

# *new*YAWL: Towards Workflow 2.0\*

Nick Russell<sup>1</sup> and Arthur H.M. ter Hofstede<sup>2</sup>

<sup>1</sup>Eindhoven University of Technology,  
PO Box 513, 5600MB, Eindhoven, The Netherlands  
`n.c.russell@tue.nl`

<sup>2</sup>Queensland University of Technology,  
PO Box 2434, QLD, 4001, Australia  
`a.terhofstede@qut.edu.au`

**Abstract.** The field of workflow technology has burgeoned in recent years providing a variety of means of automating business processes. It is a great source of opportunity for organisations seeking to streamline and optimise their operations. Despite these advantages however, the current generation of workflow technologies are subject to a variety of criticisms, in terms of their restricted view of what comprises a business process, their imprecise definition and their general inflexibility. As a remedy to these potential difficulties, in this paper we propose a series of development goals for the next generation of workflow technology. We also present *new*YAWL, a formally defined, multi-perspective reference language for workflow systems.

## 1 Introduction

Workflow management technology provides support for the execution of business processes. A workflow management system routes work to the right people or software applications at the right time. While these types of systems have been around in one incarnation or another for several decades (e.g. office automation systems originated in the seventies [26,11]), it wasn't until fairly recently that they reached a level of maturity where their broader uptake was feasible. Workflow technology allows businesses to save time and money by providing them with a means of taking charge of their processes. Not only does it support the execution of business processes, but they can also be more easily analyzed, monitored, audited, and adapted. Increasingly now, the term Business Process Management (BPM) is used to indicate that the field has moved beyond mere process specification and execution to encompass a holistic view of the business process as a corporate asset that merits ongoing maintenance and refinement.

A business process can be viewed from a number of different viewpoints (see e.g. [15]). The *control-flow* perspective deals with the control-flow dependencies that exist between the various tasks in a business process, the *data* perspective

---

\* This research was conducted in the context of the *Patterns for Process-Aware Information Systems (P4PAIS)* project which is supported by the Netherlands Organisation for Scientific Research (NWO).

deals with the data required by and produced by these tasks, while the *resource* perspective deals with the way in which tasks are allocated to resources. Starting in the late 1990's, the Workflow Patterns Initiative<sup>1</sup> began cataloging patterns in these perspectives. These patterns identify recurrent concepts that arise when modelling business processes, and provide assistance with tool selection, process specification and language development. Over time, process modelling languages, workflow management systems, research prototypes, and (proposed) standards in the BPM field have been examined in terms of these patterns, revealing their relative strengths, weaknesses and opportunities for improvement.

Despite the rapid advances in BPM technology, there have been significant obstacles to full realisation of the benefits that it promises to deliver. To some degree, these obstacles can be explained through the lack of consensus in regard to the conceptual, formal, and technological foundations of BPM. Moreover none of the standards that have been proposed have achieved any degree of uptake. As an example, consider the Workflow Management's XPDL 1.0, introduced in the 1990's. Not only does it lack a formal foundation, allowing it to be interpreted in substantially different ways (cf. [17]), but it also has a very limited range of functionality as demonstrated by its minimalistic support for the workflow patterns (cf. [1]). More recent standards proposals also lack a formal foundation (cf. BPEL, BPMN) and while their overall capabilities have noticeably improved (e.g. as demonstrated by the extent of their workflow pattern support), they still exhibit minimal support for the resource perspective [21]. It is also worth remarking that the operation of a number of commonly utilised workflow constructs are actually quite complex to capture precisely (an interesting illustration of this is the concept of the OR-join in the control-flow perspective to which whole publications have been dedicated, see e.g. [2,25]). This also holds for interdependencies between constructs in different perspectives (consider e.g. concurrency issues in the control flow perspective and how these may affect resources). These considerations underscore the fact that it is extremely difficult, if not impossible, to describe a powerful business process language informally and be precise enough to avoid ambiguities with respect to its interpretation at runtime.

Many of the challenges in the BPM field stem from a lack of a proper conceptual and formal foundation. In this paper we will focus on this issue and demonstrate that it is possible to fully define a comprehensive reference language for both the specification and enactment of business processes. In doing so, in Sect. 2 we first examine the field of BPM in an historical context. Then in Sect. 3 we propose a fundamental set of requirements for the next generation of workflow languages. Section 4 proposes a concrete reference language, *newYAWL* [22,21], which offers powerful support for the workflow patterns and is formally defined using Coloured Petri nets. Finally Sect. 5 concludes the paper.

---

<sup>1</sup> [www.workflowpatterns.com](http://www.workflowpatterns.com)

## 2 Workflow 1.0: The Journey So Far

The introduction of workflow technology follows a historical trend where application considerations have been progressively separated from file management, data management and user interface considerations through the introduction of operating systems, database management systems, and GUIs. Workflow management systems take this trend a step further by making explicit the dependencies that exist between various applications and the activities conducted by human resources as well as the strategies that are associated with allocating work to resources. An in-depth description of these historical developments can be found in [24].

Another way of looking at modern workflow management from an historical perspective is to consider the various fields that have influenced its development. These include office automation, document management, advanced transaction models and groupware. Coordination between various participants and applications in work processes are a primary concern in office automation. Similarly in document management, information needs to be routed between a number of participants and may be modified along the way.

Yet another way of looking at developments in the BPM field is by considering the lifecycle of business processes. In the past the emphasis has been on process modelling and enactment, however in recent years there is the motivation to “close the loop” and include monitoring and diagnosis as part of the BPM lifecycle, thereby viewing processes as continually adapting to changes in their operating environment [10]. This trend is reflected in the emergence of the field of process mining, where process deployments are analysed through the events that they have generated during their execution [5].

It is striking that in the BPM field there has been an abundance of (proposed) standards over the years, involving a number of standardisation bodies such as WfMC, OASIS and OMG, however their impact has not been as profound or long-lasting as it should have been. One explanation for this is that standards were proposed before a sufficient understanding had been achieved of the concepts involved. Another reason is the fact that these proposals were informally defined, leaving scope for ambiguities and resulting in implementations of the same standard exhibiting fundamental differences. In some cases the standards were not sufficiently powerful and vendors defined their own extensions to address this. For example Oracle BPEL’s support for the resource patterns [18] is significantly stronger than what is proposed by the standard itself.

The informal definition of standards poses a significant problem with distinct implementations choosing varying interpretations of individual constructs and most recently there has been an increased effort (cf [19,9]) to provide a formal semantics for widely used languages such as BPMN and BPEL. Even within widely utilised offerings such as Staffware, the lack of a precise operational semantics can result in the same design-time model yielding different runtime outcomes for the same data inputs (cf [21]).

Increasingly, information systems need to be able to operate in dynamic environments where interactions are required with distributed, autonomous and

evolving components. Technological developments in the BPM field need to follow suit. Therefore, if tasks can be assigned to web services, web service composition can be achieved through the use of a workflow management system, and Service-Oriented Architectures provide the principles upon which open and flexible systems can be built. Ultimately aspects of this type of interaction also need to be reflected at the modelling level, paving the way for new types of powerful communication primitives (e.g. [8]).

There are a series of important lessons that have been learnt during the first phase of workflow development activities. The MOBILE [15] and WIDE [12] projects have demonstrated the importance of taking multiple perspectives into account when designing and enacting business processes, yet most contemporary standards and offerings tend to be control-flow centric. Process flexibility continues to be a focus of many workflow initiatives and a recent survey [23] identified four distinct approaches (design, deviation, adaptation and change) to its facilitation, however most offerings are only able to demonstrate capabilities in one or two of these areas. The ADEPT [20] project is one of the most mature research initiatives in terms of the flexibility support that it provides and is one of the few research endeavours to approach commercial strength. Whilst many process technologies tend to focus on the normal or expected sequence of execution events, it is the ability to handle the exceptional cases that marks their effectiveness and not surprisingly there has been a wide body of research in this area (cf [13,7,6,14]) much of which has still not made its way into mainstream offerings.

The aforementioned issues point to the need for a reference language that can offer guidance for future technology initiatives on the breadth of capabilities that they need to embody. Such a language should be able to precisely capture the broad range of concepts which underpin contemporary business processes and be formally defined in order to avoid any ambiguities in their interpretation. YAWL [3] (Yet Another Workflow Language) was designed to provide comprehensive support for the original control-flow patterns [4] however it is increasingly clear that it needs to be comprehensively overhauled in order to ensure that it provides an appropriate level of support for the range of concepts described above. In this paper we propose a radical extension, termed *newYAWL*, that aims to achieve this goal and show that a comprehensive formalisation of its operation is possible using Coloured Petri nets.

### 3 Workflow 2.0: Requirements for the Next Generation

It is clear from the preceding discussion that despite the inherent benefits offered by workflow technology, they only deliver part of the solution required to effectively automate contemporary business processes. Existing offerings tend to focus on providing support for control-flow aspects and provide significantly less facilities for managing other important aspects such as the data and resource perspectives. Moreover they have a narrow view of the overall business process management lifecycle and in many cases limit their area of operation to busi-

ness process enactment with somewhat less consideration of other design-time and ongoing operational issues (e.g. monitoring, refinement) that are associated with the deployment of such processes. In an attempt to lay down a clear vision for future workflow technology, in this section, we identify a series of core requirements that the next generation of workflow tools should seek to address.

**REQ1: Multi-perspective support** Existing tools offer a control-flow centric view of a business process both in terms of the way in which they are defined and also in the manner in which they are deployed. Whilst the control-flow perspective is central to business processes, markedly more focus needs to be given to other significant aspects. The data and resource perspectives in particular need to be considered as first-class citizens and direct support for modelling and enacting the data representation, data passing, resource definition and work routing issues encountered in the context of business processes is required.

**REQ 2: Integrated modelling and enactment** Traditionally, the modelling and enactment of workflows were considered to be distinct activities. Workflows were described using high-level business process modelling formalisms that focused on capturing the "spirit" of the overall business objective. Often this activity was undertaken by business analysts and the results of this work were passed to technical staff who mapped it to an equivalent workflow definition that contained sufficient detail in order for the process to actually be enacted. Often the modelling and enactment activities utilised differing technologies. Obviously the gap between these technologies leaves open the potential for ambiguities and inconsistencies to be introduced into the resultant automated business process. For this reason, an approach to modelling workflow processes is required which involves their specification in sufficient detail that they can be directly enacted.

**REQ 3: Support for flexible process design and enactment** One of the ongoing criticisms of production workflow systems is that they enforce rigid processes on users that hamper rather than assist them in reaching their end goals hence obviating any potential benefits in process automation. Consequently there is now an increased focus on what process flexibility means and how it can be facilitated. In order to offer material benefit in this area, an offering needs to provide a wealth of design-time constructs for embodying flexibility in processes as well as offering runtime facilities that allow for controlled deviation from the prescribed process model, execution of underspecified process models, and adaptation and change of processes during execution.

**REQ 4: Language constructs mirroring those encountered in practice** One of the fundamental difficulties associated with many workflow modelling languages is the fact that their core constructs are developed in isolation from the business processes that they are ultimately intended to facilitate. This leads to difficulties in capturing many of the actual situations that are encountered in practice. The only practical solution to resolving this impasse is to actually derive the range of constructs within a workflow language from those encountered in the real world. In order to do so, a comprehensive catalogue of the actual modelling considerations encountered in practice is required.

**REQ 5: Deterministic runtime model** One of the common issues arising in workflow languages stems from their informal definition. In most cases, the constructs which make up the workflow modelling language are not formally defined and consequently do not have a precise operational semantics. This means that there is a degree of inherent ambiguity associated with their usage. In order for this difficulty to be resolved, it is necessary to provide a precise operational definition for each of the constructs in a workflow language.

**REQ 6: Comprehensive concurrency support** One of the early drivers for workflow technology was to provide more efficient ways to distribute the activities associated with business processes throughout the resources within an organisation. However in many cases current workflow offerings demonstrate a surprising lack of support for managing the concurrency inherent in these processes. There are considerations that arise at a number of levels including: providing a means of facilitating concurrent task execution within a process instance, managing the use of data elements by multiple concurrent activities, handling timing issues associated with the trafficking of data elements between concurrent activities both within and between a workflow and the broader operational environment, and managing the advertisement and distribution of work between multiple resources in a predictable and reliable way. The next generation of workflow technology needs to provide broad support for all of these needs.

**REQ 7: Graceful handling of expected and unexpected exceptions** One of the great benefits offered by widespread adoption of workflow technology is that it offers organisations the opportunity to move towards a management by exception regime. In this scenario, the handling of normal process instances is automated as far as possible and only deviations from the expected behaviour are subject to manual scrutiny. In order to accomplish this, a process needs to embody support for as much exception handling as possible in order to deal with *expected* errors and similarly, it also needs to be resilient when experiencing *unexpected* exceptions and provide users with the ability to intervene in a process instance in order to instigate appropriate corrective action.

**REQ 8: Recognition of the full BPM lifecycle** The first generation of workflow tools essentially focused on the automation of business processes. The emphasis being on the quantity rather than the quality of the automation achieved. The next generation of workflow technology needs to refine this approach to business process enablement and provide an increased range of options for analysing the resultant automation both to ensure its correctness and consistency from a design-time standpoint and also to examine its efficiency and effectiveness from a runtime perspective. In addition to a broader range of design-time aids to assist process developers, this also necessitates the recording of a much richer range of execution results for subsequent analysis and reflection.

In the following section, we will present an overview of *newYAWL*, a workflow reference language which aims to address these requirements and shows the form that automated support for business processes may take in the future.

## 4 *newYAWL*: A Blueprint for Workflow 2.0

This section provides an overview of the operational capabilities of *newYAWL*, a reference language for workflow systems based on the workflow patterns. *newYAWL* aims to lay a foundation for the next generation of workflow technology and describes a workflow modelling and execution language that incorporates comprehensive support for the control-flow, data and resource perspectives. It has a deterministic execution model whose operational semantics are defined in terms of Coloured Petri nets. This approach to characterising *newYAWL* means that it not only has a precise static definition but also that its approach to dealing with the dynamic issues that arise during execution is fully specified.

### 4.1 Language Overview and Format

*newYAWL* provides a comprehensive reference language for describing business processes that are to be enacted as workflows. The language constructs in *newYAWL* are informed by the various workflow patterns, hence they have a direct correspondence with the fundamental elements which are actually encountered in real-world business processes and consequently have general applicability. The *newYAWL* language is specified in two parts. It has a complete abstract syntax which identifies the characteristics of each of the language elements and their configuration. Associated with this is an executable, semantic model — presented in the form of Coloured Petri nets — which defines the runtime semantics of each of the language constructs.

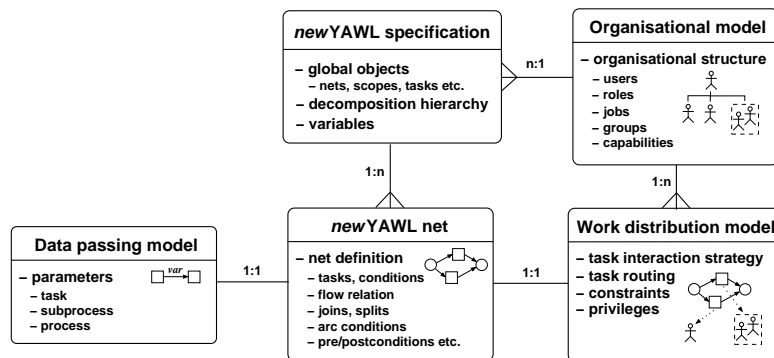


Fig. 1. Schema definition for *newYAWL* abstract syntax

The abstract syntax for *newYAWL* provides an overview of the main concepts that are captured in a design-time business process model. It is composed of five distinct schemas, each of which is specified on a set-theoretic basis. Fig. 1 summarises the content captured by each of the individual schemas and the relationships between them. Each process captured using the *newYAWL* abstract

syntax has a single instance of the *newYAWL* specification associated with it. This defines elements that are common to all of the schemas and also captures the decomposition hierarchy. Each *newYAWL* specification is associated with an instance of the organisational model that describes which users are available to undertake tasks that comprise the process and the organisational context in which they operate.

A *newYAWL* process can be made up of a series of distinct subprocesses (where each subprocess specifies the manner in which a composite task is implemented) together with the top-level process. For each of these (sub)processes, there is an instance of the *newYAWL* net which describes the structure of the (sub)process in detail in terms of the tasks that it comprises and the sequence in which they occur. Associated with each *newYAWL* net is a data passing model which defines the way in which data is passed between elements in the process in terms of formal parameters operating between these elements. There is also a work distribution model that defines how each task will be routed to users for execution, any constraints associated with this activity and privileges that specific users may have assigned to them. The collective group of schemas for a specific process model is termed a *complete newYAWL* specification.

One of the virtues of specifying the operational semantics of *newYAWL* in terms of Coloured Petri nets is that the CPN Tools [16] offering provides an executable environment for models developed in this formalism. This means a candidate *newYAWL* model can actually be executed in order to verify its consistent operation. There is a two stage process for mapping a *newYAWL* specification defined in terms of the abstract syntax to an initial marking of the semantic model in CPN Tools. In the interest of brevity, this process is not discussed here but full details of its operation can be found in [21].

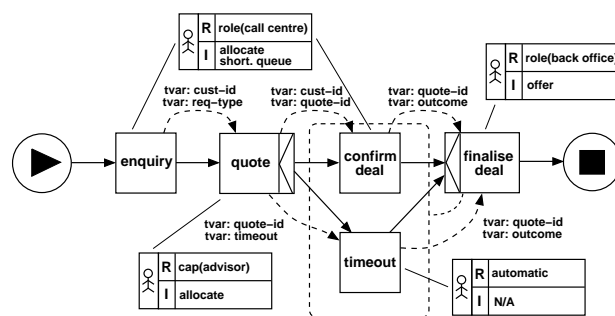
The complete operational semantics for *newYAWL* is based on a series of 55 CPN models <sup>2</sup>. Fig. 3 shows the top level model which illustrates the main events during the lifecycle of a process instance as transitions and the information elements required to support its execution as places. Each of the events takes the form of a substitution transition indicating that it has a more detailed underlying definition, however in summary, the **start case** transition is responsible for initiating a new process instance. Then there are a succession of **enter**→**start**→**complete**→**exit** transitions which fire as individual task instances are enabled, the work items associated with them are started and completed and the task instances are finalised before triggering subsequent tasks in the process model. Each atomic work item is distributed to a suitable resource for execution via the **work distribution** transition. This cycle repeats until the last task instance in the process is complete, at which point the **end case** transition terminates the process instance. Data interchange with the operating environment is facilitated by the **data management** transition and the **add** transition enables additional task instances to be dynamically instantiated for designated (multiple instance) tasks.

---

<sup>2</sup> Available from <http://www.yawl-system.com/newYAWL>.

The places in the *newYAWL* CPN model divide into two main groups: (1) *static* places which capture the various components of a *newYAWL* process model such as the flow relation, task details, variable declarations, parameter mappings, preconditions, postconditions, scope mappings and the hierarchy of processes and subprocesses. These correspond to the design-time information captured about a *newYAWL* process as illustrated in Fig. 1 and remain unchanged during the execution of a process. (2) *dynamic* places which capture the *state* of a process instance during its execution and include items such as the current marking of each place in the flow relation, variable instances and their associated values, locks which restrict concurrent access to data elements, details of subprocesses currently being enacted, folder mappings (identifying shared data folders assigned to a process instance) and the current execution state of individual work items (e.g. *enabled*, *started* or *completed*).

An indication of the information content of an actual instance of a *newYAWL* process model is illustrated in Fig. 2 which shows a simple process for responding to a customer request for foreign exchange services. On the basis of a customer enquiry about a prospective deal, a quote is prepared and forwarded to them. They have 24 hours to respond to the quote and confirm they wish to proceed, otherwise it is withdraw. No later than 24 hours after the quote is issued, the deal is finalised, either as a result of a specific customer response or because of a timeout. For each of the tasks in the process, there is a specific routing strategy describing who should undertake the task and a specific interaction strategy indicating the basis on which it should be distributed to a potential resource for subsequent execution. For example, the **quote** task is undertaken by the resource that demonstrates the capability of being the advisor for the customer and it is directly allocated to (only) them for execution where as the **finalise deal** task is offered to all resources who are part of the back office role with the expectation that one of them will elect to execute it at a future time. The relevant data elements and data passing strategy are also part of a *newYAWL* process model and the *tvar* entries indicate that certain task-level data elements are passed between tasks. For example, the **quote** task receives the



**Fig. 2.** Indicative example of a *newYAWL* process model: customer request for foreign exchange services

*cust-id* and *req-type* parameters from the `enquiry` task and passes on the *cust-id*, *quote-id* and *timeout* parameters to subsequent tasks. Despite the relative simplicity of this example, it gives an insight into the breadth of information in various perspectives that potentially can be captured in a *newYAWL* process model. Further details on the specifics of each of the perspectives are included in the following sections.

## 4.2 Control-Flow Perspective

A process model in *newYAWL* is analogous to a Petri net (although they are not the same and *newYAWL* includes some additional constructs such as the cancellation region that are not available in Petri nets). It consists of tasks (which correspond to executable activities) and conditions (that correspond to states) connected in the form of a directed graph. There is a designated start and end node for a process and there are two fundamental requirements pertaining to model structure: (1) all nodes (i.e. tasks and conditions) must be on a path from the start to the end node and (2) a condition may not be connected to another condition. Processes may be hierarchical in form with block tasks mapping to a corresponding subworkflow to which they pass control when invoked. Where multiple arcs enter or exit a task, various forms of join and split conditions are supported which describe the state requirements for task initiation and the effects of completion. In Fig. 3 the `flow relation` and `process hierarchy` places capture the details of individual *newYAWL* workflow models and the hierarchy that they form.

Control-flow in *newYAWL* is managed in an analogous manner to that in a Petri net and is based on the traversal of tokens through a process model. Each control-flow token identifies the process model and process instance to which it applies. In Fig. 3 the various control-flow tokens reside in the `process state` place. Tokens are removed from this place by enabled tasks and returned to it when they complete. The lifecycle of a task is illustrated by the `enter`, `start`, `complete` and `terminate` block and `exit` transitions which identify the various stages through which an enabled task passes as it progresses from initiation to completion.

Each process has a unique identifier known as a `ProcessID` and each process model has a unique `BlockID` (this is necessary as the hierarchy within a process means it may contain several distinct process models defining subworkflows in addition to the top-level model). Each task within a process is identified by a unique `TaskID`. In order to allow for and differentiate between concurrent execution instances, it is necessary to introduce some additional notions. First an executing instance of a process is termed a case. It has a case identifier `CID` which is unique for a given `ProcessID`. Hence the tuple `(ProcessID,CID)` uniquely identifies all cases. Similarly an enabled task instance is known as a work item. It has a more complex identification scheme denoted by the five-tuple `(ProcessID,CID,TaskID,Inst,TaskNr)` where `Inst` identifies the specific instance of the task that is being executed (thus allowing for distinct instances of a task as may occur if it is in a loop for example) and `TaskNr` which allows distinct

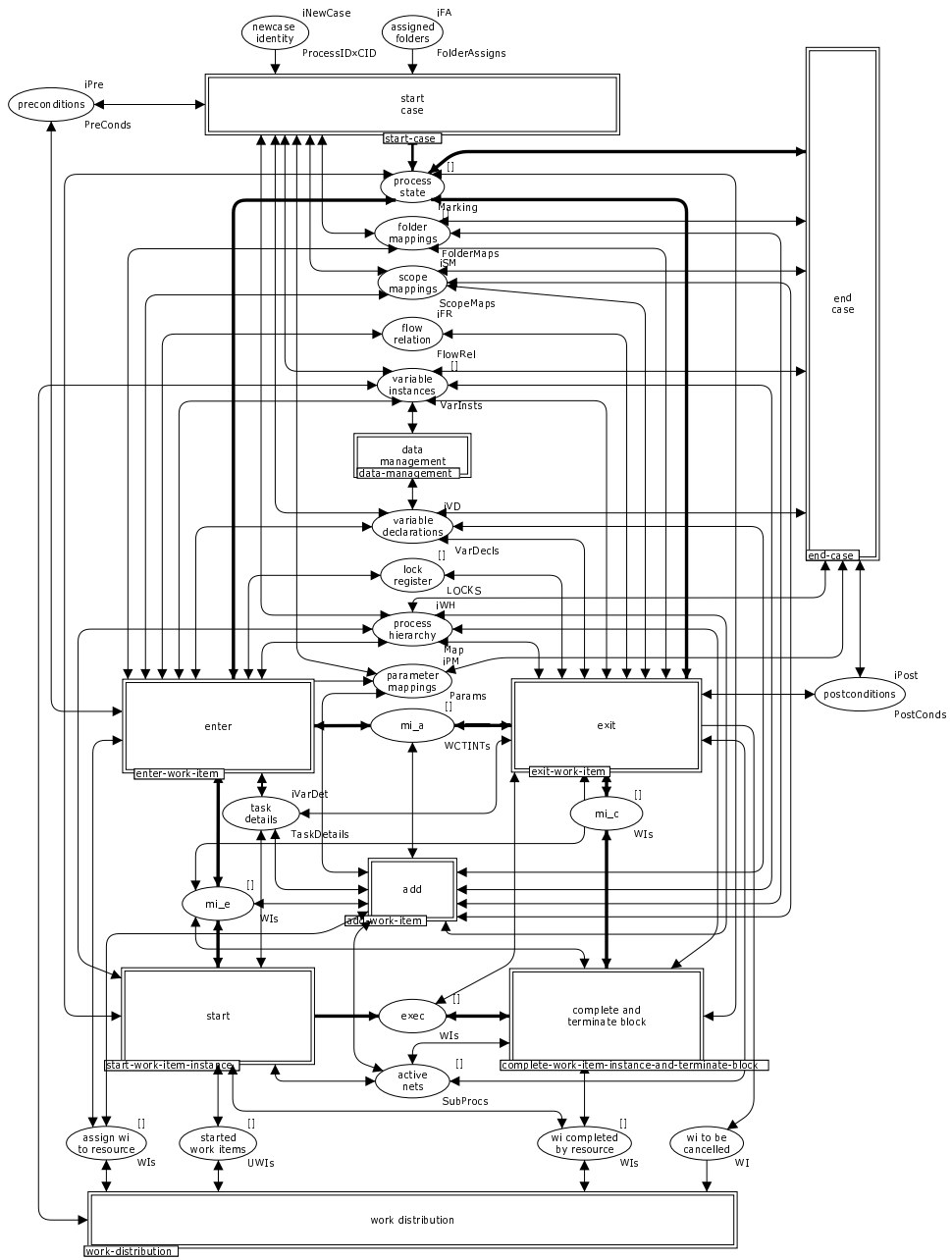


Fig. 3. Overview of the operational semantics for newYAWL

concurrent execution instances of a multiple instance task to be differentiated. By adopting this identification scheme, it is possible for the semantic model to cater for multiple concurrent processes, process instances and task instances in a common environment.

### 4.3 Data Perspective

*newYAWL* incorporates a series of features derived from the data patterns, providing coverage of issues such as persistence, concurrency management and complex data manipulation which are often absent from workflow languages. It provides support for a variety of *distinct scopes* to which data elements can be bound e.g. task, block, case etc. These are encoded in the `variable declarations` place shown in Fig. 3. Data passing between process constructs is based on the use of *formal parameters* which take a function-based approach to data passing thus supporting inline transformations during data passing events. These parameters are encoded in the `parameter mappings` place. A similar functional approach is taken to specifying *link conditions* for OR-splits and XOR-splits that allow the determination of which outgoing branches should be activated and *preconditions* and *postconditions* for tasks and processes. These are encoded in the `task details`, and `preconditions` and `postconditions` places respectively. Finally, the use of *locks* allows concurrent data usage to be managed through an approach which requires tasks to specify data elements that they require exclusive access to (within a given process instance) in order to commence. A task instance can commence execution if it can acquire locks on all required data elements. It retains these locks until it has completed execution preventing any other task instances from using the locked data elements concurrently. The locks are recorded in the `lock register` place.

### 4.4 Resource Perspective

The resource perspective is responsible for describing the resources who undertake a given business process and the manner in which associated work items are distributed to them and managed through to completion. For each task, a specific *interaction strategy* is specified which describes how the associated work item will be distributed to users, and what degree of autonomy they have in regard to choosing whether they will undertake it or not and when they will commence executing it. Similarly, a detailed *routing strategy* can be defined which identifies who can undertake the work item. Users can be specified by name, in terms of roles that they perform, based on capabilities that they possess, in terms of their job role and associated organisational relationships or based on the results of preceding execution history. The routing strategy can be further refined through the use of constraints that restrict the potential user population. Indicative constraints may include: *retain familiar* (i.e. route to a user that undertook a previous work item), *four eyes principle* (i.e. route to a different user than one who undertook a previous work item), *random allocation* (route to a user at random from the range of potential users), *round robin allocation* (route

to a user from the potential population on an equitable basis such that all users receive the same number of work items over time) and *shortest queue allocation* (route the work item to the user with the shortest work queue). *newYAWL* also supports two advanced operating modes *pled execution* and *chained execution* that are designed to expedite the throughput of work by imposing a defined protocol on the way in which the user interacts with the system and work items are allocated to them.

#### 4.5 *newYAWL*: Progress Towards Workflow 2.0

One of the major objectives of the *newYAWL* initiative was to advance the state of the art in workflow systems by providing a reference language for multi-perspective business processes (i.e. requirement REQ 1 in Sect. 3). In doing so, it has also addressed many of the goals identified in Sect. 3 for the next generation of workflow technology. The language constructs in *newYAWL* are based on the fundamental requirements for business processes identified by the 126 workflow patterns [21]. The experiential approach taken to characterising these patterns, based on a comprehensive survey of commercial offerings, standards, modelling formalisms and programming language theory, ensures a close correlation between the language constructs in *newYAWL* and those encountered in practice (REQ 4). Moreover, the fact that these constructs span multiple perspectives of a business process is also directly recognised in the breadth of the abstract syntax for *newYAWL*.

*newYAWL* is formalised in terms of a series of CPN models, which provide a precise interpretation for each of its language constructs and facilitates deterministic execution (REQ 5) of business processes captured in terms of its abstract syntax. One of the major advantages of the approach taken to the language definition for *newYAWL* is that there is a clear mapping from the abstract syntax to the runtime environment and a business process captured in terms of the abstract syntax can be directly executed without requiring further information (REQ 2). This means that business process models specified using *newYAWL* are applicable throughout the BPM lifecycle. They are used directly in the modelling, implementation and enactment phases of workflow processes and, in conjunction with the logging information recorded during execution, they provide the basis for comprehensive analysis of processes in retrospect. Moreover, the comprehensive description contained in a *newYAWL* business process when utilised in conjunction with the proposed execution logs (which include not only details of task execution, but also data transfer and the various stages of the lifecycle for individual work items) means that the basis exists for comprehensive monitoring and analysis of business processes during execution (REQ 8).

A significant consideration during the design of *newYAWL* was in ensuring that the resultant language ultimately provided a broader range of facilities at runtime than has been demonstrated by preceding workflow technology. The broad range of workflow patterns it supports (118 fully and 1 partially out of the complete set of 126 patterns) gives an indication of the breadth of its overall

capabilities, and it provides a range of useful features including support for complex data structures in workflow data elements, a variety of integration facilities with the operational environment, support for configurable exception handling (REQ 7), rich resource definition and the ability to specify a wide range of work item distribution mechanisms for routing work to users and managing it through to completion. It also demonstrates a range of facilities for flexible process enactment (REQ 3) particularly in the areas of flexibility by design and underspecification although less so in the areas of deviation, adaptation and change. Further information on *newYAWL* can be found in [22] and [21] including illustrative examples of its usage and detailed discussions of the language design and validation. One area of the *newYAWL* feature set that merits special attention, is the manner in which it facilitates concurrency in business processes (REQ 6). The following section discusses this issue in detail with reference to four specific issues that arise in the control-flow, data and resource perspectives.

#### 4.6 *newYAWL*: Better Concurrency Support for Processes

In this section, we discuss the concurrency support provided by *newYAWL* with reference to four specific examples: task enablement, concurrent data element usage, work item distribution and deferred choice.

**Task Enablement** One of the most complex activities in workflow execution is managing the enablement of a task. This entails two distinct steps: (1) determining if the various prerequisites that apply to task enablement have been met and (2) facilitating the actual enablement as a single atomic activity.

There are five requirements for a task to be enabled: (1) the precondition associated with the task must evaluate to true, (2) all data elements which are inputs to mandatory input parameters must exist and have a defined value, (3) all mandatory input parameters must evaluate to defined values, (4) all locks which are required for data elements that will be used by the work items associated with the task must be available and (5) if the task is a multiple instance task, the multiple instance parameter when evaluated must yield a number of rows that is between the minimum and maximum number of instances required for the task to be initiated. Only when all of these prerequisites have been met can the actual enabling of a task occur. The `enter` transition, illustrated in Fig. 3, is responsible for managing task enablement which involves the simultaneous completion of the following actions:

1. Removing the control-flow tokens marking input conditions to the task corresponding to the instance enabled from the `process state` place;
2. Determining which instance of the task this is;
3. Determining how many work item instances should be created. For an atomic or composite task this will always be a single work item, however for a multiple instance or composite multiple instance task, the actual number started will be determined from the evaluation of the multiple instance parameter contained in the `parameter mappings` place together with the current data

state in `variable instances` which will return a composite result containing a number of rows of data indicating how many instances are required. In all of these situations, individual work items are created which share the same `ProcessID`, `CID`, `TaskID` and `Inst` values, however the `TaskNr` value is unique for each work item and is in the range 1...number of work items created;

4. For all work items corresponding to composite tasks, distinct subprocess CIDs need to be determined from the `process hierarchy` place to ensure that any variables created for subprocesses are correctly identified and can be accessed by the work items for the subprocesses that will subsequently be triggered;
5. Creating variable instances for data elements associated with the task using the variable definitions corresponding to the task in the `variable definition` place. Data elements are added to the `variable instances` place;
6. Mapping the results of any input parameters for the task instance as identified in the `parameter mappings` place to the relevant task data elements. This uses data values from the `variable instances` place and updates any required input variables in this place created in step 5;
7. Recording any variable locks that are required for the execution of the task instance in the `lock register` place;
8. Creating work item distribution requests for the work item to be allocated to a specific resource. These are added to the `assign wi to resource` place for subsequent routing to resources; and
9. Finally, work items with an *enabled* status need to be created for this task instance and added to the `mi_e` place which identifies work items corresponding to enabled but not yet started tasks.

**Concurrent Data Element Usage** An issue that was addressed many years ago by the database community but which is surprisingly lacking in many workflow solutions is the ability to manage concurrent usage of data elements within a process instance. *newYAWL* addresses this issue by introducing the notion of locks which allow exclusive access to a data element to be retained by a specific task instance during its execution. Locks are evaluated at the time of task enablement and where a task requires a data element to which it cannot acquire a lock, then its enablement is deferred until it can do so. In Fig. 3, the `lock register` place holds details of locks that are currently pending.

**Work Distribution** The main motivation for workflow systems is achieving more effective and controlled distribution of work. Hence the actual routing of work items to specific resources and managing the interaction with the resource as they progress individual work items to completion are of particular importance. The process of managing the distribution of work items is summarised by Fig. 4 which shows how the `work distribution` transition in Fig. 3 is implemented. This transition coordinates the interaction between the workflow engine, and the `work item distribution`, `worklist handler`, `management intervention` and `interrupt handler` transitions. The specific functions provided by these transitions are as follows:



nents of the workflow engine (e.g. the control-flow process, exception handlers).

The distribution of work items to users from the workflow engine is initiated via the `work distribution` transition which forwards work items to the `work items for distribution` place. The `work item distribution` transition then determines how they should be routed to users. This may involve the services of the workflow administrator in which case they are sent to the `management intervention` transition or alternatively they may be forwarded directly to one or several users via the `worklist handler` transition. The various places between these three transitions correspond to the range of requests that flow between them.

The status of work items in progress is maintained in the `offered work items`, `allocated work items` and `started work items` places which are shared between the `work item distribution`, `worklist handler`, `management intervention` and `interrupt handler` transitions. Although much of the information about the state of work items is shared, the determination of when a work item is actually complete rests with the `work item distribution` transition. It inserts a token in the `completed work items` place when a work item is complete. Similarly, work item failures are notified via the `failed work items` place. Work items that are subject to some form of interrupt (e.g. an exception being detected and handled) are handled by the `interrupt handler` transition which manages cancellation, forced completion and failure requests received in the `cancel work item`, `complete work item` and `fail work item` places respectively. The complexity of the activities comprising the `work distribution` transition is underscored by the fact that each of them are also substitution transitions and in each case have a relatively complex underlying implementation.

**Deferred Choice** The implementation of the deferred choice construct is problematic for many workflow systems that do not have a notion of state. An example of such a situation in a process is where a commuter defers the choice as to how to get to work until after they have left the house. The actual choice is made when they either decide to *walk to work* or *take the bus*, and the selection occurs at the instigation of the commuter when they actually commence on their chosen mode of travel. At this point, the other travel option is abandoned and ceases to be a possible alternative course of action. In *newYAWL* this construct is facilitated by offering all of the tasks subject to the deferred choice to the user(s) responsible for making the choice. Once one of them is selected by a resource, then the work items corresponding to the other tasks are removed from resources' work lists via a cancellation action.

## 5 Conclusions

Workflow technology offers great promise as a general purpose means of automating business processes, however in its current incarnation it is dogged by a

series of criticisms including its narrow view of what constitutes a business process, the lack of formal foundations and its inability to characterise real-world business scenarios. This paper has examined the capabilities of the current generation of workflow technology and proposed a series of development goals for the next generation of workflow tools. As a first step towards these objectives, it has also presented *newYAWL*, a formally defined workflow reference language founded on the workflow patterns, that meets the proposed development goals and provides a yardstick against which the capabilities of future workflow offerings can be assessed. *newYAWL* is currently being used as the design blueprint for the next generation of the YAWL open-source workflow offering.

## References

1. W.M.P. van der Aalst. Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. QUT Technical report, FIT-TR-2003-06, Queensland University of Technology, Brisbane, Australia, 2003.
2. W.M.P. van der Aalst, J. Desel, and E. Kindler. On the semantics of EPCs: A vicious circle. In M. Rump and F.J. Nüttgens, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, 2002. Gesellschaft für Informatik.
3. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet another workflow language. *Information Systems*, 30(4):245–275, 2005.
4. 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.
5. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow mining: A survey of issues and approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
6. G. Alonso. *Process-Aware Information Systems*, chapter Transactional Business Processes, pages 257–278. John Wiley & Sons, 2005.
7. F. Casati, S. Ceri, S. Paraboschi, and G. Pozzi. Specification and implementation of exceptions in workflow management systems. *ACM Transactions on Database Systems*, 24(3):405–451, 1999.
8. G. Decker, O. Kopp, F. Leymann, and M. Weske. BPEL4Chor: Extending BPEL for modeling choreographies. In *Proceedings of the IEEE 2007 International Conference on Web Services (ICWS)*, pages 296–303. IEEE Computer Society, 2007.
9. R.M. Dijkman, M. Dumas, and C. Ouyang. Semantics and analysis of business process models in BPMN. *Accepted for Publication in: Information and Software Technology (IST)*, 2008.
10. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley-Interscience, Hoboken, NJ, USA, 2005.
11. C.A. Ellis. Information control nets: A mathematical model of office information systems. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, pages 225–240, Boulder, CO, USA, 1979. ACM Press.
12. P. Grefen, B. Pernici, and G. Sanchez. *Database support for workflow management: the WIDE project*. Kluwer Academic Publishers, Boston, 1999.
13. P.W.P.J. Grefen and J. Vonk. A taxonomy of transactional workflow support. *International Journal of Cooperative Information Systems*, 15(1):87–118, 2006.

14. C. Hagen and G. Alonso. Exception handling in workflow management systems. *IEEE Transactions on Software Engineering*, 26(10):943–958, 2000.
15. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture and Implementation*. Thomson Computer Press, London, UK, 1996.
16. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts*. Monographs in Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1997.
17. B. Kiepuszewski, A.H.M. ter Hofstede, and W.M.P. van der Aalst. Fundamentals of control flow in workflows. *Acta Informatica*, 39(3):143–209, 2003.
18. N. Mulyar. Pattern-based evaluation of Oracle BPEL. Technical Report BPM-05-24, 2005. [www.BPMcenter.org](http://www.BPMcenter.org).
19. C. Ouyang, H.M.V. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M. ter Hofstede. Formal semantics and analysis of control flow in WS-BPEL. *Sci. Comput. Program.*, 67(2-3):162–198, 2007.
20. M. Reichert, S. Rinderle, and P. Dadam. ADEPT workflow management system. In W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, editors, *Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*, volume 2678 of *Lecture Notes in Computer Science*, pages 370–379. Springer, 2003.
21. N. Russell. *Foundations of Process-Aware Information Systems*. PhD thesis, Faculty of Information Technology, Queensland University of Technology, 2007. [www.yawl-system.com/documents/RussellThesisFinal.pdf](http://www.yawl-system.com/documents/RussellThesisFinal.pdf).
22. N. Russell, A.H.M. ter Hofstede, and W.M.P. van der Aalst. *newYAWL*: Specifying a workflow reference language using Coloured Petri Nets. In *Proceedings of the Eighth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, number DAIMI PB-584, pages 107–126. Department of Computer Science, University of Aarhus, Denmark, 2007.
23. H. Schonenberg, R. Mans, N. Russell, N. Mulyar, and W.M.P. van der Aalst. Process flexibility: A survey of contemporary approaches. In J.L.G. Dietz, A. Albani, and J. Barjis, editors, *CIAO! / EOMAS*, volume 10 of *Lecture Notes in Business Information Processing*, pages 16–30. Springer, 2008.
24. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, 2007.
25. M.T. Wynn, D. Edmond, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Achieving a general, formal and decidable approach to the OR-join in workflow using Reset nets. In G. Ciardo and P. Darondeau, editors, *Proceedings of the 26th International Conference on Application and Theory of Petri nets and Other Models of Concurrency (Petri Nets 2005)*, volume 3536 of *Lecture Notes in Computer Science*, pages 423–443, Miami, USA, 2005. Springer-Verlag.
26. M.D. Zisman. *Representation, Specification and Automation of Office Procedures*. PhD thesis, Wharton School of Business, University of Pennsylvania, PA, USA, 1977.