A SOA-based Model Supporting Adaptive Web-based Applications

L. Ardissono, R. Furnari, A. Goy, G. Petrone and M. Segnan
Dipartimento di Informatica - Università di Torino
Corso Svizzera 185, 10149 Torino - Italy
{liliana,furnari,goy,giovanna,marino}@di.unito.it

Abstract

Service Oriented Architecture (SOA) supports the integration of distributed and heterogeneous services by offering Web Service composition standards which describe service composition as a business process. However, the composition model proposed in SOA does not explicitly deal with personalization and context-awareness. In order to address such limitations, we have developed the CAWE conceptual framework. In this paper, we describe the personalization support offered by our framework to the development of context-aware, adaptive web-based systems. CAWE handles the integration of heterogeneous services and information sources; moreover, it manages long-lasting interactions with multiple cooperating users and it personalizes the business logic of the application by adapting the workflow activities to be carried out.

1. Introduction

In the last decade, personalization has become a central feature of Web-based systems, in order to address the "one-size-fits-all" issue and to support the "one-to-one" interaction with the user. User modeling and adaptation techniques have thus been developed in order to personalize interactive systems; e.g., see [7]. Moreover, adaptation has been extended to deal with context information [1]. The introduction of new adaptation features has gradually increased the complexity of applications, leading to the development of modular systems which integrate components offering advanced personalization features. However, such systems are typically based on closed architectures and they integrate homogeneous components in rigid business logics.

The introduction of Web Services [2], based on standard languages for the specification of operations (e.g., WSDL [17]), has offered an excellent opportunity to reuse adaptive systems in distributed applications. Indeed, a new personalization perspective is to employ the architectural flexibility offered by Service Oriented Architecture (SOA) [15] to develop *personalized distributed systems*.

In this paper we propose a specialized SOA architecture extending Web Service composition with personalization features. Our SOA architecture has been exploited to design the CAWE (Context Aware Workflow Execution) conceptual framework, which supports the development of context-aware, Web-based applications adapting the business logic and the interaction with the users to various types of context information.

Standard Web Service composition languages, such as WS-BPEL [14], represent the service composition as a business process. Thus, our approach involves the management of context-aware processes and their interaction with Intelligent User Interface components. Being based on a workflow model, our approach naturally supports the following types of interaction with the user:

- The coordination of human and automated actors filling different user roles.
- The management of interactions aimed at executing activities which take a long time to complete.
- The management of interruptions in the execution of activities.

The CAWE framework manages the adaptation to multiple users and to the surrounding context by extending the workflow engine with two modules: the first one customizes the business logic of the application and the second one personalizes the interaction with the user and generates a device-dependent User Interface. We have utilized the CAWE framework in the development of an e-service which coordinates doctors, nurses and administration staff in the execution of a medical guideline monitoring the health state of patients affected by heart diseases; see [5, 3].

In the following, Section 2 sketches the workflow adaptation features and presents the interaction support offered by our framework. Section 3 discusses our experience in applying the CAWE framework to the implementation of our e-service. Section 4 compares our approach to the related work and Section 5 concludes the paper.

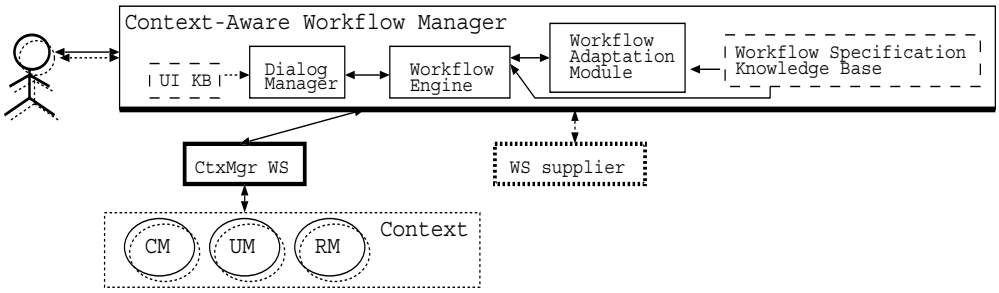

Figure 1. Architecture of the CAWE framework.

2. The CAWE conceptual framework

Figure 1 depicts the architecture of the CAWE framework. The main modules, listed in the following, offer standard Web Service interfaces, depicted as thick lines.

The *Context Manager Web Service (CtxMgr WS)* manages the information about the users of the application, the execution context and the environment surrounding the users; see Section 2.1. The *Context-Aware Workflow Manager* manages the interaction with the user and the business logic of the application taking the context information into account; see Sections 2.2 and 2.3. Other Web Service suppliers may be invoked.

2.1. Context information management

In order to support multi-user cooperation and to adapt the application accordingly, the CAWE framework structures the context information by explicitly modeling the actors (role fillers) participating to the service. During the execution of the application, the CtxMgr WS handles a Context including a Role Model, a User Model and a Context Model for each person actively or passively involved in the interaction with the application; see Figure 1:

- The Role Model (RM) stores domain-dependent, default information about the role; e.g., access rights. For instance, an important role in our e-Health application is the patient one, as the execution of the medical guideline is tailored to her health status.

- The User Model (UM) stores information about an individual user filling the role; e.g., the person filling the patient role. User Models include information such as expertise, preferences, and individual capabilities; e.g., physical and mental capabilities.

- The Context Model (CM) stores information about the context surrounding the user and relevant data about the execution context. For instance, the device utilized by the user, bandwidth, light conditions, and noise.

The RMs, UMs and CMs are represented as XML documents specifying a list of $< feature, value >$ pairs.

During the execution of the application, the components of the Context-Aware Workflow Manager interact with the CtxMgr WS (by means of SOAP messages invoking WSDL operations) in order to request/provide the CtxMgr WS with information about the users and their context. The CtxMgr WS initializes and updates the UMs and the CMs with information collected by interacting with the users via the User Interface of the application. Specifically, the Dialog Manager component of the Context-Aware Workflow Manager provides the CtxMgr WS with data about a user and her context; e.g., clickstream data, user information input by means of a form, and the device utilized by the user. The Context Models may also be updated with information collected by other information sources, such as sensors.

The preferred user modeling inference techniques, needed to acquire information about the user from her behavior, strongly depend on the application. Thus their definition is not part of the CAWE framework, but they should be selected at application design time. To this purpose, we have designed the CtxMgr WS as a Web Service, decoupled from the rest of the system. The modularity of the architecture supports a seamless integration of a user modeling component in the CtxMgr WS. Moreover, the Web Service might invoke a User Modeling Server [11] and other information sources to retrieve additional information about the users. For instance, in our e-Health application, the CtxMgr WS invokes a Clinical Record Manager WS to retrieve detailed information about the patient.

2.2. Adaptation of the business logic

The business logic of the application is adapted by selecting the courses of action to be enacted during the execution of the application. The selection is done by taking into account the users (User-Role Models) involved in the process and their Context Models.

- The business logic is represented as an *abstract workflow*, stored in the *Workflow Specification Knowledge*

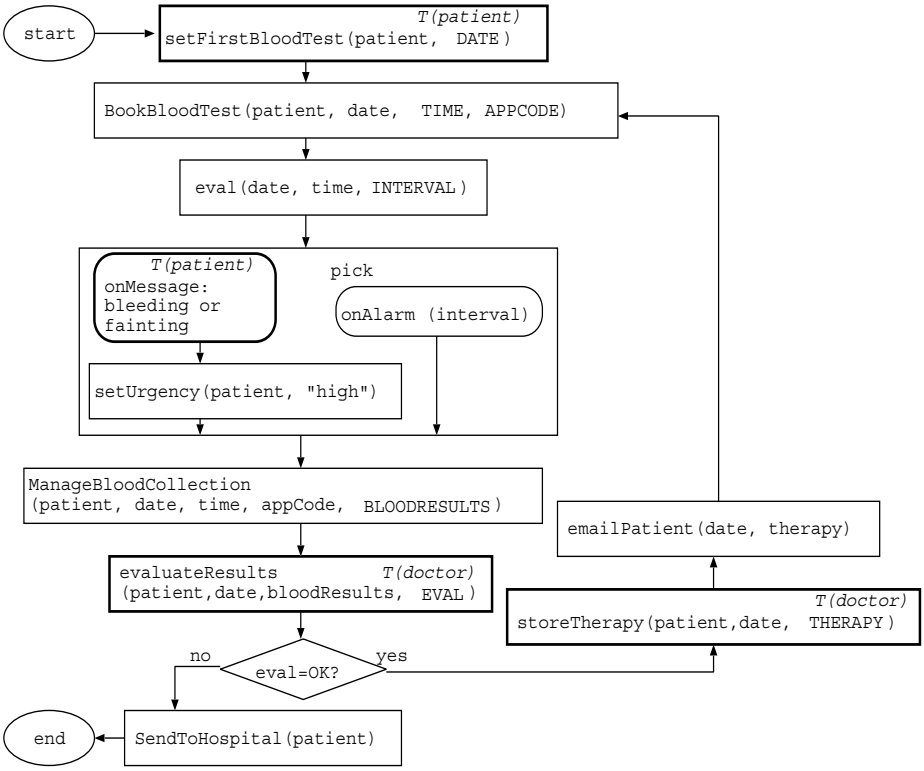

Figure 2. Abstract workflow describing the business logic of our e-Health application.

Base (see Figure 1), which separates the high-level description of the activities to be performed from their context-dependent details. In the following, we shortly describe the workflow representation and management; see [4] for technical information.

- An *abstract workflow* is a workflow whose activities may be either regular activities, or *abstract activities*. An abstract activity is a complex activity which can be exploded in alternative, context-dependent courses of action.[1] Each course of action is suitable to achieve the result of the abstract activity in a different contextual condition. To this purpose, the course of action has an *applicability condition* for the evaluation of its suitability to the current Context. A course of action may be a an abstract workflow itself; thus, hierarchies of abstract workflow may be defined.

- The *applicability conditions* are boolean conditions on the state of the UMs, RMs and CMs. The dot-notation is adopted to specify the model to be considered for the evaluation of the condition; e.g., $(\mathtt{UM}_{user_1}.\mathtt{f}_i{=}\mathtt{v}_1 \ \wedge \ \mathtt{CM}_{user_2}.\mathtt{f}_j{=}\mathtt{v}_2)$ is true if feature $\mathtt{f}_i$ of the $\mathtt{UM}_{user_1}$ model has value $\mathtt{v}_1$ and feature $\mathtt{f}_j$ of the $\mathtt{CM}_{user_2}$ model has value $\mathtt{v}_2$. Otherwise, the condition is false. Notice that the applicability conditions may concern any of the users involved in the workflow (not only those interacting with the system). Thus, the business logic can be adapted to the overall context. For instance, in our e-Health application, the patient's health conditions influence the business logic (execution of the medical guideline), although she does not necessarily interact with the system.

Figure 2 depicts the abstract workflow of our e-Health application: the abstract activity names start with an uppercase letter; e.g., `BookBloodTest`. Moreover, thick bordered rectangles represent tasks and the assigned roles `T(role)`. Figure 3 sketches the specification of two alternative courses of action for the execution of the `BookBloodTest` abstract activity. The specification includes the definition of the courses of action, the identifier (`ID`), the abstract activity (`Implements` field) and the `Applicability condition` field. For simplicity, the figure graphically sketches the workflows and it omits the fields reporting the input and output arguments of the courses of action and other technical information; see [4] for details.

Given the context-sensitive workflow specification, the Context-Aware Workflow Manager adapts the business

[1] The syntactic extensions to the workflow representation language are embedded in the bodies of the activities; therefore, they can be managed by a standard workflow engine. See [4].

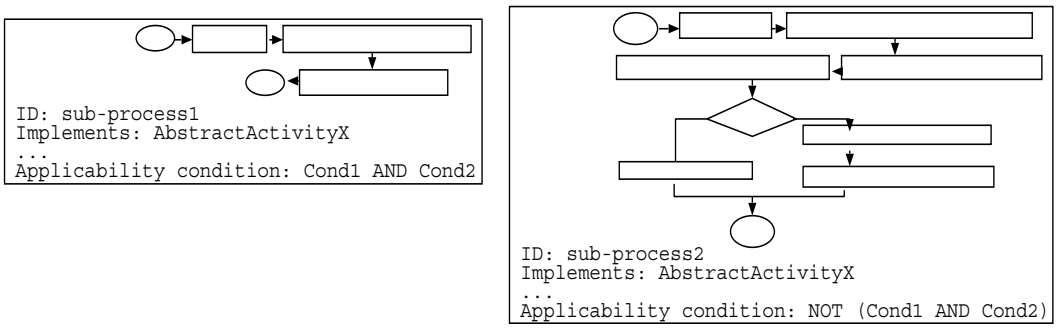

Figure 3. Alternative courses of action describing the context-aware execution of an abstract activity (`BookBloodTest`).

logic by retrieving the context information from the CtxMgr WS. As shown in Figure 1, the Context-Aware Workflow Manager includes a Workflow Engine and a Workflow Adaptation Module:

- The *Workflow Engine* executes the abstract workflow as if it were a standard workflow; however, when the Engine encounters an abstract activity, it invokes the *Workflow Adaptation Module* in order to retrieve the course of action to be enacted.
- When the *Workflow Adaptation Module* is invoked on an abstract activity, it evaluates the applicability conditions of the courses of action implementing the activity in order to identify the one suitable for the Context. To evaluate the conditions, the module interacts with the CtxMgr WS. For instance, in the execution of `BookBloodTest`, the Workflow Adaptation Module would select `MovBBT` if `UMpatient.movable` is true, `Not-MovBBT` otherwise.

The abstract workflows are organized in hierarchies which compactly represent different behaviors, therefore providing the flexibility required in order to handle context-aware interactions. For instance, different Web Service suppliers may be invoked to request ad hoc services. Moreover, multiple courses of action may be defined to represent operations which can be normally skipped, but should be performed in particular situations, or to represent different recipes for the achievement of a goal. As a concrete example, in our e-Health application, the steps to be performed in order to reserve a blood test are specified as two alternative courses of action of the `BookBloodTest` abstract activity. The first one has the `UMpatient.movable=true` applicability condition and it is suitable for patients who can go to the lab for the test by car. The second one has the opposite applicability condition and it includes additional steps aimed at organizing the collection of the blood sample at the patient's home.

2.3. Adaptation of the User Interface and of the interaction with the user

In workflow systems, communicative activities aimed at questioning the filler of a role and/or at presenting some information are specified as *tasks* assigned to the role. The task specifies the information to be asked/presented as a list of input/output parameters. When, during the workflow execution, the workflow engine encounters the task, it starts it and waits for its completion. The active tasks are asynchronously managed by a Task Manager, which waits that the target users connect to the application and then generates the User Interface (UI) pages on demand.

In the CAWE framework, the interaction with the user is carried out in context-aware mode by exploiting a Dialog Manager component which extends standard task management behavior with personalization features. Specifically, the Workflow Engine schedules the tasks to be completed in standard way, during the execution of the context-sensitive workflow which represents the business logic of the application. However, the task execution is controlled by the Dialog Manager module, which extends the default Task Manager by applying personalization rules aimed at generating context-dependent UI pages. When the user connects to the application in order to complete the task, the Dialog Manager generates one or more UI pages displaying the information specified in the task (both input and output parameters). The Dialog Manager interacts with the CtxMgr WS to retrieve the context information and handles the page generation as follows:

1. It selects the stylesheet (XSL) to be applied, depending on the user features and preferences, and on the characteristics of her device. The stylesheet specifies the layout properties; e.g., font size and background color.
2. It groups the information items to be displayed in subsets, in order to fit the size of the screen and to comply

with user features such as her receptivity and interests; see [3]. In this way, possibly more that one UI page will be presented in order to complete the task.

3. For each subset, it fills in an XML page template with the content to be displayed. Then, it applies (XSLT) the stylesheet to the filled template, in order to generate the page code. Notice that the stylesheet selection and the splitting of contents are tailored to the target user; therefore, the pages can be customized to each of the users cooperating to the service execution.

4. It cycles sending the generated pages to the user device and retrieving the responses, until it collects the last response (i.e., until all the input and output parameters specifyed in the task have been elicited/displayed).

5. It returns the collected information to the Workflow Engine, which ends the task and continues the workflow execution.

3. Implementation and preliminary evaluation

As a preliminary evaluation of the CAWE conceptual framework, we implemented its execution model and we applied it to a specific application. The CAWE prototype is implemented in the jBPM environment [12], which offers a graph-based workflow representation language supporting a clean definition of workflows, thanks to the fact that it is based on the Petri Net process model [16]. See [4] for implementation details.

We instantiated the CAWE prototype on an application supporting the management of a clinical guideline for the home assistance of patients affected by heart diseases. Clinical guidelines are an excellent testbed for multi-user adaptivity and context-awareness. In fact, they are long-lived processes, they involve actors playing different roles (e.g., doctors and administration staff) and they rely on external service providers, such as nursery and transportation services. Moreover, the activities to be performed vary depending on context conditions such as the patient's health status and mobility. Our e-Health application supports doctors, para-medical and administrative personnel by coordinating the completion of tasks and by performing automated activities. The application may be accessed from the internet, by using a PC or a Smart Phone client.

The development of our e-Health application provided us with the following feedback:

- As far as the business logic adaptation is concerned, the context-sensitive workflow representation formalism supports a synthetic description of a very different context-dependent behaviors. In fact, the applicability conditions of the courses of action are complex boolean conditions; moreover, the courses of action may be abstract workflows themselves and can be organized in specification hierarchies. Notice also that the declarative style of the representation formalism facilitates the introduction of new adaptive behaviors: a new course of action can be added to the Workflow Specification Knowledge Base without modifying the rest of the system.

- As far as the adaptation of the interaction is concerned, the workflow-based management of the interaction with the user does not support the high flexibility of traditional dialog systems, which may tailor each interaction turn on a goal basis. However, our approach suits the typical requirements of page-based interaction, as it supports the dynamic generation of personalized, device-dependent pages.

The development of the medical application also provided us with relevant feedback about the applicability of the framework when instantiating it on a specific application domain. In particular, the specification of the features characterizing the Role, User and Context Models is a standard task to be performed in personalized systems and requires limited technical skills. Moreover, thanks to the clarity of the jBPM representation language, the design of a context-sensitive workflow as a hierarchy of abstract workflows is not problematic for the designer, if the workflow hierarchy is moderately deep. In general, the hierarchical representation supports the specification of a synthetic workflow, more readable than the equivalent flat one, which becomes very large if several alternative behaviors are represented. Indeed, the specification of complex applicability conditions challenges the designer, who is required to be moderately familiar with logic. Currently, this task is even more difficult because the conditions have to be defined in internal format (dot-notation and boolean expressions). However, we plan to develop a design tool to the purpose.

4. Related work

Workflow systems mainly focus on Quality of Service (QoS) management and on the adaptation to the user's device; e.g., see [6], [10]. See also [8], which extends WebML to model multi-channel, context-aware Web applications. In comparison, the CAWE framework supports applications which adapt to user preferences (possibly including QoS), as well as to context-aware aspects such as the physical environment or the available resources. Moreover, the framework supports the adaptation to multiple cooperating users.

In the research about context-aware systems, user modeling ontologies have been introduced to allow services share information about the user, the device employed to interact with the system, etc.; e.g., GUMO [13]. As a matter

of fact, our Context Manager Web Service does not adhere to such ontologies (the User, Role and Context models are simple XML documents). However, as discussed in Section 2.1, the CAWE architecture is highly modular and the Web Service can be configured to invoke external servers for the management and processing of that kind of information.

Our work differs from other workflow-based adaptive systems (e.g., [9]) in the following aspects: first, it handles the adaptation to multiple users cooperating to the service execution, including users who do not interact with the application but are the recipients of the services. Second, our framework personalizes the workflow to the users and their context, while in [9] the workflow underlying the system behavior is the same for all the users and contexts.

Finally, hierarchical workflows are usually related to the specification of compositional workflows; e.g., see see WS-BPEL [14] and process languages such as Petri Nets [16]. Instead, our proposal introduces a specialization hierarchy supporting the actuation of the same abstract activity in different ways.

5. Conclusions

In this paper, we have described the architecture and the adaptation features offered by the CAWE conceptual framework, which supports the development of context-aware Web-based applications based on a Service Oriented Architecture (SOA). The applications based on this framework adapt the business logic, the interaction and the User Interface to a complex context which explicitly models multiple human actors involved in the service execution.

As a preliminary evaluation of the CAWE conceptual framework, we developed a prototype system supporting the execution of context-sensitive workflows and we utilized it to develop a Web-based e-service. This activity provided us with positive feedback concerning the adaptivity features offered by our framework. However, it highlighted the need of design tools supporting the administrator in the specification of context-sensitive workflows and personalization information. In our future work, we will develop such tools and we will test them with real users.

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

References

[1] G. Abowd and E. Mynatt. Charting past, present and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction, Special Issue on HCI in the new Millennium*, 7(1):29–58, 2000.

[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, architectures and applications*. Springer-Verlag, 2004.

[3] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan. Context-awareworkflow management. In *LNCS 4607, Web Engineering. Proc. of ICWE 2007*, pages 47–52, Berlin Heildelberg New York, 2007. Springer.

[4] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan. A framework for the management of context-aware workflow systems. In *Proc. of WEBIST 2007 - Third International Conference on Web Information Systems and Technologies*, pages 80–87, Barcelona, Spain, 2007.

[5] L. Ardissono, A. D. Leva, G. Petrone, M. Segnan, and M. Sonnessa. Adaptive medical workflow management for a context-dependent home healthcare assistance service. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 146(1):59–68, 2006.

[6] D. Benlismane, Z. Maamar, and C. Ghedira. A view-based approach for tracking composite Web Services. In *Proc. of European Conference on Web Services (ECOWS-05)*, pages 170–179, Växjö, Sweden, 2005.

[7] P. Brusilovsky, A. Kobsa, and W. Nejdl. *The Adaptive Web: Methods and Strategies of Web Personalization, LNCS, Vol. 4321*. Springer-Verlag, 2007.

[8] S. Ceri, F. Daniel, and M. Matera. Extending webml for modeling multi-channel contextaware web applications. In *WISE - MMIS'03 IEEE Computer Society Workshop*, 2003.

[9] S. Holden, J. Kay, J. Poon, and K. Yacef. Workflow-based personalized document delivery. *International Journal on E-Learning*, 4:131–148, 2005.

[10] M. Keidl and A. Kemper. Towards context-aware adaptable Web Services. In *Proc. of 13th Int. World Wide Web Conference (WWW'2004)*, pages 55–65, New York, 2004.

[11] A. Kobsa. Generic user modeling systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization, LNCS, Vol. 4321*, pages 136–154. Springer-Verlag, 2007.

[12] J. Koenig. JBoss jBPM white paper. http://www.jboss.com/pdf/jbpm_whitepaper.pdf, 2004.

[13] A. Krüger, J. Baus, D. Heckmann, M. Kruppa, and R. Wasinger. Web-based mobile guides. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization, LNCS, Vol. 4321*, page 521–549. Springer-Verlag, 2007.

[14] OASIS. OASIS Web Services Business Process Execution Language. http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel, 2005.

[15] M. Papazoglou and D. Georgakopoulos, editors. *Service-Oriented Computing*, volume 46. Communications of the ACM, 2003.

[16] W. van der Aalst. 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*, pages 1–22, Adelaide, South Australia, 2002.

[17] W3C. Web Services Definition Language. http://www.w3.org/TR/wsdl, 2002.