Workflow User Interfaces Patterns.

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

ABSTRACT

A collection of user interface design patterns for workflow information systems is presented that contains forty three resource patterns classified in seven categories. These categories and their corresponding patterns have been logically identified from the task life cycle based on offering and allocation operations. 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.

RESUMEN

Este trabajo presenta una colección de patrones de diseño de interfaces de usuario para sistemas de información para el flujo de trabajo; la colección incluye cuarenta y tres patrones clasificados en siete categorías identificados a partir de la lógica del ciclo de vida de la tarea sobre la base de la oferta y la asignación de tareas a los responsables de realizarlas (i.e. recursos humanos) durante el flujo de trabajo. Cada patrón de la interfaz de usuario de flujo de trabajo (WUIP, por sus siglas en inglés) se caracteriza por las propiedades expresadas en el lenguaje PLML para expresar patrones y complementado por otros atributos y modelos que se adjuntan a dicho modelo: la interfaz de usuario abstracta y el modelo de tareas correspondiente. Estos modelos se especifican en un lenguaje de descripción de interfaces de usuario. Todos los WUIPs se almacenan en una biblioteca y se pueden recuperar a través de un editor de flujo de trabajo que vincula a cada patrón de asignación de trabajo a su WUIP correspondiente.

Recibido: 10 d Enero de 2012
Aceptado: 14 de Febrero de 2012

INTRODUCTION

Today's organization increasingly prompted to integrate their business processes and to automate the largest portion possible of them. A common term used to reflect the automation of these processes and to ensure the control over them is *workflow*. The Workflow Management Coalition [22] defines workflow as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules". A possible automation of the business process is the introduction of Information Systems (IS) to perform, control and define business process.

Palabras clave:
Diseño de patrones; lenguaje de descripción de interfaces de usuario; flujo de interfaces de usuario
Keywords:
Design pattern; user interface description language; user interface flow.

The challenges to have a method to support the development of an IS from a workflow specification are considerable. Considering just the specification of the required User Interfaces (UIs) for a workflow system UIs are needed, at least, for:

1. **Execution of work** . This UI allows the performance of the business process, for instance, on an ecommerce system buy a book.
2. **Resource allocation** . This UI allows the specification of how resources will be assigned to the different tasks they have to perform, either during the design of the workflow system or in run-time, while the workflow is executed.

*Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, 14 Sur y Av. San Claudio, Edificio 136, Ciudad Universitaria (CU), c.p. 72570 , Puebla, Mexico. E-mail: {jguerrero,juan.gonzalez}@cs.buap.mx
**Louvain Interaction Lab, Louvain School of Management, Université Catholique de Louvain, Place des Doyens I, B-1348, Louvain-la-Neuve, Belgium. E-mail:jean.vanderdonckt@uclouvain.be

3. **Control of workflow** . Managers can be benefit to have a global overview on how the work is executed, for that purpose; they need a UI showing how is the evolution of the workflow.

4. **User's agendas** . This UI is assigned to each resource in the organization indicating the list of task they have to perform.

5. **Work lists** . Managers may want to know what the current status of a process is, for that purpose they need a UI showing the worklist of the complete system.

Due to the large amount of existing workflow products, especially all the commercial software that are equipped today with many different capabilities including resources handling, we came to a point where it is very difficult to compare these capabilities on a common scheme. Fortunately, a collection of workflow patterns has been identified that provide the basis for an in-depth comparison of commercially available workflow systems.*Control-flow patterns* [21] indicate basic routing constructs of the process in a workflow, such as: sequence, parallel split, synchronization and exclusive choice. From a data perspective, there is a series of characteristics that occur repeatedly in different workflow modeling paradigms. *Workflow data patterns* [15] are aimed at capturing the various ways in which data is represented and used in workflows. Finally, *Workflow resource patterns* [16] correspond to the manner in which tasks are allocated to resources, that is any entity that is capable of achieving some work unit. As we focus on resource allocation, workflow resource patterns present a relevant theoretical foundation on how UIs can be implemented.

The goal of this paper is to associate a default UI to the resources patterns each pattern [16] that will result in a set of UI patterns for resource handling in a workflow information system. The paper is organized as follows: next section reports related work, the following section describes the methodology that we used for creating *workflow UI patterns* (WUIP) that are the cornerstone of this contribution. After that, we explain how these WUIP could be then interpreted in terms of a workflow information system. The section afterwards exemplifies how to apply these WUIP on a real-world case study with software developed for this purpose. Finally, we present a conclusion of this work and some future avenues.

STATE OF THE ART

When resources are modeled in workflow information systems we can separate them in two categories: 1) for simulation, i.e., resources are just numbers and parameters for equations that will evaluate resources performance [14]; 2) for execution, task execution is simulated and resources are allocated to task from a database [20]. However, from the workflow specified on such a system and the way resources are assigned there is no knowledge about how to implement an information system for that workflow. Particularly, what kind of User Interfaces (UI) needed to handle those patterns.

The rationale for identifying these patterns was the need to master the many ways according which work can be distributed [16]. The patterns are grouped into seven categories: creation patterns, push patterns, pull patterns, detour patterns, auto-start patterns, visibility patterns, and multiple resource patterns. The researchers have developed a web site (http://www. workflowpatterns.com/patterns/resource/) that contains descriptions and examples of these patterns, along with supporting tool, papers and evaluations of how workflow products support the patterns. However, existing methods just described theoretically patterns but they do not provide any knowledge or guidance on how such patterns could be implemented in an Information System (IS). For that purpose, we explore a systematic manner to develop UIs for each workflow resource pattern following its current definition [16], then, we identify the interaction needed to manipulate such pattern and express it into a task model. Relying on UsiXML [[10]] method to derive UIs from task models a set of UIs patterns is obtained for the workflow patterns. Even that several works have addressed the specific need for modeling UI specifically for workflows, all of them adopting a model-based approach: a workflow model [3], a progression model [18], and an adapted model-driven engineering method based on agile development [19]. The present work largely contrasts with these approaches in that the essence of what needs to be modeled in such systems is already captured in appropriate WUIP and in that the resulting patterns, unlike PLML, are augmented by the corresponding models.

So far, we have introduced a specification language [3] to support a method [4][6] for the development of UIs from workflow specifications. Another tool have been developed to generate UIs for: agendas, worklists and workflow control, have been introduced in [4][5][6][9]. However, resource allocation is still an open issue, not just for us but also we are not aware of a related work on this topic.

DEVELOPING A USER INTERFACE FOR A WORKFLOW INFORMATION SYSTEM

In order to structure the development life cycle of a workflow UI, we are relying on FlowiXML [3], a structured method for developing UIs of a workflow information system that advocates the automation of business processes according to a model-driven engineering approach based on the requirements and processes of the organization. The methodology is applicable in order: i) to integrate human and machines based activities, in par-ticular those involving interaction with information technology 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 in order to pave the way for WUIPs in the next section. For the complete definition of the semantics and the syntax, we refer to [3] and to www.usixml.org/index.php?mod=pages&id=40. The un-derlying method 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 or-dering of business 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.

By exploiting task models description, different solution scenarios can be modeled. 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. 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. (for more details, see www.usixml.org). As in any model-driven engineering approach, all the components are models, transformations between models and relationships that are described in terms of a meta-model, here a UML class diagram. When combining workflow and task models, the boundaries of each model must be specified with concept criteria in order to avoid de-sign mistakes. Finally, UIs are derived from scenarios extracted from task models using a transformational approach. FlowiXML is compliant with the Cameleon Reference Framework [1] for developing multi-target UIs, which is decomposed into four steps:

1. **Task and domain modeling** (Computing Independent Model in MDA): a model is provided for the end user's task, the domain of activity and, if needed, the context of use (user, computing platform, and environment).

2. **Abstract User Interface modeling** (Platform Independent Model in MDA): this level describes UIs independently of any interaction modality and any implementation.

3. **Concrete User Interface modeling** (Platform Specific Model in MDA): this level describes a potential UI after a particular interaction modality has been selected (e.g., graphical, vocal, multimodal). This step is supported by several tools helping designers and developers to edit, build, or sketch a user interface.

4. **Final User Interface** : this level is reached when the UI code is produced by interpretation (e.g., by rendering) or compiled (e.g., by static code generation).

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). Since the concept of patterns has been introduced by Christopher Alexander, who defined them in the architectural context, many attempts have been made to document patterns thanks to a pattern language and to gather them in a library. The discipline of Human-Computer Interaction (HCI) does not escape from this initiative. A pattern is referred to as "the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts" [13]. 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 HCI. According to the Pattern Language Markup Language (PLML), aimed at defining a common format for UI patterns. A pattern is typically characterized by: an identifier, a meaningful short name, an alternate name (alias), a general description of the problem (synopsis), the solution, and its consequences (strengths, weakness, opportunities, and threads). It also gives implementation hints and examples. Many interesting works have been achieved that resulted into UI

pattern collections, most of them expressed in PLML (www.cs.kent.ac.uk/people/staff/saf/-patterns/plml.html). In HCI, UIs have been subject to the pattern-based approach [17], but also other aspects such as domain, task, dialog, and abstract UI patterns have been considered successfully [12],[17].

A workflow pattern is another instantiation of a general design pattern. Workflow patterns refer specifically to recurrent problems and proven solutions related to the development of workflow information systems 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, but they all suffer from a lack in the resource consideration and from a whole regarding the UI aspects.

Workflow resource patterns

In any organization, resources are brought and consumed in order to achieve the tasks that are the organization's responsibility. These resources can be: human (e.g., personnel), material (e.g., hardware, printer) or immaterial (e.g., network bandwidth). A human resource has a specific position and privileges. In terms of the organizational hierarchy, each resource may have a number of specific relationships with the other resources. Material or immaterial resources may be durable or consumable in nature. Of a particular interest from a resource perspective is the manner in which tasks are advertised to specific resources for execution. For this purpose, it is necessary to consider the task life cycle, which refers to a series of states that a task goes through and related transitions. The task life cycle (Fig. 1) structures workflow patterns [15][16] into seven categories:

1. **Creation patterns:** Represent all possible ways used for defining a task (define in Fig. 1) in a process. Defining a task consists in specifying its goals, pre-condition, post-conditions, required skills, and required resources. Once a task has been properly defined, it can be effectively executed.

2. **Push patterns:** Represent all possible ways for making a task ready to start (offer, allocate in Fig. 1). A task is said to be started when the resource to which the task has been allocated has initiated its execution. A task may be allocated to such a resource, but starts later on. If this allocation is not straightforward, the task can be offered to a single resource or to multiple resources. Once allocated, a task could be delegated to another resource (e.g., due to unavailability). If the resource which delegated the task wants to receive the results in return, the task is then returned. Otherwise, it can start directly.

3. **Pull patterns:** Represent all possible ways for starting a task (start in Fig. 1) and finishing it (run, redo, finish, review, undo, repeat in Fig. 1).

4. **Detour patterns:** Represent all possible digressions from the normal execution path. This may include the case when a task is suspended and resumed, when a task is cancelled, or fails. This may also include de-allocation (return to task created, task offered or task allocated states).

5. **Auto-start patterns:** Represent all possible ways to start the task automatically after creation. This is particularly the case for automated processes (e.g., batch files).

6. **Visibility patterns:** Represent all possible ways for configuring the visibility of allocated task in a process. For instance, a particular resource is responsible for execution a task, but its progress should be made visible to others.

7. **Multiple resource patterns:** Represent all possible ways to involve additional resources than the one initially allocated (e.g., in case of overload).

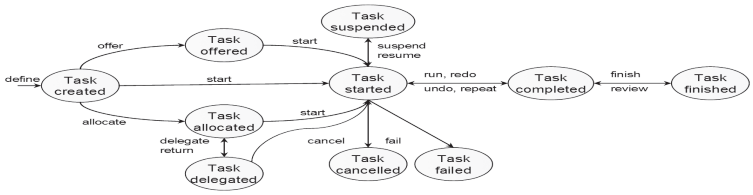

Figure 1 . Task life cycle.

Methodology

Model-based UI design [1],[11] is intended to assist designers in obtaining UIs with a formal method, preferably one that is computer-supported; model-based tools have been investigated since the late

1980's. The goal of these tools is to allow the designer to specify the UI at a level of abstraction that is independent from any implementation. The specification is usually shared between a set of components, called models, each model representing a facet of the interface characteristics. The number and type of theses models is different from one approach to another. The following methodology [5] for defining the WUIPs is adopted:

1. **Augmented UI pattern definition:** From each workflow resource pattern belonging to the seven aforementioned categories, 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 [11] in order to depict the pattern at design-time or at run-time. For this purpose, new concepts have been introduced in the semantics: new relationships between tasks (e.g., a task is grafted on another one means that a task is dynamically added in the task model at a certain node), new concepts (e.g., resource, process, and workflow), and inter-model relationship to propagate modifications between these concepts.

3. **Final WUIPs:** From the two task models resulting from the previous steps, abstract UIs and, consequently, concrete UIs have been defined in terms of the UIDL (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 UIDL.

Applying the above methodology resulted in 43 WUIPs. We give below only a snapshot of some of these patterns for facilitating the understanding and for illustration purpose.

Name:	Direct allocation
Identifier:	R-DA
Synopsis:	The ability to specify at design time the identity of the resource that will execute a task.
Strengths:	To prevent the problem of non-suitable allocation.
Weakness:	No opportunity to change the resource if he is not available to perform the task.
Opportunities:	To ensure task is routed to specific resource.
Problem:	This pattern effectively defines a static binding of tasks to a single resource.
Solution:	Probably the use of deadline and escalation mechanisms when the resource becomes overload and cannot deal with his assigned workload in a reasonable timeframe.
Example:	"Ask reviewers preferences" task must only be undertaken by "Joshua Brown".

In figure 2 the task model for such slection and its corresponding UI is presented.

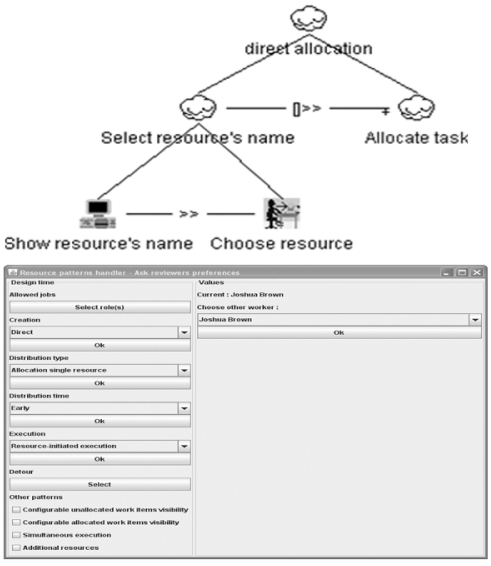

Figure 2 . Direct allocation pattern.

CASE STUDY AND TOOL SUPPORT

In order to support the application of WUIPs, a special module has been developed in Java and incorporated in our workflow model editor. This module 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).

The case study analyzes how people organize the program of small conferences by using a review tool. To organize a conference it is necessary identify the tasks, the jobs and resources which are involved. These tasks were assigned to each job taking into account the role that each of them plays in the workflow. The next table resume the principal tasks and the job in charge of develop them. Also, it is necessary specify the resources that are available for doing the work and where they are working, i.e. the organizational unit. Author resources are named as A-1, A-2, and A-3 because we do not know who will be interested in submit a paper.

Within the workflow editor, we can represent with rectangles each organizational unit, i.e. UL, Reviewer's university, and Author's university. After, we can add the different jobs and resources (workers) involve in the organizational conference. Also, we can have other attributes for the resource (worker) as the level of experience, hierarchy level. Also we have a representation (small rectangles) of each job assigned to each organizational unit, at the same time, is possible to have the number of resources (small dots with a number) available for develop tasks. After the identification of tasks, jobs, and resources, it is possible to assign tasks to resources applying the workflow resources patterns.

This assignation was elaborated after the analysis of the characteristics (qualifications, skills, abilities, experience, and hierarchy level) of each resource and considering the requirements of each task. For instance, *Install conference* tool task was assigned to *Ellen Martin* because she is engineering.

Table 1 .

Task and jobs identification.

ID	Task	Job		
		Organizer	Reviewer	Author
1	Find the program committee.	X		
2	Prepare the call for paper.	X		
3	Distribute the call for paper.	X		
4	Install conference tool.	X		
5	Configure conference tool.	X		

This representation is also possible in the workflow editor. First, we select the job and the type of workflow resource pattern, after we select the resource (worker) that will develop the task.

Table 2 .

Assigning tasks to resources.

Task	Job	Resource	Pattern
Find the program committee	Organizer	Chloé Lambin	Direct allocation
Prepare the call for paper	Organizer	Jacques Khelil	Capability based
Distribute the call for paper	Organizer	Jacques Khelil	Retain familiar
Install conference tool	Organizer	Ellen Martin	Capability based
Configure conference tool	Organizer	Ellen Martin	Retain familiar

In this case study, we show the workflow resource patterns that can be used during the design phase of the workflow. As you can see in the appendix there are some other workflow resource patterns that apply only in the execution phase. Figure 3 shows as small part of the workflow modeled with Petri net when the pattern has been selected.

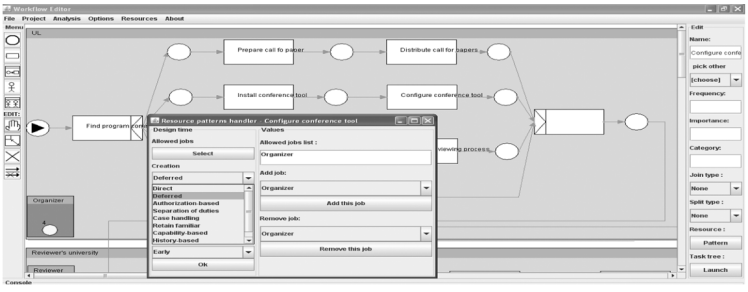

Figure 3 . Workflow resource pattern in design phase of workflow.

CONCLUSION

This paper introduced a library of user interface design patterns that are particularly applicable to user interfaces of workflow information systems. We have proposed an approach where FlowiXML, a model-based approach to develop user interface, is used in the context of workflow systems to develop WUIPs.

We rely on a proved method to generate User Interfaces, UsiXML, passing from task model and abstract user interface. We do not provide anything regarding the workflow resource patterns. However, the existing literature guided us to define its corresponding UI patterns. Each pattern can be selected in a workflow model editor so as to automatically generate the 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 as it is needed using Petri nets for modeling workflows) and the user interface model (in this way, it is no longer needed to specify again the UI supporting the workflow pattern).

ACKNOWLEDGMENT

We Acknowledge the Faculty of Computer Sciences of the University of Puebla to support this research.

REFERENCES

[1] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. (2003). A Unifying Reference Framework for Multi-Target User Interfaces.*Interacting with Computers,* 5(3):289-308.

[2] Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison-Wesley.

[3] Guerrero, J., Vanderdonckt, J. (2008). FlowiXML: a Step towards Designing Workflow Management Systems, *Journal of Web Engineering,* 4(2).

[4] Guerrero, J., Vanderdonckt, J., Gonzalez, J.M., Winckler, M. (2008). Modeling User Interfaces to Workflow Information Systems.*Proc. of 4th International Conference on Autonomic and Autonomous Systems ICAS'2008.*

[5] Guerrero, J., Vanderdonckt, J., González Calleros, J.M., Winckler, M. (2008). Towards a Library of Workflow User Interface Patterns. *Proc. of the International Conference on Design, Specification and Verification of Interactive Systems DSV-IS'2008,* Lecture Notes in Computer Science.

[6] Guerrero, J., Lemaigre, Ch., Vanderdonckt, J., González Calleros, J.M. (2008). Model-driven engineering of Workflow User Interfaces. *Proc. of Seventh International Conference on Computer-Aided Design of User Interfaces, CADUI'2008,* Lecture Notes in Computer Science.

[7] Guerrero, J., Vanderdonckt, J., Lemaigre, Ch. (2007). Identification Criteria in Task Modeling. In *1st IFIP Human-Computer Interaction Symposium HCIS'2008.*

[8] Kristiansen, R., Trætteberg, H. (2007). Model-Based User Interface Design in the Context of Workflow Models. In *Proc. of Tamodia'2007.* Springer, Berlin, pp. 227-239.

[9] Lemaigre, Ch., Guerrero, J., Vanderdonckt, J. (2008). Interface Model Elicitation from Textual Scenarios. Submitted to *1st IFIP Human-Computer Interaction Symposium HCIS'2008.*

[10] Limbourg, Q., Vanderdonckt, J. (2004). UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence. *Engineering Advanced Web Applications,* Rinton Press, Paramus, pp.325-338 .

[11] Paternò, F. (1999). Model-based design and evaluation of interactive applications, *Applied Computing.* Springer, Berlin.

[12] Radeke, F., Forbrig, P., Seffah, A., Sinnig, D. (2006). PIM Tool: Support for Pattern-Driven and Model-Based UI Development. *Proc. of Tamodia'2006,* pp. 82-96.

[13] Riehle, D., Züllighoven, H. (1996). Understanding and Using Patterns in Software Development. *Theory and Practice of Object Systems,* 2(1):3-13.

[14] Rockwell Software. (2000). *Arena Basic Edition user's Guide* , Rockwell Software Inc.

[15] Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst..W.M.P., (2004). Workflow Data Patterns.*BETA Working Paper Series, WP 127.* Eindhoven University of Technology.

[16] Russell N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D. (2005). Workflow Resource Patterns: Identification, Representation and Tool Support. *Proc. of 17th Conf. on Advanced Information Systems Egineering CAISE'05,* Lecture Notes in Computer Science, 3520:216-232.

[17] Seffah, A., Gaffar, A. (2007). Model-based user interface engineering with design patterns. *Journal of Systems and Software,* 80(8):1408-1422 .

[18] Stavness, N., Schneider, K.A. (2004). Supporting Flexible Business Processes with a Progression Model. *Proc. of MBUI 200, CEUR Workshop Proc. 103.*

[19] Stolze, M., Riand, Ph., Wallace, M., Heath, T. (2007). Agile Development of Workflow Applications with Interpreted Task Models. *Proc. of Tamodia'2007,* pp. 2-14.

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

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

[22] WfMC Terminology & Glossary. (1999). Workflow Management Coalition. *Document Number WFMC-TC-1011.* , ver. 3.0.

[23] Wurdel, M., Forbrig, P., Radhakrishnan, T., Sinnig, D. (2007). Patterns for Task- and Dialog-Modeling. *Proc. of HCI International'2007,* 1:1226-1235.