FlowiXML: a step towards designing workflow management systems

Josefina Guerrero García*,
Jean Vanderdonckt and
Juan Manuel González Calleros

Belgian Laboratory of Computer–Human Interaction (BCHI)
Université catholique de Louvain
Place des Doyens, 1 – B-1348 Louvain-la-Neuve, Belgium
Fax: +32 (010) 478324
E-mail: guerrero@isys.ucl.ac.be
E-mail: vanderdonckt@isys.ucl.ac.be
E-mail: gonzalez@isys.ucl.ac.be
*Corresponding author

Abstract: This paper addresses the need for supporting the design of user interfaces for workflow management systems. Based on the already existing task and domain models, an approach is proposed to design a workflow model that explicitly articulates its new concepts with respect to the concepts belonging to the task and the domain, but with some extensions. The specifications of the workflow user interface are then stored in a model repository where all user interface aspects are expressed in a uniform XML-compliant user interface description language. From these specifications, the user interface of the workflow could be generated in HTML based on identified design patterns, along with the dialogue expressed in SCXML, a W3C standard for expressing state charts. This process is integrated in ATOMS, a content management software which integrates the generated interfaces with the final contents. In addition, the system automatically generates a personal 'to do' list and a workflow list to locate the progress of each workflow instance. A real-world case study is presented to exemplify this process.

Keywords: process model; task model; workflow model.

Reference to this paper should be made as follows: Guerrero García, J., Vanderdonckt, J. and González Calleros, J.M. (2008) 'FlowiXML: a step towards designing workflow management systems', *Int. J. Web Engineering and Technology*, Vol. 4, No. 2, pp.163–182.

Biographical notes: Josefina Guerrero García is pursuing her Master in Philosophy in Economic and Management Sciences, in the research area of information systems. She received her Master's Degree in Management at the Institute of University Studies in 2001. Her research interests include workflow models, business process management and information systems design. She is a member of the Belgian Laboratory of Computer–Human Interaction (BCHI) and the UsiXML Consortium.

Jean Vanderdonckt is a Professor of Information Systems Unit (ISYS), School of Management (IAG) at the Université catholique de Louvain (UCL). He received his PhD degree in Computer Science at Facultés Universitaires

Notre-Dame de la Paix, in 1997. He is the Head of the Belgian Laboratory of Computer–Human Interaction (BCHI), and a Partner and Coordinator of UsiXML Consortium.

Juan Manuel González Calleros is pursuing his Master in Philosophy in Economic and Management Sciences, in the research area of information systems. He received his Master's Degree in Computer Sciences at the National Institute of Astrophysics, Electronics and Optics in 2003. His research interests include workflow models, business process management, information systems design, model-based development, and three-dimensional development. He is a member of the Belgian Laboratory of Computer–Human Interaction (BCHI) and the UsiXML Consortium.

1 Introduction

The development of workflow systems progressively represents major new challenges today for several important reasons:

- Often, individual users in organisations already have their own software to use to carry out the interactive tasks that they have been assigned. This is not problematic. But when the time comes to communicate the results of their tasks to their hierarchy or to their colleagues, apart from using traditional e-mail, they do not rely on dedicated software for supporting the communication.
- Small-scale and large-scale organisations usually experience some problems not in defining and assigning the tasks to the individual workers, but in integrating them into a complete workflow, which could be reinforced by a system. Indeed, if a workflow is implemented manually, it is more complicated with respect to the constraints imposed by this workflow as the workers could feel free to respect or not to respect the rules.
- It is often heard that people in organisations are forced to change their organisational structure and work process because of the setup of a new workflow system, as opposed to the tailoring of such system to the already existing workflow. Of course, installing a new computer-based workflow inevitably changes the procedures; this, however, should be limited.
- When the organisational structure changes, but the workflow does not change, it becomes complicated to reassign the tasks if a logical workflow has not been defined independently of the organisational structure.
- It is important to cite Mandviwalla and Olfman's criteria for support group interactions, such as the following ones we selected:
  a 'Support carrying out group tasks' from the individual level continuously throughout the global level: individual, within groups, for the group as a whole, among groups, within organisations, and among organisations.
  b 'Support multiple ways to support a group task': in principle, there should not be one unique way to carry out a single group task, but several mechanisms should be offered for this purpose. If a mechanism is no longer available, another one should be selected.

c 'Support the group evolution over time': when the group evolves over time, the workflow definition should be easily maintained and reflected in the system.

From a user interface standpoint, there exists today a gap between the development life cycle of user interfaces of individual tasks as they are supported by isolated application and the development life cycle of the complete workflow. In particular, would it be possible to generate the workflow user interface from the workflow definition and from the different constructs that link the tasks and the users together? This paper addresses this need from a fundamental point of view by reviewing some selected work (Section 2), introducing the modelling concepts supporting this (Section 3), defining an extension of a User Interface Description Language to express these concepts in an integrated way (Section 4) and transferring them into ATOMS, a software that supports defining the workflow and generates the corresponding code (Section 5). A case study is presented in Section 5. Section 6 concludes the paper.

2 Related work

2.1 Organisational structure

The user interfaces model of an organisational workflow system, must be based on how organisations works and which elements composed them. The organisational theory studies alternative structures for (business) organisations. The structure of an organisation defines the jobs, resources, their responsibilities, tasks and goals. Mintzberg (1982) proposes an organisational typology based on five components of the organisation (strategic apex, middle line, technostructure, support staff and operating core); five ways of affecting coordination (direct supervision, standardisation of work, standardisation of skill, standardisation of output and mutual adjustment); and five types of organisational decentralisation (centralisation, limited horizontal decentralisation, horizontal and vertical decentralisation, limited vertical decentralisation and selective decentralisation) resulting in five organisational configurations called Simple Structure, Machine Bureaucracy, Professional Bureaucracy, Divisionalised Form and Adhocracy. Independently of the configuration of an organisation, within it, there are works to do and resources that develop these works; these two elements could be controlled by information systems, the so-called workflow management systems.

2.2 Workflow models

Organisations are forced to increasingly integrate and automate their business process using the workflow, a common term used when processes are automated and controlled. There are several workflow definitions: zur Muehlen (2002) defines workflow as a specific representation of a process, which is designed in such a way that the formal coordination mechanisms between activities, applications, and process participants can be controlled by an information system, the so-called workflow management system; WfMC (1999) defines workflow as the automation of a business process (set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships), in whole or part, during which documents, information or tasks are passed from one participant to another for action, accordingly to a set of procedural rules.

Workflow technology facilitates modelling, redesigning and administration of process in an organisation (Eichholz *et al.*, 2004). Models workflows have been proposed for the design and specifications of it. In addition, workflow patterns called *workflow resources patterns,* have been identified for resources (Russell *et al.*, 2005), and *workflow patterns* for routing constructs (van der Aalst *et al.*, 2003).

Several *workflow management systems* have been developed to manage the workflow, such as the Progression Model (Stavness and Schneider, 2004), Action Port Model (Carlsen, 1997), State Chart XML (SCXML) (W3C, 2005), Flexo (Denali),[1] and ATOMS® (Defimedia).[2]

2.3 Task models

According to Limbourg (2004), a task model describes the various tasks to be carried out by a user in interaction with an interactive system.

There are several different approaches to task models, such as ConcuTaskTrees (CTT) (Paternò *et al.*, 1997); Goals, Operators, Methods, and Selection rules (GOMS) by (Card *et al.*, 1983); and Hierarchical Task Analysis (HTA) (Annett and Duncan, 1967).

Limbourg and Vanderdonckt (2003) made a comparative analysis of significant task models, their methods and supporting tools. In their review they identified that CTT supports engineering approaches to task modelling with five concepts: task, objects, actions, operators and roles. GOMS is an engineering model for human performance to enable quantitative predictions. Methods are a central concept in GOMS to describe how tasks are actually carried out; *methods* are a sequence of operators that describe task performance. Tasks are triggered by goals and can be further decomposed in subtasks corresponding to intermediary goals. When several methods compete for the same goal, a selection rule is used to choose the proper one. While GOMS models are useful only for tasks that involve substantial amounts of routine procedure execution, they can often enable interface designers to start evaluating usability and making design iterations before the investment in prototype development (Kieras, 1994). HTA describes tasks in terms of three main concepts: tasks; tasks hierarchy; and plans on the basis of interviews, user observation, and analysis of existing documents.

2.4 Notations

Nowadays, state chart diagrams, state machine notations such as SCXML (W3C, 2005), and the Petri Nets Notations (van der Aalst and van Hee, 2002) are used to model workflow. On the other hand, task models use the CTT, GOMS, or UAN notations. UAN provides a notation to describe the dynamic behaviour of graphical user interfaces, where the tasks are represented asynchronously with operators that denote the temporal relationships (Stavness and Schneider, 2004).

3 Conceptual modelling of workflow

3.1 FlowiXML to model workflow

FlowiXML workflow components are cases, resources and triggers, all related to a particular process, as defined by van der Aalst and van Hee (2002). Each process consists of a number of tasks and a set of conditions that determine the order of the tasks.

FlowiXML workflow models are used to represent the flow of the work inside and between organisations. Workflow models are used to model the flow of work and task models are used to describe the way humans perform tasks to accomplish a goal. To ensure that FlowiXML is developed according to organisations' requirements, we propose to have both representations and allow some flexibility to model the work with different levels of detail, as is necessary for each organisation. In fact, using tasks models to describe workflow adds the possibility of adaptation and flexibility (Eichholz *et al.*, 2004; Traetteberg, 1999).

3.2 Organisational components

Typically, only resources and their roles within organisations are modelled in most workflow models. In Russell *et al.* (2005) the workflow resource patterns are proposed to manage the task assignment, although they just focus on human resources. We adapted these patterns to represent the assignment and delegation of tasks to resources, whether they are human or not. Some other organisational components that we considered are organisational units, material and immaterial resources (not human resources), the agenda (to do list) and the tasks.

3.3 Process model

In the context of this methodology, a process is a set of tasks that is necessary to carry out. The definition of a process indicates which tasks must be performed and in what order. Similarly with WfMC (1999) we defined a process as a formalised view of a business process, represented as a coordinated set of process activities that are connected in order to achieve a common goal. Our model organises tasks in a higher level and determines the order of execution; this ensemble we called the process model. Notice that tasks themselves could be decomposed into subtasks.

3.4 FlowiXML task model

An extended version of CTT has been selected to represent tasks with their logical and temporal order. Task models are therefore composed of *tasks* and *task relationships*. Tasks are described with a name and a type. Task *type* may be: user's, interactive, system or abstract. A user task refers to a cognitive action such as taking a decision, or acquiring information. An interactive task involves an active interaction of the user with the system (*e.g.*, selecting a value, browsing a collection of items). A system task is an action that is performed by the system (*e.g.*, checking a credit card number, displaying a banner). An abstract task is an intermediary construct allowing a grouping of tasks of different types. Task relationships are of two main types: decomposition and temporal. *Decomposition* enables the representation of the hierarchical structure of a task tree. *Temporal* allows specifying a temporal relationship between sibling tasks of a task tree. LOTOS (Paternò *et al.*, 1997) operators are used here. Also, we used Limbourg (2004) relationship groups:

- Binary relationships – enabling, disabling, suspend/resume, order independence, concurrency with information passing, independent concurrency, enabling with information passing, deterministic choice and undeterministic choice.

- Unary relationships – optional, iteration and finite iteration.

Additionally, we introduce three more operators: disabling with information passing, inclusive choice and cooperation.

In order to have an appropriate representation of organisational requirements, it is important to consider:

- that a task could be defined by the user
- a task could be grafted on another one
- the way in which tasks are advertised, assigned and delegated to specific users for execution.

The term 'grafted on' (Petitjean, 1994) refers to a task (Tj) that has been started and that needs a complementary task (Ti) for its realisation. *Ti* is completely autonomous to *Tj.* In reference to the way in which the task is assigned to a user, we consider the workflow resource patterns proposed by Russell *et al.* (2005). However, the *delegation* pattern is reinforced with the work of Petitjean (1994) who implements the negotiation type. These last relationships will be defined in an intermodel relationship (*i.e.*, *mapping model*) (see Section 4.2.5).

4 User interface eXtended markup language extension to workflow

4.1 UsiXML

Multipath User Interface (UI) development (Limbourg, 2004) is based on the Cameleon Reference Framework (Calvary *et al.*, 2003), which defines UI development steps for multicontext interactive applications.

We used the User Interface Description Language (UIDL) User Interface eXtensible Markup Language (UsiXML)[3] throughout the development life cycle. This UIDL is characterised by the following principles:

- Expressiveness of UI – any UI is expressed depending on the context of use, thanks to a suite of models that are analysable, editable and manipulable by a software agent.
- Central storage of models – each model is stored in a model repository where all UI models are expressed similarly.
- Transformational approach – each model stored in the model repository may be subject to one or many transformations supporting various development steps. Each transformation is itself specified thanks to UsiXML.

Contrarily to other frequently used UIDLs, *e.g.*, UIML[4] and XISL (Katsurada *et al.*, 2003) for multiplatform, multimodal UIs, UsiXML enables the specification of all models and the transformations between until a final UI is obtained. UsiXML is able to specify various UIs with the five modalities of interaction defined in Section 1. For this purpose, UsiXML is structured according to four basic levels of abstractions defined by the Cameleon reference framework. The actual specification is composed of:

- *Task model*

  Extension of CTT has been selected to represent user's tasks along with their logical and temporal ordering.

- *Domain model*

  The domain model is generally developed by software engineers and gives 'as is' (often under the form of an Application Programming Interface (API)) to UI designers. The rest of the job consists of connecting the UI to the functional core API while respecting some architectural principles. Domain model concepts are classes, attributes, methods, objects and domain relationships.

- *Abstract user interface model*

  An AUI is a user interface model that represents a canonical expression of the rendering and manipulation of the domain concepts and functions in a way that is as independent as possible from modalities and computing platform specificities. An Abstract User Interface (AUI) is populated by *Abstract Interaction Objects* (AIO) and *abstract user interface relationships.* These concepts constitute a vocabulary that is independent of the modality and the computing resources for which a system is targeted.

- *Concrete user interface model*

  The CUI is a UI model allowing a specification of an appearance and behaviour of a UI with elements that can be perceived by users. By definition, a Concrete User Interface (CUI) is modality-dependent as any CUI instance refers to the interaction modalities that have been selected for this UI. This reference can be unique in case of a 'mono-modal' CUI or multiple in case of a multimodal CUI. A CUI model is composed of *Concrete Interaction Objects* (CIO) and *concrete relationships.* Concrete interaction objects and relationships are further refined into graphical objects and relationships and auditory objects and relationships. Other types might complement these two categories as more modalities could be taken into account.

- *Context model*

  A context model is a model describing the three aspects of a context of use in which an end user carries out an interactive task with a specific computing platform in a given surrounding environment. Consequently, a context is hereby defined as a triple of the form <*e, p, u*> where *e* is an element of the environments set considered for the interactive system, *p* is an element of the platforms set considered for the interactive system and *u* is an element of the users set for the interactive system.

- *Intermodel relationships (i.e., mapping model)*

  Model integration is a well-known issue in the transformation-driven development of UI. Rather than proposing a collection of unrelated models and model elements, this proposal provides a designer with a set of predefined relationships allowing a mapping of elements from heterogeneous models and viewpoints. This can be useful, for instance, for enabling the derivation of the system architecture (mappings between domain and CUI/AUI models), for traceability in the development cycle (reification, abstraction and translation), for addressing context-sensitive issues (has context), for dialogue control issues and for improving the preciseness of model derivation heuristics.

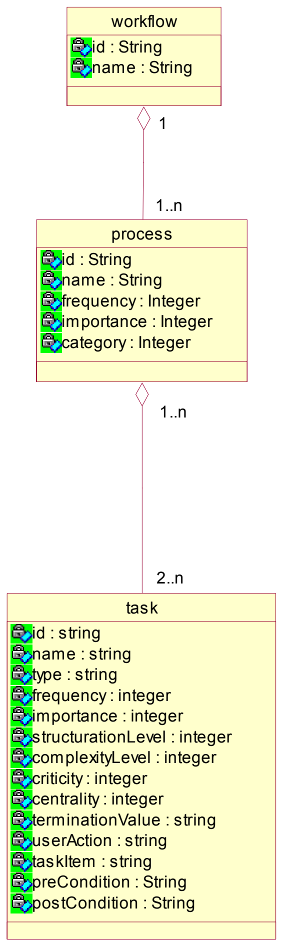
4.2 UsiXML extension to workflow

Extending the ontology of UsiXML to other types of model and concepts is one of the purposes of the language. Actually, one of the most desirable model extensions is to consider UML workflow models; as workflow models represent many advantages with respect to task models and offer an appropriate notation for collaborative applications. In order to transform the actual specification of the task model and incorporate the workflow model, it is necessary to consider other components, such as the processes, organisational units, the resources and the jobs.

4.2.1 Workflow

The workflow model consists of a number of processes and tasks that are interconnected through operators and relationships (Figure 1). Workflows are described with a name. Also, each process and task is represented inside a model. In addition, we propose to have a representation of some organisational components that are involved with the execution of work.

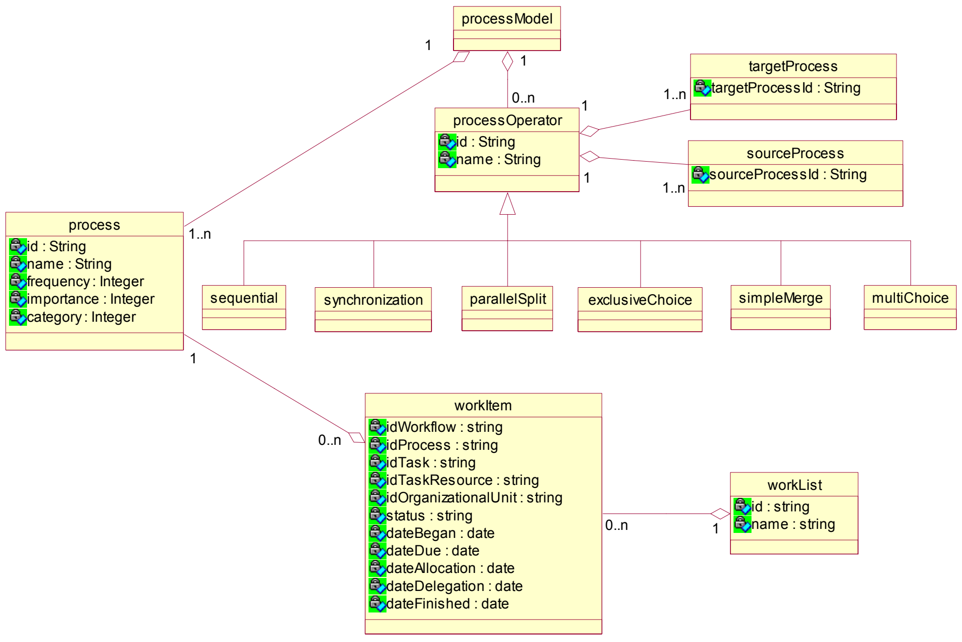
Figure 1 Conceptual view of the workflow model



4.2.2 Process model

Simple *processes* belong to the *process model*, which represents the relationships between the different processes that are involved in a workflow.

Figure 2 Conceptual view of the process model



The process model (see Figure 2) is composed of:

- *targetProces* – Designates one or several target(s) of a relationship which it is part of.
- *sourceProces* – Designates one or several source(s) of a relationship which is part of a *processOperator.*
- *workList* – A workList manages the flow of work among the taskResources.
- *workItem* – Is the representation of the task to be processed. It could contain the identification of the workflow and the identification of the process to which it belongs; the identification of the task resource that develops the task and the identification of the organisational unit where the task is performed; the actual status of the task (for instance: not started, in progress, in progress with delay, in progress with due date close, suspend, cancel, finished); the date when the task begins, the deadline (*i.e.*, date due), the date when the task could be assigned or delegated, and the date when the task was completed.
- *processOperator* which are operators that indicate the different ways in which the processes could be executed. We define them as follows:

a sequential indicates that a number of processes are performed one after the other

b synchronisation is used when multiple parallel processes converge into one single thread of control

c parallelSplit indicates that two or more process can be executed in parallel, thus allowing processes to be executed simultaneously or in any order

d exclusiveChoice indicates that one of several branches is chosen

e simpleMerge indicates that two or more alternatives branches come together without synchronisation

f multiChoice is used when any of two processes is chosen. However, it is also possible that both need to be executed.

4.2.3 Task model

As we mentioned above, a *task model* is composed of *tasks* and *task relationships* (Figure 3). The following definitions describe the elements of the task model and their relations.

- *taskModel* – Task models describe end users' view of interactive task while interacting with the system. A task model represents a decomposition of tasks into subtasks linked with task relationships. Therefore, the decomposition relationship is the privileged relationship to express this hierarchy; child temporal relationships express the temporal constraints between subtasks of the same parent task.
- *target* – Target relationships designate one or several target(s) of a relationship.
- *source* – Source relationships designate one or several source(s) of a relationship.
- *task* – Task is the basic structure that composes the task model. Tasks are activities that have to be performed to reach a goal, (Paternò *et al.*, 1997).
- *taskRelationship* – Task relationships are relationships involving several occurrences of different (or the same in some cases) tasks. Task relationships may be of two types: decomposition or temporal relationship.
- *decomposition* – Decomposition relationships enable the representation of the hierarchical structure of the task tree, such as adjacency for *graphicalIndividualComponents*. Decomposition relationships are implicit within the XML syntax of the language, as represented by the simple embedding of elements.
- *temporal* – Temporal relationships represent a specification of temporal relationships between tasks.

  1 *binaryRelationship* – Binary relationships are a type of temporal relationships that connect several instances of two different tasks.

    a *enabling* – Enabling relationships specify that a target task cannot begin until source task is finished.

    b *disabling* – Disabling relationships refer to a source task that is completely interrupted by a target task.

c *suspendResume* – Suspend resume relationships refer to source task that can be partially interrupted by a target task and after the target task is completed, the source task will be concluded.

d *orderIndependence* – Order independence relationships are when two tasks are independent of the order of execution.

e *concurrencyWithInformationPassing* – Concurrency with Information Passing relationships are a type of temporal relationships where two tasks are in concurrency execution and passing information between them.

f *independentConcurrency* – Independent concurrency relationships are a type of temporal relationships where two tasks are executed concurrently but are independent from each other and there is no information interchange.

g *enablingWithInformationPassing* – Enabling with information passing relationships specifies that a target task cannot be performed until the source task is performed, and that information produced by the source task is used as an input for the target task.

h *cooperation* – A cooperation relationship specifies the relationship of cooperation between two or more tasks.

i *inclusiveChoice* – An inclusive choice relationship specifies that both of two tasks or just one of them, or neither of them could be executed.

j *deterministicChoice* – Deterministic choice relationships refer to two source tasks that could be executed but once one task is initiated, the other cannot be accomplished anymore.

k *undeterministicChoice* – Undeterministic choice relationships define the relation between two source tasks in which both tasks could be started but once one task is finished, the other cannot be accomplished anymore.

l *disablingWithInformationPassing* – Disabling with pass information relationships occurs if one task is completely interrupted by another task; and the information produced in the first task is used as an input for the second task.

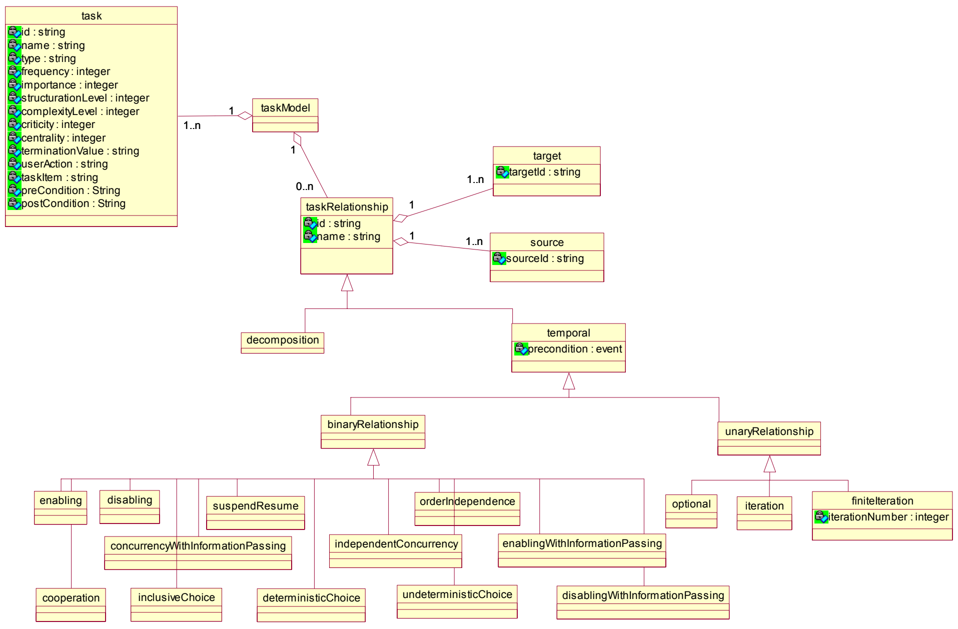
2 *unaryRelationship* – Unary relationships are temporal relationships that connect several instances of the same task.

a *optional* – Option relationships refer to source tasks that are optional.

b *iteration* – Iteration relationships indicate source tasks that may be iterated.

c *finiteIteration* – Finite iteration tasks indicate tasks that may be iterated *n* times.

Figure 3 Conceptual view of the task model



4.2.4 Organisational components

We propose an organisation framework (Figure 4) that is composed of:

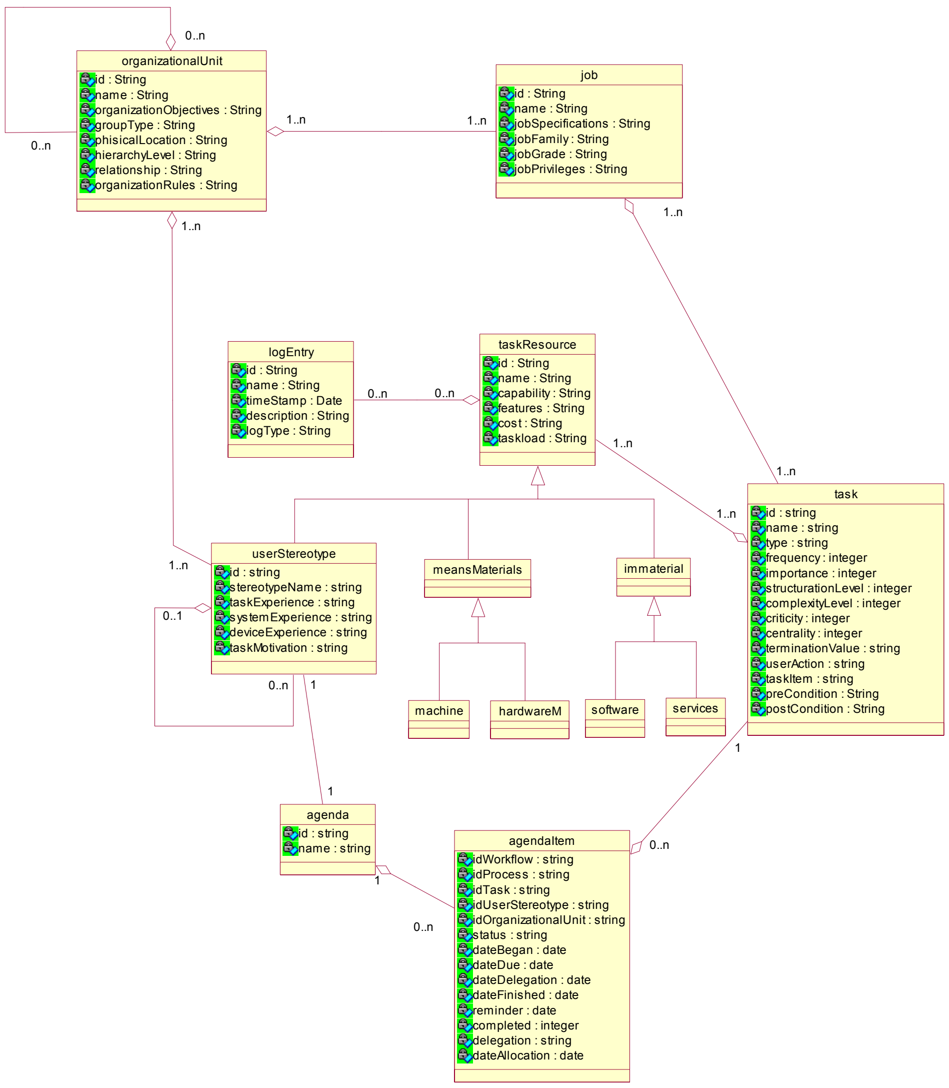
- Organisational unit – An organisational unit is a formal group of people working together with one or more shared goals or objectives. It could be composed of other organisational units.
- Task resources – A *taskResource* is an entity that is directly or indirectly involved in carrying out the work. We identify three task resources:

  1. *User stereotype* represents the set of users sharing the same values. Each stereotype may in turn be decomposed into substereotypes.
  2. *Means materials* is a type or resource that is physically tangible and is a non-human resource.
  3. *Immaterial* is a type of resource that is physically intangible; it does not have a material form or substance.

- LogEntry – LogEntry describes specific characteristics that resources may possess. Each resource may have a logEntry associated with them.
- Job – Jobs are the total collection of tasks, duties and responsibilities assigned to one or more positions that require work of the same nature and level.

- Task – Already defined above, is a task that belongs to the task model but needs resources to be carried out.
- AgendaItem – Agenda items are the tasks that a userStereotype has to perform.
- Agenda – The agenda is a list of agendaItems that are assigned to userStereotypes. A userStereotype has one and only one agenda and an agenda belongs to one and only one userStereotype.

Figure 4 Conceptual view of the organisation components

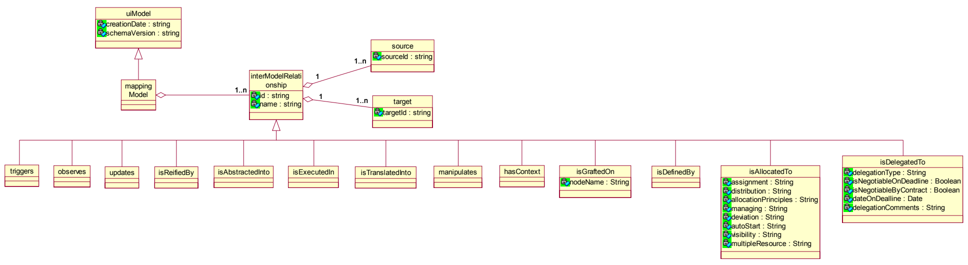


4.2.5 Mapping model extension

Based on the set of predefined relationships of UsiXML, Limbourg (2004), allows a mapping of elements from heterogeneous models and viewpoints (Figure 5). Several relationships can be defined to show the relationships between the domain model and the UI models (both abstract and concrete (see Section 4.1 for more details)):

- *Observes* is a mapping defined between an interaction object and a domain model concept (either an attribute, or an output parameter of a method).
- *Updates* is a mapping defined between an interaction object and a domain model concept (specifically, an attribute). 'Updates' describe the situation where the attribute of an object in the domain model must be synchronised with the content of a UI object.
- *Triggers* is a mapping defined between an interaction object and a domain model concept (specifically, an operation). This mapping describes that a UI object is able to trigger a method from the domain model. The mapping ensures the traceability of the development cycle.
- *Is Executed In* maps a task to an interaction object (a container or an individual component) allowing its execution. This relationship is notably useful for deriving a dialogue control component, for ensuring that all tasks are supported appropriately by the system.
- *Is Reified By* indicates that a concrete object is the reification of an abstract one through a reification transformation. *Is Abstracted Into* indicates that an abstract object is the reification of a concrete one through an abstraction transformation.
- *Is Translated* Into enables tracing of the adaptation of one component in another. It can be used while defining a transformation called translation.
- *Manipulates* maps a task to a domain concept. It may be an attribute, a set of attributes, a class (or an object), or a set of classes (or a set of objects). This relationship is useful when it comes to finding the most appropriate interaction object to support a specific task.
- *Has Context* maps any model element to one or several contexts of use.

Figure 5 Conceptual view of the mapping model



We extend these mapping by adding the following:

- *Is Grated On* grates a task on another one. This relationship is useful when a task (Tj) is being executed, and a complementary task (Ti) is defined to realise the first task. *Ti* is completely autonomous to *Tj.*
- *Is Defined By* refers to a task defined by a userStereotype.
- *Is Allocated To,* in this case we adapt the proposal of Russell *et al.* (2005). A task is assigned to a taskResource. We define several allocation relationships for this assignment:

  a Assignment – Is the manner in which tasks are advertised to specific resources for execution.

  b Distribution – Is the manner in which newly created tasks are proactively offered or allocated to resources by the workflow system.

  c Allocation principles – Is the manner in which tasks are allocated to resources by the workflow system.

  d Managing – Is the manner in which the tasks are initiated by individual resources.

  e Deviation – Is when the normal sequence of state transitions for a task is varied.

  f Auto-start – Are situations where execution of task is triggered by specific events in the life cycle of the task or the related process definition.

  g Visibility – Are the scopes in which task availability and commitment can be viewed by taskResources.

  h Multiple resources – Situations where there is often many-to-many correspondence between the taskResources and work tasks in a given allocation or execution.

- *Is Delegated to* – A userStereotype who is assigned to a task allocates it to another userStereotype. We define several delegation relationships for this assignment:

  a Delegation type – Describes the type of delegation; it could be by negotiation, by assignment or by tender.

  b Is Negotiable On Deadline – Indicates if the task is negotiable on the time limit for its execution.

  c Is Negotiable By Contract – Indicates if in the negotiation there is a contract in which some conditions for executing a task exist.

  d Date – Is the date in which the delegation was done.

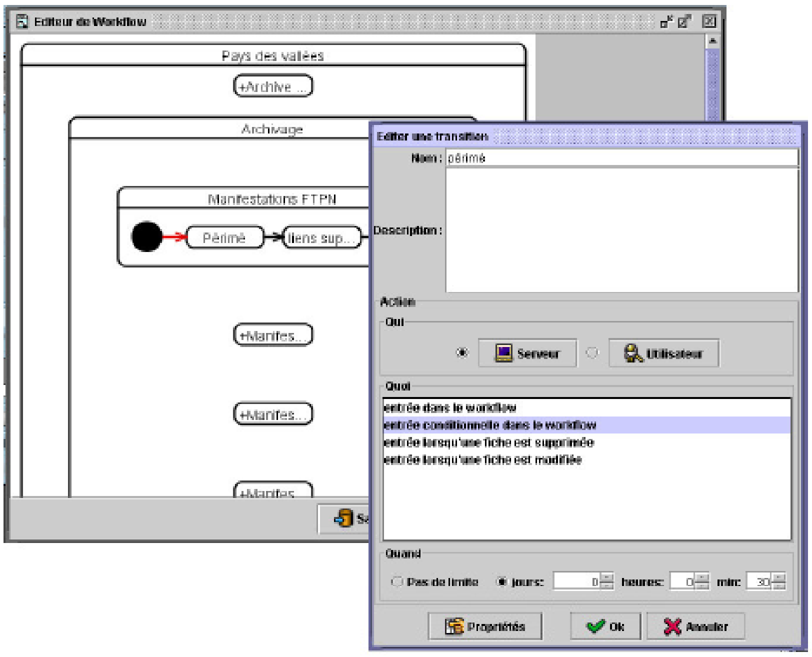
  e Delegation Comments – In delegationComments it is possible to add extra information; for instance, some observation about the task to be delegated.

5 The ATOMS system: supporting the description of a workflow system

5.1 ATOMS®

ATOMS® (Defimedia) is an information system that supports the progressive deployment of a communication systems integrated in the organisation. ATOMS is an environment of web applications deployment that allows integration in services, applications and data. It has a workflow editor (Figure 6) that allows the automation of a series of tasks which could be carried out following a predefined set of actions (add a document, arrival of a document on a certain date, *etc.*). The actual specification of the workflow module integrated in the tool is established on UML state chart diagrams. With this module it is possible to establish the logic of the flow that could be sequential, parallel, multistep, mode in which actions can be triggered by the computer (*e.g.*, date from planned publication) or by a human operator.

Figure 6 Workflow editor of ATOMS



Editeur de Workflow

Editer une transition

Nom : périmé

Description :

Action

Qui

[✓ Serveur] Utilisateur

Quoi

entrée dans le workflow
entrée conditionnelle dans le workflow
entrée lorsqu'une fiche est supprimée
entrée lorsqu'une fiche est modifiée

Quand

Pas de limite [✓ jours: 0] heures: 0 min: 30

The workflow elements could be linked to elements of the UI. The configurations of the workflow are stored in a documented file XML, editable manually or via our UML editor. As this workflow editor is based on state chart diagrams, our proposal is to extend the tool using the concepts of FlowiXML in order to represent the workflow in more detail, considering other organisational components, making work ‘controllable’ and flexible, encouraging communication between employees. State charts are useful when the exact sequence of events is not known in advance; we only know about the current state and what can be done next.

5.2 Case study

To prove the feasibility of FlowiXML, we present the Authorisation Application case study. This problem concerns the formation application made by a trainer to the administration committee. In summary this task concerns the authorisation applied to teach a training course. The request could be made in different places, all of them with access to the online system that handles the training's information among other information. The trainer fills out a form that is sent to the person responsible for the training courses, the Formation Responsible (FR) at the IFPM department, via e-mail. This event triggers the *generateTask* event, generating a message in the user interface of the FR that a new training course request has been submitted. The FR validates the information received and once validated, the training information is published and can be accessed online by the public in general. In particular, a set of known organisations will be notified by e-mail about the training course offered, also adding a new message to their UI. To be registered, the organisations must complete a form indicating participant's personal information as well as the payment method. Several administrative actions are followed such as: printing the forms; the organisation manager validates and signs the course registration; the forms are sent to the IFPM department to signify their agreement to teach the training course. A third actor, inspector Agoria, is notified about this negotiation and condenses both signed agreements and gives his approval. Then the IFPM department receives the notification that could be accepted or rejected or requires more information. If accepted, the administration committee of the training centre decides whether to approve the course or not. If so, the FR notifies the trainer who makes the application to teach the course.

To handle this workflow we have identified 48 tasks that could be used with several courses at the same time. Different organisations are involved in the workflow, as well as different actors, resources and tasks. For simplicity and space reasons we will just identify a few components of some of them in FlowiXML (Table 1) and represented Petri Nets (Figures 7–10).

Table 1 Components of tasks

Task / *Component*	*Request authorisation*	*Validate request*	*Receive notification*	*Notify deliberation*
Organisational unit	Trainer unit	IFPM	Agoria	IFPM
Job	Trainer	Formation responsible	Inspector	Responsible
taskResource	userStereotype	userStereotype	userStereotype	userStereotype
isAllocatedTo	Resource initiated allocation	Direct assignment	Direct assignment	Direct assignment

Figure 7 Request authorisation

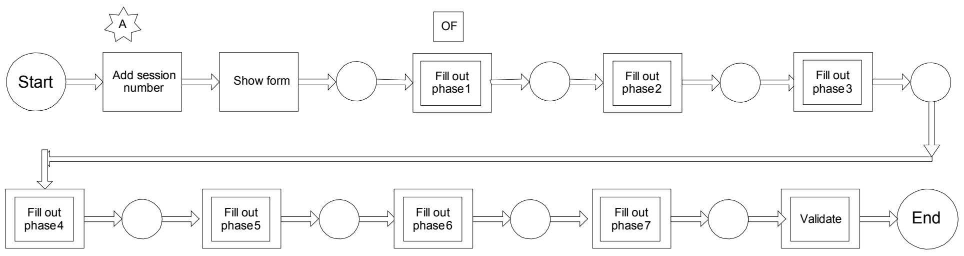


Figure 8 Validate request

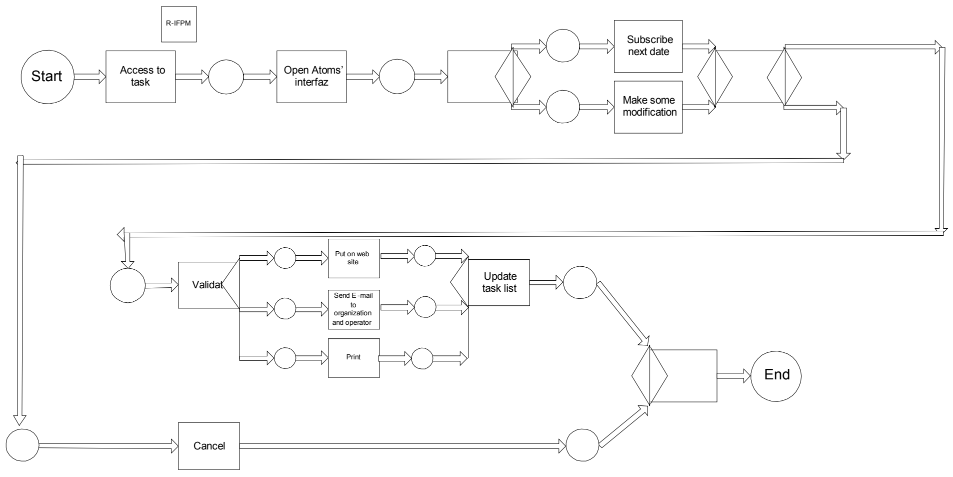


Figure 9 Receive notification

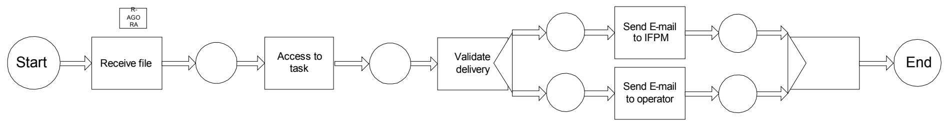
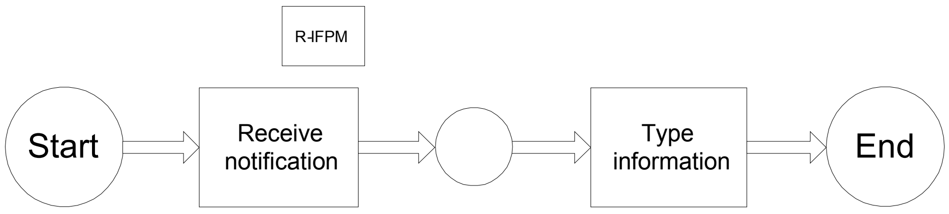


Figure 10 Notify deliberation



6 Conclusion

This paper has addressed the need for developing also the workflow user interface by integrating it into the user interface design of individual applications. For this purpose, a conceptual modelling approach has been adopted that integrates the following notions (old and new): task, domain, process, workflow, job definition, organisational structure, 'to do' list workflow list and resources. These concepts along with their attributes have been integrated in the syntax of UsiXML, a User Interface Description Language (although it could be integrated as well in other languages such as UIML or XIML). An extension to the ATOMS software has been developed in Java to support these concepts and to express the dynamic linking between these concepts in a workflow based on a state chart, which is itself specified in SCXML, a W3C standard for expressing this kind of model. A real-world case study has been reported and summarised (the complete one is available on request in an internal report). The mostly significant advantage of that system (as observed during its usage) is that the designer is not forced to design the workflow user interface separately, but that a predefined one could be automatically produced based on some design patterns. For instance, if a delegation mechanism is defined, then an abstract user interface is directly attached to this delegation, which results in a concrete user interface for which the HTML code is generated. Therefore, there is a continuity between the definition of high-level concepts (here, a delegation of tasks between users in a workflow) and its support via a final user interface (here, a graphical interface for letting the source user delegate his/her task to a target user).

In the future, we are expecting to enlarge this set of design patterns so that several levels of details could be offered for supporting the same task, as recommended by Mandviwalla and Olfman (1994).

Acknowledgement

We greatly thank the DefiMedia company for the development of this paper.

References

Annett, J. and Duncan, K. (1967) 'Task analysis and training design', *Occupational Psychology*, Vol. 41, pp.211–227.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. (2003) 'A unifying reference framework for multi-target user interfaces', *Interacting with Computers*, Vol. 15, No. 3, pp.289–308.

Card, S.K., Moran, T.P. and Newell, A. (1983) *The Psychology of Human–Computer Interaction*, Lawrence Erlbaum Associates.

Carlsen, S. (1997) 'Conceptual modeling and composition of flexible workflow models', PhD thesis, Norway: Norwegian University of Science and Technology.

Eichholz, C., Dittmar, A. and Forbrig, P. (2004) 'Using task modelling concepts for achieving adaptative workflows', *Proceedings of DSV-IS-EHCI'94*, Berlin: Springer-Verlag, LNCS, Vol. 3425.

Katsurada, K., Nakamura, Y., Yamada, H. and Nitta, T. (2003) 'XISL: a language for describing multimodal interaction scenarios', *Proc. of 5th Int. Conf. on Multimodal Interfaces ICMI'2003*, Vancouver, 5–7 November, New York: ACM Press, pp.281–284.

Kieras, D. (1994) 'A guide to GOMS task analysis', Technical report, University of Michigan, http://www.sahs.uth.tmc.edu/TRJohnson/HI%205354/guide.pdf.

Limbourg, Q. (2004) 'Multi-path development of user interfaces', PhD thesis, Belgique: Universite catholique de Louvain.

Limbourg, Q. and Vanderdonckt, J. (2003) 'Comparing task model for user interface design', in D. Diaper and N. Stanton (Eds.) *The Handbook of Task Analysis for Human–Computer Interaction*, Mahwah: Lawrence Erlbaum Associates, pp.135–154.

Mandviwalla, M. and Olfman, L. (1994) 'What do groups need? A proposed set of generic groupware requirements', *ACM Transactions on Computer–Human Interaction*, Vol. 1, No. 3, pp.245–268.

Mintzberg, H. (1982) *Structure & Dynamique des Organisations*, Paris, France: Les Editions d'Organisation.

Paternò, F., Mancini, C. and Meniconi, S. (1997) 'ConcurTaskTrees: a diagrammatic notation for specifying task models', *Proceedings Interact'97*, Chapman&Hall, pp.362–369.

Petitjean, T. (1994) 'Contribution a la specification de situations de cooperation ad hoc et a leur prise en compte dans les systèmes de Workflow', PhD thesis, Namur, Belgium: Facultés Universitaires Notre-Dame de la Paix.

Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M. and Edmond, D. (2005) 'Workflow resource patterns', *17th Conference on Advanced Information Systems Engineering (CAISE'05)*, Porto, Portugal, 13–17 June.

Stavness, N. and Schneider, K. (2004) 'Supporting flexible business processes with a progression model', in H. Trætteberg, P.J. Molina and N.J. Nunes (Eds.) 'Making model-based user interface design practical: usable and open methods and tools', *Proceedings of the First International Workshop on Making Model-based User Interface Design Practical: Usable and Open Methods and Tools MBUI 2004, CEUR Workshop Proceedings*, Funchal, 13 January, Vol. 103, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-103/.

Traetteberg, H. (1999) 'Modeling work: workflow and task modeling', in J. Vanderdonckt and A.R. Puerta (Eds.) *Proc. of 3rd Int. Conf. on Computer-Aided Design of User Interface, CADUI'99*, Louvain-la-Neuve, 21–23 October.

Van der Aalst, W. and van Hee, K. (2002) *Workflow Management. Models, Methods, and Systems*, Library of Congress Cataloging-in-Publication Data.

Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. and Barros, A.P. (2003) 'Workflow patterns', *Distributed and Parallel Databases*, Vol. 14, No. 1, pp.5–51.

W3C (2005) 'Working draft', World Wide Web Consortium, 5 July, http://www.w3.org.

WfMC (1999) 'Terminology & glossary', *Workflow Management Coalition*, Document Number WFMC-TC-1011, Document Status, No. 3.0, February.

Zur Muehlen, M. (2002) 'Workflow-based process controlling', *Foundation, Design, and Application of Workflow-driven Process Information Systems*, Berlin: Logos Verlag.

Notes

1 Denali, www.denali.be.

2 Defimedia, www.defimedia.be.

3 http://www.usixml.org

4 http://www.uiml.org