Chapter XVI
Developing User Interfaces for Community-Oriented Workflow Information Systems

Josefina Guerrero García
Université catholique de Louvain, Louvain School of Management (LSM), Belgium

Jean Vanderdonckt
Université catholique de Louvain, Louvain School of Management (LSM), Belgium

Juan Manuel González Calleros
Université catholique de Louvain, Louvain School of Management (LSM), Belgium

ABSTRACT

Technology to support groups is rapidly growing in use. In recent years, the Web has become a privileged platform for implementing community-oriented workflows, giving rise to a new generation of workflow information systems. Specifically, the Web provides ubiquitous access to information, supports explicit distribution of business process across workers, workplaces, and computing platforms. These processes could be all supported by platform-independent user interfaces. This chapter presents a model-driven engineering method that provides designers with methodological guidance on how to systematically derive user interfaces of workflow information systems from a series of models. For this purpose, the workflow is recursively decomposed into processes which are in turn decomposed into tasks. Each task gives rise to a task model whose structure, ordering, and connection with the domain model allows the automated generation of corresponding user interfaces in a transformational approach. The various models involved in the method can be edited in a workflow editor based on Petri nets and simulated interactively.

INTRODUCTION

There are a variety of definitions on virtual communities with all of them having in common the participation of people working and sharing information or knowledge in a shared space toward the accomplishment of a goal. Virtual communities are formed around different disciplines such as sociology, anthropology, medicine, computer science, management science, distance learning, bring together people sharing a common interest, concern or desire. Some times such virtual settings may impose a shared community-wide workflow, but this need not be always the case. In this chapter we will be concerned with virtual communities in which members share a common practice (i.e., learning, business process) which is to be interactively manifested to dispersed community members using different tools and computational devices. In such cases, one important aspect to consider is the design of the user interfaces (UIs) of the system that will be used by the community to foster collaboration. Ideally, such a design should be greed from the specificities of platforms, access terminals and local workflow bindings. To this effect, it is important to generate user interface software for community-oriented workflows in a manner which platform independent, customizable and extensible.

In this chapter we introduce the FlowiXML methodology for developing the various UIs of a community-oriented workflow information system (WIS). Traditionally, workflow information systems are designed to be used by different types of users to accomplish a variety of tasks and in different situations; usually include communications and coordination between people and actions of several persons on shared objects and in shared workplaces including the Internet. For several years now, people have been using online virtual spaces to communicate and carry out work-oriented tasks. Prior to the World Wide Web, BBS, or electronic bulletin boards and email loops connected folks across time and space. With the advent of the Web several issues such as ubiquitous access to information, distribution of processes, and platform-independence emerged as first-class design issues.

Workflow information systems are a specified way of working to accomplish a task in a collaborative setting, just as it is the case of a community of practice. Consequently, creating a community workflow involves a level of design above the institutionalized workflows supported by individual members of the community of practice. Our method can be exploited to facilitate user interfaces to shared community-wide workflows within the context of a cross-organizational virtual alliance, thus establishing a shared practice which is interactively manifested through dedicated interaction components.

FlowiXML method generates UIs following a model-driven engineering (MDE) approach that is user-centric, based on the requirements and processes of the community. The methodology seeks to: 1) integrate human and machines based activities, in particular those involving interaction with IT applications and tools, 2) identify how tasks are structured, who perform them, what their relative order is, how they are offered or assigned, and how tasks are being tracked.

In the remaining of the chapter, we provide a background research in understanding the variety of approaches to build workflow information systems, and then we describe a workflow framework and the FlowiXML methodology for building UIs that can be used as well for developing community-oriented workflow information systems. Following this, a case study and a tool supporting the method are presented. The chapter is wrapped up by summarizing our work, deriving conclusions and addressing future trends and challenges.

BACKGROUND

In recent years, there has been a vast interest in how groups of people work together, and in how collaboration and cooperation might be supported. Virtual communities are formed and exploited by a variety of social and professional groups interacting via the Internet. Howard Rheingold (2000) mentions that virtual communities form "*when people carry on public discussions long enough, with sufficient human feeling, to form webs of personal relationships*". A virtual Community is a network of individuals who share a domain of interest about which they communicate online. The participants share resources (for example experiences, problems and solutions, tools, methodologies) and the environment (space).

At the same time there have also been significant advances in information technology (IT) to support group work, normally with the use of an information system (IS). IS provide a technology enabler allowing corporations to gain competitive advantage, by reducing costs, automating processes, timely exchange of information, reducing production time and time-to-market or just simply to keep in business (Kitta, 2007). The users of a IS interact with it through its user interface, which is the aggregate of means by which people (the users) interact with a particular machine, device, computer program or other complex tool (the system).

Recently, the Web had become a privileged platform for implementing workflow systems. The Workflow Management Coalition (1999) defines workflow as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" (p.8). However, there is surprisingly little work emphasizing community-oriented workflows and the way in which such workflows are interactively manifested to community members. Typically, researchers have devoted their efforts to studying and developing tools for community management (i.e., Blogs, Wikis, forums) which come with a pre-packaged workflow. For instance, the workflow through which a user can contribute to a thread in an on-line discussion is fixed and cannot be easily changed. Moreover, the user interface is tightly coupled to this workflow and cannot be modified, tailored or ported to a different execution context. These, however, are shortcomings which may impede participation to the social encounters of the virtual community, especially for novice users or users acquainted with a different way of working.

Designing community-oriented workflows entails several challenges and requires attention to a variety of issues. A non-exhaustive list includes number of participants, productivity, errors in accessing community information or in interaction and variation of enactment. It also brings to the surface social and coordinative considerations such as task, activity and user awareness, synchronization, etc. All these may potentially influence the design of the user interface to a community-workflow. Consequently, a method such as FlowiXML could be exploited to facilitate the design of UIs to shared community-wide workflows of a cross-organizational alliance so as to allow member to engage in a shared practice irrespective of how business is carried out locally by individual members.

Model-Based User Interface Development for Workflows

Model-based user interface design is intended to assist in designing UIs with a more formal computer supported methodology; model-based is concerned with the development of models. A model can be defined as an international and simplified representation of a real-world thing. Model primitives (i.e., model building blocks) are gathered in meta-models i.e., models describing other model's concepts and relationships. Model-based interfaces have recognized advantages in terms of methodology, reusability, and consistency. A

fundamental requirement for a model-based to be operational consists in its relying on a specification language, with which the various models involved in the process could be obtained. Nowadays, eXtensible Mark-up Languages (XML) represent an attractive way to define a concrete syntax from the model Semantics. UsiXML is a XML-compliant mark-up language; it consists of a User Interface Description Language (UIDL) that is a declarative language capturing the essence of what a UI is or should be independently of physical characteristics. It describes at a high level of abstraction the constituting elements of the UI of an application: widgets, controls, containers, modalities, interaction techniques, etc.

Due to the importance of workflow nowadays, several workflow notation descriptions have been proposed to design and specify it, among them:

- **Statechart diagrams.** A Statechart diagram is a graph that represents a state machine describing the response, of an object of a certain class, to the receipt of outside stimuli.
- **Petri nets,** as a modeling language, graphically depict the structure of a distributed system as a directed bipartite graph with annotations. Petri Nets are a technique for modeling and analyzing processes.
- **Business Process Model Notation (BPMN)** is a standardized graphical notation for drawing business processes in a workflow. BPMN will provide a simple means of communicating process information to other business users, process implementers, customers, and suppliers.

Also, there are some workflow languages that represent complex scenarios and can describe the logical presentation. They capture their structure, dynamics and states. Workflow languages are tools that can be used to model a workflow and to execute it, for instance:

- **Yet Another Workflow Language (YAWL)** is a language based on Petri nets. A workflow specification in YAWL is a set of processes definitions that form a hierarchy. Tasks are either atomic task or composed ones. The lower level in the hierarchy refers to a process definition. Atomic tasks form the leaves of the graph structure. Each process definition consists of tasks and conditions which can be interpreted as places. Each process definition has one unique input condition and one unique output condition (van der Aalst, 2005).
- **Exchangeable Routing Language (XRL)** is a language that uses eXtensible Markup Language (XML) for the representation of process definitions and Petri nets for its Semantics. Since XRL is instance-based, workflow definitions can be changed on the fly and sent across organizational boundaries (Verbeek, 2002).

To manage the workflow, many academic and industrial research projects have been developed. The capabilities of these products are being enhanced in significant ways. Some of them are:

- The **Progression model** (Stavness, 2004) has incorporated some of the managing concepts of workflow to increase the flexibility in IS. It makes explicitly the steps and transactions as user undertakes when using an IS. As the user progresses towards accomplishing a task or goal, the progression model infrastructure records each step and the state of the transaction and workflow.
- **Microsoft Windows Workflow Foundation (WWF)** (Kitta, 2007) is an extensible framework for developing workflow solutions on the Windows platform. It provides a single, unified model to create end-to-end solutions that span categories of applications, including human workflow and system workflow.

- **Business Process Visual ARCHITEC (BP-VA)** (Visual Paradigm, 2007) is a visual modeling tool that provides the most extensive support for BPMN.
- **WebSphere® MQ Workflow** (IBM, 2005) supports long-running business process workflows as they interact with systems and people. Automates and tracks business processes in accordance with business design. Provides integration processes with rich support for human interactions.

Due to the great number of workflow products a group of researchers have identified a group of workflow patterns that provide the basis for an in-depth comparison of a number of commercially available workflow systems.

- **Control-flow patterns** identify useful routings construct as sequence, parallel split, synchronization, exclusive choice, etc. From a data perspective, there are a series of characteristics that occur repeatedly in different workflow modeling paradigms (van der Aalst, ter Hofstede, Kiepuszewski & Barros, 2003).
- **Workflow data patterns** aim to capture the various ways in which data is represented and utilized in workflows (Russell, ter Hofstede, Edmond, & van der Aalst, 2004).
- **Workflow resource patterns** correspond to the manner in which tasks are allocated to resources, i.e. an entity that is capable of doing work; the focus of these patterns is on human resources (Russell, van der Aalst, ter Hofstede, & Edmond, 2005).

Workflows and Tasks

Tasks are a fundamental aspect in workflow, a common definition for a task is "an activity performed to reach a certain goal" (van Welie, van der Veer, & Eliëns, 1998); task models play an important role because they indicate the logical activities that an application should support to reach user' goals. In the literature, there are several definitions for task models. Task modeling is "the activity of transforming raw task and user related data or envisioning ideas into structured pieces of task knowledge" (van Welie, van der Veer, & Koster, 2000). While the purpose of task analysis is to understand what tasks should be supported and what are their related attributes, the aim of task modeling is to identify more precisely the relationships among such tasks. Task models are explicit representations of user tasks that can help support certain rigorous forms of task analysis.

In the literature there are different approaches to task models with similarities, (task decomposition, task flow, and graphical representations to show the information of the model) although each task model is designed for a certain purpose. Most of the models have a tool to support the modeling of tasks. For instance, Concur Task Trees (CTT) was developed by Paternò (1999) on five concepts: tasks, objects, actions, operators and roles. CTT constructors, termed as operators, are used to link sibling tasks, on the same level of decomposition. It is also important to note that CTT holds a formal definition for its temporal operators. CTT provides with means to describe cooperative tasks. Tasks are further decomposed up to the level of basic tasks defined as tasks that could not be further decomposed. Actions and objects are specified for each basic task. Objects could be perceivable objects or application objects. Application objects are mapped onto perceivable objects in order to be presented to the user.

An interesting feature of CTT is the specification of both input actions and output actions that are associated to an object. The last modification brought to CTT is the integration of the concept of platform in the method in order to support multi-platform UI development. A task can be associated with one or several previously defined platform descriptions for which it is applicable. Views on the task model are obtained by filtering a task model depending on one or several

platform. CTT uses a tool, CTTE, for building the task model that is used to specify tasks, roles and objects as well as the a task hierarchy with temporal operators.

CONCEPTUAL MODELLING OF WORKFLOW

As indicate above, workflow is the automation of business process, so we need to know the elements that are involved in business process. We propose an ontology called FlowiXML (see Figure 1), where a workflow model is composed of:

- **Process model:** Its goal is to describe the business processes. A process consists of a number of tasks and a set of relationships among them. The definition of a process indicates which tasks must be performed and in what order. The work list allows workflow manager to view and manage the tasks that are assigned to resources.
- **Task model:** Its goal is to describe how the organization works. Task models describe end users' view of interactive tasks while interacting with the system. A task model represents a decomposition of tasks into sub-tasks linked with task relationships. A task model is a quadruple TM (T, H, R, δ) where
  - T is a nonnegative finite set of tasks $\{t_1, t_2, t_3 \ldots t_n\}$
    - $\forall t_i \in T: t_i = (\text{taskname}_i, \text{category}_i, \text{iteration}_i, \text{optional}_i)$ where taskname_i is a label, $\text{category}_i \in$ {manual, interactive, system, mechanic, abstract, cooperative, collaborative, competitive, coopetitive}, $\text{iteration}_i \in \{2, \ldots, n\}$, $\text{optional}_i \in \{0,1\}$
  - H is the hierarchy resulting from the task decomposition: H is a tree i.e. an acyclic simply connected graph: $(\forall t_i, t_j \in T : t_i \text{ dec } t_j) \wedge (\exists (t_1, \ldots, t_m)\ m \geq 3 \dagger \forall p \in \{1,..,m-1\}: t_p \text{ dec } t_{p+1}, t_l = t_m)$. Let us define t_0 as the root task: $t_0 \in T: \exists t_i \in T: t_i \text{ dec } t_0$
  - R is a finite set of temporal operators: R= {enabling, disabling, suspend/resume, order independence, concurrency with information passing, independent concurrency, enabling with information passing, cooperation, inclusive choice, deterministic choice, undeterministic choice, disabling with information passing}
  - δ is a simple transition function between siblings $\delta : T \times R \rightarrow T: t_i \times r \rightarrow t_j$
    - $\forall t_i \in T \dagger \exists! t_{i1}, \ldots t_{in} \in T: t_i \text{ dec } t_{i1}, \ldots, t_i \text{ dec } t_{in}$; $\exists! r_1, \ldots r_{n-1} \in R \dagger t_{i1}\ r_1\ t_{i2}\ r_2\ t_{i3} \ldots r_{n-1}\ t_{in}$
- **Organizational model:** Its goal is to describe the organizational environment. The key elements of an organization are its resources, structure, tasks, politics, culture, etc. An organizational unit is a formal group of resources working together with one or more shared goals or objectives; three types of resources can be found: human resources (i.e. a user stereotype), material resources (e.g., hardware, network, machines), and immaterial resources (e.g., software, operating system). The job concept allows assembling tasks under a same umbrella in a way that is independent of individual resources in the workflow. In this way, several individuals could play a particular job, and jobs could be interchanged dynamically. To organize tasks and resources we use an agenda, it is a list of activities to be taken up. It includes one or more agenda items to carry out.

More details about the attributes and methods of this workflow model could be found in (Guerrero, Vanderdonckt, & Gonzalez, 2008). Figure 1

only represents the UML class diagram without any attributes or methods.

Of a particular interest from a resource perspective is the manner in which tasks are advertised to specific resources for execution. For this purpose, workflow resource patterns (Russell, van der Aalst, ter Hofstede, & Edmond, 2005) have been considered and introduced in a mapping model.

The rationale for identifying these patterns was the need to master the many ways according which work can be distributed. The patterns are grouped into seven categories: creation patterns, push patterns, pull patterns, detour patterns, autostart patterns, visibility patterns, and multiple resource patterns. The researchers have developed a Web site (http://www. workflowpatterns. com/patterns/resource/) that contains descriptions and examples of theses patterns, along with supporting papers and evaluations of how workflow products support the patterns.

Three modeling levels have been proposed: workflow, process and task. Therefore, it is necessary to clearly specify when and where each model starts and finishes. Since a task is defined as an operation executed while four dimensions remain constant (i.e., time, space, resources, information), any variation of any of these four dimensions, taken alone or combined, thus generate a potential identification of a new task in the task modeling activity. In Table 1 we propose a set of parameters to identify a workflow, a process or a task.

Once the identification criteria have been established, from our conceptual model it is possible to:

- Identify processes and tasks; thereby specify *how we will do the work?*
- Identify organizational units to know *where we will do it?*

Figure 1. Partial view of the workflow model

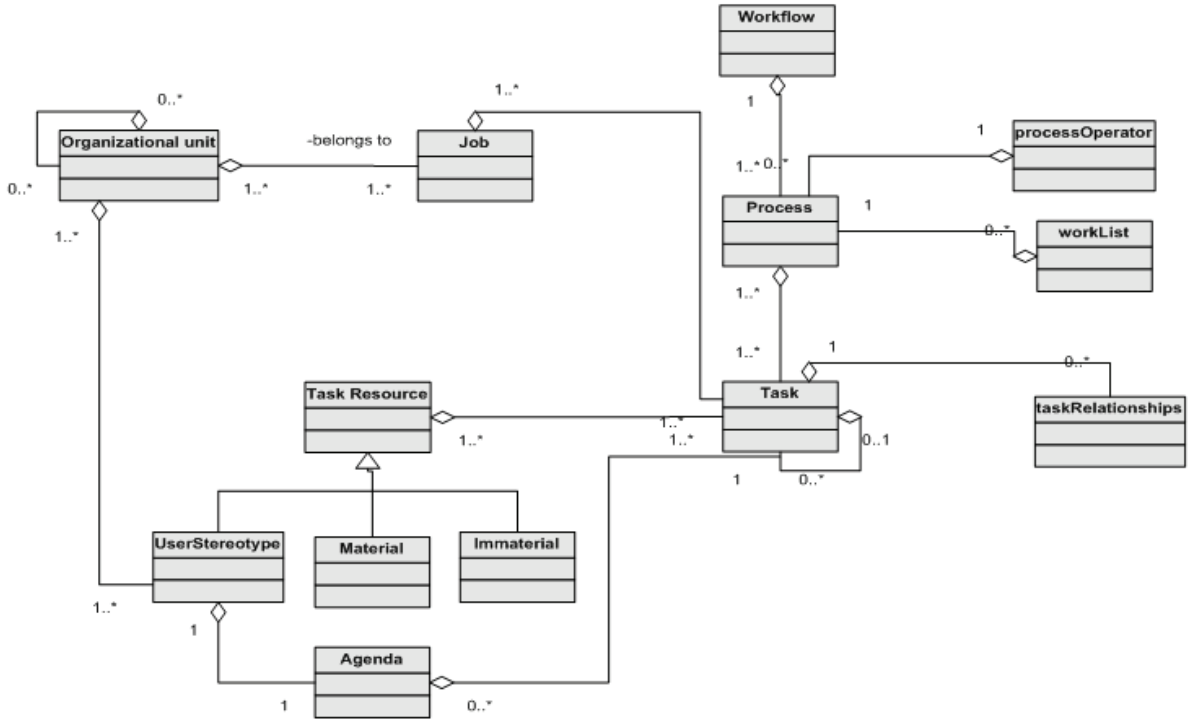


Table 1. Identification criteria

Workflow	Series of time periods	Different locations; same organization	Same or different groups of resources	-
Process	Series of time periods	Different locations	Within groups, group as a whole, or among groups	Primary (production), secondary (support), or tertiary (managerial)
Task	Same time period	Same location	One or two types of resources	User, interactive, system, abstract, or machine task

- Identify jobs and resources available, and the way to assign tasks to them, so we establish *who will do it?*

Considering that resources are not alone, that are part of an organization, software engineering must guide the development of applications to support working groups, in order to cover some aspects such as collaboration, communication, coordination, information sharing, etc., which are part of the work group activities. Some group requirements should be taking into account: (1) 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 the organization, and among organizations, (2) Support multiple ways to carry out a group task: in principle, there should not be a 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 selectable, (3) Support the group evolution over time: when the group evolves over time, the workflow definition should be easily maintained and reflected in the system, (4) Provide multiple ways of interaction: group members need multiple interaction methods such as electronic mail, audio, written, verbal, and visual, (5) Sustain several behavioral characteristics: the reaction of a group is always difficult to analyze, the dynamism of a group can be chaotic (Mandviwalla & Olfman, 1994).

A virtual community is a social network with a common interest, idea, task or goal that interacts in a virtual society across time, geographical and organizational boundaries and is able to develop personal relationships. The Web had become a privileged platform for implementing workflow systems. The Web provides ubiquitous access to information, supports inherent distribution of business process, and consists of platform-independent UIs. A Web community is simply a community that happens to exist online, rather than in the physical world (Kim, 2000).

Taking advantage of this type of communities, it is possible to coordinate work, improve remote communication, keep inform and formed all the members of the group, facilitate problems resolution in the group; despite of some inconvenient as to differentiate who is working for the group and who is been benefited from the group or the resistance to change from resources.

Creating an information system to support work group and that evolves with the organization involves the design of UIs with appropriate widgets to cover the requirements of each group member.

METHOD FOR DEVELOPING THE UI OF A WORKFLOW SYSTEM

To design UIs it is necessary a User Interface Description Language (UIDL), which consists of a high-level computer language for describing

characteristics of interest of a UI with respect to the rest of an interactive application; it helps define user interfaces linguistically with a general trend to do so in an XML-complaint way. Many UIDLs have been conceived that contain different features and focus on different levels of granularity; also, a language definition consists of three components: Semantic, syntax, and stylistic. The USer Interface eXtensible Mark-up Language (UsiXML) (Limbourg & Vanderdonckt, 2004; Vanderdonckt, 2005) has been selected as the UIDL to be used in the remainder of this work, because of its capabilities of extensiveness, availability, central storage of models, and its transformational approach.

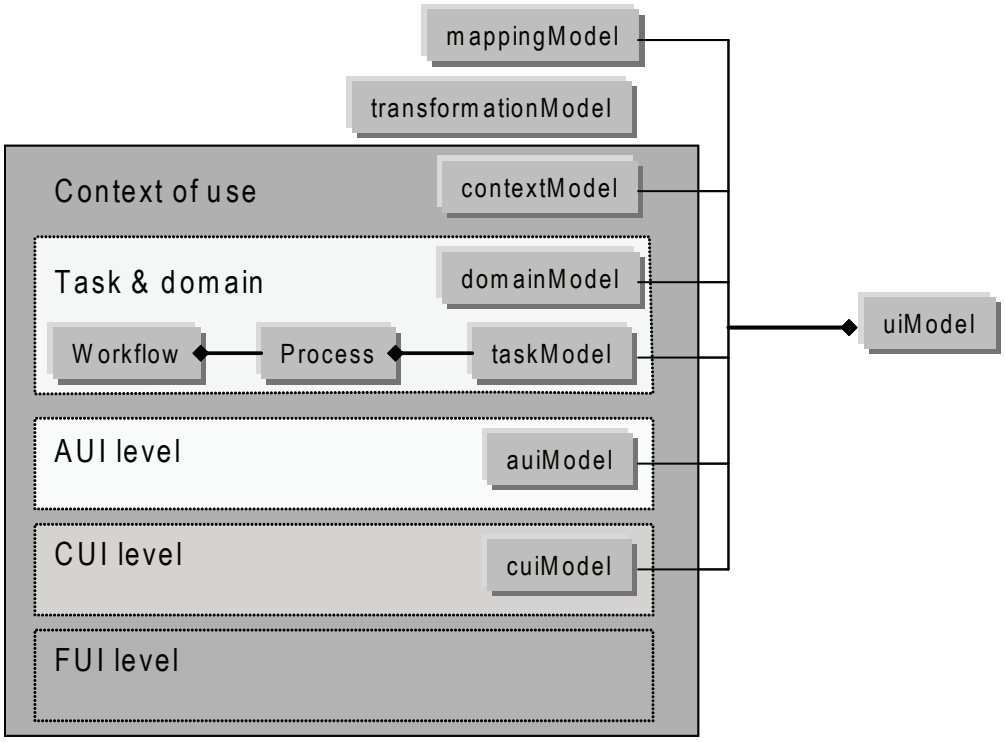
UsiXML is explicitly based on the Cameleon Reference Framework (Calvary, Coutaz, Thevenin, Limbourg, Bouillon, & Vanderdonckt, 2003). Its simplified version, reproduced in Figure 2, structures development processes for contexts of use into four development steps:

- **Task & Concepts (T&C):** Describe the various user's tasks to be carried out and the domain-oriented concepts as they are required by these tasks to be performed.
- **Abstract UI (AUI):** Defines abstract containers and individual components, two forms of Abstract Interaction Objects by grouping subtasks according to various criteria, a navigation scheme between the containers and selects abstract individual component for each concept so that they are independent of any modality. An AUI abstracts a CUI into a UI definition that is independent of any modality of interaction (e.g., graphical interaction, vocal interaction, speech synthesis and recognition, video-based interaction, virtual, augmented or mixed reality). An AUI can also be considered as a canonical expression of the rendering of the domain concepts and tasks in a way that is independent from any modality of interaction. An AUI is considered as an abstraction of a CUI with respect to interaction modality. At this level, the UI mainly consists of input/output definitions, along with actions that need to be performed on this information.
- **Concrete UI (CUI):** Concretizes an abstract UI for a given context of use into Concrete Interaction Objects (CIOs) so as to define widgets layout and interface navigation. It abstracts a final UI into a UI definition that is independent of any computing platform. Although a CUI makes explicit the final Look & Feel of a final UI, it is still a mock-up that runs only within a particular environment. A CUI can also be considered as a reification of an AUI at the upper level and an abstraction of the final UI with respect to the platform.
- **Final UI (FUI):** Is the operational UI i.e. any UI running on a particular computing platform either by interpretation or by execution.

UsiXML is a collection of models (small rectangles in Figure 2) for specifying a UI:

- **taskModel**: Is a model describing the interactive task as viewed by the end user interacting with the system.
- **domainModel**: Is a description of the classes of objects manipulated by a user while interacting with a system.
- **mappingModel**: Is a model containing a series of related mappings between models or elements of models.
- **transformationModel**: Graph Transformation (GT) techniques were chosen to formalize explicit transformations between any pair of models, except from the FUI level.
- **contextModel**: Is a model describing the three aspects of a context of use in which a end user is carrying out an interactive task with a specific computing platform in a given surrounding environment. Conse-

Figure 2. The simplified Cameleon Reference Framework and UsiXML



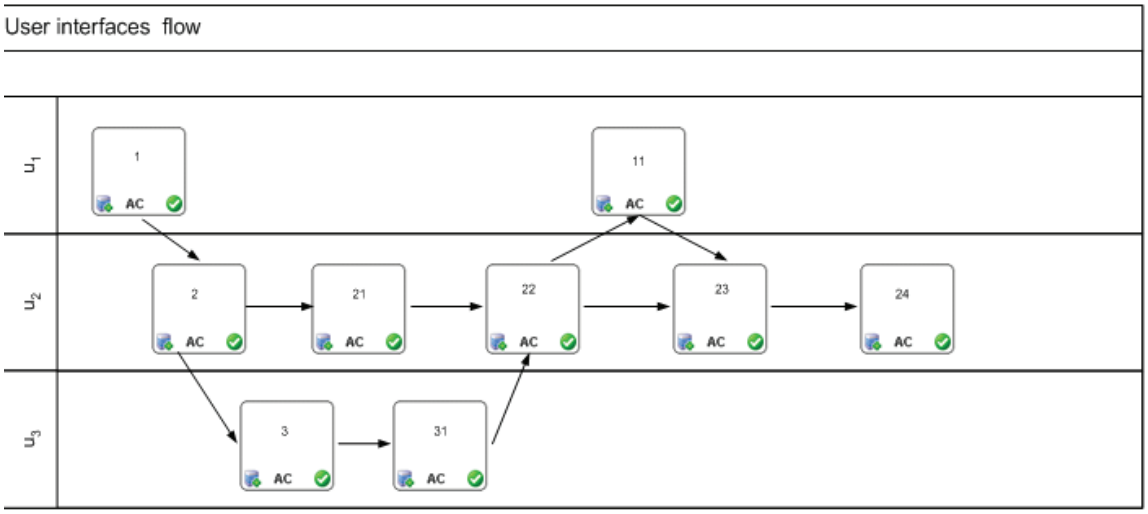
quently, a context model consists of a *user model*, a *platform model*, and an *environment model*.

- **auiModel**: Is the model describing the UI at the abstract level as previously defined.
- **cuiModel**: Is the model describing the UI at the concrete level as previously defined.
- **uiModel**: Is the topmost super class containing common features shared by all component models of a UI.

Continuing with the language definition one can say that *syntax* deals solely with the form and structure of symbols in a language without any consideration given to their meaning. The *abstract syntax* is defined as the hidden structure of a language, its mathematical background. FlowiXML uses *directed graph* as abstract syntax. A *concrete syntax* is an external appearance; the *visual syntax* consists of boxes and arrows, a somewhat classic representation for a graphical structure. This visual syntax will be mainly used to in this work as an expression means for the transformation rules that are going to be developed in a future. The *textual syntax* is described using an XML-based language. The objective of *stylistics* is to provide a representation of a set of defined objects in order to facilitate their understanding and manipulation in tools. The representation can be of different types (e.g., graphical, textual); this representation is reflected in a workflow editor tool.

After having defined the UIs involved in the workflow, we need now to link all the UIs: the ones for the workflow management and the ones for the workflow tasks. This will be achieved thanks to the *user Interface flow* (see Figure 3). During the execution of work, information passes from one resource to another as tasks are finished or delegated; in FlowiXML we use an agenda assigned to each resource to manage the tasks

Figure 3. User interfaces flow



that are allocated/offered to her/him, and a work list that allows to workflow manager views and manages the tasks that are assigned to resources. By linking UIs we expect to solve the problem of synchronizing the communication among them. The flow of User Interfaces is an octuple UIF (A, Σ, U, T, δ, ω, ai, ao) where

- A is a nonnegative finite set of Abstract Containers (AC)
- Σ is a set of input events [set of events occurring in AC]
- U is a nonnegative set of user stereotypes, such that ∀ a ∈A:∃! u ∈U † is used by (a,u) [unique] or ∃ u_1, u_2… u_n ∈U † is used by {a, u_1, u_2… u_n} [a is shared among u_1, u_2… u_n]
- T is a set of output transitions [output transitions means a navigation from starting AC to a final one, we do not want to commit ourselves to a particular type or representation]
- δ is a transition function, δ : A x Σ → A [a transition is AC + abstract event occurring in one AC]
- ω is an output function, ω : A → T
- ai is the initial AC [ai ∈ A]
- ao is the final AC [ao ∈ A, ao ≠ ai]

Hypothesis 1: *The target AC after a transition is activated.*

CASE STUDY AND TOOL SUPPORT

Considering that virtual communities are formed and exploited by a variety of social and professional groups interacting via the Internet, we want to illustrate how the methodology proposed above allows a research group to collaborate using a blog on a Web site.

Tool support: In order to support the development of UIs from a workflow model to a task model, a workflow editor has been developed. This editor allows modeling the general workflow defining processes and tasks models, defining organizational units, jobs and resources involved, allocation of tasks to resources, and to manage the flow of tasks.

Case study: A research group, working in the same university but in different departments, needs

to keep in touch in order to change information due to they are working in a specific project to be submitted to get founding. A good option to help them is creating a collaborative blog (a Web site where publishes posts are written by multiple users, called co-bloggers). Thus, it is necessary to create a blog, to select the users, to invite them to participate and collaborate through the blog posting, adding comments or just reading publishes posts and comments.

We might start to specify what we want to do:

- **What?** *Workflow specification*: the workflow specification, depicted in the process model, takes place inside the organizational units' framework. This part of the graphical notation of the workflow is based on Petri nets notation (van der Aalst, 1998). Process definition gives the paths that may be followed by a particular item through the set of tasks. In our example (Figure 4), the first step is to *log in*, and then registered users can *create a blog*, and/or *edit* an existing blog. Once the blog has been created, co-bloggers need to log in to edit the blog. Also it is possible to create another blog. However, thanks to the post-condition attribute of the task log in, it is mandatory that the blog exists to be edited.
- **How?** *Task models specification*: for each task a task model is specified to describe in detail *how* the task is performed. Task models do not impose any particular implementation so that user tasks can be better analyzed without implementation constraints.

In order to accomplish the task *log in*, it is necessary to execute a series of steps (Figure 5) which are: to *register* to the Web site, there are two options: type user's account directly if there is already an account or to provided personal information to get a new account. After the system *verifies* the *information* received, in case information is wrong the system *sends a message* to notify the user. Finally, the user could continue with the next task or not.

Then, the user proceeds to create the blog (Figure 6). The first step it to *assign a name* for the blog, after *select the URL*, then *select a template*, finally the system validates and provides feedback to the user, who *receives a message* with the results of its creation.

Once the blog has been created, it is possible to edit it (Figure 7). In this part of the design users *create an entry*, *edit a created entry*, *con-*

Figure 4. Workflow to create a blog

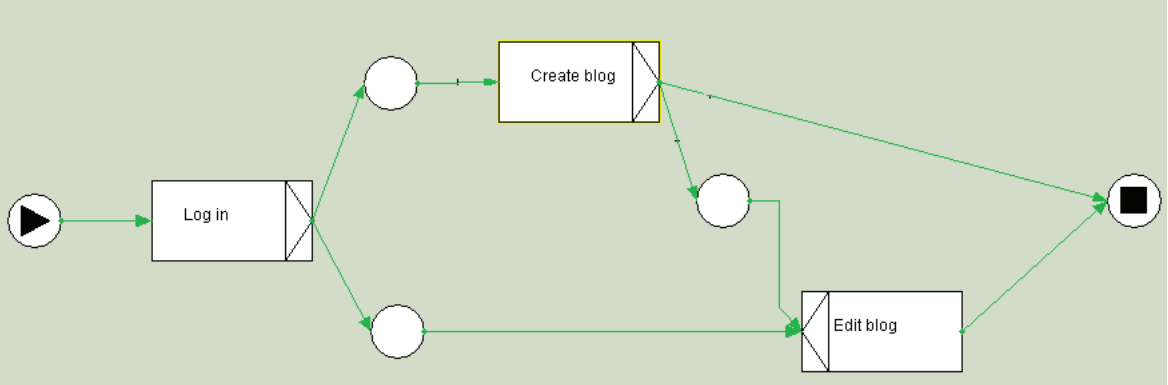


Figure 5. Log in task model

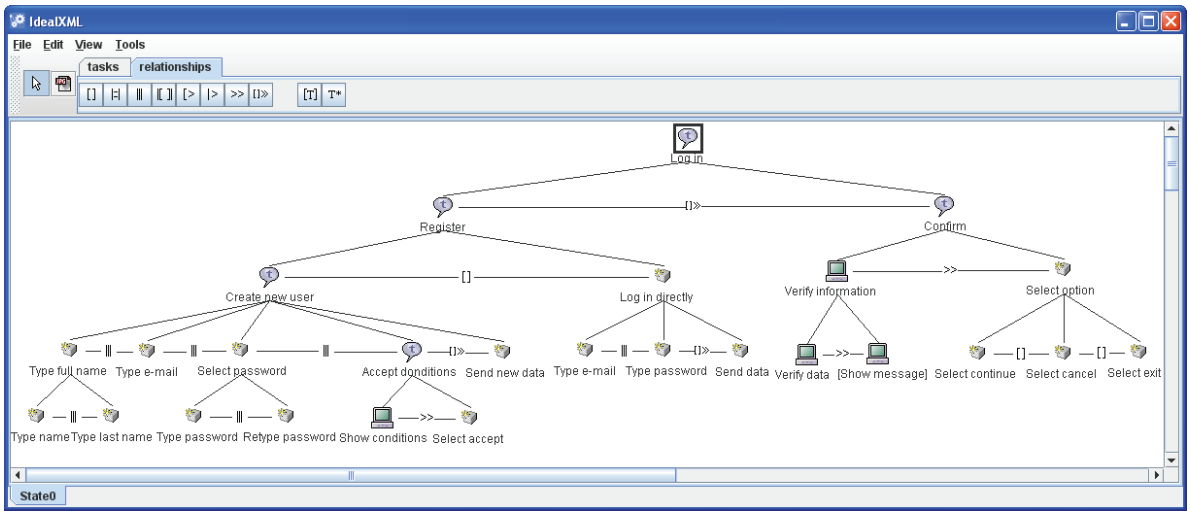


Figure 6. Create blog task model

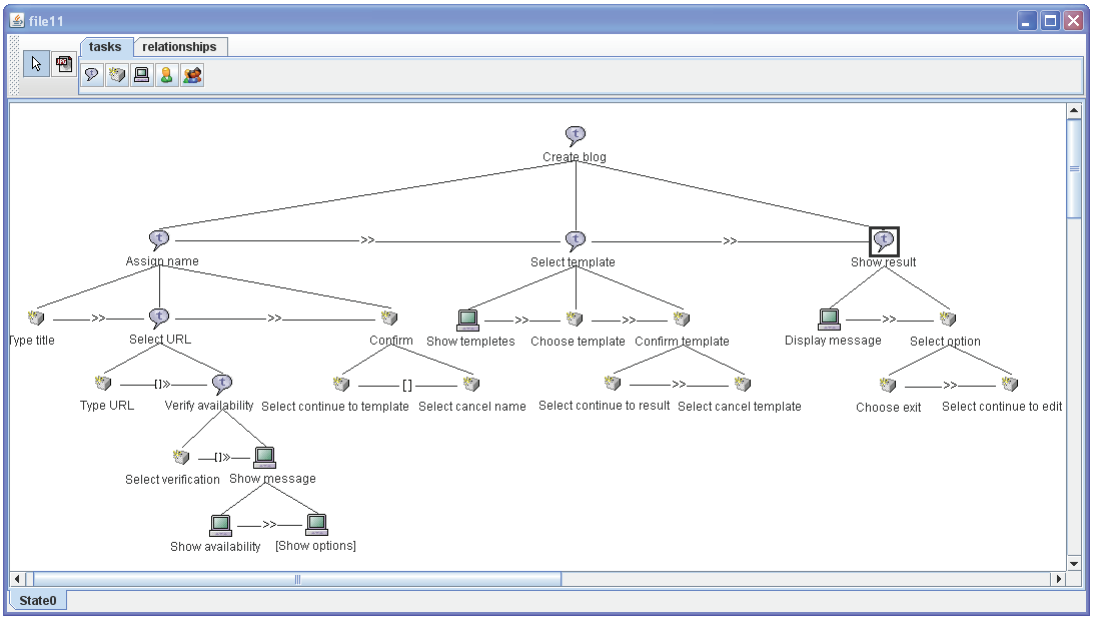


Figure 7. Edit blog task model

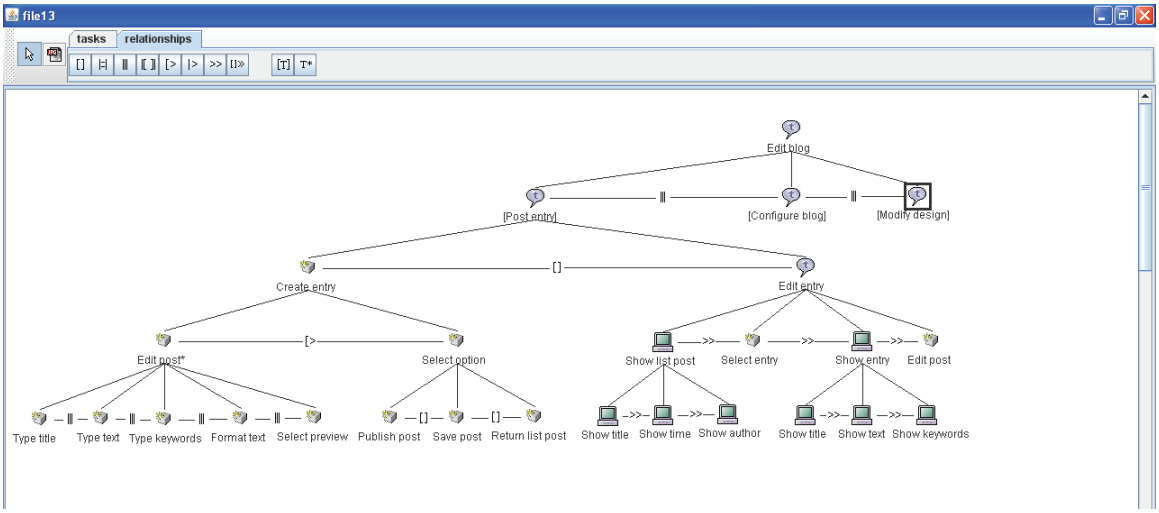


Figure 8. Configure basic elements task model

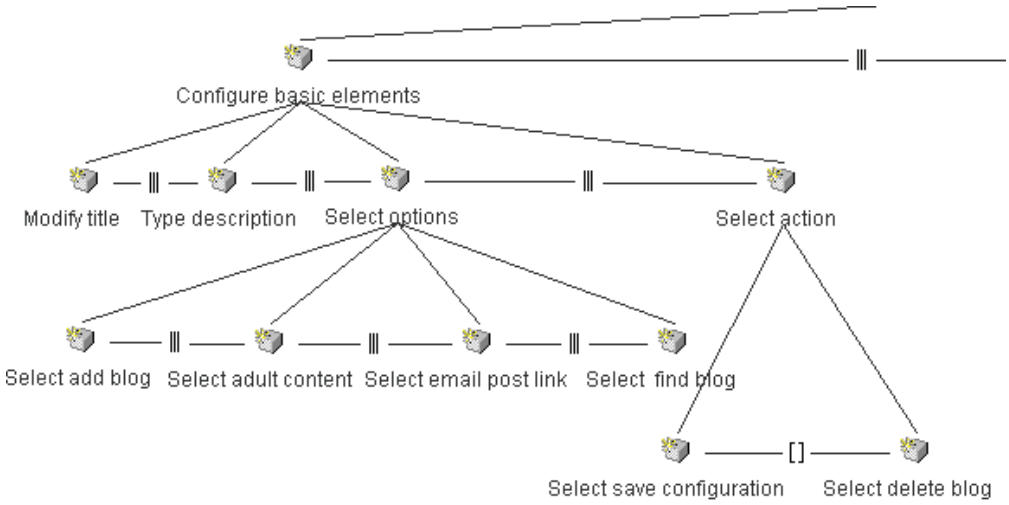


figure and modify the blog. To create an entry, it is advisable to *assign a title*, to *type the text*, to *format* it (e.g. highlight some words), *specify keywords*, *have a pre-view* of the entry, *save* the entry. Once the entry was published, it is feasible to edit it in order to make corrections, or add new information, etc.

It is important to specify that as an attribute of the task *edit entry*, it has as a precondition that an entry was created.

As part of *blog configuration* (second task in Figure 7), it is possible to *modify title* of the blog, *write* a brief *blog description*, specify if the blog will be added to a list of commercial blogs, specify if the content is just for adults, specify if the email post links let visitors easily email posts from your blog to their friends, specify if the blog will be included in blog search engines, or *delete* the blog (Figure 8).

Figure 9. Configure comments task model

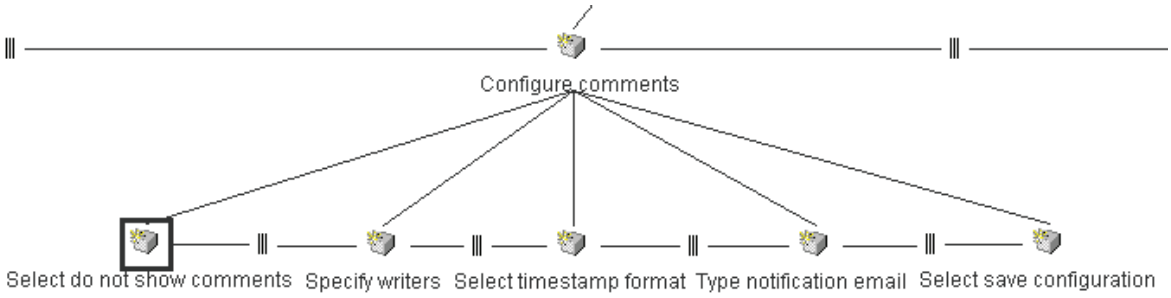


Figure 10. Configure email task model

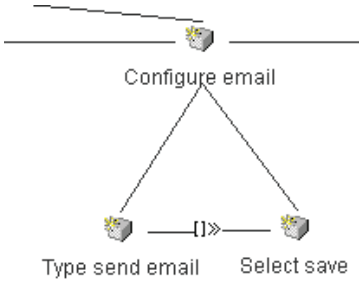
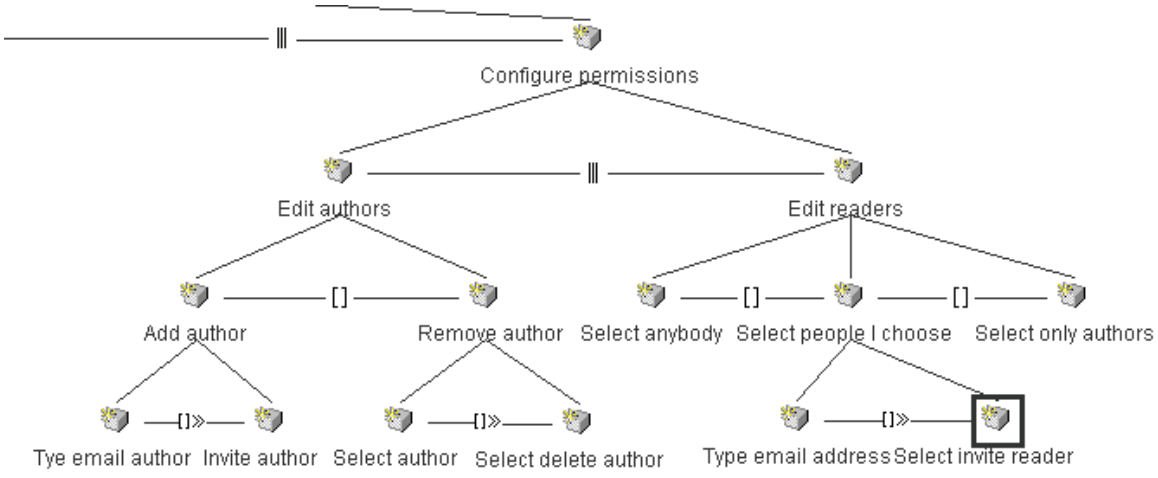


Figure 11. Configure permissions task model



Also one significant aspect is to configure the comments (Figure 9), in this case they can be visualized or not and, most important, it is possible specify who can add a comment and how the users can know that a comment was added thanks to the notification by email.

In addition, it is relevant notify to the users each time that a new posting was made (Figure 10), so in this case different email addresses are specified.

Furthermore, it is necessary to configure the permissions of the blog (Figure 11), thus specifying which users will be playing the job of co-blogger (write and read entries and comments) and readers (read entries and write comments or just read).

Now, as part of *modify the blog design* (third task in Figure 7), it refers principally to the selection of fonts and colors, and the possibility to change the template.

- **Where?** : Once the specification of what we want to do and how it will be done, we continue with the workflow model, so the next step is defining where the work will be execute, in this case the participants belong to different departments in the same university, thus we can say that in each department involved the users can be accessing to the blog. However, we are talking about a virtual space where the communication and collaboration will be taken place.
- **Who?** *Specification of jobs and users*: this step consists on describing who will be involved in the performance of tasks. Jobs are ways to structure the crew of people inside the organization. It involves the complete collection of knowledge and practices needed by a definite human resource to perform a task. The jobs specified in the definition of the current case study are: a blog designer, a blog manager, authors and users.

Once jobs are defined it is possible to incorporate workers able to carry out tasks of a particular job. Workers are defined in terms of attributes (name, experience, hierarchy level) and the list of jobs they can perform. Once the work and workers were added to the workflow, they can be linked with the tasks.

- **Whom?** *Assigning tasks to resources*: one characteristic of workflow is to determine the right person for the right task at the right moment; for that purpose, we use workflow resource patterns (Russell, van der Aalst, ter Hofstede, & Edmond, 2005) to specify who will perform tasks realization inside organizational units. We already determine the range of resources available in the different organizational units. Now, we go further by adding rules defining the way work will be undertaken. For each process in the workflow model it is possible to define one or several allocation or offering relationships: Distribution, Managing, Deviation, Auto-start, Visibility, or Multiple resources. In our case study, the task *create blog* is allocated to *Steve Geller* who has experience as *blog designer*.

As a result of these steps, we can visualize the complete workflow in our editor (Figure 14).

Generating UIs, *from task model to UI*:This step is achieved by relying on the UsiXML method that progressively moves from a task model to a final user interface (Figure 15). This approach consists of three steps: deriving abstract user interfaces from a task model, deriving concrete user interfaces from each abstract one, and producing the code of the corresponding final user interfaces. To ensure these steps, transformations are encoded as graph transformations performed on the involved models expressed in their graph

Figure 12. Specification of jobs and users in the workflow

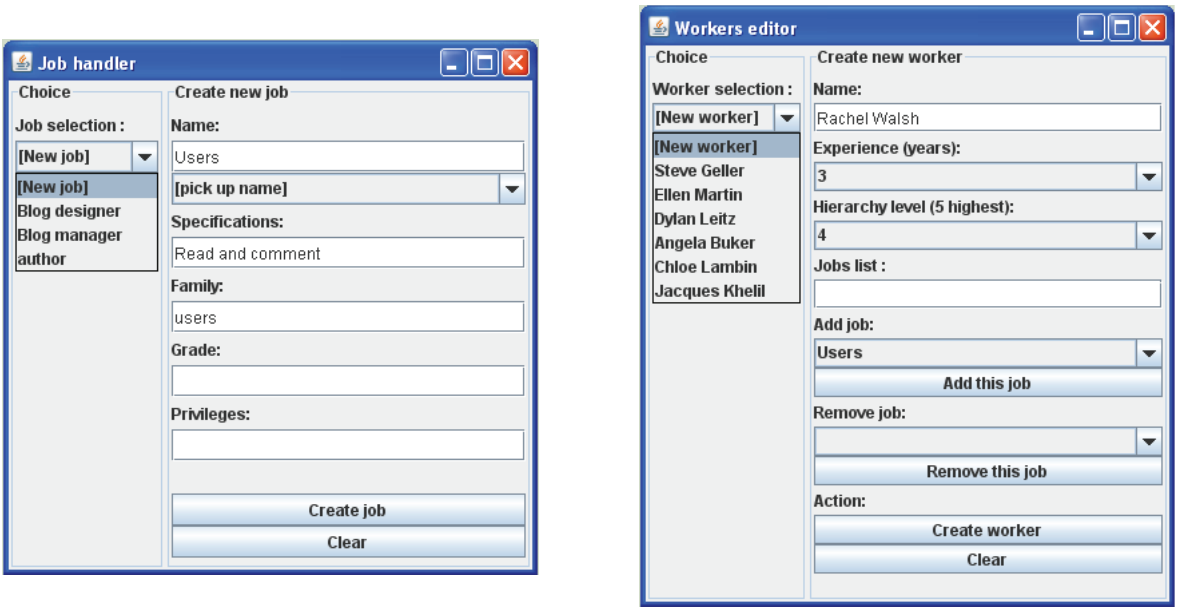
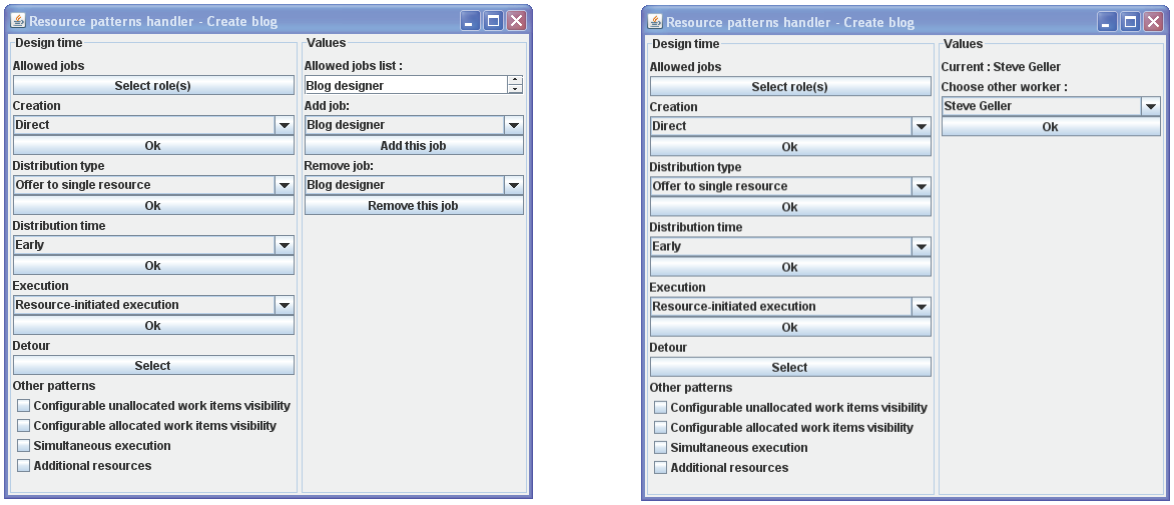


Figure 13. Assigning tasks to resources in the workflow

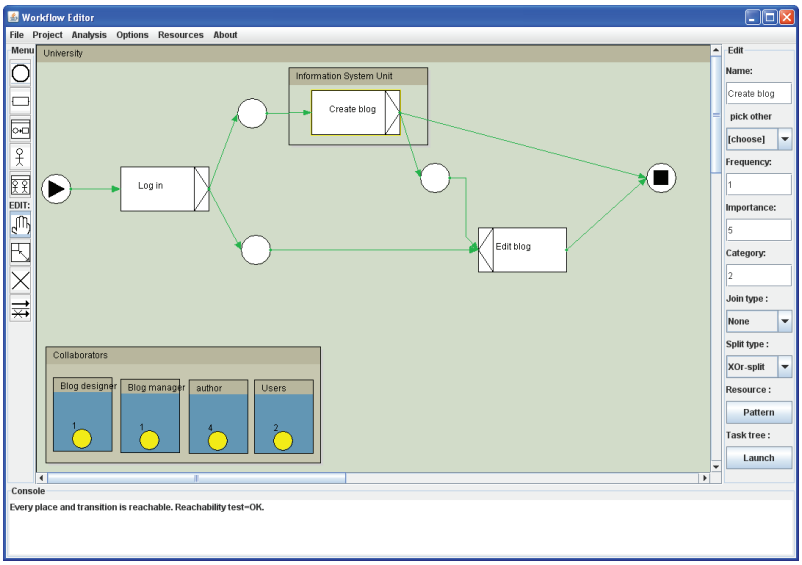


equivalent. For each step, a graph grammar gathers relevant graph transformations for accomplishing the sub-steps. For instance, applying this method to the task model *Log in* we obtain its correspondent UI. Figure shows the four levels to develop UIs (Figure 2).

Then by analogy we can obtain the complete set of UIs corresponding to a workflow model.

Selecting the option to send emails to users we can assure the notification of new posts/comments that are published on the blog.

Figure 14. Workflow editor



FUTURE TRENDS

Until recently, workflow information systems, as well as their supporting UIs, have been condemned to stay pre-defined and fixed along most dimensions. In the near future, we expect that these constraints will be relaxed progressively so as to give rise to a new series of open questions including, but not limited to:

- **User variation:** Not only a worker could evolve dynamically over time, after all human being is very adaptable, but also a worker could be replaced by another for the same job as people are asked to become more flexible in their job positions. Consequently, a workflow UI should accommodation multiple user stereotypes over time, but also variations of a given user stereotype over time.
- **Task variation:** Most of the time, the processes and their underlying tasks are defined at design time, thus preventing the workflow from supporting new tasks or evolving tasks, unless the engineer applies the required modification. Moreover, if there is needs for incorporating dynamic tasks that are know only at run-time, perhaps without the support of the workflow manager, appears a need for end-user definition of a new task.
- **Workplace variation:** As workers are requested to become more and more mobile, they are confronted with tasks and processes that are no longer executed in their traditional stationary contexts of use, but in new contexts. These tasks and processes are then redefined depending on the workplace where they are achieved. Consequently, the supporting UIs should support this variation. Task migration or redistribution is likely to appear as well. A UI could for instance be decomposed into smaller pieces, some of them being detached from the main UI and attached to another workplace, even if the main system remains at the same workplace.
- **Platform variation:** As a consequence of the previous variation or independently of that, the worker may want to use another computing platform at run-time because it

Figure 15. Generating UIs

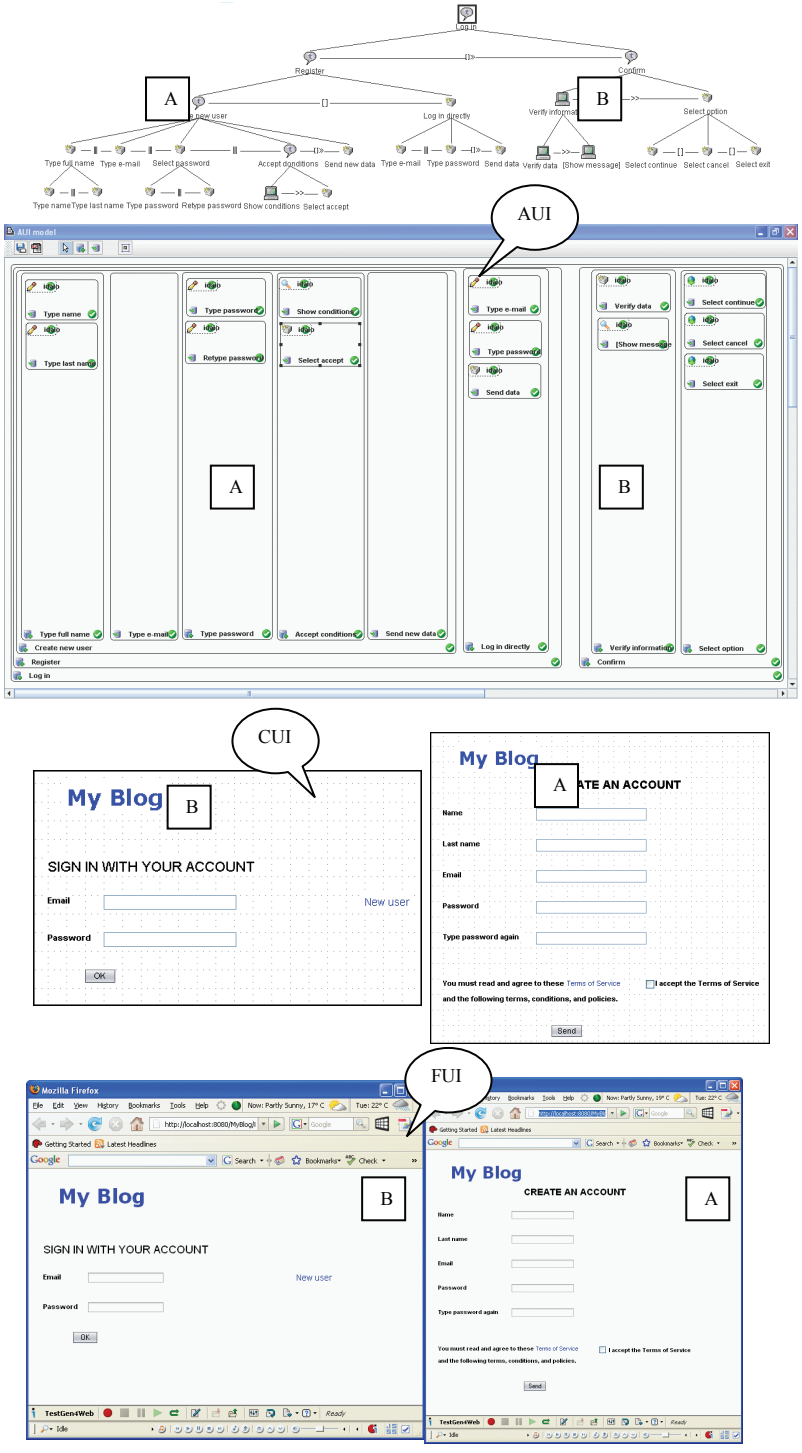
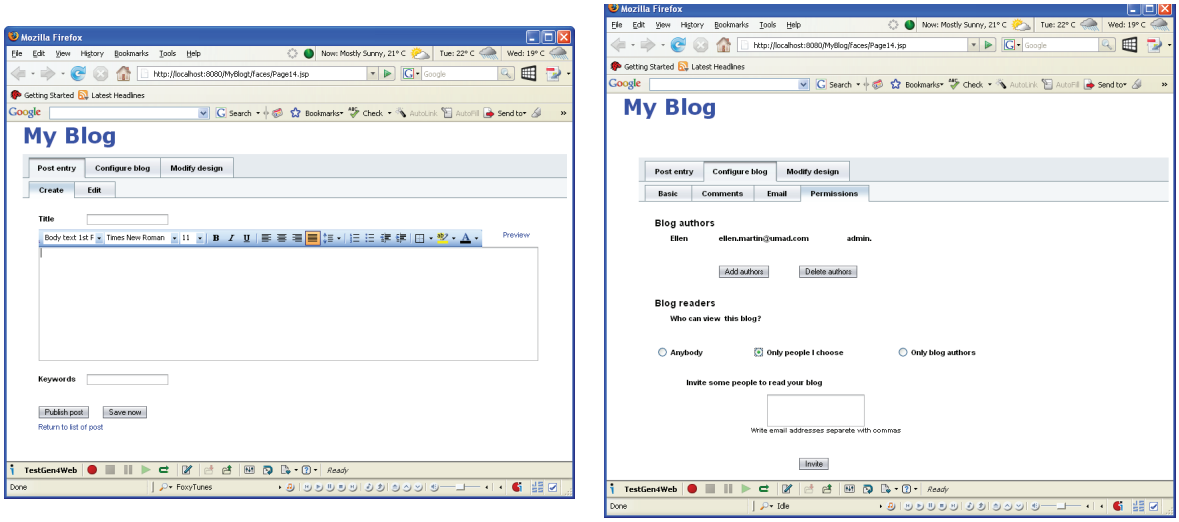


Figure 16. Some final user interfaces



is more convenient for her. Therefore, the supporting UIs should support the same task on different computing platforms and accommodate their variations.

CONCLUSION

Virtual communities are formed around different disciplines to bring together people sharing a common interest, concern or desire. One important aspect to consider is the design of the user interfaces of the system that will be used by the community to foster collaboration. Ideally, such a design should be freed from the specificities of platforms, access terminals and local workflow bindings. To this effect, it is important to generate user interface software for community-oriented workflows in a manner which platform independent, customizable and extensible.

In this chapter we have introduced a model-driven engineering method that provides designers with methodological guidance on how to systematically derive user interfaces of workflow information systems from a model of workflow, which is decomposed into processes to end up with tasks. Based on workflow patterns, it is possible to model an entire workflow with high-level mechanisms and automatically generate the workflow specifications and their corresponding UIs.

All models are uniformly expressed in the same XML-based specification language so that mappings between models are preserved at design-time and can be exploited at run-time in needed. Then, the different *steps* of the approach have been properly defined based on the underlying models and a *workflow editor-manager tool* has been developed to support the method enactment. The major benefit of the above method is that all the design knowledge required to progressively move from a workflow specification to its corresponding UIs is expressed in the model and the mapping rules. The method preserves continuity (all subsequent models are derived from previous ones) and traceability of its enactment (it is possible to trace how a particular workflow is decomposed into processes and tasks, with their corresponding user interfaces). In this way, it is possible to change any level (workflow, process, task, and UI) and to propagate the changes throughout the other levels by navigating through the mappings established at design time.

This method has been so far validated on 4 real-world case studies (e.g., a hospital dept., a triathlon organization, a cycling event, and personalized order of compression stockings over Internet).

RESOURCES

All resources related to this workflow UI development method can be found at: http://www.usixml.org/index.php?mod=pages&id=40. On this Web page, the FlowiXML software can be downloaded, along with its user's manual, and case studies with examples. A video demonstrating the system could be also downloaded.

ACKNOWLEDGMENT

We gratefully acknowledge the support of the CONACYT program (www.conacyt.mx) supported by the Mexican government and the SIMILAR network of excellence (http://www.similar.cc), the European research task force creating human-machine interfaces similar to human-human communication of the European Sixth Framework Programme (FP6-2002-IST1-507609).

REFERENCES

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A Unifying reference framework for multi-target user interfaces. *Interacting with Computers*, *15*(3), 289-308.

Guerrero, J., Vanderdonckt, J., & Gonzalez, J. M. (2008). FlowiXML: A step towards designing workflow management systems. *International Journal of Web Engineering and Technology*, *4*(2), 163-182.

Guerrero, J., Vanderdonckt, J., Gonzalez, J. M., & Winckler, M. (2008, March). Modeling user interfaces to workflow information systems. Paper presented in *4th International Conference on Autonomic and Autonomous Systems ICAS'2008*, Gosier, Guadeloupe.

IBM. (2005). *IBM WebSphere MQ Workflow V3.6 expands support for large enterprise workflow integration solutions*. Retrieved April 19, 2005, from http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&appname=GPA&htmlfid=897/ENUS205-096

Kim, A. J. (2000). *Community building on the Web: Secret strategies for successful online communities*. Peach Pit Press. From http://www.naima.com/community

Kitta, T. (2007). *Professional windows workflow foundation*. Indianapolis, IN: Wiley Publishing, Inc.

Limbourg, Q., & Vanderdonckt, J. (2004). UsiXML: A user interface description language supporting multiple levels of independence. In M. Matera & S. Comai, (Eds.), *Engineering advanced Web applications*, (pp. 325-338). Paramus, NJ: Rinton Press.

Mandviwalla, M., & Olfman, L. (1994). What do groups need? A proposed set of generic groupware requirements. *ACM Transactions on Computer-Human Interaction*, *1*(3), 245-268.

Paternò, F. (1999). *Model based design and evaluation of interactive applications*. Berlin: Springer Verlag.

Rheingold, H. (2000). *The virtual community: Homesteading on the electronic frontier.* London: MIT Press.

Russell, N., ter Hofstede, A. H. M., Edmond, D., & van der Aalst, W. M. P. (2004). *Workflow data patterns* (Tech. Rep. FIT-TR-2004-01). Brisbane, Australia: Queensland University of Technology.

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., & Edmond, D. (2005). Workflow resource patterns. In O. Pastor & J. Falcao e Cunha, (Eds.), *17th Conference on Advanced Information Systems Engineering (CAiSE'05): Vol. 3520 of Lecture Notes in Computer Science* (pp. 216-232). Berlin: Springer-Verlag.

Stavness, N. (2005) *Supporting flexible workflow process with a progression model*. Unpublished master's thesis. University of Saskatchewan, Saskatoon.

van der Aalst, W. M. P. (1998). The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, *8*(1), 21-66.

van der Aalst W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns**.** *Distributed and Parallel Databases*, *14*(3), 5-51.

van der Aalst, W. M. P., & ter Hofstede, A. H. M. (2005). YAWL: Yet Another Workflow Language. *Information Systems, 30*(4), 245-275

Vanderdonckt, J. (2005). A MDA-compliant environment for developing user interfaces of information systems. In O. Pastor & J. Falcao & Cunha, (Eds.), *17th Conference on Advanced Information Systems Engineering (CAiSE'05): Vol. 3520 of Lecture Notes in Computer Science* (pp. 16-31). Berlin: Springer-Verlag.

van Welie M., van der Veer, G. C., & Eliëns, A. (1998). An ontology for task world models. In *Proceeding of the 5th International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems* DSV-IS98 (pp. 57-70). Abingdon, UK.

van Welie, M., van der Veer, G. C., & Koster, A. (2000). Integrated representations for task modeling. In *Proceedings of the Tenth European Conference on Cognitive Ergonomics* (pp. 129-138) . Linköping, Sweden.

Verbeek, H. M. W., Hirnschall, A., & van der Aalst, W. M. P. (2002, May). *XRL/Flower: Supporting inter-organizational workflows using XML/petri-net technology*. Paper presented at *Web Services, e-Business, and the Semantic Web (WES): Foundations, Models, Architecture, Engineering and Applications, held in conjunction with CAiSE 2002* [The Fourteenth International Conference on Advanced Information Systems Engineering], Toronto, Ontario, Canada.

Visual Paradigm International Ltd. (2007). *Business Process Visual ARCHITECT, User's guide*. Retrieved January 2, 2007, from www.visual-paradigm.com/product/bpva/bpvadocuments.jsp

Workflow Management Coalition (1999). *Workflow management coalition terminology & glossary*. Document Number WFMC-TC-1011. Document Status – Issue 3.0.

KEY TERMS & DEFINITIONS

Aui Model: It is the model describing the UI at the abstract level.

Collaborative Blog: Also known as a group blog. It is a web log (blog) written by multiple people and based on a single unifying theme.

Context Model: It is a model describing the three aspects of a context of use in which a end user is carrying out an interactive task with a specific computing platform in a given surrounding environment. Consequently, a context model consists of a user model, a platform model, and an environment model.

Cui Model: Is the model describing the UI at the concrete level.

Domain Model: It describes the real-world concepts, and their interactions as understood by users and the operations that are possible on these concepts.

FlowiXML: It is a methodology for developing the various UI of a Workflow Information System, which are advocated to automate processes, following a model-centric approach based on the requirements and processes of the organization.

Mapping Model: It is a model containing a series of related mappings between models or elements of models. A mapping model serves to gather a set of intermodel relationships that are Semantically related. It expresses reification, abstraction, and translation.

Petri Net: Also known as a place/transition net or P/T net, is a directed bipartite graph, in which the nodes represent transitions (i.e. discrete events that may occur), places (i.e. conditions), and directed arcs (that describe which places are pre- and/or post conditions for which transitions).

Process: It is a collection of tasks linked by relationships. The definition of a process indicate which tasks must be performed and in what order.

Task: An activity performed to reach a certain goal.

Task Model: It is a model describing the interactive task as viewed by the end user interacting with the system. A task model represents a decomposition of tasks into sub-tasks linked with task relationships. Therefore, the decomposition relationship is the privileged relationship to express this hierarchy, while temporal relationships express the temporal constraints between sub-tasks of a same parent task.

Ui Model: It is the topmost superclass containing common features shared by all component models of a UI. A uiModel may consist of a list of component model sin any order and any number, such as task model, a domain model, an abstract UI model, a concrete UI model, mapping model, and context model. A user interface model needs not include one of each model component. Moreover, there may be more than one of a particular kind of model component.

UsiXML: It stands for USer Interface eXtensible Markup Language, a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces. In other words, interactive applications with different types of interaction techniques, modalities of use, and computing platforms can be described in a way that preserves the design independently from peculiar characteristics of physical computing platform.

User Interface (UI): The user interface is the aggregate of means by which people (the users) interact with a particular machine, device, computer program or other complex tool (the system).

Workflow: It is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

Workflow Information Systems (WIS): It refers to the application of information technology to business problems. Its primary characteristic is the automation of process involving combinations of human activities with information technology applications.