

Chapter 2

Model-Driven Engineering of Workflow User Interfaces

Josefina Guerrero García, Christophe Lemaigre, Jean Vanderdonckt and Juan Manuel González Calleros

Abstract A model-driven engineering method is presented 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, a workflow is recursively decomposed into processes that are in turn decomposed into tasks. Each task gives rise to a task model whose structure, ordering, and connection with the domain model allows a semi-automated generation of corresponding user interfaces by model-to-model transformation. Reshuffling tasks within a same process or reordering processes within a same workflow is straightforwardly propagated as a natural consequence of the mapping model used in the model-driven engineering. The various models involved in the method can be edited in a graphical editor based on Petri nets and simulated interactively. This editor also contains a set of workflow user interface patterns that are ready to use. The output file generated by the editor can then be exploited by a workflow execution engine to produce a running workflow system.

2.1 Introduction

The introduction of Workflow Management Systems (WfMS) in organizations has emerged as a major advantage to plan, control, and organize business process. The WfMS in a modern organization should be highly adaptable to the frequent changes. The adaptability of the WfMS includes changes on User Interfaces (UIs) that are used to control business process. To increase adaptability of contemporary WfMS, a mechanism for managing changes within the organizational structure and changes in business rules needs to be reinforced [1, 2]. Even that several approaches have addressed workflow modeling problems, including: graphical notations [3, 4],

J.G. Garcia (✉), C. Lemaigre, J. Vanderdonckt and J.M.G. Calleros
Université Catholique de Louvain, Louvain School of Management (LSM),
Place des Doyens 1, 1348, Louvain-la-Neuve, Belgium
e-mail: josefina.guerrero@uclouvain.be

description languages [3–5], supporting tools [1, 4, 6, 7], workflow patterns [8], and UIs derivation from workflow specifications [9, 10]; integrate all the domains have been poorly explored. Some issues encountered while deriving UI from a workflow specification are the following:

- *User interface hand coded design.* UI derivation from a workflow specification has been used on commercial tools [9], even though the UI is still manually designed and correlated to workflow components. In some cases, several UIs can be predefined for basic UI action types, for instance, Open a File.
- *Lack of integration models of the organization and UI generation.* There are some efforts [4] trying to model the organization and workflow. This second category refers to a totally different problem and is not intended to generate information systems (IS) but to model workflow.
- *Lack of adaptation to organizational changes.* Workflow tools allow managers to design their organization “how it is” and simulate changes on the workflow models to compare whether there are improvements in time, cost, etc. The problems arise when the changes are applied to the organization. Especially when IS are affected. The correct propagation of changes is very difficult to assure, what is more, this work must be hand coded.

These shortcomings stem from the need for a logical definition of workflow models to derive UIs that further allows a computational handling of them as opposed to a physical handling hard coded in particular software. The remainder of this chapter is structured as follows. Section 2.2 explains the conceptual model. Section 2.3 illustrates the different steps that followed in order to derive UIs. Section 2.4 introduces a case study using a tool support. Section 2.5 provides a brief discussion and a comparison with the related work. Section 2.6 gives a final conclusion.

2.2 Conceptual Model of a Workflow Information System

FlowiXML is a methodology [11] for developing the various user interfaces (UIs) of a workflow information system (WIS), which are advocated to automate processes, following a model-driven engineering based on requirements and processes of the organization. The methodology applies 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. Figure 2.1 represents the UML class diagram of this meta-model without any attributes or methods, more details about the attributes and methods of these classes could be found in [11]. The meta-model involves the following models:

- *Workflow model.* It describes how the work in organization flows by defining models of process (what to do?), tasks (how to do it?), and the organizational structure (where and who will perform it?). A workflow model has at least one process and each process has at least two tasks. The heuristics to identify a

to reach a given goal related to the business processes. Introducing task models description to the workflow models corresponds, but is not limited, to the following reasons: (1) Task models describe, opposed to process models, end users' view of interactive tasks while interacting with the system. This allows describing how a task is performed. (2) It is true that in a process model we can add the detail desired, with process hierarchies, to represent a detailed task description. However, we consider that specific temporal operators, iteration, suspend/resume, applied to task, can be more naturally defined in a task model rather into a process model, that implies the creation of dummy transitions. The heuristic to identify a task model are same place, same type of resource, same period of time, and the work is developed by one resource (individual), it could be user, interactive, system or abstract task. Based on the organizational model, we can add a machine task (develop by any mechanical or electrical device that transmits or modifies energy to perform or assist in the performance of tasks. For instance: fax, robot).

- *Organizational model.* It describes the places where work is performed, the users that perform the work, and so on. This part contributes to UI adaptation to different categories of users and security of IS by blocking access to UIs when the user does not have the permission to perform the task. An organizational Unit describes a formal group of people working together with one or more shared goals or objectives. It could be composed of other organizational units. Inside these units a task resource is directly or indirectly involved in carrying out the work. The LogEntry describes specific characteristics of the resources. Each resource may have a log Entry associated with them. A Job represents the total collection of tasks, duties, and responsibilities assigned to one or more positions which require work of the same nature and level, for instance, a surgeon. At this level an Agenda is defined showing assigned tasks to the user. It allows the description of the different status of a task (for instance: not started, in progress), the date when the task begins, the deadline, the date when the task could be assigned or delegated, and the date when the task is completed.
- *Mapping model.* In a model-based approach [14] all the components are models. Even transformation among models and relationships are described in terms of a meta-model. The mapping model defines the relationships between the models. This mapping model allows the specification of the link of elements from heterogeneous models and viewpoints. Several relationships can be defined to explicit the relationships between models. We extended the existing mapping model of UsiXML (www.usixml.org) (as depicted in Fig. 2.2. The extended model contains mappings describing task execution (rules to specify: complex and dynamic users' interaction within the organization), such as: *Is Grafted On* mapping, this relationships is useful when a task (T_j) has been executed, and a task complementary (T_i) is defined to realize the first task where T_i is completely autonomous to T_j . When work is executed tasks are *defined by* a userStereotype. Then, they can be *allocated to task Resources*, following the set of predefined workflow resource patterns, proposed in [8]. These patterns represent the different ways in which tasks are advertised and ultimately bound to specific resources for execution.

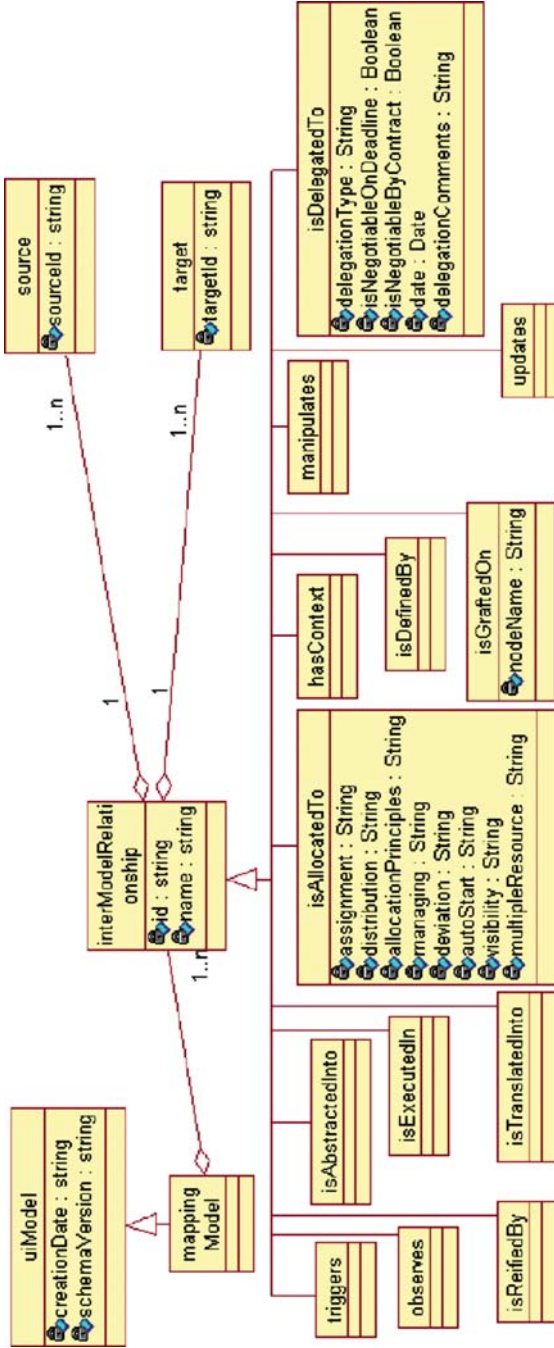


Fig. 2.2 Mapping model

2.3 A Method to Design Workflow User Interfaces

A User Interface Description Language (UIDL) 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 UIs linguistically with a general trend to do so in an XML-complaint way. In a previous work [15] a number of XML-compliant languages for defining user interfaces were identified and analyzed. We select for our work UsiXML as a UIDL for several reasons. The most relevant is its flexibility to be expanded with the models that we proposed. Also, more than a language, UsiXML is a methodology to generate UIs on a model-based approach. The conceptual framework of UsiXML relies on the Cameleon Reference Framework [16]. Reusing this mechanism the UI of a workflow model, that includes task models, can be generated. Model-based approach is intended to assist in designing UIs with a more formal computer supported methodology rather than the more common information paper design, such as storyboarding. It attempts to explicitly represent knowledge that is often hidden in the application code. The problem of generating user interfaces from a workflow specification has several dimensions to be tackled. It is necessary to have UIs to support user's tasks specified in task models, user's communication with agendas which must be updated accordingly as tasks are assigned or ended, and tasks allocation with workflow resource patterns. Also we need a framework not just to generate those UIs automatically but also to specify workflows and task models, integrating the concepts that we propose in previous section. Hence, our method is composed on the following steps to achieve these goals: (1) define the organizational units, (2) define the jobs and user stereotypes, (3) define the workflow, which includes process model, (4) define workflow patterns, (5) define the task models, (6) mapping model from task models to UIs, (7) generate UIs: agendas, UI for each task model.

2.4 Case Study and Tool Support

The purpose of the case study is to give a concrete application of the concepts through the specification of a workflow representing a medical center. We developed a tool (Fig. 2.3) to support the description of workflow models. This workflow editor allows the graphical specification of workflow.

- *Step 1: where? Organizational units' specification.* The first step, which is not mandatory to be the first, consists in specifying the location in which the work must be done. Organizational units' attributes are then specified in the editor and graphically the workflow designer identifies the different components of the organization. Organizational units are represented by rectangles (big rectangles in Fig. 2.3), which will contain a set of ordered tasks and the available resources. It is the way to locate those elements inside the organization. The following organizational units are the structural decomposition of the hospital: (i) reception:

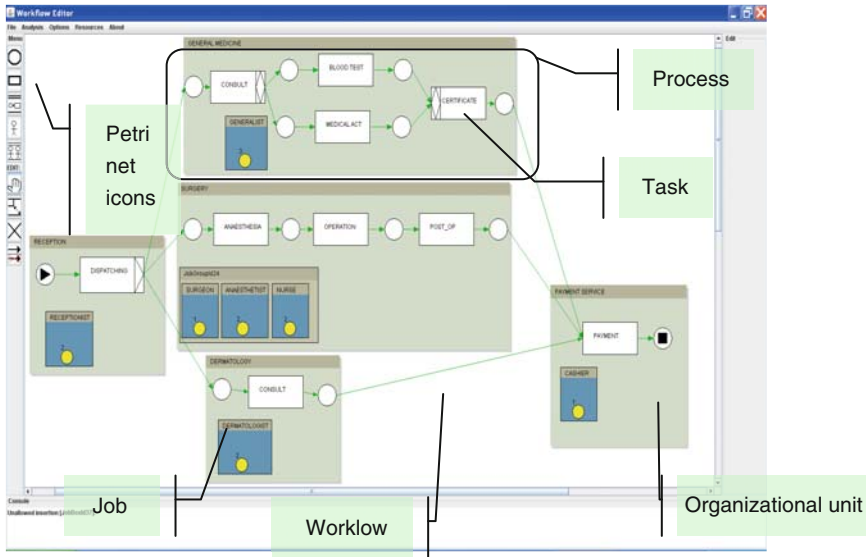


Fig. 2.3 Workflow editor

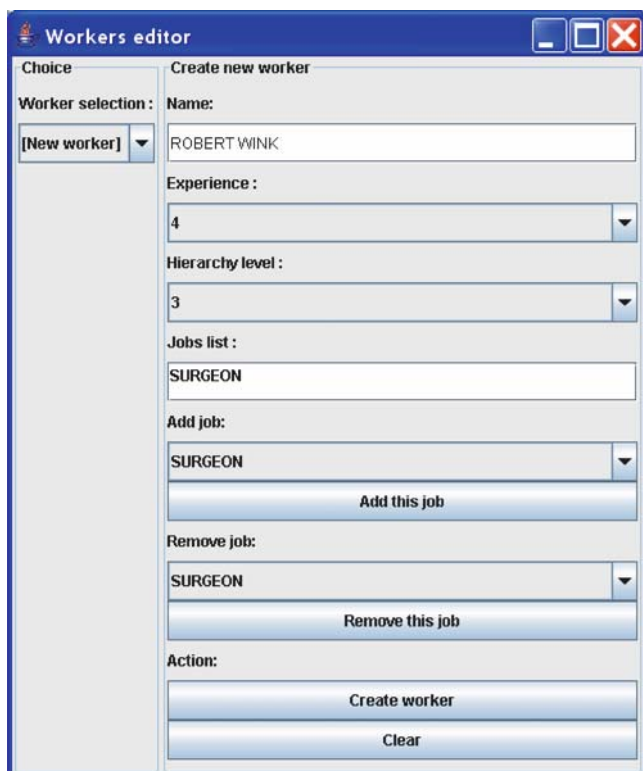
patients coming to this unit will be dispatched through the medical units of the hospital; (ii) general medicine: diagnostic and simple medical acts are realized in this unit; (iii) surgery: patients will be operated in this unit; (iv) dermatology: unit involved in every dermatological resource and the performance of the related medical acts; (v) payment service.

- Step 2: who? Specification of jobs and user stereotypes.* This step consists in the description of all the actors involved in the workflow. For this purpose we define different levels of users, who are the resources that will be in charge of performing the organization work. Jobs are ways to structure the crew of people inside the organization (Fig. 2.4). It involves the complete collection of knowledge and practices needed by a definite human resource to perform a task. Jobs specified in the definition of the case study are the following: Receptionist, Generalist, Surgeon, Anesthetist, Nurse, Dermatologist, and Cashier. Once jobs are defined it is possible to incorporate user stereotypes, people able to carry out tasks of a particular job. The workers editor (Fig. 2.5) is used for this purpose. Workers are defined in terms of attributes (name, experience, hierarchy level) and the list of jobs they can perform. For instance, we define a user stereotype called Robert Wink, having 4 years experience in the third hierarchy level. He is able to carry out tasks as a generalist and surgeon. Also, it is necessary to assign them a place into the organizational scheme. A user stereotype may be assigned to several organizational units. The graphical representation used for the workflow editor is based on a first resource container inside the organizational unit. It allows the workflow designer to group resources. Job boxes are put inside of the main resource box. Each job box is instantiated by user stereotypes able to perform

Fig. 2.4 Job handler editor

the job of the box. This leads to the kind of representation given in Fig. 2.3 (small rectangles). The organizational unit contains a resource box made of three job boxes. Every job box instantiates user stereotypes of a certain job (there are two surgeons, one anesthetist and one in the given example). This lets managers know which resources are available for execute a task in an organizational unit.

- *Step 3: what? Workflow specification.* The workflow specification, depicted in the process model, takes place inside of the organizational unit framework. Concretely, the workflow represents the business process and determines the right resource for the right task at the right time. This part of the graphical notation (Fig. 2.3) of the workflow is based on Petri nets [12].
- *Step 4: whom? Defining workflow resource patterns.* It is important to specify who will be in charge of what. For that purpose, we use workflow resource patterns [8] to assign or offer tasks. As, we have already defined jobs and user stereotypes, now we add rules defining the way work will be undertaken. The resource pattern editor (Fig. 2.6) allows the workflow designer to specify resource patterns. At first a list of jobs required to carry out task is specified in the editor. The workflow designer selects one ore more jobs allowing a user stereotype to realize the task. For the moment, 43 workflow resource patterns [8] have been incorporated so that



Workers editor

Choice

Worker selection : [New worker] ▼

Create new worker

Name: ROBERT WINK

Experience : 4 ▼

Hierarchy level : 3 ▼

Jobs list : SURGEON

Add job: SURGEON ▼

Add this job

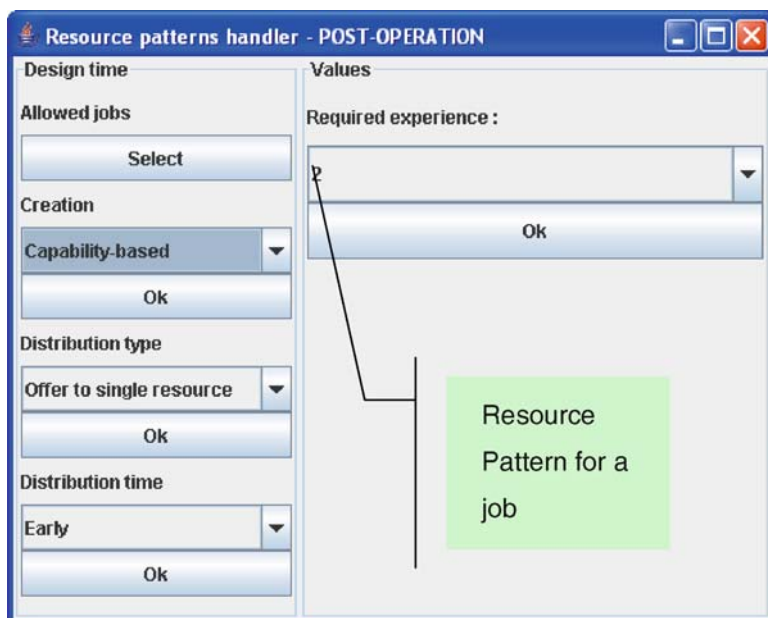
Remove job: SURGEON ▼

Remove this job

Action: Create worker

Clear

Fig. 2.5 Workers editor



Resource patterns handler - POST-OPERATION

Design time

Allowed jobs

Select

Creation

Capability-based ▼

Ok

Distribution type

Offer to single resource ▼

Ok

Distribution time

Early ▼

Ok

Values

Required experience :

2 ▼

Ok

Resource Pattern for a job

Fig. 2.6 Resource patterns editor

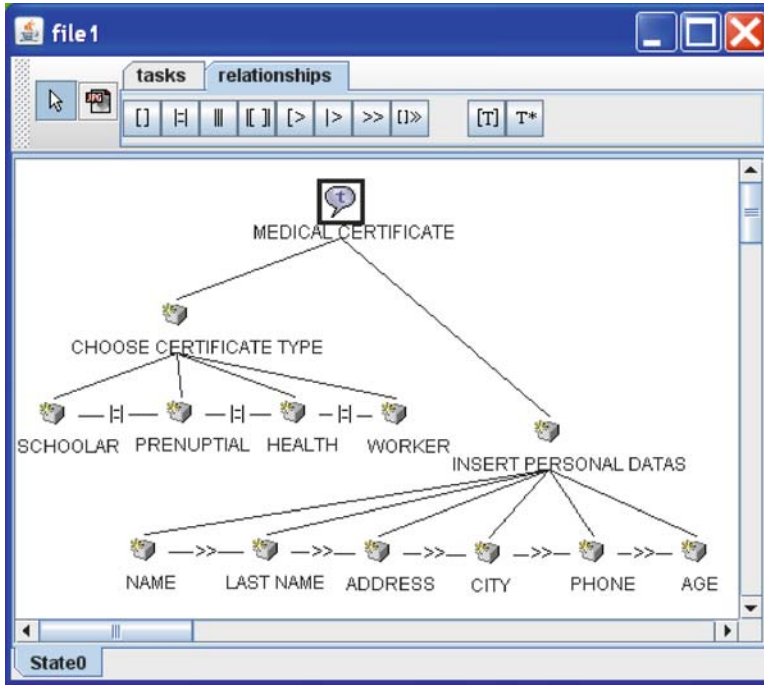


Fig. 2.7 Task model editor

the designer may apply them directly using a predefined UI. Each UI pattern is expressed in UsiXML and is stored in a pattern repository. For the moment, there is a one-to-one mapping between the workflow pattern and the UI pattern. In the future, we plan to expand this mapping with parameters.

- *Step 5: how? Task models specification.* For each process a task model can be specified to describe in detail how the task is performed. By exploiting task model descriptions different scenarios could be conducted. Each scenario represents a particular sequence of actions that can successfully be performed to reach a task goal (Fig. 2.7).
- *Step 6: Mapping the workflow to UI.* Finally we have to deal with the problem of generating the complete UIs set to support all the designed workflow in run-time. This step is achieved by relying on the UsiXML method that progressively moves from a task model to a final user interface. This approach consists of three steps: deriving one or many abstract user interfaces from a task model, deriving one or many 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 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 we obtain its correspondent UI (Fig. 2.8).

The image shows a 'Medical Services' dialog box. It has a title bar with standard Windows window controls. The main content is divided into two panes. The top pane, 'Medical Certificate', contains four radio buttons for selecting a certificate type: 'Scholar', 'Prenuptial', 'Health', and 'Worker'. The bottom pane, 'Personal Data', contains six text input fields for 'Name', 'Last Name', 'Address', 'City', 'Phone Number', and 'Age'. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Fig. 2.8 User interface (UI) derived from task model

2.4.1 The Simulator Tool

After we develop all the UIs for each task, we have control of how the work is flowing inside the organization, for this purpose we have a workflow editor. Following the Petri net representation, resource choice is made when a token is in place preceding a transition. It is managed following resource patterns defined with the editor. When a task is started the associated token goes from a place to the associated transition. In this way, work in progress is represented in the workflow simulation diagram. Each user that participated in the workflow should have an *agenda* to view and manage the tasks that are assigned or offered to him. Each agenda can be visualized as a queue of tasks assigned to a resource. Through agendas we can support the work among resources or groups (Fig. 2.9). As we said, one important aspect to consider is any change in the workflow and to have the possibility to manage it.

2.5 Discussion and Related Work

While reviewing the literature one can easily see the extensive research of the organization, their process, adaptability, etc. In the same venue, WfMS research includes graphical notations [3, 4], description languages [3–5], supporting tools [1, 4, 6, 7],

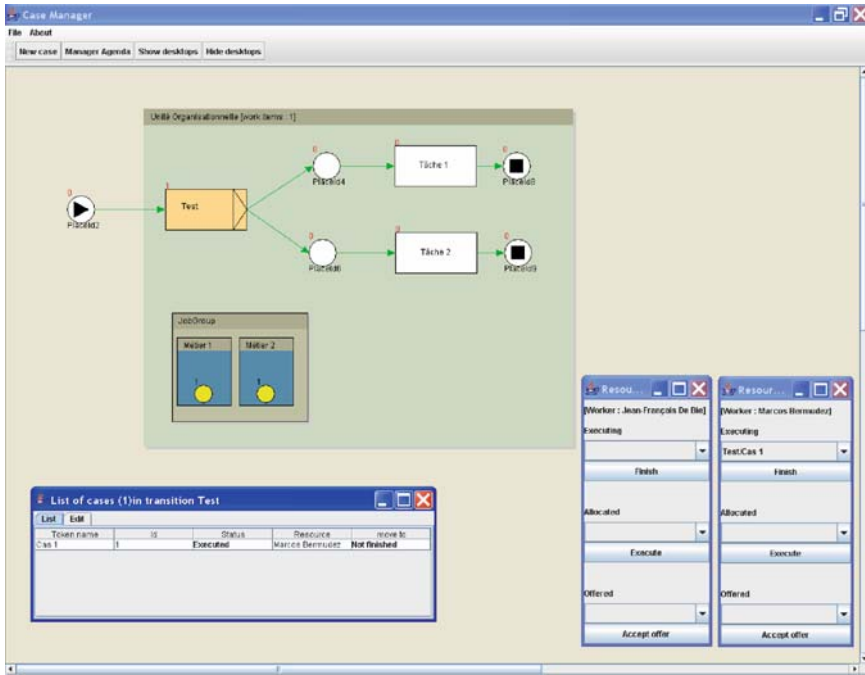


Fig. 2.9 Workflow manager tool

and workflow patterns [8], each tackling specific and independent issues of modern organizations. In this chapter we introduced a model that includes all these aspects, which are relevant and have an impact one to each other when changes are applied. We use a model-driven engineering approach for the user interface design, as it aids in creating interactive software that considers multiple factors, such as users, tasks, and so on. Still there are missing points regarding our model. First, we consider that it is fundamental to address Mandviwalla & Olfman [17] criteria for support group interactions, such as the following ones: (a) support multiple group tasks, (b) support multiple work methods, (c) support the development of the group, d) provide interchangeable interaction methods, (e) sustain multiple behavioral characteristics, (f) accommodate permeable group boundaries, (g) adjustability to the group context. In [18] there are usability guidelines that can be considered, for a future work, as a principle that has to be taken into account for building UIs respecting cognitive and sensory-motor capabilities of users. By linking user interfaces of a WfMS we expect to solve the problem of synchronizing the communication between UIs (agendas and task UIs) and the workflow view. One option can be client–server architecture. So far we can just simulate agendas interaction. The solution should provide communication channels from the workflow manager application (server) to every userStereotype agenda (clients). In the domain of model-driven engineering, Stavness [1] presents a progression model in order to support workflow execution, but not a complete decomposition of processes along with jobs and organizational units is included.

The same observation holds for [6, 10]. In particular, in [10], a task model is indeed used, but only its hierarchical decomposition is used. Therefore, our method and our supporting tool differ from the state-of-the-art in that it is based on several models (not just data or tasks), some coming from theory of organizations. The graphical notation is based on Petri nets as in [2, 3]. In [19] a method called AMOMCASYS is presented, this method is also based on Petri nets, it is aimed at modeling and simulating complex administrative systems.

2.6 Conclusion

This chapter defined a method for designing UI of WISs where UI are directly derived from a model of the 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 *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. In order to partially support this method, a software tool has been developed in Java 1.5 that supports the graphical editing of the concepts introduced in an integrated way. It then enables designers to pick any of the predefined 43 workflow resource patterns that are later attached to a corresponding UI pattern in UsiXML. This method has been so far validated on four real-world case studies (e.g., a hospital dept., a triathlon organization, a cycling event, and personalized order of compression stockings over Internet). More information, including a video demo of the software can be found at: <http://www.usixml.org/index.php?mod=pages&id=40>.

References

1. Stavness, N., Schneider, K.A.: Supporting Flexible Business Processes with a Progression Model. In: Proc. of the 1st Int. Workshop on Making model-based user interface design practical: usable and open methods and tools MBUI'2004 (Funchal, January 13, 2004) CEUR Workshop Proceedings, Vol. 103. Accessible at <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-103/stavness-et-al.pdf>.

2. van der Aalst, W.M.P., van Hee, K.: *Workflow Management: Models, Methods, and Systems*. The MIT Press, Cambridge (2002).
3. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. *Information Systems* 30, 4 (2005) 245–275.
4. van Hee, K., Oanea, O., Post, R., Somers, van der Werf, J.M.: Jasper: a tool for workflow modeling and analysis. In: *Proc. of 6th Int. Conf. on Application of Concurrency to System Design* (2006) 279–282.
5. Dumas, M., ter Hofstede, A.: *UML Activity Diagrams as a Workflow Specification Language*. In: *Proc. of 4th Int. Conf. on the Unified Modeling Language, Concepts, and Tools UML'2001* (Toronto, October 1–5, 2001). *Lecture Notes in Computer Science*, Vol. 2185. Springer, Berlin (2001) pp. 76–90.
6. Lee, H. B., Kim, J. W., Park, S. J.: KWM: Knowledge-based Workflow Model for Agile Organization. *Journal of Intelligent Information Systems* 13 (1999) 261–278.
7. van der Aalst, W.M.P., Kumar, A.: XML Based Schema Definition for Support of Inter-organizational Workflow. In: *Proc. of 21st Int. Conf. on Application and Theory of Petri Nets ICATPN'2000*. LNCS, Vol. 1825. Springer, Berlin (2000) 475–484.
8. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow Resource Patterns: Identification, Representation, and Tool Support. In: *Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'2005* (Porto, June 13–17, 2005). *Lecture Notes in Computer Science*, Vol. 3520. Springer, Berlin (2005) 216–232.
9. Kristiansen, R., Trøttemberg, H.: Model-Based User Interface Design in the Context of Workflow Models. In: *Proc. of 6th Int. Workshop on Task Models and Diagrams for User Interface Design TAMODIA'2007* (Toulouse, November 7–9, 2007). *Lecture Notes in Computer Science*, Vol. 4849. Springer, Berlin (2007) pp. 227–239.
10. Stolze, M., Riand, Ph., Wallace, M., Heath, T.: Agile Development of Workflow Applications with Interpreted Task Models. In: *Proc. of 6th Int. Workshop on Task Models and Diagrams for User Interface Design TAMODIA'2007* (Toulouse, November 7–9, 2007). *Lecture Notes in Computer Science*, Vol. 4849. Springer, Berlin (2007) 2–14.
11. Guerrero, J., Vanderdonckt, J., Gonzalez, J.M.: FlowiXML: A Step Towards Designing Workflow Management Systems. *Journal of Web Engineering* 4, 2 (2008) 163–182.
12. van der Aalst, W.M.P.: The application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers* 8, 1 (1998) 21–66.
13. Paternò, F. *Model-based design and evaluation of interactive applications*. Springer (1999).
14. Puerta, A.R.: A Model-Based Interface Development Environment. *IEEE Software* 14, 4 (1997) 41–47.
15. Souchon, N., Vanderdonckt, J.: A review of XML-compliant user interface description languages. In: *Proc. of DSV-IS'2003*. Springer, Berlin (2003) 377–391.
16. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computers* 15, 3 (2003) 289–308.
17. Mandviwalla, M., Olman, L.: What do groups need? A proposed set of generic groupware requirements. *ACM Transactions on Computer-Human Interaction* 1, 3 (1994) 245–268.
18. Palanque, P., Farenc, Ch., Bastide, R.: Embedding Ergonomic Rules as Generic Requirements in a Formal Development Process of Interactive Software. In: *Proc. of IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact'99* (Edinburgh, September 1–4, 1999). IOS Press, Amsterdam (1999) 408–416.
19. Adam, E., Kolski, C., Mandiau, R., Vergison, E.: A software engineering workbench for modeling groupware activities. In: C. Stephanidis (Ed.), *Universal Access in HCI: inclusive design in the information society*. Lawrence Erlbaum Associates, Mahwah, NJ (2003) pp. 1499–1503.

Computer-Aided Design of User Interfaces VI

Lopez Jaquero, V.; Montero Simarro, F.; Molina Masso, J.P.; Vanderdonckt, J. (Eds.)

2009, IX, 319 p. 139 illus., Hardcover

ISBN: 978-1-84882-205-4