A Framework for the Development of Distributed, Context-aware Adaptive Hypermedia Applications

Liliana Ardissono, Anna Goy, and Giovanna Petrone

Dipartimento di Informatica - Università di Torino
Corso Svizzera 185, 10149 Torino - Italy
{liliana,goy,giovanna}@di.unito.it
http://www.di.unito.it

Abstract. The CAWE framework supports the development of context-aware, Service Oriented applications which integrate heterogeneous services and customize the cooperation among multiple users. We present the techniques adopted in the framework to manage a context-sensitive interaction with the users.

1 Introduction

Service Oriented Architectures (SOA) [1] offer an excellent opportunity to develop distributed Adaptive Hypermedia applications but the composition model proposed in SOA does not explicitly deal with personalization. In order to address this issue, we have developed the CAWE (Context Aware Workflow Execution) conceptual framework, which extends standard Web Service composition with the management of context-aware processes and of Intelligent User Interface components. Being based on a workflow model, our approach advances the state of the art in Adaptive Hypermedia by supporting the following types of interaction with the user: a) the coordination of users playing different roles in the execution of a complex process; b) long lasting interaction sessions supporting the completion of operations which are not instantaneous; c) the management of interruptions in the execution of activities.

This paper presents the techniques adopted in the CAWE framework to handle context information (Section 2) and to adapt the interaction with the user accordingly (Section 3). In [2], we described the CAWE architecture and the context-aware management of the business logic of applications.

Role Model (RM):
  ID: String (RM identifier)
  currentUM: String (reference to the UM of the role filler in focus)
  UMList: list of IDs (references of the UMs of -all- the role fillers)
  FeatureList: (possibly void) sequence of FeatureType elements
  PermissionList: (possibly void) sequence of FeatureType elements
User Model (UM):
  ID: String (UM identifier)
  CMRef: String (reference to the CM associated to the UM)
  RMList: list of IDs (references of the RMs filled by the user)
  FeatureList: sequence of FeatureType elements
Context Model (CM):
  ID: String (CM identifier)
  UMRef: ID (reference to the UM of the user)
  FeatureList: sequence of FeatureType elements
FeatureType:
  featureName: String
  featureVal: String

Fig. 1. Structure of the RMs, UMs and CMs.

2 Context Information Management

The architecture of the CAWE framework, described in [2], includes a *Context Manager Web Service (CtxMgr WS)*, which manages the information about the users of the application, their environment and the execution context. In an application based on CAWE, the CtxMgr WS handles a Context including a Role Model for each role to be handled in the workflow, as well as a User Model and a Context Model for each person actively or passively involved in the interaction. Notice that a role may be filled by multiple interchangeable users, and a user may fill more than one role. Figure 1 shows the structure of the models, which are represented as XML documents.

- The Role Model (RM) stores the references of the User Models of the human actors filling the role. Moreover, it stores domain-dependent, default information about the role. The RM is utilized to assign workflow tasks to the human actors involved in the task execution, or indirectly influencing the business logic of the application. E.g., the patient plays an important role in an e-Health application, which must be tailored to her health status.
- The User Model (UM) stores information about an individual user filling one or more roles (e.g., 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 environment in which the user operates (e.g., light conditions), the device she utilizes during the interaction and other information such as the network bandwidth.

For generality purposes, the CAWE framework does not prescribe the introduction of specific features in the UMs, RMs and CMs; the application developer is free to define the information items relevant to a specific application.

During the execution of an instance of the application, serving a particular set of cooperating users, the CtxMgr WS instantiates the Role Models. Moreover, it updates the corresponding User Models and the Context Models with information collected by interacting with the users via the User Interface of the application; e.g., clickstream data, user information input by means of a form, and the device utilized by the user. The Context Models might also be updated with data collected by other information sources, such as sensors.

Notice that the CtxMgr WS is designed as a Web Service in order to decouple it from the other components of the framework. In this way, the framework enables the application developer to embed in the module, or to invoke, the user modeling and context management components most suitable to the application.

3 Context-aware Interaction Management

The architecture of the CAWE conceptual framework also includes a *Context-Aware Workflow Manager*, which adapts to the context the interaction with the user and the business logic of the application. As described in [3], the business logic is a hierarchical workflow specifying context-dependent courses of action, which are selected at execution time depending on context information.

In a workflow, the interaction with the user is specified by means of a *task* assigned to the user role. The task specifies the information to be asked/presented as a list of input/output parameters. When the workflow engine selects a task for execution, it delegates a Task Manager for its management. The Task Manager operates asynchronously: when the user connects to the application, it generates the User Interface (UI) pages presenting the input/output parameters.

In the CAWE framework, a *Dialog Manager* module extends the default Task Manager by applying personalization rules aimed at generating context-dependent UI pages. The Dialog Manager interacts with the CtxMgr WS in order to retrieve the context information and the values of input parameters that are available in the user's UM and CM, and can be visualized as default values. Then, the Dialog Manager generates one or more UI pages displaying the information specified in the task; the module generates the pages 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 the user's features.
3. For each subset, it fills in an XML page template with the content to be displayed. Then, it applies the stylesheet to the filled template. For each displayed parameter, it is possible to include a "more info" button linking a textual description of the data item. Notice that the splitting of contents and the stylesheet selection are tailored to the target user; therefore, the pages can be customized to each of the users cooperating to the service execution.

4. The Dialog Manager cycles sending the generated pages to the user's device and retrieving the responses, until all the input and output parameters specifyed in the task have been elicited/displayed.
5. Finally, the Dialog Manager returns the collected information to the Workflow Engine, which continues the workflow execution.

Our framework manages persistent and loosely coupled interactions with the users; in fact, the tasks are suspended until the users connect to the application. Moreover, the interaction with the user (within a task) can be suspended in order to handle interruptions, by saving the user session for later resumption.

4 Discussion

In this paper, we proposed the adoption of Service Oriented Architectures for the development of distributed AH applications integrating heterogeneous services and information sources. Moreover, we presented the adaptation features offered by CAWE framework in order to handle personalized interactions with the users.

The CAWE prototype [2] is implemented in the jBPM environment [4], which offers a workflow representation language based on the Petri Net process model. In order to test the framework, we instantiated the CAWE prototype on an application supporting the management of a clinical guideline. The application may be accessed by using a PC or a Smart Phone client. The development of our application showed that the workflow-based management of the interaction with the user does not support the flexibility of goal-based dialog systems. However, it suits the requirements of page-based interaction, as it supports the dynamic generation of device-dependent pages, whose contents are tailored to the interacting users, on the basis of their features and of the process activities.

Being based on workflow management, our work overcomes the limitations of planning technology, which does not support asynchronous, persistent interactions with the user. Moreover, our work advances the state of the art in context-aware workflows, which is mainly focused on Quality of Service (QoS) and on the adaptation to the user's device; e.g., see [5]. In fact, the CAWE framework supports the adaptation to multiple users who cooperate to the service execution, taking into account their preferences, requirements (e.g., QoS), and context features such as the physical environment or the available resources. Our work also differs from other workflow-based adaptive systems (e.g., [6]) in the following aspects: first, it handles the adaptation to multiple users cooperating to the service execution, including the indirect service recipients. Second, it personalizes the workflow to the users and their context, while in [6] the workflow underlying the system behavior is the same for all the users and contexts.

This work was funded by projects WS-Diamond (IST-516933) and QuaDRAnTIS (MUR).

References

1. Papazoglou, M., Georgakopoulos, D., eds.: Service-Oriented Computing. Volume 46. Communications of the ACM (2003)
2. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: A framework for the management of context-aware workflow systems. In: Proc. of WEBIST 2007, Barcelona, Spain (2007) 80–87
3. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: Context-aware workflow management. In: LNCS 4607, Web Engineering, Springer (2007) 47–52
4. Koenig, J.: JBoss jBPM white paper, http://www.jboss.com/pdf/jbpm_whitepaper.pdf (2004)
5. Benlismane, D., Maamar, Z., Ghedira, C.: A view-based approach for tracking composite Web Services. In: Proc. of ECOWS-05, Växjö, Sweden (2005) 170–179
6. Holden, S., Kay, J., Poon, J., Yacef, K.: Workflow-based personalized document delivery. International Journal on E-Learning **4** (2005) 131–148