directly supported by the workflow engine, modifications are required to the workflow process in order to achieve this. The only recommendation that can be made in this situation is to gather and manage the least amount of execution history for each resource that is required to facilitate the chosen work distribution strategy.

10. Pattern R-OA (Organisational Allocation)

Description The ability to offer or allocate instances of a task to resources based their position within the organisation and their relationship with other resources.

Example

- The *Review Audit* work item must be allocated to a *Partner* resource.
- The Authorise Expenditure work item must be allocated to the *Manager* of the resource that undertook the *Claim Expenditure* work item in a given case.

Related Patterns R-RBA (Role-based Allocation), R-RA (Authorisation)

Motivation Most workflow systems provide some degree of support for modelling the organisational context in which a given process operates. This is an important aspect of process modelling and implementation as many work allocation decisions are made in the context of the organisational structure and the relative position of individual resources both in the overall hierarchy and also in terms of their relationships with other resources. The ability to capture and emulate these types of work allocation strategies are an important requirement if workflow systems are to provide a flexible and realistic basis for managing work in an organisational setting.

Implementation The degree of support for this Pattern varies widely. Staffware does not incorporate an organisational model and only provides support for role and group based work allocation. iPlanet is similar and only provides for role-based allocation, however it lacks any form of integrated organisational model. FLOWer extends the notion of role-based allocation and provides limited support for organisational structure in the form of a role hierarchy. WebSphere MQ supports a hierarchical organisational model and in addition to direct and role-based allocation, it allows organisational relationships such as coordinator of role, member of organisational unit, manager of organisation and starter of activity to be for work item allocation. COSA also incorporates a hierarchical organisational model and supports work allocation based either on roles or characteristics of the organisational model (e.g. supervisor, group membership).

Issues None identified.

Solutions N/A.

11. Pattern R-AE (Automatic Execution)

Description The ability for an instance of a task to execute without needing to utilise the services of a resource.

Example

- The *End of Day* work item executes without needing to be allocated to a resource.

Related Pattern None

Motivation Not all tasks within a workflow need to be executed under the auspices of an actual human resource, some are able to execute independently once the specified enabling criteria are met.

Implementation Staffware, FLOWer, COSA and iPlanet all provide facilities for defining tasks which can run automatically within the context of the workflow without requiring allocation to a resource.

Issues None identified.

Solutions N/A.

3.2 Push Patterns

Push Patterns characterise situations where newly created work items are proactively offered or allocated to resources by the workflow system. These may occur indirectly by advertising work items to selected resources via a shared work list or directly with work items being allocated to specific resources. In both situations however, it is the workflow engine that takes the initiative and causes the distribution process to occur. Figure 8 illustrates (as bold arcs) the potential state transitions associated with push-based distribution:

- `S:offer_s` corresponds to a work item being offered to a single resource.
- `S:offer_m` corresponds to a work item being offered to multiple resources (one of which will ultimately execute it).
- `S:allocate` corresponds to a work item being directly allocated to a resource immediately after it has been created.

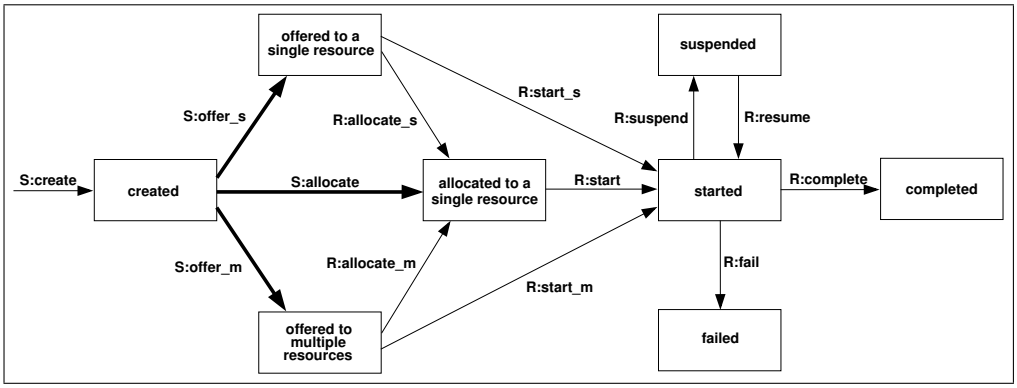

Figure 8: Push Patterns

Nine push Patterns have been identified. These divide into three distinct groups. The first three Patterns identify the actual manner of work distribution - whether the workflow system offers the work item to a single resource, to multiple resources

or whether it allocates it directly to a single resource.[2] These Patterns correspond directly to the bold arcs in Figure 8.

The second group of Patterns relate to the means by which a resource is selected to undertake a work item where there are multiple possible resources identified. Three possible strategies are described – random allocation, round robin allocation and shortest queue. These Patterns correspond to alternate ways in which the `S:offer_s` and `S:allocate` transitions may occur.

The final three Patterns identify the timing of the distribution process and in particular the relationship between the availability of a work item for offering/allocation to resources and the time at which it commences execution. Three variants are possible – work items are offered/allocated before they have commenced (early distribution), after they have commenced (late distribution) or the two events are simultaneous (distribution on enablement). These Patterns do not have a direct analogue in Figure 8 but relate to the time at which the `S:offer_s`, `S:offer_m` and `S:allocate` transitions may occur with respect to the work item's readiness to be executed (i.e. already started, immediate start or subsequent start).

12. Pattern R-DBOS (Distribution by Offer – Single Resource)

Description The ability to offer a work item to a selected individual resource.

Example

- The *Prepare defense* work item is offered to a selected *Barrister*.

Related Patterns R-DA (Direct Allocation), R-DBOM (Distribution by Offer – Multiple Resources), R-DBAS (Distribution by Allocation – Single Resource)

Motivation Once a work item has been created and it has been determined that the work item should be offered to a single distinct resource for potential execution, a means of actually informing the resource of the pending work item is required. The mechanism should notify the resource that a work item exists that it may wish to undertake, however it should not commit the resource to its execution and it should not advise any other resources of the potential work item.

This Pattern directly corresponds to the state transition denoted by arc `S:offer_s` in Figure 8.

Implementation Of the workflow engines examined, only iPlanet directly supports the ability to offer a work item to a single resource without the resource being committed to executing the work item. COSA provides a close analogy to this concept in that it allows a resource to reject a work item that has been allocated to it and placed on its work queue. When this occurs, the work item goes through a subsequent reallocation process, ultimately resulting in it being assigned to a different resource.

Issues None identified.

Solutions N/A.

13. Pattern R-DBOM (Distribution by Offer – Multiple Resources)

[2] In this paper, we assume a one-to-one correspondence between resources working on a work item and work items being processed. In other words, resources cannot work on different work items simultaneously and it is not possible that multiple resources work on the same work item. In Section 3.7 we will discuss and relax this requirement slightly.

Description The ability to offer a work item to a group of selected resources.

Example

- The *Sell portfolio* work item is offered to multiple *Stockbrokers*.

Related Patterns R-RBA (Role-based Allocation), R-CBA (Capability-based Allocation), R-DBOS (Distribution by Offer – Single Resource), R-DBAS (Distribution by Allocation – Single Resource)

Motivation Offering a work item to multiple resources is the workflow analogy to the act of "calling for a volunteer" in real life. It provides a means of advising a suitably qualified group of resources that a work item exists but leaves the onus with them as to who actually commits to undertaking the activity.

This Pattern directly corresponds to the state transition denoted by arc `S:offer_m` in Figure 8.

Implementation Several workflow engines support the notion of work groups and allow work items to be allocated to them. A work group is a group of resources with a common organisational focus. When a work item is allocated to the group, each of the members of the group is advised of its existence, but until one of them commits to starting it and advises the workflow engine of this fact, it remains on the work queue for each of the resources.

There are several possibilities for resources being advised of group work items – they may appear on each of the individual resource's work queues, each resource may have a distinct work queue for group items on which they may appear or all resources in a work group may have the ability to view a shared group work queue in addition to their own dedicated work queue[3].

Different workflow engines handle the offering of a work item to multiple resources in different ways:

- WebSphere MQ treats work items offered to multiple resources in the same way as work items allocated to a specific resource and they appear on the work list of resources to whom they are offered. When a multiply-offered work item is accepted by one of the resources to which it is offered, it is removed from the work lists of all other resources.

- Staffware and COSA support the concept of distinct user specific work queues and group work queues. Where a multiply-offered work item is accepted by a resource, it remains on the group work list but is not able to be selected for execution by other resources.

- iPlanet supports distinct work queues for offered and queued (i.e. allocated) work items. Once a multiply-offered work item has been accepted by a resource, it is removed from all offered work queues and only appears on the queued list for the resource which has accepted it.

Issues None identified.

Solutions N/A.

[3]Note that it is impossible to actually differentiate between the last two alternatives.

14. Pattern R-DBAS (Distribution by Allocation – Single Resource)

Description The ability to directly allocate a work item to a specific resource for execution.

Example

- The *Cover Comalco AGM* work item should be allocated to the *Finance Sub-editor*.

Related Patterns R-DA (Direct Allocation), R-DBOS (Distribution by Offer – Single Resource), R-DBOM (Distribution by Offer – Multiple Resources)

Motivation Distribution by allocation to a single resource corresponds to the notion of the workflow engine directly assigning a work item to a resource without first offering it to other resources or querying whether the resource will undertake it. This approach to work distribution is also known as "heads down" processing as it offers the resource little or no input in the work that they are allocated and the main focus is on maximising work throughput by keeping the resource busy. In many implementations, resources are simply allocated a new work item once the old one is completed and they are not offered any insight into what work items might lay ahead for them.

This Pattern directly corresponds to the state transition denoted by arc `S:allocate` in Figure 8.

Implementation Where a specific resource has been identified during the course of work item distribution, this is the standard means of allocating a work item to a resource. It is done pre-emptively by the workflow engine and necessitates that the resource actually execute the work item unless it has recourse to a means of rejecting it. Staffware, WebSphere MQ, FLOWer, COSA and iPlanet all support direct allocation of work items to resources.

Issues None observed.

Solutions N/A.

15. Pattern R-RMA (Random Allocation)

Description The ability to offer or allocate work items to suitable resources on a random basis.

Example

- The *Judge case* work item is allocated to a *Magistrate* on a random basis.

Related Patterns None.

Motivation Random allocation provides a non-deterministic mechanism for allocating work items to resources in workflow systems. Once the possible range of resources that a work item can be allocated to have been identified at runtime, one of these is selected at random to execute the work item.

Implementation Of the systems examined, only COSA provides direct support for work allocation on a random basis using the `random` operator which forms part of the user/group language. This is a scripting language which allows relatively complex work allocation rules to be specified. Similarly, iPlanet allows the work distribution

algorithm to be extended programmatically although there is no direct support for random allocation within the product.

Issues None identified.

Solutions N/A.

16. Pattern R-RRA (Round Robin Allocation)

Description The ability to allocate a work item to available resources on a cyclic basis.

Example

- Work items corresponding to the *Umpire Match* task are allocated to each available *Referee* on a cyclic basis.

Related Patterns None.

Motivation A round robin allocation strategy provides a means of ensuring that all resources are allocated the same number of work items.

Implementation None of the workflow engines examined provide direct support for round robin allocation. However both COSA and iPlanet provide facilities for specifying custom allocation strategies for workflow tasks. In the case of COSA, a custom distribution algorithm can be specified (incorporating an external program) that implements round robin allocation. As the total available working time for each user can be specified (as a percentage between 0 and 100%), there is the opportunity to establish a relatively fair basis for round robin allocation.

For iPlanet, it is possible to develop an Evaluate method that achieves a similar result.

Issues By its nature, round robin allocation requires details of individual resource allocations to be maintained so that a decision can be made as to which resource should be used when the next allocation decision is made.

Solutions Where a workflow engine does not directly support round robin allocation, it is left to the auspices of the workflow developer to implement a strategy for this form of allocation. For the systems described above, COSA relies on the use of an external program to manage the allocation decision and keep track of previous allocations. iPlanet utilises Evaluate methods based on the TOOL language and access to an external SQL database for managing allocations.

17. Pattern R-SHQ (Shortest Queue)

Description The ability to allocate a work item to the resource that has the least number of work items allocated to it.

Example

- The *Heart Bypass Procedure* is allocated to the *Surgeon* who has the least number of operations allocated to them.

Related Patterns None

Motivation This allocation mechanism seeks to expedite the throughput of a workflow process by ensuring that work items are allocated to the resource that is able to undertake them in the shortest possible timeframe. Typically the shortest timeframe

means the resource with the shortest work list queue although other interpretations are possible.

Implementation In order to implement this allocation method, workflow engines need to maintain information on the work items currently allocated to resources and make this information available to the work item distribution algorithm. Of the workflow engines examined, COSA provides the `fewwork()` function which allows this Pattern to be directly realised. iPlanet provide facilities for programmatically extending the work item distribution algorithm and enabling this to be achieved indirectly.

Issues None identified.

Solutions N/A.

18. Pattern R-ED (Early Distribution)

Description The ability to advertise and potentially allocate work items to resources ahead of the moment at which the work item is actually enabled for execution.

Example

- The *Captain BA12 London – Bangkok flight* work item is offered to potential *Chief Pilots* at least two weeks ahead of the time that it will commence.

Related Patterns R-CA (Commencement on Allocation)

Motivation Early distribution provides a means of notifying resources of upcoming work items ahead of the time at which they need to be executed. This is useful where resources are able to provide some form of forward commitment (or booking) indicating they they will execute and complete a work item at some future time. It also provides a means of optimising the throughput of a workflow case by ensuring that minimal time is spent waiting for resource allocation during case execution.

Implementation None of the workflow systems examined directly support this Pattern, suggesting that the focus of production workflow systems tends to be on the management and completion of current work rather than on planning the optimal execution strategy for future work items. FLOWer (a case handling system) provides the ability for a resource to view future work items and potentially commence work on them even though they are not the next items in the process sequence. The case handling paradigm offers a different approach to work allocation. It is not discussed in detail here and interested readers are referred to [AWG05] for further information.

Issues None observed.

Solutions N/A.

19. Pattern R-DE (Distribution on Enablement)

Description The ability to advertise and allocate work items to resources at the moment they are enabled for execution.

Example

- The *Delivery Round* work item is allocated to a *Paper boy* at the time it is required to commence.

Related Patterns R-CC (Commencement on Creation), R-CE (Chained Execution)

Motivation Distribution of work items at the time that they are enabled for execution is effectively the standard mechanism for work distribution in a workflow system. The enablement of a work item serves as the trigger for the workflow engine to make it available to resources for execution. This may occur indirectly by placing it on the worklists for individual resources or on the global work list or directly by allocating it to a specific resource for immediate execution.

Implementation All of the systems examined – Staffware, WebSphere MQ, FLOWer, COSA and iPlanet – directly support this approach to work distribution in some form.

Issues None observed.

Solutions N/A.

20. Pattern R-LD (Late Distribution)

Description The ability to advertise and allocate work items to resources after the work item has been enabled.

Example

- The *Service Car* work item is allocated to a *Mechanic* after the car has been delivered for repair.

Related Patterns None

Motivation Late distribution of work items effectively provides a means of demand driving a workflow process by only advertising or allocating work items to resources when the work item has already been enabled for execution, possibly at some previous time. By adopting this approach, it is possible to reduce the current volume of work in progress within a workflow. This may involve stopping work items (or cases) from executing once the workload exceeds a certain threshold or restricting the amount of work in specific segments of the workflow. Often this strategy is undertaken with the aim of preventing resources from becoming overwhelmed by the apparent workload even though they may not be required to undertake all of it themselves.

Implementation None of the workflow engines examined support the notion of late distribution for newly created work items. However, a similar notion is used by some workflow engines for redeploying work items that have been allocated to resources or possibly have even commenced execution. COSA supports manual rerouting of work items by workflow users. WebSphere MQ provides an API for rerouting of work items.

Issues None identified.

Solutions N/A.

3.3 Pull Patterns

Pull Patterns correspond to the situation where individual resources are made aware of specific work items, that require execution, either via a direct offer from the workflow system or indirectly through a shared work list. The commitment to undertake a specific task is initiated by the resource itself rather than the workflow system. Generally this results in the work item being placed on the specific work list for the individual resource for later execution although in some cases, the resource may elect

to commence execution on the work item immediately. The various state transitions associated with Pull Patterns are illustrated in Figure 9:

- `R:allocate_s` corresponds to a work item offered to a single resource that the resource has indicated it will commit to executing at some future time.
- `R:allocate_m` corresponds to a work item offered to multiple resources that one of the resources has indicated it will commit to executing at some future time. The work item is deemed to be allocated to that resource and is no longer available to the other resources to which it was offered.
- `R:start_s` corresponds to a work item which has been offered to a single resource being started by that resource.
- `R:start_m` corresponds to a work item which has been offered to multiple resources being started by one of those resources.
- `R:start` corresponds to a work item which has been allocated to a single resource being started by that resource.

Six Pull Patterns have been identified. These divide into two distinct groups. The first three Patterns identify the specifics of the actual "pull" action initiated by the resource, with a particular focus on the work item state before and after the interaction. These Patterns correspond directly to the bold arcs in Figure 9.

The second group of Patterns focus on the sequence in which the work items are presented to the resource and the ability of the workflow system and the individual resource to influence the sequence and manner in which they are displayed. The final Pattern in this group illustrates the degree of freedom that the resource has in selecting the next work item to execute. These Patterns do not have a direct analogue in Figure 9 but apply to all of the "pull" transitions illustrated as bold arcs.

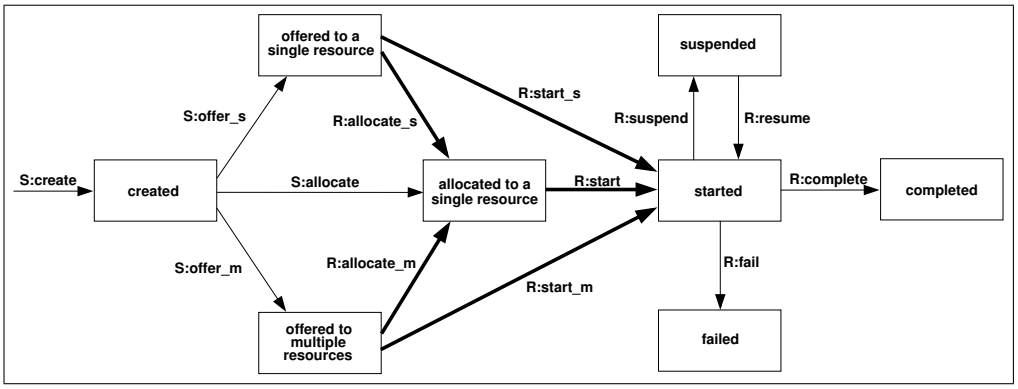

Figure 9: Pull Patterns

Note that the distinction between Push and Pull Patterns is identified by the initiator of the various transitions. For the Push Patterns in Figure 8, the state transitions for work items are all triggered by the workflow system, whereas in Figure 9 which denotes Pull Patterns, the transitions are initiated by individual resources.

Other characteristics of interest which ultimately lead to additional Pull Patterns, relate to whether the resource has the ability to reorder the work sequence or it is determined by the workflow system, and whether a resource can select which work item they wish to commence next from those on its work queue.

21. Pattern R-RIA (Resource-Initiated Allocation)

Description The ability for a resource to commit to undertake a work item without needing to commence working on it immediately.

Example

- The *Clerk* selects the Town Planning work items that she will undertake today although she only commence working on one of these at this point.

Related Patterns None

Motivation This Pattern provides a means for a resource to signal its intention to execute a given work item at some point although it may not commence working on it immediately. As a consequence of this action, the work item is considered to be allocated to the resource and it cannot be allocated to or executed by another resource. There are two variants of this Pattern as illustrated by the bold arcs in Figure 9, depending on whether the work item has been offered to a single resource (`R:allocate_s`) or to multiple resources (`R:allocate_m`).

Implementation The implementation of this Pattern generally involves the removal of the work item from a globally accessible or shared work list and its placement on a work queue specific to the resource to which it is allocated. Surprisingly only two of the workflow engines examined supports this function.

COSA allows a resource to reserve a work item that is displayed on a shared or global worklist for later execution by a user, however in doing so, the entire process instance is locked by the resource until the work item is completed or the reserve timeout is reached.

In FLOWer, cases are retrieved for a given resource via a case query which specifies the distribution criteria for cases that can be allocated to the resource. Where a resource executes a case query and a matching case is identified, all of the work items in the case are effectively allocated to the resource. Each of these work items is listed in the resource's work tray but is not commenced until specifically requested by the resource.

Issues None identified.

Solutions N/A.

22. Pattern R-RIEA (Resource-Initiated Execution – Allocated Work Item)

Description The ability for a resource to commence work on a work item that is allocated to it.

Example

- The *Courier Driver* selects the next *Delivery* work item which is allocated to it and commences work on it.

Related Patterns R-CC (Commencement on Allocation)

Motivation Where a resource has work items that it has committed to execute, but has not yet commenced, a means of signalling their commencement is required. This Pattern fulfils that requirement. It corresponds to the `R:start` transition illustrated in Figure 9.

Implementation The general means of handling that a work item has been allocated to a resource is to place it on a resource-specific work queue. This ensures that the work item is not undertaken by another resource and that the commitment made by the resource to which it is allocated is maintained. Staffware, WebSphere MQ, FLOWer and COSA all support the concept of resource-specific work queues and provide mechanisms in the work list handlers for resources to indicate that an allocated work item has been commenced.

Issues None identified.

Solutions N/A.

23. Pattern R-RIEO (Resource-Initiated Execution – Offered Work Item)

Description The ability for a resource to select a work item offered to it and commence work on it immediately.

Example

- The *Courier Driver* selects the next *Delivery* work item from those offered and commences work on it.

Related Patterns None

Motivation In some cases it is preferable to view a resource as being committed to undertaking a work item only when the resource has actually indicated that it is working on it. This approach to work distribution effectively speeds throughput by eliminating the notion of work item allocation. Work items remain on offer to the widest range of appropriate resources until one of them actually indicates they can commence work on it. Only at this time is the work item removed from being on offer and allocated to a specific resource.

This Pattern corresponds to the `R:start_s` and `R:start_m` transitions shown in Figure 9.

Implementation This approach to work distribution is adopted by Staffware, WebSphere MQ and COSA for shared work queues (e.g. group queues). For these systems, a work item remains on the queue until a resource indicates that it has commenced it. At this point, its status changes and no other resource can execute it although it remains on the shared queue until it is completed. iPlanet adopts this approach for all work items and effectively presents each resource with a single amalgamated queue of work items allocated directly to it and also those offered to a range of resources. The resource must indicate when it wishes to commence a work item. This results in the status of the work item changing and it being removed from any other work queues on which it might have existed.

Issues None identified.

Solutions N/A.

24. Pattern R-OBS (System-Determined Work Queue Content)

Description The ability of the workflow engine to order the content and sequence in which work items are presented to a resource for execution.

Example

- The *Staffware* workflow engine presents work items to resources in order of work item priority.

Related Patterns None

Motivation Where a workflow engine provide facilities for specifying the default ordering in which work items are presented to resources, the opportunity exists to enforce a work ordering policy for all workflow resources or on a group-by-group or individual resource basis. Such ordering may be time-based (e.g. FIFO, LIFO, EDD) or relate to data values associated with individual work items (e.g. cost, required effort, completion time).

Implementation Where this concept is supported by individual workflow engines, it is generally done so in terms of a single ordering sequence for all resources. Both Staffware and iPlanet support the ordering of work items on a priority basis for each resource's worklist. In both cases they also support the dynamic reordering of worklists as the priorities of individual work items change.

Issues None identified.

Solutions N/A.

25. Pattern R-OBR (Resource-Determined Work Queue Content)

Description The ability for resources to specify the format and content of work items listed in the work queue for execution.

Example

- The *Coordinator* resource has a work list ordered by time of receival.

Motivation Enabling resources to specify the format, content and ordering of their work queue provides them with a greater degree of flexibility in how they go about tackling the work items to which they have committed.

Implementation For those workflow engines which provide a client application for resources to interact with the workflow engine, the ability to be able to sort and filter work items is relatively commonplace. Staffware and WebSphere MQ allow any work item attribute to be used as the basis of the sort criterion or for filtering the work items that are displayed. FLOWer goes a step further and allows the user to specify "case queries" which define the type of cases that are retrieved into their work tray. COSA allows multiple views of available work to be defined and used at the resource level and includes support for the filtering of work items and specification of worklist queries.

Issues None identified.

Solutions N/A.

26. Pattern R-SA (Selection Autonomy)

Description The ability for resources to select a work item for execution based on its characteristics and their own preferences.

Example

- Of the outstanding Pruning work items, the *Head Gardener* chooses the one for execution they feel they are best suited to.

Related Patterns None

Motivation The ability for a resource to select the work item that they will commence next is a key aspect of the "heads up" approach to workflow execution. It aims to empower resources and let them have the flexibility to prioritise and organise their own individual work sequence.

Implementation All of the workflow engines examined provide support for this Pattern.

Issues One consideration with this Pattern is whether resources are still offered complete flexibility to choose which work item they will undertake next when there are urgent work items allocated to them or whether the workflow engine can guide their choice or dictate that a specific work item will be undertaken next.

Solutions Where autonomy is offered to resources in terms of the work items that they choose to execute, it is typically not revoked even in the face of pressing work items. Staffware and WebSphere MQ provide a means of highlighting urgent work items but do not mandate that these should be executed. Other workflow engines examined do not provide any facilities in this regard.

3.4 Detour Patterns

Detour Patterns refer to situations where work allocations that have been made for resources are interrupted either by the workflow system or at the instigation of the resource. As a consequence of this event, the normal sequence of state transitions for a workflow item is varied. The range of possible scenarios for Detour Patterns are illustrated in Figure 10.

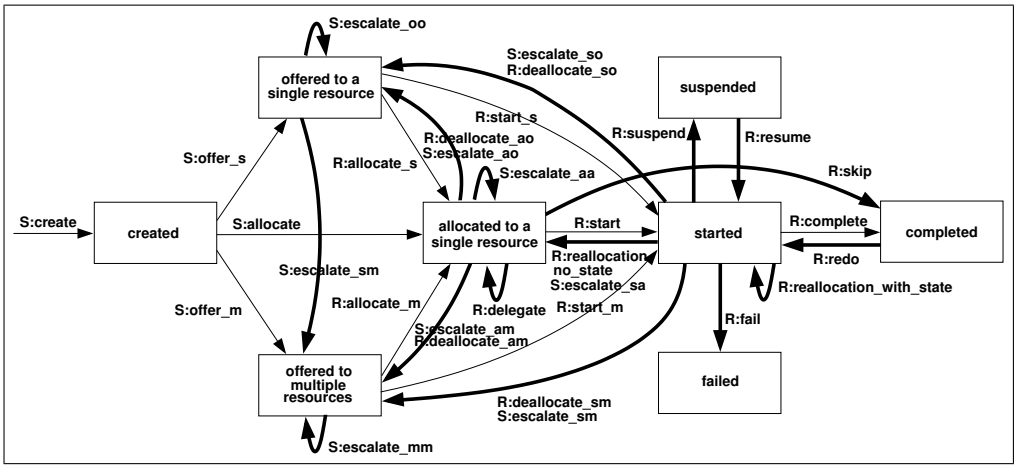

Figure 10: Detour Patterns

There are a number of possible impacts on a work item, depending on its current state of progression and whether the detour was initiated by the resource with which the work item was associated or by the workflow system. These include:

- *delegation* – where a resource allocates a work item previous allocated to it to another resource.
- *escalation* – where the workflow system attempts to progress a work item that has stalled by offering or allocating it to another resource.
- *de-allocation* – where the system makes a previously allocated or started work item available for offer and subsequent allocation.
- *reallocation* – where a resource allocates a work item that it has started to another resource.
- *suspension/resumption* – where a resource temporarily suspends execution of a work item or recommences execution of a previously suspended work item.
- *skipping* – where a resource elects to skip the execution of a work item allocated to it.
- *redo* – where a resource repeats execution of a work item completed earlier.
- *pre-do* – where a resource executes a work item that is ahead of the current execution point of a workflow case.

Each of these actions relate to one or more transitions in Figure 10 and corresponds to specific Patterns as described below.

27. Pattern R-D (Delegation)

Description The ability for a resource to allocate a work item previously allocated to it to another resource.

Example

- Before going on leave, the *Chief Accountant* passed all of their outstanding work items onto the *Assistant Accountant*.

Related Patterns None

Motivation Delegation provides a resource with a means of re-routing work items that it is unable to execute. This may be because the resource is unavailable (e.g. on vacation) or because they do not wish to take on any more work. It is illustrated by the `R:delegate` transition in Figure 10.

Implementation Generally the ability to delegate work items is included in the client work list handler for a workflow engine. Staffware, WebSphere MQ and COSA all provide the ability to manually redirect queued work items to a nominated resource. COSA supports an enhanced notion of delegation in that it redirects all work items corresponding to a specific task definition to a specified resource.

Issues One consideration associated with delegation is what happens where a work item is delegated to a user who is not authorised to execute it.

Solutions This scenario is only a problem for workflow engines that support distinct task routing and authorisation mechanisms. Both Staffware and WebSphere MQ allow a resource to execute any work item that is routed to them. However COSA provides an authorisation framework for work items that operates alongside

the distribution mechanism. In COSA, a work item could be distributed to a resource that does not have authorisation rights for it. Where this occurs, the resource can view the work item in their work list but cannot execute it. The only resolution is for them to delegate the work item to another resource that does have the required authorisation rights, or else acquire those rights themselves.

28. Pattern R-E (Escalation)

Description The ability of the workflow system to offer or allocate a work item to a resource or group of resources other than those it has previously been offered or allocated to in an attempt to expedite the completion of the work item.

Example

- The *review earnings* work item was reallocated to the *CFO*. It had previously been allocated to the *Financial Accountant* but the deadline for completion had been exceeded.

Related Patterns None

Motivation Escalation provides the ability for the workflow system to intervene in the conduct of a work item and assign it to alternative resources. Generally this occurs as a result of a specified deadline being exceeded, but it may also be a consequence of pre-emptive load balancing of work allocations undertaken by the workflow system or administrator in an attempt to optimise workflow throughput.

There are various ways in which a work item may be escalated depending on its current state of progression and the approach that is taken to identifying a suitable party to which it should be reassigned. The possible range of alternatives are illustrated by the `R:escalate_sm`, `R:escalate_am`, `R:escalate_mm`, `R:escalate_ss`, `R:escalate_as` and `R:escalate_aa` transitions in Figure 10.

Implementation Staffware, COSA and iPlanet provide direct support for deadlines on work items and allow alternate work items to be triggered (with distinct routing options) in the event that a work item fails to be completed in the required timeframe. WebSphere MQ provides reminders that notify a nominated resource that a given work item has exceeded a specified deadline.

Issues None identified.

Solutions N/A.

29. Pattern R-SD (Deallocation)

Description The ability of a resource (or group of resources) to relinquish a work item which is allocated to it and make it available for allocation to another resource or group of resources.

Example

- As progress on the *Conduct initial investigation* work item is not sufficient, the *Level 1 support officer* resource has made it available for reallocation to another *Support Consultant*.

Related Patterns R-E (Escalation)

Motivation Deallocation provides resources with a means of relinquishing work items allocated to them and making them available for re-allocation to other resources.

This may occur for a variety of reasons including insufficient progress, availability of a better resource or a general need to unload work from a resource.

There are two possible variations to deallocation – either the work item can be offered to a single resource or to multiple resources. These transitions are illustrated by the `R:deallocate_s` and `R:deallocate_m` arcs in Figure 10.

Implementation Despite the potential that this Pattern offers for actively managing the workload across a process, it is not widely implemented. COSA supports this Pattern through the redistribution function. iPlanet provides the ability for the workflow engine to reset the status of an active work item to ready. This has the effect of causing the work item to be reallocated using the same set of distribution criteria as were previously utilised for the work item.

Issues One problem that can arise when deallocating a work item is that it could ultimately be re-allocated to the same resource that it was previously retrieved from.

Solutions As the act of deallocating a work item is generally disjoint from that of reallocating it, the potential always exists for reallocation to the same resource unless active measures are taken to ensure that this does not occur. Generally there are three approaches for doing this:

- Make the resource unavailable for the period in which the reallocation will occur so that it is not considered in the work item redistribution.
- Stop the resource accepting new allocations or offers.
- Ensure that the distribution algorithm does not attempt to allocate a work item to a resource to which it has previously been allocated.

For iPlanet, the second and third options are both possible solutions where the workflow is running in "heads up" mode and resources have work items offered to them. Where it is running "heads down" and resources are directly allocated the next work item without an offer occurring, only the third option is feasible. In COSA, there is no direct solution to this problem.

30. Pattern R-PR (Stateful Reallocation)

Description The ability of a resource to allocate a work item to another resource without loss of state data.

Example

- The *Senior Partner* has suspended work on the *Building Society Audit Plan* work item and passed it to the *Junior Project Manager* for further work.

Related Patterns R-UR (Stateless Reallocation), R-D (Delegation)

Motivation Planned reallocation provides a resource with the ability to offload both pending and currently executing work items to other resources whilst maintaining the current state of the work item and the results of work undertaken on it to date. In the main, this centres on the ability to retain the current values of all data elements associated with the work item.

This Pattern corresponds to the `R:reallocation_with_state` arc in Figure 10. It is interesting to note the similarities between this Pattern and the Delegation

Pattern. Both Patterns result in a work item being reassigned to another resource. The main difference is that Delegation can only occur for a work item that has not yet commenced execution where as this Pattern applies to work items that are currently being executed.

Implementation Staffware, WebSphere MQ and COSA all support the notion of reallocating a work item to another resource with preservation of state albeit in slightly differing ways. Staffware only allows pending work items to be reallocated. WebSphere MQ requires that the work item is either pending or suspended in order for it to be reallocated. COSA provides support for reallocation through the reroute function.

Issues There are two potential issues associated with the reallocation of a work item to another resource whilst still preserving state information:

- Managing the transfer of state data.
- Ensuring the resource to which the work item is reallocated is entitled to execute it and access the associated state information.

Solutions There are a number of potential solutions to the first of these issues. One of these is to limit access to relevant state data elements to the resource executing the work item. This is the approach adopted by WebSphere MQ and COSA which use data containers to manage the data elements being passed between work items and work item specific data elements to manage state respectively. Staffware neatly avoids this issue by only allowing work items that have not been started to be reallocated.

The second of these issues is potentially more problematic. Staffware and WebSphere MQ do not impose any restrictions on the resources to which work items can be reallocated and any reassignments that a resource makes may be potentially inconsistent with the work distribution strategy implied by the process model. COSA provides an authorisation framework over work items in addition to the work distribution mechanism. Where a work item is reallocated to another resource, that resource must have the required authorisation to execute the task otherwise they will not be able to undertake it and will be required to further reallocate it to a resource that can.

31. Pattern R-UR (Stateless Reallocation)

Description The ability for a resource to reallocate a work item currently being executed to another resource without retention of state.

Example

- As progress on the *Recondition Engine* work item is not sufficient, it has been reallocated to another *Mechanic* who will restart it.

Related Patterns R-PR (Stateful Reallocation), R-D (Delegation), R-E (Escalation)

Motivation Stateless reallocation provides a lightweight means of reallocating a work item to another resource without needing to consider the complexities of state preservation. In effect, when this type of reallocation occurs all state information

associated with the work item (and hence any record of effective progress) is lost and the work item is basically restarted by the resource to which it is reassigned.

This Pattern is illustrated by the `R:reallocation_no_state` arc in Figure 10. It has similarities in terms of outcome with Delegation and Escalation Patterns in that the work item is restarted except that in this scenario, the work item has already been partially executed prior to the restart. This Pattern can only be implemented for work items that are capable of being redone without any consequences relating to the previous execution instance(s).

Implementation None of the workflow engines directly implement this approach to reallocation.

Issues None identified.

Solutions N/A.

32. Pattern R-SR (Suspension/Resumption)

Description The ability for a resource to suspend and resume execution of a work item.

Example

- The *Secretary* has suspended all *Board Meeting* work items whilst the *Board* is being reconstituted.

Related Patterns None

Motivation In some situations, during the course of executing a work item, a resource reaches a point where it is not possible to progress it any further. Suspension provides the ability for the resource to signal a temporary halt to the workflow engine of any work on the particular work item and switch its attention to another. The work item remains in the resource's work list but is generally notated as suspended. It is able to be restarted at some future time.

This Pattern is illustrated by the `R:suspend` and `R:resume` arcs in Figure 10.

Implementation This Pattern is implemented in a variety of different ways. Staffware allows work items that utilise a form to be suspended at any stage via the Keep option. Kept work items stay on the resource's work list and can be re-started later. WebSphere MQ doesn't allow individual work items to be suspended but does support the suspension of an entire workflow case. COSA directly supports the notion of suspension and where a work item is suspended, it is removed from the resource's work list and placed in a resubmission queue. At the point of suspension, a timeframe is nominated and after this has expired, the work item is again placed on the resources work list.

Issues One issue that can arise for suspended items that remain in a shared queue is whether they can be executed by other resources that may have access to the same queue.

Solutions This situation arises in Staffware and is actually used as a means of sharing a work item to which several resources may wish to contribute. When an item is suspended, all data that is associated with the work item (e.g. form data elements) are saved and become available to any other resource that may wish to resume the task. Any resource that can access a work item can signal its completion via the Release function.

33. Pattern R-SK (Skip)

Description The ability for a resource to skip a work item allocated to it and mark the work item as complete.

Example

- The *Ground Curator* has elected to skip the *Roll Pitch* work item previously allocated to it.

Related Patterns None

Motivation The ability to skip a work item reflects the common approach to expediting work processes by simply ignoring non-critical activities and assuming them to be complete such that subsequent work items can be commenced.

This Pattern is illustrated by the `R:skip` arc in Figure 10.

Implementation WebSphere MQ, FLOWer and COSA directly support the ability for a resource to skip work items allocated to them with the process client application.

Issues The main consideration that arises where work items could potentially be skipped is how to deal with data gathering requirements (e.g. forms that need to be completed by the resource) that are embodied within the work item. In the situation where a work item is skipped, it is generally just marked as complete and no execution is attempted. Subsequent work items that may be expecting data elements or other side-effects resulting from the skipped work item could potentially be compromised.

Solutions Where a workflow system supports the ability for work items to be skipped, it is important that subsequent work items do not necessarily rely on the output of previous work items unless absolutely necessary. The use of static data elements such as default parameter values can avoid much of the consequences of data not being received. More generally however in order to avoid these problems, the ability is required within a workflow system to specify work items that must be completed in full.

34. Pattern R-REDO (Redo)

Description The ability for a resource to redo a work item that has previously been completed in a case.

Example

- The *Inspector* has decided to redo the *Interview Key Witness* work item.

Related Patterns None

Motivation The Redo Pattern allows a resource to repeat a work item that has previously been completed. This may be based on a decision that the work item was not undertaken properly or because more information has become available that alters the potential outcome of the work item.

This Pattern is illustrated by the `R:redo` arc in Figure 10.

Implementation Of the workflow systems examined, only FLOWer provides the ability to redo a previously completed work item.

Issues Redoing a previously completed work item can have significant consequences on the execution of a case. In particular, the validity of any subsequent work items

is questionable as redoing a preceding work item may impact data elements utilised by these work items during their execution.

Solutions FLOWer addresses this issue by requiring any work items that depend on a "redone" work item to also be repeated before the case can be marked as complete.

35. Pattern R-PRE (Pre-Do)

Description The ability for a resource to execute a work item ahead of the time that it has been offered or allocated to resources working on a given case.

Example

- The *Inspector* has completed the *Charge Suspect* work item even though the preceding *Interview Witness* work items have not yet been completed.

Related Patterns None

Motivation The Pre-Do Pattern provides resources with the ability to complete work items in a case ahead of the time that they are required to be executed i.e. prior to them being offered or allocated to resources working on the case.

This Pattern is not illustrated in Figure 10.

Implementation Of the workflow systems examined, only FLOWer provides the ability to pre-do a work item.

Issues One consideration associated with pre-doing work items is the fact that outcomes of preceding work items that are executed after the time at which the "pre-done" work item is completed may result in the "pre-done" work item being repeatedly re-executed.

Solutions There is no immediate solution to this problem other than careful selection of work items that are to be done in advance. As a general rule, work items that are to be "pre-done" should not be dependent on data elements that are shared with preceding work items or the outcome of these work items.

3.5 Auto-start Patterns

Auto-start Patterns relate to situations where execution of work items is triggered by specific events in the lifecycle of the work item or the related process definition. Such events may include the creation or allocation of the work item, completion of another instance of the same work item or a work item that immediately precedes the one in question. The state transitions associated with these Patterns are illustrated by the bold arcs in Figure 11.

36. Pattern R-CC (Commencement on Creation)

Description The ability for a resource to commence execution on a work item as soon as it is created.

Example

- The *End of Month* work item commences execution as soon as it is created.

Related Patterns R-CE (Chained Execution)

Motivation The ability to commence execution on a work item as soon as it is created offers a means of expediting the overall throughput of a workflow case as it

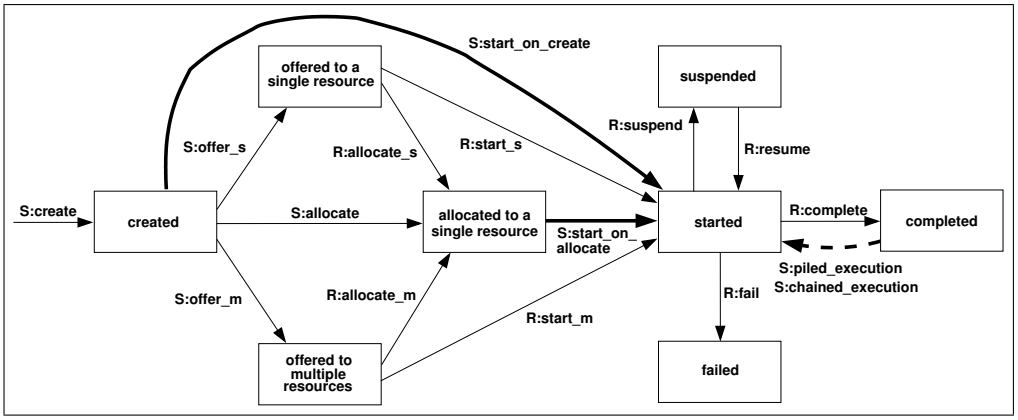

Figure 11: Auto-start Patterns

removes the delays associated with allocating the work item to a suitable resource and also the time that the work item remains in the resource's work queue prior to it being started. This Pattern is illustrated by the transition `S:start_on_create` in Figure 11.

Implementation All workflow engines which support automatic work items (i.e. work items that can execute without requiring allocation to a resource) provide limited support for the notion of commencement on creation. More complex however is the situation where a work item must be allocated to a resource as this implies that both creation and allocation must occur simultaneously.

COSA can support this method of operation where a work item is initiated via a trigger. It provides for a work item to be created and assigned to a specific resource in the same command.

Issues None identified.

Solutions N/A.

37. Pattern R-CA (Commencement on Allocation)

Description The ability to commence execution on a work item as soon as it is allocated to a resource.

Example

- Work on the *Fight Tower Block Fire* work item commences as soon as it is allocated to a *Fire Team* resource.

Related Patterns R-LD (Late Distribution)

Motivation Although combined creation, allocation and commencement of work items promotes more efficient workflow throughput, it effectively requires "hardcoding" of resource identities in order to manage work item allocation at creation time. This obviates much of the advantage of the flexible resource assignment strategies offered by workflow systems. Commencing work items at the point of allocation does not require resource identity to be predetermined and offers a means of expediting workflow throughput without necessitating changes to the underlying process model.

This Pattern is illustrated by the transition `S:start_on_allocate` in Figure 11.

Implementation The potential exists to implement this Pattern in one of two ways:

- Commencement on allocation can be specified within the workflow model.
- Individual resources can indicate that items in their work list are to be initiated as soon as they are received.

WebSphere MQ provides support for the second approach.

Issues None identified.

Solutions N/A.

38. Pattern R-PE (Piled Execution)

Description The ability of the workflow system to initiate the next instance of a workflow task (perhaps in a different case) once the previous one has completed.

Example

- The next *Clean Hotel Room* work item can commence immediately after the previous one has finished and it can be allocated to the same *Cleaner*.

Related Patterns R-CE (Chained Execution)

Motivation Piled execution provides a means of optimising task execution by pipelining instances of the same task and allocating them to the same resource. The resource undertakes work items sequentially and once a work item is completed, if another work item of the same type is present in the work queue, it immediately commences work on it – in effect it attempts to work on *piles* of the same types of work items. The aim with this approach to work distribution is to allocate similar work items to the same resource which aims to undertake them one after the other thus gaining from the benefit of exposure to the same type of task.

This Pattern is illustrated by the transition `R:piled_execution` in Figure 11. It is important to note that this transition is represented by a dashed line because it jumps from one work item to another, i.e., it links the life-cycles of two different work items.

Implementation To implement this Pattern requires like work items to be allocated to the same resource and the ability for the resource to undertake related work items on a sequential basis, immediately commencing the next one when the previous one is complete. This is a relatively sophisticated requirement and none of the workflow engines examined support it.

Issues None identified.

Solutions N/A.

39. Pattern R-CE (Chained Execution)

Description The ability of the workflow engine to automatically start the next work item in a case once the previous one has completed.

Example

- Immediately commence the next work item in the *Emergency Rescue Coordination* process when the preceding one has completed.

Related Patterns R-PE (Piled Execution)

Motivation The rationale for this Pattern is that case throughput is expedited when a resource is allocated sequential work items within a case and when a work item is completed, its successor is immediately initiated. This has the effect of keeping the resource constantly progressing a given case. This Pattern is illustrated by the transition `R:chained_execution` in Figure 11. Note that, similar to Pattern R-PE (Piled Execution), the transition is dashed because it connects the life cycles of different work items.

Implementation In order to implement this Pattern effectively, the majority (if not all) of the work items for a given case need to be allocated to the same resource and it must execute them in a strict sequential order. This approach to work distribution is best addressed by a case handling system and not surprisingly FLOWer offers direct support for it.

Issues Chained execution offers a means of achieving rapid throughput for a given workflow case however in order to ensure that this does not result in an arbitrary delay of other cases, it is important that cases are distributed across the widest possible range of resources and that the distribution only occurs when a resource is ready to undertake a new case.

Solutions This issue is managed in FLOWer by defining Work Profiles that distribute cases appropriately and ensuring that resources only request new case allocation when they are ready to commence the associated work items.

3.6 Visibility Patterns

Visibility Patterns classify the various scopes in which work item availability and commitment are able to be viewed by workflow resources. They give a indication of how open to scrutiny the operation of a workflow system is.

40. Pattern R-CUWV (Configurable Unallocated Work Item Visibility)

Description The ability to configure the visibility of unallocated work items by workflow participants.

Example

- The *Process Worker* can only see the unallocated work items that may be subsequently allocated to them or they can volunteer to undertake.

Related Patterns None

Motivation The Pattern denotes the ability of a workflow engine to limit the visibility of unallocated work items – either to potential resources to which they may subsequently be offered or allocated, or to completely shield knowledge of created but not yet allocated work items from all workflow resources.

Implementation None of the workflow engines examined support this Pattern.

Issues None identified.

Solutions N/A.

41. Pattern R-CAWIV (Configurable Allocated Work Item Visibility)

Description The ability to configure the visibility of allocated work items by workflow participants.

Example

- All *site workers* can view the allocated work items list for the day.

Related Patterns None

Motivation The Pattern indicates the ability of a workflow engine to limit the visibility of allocated work items.

Implementation Of the workflow engines examined, only FLOWer provides support for this Pattern. It does this by limiting the visibility of allocated work items to those resources that have the same role as the resource to which a work item is allocated.

Issues None identified.

Solutions N/A.

3.7 Multiple Resource Patterns

As indicated in the introduction and Section 3.2, the focus of this paper is on situations where there is a one-to-one correspondence between the resources and work items in a given allocation or execution. In other words, resources cannot work on different work items simultaneously and it is not possible that multiple resources work on the same work item. In situations where people are not restricted by information technology, there is often a many-to-many correspondence between the resources and work items in a given allocation or execution. Therefore, it may be desirable to support this using workflow technology. In this section, we discuss Patterns relaxing the one-to-one correspondence between resources and work items that we have assumed previously.

Let us first consider the one-to-many situation, i.e., resources *can* work on different work items simultaneously. This is a fairly simple requirement, supported by most systems.

42. Pattern R-SE (Simultaneous Execution)

Description The ability for a resource to execute more than one work item simultaneously.

Example

- The *Bank Teller* can conduct multiple *foreign exchange* work items at the same time.

Related Patterns None

Motivation In many situations, a resource does not undertake work items allocated to it on a sequential basis, but rather it commences work on a series of work items and multi-tasks between them.

Implementation All of the workflow engines examined support the ability for a resource to execute multiple work items simultaneously. In most tools, the resource can undertake any combination of work items although FLOWer (being a case handling tool) limits the group of simultaneous work items to those which comprise the activities in a dynamic plan.

Issues None identified.

Solutions N/A.

Pattern R-SE (Simultaneous Execution) is easy to support and contemporary systems support this one-to-many correspondence between the resources and work items in a given allocation or execution. Unfortunately, it is more difficult to support a many-to-one correspondence, i.e., multiple resources working on the same work item. This is a pity since for more complicated activities people tend to work in teams and collaborate to jointly execute work items. Given the limited support of today's workflow systems, we provide only one pattern implying a many-to-one correspondence.

43. Pattern R-AR (Additional Resources)

Description The ability for a given resource to request additional resources to assist in the execution of a work item that they are currently undertaking.

Example

- The *Blast Furnace Operator* has requested additional *Propane Gas Supplies* before continuing with the *Alloy Preparation* work item.

Related Patterns None

Motivation In more complex scenarios, a given work item may require the services of multiple resources in order for it to be completed (e.g. a machine operator, machine and fuel). These resources may be durable in nature and capable of continual reuse or they may be consumable. By providing the ability to model scenarios such as these, workflow engines provide a more accurate depiction of the way in which work is actually undertaken in a production environment.

Implementation None of the tools examined provide direct support for this requirement in a production context, however COSA offers limited simulation capabilities which allow the operation of a workflow to be evaluated. Included with the simulation environment is the ability to model the various operational resources required by a task – both durable and consumable – together with the associated rate of use on a task-by-task basis.

Issues None identified.

Solutions N/A.

To the best of our knowledge, all commercial workflow products assume a functional relation (in the mathematical sense) between (executed) work items and workers, i.e., from the viewpoint of the workflow management system each work item is executed by a single worker (this also holds for COSA). A worker selects a work item, executes the corresponding actions, and reports the result. It is not possible to model or to support the fact that a group of people, i.e., a *team*, executes a work item. Note that current workflow technology does not prevent the use of teams: Each step in the process can be executed by a team. However, only one team member can interact with the workflow management system with respect to the selection and completion of the work item. Thus, current workflow technology is not cognisant of teams. This is a major problem since teams are very relevant when executing workflow processes.

Consider for example the selection committee of a contest, the management team of a subdivision, the steering committee of an IT project, and the board of directors of a car manufacturer. In addition to providing explicit support for modelling teams, it is also important to recognize that individuals typically perform different roles within different teams. For example, a full professor can be the secretary of the selection committee for a new dean, and the head of the selection committee for tenure track positions. These examples show that existing systems, as well as the concepts used to discuss them, are still in their infancy when it comes to teams [AK01].

Groupware technology ranging from message-based systems such as Lotus Notes to group decision support systems such as GroupSystems offer support for people working in teams. However, these systems are not equipped to design and enact workflow processes. Based on this observation a marriage between groupware technology and workflow technology seems to be an obvious choice for developing team-enabled workflow solutions. Systems such as Lotus Domino Workflow [NEG+00] provide such a marriage between groupware and workflow technologies. Unfortunately, these systems only partially support a team working on a work item. For example, in Lotus Domino Workflow, for each work item one needs to appoint a so-called activity owner who is the only person who can decide whether an activity is completed or not, i.e., a single person serves as the interface between the workflow engine and the team. Clearly such a solution is not satisfactory.

Supporting a many-to-one correspondence between the resources and work items in a given allocation or execution (e.g., through teams) is not a simple as it may seem. For example, the moment of completion of a work item may be ambiguous, e.g. the teams may have to vote on the outcome of a successfully completed work item. In fact, the completion of a work item executed by a team could be subject to discussion, e.g., there can be a conflict: Some team members may dispute the completion of work item reported to be finished by other team members. In the traditional setting, one worker indicates the completion of a work item. This is not necessarily the case for teams. Other issues related to the operation of a team are: working at same time/different time, same place/different places, scheduled/ad-hoc meetings, etc. In [AK01] we classify and structure these issues in more detail and discuss possible realizations of the team concept. Moreover, we are currently looking at realising a "team-enabled workflow system" based on a sociotechnical approach. This system will be driven by resource patterns. However, these more advanced patterns are out of the scope of this paper.

4 Survey of existing workflow systems and business processing languages

This section presents the results of a detailed evaluation of support for the 43 Resource Patterns described above for five workflow and case handling systems. A broad range of offerings were chosen for this review in order to validate the applicability of each of the Patterns to the various types of tools that fall under the "workflow umbrella" [GHS95].

In summary, the tools evaluated were:

- **Staffware Process Suite version 9**[4] [Sta02a, Sta02b] – a widely used fully-integrated commercial workflow system.
- **WebSphere MQ Workflow 3.3.4** [IBM03a, IBM03b] – another widely used commercial workflow system that coordinates tasks based on 3GL programs.
- **FLOWer 3** [WF04] – a case handling system that adopts a data-driven approach to workflow execution.
- **COSA 4.2** [TRA03, TRA03] – a workflow system based on the Petri-Net modelling formalism.
- **iPlanet Integration Server 3.1** [Sun03] – a workflow engine that is part of a broad EAI integration suite developed by Sun.

4.1 Evaluation Results

A three scale assessment scale is used with "+" indicating direct support for the Pattern, "+/–" indicating partial support and "–" indicating that the Pattern is not implemented. The specifics of the rating criteria used are contained in appendix F.

Table 1 lists the results for Creation Patterns. It is immediately obvious that both direct and role-based allocation are standard mechanisms for work distribution and these Patterns are supported by all of the systems examined. For the other Creation Patterns, the extent of support is more varied.

Staffware provides relatively minimal coverage for this group of Patterns. It supports deferred allocation and basic organisational notions such as users, groups and roles but does not have a fully fledged organisational model that can be employed in workflow operation. It also does not allow work distribution in a case to be influenced by earlier runtime allocation decisions.

In comparison, WebSphere MQ does have an integrated organisational model which can be used to influence work distribution at runtime and as a consequence of both this and the history it maintains of runtime work allocation, enables both the Separation of Duties and Retain Familiar Patterns to be supported. One notable omission is that it is the only workflow system not to support Automatic Execution.

The strengths of the case handling paradigm, particularly in terms of the flexibility it provides for specifying a variety of runtime work allocation requirements in the design time model, are illustrated by the results for FLOWer which was the only case handling system to be examined. Whilst it doesn't support deferred or history-based allocation and its organisational model is heavily role-based, FLOWer fully supports all of the other Creation Patterns identified.

COSA is the other workflow system to incorporate a relatively comprehensive organisational model which is able to be used as a basis for specifying runtime work distribution. It also provides broad support for capability-based work allocation and possesses an effective authorisation framework.

In contrast, iPlanet lacks an organisational model and as a consequence its design time model is only able to specify Separation of Duties and Retain Familiar constraints on runtime work allocation. However, it does provide a range of options for work distribution based on resource capabilities and preceding work history.

[4] Although not the latest version, the functionality from a resource perspective is representative of the latest offering.

Nr	Pattern	*Staffware*	*WebSphere MQ*	*FLOWer*	*COSA*	*iPlanet*
1	R-DA (Direct Allocation)	+	+	+	+	+
2	R-RBA (Role-based Allocation)	+	+	+	+	+
3	R-FBA (Deferred Allocation)	+	+	–	–	–
4	R-RA (Authorisation)	–	–	+	+	–
5	R-SOD (Separation of Duties)	–	+	+	+/–	+
6	R-CH (Case Handling)	–	–	+	–	–
7	R-RF (Retain Familiar)	–	+	+	+	+
8	R-CBA (Capability-based Allocation)	–	–	+	+	+
9	R-HBA (History-based Allocation)	–	–	–	+/–	+
10	R-OA (Organisational Allocation)	+/–	+	+/–	+	–
11	R-AE (Automatic Execution)	+	–	+	+	+

Table 1: Support for Creation Patterns in Workflow Systems

Table 2 presents the evaluation results for Push Patterns. It can be seen that the (non-binding) offering by the workflow system of work items to multiple resources is the most widely supported distribution strategy together with direct allocation to a single resource where the routing criteria are specific enough. All of the systems examined implement these approaches. iPlanet (and in a limited way COSA as well) also allows work items to be offered to a single resource.

An interesting observation in this group of Patterns is the limited support for prioritising the sequence of offers where multiple resources are identified. Only COSA provides the ability to select a target resource on a random basis or with reference to the size of their existing work queue. iPlanet indirectly allows for these selection mechanisms through programmatic extensions. Round robin allocation – a common work distribution mechanism in real life – is not directly supported by any of the systems examined.

The timing of work distribution with respect to the time a work item is enabled tends to be the same across all of the systems examined. In general, work items become available for routing to resources at the time they are enabled. Only FLOWer provides the ability to distribute and execute work items ahead of the time they are enabled. None of the systems examined allow work items to be distributed later than the time of enablement, thus limiting the potential for throttling the rate of work distribution to the work capabilities of the currently available resource base.

Support for Pull Patterns is listed in Table 3. Pull Patterns illustrate the degree of autonomy workflow resources have in committing to and undertaking work items. The type of support provided by each tool in this area differs markedly. FLOWer is the offering that provide resources with the greatest degree of autonomy in that the timing of both the allocation and commencement of work items are at the complete

Nr	Pattern	*Staffware*	*WebSphere MQ*	*FLOWer*	*COSA*	*iPlanet*
12	R-DBOS (Distribution by Offer – Single Resource)	–	–	–	+/–	+
13	R-DBOM (Distribution by Offer – Multiple Resources)	+	+	+	+	+
14	R-DBAS (Distribution by Allocation – Single Resource)	+	+	+	+	+
15	R-RMA (Random Allocation)	–	–	–	+	+/–
16	R-RRA (Round Robin Allocation)	–	–	–	+/–	+/–
17	R-SHQ (Shortest Queue)	–	–	–	+	+/–
18	R-ED (Early Distribution)	–	–	+	–	–
19	R-DE (Distribution on Enablement)	+	+	+	+	+
20	R-LD (Late Distribution)	–	–	–	–	–

Table 2: Support for Push Patterns in Workflow Systems

discretion of the resource. Staffware, WebSphere MQ and COSA are similar in that they allow resources to control the allocation and execution time of work items offered to them, but they are not able to influence the manner or timing at which work items are offered to them. iPlanet has the least flexibility in this area and only allows a resource to control the timing at which it commences work items that are offered to it. It cannot control the timing or manner of work item offering or allocation.

Staffware, FLOWer and iPlanet allow the workflow system to specify the default ordering of work items in a resource's work queue. All of the systems (other than iPlanet which does not provide a worklist handler) allow resources to vary the sequence and properties of work items displayed in their work queue.

All of the offerings examined provide the resource with the ability to choose the next work item that they wish to execute from those currently listed in their work queues.

Table 4 illustrates the support that individual offerings provide for resources to vary the work item distribution that is effected by the workflow system. iPlanet (which does not provide an integrated worklist handler for resources) provides the most limited capabilities in this area, only allowing for deallocation and limited work item escalation.

The results obtained for FLOWer reinforce the implicit basis of the case handling approach it employs, in that all work within a given case is intended to be handled by the same resource. Therefore it provides no facilities for resources to reassign work items allocated to them. However, there is provision for resources to skip, redo and pre-do work items.

Staffware, WebSphere MQ and COSA all provide a range of capabilities that allow resources to vary the default work distribution imposed by the workflow system.

Nr	Pattern	*Staffware*	*WebSphere MQ*	*FLOWer*	*COSA*	*iPlanet*
21	R-RIA (Resource-Initiated Allocation)	–	–	+	+/–	–
22	R-RIEA (Resource-Initiated Execution – Allocated Work Item)	+	+	+	+	–
23	R-RIEO (Resource-Initiated Execution – Offered Work Item)	+	+	–	+	+
24	R-OBS (System-Determined Work List Management)	+	–	+	–	+
25	R-OBR (Resource-Determined Work List Management)	+	+	+	+	–
26	R-SA (Selection Autonomy)	+	+	+	+	+

Table 3: Support for Pull Patterns in Workflow Systems

COSA provides the broadest range of facilities in this area.

As is illustrated by Table 5, there is minimal support for Auto-Start Patterns. In particular, the Piled Execution Pattern (which allows for work items relating to the same task to be pipelined for execution by the same resource) is not supported at all.

The results obtained in Table 6 for configurable work item visibility indicate that this feature is not widely available and there is limited scope for varying the default visibility of allocated and unallocated work items imposed by the system.

Table 7 illustrates the extent of support for Multiple Resource Patterns. All of the systems examined support simultaneous execution of multiple work items by a resource except for FLOWer which limits this capability to Dynamic Plans. None of the systems examined provide production support for the Additional Resource Pattern.

5 Discussion

The objective of this research was to provide a comprehensive investigation of the way in which work is distributed amongst and undertaken by resources in workflow systems and to catalogue the results in the form of Patterns. It extends the previous work on the control-flow [AHKB03] and data [RHEA04] perspectives that was undertaken as part of the *Workflow Patterns Initiative*[5] to the resource perspective.

[5]See `www.workflowpatterns.com` for more details.

Nr	Pattern	*Staffware*	*WebSphere MQ*	*FLOWer*	*COSA*	*iPlanet*
27	R-D (Delegation)	+	+	–	+	–
28	R-E (Escalation)	+	+	–	+	+/–
29	R-SD (Deallocation)	–	–	–	+	+
30	R-PR (Stateful Reallocation)	+/–	+	–	+	–
31	R-UR (Stateless Reallocation)	–	–	–	–	–
32	R-SR (Suspension/Resumption)	+/–	+/–	–	+	–
33	R-SK (Skip)	–	+	+	+	–
34	R-REDO (Redo)	–	–	+	–	–
35	R-PRE (Pre-Do)	–	–	+	–	–

Table 4: Support for Detour Patterns in Workflow Systems

Nr	Pattern	*Staffware*	*WebSphere MQ*	*FLOWer*	*COSA*	*iPlanet*
36	R-CC (Commencement on Creation)	–	–	–	+	–
37	R-CA (Commencement on Allocation)	–	+	–	–	–
38	R-PE (Piled Execution)	–	–	–	–	–
39	R-CE (Chained Execution)	–	–	+	–	–

Table 5: Support for Auto-Start Patterns in Workflow Systems

The results listed in Section 4 raise some interesting points that merit further discussion. The first observation is that each workflow system has a distinct set of evaluation results. Indeed there is no marked similarity between the results obtained for any two products. This serves as an effective illustration of the varied ways in which the resource perspective is implemented across the range of workflow systems examined and reinforces the need for a fundamental taxonomy of the various concepts relevant to the resource perspective. It is this need that the Patterns identified in this paper hope to address.

A consideration that stems from these variations is that although all of the offerings examined are classified as workflow systems, their individual capabilities differ significantly. This raises the issue of *suitability* and the need to determine which sets of Patterns a workflow system should support in order to fulfil a specific operational need. For example, which set of Patterns is required for scheduling and managing

Nr	Pattern	*Staffware*	*WebSphere MQ*	*FLOWer*	*COSA*	*iPlanet*
40	R-CUWV (Configurable Unallocated Work Item Visibility)	–	–	–	–	–
41	R-CAWV (Configurable Allocated Work Item Visibility)	–	–	+	–	–

Table 6: Support for Visibility Patterns in Workflow Systems

Nr	Pattern	*Staffware*	*WebSphere MQ*	*FLOWer*	*COSA*	*iPlanet*
42	R-SE (Simultaneous Execution)	+	+	+/–	+	+
43	R-AR (Additional Resources)	–	–	–	+/–	–

Table 7: Support for Multiple Resource Patterns in Workflow Systems

production in a factory as against tracking claims in an insurance company. Through better understanding of the conceptual requirements of a particular problem domain in terms of the Patterns identified above, it should be easier to select the required technology support.

Another interesting observation is that Staffware, the workflow system which supports least Patterns, is the one which has the greatest market share. This further reinforces the notion of suitability – assumedly Staffware is widely used because it has features which are directly relevant to a broad range of problem domains – but it also suggests that the BPM marketplace is relatively immature and that the conceptual requirements of specific problem domains and the technology support that might be appropriate to them need to be better understood.

The timing of work item enablement with respect to distribution is one area that merits further investigation. All of the systems examined support the notion of enablement on distribution but there was minimal support for early or late distribution. Both of these alternatives offer opportunities for improving overall workflow throughput. In the case of early distribution, the onus for scheduling the best use of available work time can be placed on individual resources who can be presented with a pipeline of upcoming work items that they need to plan to complete in the most effective way. Late distribution provides the workflow system with the ability to actively match the amount of work in the system with available resources. This ensures that resources are not overwhelmed with the relentless addition of new work items and removes the potential for "thrashing" where resources spend more time organising and switching between concurrent work items than actually working on them.

Another interesting shortcoming of several of the systems examined is their inability to prioritise the way in which work items are offered to multiple resources. In most cases, the work item is simply presented to all identified resources simultaneously. Other than for COSA, there is no integrated ability to select a preferred resource on a round robin, shortest queue or even random basis.

Another area for potential improvement is illustrated by the lack of support for Auto-Start Patterns. These Patterns aim to increase overall work throughput by automatically starting the next work item for a resource and by pipelining like work items thus minimising familiarisation and switching time for resources.

One final observation relates to the amount of autonomy granted to resources by the workflow system. Case handling systems such as FLOWer provide significant latitude to resources by allowing them to organise the sequence in which they will undertake the work items in a given case. One of the drawbacks of the case handling approach is that the act of allocating all of the work items in a case to a single resource potentially removes the opportunities that may exist for executing work items in parallel and allocating them to several distinct resources. In a workflow system context, work items in a given case can be allocated to multiple resources and there is the option for executing work items in parallel. One opportunity for improving workflow throughput is by providing resources with the ability to redistribute work items where the routing decisions made by the workflow are not aligned with current resource workloads. The results obtained for Detour Patterns indicate that whilst workflow systems already offer some support, there is further opportunity in this area.

6 Related work

Despite the central role that resources play in workflow systems, there is a surprisingly small body of research into resource and organisational modelling in a workflow context [Aal03, AKV03]. In early work, Bussler and Jablonski [BJ95] identified a number of shortcomings of workflow systems when modelling organisational and policy issues. In subsequent work [JB96], they presented one of the first broad attempts to model the various perspectives of workflow systems in an integrated manner including detailed consideration of the organisational view.

One line of research into resource modelling and enactment in a workflow context has focussed on the characterisation of resource managers which can manage organisational resources and enforce resource policies. In [DS99], the design of a resource manager is presented for a workflow system. It includes a high level resource model together with proposals for resource definition, query and policy languages. Similarly in [LNOP00], an abstract resource model is presented in the context of a workflow system although the focus is more on the efficient management of resources in a workflow context than the specific ways in which work is allocated to them. In [HS99], a proposal is presented for handling resource policies in a workflow context. Three types of policy – qualification, requirement and substitution – are described together with a means for efficiently implementing them when allocating resources to activities.

Another area of investigation has been into ensuring that only suitable and authorised users are selected to execute a given work item. The RBAC (Role-Based

Access Control) model [FSG+01] presents an approach for doing this. Whilst effective, RBAC models tend to focus on security considerations and neglect other organisational aspects such as resource availability.

Several researchers have developed meta-models, i.e., object models describing the relation between workflow concepts, which include work allocation aspects, cf. [AK01, Mue99, Mue04, RM98]. However, these meta-models tend to focus on the structural description of resource properties and typically do not describe the dynamics aspects of work distribution.

7 Conclusion

This paper has identified 43 *Workflow Resource Patterns* which describe the manner in which work items are distributed and executed by resources in workflow systems. These Patterns have been validated through a detailed review of five workflow and case handling systems.

These Patterns are the logical progression of previous work investigating Workflow Control-Flow and Data Patterns as part of the *Workflow Patterns Initiative*. The aim of this research project is to identify generic structures in the various perspectives that constitute workflow systems. In doing so, it is anticipated that the key elements of a workflow description language can be identified that are applicable across the entire workflow domain.

Validation of these constructs also gives a critical insight into the operation of workflow systems and provides the opportunity to identify possible enhancements to current offerings in both the commercial and research domains. It supports comparison of the feature sets supported by individual products in a manner that is independent of their conceptual underpinnings and gives an indicative measure of the capabilities of specific offerings.

Acknowledgments

This work was partially supported by the Dutch research school BETA as part of the PATINT program and Australian Research Council under the Discovery Grant "*Expressiveness Comparison and Interchange Facilitation between Business Process Execution Languages*".

Several people have contributed to this research during its genesis. In particular the authors would like to acknowledge the contribution of Michael zur Muehlen during its early stages of development. Also, the feedback received from members of the BPM and IS research groups at QUT and EUT respectively was invaluable.

Disclaimer

We, the authors and the associated institutions, assume no legal liability or responsibility for the accuracy and completeness of any product-specific information contained in this paper. All reasonable efforts have been make to ensure that the results presented are, to the best of our knowledge, up to date and correct.

References

[Aal03] W.M.P. van der Aalst. Dont Go With the Flow: Web Services Composition Standards Exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003.

[ACD+03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services Version 1.1. Technical report, 2003. Accessed at `http://xml.coverpages.org/BPELv11-May052003Final.pdf` on 27 November 2004.

[AHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(3):5–51, 2003.

[AK01] W.M.P. van der Aalst and A. Kumar. Team-Enabled Workflow Management Systems. *Data and Knowledge Engineering*, 38(3):335–363, 2001.

[AKV03] W.M.P. van der Aalst, A. Kumar, and H.M.W. Verbeek. Organizational Modeling in UML and XML in the Context of Workflow Systems. In H. Haddad and G. Papadopoulos, editors, *18th Annual ACM Symposium on Applied Computing (SAC 2003)*, pages 603–608, Melbourne, Florida, USA, 2003. ACM Press.

[AWG05] W.M.P. van der Aalst, M. Weske, and D. Grnbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.

[BJ95] C. Bussler and S. Jablonski. Policy Resolution for Workflow Management Systems. In *28th Hawaii International Conference on System Sciences*, Hawaii, USA, 1995. IEEE Computer Society.

[DS99] W. Du and M.C. Shan. Enterprise Workflow Resource Management. In *Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE'99)*, pages 108–115, Sydney, Australia, 1999. IEEE Computer Society Press.

[FSG+01] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

[GHS95] D. Georgakopoulos, M.F. Hornick, and A.P. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

[Hal01] T.A. Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann Publishers, San Francisco, 2001.

[HS99] Y.N. Huang and M.C. Shan. Policies in a Resource Manager of Workflow Systems: Modeling, Enforcement and Management. Technical Report HP Tech. Report, HPL-98-156, 1999. Accessed at `http://www.hpl.hp.com/techreports/98/HPL-98-156.pdf` on 20 March 2005.

[IBM03a] IBM. *IBM Websphere MQ Workflow – Getting Started with Buildtime – Version 3.4*. IBM Corp., 2003.

[IBM03b] IBM. *IBM Websphere MQ Workflow – Programming Guide – Version 3.4*. IBM Corp., 2003.

[JB96] S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture and Implementation*. Thomson Computer Press, 1996.

[LNOP00] B.S. Lerner, A.G. Ninan, L.J. Osterweil, and R.M. Podorozhny. Modeling and Managing Resource Utilization in Process, Workflow, and Activity Coordination. Technical Report UM-CS-2000-058, Department of Computer Science, University of Massachusetts, August 2000. Accessed at `http://laser.cs.umass.edu/publications/?category=PROC` on 20 March 2005.

[Mue99] M. zur Muehlen. Evaluation of Workflow Management Systems Using Meta Models. In R. Sprague Jr, editor, *32nd Annual Hawaii International Conference on Systems Sciences*, Wailea, Hawaii, USA, 1999.

[Mue04] M. zur Muehlen. *Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems*. Logos, Berlin, 2004.

[NEG+00] S.P. Nielsen, C. Easthope, P. Gosselink, K. Gutsz, and J. Roele. *Using Lotus Domino Workflow 2.0, Redbook SG24-5963-00*. IBM, Poughkeepsie, USA, 2000.

[RHEA04] N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns. Technical Report QUT Technical Report FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004. Accessed from `http://www.citi.qut.edu.au/about/research_pubs/technical.jsp` on 27 November 2004.

[RM98] M. Rosemann and M. zur Muehlen. Evaluation of Workflow Management Systems - a Meta Model Approach. *Australian Journal of Information Systems*, 6(1):103–116, 1998.

[Sta02a] Staffware. *Staffware Process Suite – Defining Staffware Procedures – Issue 2*. Staffware plc, Maidenhead, 2002.

[Sta02b] Staffware. *Staffware Process Suite – Integrating Staffware with Your Enterprise Applications – Issue 2*. Staffware plc, Maidenhead, 2002.

[Sun03] Sun Microsystems. *Sun ONE Integration Server EAI, Version 3.1, Documentation*. Sun Microsystems, Palo Alto, California, USA, 2003.

[TRA03] TRANSFLOW. *COSA 4 Business-Process Designer's Guide*. TRANSFLOW AG, Pullheim, 2003.

[WF04] Wave-Front. *FLOWer 3 Designers Guide*. Wave-Front BV, 2004.

[Wor02] Workflow Management Coalition. Workflow Process Definition Interface – XML Process Definition Language. Technical Report Document Number WFMC-TC-1025, Workflow Management Coalition, 2002. Accessed at `http://www.wfmc.org/standards/docs.htm` on 27 November 2004.

A Staffware

This evaluation is based on Staffware version 9.

Nr	Pattern	Score	Motivation
1	R-DA	+	Directly supported
2	R-RBA	+	Directly supported
3	R-FBA	+	Directly supported
4	R-RA	–	Not supported
5	R-SOD	–	Not supported
6	R-CH	–	Not supported
7	R-RF	–	Not supported
8	R-CBA	–	Not supported
9	R-HBA	–	Not supported
10	R-OA	+/–	Partial support for roles and groups
11	R-AE	+	Directly supported via automatic steps
12	R-DBOS	–	Not supported
13	R-DBOM	+	Directly supported for group queues
14	R-DBAS	+	Directly supported where a single resource is identified during resource selection for a work item
15	R-RMA	–	Not supported
16	R-RRA	–	Not supported
17	R-SHQ	–	Not supported
18	R-ED	–	Not supported
19	R-DE	+	Directly supported as standard means of work item distribution
20	R-LD	–	Not supported
21	R-RIA	–	Not supported
22	R-RIEA	+	Directly supported as a standard consequence of starting an item on a work queue
23	R-RIEO	+	Directly supported for group queues
24	R-OBS	+	Directly supported with work items ordered by priority
25	R-OBR	+	Directly supported through resource-specific work queue customisation and filtering
26	R-SA	+	Directly supported as the standard means for a resource to select the next work item
27	R-D	+	Directly supported through task forwarding
28	R-E	+	Directly supported through withdraw actions and reallocating another instance of the same task
29	R-SD	–	Not supported
30	R-PR	+/–	Only supported for pending (not started) activities
31	R-UR	–	Not supported
32	R-SR	+/–	Supported for activities that have forms associated with them

Nr	Pattern	Score	Motivation
33	R-SK	–	Not supported
34	R-REDO	–	Not supported
35	R-PRE	–	Not supported
36	R-CC	–	Not supported
37	R-CA	–	Not supported
38	R-PE	–	Not supported
39	R-CE	–	Not supported
40	R-CUWV	–	Not supported
41	R-CAWV	–	Not supported
42	R-SE	+	Directly supported. All resources can execute multiple activities simultaneously
43	R-AR	–	Not supported

B WebSphere MQ Workflow

This evaluation is based on WebSphere MQ Workflow 3.4.

Nr	Pattern	Score	Motivation
1	R-DA	+	Directly supported
2	R-RBA	+	Directly supported
3	R-FBA	+	Directly supported
4	R-RA	–	Not supported
5	R-SOD	+	Directly supported via task linking between activities in the process model that cannot have the same resource allocation at runtime within a case
6	R-CH	–	Not supported
7	R-RF	+	Common resource allocation can be specified for specific tasks in the process model requiring the same resource allocation at runtime within a case
8	R-CBA	–	Not supported
9	R-HBA	–	Not supported
10	R-OA	+	Directly supported
11	R-AE	–	Not supported
12	R-DBOS	–	Not supported
13	R-DBOM	+	Work queues combine work item items specifically offered to this resource and those offered to multiple resources
14	R-DBAS	+	Directly supported for work items allocated to a single resource
15	R-RMA	–	Not supported
16	R-RRA	–	Not supported
17	R-SHQ	–	Not supported
18	R-ED	–	Not supported
19	R-DE	+	Standard mechanism for work item distribution
20	R-LD	–	Not supported
21	R-RIA	–	Not supported
22	R-RIEA	+	Standard means for a resource to initiate a work item is to select one from those allocated to it
23	R-RIEO	+	Supported for work items distributed via shared work queues.
24	R-OBS	–	Not supported
25	R-OBR	+	Work queues can be sorted or filtered on any work item attribute at the discretion of individual resources
26	R-SA	+	Resources can select the next item for execution from those on their work queue
27	R-D	+	Work items can be manually redirected by resources
28	R-E	+	Directly supported via reminders
29	R-SD	–	Not supported

Nr	Pattern	Score	Motivation
30	R-PR	+	Supported for pending and suspended items
31	R-UR	–	Not supported
32	R-SR	+/–	Indirectly supported via case suspension
33	R-SK	+	Directly supported in the worklist handler
34	R-REDO	–	Not supported
35	R-PRE	–	Not supported
36	R-CC	–	Not supported
37	R-CA	+	Resources can configure work queues to initiate work items on arrival
38	R-PE	–	Not supported
39	R-CE	–	Not supported
40	R-CUWV	–	Not supported
41	R-CAWV	–	Not supported
42	R-SE	+	Resources can execute multiple work items simultaneously
43	R-AR	–	Not supported

C FLOWer

This evaluation is based on FLOWer 3.

Nr	Pattern	Score	Motivation
1	R-DA	+	Where there is a 1-1 correspondence between a specific role and an actual resource
2	R-RBA	+	Organisational hierarchy is modelled in terms of roles
3	R-FBA	–	Not supported
4	R-RA	+	Use of execute roles allows authentication to be enforced
5	R-SOD	+	Achieved by specifying additional execution constraints (not same user) on plan elements
6	R-CH	+	Achieved by specifying that all plan elements should have the same user
7	R-RF	+	Directly supported at plan element level
8	R-CBA	+	Cases can be assigned based on data or process elements
9	R-HBA	–	Not supported
10	R-OA	+/–	Role hierarchy provides limited support for specifying an organisational structure for use in work distribution
11	R-AE	+	Facilitated through the use of automatic actions
12	R-DBOS	–	Not supported
13	R-DBOM	+	Default work allocation mechanism is to offer a work item to all participants in a role
14	R-DBAS	+	Directly supported where a role corresponds to a specific resource
15	R-RMA	–	Not supported
16	R-RRA	–	Not supported
17	R-SHQ	–	Not supported
18	R-ED	+	Resources are able to view and execute tasks ahead of the wavefront
19	R-DE	+	In general, resources are allocated and execute work items on the wave front (i.e. one that have been enabled)
20	R-LD	–	Not supported
21	R-RIA	+	Directly supported
22	R-RIEA	+	Directly supported
23	R-RIEO	–	Not supported
24	R-OBS	+	The system presents work items in order of execution sequence
25	R-OBR	+	Distinct work trays can be configured for users
26	R-SA	+	Users can execute any work item at, ahead or behind the wavefront

Nr	Pattern	Score	Motivation
27	R-D	–	Not supported
28	R-E	–	Not supported
29	R-SD	–	Not supported
30	R-PR	–	Not supported
31	R-UR	–	Not supported
32	R-SR	–	Not supported
33	R-SK	+	Directly supported via skip action
34	R-REDO	+	Users can redo a work item that has already been completed
35	R-PRE	+	Users can undertaken a work item ahead of the wavefront
36	R-CC	–	Not supported
37	R-CA	–	Not supported
38	R-PE	–	Not supported
39	R-CE	+	Directly supported by Open Action Mode setting
40	R-CUWV	–	Not supported
41	R-CAWV	+	Supported through use of roles to limit visibility of case activities
42	R-SE	+/–	Only for elements of a dynamic plan
43	R-AR	–	Not supported

D COSA

This evaluation is based on COSA 4.

Nr	Pattern	Score	Motivation
1	R-DA	+	Directly supported
2	R-RBA	+	Directly supported
3	R-FBA	–	Not supported
4	R-RA	+	Distinct authorisation and distribution mechanisms are supported
5	R-SOD	+/–	Indirectly achievable through user access rights
6	R-CH	–	Not supported
7	R-RF	+	Supported via a customised distribution algorithm
8	R-CBA	+	Supported via competency definitions for users, groups and activity definitions
9	R-HBA	+/–	Indirectly achievable via a custom (external) distribution algorithm
10	R-OA	+	Directly supported through user/group and user/group script languages
11	R-AE	+	Achievable by assigning work items to internal system users
12	R-DBOS	+/–	Non-binding offers not supported but allocated work items can be rejected for reallocation
13	R-DBOM	+	Directly supported for multiple resources via group queues
14	R-DBAS	+	Directly supported
15	R-RMA	+	Directly supported via random function in user/group language
16	R-RRA	+/–	Indirectly achievable via a custom (external) distribution algorithm
17	R-SHQ	+	Supported via a customised distribution algorithm based on the fewwork function
18	R-ED	–	Not supported
19	R-DE	+	Standard means of work distribution
20	R-LD	–	Not supported
21	R-RIA	+/–	The act of a resource reserving a work item on a shared work list has the effect of locking the process instance
22	R-RIEA	+	Resources can initiate allocated work items from their work queues
23	R-RIEO	+	Supported for work items on shared work queues
24	R-OBS	–	Not supported
25	R-OBR	+	A series of options are provided for configuring views in the worklist handler
26	R-SA	+	Resources can select any of the work items on their queue to initiate next

Nr	Pattern	Score	Motivation
27	R-D	+	Resources can manually reroute work items from their work queues
28	R-E	+	Supported via the Target Dates option for processes together with a trigger to reroute the work item
29	R-SD	+	Directly supported through redistribution
30	R-PR	+	Supported through the reroute function
31	R-UR	–	Not supported
32	R-SR	+	Resources can suspend work items currently being executed
33	R-SK	+	Directly supported via skip option in worklist handler
34	R-REDO	–	Not supported
35	R-PRE	–	Not supported
36	R-CC	+	Supported for tasks initiated via a trigger
37	R-CA	–	Not supported
38	R-PE	–	Not supported
39	R-CE	–	Not supported
40	R-CUWV	–	Not supported
41	R-CAWV	–	Not supported
42	R-SE	+	Resources can execute multiple work items simultaneously
43	R-AR	+/–	Simulation environment provides multiple resource modelling capabilities for a single work item

E iPlanet

This evaluation is based on iPlanet.

Nr	Pattern	Score	Motivation
1	R-DA	+	Directly supported
2	R-RBA	+	Directly supported
3	R-FBA	–	Not supported
4	R-RA	–	Not supported
5	R-SOD	+	Directly supported through linked activities
6	R-CH	–	Not supported
7	R-RF	+	Directly supported through linked tasks with common resources
8	R-CBA	+	Directly supported through customised distribution algorithms
9	R-HBA	+	Achievable through extended user profiles and customised distribution algorithms
10	R-OA	–	Not supported
11	R-AE	+	Directly supported
12	R-DBOS	+	Directly supported for offered work items
13	R-DBOM	+	Directly supported for offered and queued work items
14	R-DBAS	+	Directly supported where a work item is allocated to a single resource in “heads down” mode
15	R-RMA	+/–	Indirectly achievable via customised (external) distribution algorithm
16	R-RRA	+/–	Indirectly achievable via customised (external) distribution algorithm
17	R-SHQ	+/–	Indirectly achievable via customised (external) distribution algorithm
18	R-ED	–	Not supported
19	R-DE	+	Standard means of work item distribution
20	R-LD	–	Not supported
21	R-RIA	–	Not supported
22	R-RIEA	–	Not supported
23	R-RIEO	+	Standard approach to initiating work items
24	R-OBS	+	Work items are ordered by priority by default
25	R-OBR	–	Not supported
26	R-SA	+	Directly supported for offered work items
27	R-D	–	Not supported
28	R-E	+/–	Indirectly supported via timers but the cleanup action for the escalated work item must be specified programmatically
29	R-SD	+	Achieved by changing the status of a work item to READY

Nr	Pattern	Score	Motivation
30	R-PR	–	Not supported
31	R-UR	–	Not supported
32	R-SR	–	Not supported
33	R-SK	–	Achieved by changing the status of a work item to COMPLETED where it is in the PENDING state
34	R-REDO	–	Not supported
35	R-PRE	–	Not supported
36	R-CC	–	Not supported
37	R-CA	–	Not supported
38	R-PE	–	Not supported
39	R-CE	–	Not supported
40	R-CUWV	–	Not supported
41	R-CAWV	–	Not supported
42	R-SE	+	Resources can execute multiple work items simultaneously
43	R-AR	–	Not supported

F Evaluation Criteria

The following tables summarise the evaluation criteria for each of the Workflow Resource Patterns.

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/– rating it must satisfy at least one of the criteria listed.
- Otherwise a – rating is recorded.

Nr	**Pattern**	**+ Rating**	**+/– Rating**
1	R-DA (Direct Allocation)	1. Workflow has an integrated organisational model. 2. Work items can be allocated to a specific resource identified in the organisational model.	1. Work items can be routed to a specific resource by some means (e.g. queue, work list etc.)
2	R-RBA (Role-based Allocation)	1. Workflow has an integrated organisational model that directly supports roles. 2. Routing a work item via role-based allocation involves directing the work item to one specific resource that participates in the nominated role.	1. Roles are not directly supported by the workflow engine but the same effect can be achieved via other mechanisms.
3	R-FBA (Deferred Allocation)	1. Workflow engine provides support for nominating the allocation strategy for a work item at run-time. 2. Identification for the resource allocation is derived from a data fields. 3. Direct, group or role-based allocation is supported.	1. Similar effect can be achieved through programmatic extensions.

Nr	Pattern	+ Rating	+/− Rating
4	R-RA (Authorisation)	1. Workflow engine provides a distinct security framework defining which resources can execute which work items. 2. Authorisation to execute a work item is independent of the routing mechanism.	1. Authorisation is incorporated in the work distribution mechanisms. 2. Achievable via precondition specification or programmatic extensions for each task.
5	R-SOD (Separation of Duties)	1. Workflow engine supports the ability to prevent nominated work items within a case being allocated to the same resource as other specified work items in the same case. 2. The required separation of duties can be specified in the process model in terms of relationships between tasks.	1. Separation of duties is based on security mechanisms rather than the work distribution framework. 2. Programmatic extensions required to the work distribution process or individual task definitions.
6	R-CH (Case Handling)	1. Workflow engine supports the ability to allocate all work items in a given case to the same resource. 2. Allocations are implied/specified at process level rather than for individual task definitions.	1. N/A.
7	R-RF (Retain Familiar)	1. The process model provides support for specifying that the resource to which a work item is allocated should be the same as that to which a preceding work item was allocated within a given case.	1. N/A.
8	R-CBA (Capability-based Allocation)	1. Workflow engine supports definition of resource capabilities in the process model. 2. Resource requirements can be specified (in terms of capabilities) for individual task in the process model. 3. Work distribution strategy takes capabilities into account when allocating work items.	1. A similar effect is achieved indirectly through a combination of rules, role or group-based allocation and programmatic extensions. 2. Allocation of a work item to an optimal resource (in terms of specified capabilities) cannot be guaranteed.

Nr	Pattern	+ Rating	+/− Rating
9	R-HBA (History-based Allocation)	1. The workflow engine provide facilities for utilising previous execution history when distributing work items. 2. The required allocation criteria can be specified in the process model.	1. The use of previous execution history requires (external) programmatic extensions to the work item allocation algorithm.
10	R-OA (Organisational Allocation)	1. The workflow engine provides support for specifying the relationships between workflow users (i.e. resources) in the form of an organisational model. 2. These relationships can be used as the basis of work item offering and allocation.	1. A similar effect can be achieved through programmatic extensions. 2. There is partial support for an organisational model.
11	R-AE (Automatic Execution)	1. Provision exists within the process model to nominate tasks that can execute without requiring their corresponding work items to be allocated to a resource.	1. Work arounds exist to execute a task automatically by allocating it to an internal resource or by executing a linked external program.
12	R-DBOS (Distribution by Offer – Single Resource)	1. The workflow engine provides a mechanism to advertise a work item to a specific resource. The resource is not obliged to commit to executing the work item.	1. The workflow engine indirectly supports the concept by allowing resources to reject work items assigned to them.
13	R-DBOM (Distribution by Offer – Multiple Resources)	1. The workflow engine provides a mechanism to advertise a work item to several resources simultaneously. 2. None of the resources are obliged to commit to executing the work item.	1. N/A.
14	R-DBAS (Distribution by Allocation – Single Resource)	1. The workflow engine provides a means of directly allocating a work item to a single resource.	1. N/A.
15	R-RMA (Random Allocation)	1. Facilities exist (either in the process model or at run-time) to nominate that the resource to which a work item is allocated is picked at random from those that satisfy the selection criteria for potential assignment.	1. N/A.

Nr	Pattern	+ Rating	+/− Rating
16	R-RRA (Round Robin Allocation)	1. Facilities exist (either in the process model or at run-time) to nominate that the resource to which a work item is allocated is chosen on a cyclic basis from the group of resources which satisfy the selection criteria for potential assignment.	1. N/A.
17	R-NAR (Shortest Queue)	1. Facilities exist (either in the process model or at run-time) to nominate that a work item should be allocated to the resource with the shortest work queue from the group of resources that meets the selection criteria for potential assignment.	1. N/A.
18	R-ED (Early Distribution)	1. Resources can commit to executing potential work items ahead of the time that the work items exist or become available for execution.	1. N/A.
19	R-DE (Distribution on Enablement)	1. Work items become available for advertising/allocation to resources at the time that they are enabled.	1. N/A.
20	R-LD (Late Distribution)	1. Work items can be advertised/allocated to resources after the time at which they are enabled for execution (i.e. general knowledge of their existence can be delayed).	1. N/A.
21	R-RIA (Resource-Initiated Allocation)	1. Resources are able to commit to executing an offered work item, resulting in it being allocated to their work queue. They are not obliged to commence working on it immediately.	1. Resources can reserve a work item for later execution by suspending execution of the workflow case.
22	R-RIEA (Resource-Initiated Execution – Allocated Work Item)	1. Resources are able to commence execution of a work item allocated to them at a time of their own choosing.	1. N/A.

Nr	Pattern	+ Rating	+/− Rating
23	R-RIEO (Resource-Initiated Execution – Offered Work Item)	1. Resources are able to simultaneously request allocation and commence execution on work item(s) offered to them.	1. N/A.
24	R-OBS (System-Determined Work List Management)	1. The workflow engine is able to impose a default ordering of the work items in a resource's work queue.	1. N/A.
25	R-OBR (Resource-Determined Work List Management)	1. Each resource is able to impose a filtering and/or an ordering sequence on the work items appearing in its work queue.	1. The extent of the ordering sequence is limited. 2. Filtering of work items is supported but not reordering.
26	R-SA (Selection Autonomy)	1. A resource is able to select the next work item that it will commence from any of those currently in its work queue.	1. N/A.
27	R-D (Delegation)	1. Resources can redirect a work item allocated to them (but not initiated) to another resource either manually or automatically.	1. N/A
28	R-E (Escalation)	1. The workflow system can redirect a work item to another resource (or group of resources) either manually or automatically. 2. Work items in any state (offered, allocated or commenced) can be redirected.	1. Only work items in certain states can be redirected.
29	R-SD (Deallocation)	1. A resource is able to remove (allocated or commenced) items from its work queue and make them available for reallocation to other resources.	1. Only allocated work items can be redirected.
30	R-PR (Stateful Reallocation)	1. A resource is able to reallocate a work item on which it commenced execution to another resource without loss of state (i.e. values of current state variables).	1. N/A.

Nr	Pattern	+ Rating	+/− Rating
31	R-UR (Stateless Reallocation)	1. A resource is able to reallocate a work item on which it commenced execution although state information is not preserved (and the work item must be recommenced).	1. N/A.
32	R-SR (Suspension/Resumption)	1. Resources can suspend the execution of a work item on which they are currently working. It remains allocated to them. They can resume it at a later time of their choosing.	1. Resources can suspend a work item by suspending execution of the entire case to which it relates. 2. A resource can suspend a work item but it may be picked up for execution by other resources.
33	R-SK (Skip)	1. A resource can skip the execution of a work item allocated to it by marking it as completed.	1. N/A.
34	R-REDO (Redo)	1. A resource can repeat the execution of a work item that has already been completed. 2. Any subsequent work items are also repeated.	1. Subsequent work items are not repeated.
35	R-PRE (Pre-Do)	1. A resource can execute a work item that is ahead of the current execution point of the case.	1. N/A.
36	R-CC (Commencement on Creation)	1. The workflow engine supports simultaneous creation, allocation and commencement of a work item.	1. N/A.
37	R-CA (Commencement on Allocation)	1. The workflow engine supports automatic commencement of a work item at the point of allocation.	1. N/A.
38	R-PE (Piled Execution)	1. The workflow engine is able to allocate and initiate work items associated with the same task to the same resource such that the completion of one work item results in automatic initiation of the next work item.	1. N/A.

Nr	Pattern	+ Rating	+/− Rating
39	R-CE (Chained Execution)	1. The ability of the workflow engine to automatically trigger the execution of subsequent work items in a case once the preceding work item is finished. Subsequent work items may not necessarily be routed to the same resource.	1. N/A.
40	R-CUWV (Configurable Unallocated Work Item Visibility)	1. The workflow engine provides the ability to configure the extent to which knowledge of unallocated work items is accessible to workflow users.	1. N/A
41	R-CAWV (Configurable Allocated Work Item Visibility)	1. The workflow engine provides the ability to configure the extent to which knowledge of allocated work items is accessible to workflow users.	1. N/A
42	R-SE (Simultaneous Execution)	1. Resources are able to execute more than one work item simultaneously.	1. N/A.
43	R-AR (Additional Resources)	1. A resource may request allocation of additional resources during execution of a work item.	1. Facilities exist to simulate complex resource requirements during workflow execution.

Table 8: Resource Pattern Evaluation Criteria for Workflow Systems