Towards a Library of Workflow User Interface Patterns

Josefina Guerrero García[1], Jean Vanderdonckt[1], Juan Manuel González Calleros[1], and Marco Winckler[1,2]

[1]Université catholique de Louvain, Belgian Laboratory of Computer-Human Interaction, Place des Doyens, 1 – B-1348 Louvain-la-Neuve, Belgium
[2] IHCS-IRIT, Université Paul Sabatier, 118 route de Narbonne, Toulouse F-31062, France
{josefina.guerrero, juan.gonzalez}@student.uclouvain.be, jean.vanderdonckt@uclouvain.be, winckler@irit.fr

Abstract. A collection of user interface design patterns for workflow information systems is presented. Each Workflow User Interface Pattern (WUIP) is characterized by properties expressed in the PLML markup language for expressing patterns and augmented by additional attributes and models attached to the pattern: the abstract user interface and the corresponding task model. These models are specified in a User Interface Description Language. All WUIPs are stored in a library and can be retrieved within a workflow editor that links each workflow pattern to its corresponding WUIP, thus giving rise to a user interface for each workflow pattern. The software then gathers these UIs and the ones corresponding to workflow tasks into a user interface flow, a new concept introduced for specifying the intertwining of interfaces used by workers and the workflow manager in a single workflow.

Keywords: design pattern, user interface description language, user interface flow, workflow information system, workflow model editor, WUIP.

1 Introduction

Workflow is defined as the automation of business process. It allows better alignment of Information Technology (IT) with business because organization applications can be expressed in a way that makes sense to business users. Business users are the organization's resources who are performing work, accomplishing business goals. Assigning tasks to resources is complicated due to the different levels of skills they have, for instance: experience or availability to do the task. To address the allocation problem, a collection of workflow resource patterns [9] has been identified that provide the manner in which tasks are allocated or offered to resources. Generally a resource needs an agenda to handle their tasks, and a manager needs to control the way tasks are assigned and their progress. A Workflow Information System (WfIS) is a system that defines, creates and manages the execution of workflows through the use of software; the users of a WfIS interact with it through its user interfaces (UIs).

Developing UIs for WfIS represents new challenges today, not only for its diversity but also because user interaction takes place in two different logical levels synchronously. Interaction at the higher level means the manager specifying and designing the system, for that purpose UIs for workflow resource patterns are needed; in addition,

T.C.N. Graham and P. Palanque (Eds.): DSVIS 2008, LNCS 5136, pp. 96–101, 2008.

managers needs a UI monitoring workflow execution. Interaction at the lower level, resources are carrying out (UIs are needed for the actual execution of tasks) their allocated tasks (UIs with users agendas) whose current status is then communicated to the manager.

This paper aims to define a library of UI patterns for WfIS addressing the aforementioned challenges; the library is intended to represent the largest collection as possible of UI design patterns that are applicable to workflow resources patterns in a WfIS. The paper is organized as follows: Section 2 reports the state of the art, Section 3 describes the methodology for creating *workflow UI patterns* (WUIP), Section 4 explains how these WUIP could be then interpreted in terms of a WfIS. Section 5 presents how to link the UIs generated. Section 6 presents a conclusion of this work and some future avenues.

2 State of the Art

A *pattern* is referred to as "the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts" [8]. A *design pattern* systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems [2]. A UI design pattern is a particular instantiation of the pattern concepts in Human-Computer Interaction (HCI). According to the Pattern Language Markup Language (PLML) that resulted from two ACM CHI workshops aimed at defining a common format for UI patterns, a pattern is typically characterized by: a meaningful short name, an alternate name (alias), a general description of the problem, and the solution. It also gives implementation hints and examples. Many interesting works have been achieved that resulted in UI pattern collections (www.cs.kent.ac.uk/people/staff/saf/patterns/plml.html). In HCI, UIs have been subject to the pattern-based approach [10], but also other aspects such as domain, task, dialog, and abstract UI patterns have been considered successfully [7,10,14].

Workflow patterns refer specifically to recurrent problems and proven solutions related to the development of WfIS in particular, and more broadly, of process-oriented applications. On the one hand, several languages have been proposed for designing, specifying, and verifying workflow processes and patterns, and on the other hand, there are many commercial workflow management systems where control-flow and data-flow are well addressed. Workflow resource patterns have been identified that capture the different manners in which resources are presented and used in workflows [9]. The rationale for identifying these patterns was the need to master the many ways according which work can be distributed. The researchers have developed a web site (http://www. workflowpatterns.com/patterns/resource/) that contains descriptions and examples of theses patterns, along with supporting tool (YAWL), papers and evaluations of how workflow products support the patterns. However, not all considers mechanisms for resource handling and they all lack from UI generation from workflow specifications.

In order to structure the development cycle of a workflow UI, we are relying on FlowiXML [4], a structured method for developing UIs of a WfIS that advocates the automation of business processes according to a model-driven engineering approach based on the requirements and processes of the organization. Model-driven UI design [1, 5, 6] is intended to assist designers to obtain UIs with a formal method, preferably

one that is computer-supported. Several works have addressed the specific need for modeling UI for workflows [11, 12], all of them adopting a model-based approach but none of them generating UIs.

3 Developing User Interfaces for Workflow Information Systems

The methodology is applicable: i) to integrate human and machines based activities, in particular those involving interaction with IT applications and tools; and ii) to identify how tasks are structured, who perform them, what their relative order is, how they are offered or assigned, how tasks are tracked.

In this section, only an overview of this method is provided, for the complete definition of the semantics and the syntax, we refer to [4, 13]. The underlying conceptual model is composed of four models: *workflow*, *process*, *task*, and *organization*. The workflow model is recursively decomposed into processes which are in turn decomposed into tasks. A *process model* indicates the ordering of processes in time, space, and resources. Each process gives rise to a process model structured and ordered with process operators. Process operators determine whether the flow of work is sequential, parallel split, exclusive choice or multiple choices, with the corresponding merger operators, synchronization and simple merge. A *task model* represents a decomposition of tasks into sub-tasks linked with task relationships. Transformations are applied in cascade through the workflow layers using a mapping model. In order to support the mapping between the layers, predefined relationships were used: reification, decomposition, isExecutedIn, etc. [13].

By exploiting task models description, different solution scenarios can be modeled [4]. Each scenario represents a particular sequence of actions to be performed. Task models do not impose any particular implementation so that user tasks can be better analyzed without implementation constraints; it is, even possible to analyze user activities. Finally, the UI is derived from scenarios extracted from task models using a transformational approach [3]. FlowiXML is compliant with the Cameleon Reference Framework [1] for developing multi-target UIs.

4 Workflow User Interface Patterns

After having defined the methodological context in the previous section, this section will introduce, define, and explore the original concept of Workflow User Interface Pattern (WUIP).

We adopted the following methodology for defining the WUIPs:

1. *Augmented UI pattern definition*: from each workflow resource pattern a WUIP is created and consistently described through PLML attributes. In addition to those attributes, we also introduced the following fields that we believed that were missing from the version 1.1 of PLML: strengths, weaknesses, opportunities, and threads (according to a SWOT analysis that is missing because PLML only incorporates forces), a category, an evidence scale (from 0=no evidence supports the pattern to 5=two or more experiments support the pattern), a taxonomy of links between patterns (e.g., X uses Y in its solution, X is a variant of Pattern Y, X has a similar problem as Y, X is related in the related patterns section to Y, X specializes Y, X connects to Y), bibliographic reference, domains of human activity.

2. *Incorporation in the model-driven engineering method*: for each initial pattern definition resulting from the previous step, a task model has been specified using CTT notation [6]. This task model may serve as task patterns for WfIS like they serve in related work [7, 14].
3. *Final WUIPs*: from the task model resulting from the previous steps, an abstract UI and, consequently, a concrete UI have been defined in terms of the User Interface Description Language (here, UsiXML) so as to form corresponding abstract and concrete UI models. These two pairs of models have then been attached to the current pattern definition to finally obtain a complete WUIP. Each aspect of the abstract or concrete UI that tackles some concept incorporated in the model-driven engineering method can now be expressed in terms of the expanded UIDL.

Applying the above methodology resulted in 43 WUIPs [3]. We give below only a snapshot of some of these patterns for facilitating the understanding and for illustration purpose. Also, to support the application of the 43 WUIPs, a special module has been developed in Java and incorporated in our workflow model editor, see Fig. 1. This module B) enables the designer, while modeling the general workflow, to retrieve any WUIP from the library, to configure it, and to automatically incorporate it in the current model. Therefore, instead of redefining the complete pattern in terms of model elements found in the model editor (the workflow is defined by Petri nets), the application of a WUIP automatically includes the corresponding definition in the model and generates the corresponding UsiXML files for the UI that has been predefined for each WUIP and for defining the workflow (being itself entirely described in UsiXML).

Deferred allocation pattern – The ability to defer specifying the identity of the resource that will execute a task until runtime. Fig. 1 B reproduces how the pattern is retrieved from the library at design-time and precisely defined in the workflow editor (Fig, 1 A).

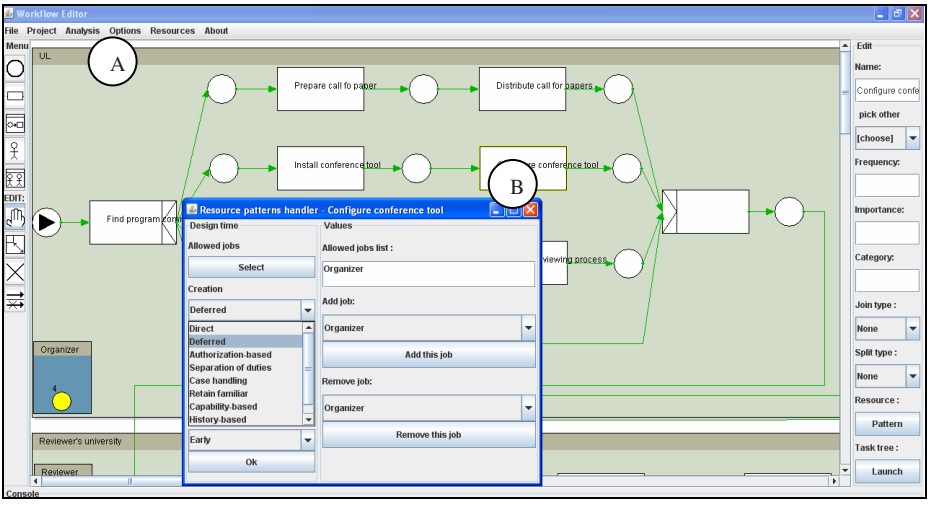

Fig. 1. Workflow resource pattern in design phase of workflow

5 Linking All User Interfaces

After having defined the UIs corresponding to the workflow patterns through the WUIP mechanism, we still need to produce UIs corresponding to tasks found in the process. These UIs can be produced by any appropriate method, such as [5, 6]. After that, 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 a new concept introduced in this paper; the *user Interface flow*. During the execution of work, information passes from one resource to another as tasks are finished or delegated; in our method we use an agenda assigned to each resource to manage the tasks 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.

A *User Interface Flow* is defined as an octuple UIF (A, Σ, U, T, δ, ω, ai, ao) where (Fig. 2 depicts it graphically):

- 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 $\forall$ a $\in$ A: $\exists$! u $\in$ U † is used by (a,u) [unique] or $\exists$ $u_1, u_2 \ldots u_n \in$ U † is used by $\{a, u_1, u_2 \ldots u_n\}$ [a is shared among $u_1, u_2 \ldots u_n$].
- 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
- a_i is the initial AC [$a_i \in$ A], a_o is the final AC [$a_o \in$ A, $a_o \neq a_i$]

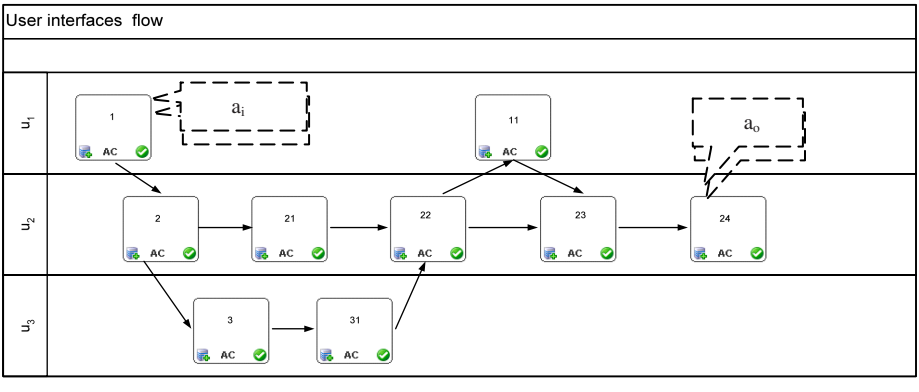

Fig. 2. User interface flow

6 Conclusion

This paper introduced a library of user interface design patterns that are particularly applicable to user interfaces of workflow information systems. Each pattern is compatible

with the literature and has been integrated in a workflow model editor. Designers are able now to specificity resource allocation patterns using UIs that fits: both at design-time (when everything is clear) and at run-time (when design decisions were postponed and manager must decide how to allocate the task), considering constraints imposed by mutually excluded patterns (for instance, once a task has been directed allocated it can not be defined as deferred). Of course, these specifications can be edited before producing the system code. The results of the modeling phase with respect to the UI viewpoint introduces a the concept of user interface flow we have formally defined and illustrated including how various users involved in the workflow will collaborate and their corresponding user interface. Finally, from our previous work, we are benefit from its capability to automatically generate UIs from specifications for both the workflow model (in this way, it is no longer needed to redraw the definition of the pattern in terms of places and transitions) and the user interface model (in this way, it is no longer needed to specify again the UI supporting the workflow pattern).

References

1. 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), 289–308 (2003)
2. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1994)
3. Guerrero, J., Vanderdonckt, J.: Workflow user interfaces patterns. Working Paper IAG 08/08, Université catholique de Louvain, Louvain-la-Neuve (2008)
4. Guerrero, J., Vanderdonckt, J.: FlowiXML: a Step towards Designing Workflow Management Systems. Journal of Web Engineering 4(2) (2008)
5. Kristiansen, R., Trætteberg, H.: Model-Based User Interface Design in the Context of Workflow Models. In: Proc. of Tamodia 2007, pp. 227–239. Springer, Berlin (2007)
6. Paternò, F.: Model-based design and evaluation of interactive applications. Applied Computing. Springer, Berlin (1999)
7. Radeke, F., Forbrig, P., Seffah, A., Sinnig, D.: PIM Tool: Support for Pattern-Driven and Model-Based UI Development. In: Proc. of Tamodia 2006, pp. 82–96. Springer, Heidelberg (2006)
8. Riehle, D., Züllighoven, H.: Understanding and Using Patterns in Software Development. Theory and Practice of Object Systems 2(1), 3–13 (1996)
9. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Berlin (2005)
10. Seffah, A., Gaffar, A.: Model-based user interface engineering with design patterns. Journal of Systems and Software 80(8), 1408–1422 (2007)
11. Stavness, N., Schneider, K.A.: Supporting Flexible Business Processes with a Progression Model. In: Proc. of MBUI 2004. CEUR Workshop, January 13, 2004, vol. 103 (2004)
12. Stolze, M.: Riand, Ph., Wallace, M., Heath, T.: Agile Development of Workflow Applications with Interpreted Task Models. In: Proc. of Tamodia 2007, pp. 2–14. Springer, Heidelberg (2007)
13. UsiXML, http://www.usixml.org
14. Wurdel, M., Forbrig, P., Radhakrishnan, T., Sinnig, D.: Patterns for Task- and Dialog-Modeling. In: Proc. of HCI International 2007, vol. 1, pp. 1226–1235 (2007)