Workflow Resource Patterns: Identification, Representation and Tool Support*

Nick Russell[1], Wil M.P. van der Aalst[2,1], Arthur H.M. ter Hofstede[1], and David Edmond[1]

[1] Centre for IT Innovation, Queensland University of Technology
P.O. Box 2434, Brisbane QLD 4001, Australia.
{n.russell,a.terhofstede,d.edmond}@qut.edu.au
[2] Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NIL-5600 MB, Eindhoven, The Netherlands.
w.m.p.v.d.aalst@tm.tue.nl

Keywords: Workflow, Patterns, Resources, Business Process Modelling.

Abstract. In the main, the attention of workflow researchers and workflow developers has focussed on the process perspective, i.e., control-flow. As a result, issues associated with the resource perspective, i.e., the people and machines actually doing the work, have been largely neglected. Although the process perspective is of most significance, appropriate consideration of the resource perspective is essential for successful implementation of workflow technology. Previous work has identified recurring, generic constructs in the control-flow and data perspectives, and presented them in the form of control-flow and data patterns. The next logical step is to describe workflow resource patterns that capture the various ways in which resources are represented and utilised in workflows. These patterns include a number of distinct groupings such as push patterns ("the system pushes work to a worker") and pull patterns ("the worker pulls work from the system") to describe the many ways in which work can be distributed. 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.

1 Introduction

Over the last decade there has been increasing interest in *process-aware information systems* (PAIS), 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,

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

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, [6]) 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 the language BPEL4WS [6] which does not provide any degree of direct support for resources in business processes based on web-services. Similarly, languages like XPDL [13], 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, organisations, work distribution mechanisms, etc. A quote attributable to 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 PAIS. 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 PAIS 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 they can do is execute the work items in their work queues, i.e., people are treated as automatons and have little influence over the way in which 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*[3] to the resource perspective. This research project seeks to identify recurring generic workflow constructs through review of the conceptual foundations of workflow systems together with a comprehensive survey of commercial product offerings. These constructs are then described in the form of patterns. Initially, it examined control-flow dependencies in workflow languages [2]. Later it was extended to include web-services

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

composition languages [1] and the data perspective [11]. This work[3] has directly influenced tool selection processes, commercial and open-source workflow systems, and workflow standards. In this paper, we adopt an approach similar to that in [2] although in this case, the focus is on the resource perspective.

The remainder of the paper is organised as follows. We first introduce some basic concepts relating to workflow management in general and resource modelling in particular. Then we describe a series of selected resource patterns drawn from the various categories that have been identified. Note that of the more than 40 resource patterns identified, we only describe a few typical examples and refer the reader to [12] for a comprehensive discussion of the complete set of patterns. Section 4 evaluates five existing workflow products using these patterns. Section 5 discusses related work and Section 6 concludes the paper.

2 Workflow and Resource Concepts

Before we describe the resource patterns in detail, we present a standard set of definitions for the various workflow concepts that we will utilise throughout this paper. In doing so, we first introduce some general terminology for workflow models and then focus on the concepts relating to the resource perspective.

2.1 Workflow Model

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.

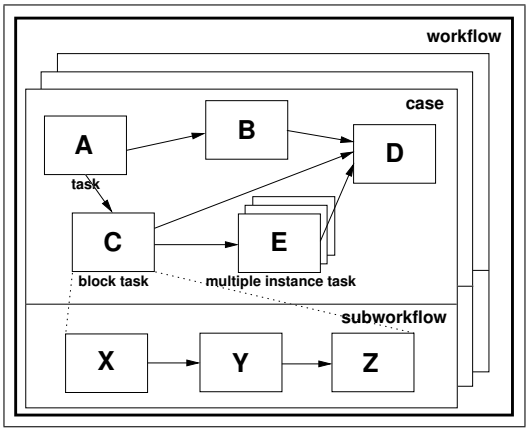

Fig. 1. A overview of the basic workflow concepts

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*, *multiple-instance* and *multiple-instance block*. 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 and 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 for 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 1, 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.

Figure 1 shows the control-flow perspective and only hints at the other perspectives (e.g., resources and data). Nevertheless, it sets the scene for discussing the resource perspective.

2.2 Resource Perspective

Typically, resources are needed to execute *work items*, i.e., invocations of tasks for specific cases. A resource is an entity that is capable of doing work and can be classified as either *human* or *non-human*, i.e., a resource that does not correspond to an actual person - e.g. plant and equipment. As mentioned in the introduction, we focus on human resources. However, many of the concepts and patterns also apply to non-human resources.

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 com-

mon set of business objectives. They usually have a specific *position* within that organisation, which may have specific *privileges* associated with it. 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 specific set of business functions. Details of the organisational structure in which a human resource may operate are not discussed in this paper. Interested readers are referred to [12], where these issues are examined in more detail.

A resource may have one or more associated *roles*. *Roles* serve as another grouping mechanisms 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. A comprehensive resource meta-model is presented in [12].

2.3 Lifecycle of a Work Item

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 2 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.

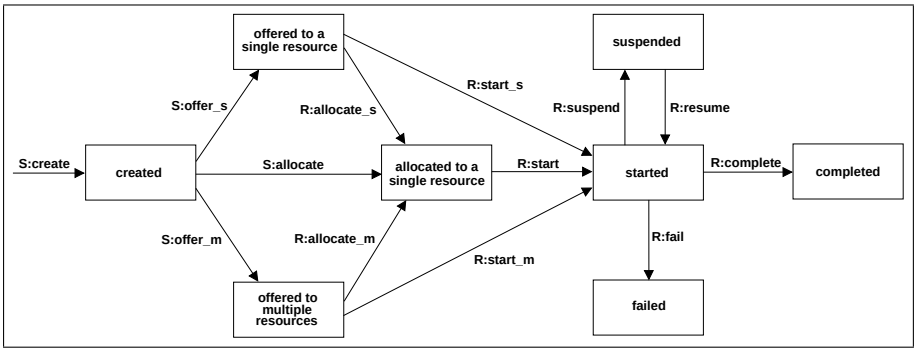

Fig. 2. State Transition Diagram for Work Distribution

Each node in Figure 2 represents a possible state of a work item. Each edge within this diagram is prefixed with either an S or an R indicating that the transition is initiated by the workflow system (S) or resource (R) respectively.

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 been allocated to a resource for execution. In state *created* the system can take one

of three possible courses of action: (1) offer the work item to a single resource, (2) allocate it to a resource, or (3) offer it to multiple resources. The difference between *allocating* and *offering* to a single resource is subtle. If a work item is allocated, there is a commitment on the part of the resource to execute the work item. Note that this commitment may be imposed by the system via *S:allocate* or by the resource him/herself (via *R:allocate_s* or *R:allocate_m*). No such commitment is implied for offered work items. Depending on the system and the resource, an offered or allocated work item can be started (cf., state *started*). Started work items can be completed, suspended (followed by a resume) or failed.

It is important to note that Figure 2 is just an example of a lifecycle model for work items. Some of the states and transitions may not be present in a given system. Moreover, when we consider more advanced patterns we need to assume an extended lifecycle model, e.g., to capture delegation.

3 Resource Patterns

The patterns presented in this section are intended to be language independent and do not assume a concrete syntax. In the absence of a commonly agreed workflow model, the aim is to define them in a form that ensures they are applicable to the broadest possible range of PAIS. The patterns are grouped into a number of categories: *creation patterns*, *push patterns*, *pull patterns*, *detour patterns*, *auto-start patterns*, *visibility patterns*, and *multiple resource patterns*. In this section we only describe a selection of the resource patterns. For a complete overview and detailed descriptions we refer the reader to [12].

3.1 Creation Patterns

The first category of patterns refers to the various restrictions that can be defined for a work item at design time. These restrictions can refer to the range of resources that may execute the work item ("Who?"), the time at which the work item should be executed ("When?") and the location at which the work item can be processed ("Where?"). Note that the creation patterns refer to the design phase, i.e., before there are any work items in existence. Therefore, they cannot be linked directly to Figure 2.

We have identified eleven creation patterns but in this paper we only describe the second of these: role-based allocation.

Pattern 2: 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*.

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 runtime. 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 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 runtime.

Implementation All of the workflow systems examined support role-based allocation.

The other ten creation patterns describe additional constraints that can be specified at design time, e.g., the "separation of duties" pattern (also known as the four-eyes principle) which specifies that certain tasks cannot be executed by the same person within the same case. Table 1 lists all eleven creation patterns.

3.2 Push Patterns

Push patterns refer to the ability of the *system* to offer or allocate work items. These patterns relate to state transitions `S:offer_s`, `S:offer_m` and `S:allocate` shown in Figure 2. Transition `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, and `S:allocate` corresponds to a work item being directly allocated to a specific resource immediately after it has been created. The main difference between an *offered* work item and an *allocated* work item is the level of commitment implied (i.e. whether the resource is committed to executing the work item or has merely been advised of its existence). In the case where a work item is offered to multiple resources, the initiative is left with the resources to determine which of them will actually execute it. This contrasts with allocated work items where the decision is made by the system. However, if the work item is offered to a single resource, the difference is more subtle and in some systems it will be difficult, if not impossible, to distinguish between `S:offer_s` and `S:allocate_s`.

There are nine push patterns (cf. Table 1). Here we only discuss two of them: patterns 13 and 17.

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

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*.

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 undertake the activity. Note that this pattern corresponds to the ability to take transition `S:offer_m` in Figure 2.

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 group members 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 ways in which a resource can be advised of group work items: (a) it may appear on each of their individual work queues, (b) each resource may have a distinct work queue for group items on which it may appear, or (c) 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.

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

- WebSphere 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 all resources to whom they are offered. When an advertised 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 an advertised 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 an advertised work item has been accepted by a resource, it is removed from all offered work queues and only appears on the work queue for the resource which has accepted it.

Pattern 17: 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.

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 queue although other interpretations are possible. Note that this pattern corresponds to a specific realization of transition `S:allocate` in Figure 2, i.e., it is an allocation by the system based on the number of already allocated work items.

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. COSA directly implements this allocation method via the `fewwork()` function. iPlanet provide facilities for programmatically extending the work item distribution algorithm indirectly enabling this pattern to be achieved.

3.3 Pull Patterns

While push patterns focus on work distribution from the system's perspective, pull patterns consider the issue from the resource's viewpoint. After the system offers or allocates a work item, the resource can take the initiative to allocate or start a work item. This is reflected by the transitions `R:allocate_s`, `R:allocate_m`, `R:start_s`, `R:start_m`, and `R:start` in Figure 2. There are six pull patterns, all related to these transitions. Here, we only discuss two of them: patterns 21 and 23.

Pattern 21: 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 commences working on one of these now.

Motivation This pattern provides a means for a resource to signal its intention to execute a given work item at some point in time although it may not commence working on it immediately. As a consequence, the work item is considered to be allocated to the resource and it cannot be allocated to or executed by another resource. Clearly this pattern corresponds to transitions `R:allocate_s` and `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. FLOWer directly supports this pattern through its case query construct. 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.

Pattern 23: 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.

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 started working on it, i.e., there is no explicit allocation phase. 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 transitions `R:start_s` and `R:start_m` in Figure 2.

Implementation This approach to work distribution is adopted by Staffware, WebSphere 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 as well as 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.
Patterns 21 and 23 do not convey the fact that work can be offered in various ways, e.g. the order in which work is offered may be determined by the system or by the resources. Table 1 provides a complete listing of pull patterns and gives an insight into the various bases on which work items can be offered to resources.

3.4 Detour Patterns

The state transitions shown in Figure 2 refer to the basic lifecycle of a work item, i.e., the "normal way" of handling work. However, there are situations where pre-existing work allocations are interrupted either by the workflow system or at the instigation of the associated resource. As a consequence of such events, the normal sequence of state transitions for a workflow item is varied. Patterns for dealing with these exceptional situations are called detour patterns. The range of possible scenarios for detour patterns is illustrated in Figure 3.

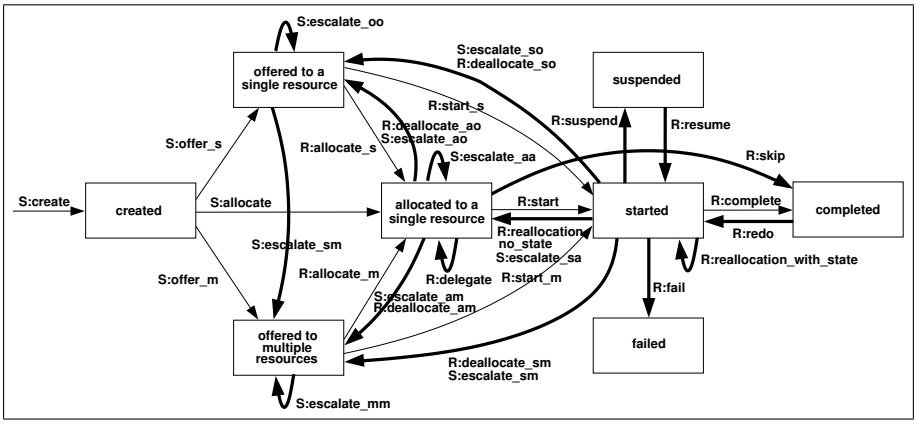

Fig. 3. Detour Patterns

As shown in Figure 3, detour patterns allow for alternative state transitions. Possible detour patterns 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.

– *deallocation* – 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 work on to another resource.
– *suspension/resumption* – where a resource temporarily suspends execution of a work item and recommences execution of it at a later time.
– *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.

In this paper we only describe one of the detour patterns in detail.

Pattern 27: 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 his outstanding work items onto the *Assistant Accountant*.

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. The pattern corresponds to transition `R:delegate` in Figure 3.

Implementation Generally the ability to delegate a work item is included in the client work list handler for a workflow engine. Staffware, WebSphere and COSA allow individual queued work items to be redirected to a given resource. COSA also has an enhanced notion of delegation in which all of the work items corresponding to a specific task definition can be redirected to another resource.

3.5 Auto-start Patterns

Auto-start patterns relate to situations where the 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 a 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 in Figure 4.

In this section we describe two of the four patterns: piled execution (pattern 38) and chained execution (pattern 39). These correspond to transitions `S:piled_execution` and `S:chained_execution` in Figure 4. Note that the corresponding arc is dashed because the `completed` state of one work item is connected to the `started` state of the *next* work item.

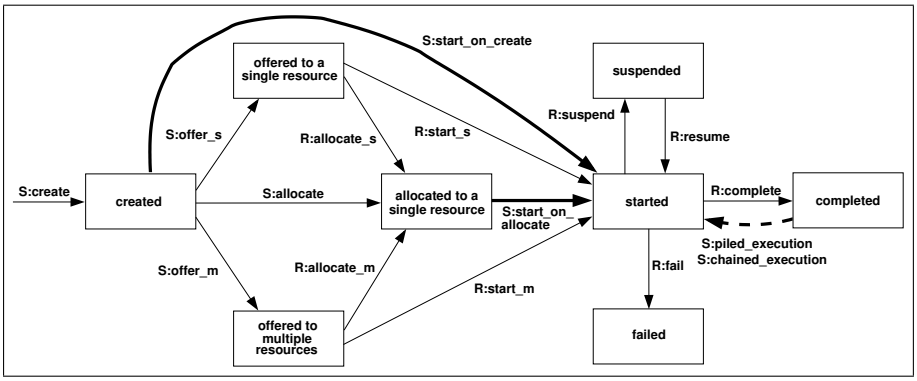

Fig. 4. Auto-start Patterns

Pattern 38: 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*.

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 (e.g., reduced set-up time, increased familiarity with the task).

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 systems examined support it.

Pattern 39: R-CE (Chained Execution)

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

Example Immediately commence the next work item(s) in the *Emergency Rescue Coordination* process when the preceding one has completed.

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 busy progressing work items in a given case.

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.

3.6 Additional Patterns

In [12] we identify additional patterns dealing with *visibility* (which work items are visible and who can see them) and *multiple resources* working on the same work item (e.g., patterns related to the formation of project teams). In this paper, we will not elaborate on these patterns and we will also not include them in the product evaluation presented in the next section.

4 Evaluation of Existing Workflow Products

The resource patterns discussed in this paper have been collected for several reasons. First of all, the patterns serve as a way to describe workflow functionality in a tool-independent manner. Second, the patterns can be used to train workflow developers and consultants. Last but not least, the patterns can be used to select workflow products or other PAIS. In this paper, we focus on five products: Staffware Process Suite version 9 (TIBCO), WebSphere MQ Workflow 3.4 (IBM), FLOWer 3 (Pallas Athena), COSA 4.2 (TRANSFLOW) and iPlanet 6.0 (SUN). An assessment scale with three possible values is used for evaluating these products with "+" indicating direct support for the pattern, "+/–" indicating partial support and "–" indicating that the pattern is not supported. The specifics of the rating criteria used are described in [12]. Although "–" indicates that there is no (direct) support for the pattern, it does not imply that it is impossible to realise the pattern. Often it is possible to realise the functionality outside of the system, e.g., by calling an external program. However, such workarounds are not considered to be a feature of the product itself. See [2] for a similar distinction used when evaluating systems against control-flow patterns.

Lines 1 to 11 of Table 1 shows the support for creation patterns. Only the first two patterns are supported by all of the products examined.

Lines 12 to 20 of Table 1 report on push patterns. It is surprising to note that not all systems support simple allocation mechanisms such as round robin allocation. Lines 21 to 26 illustrate that pull patterns are supported by most systems. In particular, Staffware, FLOWer and COSA allow for greater initiative on the part of resources in selecting and commencing work items.

Detour patterns support is shown by lines 27 to 35 of Table 2. COSA provides resources with a significant degree of autonomy in managing their workload. Other workflow systems do not support the same degree of flexibility although it is worth noting that the case handling foundation of FLOWer lessens the need for such "detours" as work items within a case are generally handled by the same resource. Finally, lines 36 to 39 show the support for auto-start patterns. As was

Nr	Pattern	Staffware	WebSphere	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)	+	–	+	+	+
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)	–	–	–	–	–
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 1. Support for creation, push and pull patterns in workflow systems

discussed in Section 3.5, it is remarkable that none of the systems support piled execution while only one of them supports chained execution.

5 Related Work

This paper is part of a bigger initiative (cf. www.workflowpatterns.com) to capture the functionality of PAIS in terms of patterns. It complements the control-flow [2] and data [11] patterns. This initiative provides a comprehensive pattern library for PAIS. There are a variety of other patterns libraries which target specific domains such as software engineering (`http://hillside.net/pattern`),

Nr	Pattern	Staffware	WebSphere	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-S (Suspension)	+/–	+/–	–	+	–
33	R-SK (Skip)	–	–	+	+	–
34	R-REDO (Redo)	–	–	+	–	–
35	R-PRE (Pre-Do)	–	–	+	–	–
36	R-CC (Commencement on Creation)	–	–	–	+	–
37	R-CA (Commencement on Allocation)	–	+	–	–	–
38	R-PE (Piled Execution)	–	–	–	–	–
39	R-CE (Chained Execution)	–	–	+	–	–

Table 2. Support for detour and auto-start patterns

interaction design (`http://www.welie.com`), and user interface and HCI design (`http://www.cs.ukc.ac.uk/people/staff/saf/patterns/gallery.html`).

Research on resource/organisational modelling in workflow systems has been relatively limited as discussed in [4, 8]. The RBAC (Role-Based Access Control) model [7] represents one approach for determining suitable users for a task. The salient features of RBAC are that permissions are associated with roles and users are made members of roles, thereby acquiring the associated permissions. RBAC models are useful but they have limitations, in particular they are primarily permission oriented, from a security point of view, and neglect other aspects of the organisation such as resource availability. Bussler and Jablonski [5] undertook pioneering work in the area, identifying many limitations of workflow systems in modelling policy and organisational issues. Several researchers [3, 9, 10] have developed so-called meta-models, i.e., object models describing the relationships between workflow concepts, including aspects of work allocation. However, these meta-models typically do not describe the dynamic aspects of work distribution.

6 Conclusion

In this paper we have presented a collection of resource patterns in the context of process-aware information systems. In addition, we have evaluated five workflow management systems using these patterns. We have only highlighted a selection of the available patterns and refer the reader to [12] for a complete description of all patterns and a detailed assessment of the five products. The main contribution of this work is that it is the first systematic analysis of the resource allocation/work distribution functionality desired in PAIS. The "patterns paradigm" was used to represent the different types of functionality. We do not claim completeness, but it is fair to say that the more than forty patterns identified provide a good coverage of the functionality provided by existing

workflow management systems, as illustrated by the results presented in Section 4. Moreover, the patterns do not only apply to workflow technology but also to other process-aware systems, e.g., ERP systems such as SAP and PeopleSoft, which can be analysed and improved through their use.

References

1. W.M.P. van der Aalst. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003.
2. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
3. W.M.P. van der Aalst and A. Kumar. Team-Enabled Workflow Management Systems. *Data and Knowledge Engineering*, 38(3):335–363, 2001.
4. 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, *Proceedings of the 18th Annual ACM Symposium on Applied Computing (SAC 2003)*, pages 603–608. ACM Press, 2003.
5. C. Bussler and S. Jablonski. Policy Resolution for Workflow Management Systems. In *Proceedings of the 28th Hawaii International Conference on System Sciences*, page 831. IEEE Computer Society, 1995.
6. F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.0. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2002.
7. 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.
8. A. Kumar, W.M.P. van der Aalst, and H.M.W. Verbeek. Dynamic Work Distribution in Workflow Management Systems: How to Balance Quality and Performance? *Journal of Management Information Systems*, 18(3):157–193, 2002.
9. M. zur Muehlen. *Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems.* Logos, Berlin, 2004.
10. 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.
11. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns. QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004. `http://www.citi.qut.edu.au/about/research_pubs/technical.jsp`.
12. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Resource Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004. `http://fp.tm.tue.nl/beta/`.
13. WFMC. Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface – XML Process Definition Language (XPDL) (WFMC-TC-1025). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2002. `http://www.wfmc.org/standards/docs.htm`.