Computer-Aided Design of User Interfaces VI

Víctor López-Jaquero • Francisco Montero
José Pascual Molina • Jean Vanderdonckt
Editors

Computer-Aided Design of User Interfaces VI

Proceedings of the Seventh International Conference on Computer-Aided Design of User Interfaces (CADUI 2008)

Springer

Editors
Víctor López-Jaquero
Escuela Superior de Ingeniería Informática
University of Castilla-La Mancha
Albacete, Spain

José Pascual Molina
Escuela Superior de Ingeniería Informática
University of Castilla-La Mancha
Albacete, Spain

Francisco Montero
Escuela Superior de Ingeniería Informática
University of Castilla-La Mancha
Albacete, Spain

Jean Vanderdonckt
Université catholique de Louvain
Louvain-la-Neuve
Belgium

ISBN: 978-1-84882-205-4 e-ISBN: 978-1-84882-206-1
DOI 10.1007/978-1-84882-206-1

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2008944094

Printed on acid-free paper

Springer Science+Business Media
springer.com

Contents

Sponsors

Oficial CADUI Web site

Computer-Aided Design of User Interfaces
http://www.isys.ucl.ac.be/bchi/cadui
http://cadui2008.albacete.org

Scientific Sponsors

Corporate Sponsors

Programme Committe Members

General co-chairs

Víctor López-Jaquero, University of Castilla La Mancha, Spain
José Pascual Molina, University of Castilla La Mancha, Spain
Francisco Montero, University of Castilla La Mancha, Spain

PC members

Julio Abascal, Univ. of País Vasco, Spain
Lawrence Bergman, IBM T.J. Watson Research Center, USA
Niels Ole Bernsen, University of Southern Denmark, Denmark
Birgit Bomsdorf, Univ. Hagen, Germany
Marcos Borges, Univ. Federal do Rio de Janeiro, Brasil
Gaelle Calvary, Univ. Joseph Fourier, France
Pedro Campos, Univ. of Madeira, Portugal
Karin Coninx, Univ. of Hasselt, Belgium
Larry Constantine, Univ. of Madeira, Portugal
Angélica de Antonio Jiménez, Univ. Politécnica de Madrid, Spain
Olga De Troyer, Vrije Univ. Brussel, Belgium
João Falcão e Cunha, FEUP, Porto, Portugal
Clive Fencott, University of Teesside, UK
Peter Forbrig, University of Rostock, Germany
Elizabeth Furtado, Univ. Fortaleza, Brazil
Toni Granollers, Univ. of Lérida, Spain
Geert-Jan Houben, Vrije Univ. Brussel, Belgium
Javier Jaén, Univ. Politécnica de Valencia, Spain
Anthony Jameson, DFKI, Germany
Joaquim Jorge, IST, Lisbon, Portugal
Christophe Kolski, Univ. de Valenciennes, France
Quentin Limbourg, SMALS-MVM, Belgium
Kris Luyten, Univ. of Hasselt, Belgium

José Antonio Macías, Univ. Autónoma de Madrid, Spain
Mark Maybury, The Mitre Corp., USA
Pedro J. Molina, Capgemini, Spain
Kizito Ssamula Mukasa, Fraunhofer IESE, Germany
Jeff Nichols, IBM Almaden Research Center, USA
Erik Nilsson, SINTEF, Norway
Nuno Nunes, Univ. of Madeira, Portugal
Philippe Palanque, IRIT, Université Paul Sabatier - Toulouse III, France
Fabio Paternò, ISTI-CNR, Italy
Óscar Pastor, Univ. Politécnica de Valencia, Spain
Beryl Plimmer, Univ. Of Auckland, New Zealand
Angel Puerta, RedWhale Corp., USA
David Ragget, W3C, UK
Arcadio Reyes Lecuona, Univ. of Málaga, Spain
Gustavo Rossi, Univ. De La Plata, Argentina
Dominique Scapin, INRIA, France
Robbie Schaefer, Universitaet Paderborn, Germany
Montserrat Sendín, Univ. of Lérida, Spain
Orit Shaer, Tufts University, USA
Daniel Schwabe, Pontifícia Universidade Católica do Rio de Janeiro, Brazil
Constantine Stephanidis, ICS-Forth, Greece
Hallvard Traetteberg, Norwegian Univ. of Science and Techn., Norway
Jean Vanderdonckt, Université Catholique de Louvain, Belgium
Marco Winkler, IRIT, Université Paul Sabatier - Toulouse III, France

Steering committee

Gaelle Calvary, Univ. Joseph Fourier, France
Christophe Kolski, Univ. de Valenciennes, France
Fabio Paternò, ISTI-CNR, Italy
Angel Paternò, RedWhale Corp., USA
Jean Vanderdockt, Université Catholique de Louvain, Belgium

Organizing committee

José Eduardo Córcoles, University of Castilla-La Mancha, Spain
José A. Gallud, University of Castilla La Mancha, Spain
Arturo S. García, University of Castilla La Mancha, Spain
Jose Manuel Gascueña, University of Castilla La Mancha, Spain
Pascual González, University of Castilla La Mancha, Spain
María Teresa López, University of Castilla La Mancha, Spain
María Dolores Lozano, University of Castilla La Mancha, Spain
Diego Martínez, University of Castilla-La Mancha, Spain
Elena Navarro, University of Castilla-La Mancha, Spain
Víctor M. R. Penichet, University of Castilla La Mancha, Spain
Manuel Tobarra, University of Castilla-La Mancha, Spain

Chapter 1
The Challenges of User-Centred Design

William Hudson

Abstract A number of perceptual and psychological issues conspire to make the successful design of interactive systems – and user interfaces in particular – much more difficult than it would seem at first sight. This paper describes the author's keynote address to the CADUI conference and investigates these issues, touching on attentional, change and mud splash blindnesses. It also explores the diffculties technologists can have in understanding user's needs, as demonstrated by a recent study in empathizing and systemising skills within the IT sector.

1.1 People Are Not Perfect

While human beings are amazing creatures, we have our limitations. In the field of design, one glaring limitation is our willingness to overlook them. We design and develop systems that assume the visual acuity of an eagle; memory of an elephant; navigation skills of a bat; stamina of a camel; and the dexterity of a monkey (see Fig. 1.1) [1].

There are several reasons for this. The first is that, by default, designers and developers focus very intently on the problem at hand *in the abstract*. Issues that stem from human limitations or needs (such as leaving the office to eat or sleep) are peripheral to the solution being designed. However, to make matters worse, there are several human limitations relevant to interactive systems that are not very well known within the field of Human–Computer Interaction (HCI). All these stem from failings of visual perception and so are called 'blindnesses': attentional blindness, change blindness, and mud splash blindness.

Attentional blindness is well known within the field of visual perception [2]. It is best illustrated through demonstration, but even a description of the problem is fairly impressive. Perhaps the best-known example is a short video clip of two teams of students wearing either black or white T-shirts (depending on the team). The audience is told simply to count the number of times the teams pass a ball between them as

W. Hudson
Syntagm Ltd, Oxford, UK
e-mail: william.hudson@syntagm.co.uk

V. López-Jaquero et al. (eds.), *Computer-Aided Design of User Interfaces VI*,
DOI: 10.1007/978-1-84882-206-1_1,

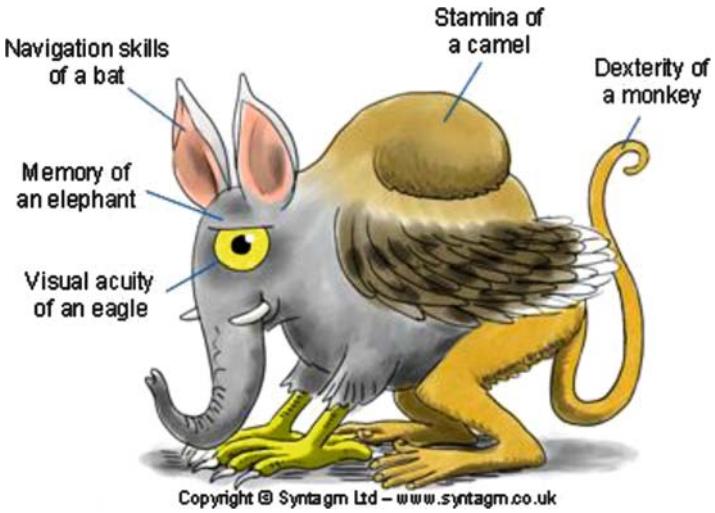

Fig. 1.1 The perfect user

they move about in a fairly distracting manner. About half way through the clip, someone dressed in a gorilla suit walks into the scene, beats their chest and then walks off. They are on screen altogether for about 5 s. At the end of the clip the audience is asked if they observed anything unusual. Only about half of the audience will have noticed the gorilla. The other half of the audience was so intent on performing the task in hand that they were oblivious to this unexpected event.

Another surprising aspect of visual attention is our inability to see changes on a screen when a brief blanking field is present – the kind that separates virtually all web pages as the browser loads new content. The phenomenon is called change blindness [3]. Its effect is a little harder to predict than inattention blindness as some participants will notice the change straight away but others may give up after a minute or two.

The third perceptual issue is related to change blindness. Rather than a blanking field between screens, its contents are changed at the same time as simulated mud splashes – hence its name, mud-splash blindness. Participants find it almost impossible to say what has changed.

All three of these issues have important implications for design. Users who might be very distracted by their tasks risk not noticing important information (a gorilla!) on their screens. Changes to web pages may not be seen on reload because of change blindness. And finally, animations or popup boxes, similar to mud splashes in their effect, may mask other changes that occurred at the same time.

1.2 Designers Are Not Perfect

Twenty years on from Don Norman's *The Psychology of Everyday Things* [4] designers are still creating even simple technology with unhelpful user interfaces. The two examples shown here from a recent hotel stay made it difficult to know

Fig. 1.2 Designs are not perfect

what temperature the water would be (contrary to what might be expected from the left-hand image in Fig. 1.2, where the dot marked 'A' is red, this is the cold setting). In the same hotel room, it is hard to understand why a toilet would have two different flush controls when it is impossible to guess what they do.

The difficulty in these and many screen-based examples of poor design is that we still do not teach (or understand) visual language. We would understand it better if we worked more directly with the users of our creations, but that is still relatively rare. So, for every well-designed web site, desktop application or phone, there are hundreds that could be more self-explanatory and easier to use.

For example, an early version of the Microsoft web site page for Intranet Explorer version 8 should have been fairly straightforward (see Fig. 1.3). But the visual language used suggests that selecting an operating system (A) will show appropriate system requirements (B). On the contrary, the two parts of the page are unrelated. Once the operating system is selected and the Go button pressed, the page is abandoned and replaced with a new one to perform a download.

1.3 User-Centred Design (UCD) ≠ Usability ≠ Cool

Design examples of this kind are plentiful, but there is an even deeper problem. The pressures to engage and excite customers have created a fog of confusion around the concepts of user-centred design (UCD), usability, and 'coolness'.

These three ideas are related but, as Fig. 1.4 shows, not equivalent. User interfaces can be *usable* without being *useful* (as represented by the UCD circle) and they can be *cool* without being either. And regrettably, for customers and users, the current trend is towards coolness without substance. Microsoft Windows Vista,

Fig. 1.3 The new Microsoft web site page for Intranet Explorer version 8

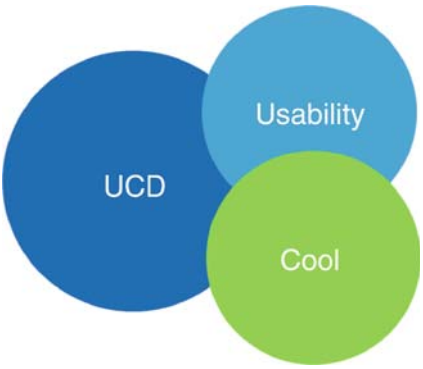

Fig. 1.4 The concepts of user-centred design (UCD), usability, and 'coolness'

Office 2007 and Apple's iPhone are all examples of user interfaces that have been designed to be appealing, but in many cases are actually more difficult to use than their predecessors. (The iPhone requires that users have appropriate-sized fingers, for example. It does not recognize a stylus.)

Fig. 1.5 Two views of the same toolbar from Microsoft PowerPoint 2007

Consider Fig. 1.5. This shows two views of the same toolbar from Microsoft PowerPoint 2007. The only difference is the window size. In smaller windows, the toolbar is compressed to fit. Cool, but very difficult for technical support departments who are trying to assist colleagues without seeing their screen. And, unlike all previous versions of Microsoft Office applications, the 'ribbon' as this interface is called, completely replaces the menus.

An additional challenge has been introduced for Windows Vista. The title bars are translucent, which, although attractive to some, makes it difficult to see where the title bar ends and the next window begins. Since users must drag the title bar to move windows on the screen, they sometimes end up clicking in the wrong window. It is hard to imagine what user need has been addressed by these and many other changes on the path to coolness. Yet, at the same time, truly helpful features are overlooked. As a case in point (and through no fault of Microsoft), it is not possible to buy a flight and a hotel package from a travel web site if you would like a hotel that is not near an airport. So, booking a flight and overnight stay in Heidelberg (Germany) is impossible in a single transaction since there is no airport in Heidelberg. It is left to the customer to find an appropriate airport, means of transport, and hotel.

1.3.1 Why Is There Not More UCD?

Apart from the drive for coolness, what is holding UCD back? One of the most common reasons was expressed perfectly by Jack Warner of the Hollywood studio bearing his name:

> I don't want it good, I want it Tuesday.

UCD and usability are thought of as either optional (when thought of at all) or as enhancements that can be added later. A further complication with usability is that it is actually very limited in its scope. If a travel web site does not offer the means of booking a hotel away from an airport, then that missing functionality will not be usability-tested by definition. It is a very brave usability specialist that tells their customer or employer that they have built the wrong system.

Many of these shortcomings stem from an unwillingness to conduct early user research and the continuing trend of hiding systems builders away in back rooms. The 'back room' approach is fine in large companies with well-established processes for user-research and communicating user needs in detail to system builders. But given that the majority of interactive systems are built in small companies with small teams having little or no understanding of UCD, such a pronounced separation of technologists from their users is extremely counter-productive.

In organizations that do employ usability professionals, their efforts are often misdirected for two reasons. The first is that many commercial organizations are reluctant to allow anyone other than sales staff to have direct contact with customers. The second is that where bespoke usability facilities exist (such as an expensive lab with video cameras and observation rooms), there is enormous pressure to make good use of them, at the expense of the design process itself. In this latter case, success is often measured as a fully booked usability lab, even if the work that is booked – user research, for example – should be conducted in the field [5].

1.3.2 Empathetic Design

Before we look at solutions to some of these challenges, there is one further problem area to explore. Like the visual perception issues discussed earlier, it is inherent in the human condition: the people who are best at creating technology are often the worst at the understanding how and why other people find it difficult to use. The evidence for this comes from a different branch of psychology, investigating the causes of autism and Asperger's syndrome (AS). Simon Baron-Cohen and his colleagues at the Autism Research Centre have developed a model they use to explain the significant differences in behaviour between men and women, called empathizing–systemizing theory [6, 7]. A related theory, known as the 'extreme male brain' [8] characterizes the more extreme differences between the normal population and suffers of autism and AS.

Empathizers are interested in people and social interaction while systemizers are more focused on the physical world and causality. On average, men score higher than women on systemizing while women score higher than men on empathizing. Not surprisingly, a large study of empathizing and systemizing within the IT field (441 participants) showed systemizing scores for men and women that were both substantially higher than the average population [9]. However, men whose job roles were predominantly technical had significantly lower empathizing skills, as illustrated in Fig. 1.6. (The few women who stated that their job roles were primarily technical also showed this effect, but it was less significant.)

Ideally, we would have equivalent technology for interactive systems that would allow designers and developers to empathize with users. They would do this by showing how a web page looked to a 60-year-old (that is the purpose of the yellowed goggles and helmet visor in Fig. 1.7) or simulate how difficult it is to select a menu when you have trouble moving the mouse in a straight line.

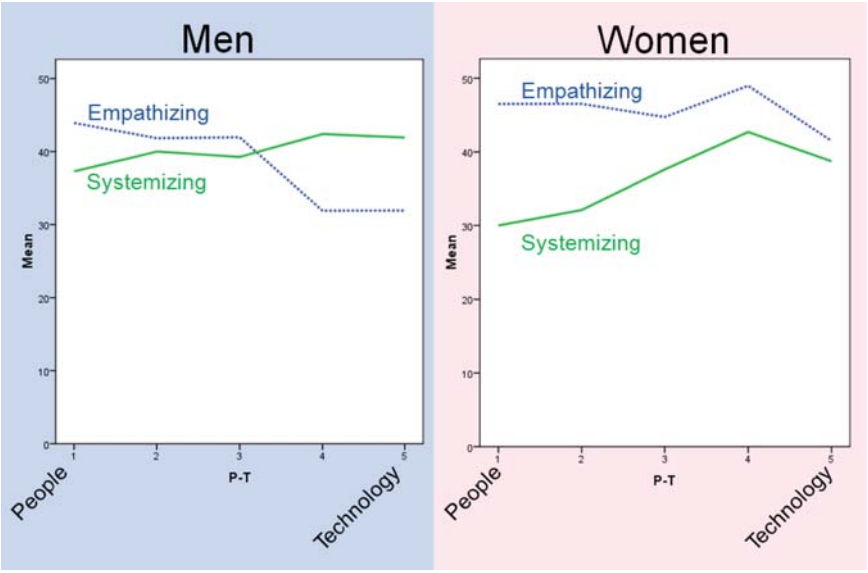

Fig. 1.6 Men whose job roles were predominantly technical had lower empathizing skills

Fig. 1.7 Examples of empathy-assistive technology (images courtesy of the University of Loughborough, except bottom-right: Meyer-Hentschel [10])

A big part of selling empathetic design, though, is persuading people that it is necessary. Happily, this is where the gorilla returns. Many of the audience in the visual perception demonstrations mentioned earlier are truly stunned by what they have learned of the human condition. The same revelations occur almost every time a developer watches one participant after another fail at the same point in a task during a usability evaluation. Many technologists may not be naturally empathetic, but the difficulties that users face are not beyond their understanding.

So, humans have shortcomings not only as users but also as designers and developers (and possibly managers, executives, entrepreneurs, and other roles in which systemizing skills are valued). To overcome them – to design useful and usable systems – we must recognize those limitations and take steps to compensate for them. In UCD in particular, it further emphasizes the need for multidisciplinary design, field research of users, and collaborative design techniques such as card sorting or affinity diagramming.

But for everyone concerned with creating technological solutions, it means a much greater emphasis on understanding people and seeing problems through their eyes. To do that means involving more empathizers in the design process as well as persuading more technologists of the need for empathetic design.

References

1. Hudson, W. (2002). Simulating the Less-Than-Perfect User. SIGCHI Bulletin, 34 (March–April 2002).
2. Simons, D. and Chabris, C. (1999). Gorillas in our midst: sustained in attentional blindness for dynamic events. Perception, 28, 1059–1074.
3. Hudson, W. (2002). Designing for the Grand Illusion. SIGCHI Bulletin, 33 (November–December 2001).
4. Norman, D. A. (2002). The Design of Everyday Things. (Basic Books).
5. Hudson, W. (2004). My Place or Yours: Use and Abuse of Research Facilities. In Interactions, 11(3).
6. Baron-Cohen, S. (2003). The Essential Difference: Men, Women and the Extreme Male Brain Allen Lane (Penguin Press).
7. Baron-Cohen, S., Richler, J., Bisarya, D., Gurunathan, N. and Wheelright, S. (2003). The systemizing quotient: an investigation of adults with Asperger syndrome or high-functioning autism, and normal sex differences. Phil. Trans. R. Soc. Lond. B, 358(1430), 361–374.
8. Baron-Cohen, S. (2002). The extreme male brain theory of autism. Trends Cogn. Sci., 6(6), 248–254.
9. Hudson, W. (2009). Reduced Empathizing Skills Increase Challenges for User-Centered Design. In Proceedings of CHI 2009.
10. Meyer-Hentschel. URL http://www.mhmc.de/HTML/age_explorer.html.

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

V. López-Jaquero et al. (eds.), *Computer-Aided Design of User Interfaces VI*,
DOI: 10.1007/978-1-84882-206-1_2,

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

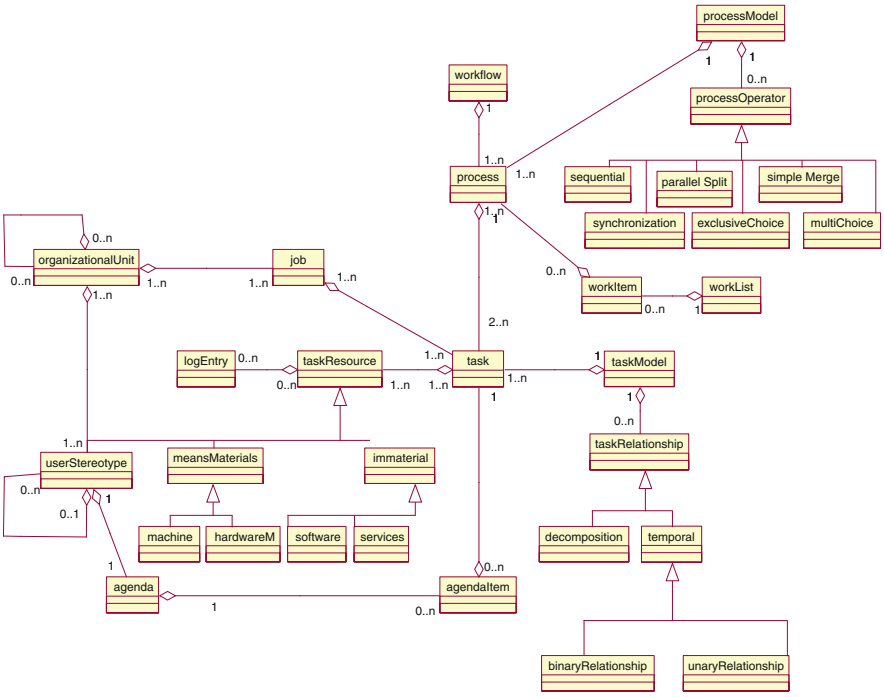

Fig. 2.1 Partial view of meta-model

workflow model are: it is associated to the operational and/or administrative objectives of organization, is performed within the same organization and it is associated to the automation of a business process.

- *Process model.* The definition of a process indicates the ordering of tasks in time, space, and resources. Our model is an adaptation of the Petri net notation proposed in [2, 12] and is compatible with the workflow resource patterns proposed in [8]. The concept of *work List* is introduced, which stocks the processes of the whole organization. Managers are benefited as they can identify resources performing tasks, status of the workflow, bottlenecks in the processes and the identification of the organizational unit where the task is performed. The heuristics to identify a process model are same group of resources, continuous period of time, specific ordering of tasks, the work is developed within groups, among groups, or by a group as a whole, is not further divided into sub-processes and it could be primary (production), secondary (support), or tertiary (managerial).
- *Task model.* Task models are used to collect the requirements of a workflow system. Task models are mechanisms to represent user's tasks along with their logical and temporal ordering. An adapted version of ConcurTaskTree (CTT) [13] is used in this work. A task is an activity that has to be performed by users (human, systems, humans interacting with systems, or a combination of them)

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 (*Tj*) has been executed, and a task complementary (*Ti*) is defined to realize the first task where *Ti* is completely autonomous to *Tj.* When work is executed tasks are *defined by* a userStereotype. Then, they can be *allocated to task Resource*s, 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.

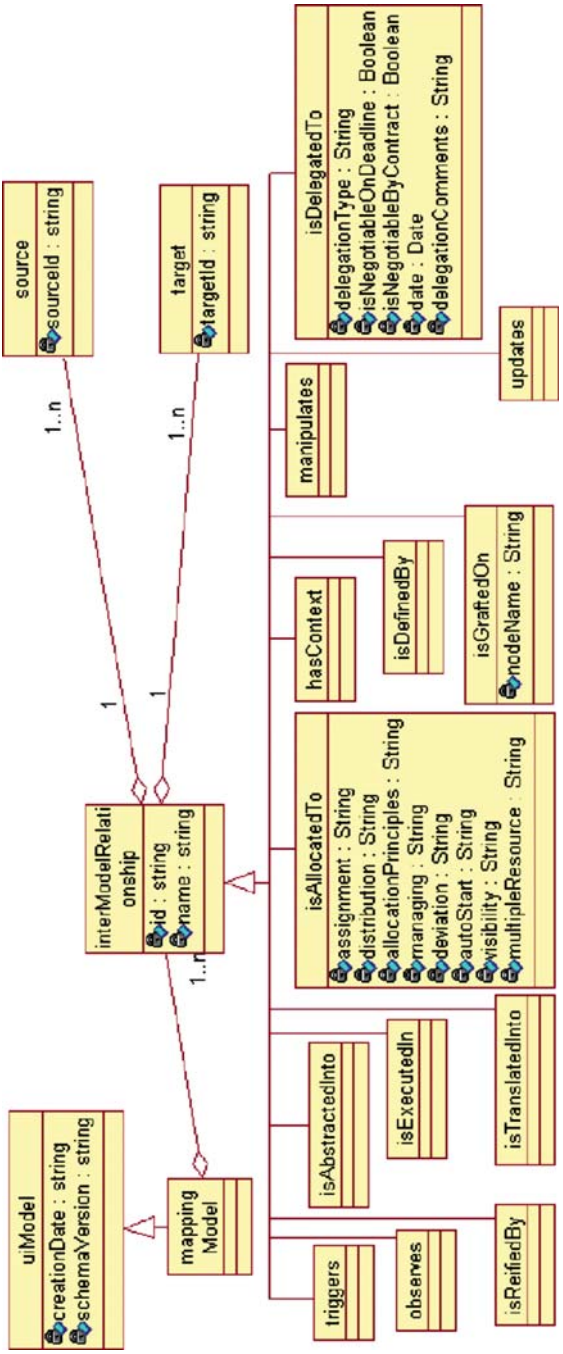

Fig. 2.2 Mapping model

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:

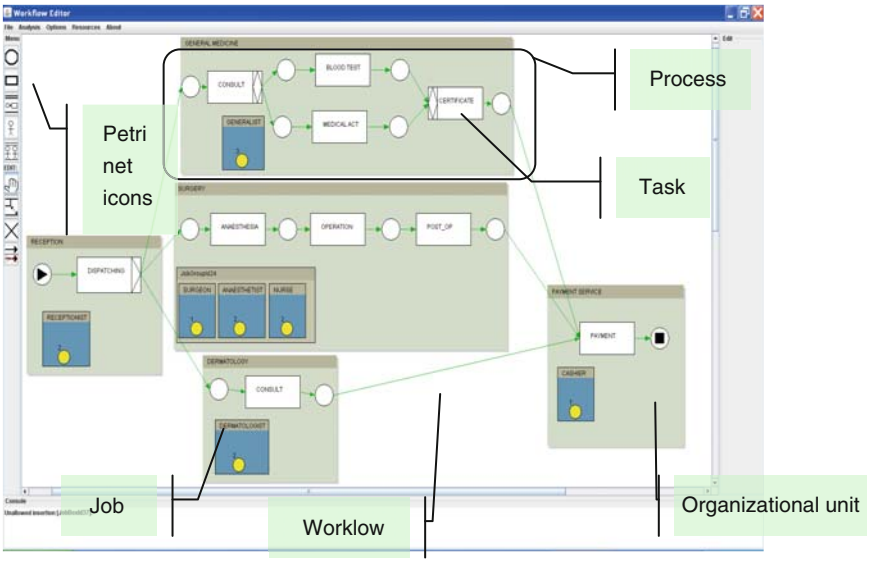

Fig. 2.3 Workflow editor

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

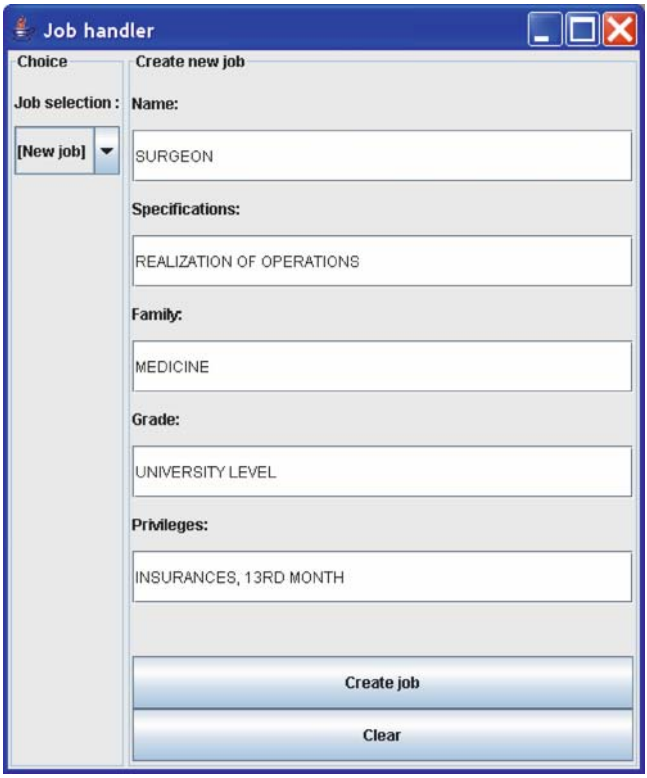

Fig. 2.4 Job handler editor

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

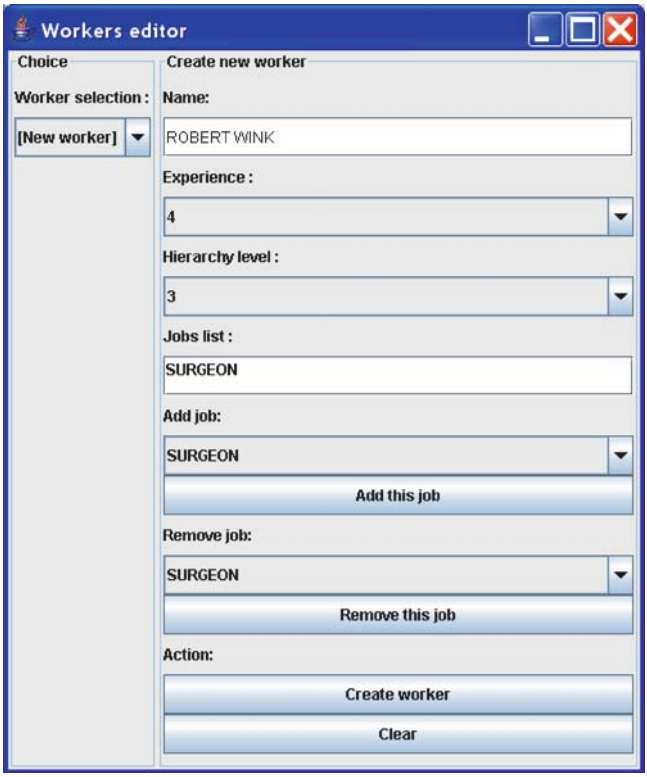

Fig. 2.5 Workers editor

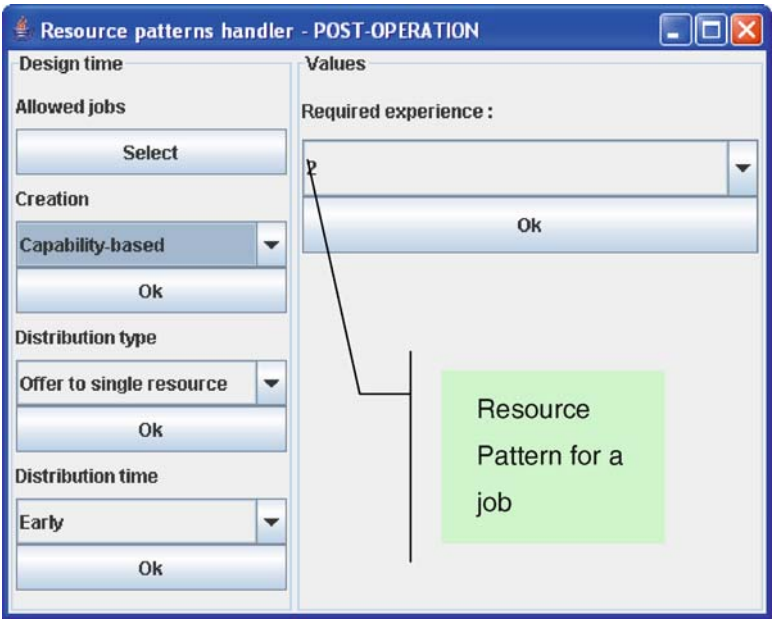

Fig. 2.6 Resource patterns editor

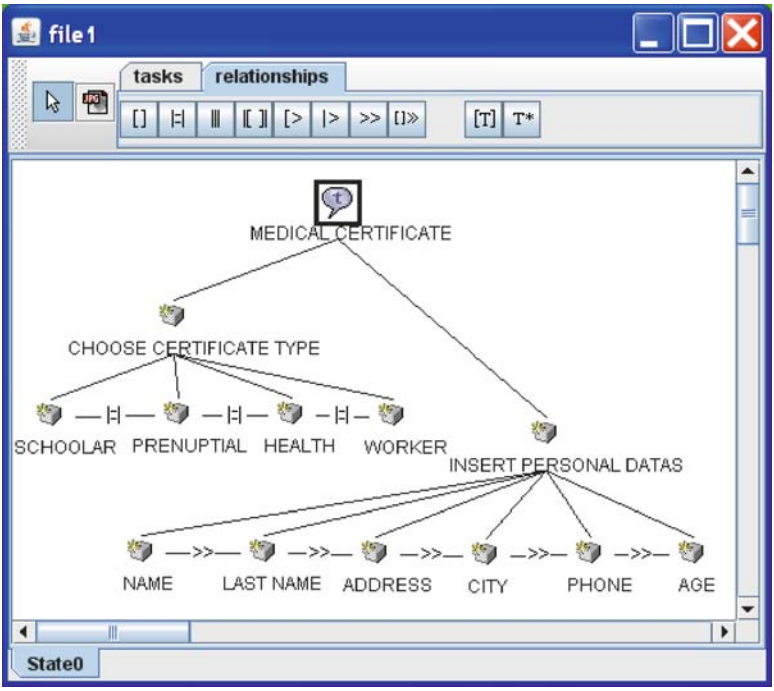

Fig. 2.7 Task model editor

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).

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],

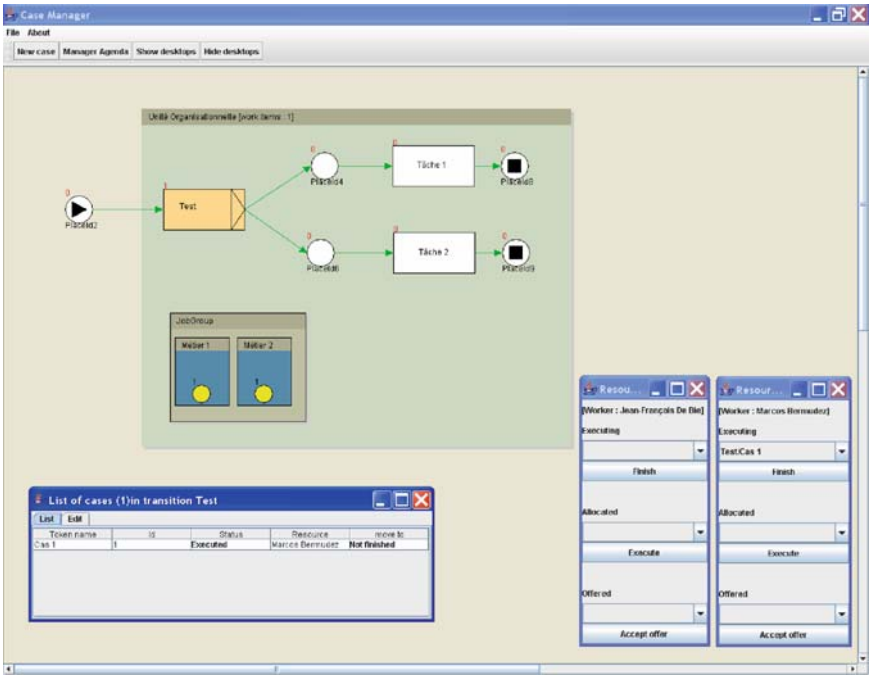

Fig. 2.9 Workflow manager tool

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.