Context-Aware Workflow Management*

Liliana Ardissono, Roberto Furnari, Anna Goy, Giovanna Petrone, and Marino Segnan

Dipartimento di Informatica - Università di Torino
Corso Svizzera 185, 10149 Torino - Italy
{liliana,furnari,goy,giovanna,marino}@di.unito.it

Abstract. We describe the CAWE framework for the management of context-aware workflow systems, based on Web Services. The framework is based on a hierarchical workflow representation supporting a synthetic and extensible specification of context-sensitive workflows, which can be executed by standard workflow engines. We have exploited the CAWE framework to develop a prototype application handling a medical guideline specifying the activities to be performed in order to monitor patients treated with blood thinners.

1 Introduction

We present the CAWE (Context-Aware Workflow Execution) framework for the development of context-aware applications based on Web Service technologies, which adapt to the user features and the execution context. In order to support a flexible adaptation of the business logic, we have adopted a workflow-based approach, which enables the declarative specification of such logic. The CAWE framework is based on a hierarchical representation of the workflow, which specifies alternative, context-dependent courses of action, and on a declarative specification of the conditions determining the runtime selection of the appropriate context-dependent part of the workflow.

In order to test the functionality offered by our framework, we developed a prototype system and we have instantiated it on the execution of medical workflows involving the cooperation of human actors playing different roles. The system may be accessed from the internet, by using a PC or a Smart Phone client. The resulting medical application supports the home assistance of patients affected by heart diseases and coordinates the activities of doctors, patient and other technical and administrative personnel during the execution of the clinical guidelines to be applied.

2 Dimensions in Context-Aware Adaptation

Often, context-aware applications proposing personalized products/services to their users do not consider (at least explicitly) the fact that there are two different types of *roles*: the beneficiary of the product/service (henceforth, the *service-user*) and the user who interacts with the system to get the product/service (*interacting-user*). For instance, in an adaptive e-commerce application, a person could buy a product on behalf

* This work is supported by the EU (project WS-Diamond, grant IST-516933) and by MIUR (project QuaDRAnTIS).

L. Baresi, P. Fraternali, and G.-J. Houben (Eds.): ICWE 2007, LNCS 4607, pp. 47–52, 2007.

of somebody else [1]. Although the same person can play different roles, it is useful to distinguish them. In fact, both the service-user and the interacting-user can be the targets of adaptation, but the features to be taken into account, and what is adapted, differ in the two cases. For instance, a different product might be proposed, depending on the service-user's preferences; however, the User Interface (UI) should be adapted to the device utilized by the interacting-user to connect to the system. The CAWE framework supports UI adaptation, targeted to the interacting-user's context, and workflow adaptation, based on the context of the service-user; the latter is the focus of this paper.

2.1 UI and Workflow Adaptation in the CAWE Framework

In the CAWE framework, the interaction is tailored to the features of the device used to connect to the system (e.g., screen size), and to the role of the interacting-user (e.g., a relative or a doctor), by generating UI pages with different content and layout.

The adaptation of the workflow, described in detail in the following, consists of actuating different courses of action (activity paths) on the basis of the features of the context of execution. These context features belong to the service-user context. For instance, the clinical guideline of our medical application prescribes blood tests which can be performed at home or at the blood test lab, depending on the fact that the patient (service-user) is movable or not. The two alternative behaviors are associated to different activity paths, as the first one requires that a nurse visits the patient at home, while the second can be managed by taking the patient to the lab.

2.2 Service-User's Context in Our Medical Application

In order to provide a more concrete idea of the service-user context, we briefly decribe its representation within our medical application. The service-user is the patient, who benefits from the home assistance service. The patient's context includes long-term and short-term information to separate stable data from features which might change during the service execution.

The *long-term* information includes:
(a) The *personal context:* this corresponds to the clinical record, which includes features such as the patient id, contact information, gender, age, prescribed therapy, date of the most recent blood test, and so forth.
(b) The *environment:* this includes features such as the available resources (i.e., the medical equipment available at home).

The *short-term* information includes:
(a) The *Personal context:* this includes physiological data (e.g., blood pressure, hearth pulsation, ...), the patient's mobility state (the "movable" feature specifies whether she can be transported by car, or she needs an ambulance), and the degree of urgency in treatments due to the patient's health state.
(b) The *Environment:* it includes the "social context" (i.e., the people who next to the patient), which can influence the course of action to be selected.

3 The CAWE Framework

The architecture of the CAWE framework includes various modules, wrapped by Web Service interfaces. Specifically, the context-aware workflow execution is supported by two modules: the Context-Aware Workflow Manager and the Context Manager. The Context-Aware Workflow Manager runs a workflow engine which executes the workflow activities by following the flow specification. However, when it encounters a context-dependent portion of the workflow, the engine invokes a Personalization Module which applies adaptation strategies to select the course of action to be performed. The Context Manager provides the other modules with contextual information during the execution of the application.

3.1 Workflow Representation

In order to represent the context-dependent parts of the workflow, we introduce an abstraction hierarchy whose higher-level elements describe the activities to be performed in generic way. Specifically, we introduce the concept of *abstract activity* to denote an activity schema which does not directly specify a piece of business logic of the application. The actions to be executed in order to complete the abstract activity are selected at runtime, depending on the context state.

Moreover, we define *abstract workflow* a workflow schema including at least one abstract activity: the workflow abstracts from the details of execution of at least one (context-dependent) activity. If a workflow does not contain any abstract activity, it represents a *concrete workflow* and it can be executed by the workflow engine without invoking the Personalization Module.

Finally, each abstract activity is associated with a set of *context-dependent implementations*, representing the alternative courses of action which the workflow engine should execute, depending on the context. Each context-dependent implementation is a (possibly abstract) workflow and represents a subprocess to be performed in order to achieve the results of the abstract activity.

Figure 1 depicts the abstract workflow of our medical application, which includes the following activities (abstract/concrete activity start with an uppercase/lowercase letters):

1. A doctor starts the workflow by setting the first blood test that the patient has to undergo (*setFirstBloodTest(patient, date)*).
2. The application books a blood test with a lab at the specified date (*BookBloodTest(patient, date)*). Then, it evaluates the time interval between the current date and the date of the blood test (*eval(interval)*).
3. If the patient's health state is regular, she waits until the date of the test before doing any actions (*onAlarm(date)*). However, if warning symptoms, e.g., bleeding or fainting, occur before that date (*onMessage ...*), the urgency feature within the patient's context is set to "high" (*setUrgency(patient, "high")*), and the patient skips the wait. This is represented by means of a *pick* scope which includes the two competing courses of action.

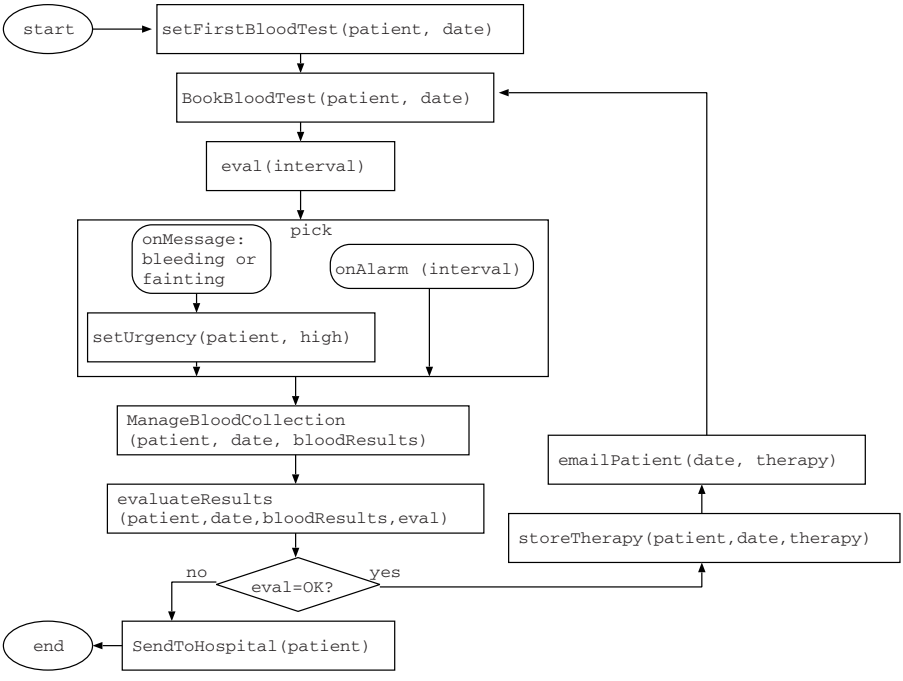

Fig. 1. Abstract Workflow of Medical Guideline

4. At the specified date, or immediately after the occurrence of a warning symptom, a sample of blood is taken from the patient for the lab. Then, the lab analyzes the blood sample and returns the values of the measured blood parameters (*ManageBloodCollection(patient, date, bloodResults)*).
5. A doctor evaluates the results that are then stored, together with the doctor evaluation, into the patient's context (*evaluateResults(patient, date, bloodResults, eval)*).
6. If the test results are good (*eval=OK?*), the doctor selects the date for the next blood test and sets the therapy to be followed until then (*storeTherapy(patient, date, therapy)*). Moreover, the application sends the patient an e-mail with the needed information (*emailPatient(date, therapy)*) and the flow restarts from item 2. Otherwise, the patient is notified to go to the hospital for further analyses (*SendToHospital(patient)*).

Figure 2 shows the workflow specification of a context-dependent implementation suitable for patients who may be taken to the lab by car. The applicability condition of this implementation is *movable=true*, where *movable* is a context variable included in the short-term patient's context. The application requests the appointment with the lab (*requestLabApp(patient, date)*), by invoking the Blood Test Lab Service[1], which returns a notification of the appointment, specifying time and place where to show up

[1] We assume that the actors involved in the workflow have Web Service interfaces.

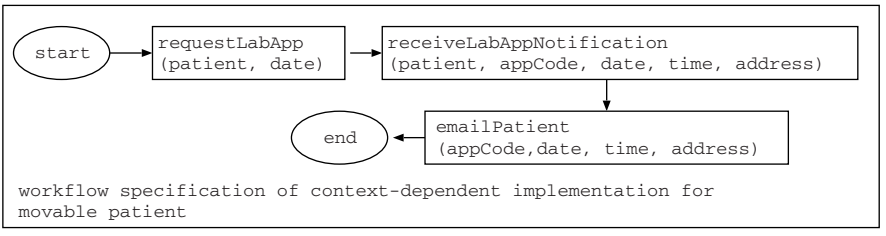

Fig. 2. Worflow Specifications of one of the Context-Dependent Implementations of Abstract Activity "BookBloodTest"

(*receiveLabAppNotification(patient, appCode, date, time, address)*). In turn, the application notifies the patient about the appointment (*emailPatient(appCode, date, time, address)*).[2]

3.2 Context-Aware Workflow Execution

The workflow engine wrapped by the Context-Aware Workflow Manager creates an instance of the abstract workflow each time the starting activity of the workflow is performed for the first time. The context-aware execution of the workflow is based on the selection of paths, depending on decision points and on the selection of the concrete implementations of the abstract activities to be performed. When the workflow engine has to execute an abstract activity, it first invokes the Personalization Module, which retrieves the context-dependent implementations of the abstract activity and returns the one to be executed, together with the bindings for its variables. Then, the workflow engine executes the selected context-dependent implementation as a subprocess of the main process instance. At subprocess completion, the control returns to the abstract workflow, which moves to the execution of the next activity.

The Personalization Module, invoked on an abstract activity, works as follows: first of all, if the abstract activity is associated to more than one context-dependent implementation, the module evaluates the applicability conditions of each candidate and it selects one of the applicable implementations for execution. Then, the module binds the context-dependent variables of the implementation to their current values, provided by the Context Manager.

4 Related Work

The introduction of context awareness in workflow systems is mainly focused on Quality of Service management (e.g., [2]) and on the adaptation to the user's device (e.g., [4]). An interesting approach in the area of Web Engineering is presented in [3], where the authors propose an extension to WebML to model multi-channel, context-aware Web applications.

[2] Another context-dependent implementation, not shown for space reasons, specifies that, if the patient is non-movable, a nurse is requested, to collect the patient's blood at home.

In this paper, we attempt to look at workflow systems from a more general perspective, and we propose a framework for the management of applications which adapt to user preferences and requirements, as well as to context-aware aspects such as the physical environment or the available resources.

The introduction of hierarchical workflows is usually related to the specification of compositional workflows. For instance, several workflow languages enable the designer to define complex activities which expand in workflows forming a composition hierarchy; e.g., see WS-BPEL [5] and process languages such as Petri Nets [6]. Our proposal differs from those approaches because it introduces a specialization hierarchy supporting the actuation of the same abstract activity in different ways.

5 Conclusions

We have presented the CAWE framework for the development of applications composing Web Service suppliers in complex, long-lasting workflows. CAWE supports the customization of the service offered by the application to the beneficiary of the service, taking the execution context into account; moreover, it personalizes the UI to the user(s) interacting with the application to carry out the assigned tasks.

The abstract activities and contextual conditions can be executed by a standard workflow engine because they simply include the invocation of a piece of code in their body. Indeed, the hierarchical workflow could be pre-processed by translating it to a flat workflow where abstract activities are replaced with decision points which fork on alternative workflow paths. However, our approach supports the conciseness and the readability of the resulting workflows, which could be very hard to understand, if represented as a flat graph including all the alternative courses of action. Moreover, it supports a seamless extension of the business logic of an application to take new contextual conditions and new courses of action into account. Finally, the introduction of the Personalization Module for the selection of context-dependent implementations supports the extension of the framework to the adoption of possibly complex personalization rules, without imposing restrictions on the workflow engines employed.

References

1. Ardissono, L., Goy, A.: Tailoring the interaction with users in Web stores. User Modeling and User-Adapted Interaction 10(4), 251–303 (2000)
2. Benlismane, D., Maamar, Z., Ghedira, C.: A view-based approach for tracking composite Web Services. In: Proc. of European Conference on Web Services (ECOWS-05), Växjö, Sweden, pp. 170–179 (2005)
3. Ceri, S., Daniel, F., Matera, M.: Extending webml for modeling multi-channel contextaware web applications. In: WISE - MMIS'03 IEEE Computer Society Workshop (2003)
4. Keidl, M., Kemper, A.: Towards context-aware adaptable Web Services. In: Proc. of 13th Int. World Wide Web Conference (WWW'2004), New York, pp. 55–65 (2004)
5. OASIS: OASIS Web Services Business Process Execution Language (2005), `http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel`
6. van der Aalst, W.: Making work flow: on the application of Petri Nets to Business Process Management. In: Proc. of 23rd Int. Conf. on Applications and Theory of Petri Nets, Adelaide, South Australia, pp. 1–22 (2002)