The CAWE Framework - Enhancing Service Oriented Architecture with Context Awareness Extended Abstract

L. Ardissono, R. Furnari, A. Goy, G. Petrone, M. Segnan
Dipartimento di Informatica
Università di Torino
Torino, Italy
Email: {liliana,furnari,goy,giovanna,marino}@di.unito.it

***Abstract*—The development of Web applications based on Service Oriented Architectures (SOA) is challenged by the lack of support to the specification of explicit context adaptation policies.**

As an answer to this issue, we present the Context Aware Workflow Execution framework (CAWE), which enriches SOA with (a) context-aware workflow management; (b) dialog management capabilities supporting the adaptation of the interaction with the individual user, and (c) context-dependent User Interface generation.

I. Introduction

Context-awareness is particularly important in Web applications, which are accessed by large numbers of users, having diverse preferences, needs and capabilities, and using heterogeneous devices to interact with the business services. In order to suitably handle such variability, a self-managing system should be able to adapt both the service and the User Interface to the individual user and to the dynamic environment surrounding her/him.

However, Service Oriented Architecture (SOA, [1]), the reference model for the development of composite applications, does not explicitly deal with personalization and context-awareness. In fact, it embeds all the adaptation decisions in the process specifying the business logic of the applications.

In order to address this limitation, we designed a vertical SOA architecture which extends Service Oriented Computing with context-awareness and personalization capabilities. This extended abstract shortly presents the CAWE (Context Aware Workflow Execution) framework for the development of composite Web applications. The framework supports the adaptation of the business logic, interaction logic and User Interface to the users and to their context. Specifically, the framework supports:

- The context-dependent selection of the courses of action to be enacted, and of the service providers to be invoked, during the execution of the application.
- The generation of a context-dependent User Interface, tailored to the user's device (e.g., to its screen size) and to the user's preferences (e.g., background colors and font size).
- The management of tasks as dialogs with the user, the provision of extra-helpful information for non-expert users, and the management of a User Interface fitting the size of her/his device.

We exploited the CAWE framework to develop an e-Health prototype application supporting the management of a clinical guideline which coordinates the activities to be performed in order to monitor the health state of patients affected by heart diseases. The analysis of the e-Health domain, and the development of the application, proved the suitability of the adaptive features offered by our framework, as well as its applicability to real-world use cases.

II. The CAWE Framework

Service Oriented Architecture provides limited support to and context-awareness because it fails to recognize the central role of the adaptation logic and thus it embeds all the adaptation decisions in the workflow specifying the business logic of the applications. Specifically:

- As far as the business logic is concerned, the workflow underlying the applications embeds the variables to be taken into account and describes the alternative courses of action in a flat graph. Although this approach works well in simple cases, it does not scale to complex contexts.
- The User Interface (UI) and the interaction with the user lack flexibility because they are based on minimalistic techniques for the generation of device-dependent UI pages which fail to support the management of flexible dialogs with the user.

The CAWE framework supports the development of composite applications which tailor the business logic, the interaction with the user and the User Interface to the user and to her/his context. The key concept is the fact that the adaptation logic has to be explicitly represented. By extracting such logic from the application workflow, flexible techniques can be applied to steer the system behavior. The CAWE architecture includes two core components:

- The *Context Manager service (CtxMgr WS)* handles the context information during the execution of the application. Specifically, it handles a Role Model for each role

defined in the application workflow, as well as a User Model and a Context Model for each involved actor.

- The *Context-Aware Workflow Manager (CA-WF-Mgr)* enacts a context-sensitive workflow which defines the business logic of the application. For this purpose, it exploits two modules: the Workflow Adaptation Module shapes the workflow depending on the context; the workflow engine enacts the resulting workflow.
  Within the CA-WF-Mgr, the *Dialog Manager* module acts as a bridge between the user and the workflow engine. When the user logs in the application, the Dialog Manager is invoked and takes the control of the interaction. The module adapts the User Interface to a context including both the user's device and her/his layout preferences.

A. business logic

In the CAWE framework, the business logic of an application is represented as a context-sensitive workflow organized in an abstraction hierarchy which specifies the system behavior at different levels of detail. Specifically:

- Besides the standard workflow activities (prescribing the invocation of service providers, the management of tasks, or some internal computation), a context-sensitive workflow can include *abstract activities* which describe a generic type of behavior, to be decided at runtime.
- Each *abstract activity* is associated with a set of *implementations* which describe different courses of action that the workflow engine should enact to complete the activity, depending on the context. Each implementation is a workflow which can specify rather different behaviors; e.g., starting a task to be performed by a human actor, invoking a Web Service, starting a complex subprocess, or carrying out some internal computation. Notice that an implementation may include itself some abstract activities; therefore, the context-sensitive workflow can be organized as a multi-level hierarchy.
- The *business logic adaptation policies* steer the selection of the implementations to be enacted during the execution of the abstract activities. These policies are described as (chains of) condition-action rules: the precondition of a rule is a boolean condition on context variables. The action can be the reference to another rule (rule chaining), or the name of the implementation to be enacted.

During the execution of the application, the workflow underlying the application is composed by recursively selecting the implementations of the abstract activities to be enacted, until the system's behavior is completely specified. This selection is steered by the business logic adaptation policies.

Specifically, the Context-Aware Workflow Manager wraps a workflow engine which executes the context-sensitive workflow as if it were a standard one. However, when the engine encounters an abstract activity, it invokes the Workflow Adaptation Module on the abstract activity. When the module returns the implementation to be enacted, the engine performs it as a subprocess of the main process instance. At subprocess completion, the engine resumes the execution of the main workflow.

B. user interface

The Dialog Manager handles each task to be completed as a communicative goal to be achieved by carrying out a dialog with the user. Each dialog step is aimed at achieving a part of the task and is managed by generating a UI page. A Finite state Automaton specifies the interaction logic of the Dialog Manager: each state of the automaton corresponds to a UI page type and each state transition is performed as a consequence of a user action. The automaton describes the whole interaction with the user as far as task management is concerned. Specifically, a task is handled as follows:

1) The Dialog Manager sends the user's browser a personalized UI page representing an interaction turn. The page includes a set of input/output parameters to be acquired/presented and the navigation links enabling the user to continue the interaction. Moreover, the page includes a set of help links to get more specific information about the task and its parameters.
2) The user may perform different actions on the UI page: e.g., each help link, and each information link associated to the parameters, activate a nested dialog. Moreover, suitable transitions lead to the next, or to the previous step of the dialog, respectively.
3) At task completion time, the Dialog Manager notifies the workflow engine and feeds it with the acquired data.

The generation of the personalised pages is based on the evaluation of a set of UI adaptation policies, which steer the selection of the layout to be applied (given the user's preferences and device) and, consequently, determine the maximum number of parameters which can be put in each UI page.

Figure 1 shows a sample UI page, targeted to a desktop device, generated by our e-Health application during the management of a task (`storeTherapy()`).

- The top bar of the page includes the name of the application (eHealth) and reports the username (house) and the logout button.
- The middle bar of the page is organized as follows:
  - The higher portion shows the task name, the task ID (744) and the user's role (doctor).
  - The lower portion includes: a help link for the visualization of the task description; the position of the current interaction turn within the overall dialog (Page 1 of 2); the *continue* link (>>) taking to the next dialog turn, and the Cancel link.
- The lower part of the page is devoted to the visualization of the input and output parameters of the task. Specifically, the Form area shows the input ones, while the Information area displays the output ones. Each parameter name has a link to its more specific information (`h`).

Figure 2 shows the sequence of pages devoted to the same task, if the user uses a PDA to connect to the application. In order to cope with the smaller screen size, the dialog is performed in more steps than in the desktop case.

Fig. 1. First dialog turn in the management of task `storeTheraphy`, tailored to a desktop device.

Fig. 2. First two dialog turns in the management of task `storeTheraphy`, tailored to a PDA.

III. Related Work

In Service Oriented Computing, some contributions extend standard Web Service composition languages with context-awareness features (e.g., C-BPEL; see [2]), in order to comply with Quality of Service (QoS) requirements. These approaches are affected by the limitations of standard Web Service composition languages, such as WS-BPEL ([3]) and its context-aware extensions (e.g., Context4BPEL, see [4]), which embed the adaptation logic in the workflow specification.

Our work overcomes the limitations of these works by introducing the abstract activities and by exploiting declarative adaptation rules for the runtime, context-dependent selection of the courses of action to be enacted. In this way, the business logic of the application is shaped during its execution.

In the Semantic Web research, planning technology is applied to enhance the flexibility in Web Service composition. Moreover, plan-based approaches are applied to invoke Web Service providers in context-aware mode; e.g., see [5], [6], [7]. However, planning technology is not suitable to handle long-lasting services and processes because it does not support persistence management. Therefore, up to now it has only been used to handle short-lived composition plans. In fact, several proposals for the adoption of planners in Web Service

composition turned out to exploit workflow engines for the service execution; e.g., see [8] and [9].

Concerning the management of the interaction with the user, context-aware workflow systems only provide the adaptation of the User Interface (UI) to the user's device, in terms of stylesheet selection; e.g., see [7] and [10]. In comparison, CAWE supports applications which adapt both the code of the UI pages and the interaction logic to a complex context. Moreover, it supports the adaptation to multiple users, by tailoring the UI and the interaction logic on an individual basis.

In the research about dialog-based systems, some researchers employed scripts describing domain-level activities and linguistic behavior to model articulated task-oriented dialogs; e.g., see [11]. Moreover, planning technology was applied to manage short-lived interactions with the user; e.g., see [12]. We adopt Finite State Automata to handle the interaction with the user; although these are less flexible than plans, they are more robust and lightweight, and they support a predictable behavior.

IV. Conclusion

This extended abstract has presented the Context Aware Workflow Execution framework for the development of context-aware composite Web applications. The framework enriches Service Oriented Architecture with (a) adaptation techniques supporting the execution of context-sensitive workflows; (b) dialog management capabilities supporting flexible user interactions, and (c) context-dependent User Interface generation techniques aimed at presenting personalized information on different devices. As such, it supports the development of Web applications which can self-adapt to meet the requirements of heterogeneous users in dynamic usage environments.

More information about the CAWE framework can be found in [13], [14], [15].

References

[1] M. Papazoglou and D. Georgakopoulos, Eds., *Service-Oriented Computing*. Communications of the ACM, 2003, vol. 46, no. 10.

[2] C. Ghedira and H. Mezni, "Through personalized web service composition specification: from bpel to c-bpel," *Electronic Notes in Theoretical Computer Science*, no. 146, pp. 117–132, 2006.

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

[4] M. Wieland, O. Kopp, D. Nicklas, and F. Leymann, "Towards context-aware workflows," in *Proc. Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS 2007) at CAiSE'07*, Trondheim, Norway, 2007.

[5] S. McIlraith, T. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46–53, 2001.

[6] W. Balke and M. Wagner, "Through different eyes - assessing multiple conceptual views for querying Web Services," in *Proc. of 13th Int. World Wide Web Conference (WWW'2004)*, New York, 2004.

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

[8] D. J. Mandell and S. A. McIlraith, "Adapting BPEL4WS for the Semantic Web: The bottom-up approach to Web Service interoperation," in *LNCS 2870, Proc. 2nd International Semantic Web Conf. (ISWC 2003)*. Sanibel Island, Florida: Springer-Verlag, 2003, pp. 227–241.

[9] G. Laures and K. Jank, "Adaptive Services Grid Deliverable D6.V-1. Reference architecture: requirements, current efforts and design," http://asg-platform.org/cgi-bin/twiki/view/Public/ReferenceArchitecture, Tech. Rep., 2005.

[10] 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.

[11] J. Chu-Carroll and S. Carberry, "Collaborative response generation in planning dialogues," *Computational Linguistics*, vol. 24, no. 3, pp. 355–400, 1998.

[12] C. Rich, D. McDonald, N. Lesh, and C. Sidner, "COLLAGEN: Java middleware for collaborative agents services with multiple suppliers," http://www.merl.com/projects/collagen, 2002.

[13] 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*, Barcelona, Spain, 2007, pp. 80–87.

[14] L. Ardissono, A. Goy, and G. Petrone, "A framework for the development of distributed, context-aware Adaptive Hypermedia applications," in *LNCS 5149, Adaptive Hypermedia and Adaptive Web-Based Systems, 5th Int. Conference, AH2008*, W. Nejdl, J. Kay, P. Pu, and E. Herder, Eds. Berlin Heildelberg New York: Springer-Verlag, 2008, pp. 259–262.

[15] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan, "A SOA-based model supporting adaptive web-based applications," in *Proc. of 3rd Conference on Internet and Web Applications and Services (ICIW 2008)*. Athens, Greece: IEEE, 2008, pp. 708–713.