CSRD (2009) 23: 67–79
DOI 10.1007/s00450-009-0059-7

SPECIAL ISSUE PAPER

Surmounting BPM challenges: the YAWL story

Nick Russell · Arthur H. M. ter Hofstede

Received: 9 June 2008 / Accepted: 13 January 2009 / Published online: 6 March 2009

Abstract The field of Business Process Management (BPM) has evolved considerably over the past decade. Many proposals for business process modelling and/or execution have emerged and some of these have faded into oblivion again. The Workflow Patterns Initiative aimed at achieving a more structured approach to language comparison and development. The patterns that were distilled served as the basis for YAWL (Yet Another Workflow Language). In this paper YAWL is positioned with respect to historical developments in BPM and current challenges in the field.

Keywords Workflow management · Workflow patterns · YAWL

1 Introduction

If we cast our eyes back to the late nineties it is interesting to observe the state of play in the field of workflow. In some ways one could argue that many earlier promises were left largely unfulfilled. In contrast to the golden vision that workflow technology offered as a means of rapidly and rigorously automating repetitive processes thereby streamlining their operation, the majority of offerings proved to lack support for all but the most common process constructs. Moreover they were subject to inconsistent execution behaviours and were generally perceived to impose restrictive work practices on users which ultimately retarded rather than expedited their work throughput.

The XPDL[1] 1.0 standard proposed by the Workflow Management Coalition, was a language which was weak in terms of the concepts it supported (essentially supporting not much more than basic control flow operators, see e.g. [3]) and due to its informal definition left room for substantially varying interpretations, which were semantically incompatible (a topic explored in depth in [35]). Staffware was, according to [20], the workflow management system with the largest market share, but, as is clear in hindsight, it was not a particularly powerful product and had a number of idiosyncracies (e.g. roles can only have one participant, AND-joins are asymmetric [9], and race conditions can cause the OR-join to behave somewhat counterintuitively [9]). In the academic realm, there were a number of efforts to devise workflow languages from first principles, however this work was sometimes undertaken on an isolated basis, and the results tended to be rather specific and proved to have minimal impact on the workflow community at large.

Broadly speaking, it was difficult to define process models that replicated the complexities and variations frequently encountered in practice. Whilst a substantial number of papers (e.g. [4, 18, 19, 27, 30, 31, 33, 34, 37, 53, 56]) proposed solutions for exception handling and adaptive workflow, it was still hard to implement a comprehensive solution and the emphasis was often on transactional aspects which had questionable value in actual business workflows (for a discussion on transactional aspects in BPM, see [17]). In terms of the business process lifecycle, the emphasis remained on specification and enactment. Little attention was paid to monitoring and post-execution analysis. Workflow applications tended to be developed "in house" from scratch and there was little need for interaction with other workflows and/or external applications.

N. Russell (✉)
Eindhoven University of Technology,
P.O. Box 513, 5600MB Eindhoven, The Netherlands
e-mail: n.c.russell@tue.nl

A. H. M. ter Hofstede
Queensland University of Technology,
GPO Box 2434, QLD 4001 Brisbane, Australia
e-mail: a.terhofstede@qut.edu.au

[1] XML Process Definition Language.

In order to gain a more fundamental insight into the concepts required for workflow specification, the Workflow Patterns Initiative[2] was started in 1999. This initiative commenced by taking a closer look at a number of commercially available systems and research prototypes and distilling the various (control-flow) concepts encountered in the form of patterns. One of the key advantages offered by describing these concepts in terms of patterns was that it provided definitions for them that were technology-agnostic and therefore applicable on a more general basis than approaches based on detailed specification of specific process constructs.

As such the patterns provided a foundation for system comparison, language development and also for workflow specification. Over time the patterns were used in official tendering processes and they influenced commercial and open source systems. A number of key questions remained though: Is it possible to provide comprehensive support for the patterns in a single language? Is it possible to keep this language relatively simple? Can such a language be automatically supported? The quest for answers to these questions led to the development of YAWL (an acronym for Yet Another Workflow Language).

YAWL took Petri nets as a starting point, as it had been argued that Petri nets were particularly well suited to workflow specification [1] and indeed, many of the workflow patterns can be directly expressed in terms of Petri nets [6]. However, some of the patterns are more difficult to directly realise in this way and these led to specific language constructs in YAWL. The original formal semantics of YAWL were specified as a transition system, but more recently, the CPN formalism [32] was used to provide an operational semantics for *new*YAWL, an extension of YAWL providing comprehensive support for the latest collections of control-flow patterns, data and resource patterns [46].

In the following parts of this article we first provide a brief overview of the main ingredients of YAWL in terms of its design rationale, language features and operational environment in Sects. 2–4 respectively. Following on from this in Sect. 5 we examine the state-of-the-art in the field of workflow management, and more broadly that of business process management, in order to see where the challenges lie and what role YAWL can play in their future resolution. The article concludes with a brief epilogue in Sect. 6.

2 Design criteria

In this section we briefly examine the design criteria that underpin the YAWL formalism. These criteria have been adopted throughout the development process for YAWL and aim to ensure that it provides a mechanism for supporting business processes that demonstrates broad applicability and also that it extends the current state of the art.

Suitability The workflow patterns provide a source book of *desirable* process constructs and, by definition, their usefulness is underscored by the fact that each of them have been observed in an actual tool and/or language. Although support for specific workflow patterns may not be required in some situations, generally speaking, they all represent meaningful functionality. Hence, the more patterns a language supports, the more *suitable* it is for general workflow specification.

Formality Workflows can be quite complex due to e.g. parallelism, complex data and resourcing strategies. In order to guarantee that workflow execution is not arbitrary but can be reasoned about at design time, a workflow language should have a *formal foundation*. This foundation then not only enables design time verification, but can also act as the blueprint for the runtime environment defining the manner in which language constructs should actually be enacted. An example of a formally defined language are the Workflow nets presented in [5].

Comprehensibility In order to cater for domain experts, who are not necessarily knowledgeable in IT or BPM, as far as possible it should be possible to understand workflow specifications on an intuitive basis. Graphical representations in particular can assist in making workflows more *comprehensible*, although there are other mechanisms that can also be helpful in this regard, e.g. the ability to structure the workflow into various components by means of a decomposition mechanism or, a more recent development, by providing support for aspect orientation [21].

Enactability As workflows ultimately need to be enacted in some form of operational environment, they need to be *executable*. At the same time, they should not be tied to a specific execution environment or vendor and thus they should be *technology independent*. This gives the greatest degree of freedom to business analysts and liberates them from unnecessary constraints that may be imposed by certain platforms or even by certain (proposed) standards which are too closely aligned with individual vendor interests.

Minimality The workflow language should be as *minimal* as possible. For example, not every pattern should lead to yet another language construct. This approach makes the language easier to explain and understand, and facilitates its implementation. In this regard it is interesting to observe that Petri nets, whilst they constitute a complete formal theory for modelling dynamic systems, have a minimal set of constructs. As a consequence, determining their wellformedness is straightforward, and their semantics are easily expressed and explained, particularly compared to some contemporary proposals for business process modelling notations.

[2] http://www.workflowpatterns.com.

3 Core YAWL

In [1] the benefits of using Petri nets for workflow specification were advocated. By way of validation for this proposal, in [6] the CPN formalism was subjected to a patterns-based analysis. This evaluation confirmed the argument that Petri nets demonstrate some significant strengths when used for workflow specification, however it also identified some areas of weakness. In particular, Petri nets have difficulties with expressing

- patterns that deal with situations where a number of instances of the same task execute in parallel;
- the concept of an OR-join, a construct which captures the notion of "wait only if you need to";
- cancellation where the execution of an activity, a whole case, or anything in between, needs to terminated.

Workflow nets or WF-nets for short (see e.g. [5]) form a subclass of Petri nets defined for the purposes of workflow specification. Each net has a unique start place and a unique end place and all tasks are on a path from the start place to the end place. Notational abbreviations exist for the various splits and joins which are encountered when modelling processes thereby allowing for more compact expression of workflows. The design of YAWL [7] took Workflow nets as a starting point and added constructs for the OR-join (in fact also for the OR-split), multiple instances, and cancellation. In addition, tasks can be directly connected if they only have one place in between them (which has no other inputs and outputs).

In Fig. 1 an overview is provided of the various notational elements of YAWL. A task can have join and split behaviour, can have a decomposition, and can be of type multiple instances (thereby indicating that multiple concurrent instances of it are allowed). A task may have an associated cancellation region from where, at runtime, tokens are removed whenever any of the instances of the task completes execution. The term condition is used to capture the

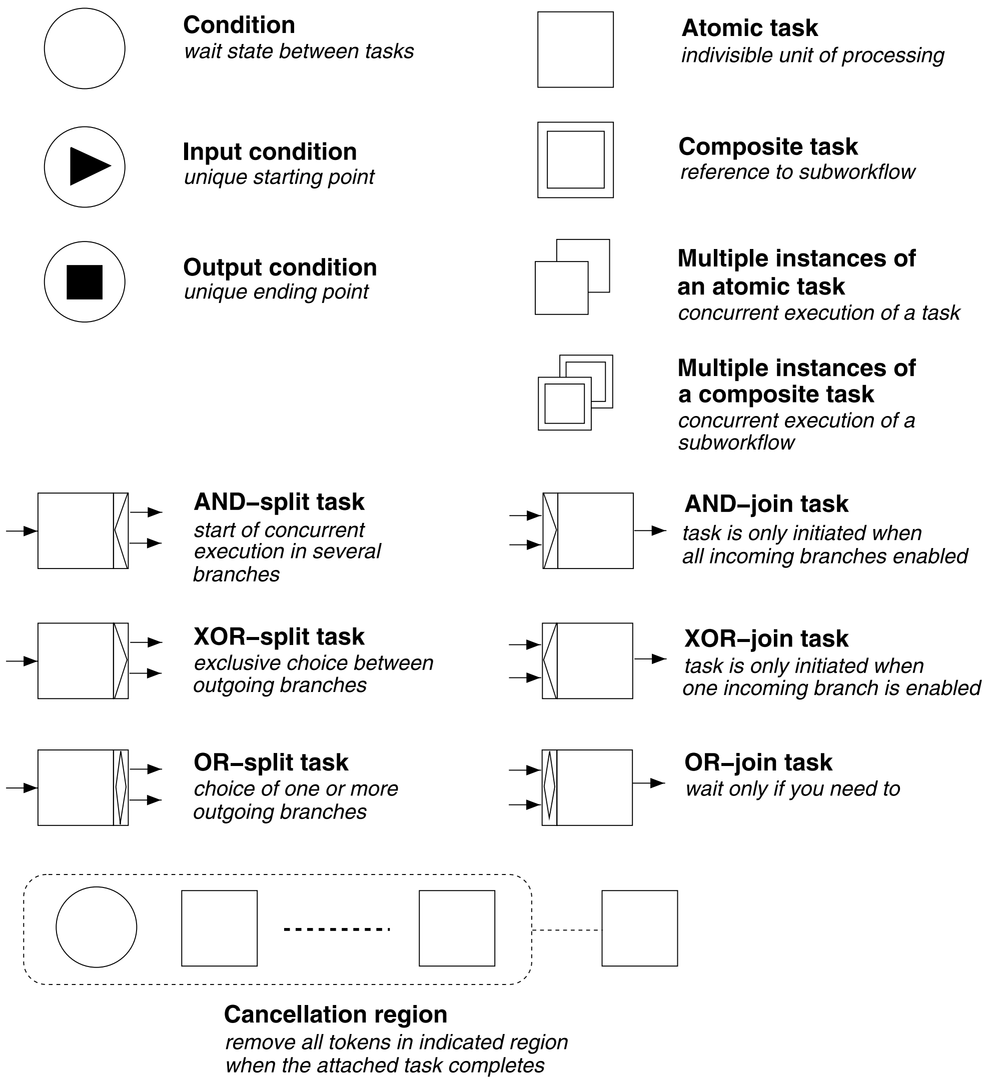
Condition
wait state between tasks
Atomic task
indivisible unit of processing
Input condition
unique starting point
Composite task
reference to subworkflow
Output condition
unique ending point
Multiple instances of an atomic task
concurrent execution of a task
Multiple instances of a composite task
concurrent execution of a subworkflow
AND–split task
start of concurrent execution in several branches
AND–join task
task is only initiated when all incoming branches enabled
XOR–split task
exclusive choice between outgoing branches
XOR–join task
task is only initiated when one incoming branch is enabled
OR–split task
choice of one or more outgoing branches
OR–join task
wait only if you need to
Cancellation region
remove all tokens in indicated region when the attached task completes

Fig. 1 Standard YAWL symbology

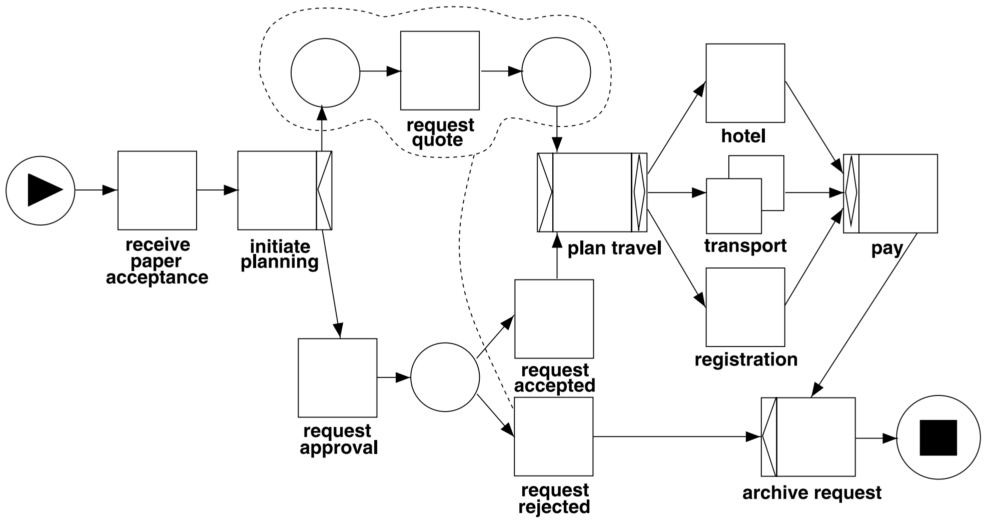

Fig. 2 A simple YAWL net for conference travel

notion of the place as it exists in WF-nets and Petri nets. It is interesting to observe that despite the limited number of notational elements that YAWL comprises, it demonstrates comprehensive support for the workflow control-flow patterns.

The OR-join has been the subject of a number of papers (e.g. [36, 39]) and in one incarnation or another it occurs in a number of systems and languages. The original definition of the OR-join for YAWL in [7] was superceded by the definition provided in [57]. The latter approach treats decomposed tasks on the path to an OR-join as if they are atomic and treats other OR-joins on the path to the OR-join as if they are XOR-joins. An algorithmic approach to computing whether an OR-join is enabled was derived through a mapping of YAWL (without OR-joins) to Reset-nets (see e.g. [25]) and exploiting an existing algorithm to computing coverability. An OR-join can fire if and only if one of its input places contains a token and it is not possible to reach a marking where an as-yet not marked input place also receives a token and all input places that were already marked remain marked.

In Fig. 2, a YAWL net is shown that captures a simplified version of how conference travel could be planned at a university. The case starts with the receipt of a paper acceptance. After some initial planning a quote is requested and approval is sought. The request for approval is captured through a deferred choice as the choice is made by the head of school. If the approval is withheld, the request for a quote is cancelled, paperwork is filed accordingly, and the case is closed. Note that we need to explicitly represent the conditions before and after the task *request quote* as this task may not have started yet or may have already finished. If approval is granted, then, if the quote has been received as well, the trip can be planned. For simplicity, we assume that this planning may be followed by the booking of an hotel, registration for the conference, and arrangement of transport for one or more destinations (hence the use of a multiple instance task). When all these reservations have been completed, payment is made and the paperwork related to the case can be archived.

As a consequence of YAWL's formal foundation, it is possible to verify workflows before their deployment. The well-known soundness property (see e.g. [2]) captures certain desirable properties of workflows, e.g. whether they can always terminate, whether they always terminate properly (i.e. don't leave tokens behind) and whether all tasks can be reached in some scenario. Over time, many variants of this correctness notion have been developed. An overview of correctness properties for workflows and their decidability for various classes of workflow languages can be found in [13]. For YAWL there are two verification approaches, neither of which is complete, i.e. not all errors may be detected, due to the high expressive power of YAWL [13]. One approach, which is documented in [54], is based on the notion of relaxed soundness [23, 24] and uses T-invariants. Another approach, which is documented in [58], uses, among others, the notion of weak soundness for YAWL nets, which is decidable for YAWL nets without OR joins, and exploits the mapping of (a subset of) YAWL to Reset nets. The notion of weak soundness is less restrictive than the notion of soundness as instead of requiring that all reachable markings can mark the final condition, it is required that the final condition can be marked when starting from the start condition. Hence a model can be weak sound, but still contain a deadlock. For YAWL models with one or more OR-joins a state space analysis can be performed for models with a finite state space or the OR-joins can be changed into XOR-joins (but this has some real limitations). The approach described in [58] also considers correctness issues involving OR-joins which can be replaced by an XOR-join or an AND-join, and additional elements in cancellation regions.

YAWL's exception handling capabilities are based on the framework presented in [49]. In this framework a number of exception types are identified and the concept of an exception pattern is defined which consists of three components: (1) what should be done with the work item involved (e.g. forcibly fail it), (2) what should be done with other related work items (e.g. cancel the current case), and (3) what corrective action should be taken (e.g. execution of a compensatory workflow). A modern workflow environment should be able to deal with exceptions ([26, 31]) that were not anticipated at design time, and the framework and the related language proposal described in [49] served as the starting point for the concept of *exlets* in YAWL. Figure 3 illustrates two exlets that define exception handling behaviours for tasks in the conference travel process illustrated in Fig. 2. The first of them handles the situation where a deadline associated with the *request approval* task is exceeded. Should this exception arise, the work item associated with the *request approval* task is suspended, reassigned to another resource (probably via some form of escalation), rewound back to the beginning and then allowed to continue executing (in effect being restarted). The second exlet demonstrates how a task failure of a *request quote* task should be dealt with. In this situation, the work item is forcibly stopped (in case it has not already done so) and then a new instance of it is commenced. A comprehensive discussion of the operation and implementation of exlets in YAWL can be found in [16].

Business processes are often not static in nature, but evolve over time [42]. This may be due to a number of factors, e.g. new legislation, changes in the market or learning experiences. It is important that when necessary changes are identified, they can be directly supported by the workflow system and offline resolution is not required [14]. In YAWL the concept of *worklets* is used to deal with evolving workflows. Over time selected tasks may have an associated repertoire of workflows and the runtime selection of the "right" workflow is guided by Ripple–Down Rules [22]. If a suitable workflow cannot be identified, a new one can be created. The analyst then defines what the essential differences are between the context in which this situation was encountered and the current context for the rule that was selected, but was judged to be not applicable. This then leads to an extension of the Ripple–Down Rules. Figure 4 illustrates the form that a Ripple–Down Rule may take. In this situation, it is associated with the *pay* task. Two distinct variations of this task can be identified: in general for conference delegates, the implementation of the task involves the sequence of subtasks: *record details*, *receive payment* and *confirm registration*. However for invited speakers, no payment is required and the *receive payment* task is omitted from the sequence. Should additional classes of conference

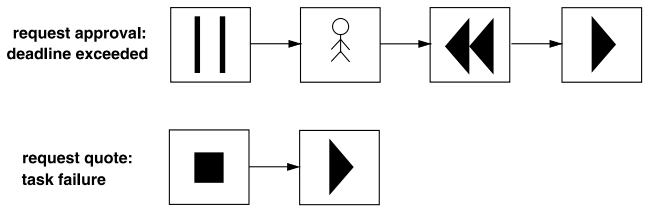

Fig. 3 Conference travel exlets for the *request approval* and *request quote* tasks

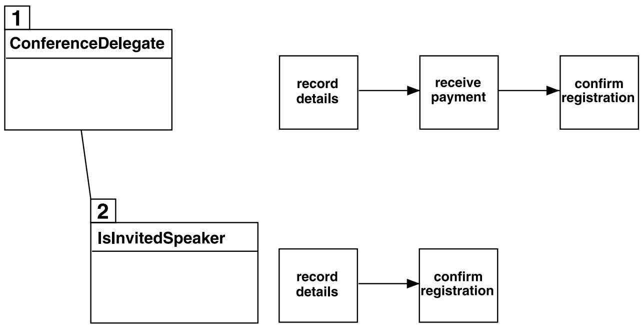

Fig. 4 Conference travel worklets for the *pay* task

attendees be identified that need to be handled in a distinct way, then the rule hierarchy for the task can be further extended on a dynamic basis to cater for them. A detailed description of the worklet approach in YAWL can be found in [14, 15].

Recently, the original set of control-flow patterns was reviewed. This led to adaptations of existing patterns and the introduction of a number of new patterns [50]. Patterns for the data perspective [48] and the resource perspective [47] have also been introduced over time. These pattern collections serve as the basis for *new*YAWL [46], a workflow reference language providing comprehensive support for the control-flow, data and resource perspectives. *new*YAWL is formalised using Coloured Petri nets (CPN [32]) and consequently the semantic model for *new*YAWL can be directly executed in the CPN Tools environment, thus providing the ability to analyse the complex relationships that may exist between concepts within and between workflow perspectives.

4 The YAWL environment

The YAWL system[3] provides a reference implementation of the YAWL language. Originally developed to validate that the workflow patterns provided a suitable basis for defining the desirable properties of a workflow system, the YAWL system has subsequently evolved into a high profile work-

[3] Available from http://www.yawl-system.com.

flow initiative that has served to guide future research efforts in the workflow domain. In this section, we briefly examine the development principles that have evolved for the system during this time, introduce the operational architecture of the system and discuss early deployments of the YAWL offering in an operational context.

4.1 Development principles

The YAWL system has been under development for over five years. During this time its capabilities have expanded markedly and a set of associated development principles have also evolved that now guide system engineering efforts. These include:

Best practice The YAWL system is intended to demonstrate best practice in the field of workflow. For this reason, its fundamental language capabilities are tightly coupled to the YAWL language and the workflow patterns initiatives;
Open source The YAWL system is developed under an open-source licence model. It is freely available available for download and use in an effort to encourage widespread adoption of the software. Moreover, such an approach provides valuable feedback and increases the opportunities for potential collaboration;
Extensible The YAWL environment is developed using a service-oriented architecture. There are multiple opportunities for users to integrate their own (or third party) software within a YAWL deployment hence providing a wide variety of options to extend its capabilities and adapt it to their specific needs;
Modular The YAWL system deliberately divides its operational environment into a series of distinct software modules along functional lines. This provide significant flexibility when deploying a YAWL instance. The specific breakdown of these modules is described in Sect. 4.2;
Easy to use A significant focus of the YAWL initiative has been to simplify the overall process of developing an automated business process. It is possible to both design and deploy simple processes without requiring any programming on the part of the user. Even in more complex situations, the extent of custom development is minimised. Some situations still require relatively complex programmming (e.g. data passing, custom forms and external interfaces) however it is hoped that even in these areas, the complexity of development can be simplified over time;
Easy to deploy The YAWL system is based on widely available technology and is packaged in a manner that facilitates rapid deployment without requiring extended installation and configuration activities before it can be utilised; and
Resilient The ultimate goal for the YAWL system is to be production strength such that it is capable of supporting industrial scale process deployments.

4.2 Operational overview

In Fig. 5 (taken from [12]) an overview of the YAWL architecture is provided. As it illustrates, the YAWL environment consists of a number of components that communicate through defined interfaces. YAWL specifications are developed in the YAWL Process Editor. These specifications can be validated, exported in XML format and subsequently be loaded into the YAWL Engine through interface A. The YAWL engine interprets these specifications and during execution it can communicate with other components about work progression through interface B, e.g. the engine can let components know that certain work items are enabled and other components can let the engine know about work item completion. Similarly, through this interface, cases can be started and the engine can inform the outside world about their completion. Although the vision of the workflow engine being completely resource agnostic has not yet been fully achieved (the engine log events still contain a designated resource field), dealing with resources is the domain of the Resource Service. This service can interact with organizational models in a generic manner through interface O (i.e. these models can be defined in a relational database or in an LDAP directory). Resourcing requirements are defined through the Process Editor and the Resource Service can obtain these through interface R. The Resource Service also maintains the work lists for resources. The Worklet Service provides support for dynamic workflow and for exceptions and it keeps its own event log and information about worklets as they are created over time. Interface X of this service was provided for the purpose of communication with the engine about exceptions. There are other services (e.g. the Declare service that supports declarative specification and execution of workflows, see [8]) that are available for the YAWL environment. As can be seen, the YAWL architecture provides a relatively open and extensible environment and one can replace existing components with custom-made components or add completely new components.

4.3 Early adoption

Over the past five years, the number of software downloads for the system has exceeded the 50 000 mark. Several of these downloads have progressed to fully operational status and it is interesting to examine three such deployments in more detail.

YAWL4Film [41] is an initiative undertaken by the Australian Film, Television and Radio School in conjunction

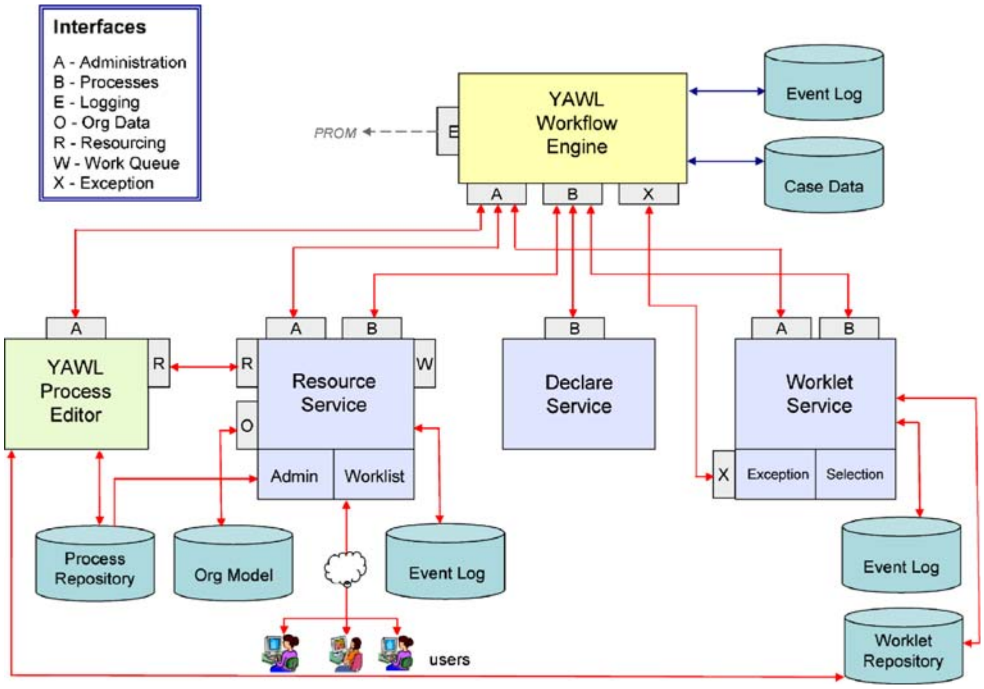

Fig. 5 YAWL system components [12]

with QUT. This work centred on the development of an application to support entry and management of production data associated with film shoots. A critical part of this project was the establishment of role-specific data entry points that allowed for the capture of key production data during a film shoot. The data collected was then used to create multiple reports automatically for distribution to other production crew menbers. The entire process was supported by YAWL. This system has already been utilised in two student film projects on a pilot basis, and also in a full commercial featured film (produced by Porchlight Films) in 2008.

The Traditional Medicine Hospital in Zhejiang Province, China adopted the YAWL system as part of their Brain-Bridge Project which started in May 2007. One of the initial objectives of this work was to build a workflow-aware radiology information system (RIS). In this project, the system was based on YAWL and several key processes including the radiology examination process were developed as custom YAWL services. The resultant software is now in a pilot phase and is currently under evaluation by hospital staff.

The companies first:utility and first:telecom in the UK (both part of the Impello plc group), which provide energy and telecommunications services respectively, have built software based on the YAWL system that provides a novel approach to page navigation for web based systems together with the more traditional use of workflow for choreographing long-lived business processes.[4]

[4] http://www.yawlfoundation.org/about/adoption.html.

5 Reflections on the state of the art

The development of the YAWL language and system have given an important insight into the needs of the increasingly broad range of BPM users and the complexities of developing a solution that addresses these needs. Whilst the BPM field has matured during this time and is now recognised as an important domain in its own right, there are still a wide range of issues that need further consideration. In this section, we will examine some of them in further detail.

5.1 There is no common vocabulary for BPM concepts

One of the major aims of the Workflow Management Coalition was to establish a common vocabulary for the workflow domain as a precursor to a conceptual workflow reference model that facilitated the interchange of process models between distinct systems. The aim of this model was to provide a common foundation that allowed collaboration between different systems regardless of the technology on which their individual process models were based. To some degree the effort was successful in establishing a basic vocabulary for common workflow terms (e.g. task, case etc.) but attempts at establishing a common reference model were largely ignored by workflow developers who did not want to tie their proprietary modelling formalisms to a common technique. In effect the workflow reference model became the lowest common denominator for workflow technology categorising common concepts encompassed by every system rather than providing a comprehensive basis for comparison against which systems could be assessed. The problem remains however and as the BPM field has matured, there are now an increasing number of terms and concepts that do not have a specific, broadly agreed definition.

5.2 There is significant disparity between modelling and enactment tools

In recent years, a wide range of business process modelling formalisms have been proposed and several of these (cf. EPCs,[5] BPMN,[6] UML[7] activity diagrams) have come to the fore as widely used approaches to capturing business processes. These formalisms however are designed to operate at a conceptual level and deliberately abstract from the details associated with process enactment. Indeed there are no specific guidelines for any of these notations that describe precisely how a candidate process should actually be enacted. The most recent revision of XPDL (v2.0) attempts to provide a serialised definition for BPMN processes but as with the modelling notation, its actual execution semantics are only informally defined and are potentially subject to alternative interpretations. Similarly, the BPEL[8] language, which has garnered significant interest, does not have a formally specified execution model and the much lauded transformation of conceptual BPMN process models to executable BPEL processes in fact proves to be much more complex than was originally anticipated [40]. Clearly in order to close this gap, there needs to be an increased focus on developing means of operationalising business process definitions. This does not mean ascribing an execution semantics to common modelling approaches but rather focussing on the establishment of objective means of transforming between modelling and enactment notations for business processes. YAWL illustrates one possible approach to resolving this problem. It provides a language that can operate both at a conceptual and execution level. Whilst the language elements can be used simply for modelling business processes, there is sufficient detail captured in a YAWL model to enable it to be directly executed.

5.3 The focus must shift from comprehensive modelling notations to enactment predictability

To date in the BPM field the focus has been on modelling richness without regard to the manner in which the conceptual constructs underpinning these models will ultimately be realised in practice. Not only are there ambiguities in the way in which various constructs are operationalised, but in many cases, it is not clear exactly how some constructs can be facilitated at runtime. To further complicate matters, in many formalisms, it is possible for the same fundamental concept to be represented in several distinct ways. As a consequence of these issues, it is possible for the same business process to be subject to varying execution outcomes in distinct operational environments. The remedy to these difficulties lies in providing precise execution semantics for business processes, typically based on a formal foundation that provides a deterministic means of describing the state transformations that occur when a process is enacted. In YAWL, each of the language elements have a precise formal semantics and as a result, the execution behaviour of a YAWL model is precisely and unambiguously defined.

5.4 BPM is more than just task coordination or workflow

Until recently process modelling techniques have tended to focus on describing the control-flow perspective of business processes and have ascribed significantly less importance to other fundamental aspects, e.g. data representation and management, resource definition, work distribution, exception handling and so on. This bias has also been reflected in workflow technology, which traditionally been viewed as the natural implementation vehicle for business processes, and many offerings conceptualise a business process in terms of its control-flow representation. Yet, it is increasingly recognised that in order to be effective, a business process needs to be multi-perspective in form and capture details of a number of distinct process viewpoints. As a minimum, the data, resource and exception handling perspectives need to be considered in addition to control-flow aspects in order to capture a holistic view of a business process. This shortcoming was recognised early in the YAWL development process and a number of data and resource-related extensions were applied to the system. Most recently a comprehensive language extension, termed *new*YAWL, has been proposed that radically extends the extent of support that YAWL provides for the broad range of control-flow, data and resource patterns.

5.5 Business processes are inherently flexible in nature

Perhaps the major criticism that workflow technology has faced in recent years has related to its inflexibilty and the difficulty in reconciling relatively static business process definitions with the realities of the modern business environment and the need to dynamically adapt to changing circumstances. It is now widely recognised that process flexibility or agility is a key requirement for effective business operation and consequently, this is a property that needs to be supported in enabling technologies for business processes. However the notion of process flexibility is subject to a wide variety of interpretations. In [52], four distinct operational

[5] Event-driven Process Chains.
[6] Business Process Modeling Notation.
[7] Unified Modeling Language.
[8] Business Process Execution Language.

approaches are described to facilitating process flexibility in the form of a taxonomy. These include (1) *flexibility by design* where alternative execution options are explicitly included in a business process at design-time, (2) *flexibility by deviation* where a workflow offering provides facilities for individual cases to temporarily deviate from the strict execution sequence prescribed by the associated process model, (3) *flexibility by underspecification* where there is the ability to leave sections of a process model undefined and facilities are available in a workflow offering to specify the structure of these missing portions immediately prior to their execution at runtime and (4) *flexibility by change* where a workflow offering provides the ability to change the form of a process model during execution for one, several or all process instances. The fundamental aspects of each of these approaches are relatively well understood.[9] The workflow control-flow patterns [9, 28] characterise the potential range of flexibility by design constructs, the pockets of flexibility proposal [59] describes a mechanism for supporting flexibility by underspecification, and the change patterns and change support features in [55] comprehensively describe approaches to facilitating flexibility by underspecification and change.

Broad support for flexibility by design in contemporary workflow offerings is relatively commonplace, however other forms of flexibility support are less widely implemented. Moreover where an offering does provide other approaches to facilitating flexible process execution, it tends to focus on one specific area, e.g. the case handling tool FLOWer [10] provides a number of deviation facilities and the adaptive workflow system ADEPT [42, 43] is particularly strong in terms of its support for dynamic process change. By virtue of its extensible service-oriented architecture, YAWL offers a variety of forms of flexible process support. Its *worklet* and *exlet* facilities provide mechanisms for dealing with changes in the operating environment and unexpected exceptions that arise during execution. Moreover the augmentation of YAWL with the Declare service as part of the *flexibility as a service* initiative [12] demonstrates how it is possible to support each of the forms of process flexibility identified in the taxonomy in a common operating environment.

5.6 Business process enactment is increasingly complex

One of the fundamental recognitions in recent years has been that most significant business processes operate on a "whole-of-organisation" basis rather than in departmental terms. This means that business processes involve the coordination of numerous individuals throughout an organisation in working towards specific organisational objectives. Often business processes are so broad in form that they do not fall under the auspices of any one individual but instead have specific sections which are managed by distinct resources. Furthermore, it is possible that all of the expertise required to undertake a given process does not lie within the organisation and that the business process also needs to integrate external resources and services in order to achieve corporate goals. As such the automated business process serves as a means of distributing work across a suitably qualified range of resources and coordinating their efforts. In order to achieve this aim, the enabling technology must be capable of integrating a wide variety of systems and resources operating in distinct locations and be based on technologies of varying sophistication. Moreover it should allow individual sections of the overall business process to be substituted with alternative execution mechanisms as better facilities are identified over the life of the process. Recent development in the web services areas, particularly the emergence of the service oriented architecture, offer a wealth of opportunities for dealing with these issues. The YAWL system is based on a distributed architecture that ensures that the design-time environment, workflow engine, resource manager and worklist handlers that make up the operational environment function independently of each other and interact via defined interfaces. Moreover there are a series of additional interfaces that allow custom services to be developed for YAWL processes thus catering for future operational enhancement and integration with external systems and services.

5.7 User empowerment offers a range of opportunities for improving process performance

Whilst there has been significant focus on the optimisation of business processes from a global standpoint, in many cases, it is the individual users responsible for completing specific activities within a process that are best positioned to identify opportunities for improvement. With this in mind, it is potentially beneficial to furnish users with better intelligence regarding the impact of their individual work activities on the broader organisational process. This intelligence may take a number of distinct forms. One approach is based on the use of *process mining* which aims to provide users with decision making information based on analysis of preceding executions of the same process. A multitude of different analysis techniques are possible however the most important consideration when using these approaches is ensuring that the necessary information is presented to the user in a meaningful way at the time that they require decision support. This approach can be especially powerful when linked with richer work list visualisation facilities which provide workflow users with a much broader range

[9] A comprehensive overview of the literature in the area of process flexibility can be found in [51].

of information about outstanding work items when making a decision as to which work item to execute next or which work items need prioritising. To this end, research is currently being conducted with YAWL which focuses on integrating one of the most comprehensive process mining offerings currently available (the ProM [11] suite of tools) in order to provide better decision support for workflow users. There is also research [38] investigating various approaches for improving work list visualisation.

5.8 Simulation offers opportunities for validating likely process performance

Typically one of the motivations for using workflow technology is that it offers the opportunity to automate complex business processes with the expectation that they will deliver more effective and efficient process performance. However it can be difficult to establish at design-time whether a proposed process model will actually function in accordance with expectations. Indeed, it is often only at runtime that the operational dynamics of a process become clear. Simulation offers one opportunity for providing earlier validation of likely process performance in a given application context. By providing the ability to automatically execute large numbers of process instances for a given process model in an environment that utilises realistic data values and distributes work amongst a similar set of resources to those encountered in the target operating environment, vital insights can be gained as to the probable performance of a given process model and the opportunities that might exist for further improving its performance. With these opportunities in mind, research [45] is currently being conducted to provide simulation capabilities for the YAWL system that allow accurate prediction of potential short-term behaviours based on analysis of the current state and historical information analysed using the ProM offering.

5.9 Configurable reference models offer a means of disseminating best practice knowledge

The use of reference models has long been advocated as a means of distributing knowledge in regard to optimal process designs for a given domain. Whilst the rationale for the concept is sound, in practice however it is often found that the specific context in which a reference model can be applied is extremely limited. As a remedy to this difficulty, the notion of configurable reference models was proposed [44] where the various configuration alternatives which might exist for a given reference model are specifically encoded within the model. In doing so, the generality of a given reference model is radically increased by allowing its use to be tailored to a variety of specific scenarios. The notion of configurable reference models has been adapted to the YAWL environment and a set of configurable extensions have been proposed for the YAWL language [29].

6 Epilogue

In this paper we have looked at YAWL's roots in an historical context, promulgated its raison d'être, briefly outlined its main ingredients and system support facilities, and examined the state of play in BPM, specifically identifying how YAWL addresses some of the challenges this field currently faces.

BPM has evolved considerably over the last decade and there is increasing recognition that modelling languages should be more expressive and provide comprehensive support for the control-flow, data, resource and exception handling perspectives, that they should be formally defined, and that process automation environments should support evolving business processes, empower users in their decision making and offer sophisticated runtime as well as post-execution diagnostics. It is hoped that YAWL can play a guiding role in the future development of the BPM field.

Acknowledgement The authors would like to thank all members of the YAWL Foundation for their assistance in the development of YAWL. Particular thanks go to Wil van der Aalst as co-developer of the YAWL language and his substantial contributions to both the language and the support environment over the years.

This research is 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).

References

1. van der Aalst W (1996) Three good reasons for using a Petri-net-based workflow management system. In: Navathe S, Wakayama T (eds) Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC-96), Cambridge, MA, pp 179–201
2. van der Aalst W (2000) Workflow Verification: Finding Control-Flow Errors using Petri Net-Based Techniques. In: van der Aalst W, Desel J, Oberweis A (eds) Proceedings of Business Process Management: Models, Techniques and Empirical Studies. Springer-Verlag, Lecture Notes in Computer Science, vol 1806, pp 161–183
3. van der Aalst W (2003) Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language. QUT Technical report, FIT-TR-2003-06, Queensland University of Technology, Brisbane
4. van der Aalst W, Basten T (2002) Inheritance of Workflows: An Approach to Tackling Problems Related to Change. Theor Comput Sci 270(1-2):125–203

5. van der Aalst W, van Hee K (2002) Workflow Management: Models, Methods and Systems. MIT Press, Cambridge, MA
6. van der Aalst W, ter Hofstede A (2002) Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages. In: Jensen K (ed) Proceedings of the Fourth International Workshop on Practical Use of Coloured Petri Nets and the CPN Tools, University of Aarhus, Aarhus, Denmark, vol 560 of DAIMI, pp 1–20
7. van der Aalst W, ter Hofstede A (2005) YAWL: Yet another workflow language. Inf Syst 30(4):245–275
8. van der Aalst W, Pesic M (2006) DecSerFlow: towards a truly declarative service flow language. In: Bravetti M, Nunez M, Zavattaro G (eds) International Conference on Web Services and Formal Methods (WS-FM 2006), Springer, vol 4184, pp 1–23
9. van der Aalst W, ter Hofstede A, Kiepuszewski B, Barros A (2003) Workflow patterns. Distrib Parall Databases 14(3):5–51
10. van der Aalst W, Weske M, Grünbauer D (2005) Case handling: A new paradigm for business process support. Data Knowl Eng 53(2):129–162
11. van der Aalst W, van Dongen B, Günther C, Mans R, Alves de Medeiros A, Rozinat A, Rubin V, Song M, Verbeek H, Weijters A (2007) ProM 4.0: Comprehensive Support for *real* Process Analysis. In: Kleijn J, Yakovlev A (eds) Petri Nets and Other Models of Concurrency – ICATPN 2007, 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007, Siedlce, Poland, 25–29 June 2007, Proceedings, Springer, Lecture Notes in Computer Science, vol 4546, pp 484–494
12. van der Aalst W, Adams M, ter Hofstede A, Pesic M, Schonenberg M (2008a) Flexibility as a Service. Tech. Rep. BPM-08-09, http://www.BPMcenter.org
13. van der Aalst W, van Hee K, ter Hofstede A, Sidorova N, Verbeek H, Voorhoeve M, Wynn M (2008b) Soundess of workflow nets: Clasification, decidability, and analysis. Tech. Rep. BPM-08-02, http://www.BPMcenter.org
14. Adams M (2007) Facilitating Dynamic Flexibility and Exception Handling for Workflows. PhD Thesis, Queensland University of Technology, Brisbane, Australia, available through http://www.yawl-system.com
15. Adams M, ter Hofstede A, Edmond D, van der Aalst W (2006) Worklets: A service-oriented implementation of dynamic flexibility in workflows. In: Meersman R, Tari Z (eds) On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006, Montpellier, France, 29 October–3 November 2006. Proceedings, Part I., Springer, Lecture Notes in Computer Science, vol 4275, pp 291–308
16. Adams M, ter Hofstede A, van der Aalst W, Edmond D (2007) Dynamic, Extensible and Context-Aware Exception Handling for Workflows. In: Meersman R, Tari Z (eds) On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, 25–30 November 2007, Proceedings, Part I, Springer, Lecture Notes in Computer Science, vol 4803, pp 95–112
17. Alonso G (2005) Transactional Business Processes. In: Dumas M, van der Aalst W, ter Hofstede A (eds) Process-Aware Information Systems: Bridging People and Software through Process Technology, Wiley-Interscience, New York, pp 257–278
18. Casati F, Ceri S, Pernici B, Pozzi G (1996) Workflow evolution. In: Thalheim B (ed) Conceptual Modeling – ER'96, 15th International Conference on Conceptual Modeling, Lecture Notes in Computer Science, vol 1157, Springer, Cottbus, Germany, pp 438–455
19. Casati F, Ceri S, Paraboschi S, Pozzi G (1999) Specification and implementation of exceptions in workflow management systems. ACM Trans Database Syst 24(3):405–451
20. Casonato R (1998) Gartner group research note 00057684, production-class workflow: A view of the market, http://www.gartner.com
21. Charfi A, Mezini M (2006) Aspect-oriented workflow languages. In: Meersman R, Tari Z (eds) On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006, Montpellier, France, 29 October–3 November 2006. Proceedings, Part I, Springer, Lecture Notes in Computer Science, vol 4275, pp 183–200
22. Compton P, Jansen B (1988) Knowledge in context: A strategy for expert system maintenance. In: Siekmann J (ed) Proceedings of the 2nd Australian Joint Artificial Intelligence Conference, Springer-Verlag, Adelaide, Australia, Lecture Notes in Artificial Intelligence, vol 406, pp 292–306
23. Dehnert J, van der Aalst W (2004) Bridging the Gap Between Business Models and Workflow Specifications. Int J Cooperat Inform Syst 13(3):289–332
24. Dehnert J, Rittgen P (2001) Relaxed Soundness of Business Processes. In: Dittrich K, Geppert A, Norrie M (eds) Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), Lecture Notes in Computer Science, vol 2068, pp 157–170
25. Dufourd C, Finkel A, Schnoebelen P (1998) Reset Nets Between Decidability and Undecidability. In: Larsen K, Skyum S, Winskel G (eds) Proceedings of the 25th International Colloquium on Automata, Languages and Programming, Springer-Verlag, Aalborg, Denmark, Lecture Notes in Computer Science, vol 1443, pp 103–115
26. Eder J, Liebhart W (1996) Workflow recovery. In: Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), IEEE Computer Society, Brussels, Belgium, pp 124–134
27. Ellis C, Keddara K, Rozenberg G (1995) Dynamic change within workflow systems. In: COCS '95: Proceedings of conference on Organizational computing systems, ACM, New York, NY, pp 10–21
28. Farrell A, Sergot M, Bartolini C (2006) Formalising workflow: A CCS-inspired characterisation of the YAWL workflow patterns. Group Decis Negot 61(3):213–254
29. Gottschalk F, van der Aalst W, Jansen-Vullers M, La Rosa M (2008) Configurable workflow models. Int J Cooperat Inform Syst 17(2):177–221
30. Grefen P, Vonk J (2006) A taxonomy of transactional workflow support. Int J Cooperat Inform Syst 15(1):87–118
31. Hagen C, Alonso G (2000) Exception handling in workflow management systems. IEEE Trans Softw Eng 26(10):943–958
32. Jensen K (1997) Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, vol 1, Basic Concepts. Monographs in Theoretical Computer Science, Springer-Verlag, Berlin
33. Joeris G (1999) Defining flexible workflow execution behaviors. In: Dadam P, Reichert M (eds) Workshop Informatik '99: Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, CEUR-WS.org, Paderborn, Germany, CEUR Workshop Proceedings, vol 24, pp 49–55
34. Kammer P, Bolcer G, Taylor R, Hitomi A, Bergman M (2000) Techniques for supporting dynamic and adaptive workflow. Computer Supported Cooperative Work 9(3/4):269–292, http://citeseer.ist.psu.edu/kammer98techniques.html
35. Kiepuszewski B, ter Hofstede A, van der Aalst W (2003) Fundamentals of control flow in workflows. Acta Inform 39(3):143–209
36. Kindler E (2006) On the semantics of EPCs: Resolving the vicious circle. Data Knowl Eng 56(1):23–40
37. Klein M, Dellarocas C, Bernstein A (eds) (2000) Adaptive Workflow Systems, Special issue of the journal of Computer Supported Cooperative Work, vol 9

38. de Leoni M, van der Aalst W, ter Hofstede A (2008) Visual support for work assignment in process-aware information systems. In: Dumas M, Reichert M, Shan MC (eds) Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, 2–4 September 2008. Proceedings, Springer, Lecture Notes in Computer Science, vol 5240, pp 67–83
39. Mendling J, van Dongen B, van der Aalst W (2007) Getting Rid of the OR-Join in Business Process Models. In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC'07), IEEE Computer Society, Annapolis, Maryland, pp 3–14
40. Ouyang C, Dumas M, van der Aalst W, ter Hofstede A (2006) From business process models to process-oriented software systems: The BPMN to BPEL way. Tech. Rep. BPM-06-27, http://www.BPMcenter.org
41. Ouyang C, La Rosa M, ter Hofstede A, Dumas M, Shortland K (2008) Toward Web-Scale Workflows for Film Production. IEEE Internet Computing 12(5):53–61
42. Reichert M, Rinderle S, Dadam P (2003) ADEPT workflow management system. In: van der Aalst W, ter Hofstede A, Weske M (eds) Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26–27, 2003, Proceedings, Springer, Lecture Notes in Computer Science, vol 2678, pp 370–379
43. Reichert M, Rinderle S, Kreher U, Dadam P (2005) Adaptive process management with ADEPT2. In: Proceedings of the 21st International Conference on Data Engineering (ICDE'05), IEEE Computer Society Press, Tokyo, Japan, pp 1113–1114
44. Rosemann M, van der Aalst W (2007) A configurable reference modelling language. Inf Syst 32(1):1–23
45. Rozinat A, Wynn M, van der Aalst W, ter Hofstede A, Fidge C (2008) Workflow simulation for operational decision support using design, historic and state information
46. Russell N (2007) Foundations of Process-Aware Information Systems. PhD Thesis, Queensland University of Technology, Brisbane, Australia, available through http://www.yawl-system.com
47. Russell N, van der Aalst W, ter Hofstede A, Edmond D (2005a) Workflow resource patterns: Identification, representation and tool support. In: Pastor O, Falcão e Cunha J (eds) Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05), Springer, Porto, Portugal, Lecture Notes in Computer Science, vol 3520, pp 216–232
48. Russell N, ter Hofstede A, Edmond D, van der Aalst W (2005b) Workflow data patterns: Identification, representation and tool support. In: Delcambre L, Kop C, Mayr H, Mylopoulos J, Pastor O (eds) Proceedings of the 24th International Conference on Conceptual Modeling (ER 2005), Springer, Klagenfurt, Austria, Lecture Notes in Computer Science, vol 3716, pp 353–368
49. Russell N, van der Aalst W, ter Hofstede A (2006a) Workflow exception patterns. In: Dubois E, Pohl K (eds) Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06), Springer, Luxembourg, Luxembourg, Lecture Notes in Computer Science, vol 4001, pp 288–302
50. Russell N, ter Hofstede A, van der Aalst W, Mulyar N (2006b) Workflow control-flow patterns: A revised view. Tech. Rep. BPM-06-22, http://www.BPMcenter.org
51. Schonenberg H, Mans R, Russell N, Mulyar N, van der Aalst W (2007) Towards a taxonomy of process flexibility (extended version). Tech. Rep. BPM-07-11, http://www.BPMcenter.org
52. Schonenberg H, Mans R, Russell N, Mulyar N, van der Aalst W (2008) Process flexibility: A survey of contemporary approaches. In: Dietz J, Albani A, Barjis J (eds) Advances in Enterprise Engineering I, 4th International Workshop CIAO! and 4th International Workshop EOMAS, held at CAiSE 2008, Montpellier, France, 16–17 June 2008. Proceedings, Springer, Lecture Notes in Business Information Processing, vol 10, pp 16–30
53. Sheth A (1997) From contemporary workflow process automation to adaptive and dynamic work activity coordination and collaboration. In: DEXA '97: Proceedings of the 8th International Workshop on Database and Expert Systems Applications, IEEE Computer Society, Washington, DC, p 24
54. Verbeek H, van der Aalst W, ter Hofstede A (2007) Verifying Workflows with Cancellation Regions and OR-joins: An Approach Based on Relaxed Soundness and Invariants. Comput J 50(3):294–314
55. Weber B, Reichert M, Rinderle-Ma S (2008) Change patterns and change support features – Enhancing flexibility in process-aware information systems. Data Knowl Eng 66(3):438–466
56. Weske M (2001) Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 7, IEEE Computer Society, Washington, DC, p 7051
57. Wynn M, Edmond D, van der Aalst W, ter Hofstede A (2005) Achieving a general, formal and decidable approach to the OR-join in workflow using Reset nets. In: Ciardo G, Darondeau P (eds) Proceedings of the 26th International Conference on Application and Theory of Petri nets and Other Models of Concurrency (Petri Nets 2005), Springer-Verlag, Miami, Lecture Notes in Computer Science, vol 3536, pp 423–443
58. Wynn M, Edmond D, van der Aalst W, ter Hofstede A (2006) Verifying workflows with Cancellation Regions and OR-joins: An Approach Based on Reset nets and Reachability Analysis. In: Dustdar S, Fiadeiro J, Sheth A (eds) Proceedings of 4th International Conference of Business Process Management, Springer-Verlag, Vienna, Austria, Lecture Notes in Computer Science, vol 4102, pp 389–394
59. Zisman M (1977) Representation, specification and automation of office procedures. PhD thesis, Wharton School of Business, University of Pennsylvania, PA

Nick Russell has over 20 years experience in the Australian IT industry in a series of technical and senior management roles. He received his PhD from Queensland University of Technology in 2007 and is currently a Postdoctoral Fellow at Eindhoven University of Technology in The Netherlands. Nick has been the driving force for the development of the workflow data and resource patterns and the recent revision of the control-flow patterns. He is also responsible for the development of the newYAWL business process reference language.

Arthur ter Hofstede received his PhD from the University of Nijmegen in the Netherlands in 1993. He is a Professor in the Faculty of Information Technology of Queensland University of Technology and co-leader of its BPM Group. Arthur is an original and ongoing contributor to the Workflow Patterns initiative and he manages the YAWL initiative at QUT.