The Context Aware Workflow Execution Framework

The Context Aware Workflow Execution Framework

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

Dipartimento di Informatica,
Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy
Fax: +39 011 751603, E-mail: name.surname@di.unito.it
*Corresponding author

Abstract: The development of self-adapting Web applications based on composite architectures, such as Service Oriented Architectures (SOA), is challenged by the lack of support to the specification of explicit adaptation policies for the context-aware management of the business, interaction and presentation logics.

In order to address this limitation, we propose a vertical architecture extending SOA with advanced adaptation features. This article presents the Context Aware Workflow Execution framework (CAWE), which enriches SOA with (a) context-aware workflow management; (b) dialogue management capabilities supporting the adaptation of the interaction with the user, and (c) context-dependent User Interface generation.

The article also briefly presents a prototype application developed by exploiting the CAWE framework.

Keywords: Context-aware adaptation; Service Oriented Architectures; Web applications; Context-dependent workflow execution

Biographical Notes: Liliana Ardissono is an Associate Professor at the Dipartimento di Informatica of the Università di Torino, where she obtained her University Degree and her Ph.D in Computer Science. Her research interests include User Modeling, Adaptive Hypermedia and Service Oriented Computing. She is Secretary of the Board of Directors of User Modeling Inc. and she is a member of the Editorial Board of User Modeling and User-Adapted Interaction - The Journal of Personalization Research.

Roberto Furnari received a Laurea Degree in Computer Science in 1999 at the Università di Torino. In 2007 he started its Ph.D. program in

Computer Science within the Doctoral School of Science and High Technology of the Università di Torino. His main research interests are in Web Service orchestrations and choreographies.

Anna Goy is Researcher at the Department of Computer Science of the Università di Torino, working in the area of Web-based systems. She completed her Ph.D. in Cognitive Science (Università di Torino) in 1998, in the area of lexical semantics. Since many years she works on distributed Web-based applications, Adaptive Hypermedia, and context-aware systems.

Giovanna Petrone is a researcher of Computer Science at the Università di Torino. Her research interests concern two main areas: Multi-agent systems (with specific interest for distributed systems and Web Services) and Intelligent User Interfaces (with specific attention to personalisation in Web-based services). Previously, she has worked for several years as a software engineer and architect in large US and Italian computer companies and she was also Visiting Scholar at the Stanford University.

Marino Segnan is a Researcher at the Computer Science Department, Università di Torino, working with the Advanced Service Architectures group. His recent research activities deal with interaction models for Web Services, choreographies, monitoring. His previous activity focused on the development of a Qualitative Simulation Tool for Model-Based Diagnosis. Previously, he worked with several companies, mainly on projects involving Integrated Development Environments, User Interfaces, compilers, Transaction Management.

1 Introduction

The research on self-adapting systems mainly deals with the management of Quality of Service, focusing on aspects such as load-balancing and failure recovery; e.g., see Ghedira and Mezni (2006), Benlismane *et al.* (2005), Ardagna and Pernici (2007), and Baresi *et al.* (2007). In contrast, the adaptation to the users and to their surrounding context has been somehow neglected so far. However, this feature has become particularly relevant, given the large number of systems which are now available on the Web. In fact, Web applications can be used in rather different contexts:

- With the large availability of broadband internet and wireless access, people use various types of devices, in different environments, to interact with the business services. Therefore, such services should tailor the User Interface and the interaction logic accordingly.
- Web applications are used by a large user population having diverse preferences and capabilities. Thus, they should adapt their business logic and the offered functions to specific requirements.

In order to improve the flexibility of such systems, the central role of the adaptation logic should be recognised. However, this has not happened yet in the development of systems based on composite architectures. As discussed in this article, Service Oriented Architecture (SOA, Papazoglou and Georgakopoulos (2003)), the reference model for the development of composite applications, does not explicitly deal

with 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 capabilities. This article 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, it 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 and preferences; e.g., background colours and font size.
- The management of tasks as dialogues with the user, supporting both the provision of extra-helpful information, and the management of a User Interface fitting the size of her/his device.

These capabilities are based on the following architectural features:

- An explicit representation of the context variables to be taken into account.
- A declarative representation of the business, interaction and presentation logics of the application.
- A declarative representation of the policies steering the adaptation of the service to the user and to her/his context.

The analysis of two real-world application domains (an e-Health one, and a travel one) proved the usefulness and the suitability of the adaptive features offered by the framework. Moreover, the development of a prototype Web Application in the first domain confirmed the applicability of the framework to real-world use cases.

The remainder of this article is organised as follows: Section 2 discusses the problem addressed in this work. Section 3 describes CAWE and Section 4 provides some technical details. Section 5 briefly presents our e-Health application, Section 6 positions our work in the related research and Section 7 concludes the article.

2 The problem

The research on User Modeling and Adaptive Hypermedia proposed techniques supporting the adaptation of applications to the user's preferences and characteristics; e.g., see Maybury and Brusilovsky (2002), and Brusilovsky *et al.* (2007). Moreover, the research on context-aware systems proposed techniques to model and manage context information in ubiquitous systems; e.g., see Abowd and Mynatt (2000), Dey and Abowd (2000), Dourish (2004), Baldauf *et al.* (2007) and Gross (2008). However, these techniques were applied to applications based on monolithic architectures and having a simple business logic. In contrast, complex systems, such as those which compose external services, adopt ad hoc solutions for context awareness purposes.

As a matter of fact, Service Oriented Architecture provides limited support to context awareness because it fails to recognise the central role of the adaptation logic. Thus, it embeds the adaptation decisions in the workflow defining the business logic of an application. Specifically:

- As far as the business logic is concerned, the workflow underlying the application 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 support the reactive composition of the business logic, depending on the evolution of a dynamic context. In fact, it fails to support the adoption of powerful decision making techniques to select the most appropriate system behaviour. Moreover, it leads to the specification of complex and articulated workflows, including a large number of paths describing the alternative courses of action to be enacted. This complexity makes the workflows hard to read and to modify, and challenges the specification of flexible adaptation strategies. On the contrary, Service Oriented Computing poses complex adaptation requirements concerning, e.g., the dynamic composition of Web Service providers on the basis of their availability and of various Quality of Service requirements.

- Similarly, the User Interface (UI) and the interaction with the user lack flexibility:

  - On the one hand, Web Service composition environments adapt the UI pages to the user by applying device-dependent stylesheets. However, they do not plan the distribution of content in the UI pages. In fact, they present the same content on any type of device, without taking its screen size, or other similar features, into account. This surface-level adaptation leads to the generation of sub-optimised UI pages which either force the user to do scrolling, or increase the length of the interaction during the task completion.

  - On the other hand, these environments only support the management of one-shot interactions in which, for each workflow task, a UI page is generated. Similar to the previous case, the strict association between tasks and pages may cause the generation of oversize pages, if the tasks are complex or the device has a small screen.

In order to address such limitations, we propose to extend Service Oriented Architecture with dialogue management capabilities supporting the dynamic selection of the content to be displayed during the interaction with the user. Moreover, we introduce the management of explicit adaptation policies, which can be exploited to steer both the selection of the business activities to be performed and the generation of the User Interface. The introduction of such policies has two main advantages: first of all, policy languages enable the specification of adaptation strategies based on the evaluation of complex conditions, thus supporting a fine-grained tuning of the system behaviour. Second, the adaptation strategies can be specified declaratively, supporting their revision and extension during the application life-cycle.

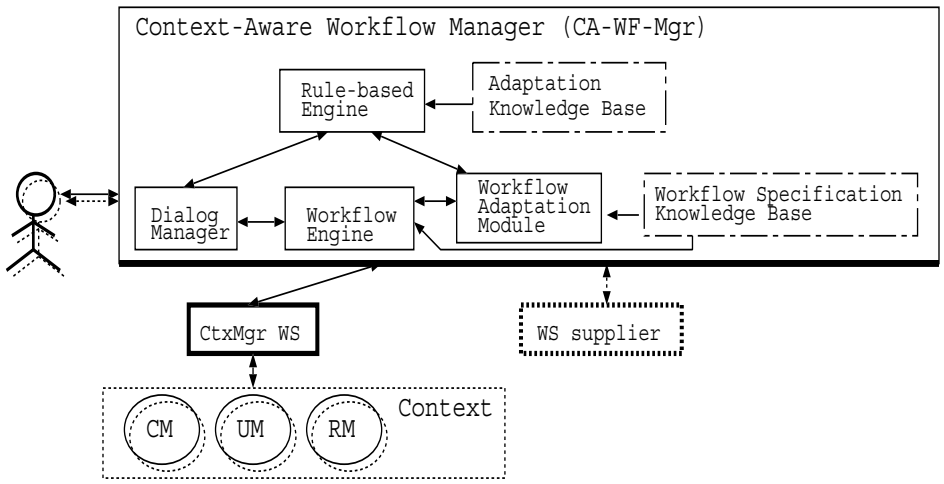

Figure 1 Architecture of the CAWE framework. Web Service interfaces are depicted as thick lines.

3 The CAWE framework

The CAWE framework supports the development of composite applications which tailor the business and the interaction logic, as well as the User Interface, to the user and to her/his surrounding context. Thanks to the explicit representation of the adaptation logic, the framework enables the adoption of flexible techniques to steer the system behaviour. As show in Figure 1, the CAWE architecture includes two core components:

- The *Context Manager service (CtxMgr WS)* handles the context information during the execution of the application; see Section 3.1.

- The *Context-Aware Workflow Manager (CA-WF-Mgr)* enacts a context-sensitive workflow which defines the business logic of the application. For this purpose, the CA-WF-Mgr employs two software components: the Workflow Adaptation Module shapes the workflow depending on the context; the workflow engine enacts the resulting workflow. See Section 3.2.

  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 informs the CtxMgr WS about the user's identity and the device (s)he is using. Then, it handles the tasks to be completed as dialogues with her/him; moreover, it adapts the User Interface to a context including both the user's device and her/his layout preferences. See Section 3.3.

The CA-WF-Mgr and the Dialog Manager invoke a rule-based engine to evaluate the adaptation policies, which are represented as declarative rules: their precondition is a boolean condition on context variables and their action specifies the adaptation decision to be applied. Given the result of the rule evaluation, the two modules apply the selected behavior.

```
Role Model (RM):
  role: String (role name)
  currentUM: String (reference to UM of current role filler)
  UMList: list of String (references of UMs of role fillers)
  FeatureList:  sequence of FeatureType elements

User Model (UM):
  ID: String (UM identifier)
  CMRef: String (reference to the CM associated to the UM)
  FeatureList:  sequence of FeatureType elements

Context Model (CM):
  ID: String (CM identifier)
  UMRef: String (reference to the UM of the user)
  FeatureList:  sequence of FeatureType elements

FeatureType:
  featureName: String
  featureVal: String
```

Figure 2 Structure of the RMs, UMs and CMs. These models are represented as XML documents; however, the figure presents them in a simplified format for readability purposes.

The adaptation policies are stored in the Adaptation Knowledge Base and they are grouped in packages, depending on the kind of decision they are devoted to. Specifically, the `business-logic` package stores the policies steering the selection of the courses of action to be enacted; the `layout` package supports the context-dependent distribution of content in the UI pages; the `style-selection` package specifies the selection of the (XSL) stylesheets to be applied. Henceforth, the policies concerning the generation of the User Interface, and the management of the interaction with the user, are denoted as *UI adaptation policies*.

3.1 Context information

In order to support the adaptation of the application to the actors involved in the service, and to their surrounding contexts, the CtxMgr WS handles a Role Model for each role defined in the service, as well as a User Model and a Context Model for each involved actor. Moreover, the CtxMgr WS handles, for each interaction session, an *i-user* variable, which refers to the User Model of the actor interacting with the application. Figure 2 shows the structure of the models:

- The Role Model (RM) associated to a role r stores the references to the User Models of the actors who can fill r (`UMList`). Moreover, it stores a reference to the UM of the current role filler (`currentUM`) and some domain-dependent, default information about the role (`FeatureList`).

```
Rule 1:
  package: business-logic
  precondition: abstract-activity-name==BookBloodTest and
                patient.UM.movable
  action: implementation="WF10"
Rule 2:
  package: business-logic
  precondition: abstract-activity-name==BookBloodTest and
                !patient.UM.movable and nurse-available
  action: implementation="WF11"
Rule 3:
  package: business-logic
  precondition: abstract-activity-name==BookBloodTest and
                !patient.UM.movable and !nurse-available
  action: implementation="WF12"
```

Figure 3 Sample business logic adaptation rules (e-Health application). For readability purposes, the rules are described in a simplified form.

- The User Model (UM) stores information about an individual actor; e.g., expertise, preferences, and physical capabilities (`FeatureList`). Moreover, it contains a reference to the associated Context Model (`CMRef`).
- The Context Model (CM) stores information about the context surrounding the actor; e.g., the device used to interact with the application.

As the features of the RMs, UMs and CMs depend on the application domain, the developer has to define them at set-up time. However, the CAWE framework provides some templates, which can be taken as a starting basis. For instance, the UM and the CM templates include, respectively, the user's `font-preference` (for the UI layout) and her/his `device`.

3.2 Adaptation of the business logic

3.2.1 Business logic representation

The business logic of an application is represented as a context-sensitive workflow organised in an abstraction hierarchy which specifies the system behaviour 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 some *abstract activities*. These describe a generic type of behaviour, to be decided at runtime. An abstract activity is thus an activity schema which does not directly define the operations to be performed when it is enacted.
- Each abstract activity is associated with a set of *implementations* describing different courses of action to be selected for the completion of the activity,

depending on the context. Each implementation is a workflow which can specify rather different behaviours; e.g., starting a task to be performed by a human actor, invoking a Web Service, starting a subprocess, or carrying out some internal computation. Notice that an implementation may include other abstract activities; therefore, the context-sensitive workflow can be organised as a multi-level hierarchy.

- The *business logic adaptation policies* (in package `business-logic`) steer the context-dependent selection of the implementations to be enacted during the execution of the abstract activities. These policies are described as condition-action rules:
  - The precondition of a rule is a boolean condition on context variables.
  - The action is either the name of the implementation to be enacted, or the reference to a group of rules to be evaluated for refining the adaptation decision (rule chaining).

For instance, Figure 3 shows three sample policies defined in our e-Health application; see Section 5. The preconditions of the rules specify the name of the reference abstract activity (`BookBloodTest`) and the relevant context variables; e.g., the patient's mobility state (`patient.UM.movable`) and the availability of a nurse (`nurse-available`). The action part specifies the identifiers of the implementations to be performed in each case; e.g., `WF10`. For readability purposes, in the figure, we have reported the rules in a simplified, Java-like form; for details, see Section 4.

3.2.2 Context-aware workflow execution

At runtime, the business logic of the application is composed by recursively selecting the implementations of the abstract activities to be enacted. This selection is steered by the business logic adaptation policies, which support a reactive planning of the system behaviour.

The Context-Aware Workflow Manager wraps a workflow engine which executes the context-sensitive workflow as if it were a standard one.[a] However, when the engine encounters an abstract activity, it works as follows:

1. First, it invokes the Workflow Adaptation Module on the abstract activity.
2. When the module returns the implementation to be enacted, the engine performs it as a subprocess of the main process instance.
3. At subprocess completion, the engine resumes the execution of the higher-level workflow.

For the selection of the implementation, the Workflow Adaptation Module employs the previously described rule-based engine, which works on the business logic adaptation rules and fires the one best suiting the context. Given the name of the selected implementation, the Workflow Adaptation Module retrieves the corresponding workflow and binds the input and output parameters to their current values. Then, it returns the result to the caller.

[a] As explained in Section 4, the abstract activities are syntactically similar to the other workflow activities. Therefore, they can be performed by a standard Web Service composition engine.

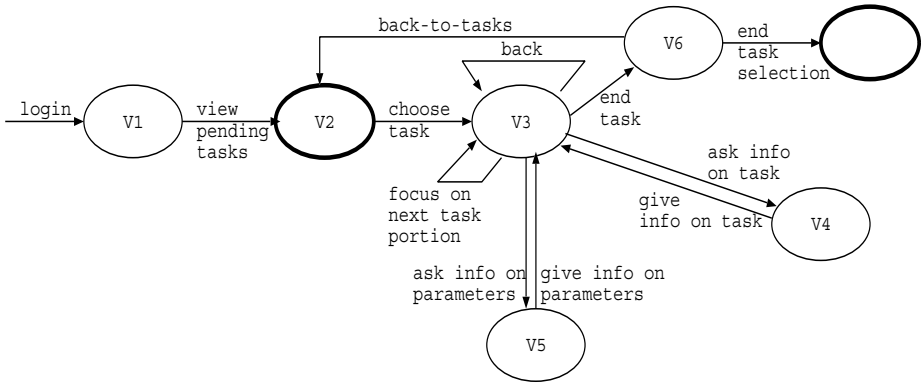

Figure 4 Finite State Automaton describing the interaction logic of the Dialog Manager.

3.3 Adaptation of the interaction with the user

The Dialog Manager handles a task as a communicative goal to be achieved by carrying out a dialogue with the user. Each dialogue step is aimed at achieving a task portion and is managed by generating a UI page. In order to handle flexible, but lightweight interactions, a dialogue management technique based on Finite State Automata is applied. Figure 4 shows the automaton describing the interaction logic of the Dialog Manager: the states correspond to the page types and the state transitions are performed as a consequence of the user actions.

1. After the user has logged in the application (`login` arc), (s)he can ask for the list of pending tasks assigned to her/him (`view pending tasks`). In order to retrieve information about the tasks to be performed, and their parameters, the Dialog Manager invokes the workflow engine via API.

2. Then, from state V2 the user selects the pending task (s)he wants to complete (`choose task` arc). This action starts the task execution, which may involve several request/response turns, organised as follows:

   (a) The Dialog Manager sends the user's browser a personalised UI page representing an interaction turn (V3). 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 the help links to get more specific information about the task and its parameters.

   (b) In turn, the user may perform different actions:

   - Each help link, and each information link associated to the parameters, activates a nested dialogue. For instance, the `ask info on task` transition leads to state V4, which represents the UI page presenting specific information about a task.
   - The `focus on next task portion` and the `back` transitions move to the next, or to the previous dialogue step, respectively.
   - The `end task` transition closes the dialogue (state V6).

```
Task-page-template
stylesheet: xsl stylesheet id
  user-role: String
  task-name: String
  task-ID: integer
  task-help-link: url
  task-portion: integer
  number-of-task-portions: number
  back-link, continue-link, cancel-link, finish-link: url
  input section
    parameters: sequence of I-param elements
  output section
    parameters: sequence of O-param elements
    other-info: sequence of O-param elements
  I-param
    name: String
    value: boolean or number or String
    help-link: url
  O-param
    name: boolean or number or String
    help-link: url
```

Figure 5 Template specifying the structure of the UI pages for the visualisation of the task portions. For readability purposes, the template is presented in simplified format.

3. When the Dialog Manager reaches state V6, it enables the user to inspect another pending task (`back-to-tasks`), or to end the task management activity (`end task selection`). Meanwhile, the Dialog Manager notifies the workflow engine about the task completion, and feeds it with the acquired data.

The generation of the personalised pages is based on the evaluation of the UI adaptation policies. For example, Step 2a above is handled as follows:

1. The Dialog Manager selects the page layout by evaluating (via rule-based engine) the rules belonging to the `style-selection` package.
2. Given the input and output parameters of the task, the Dialog Manager groups the information items to be displayed in one or more subsets, in order to fit the size of the screen of the user's device, and to comply with the her/his features (e.g., known vision impairments) and preferences (e.g., font size). This is done by evaluating the rules of the `layout` package.
3. Then, the module fills an XML page template with the content to be displayed. For instance, Figure 5 shows the template of the pages devoted to the management of the task portions. The input/output sections of the template store the parameters of the task. The (optional) help-links refer to additional information about the task and its parameters.
4. Finally, the Dialog Manager generates the page code by applying the stylesheet to the filled template.

```
Rule 1:
  package: layout
  precondition: i-user.UM.font-preference==x and
                i-user.CM.device=="desktop"
  action: maxParameters = 20 * |18/fontSize|
```

Figure 6 Sample layout rule (e-Health application). For readability purposes, the rule is described in a simplified form.

The UI adaptation rules are domain-dependent, but the CAWE framework offers some templates, to be extended and modified. For instance, the rule in Figure 6 sets the maximum number of items to be displayed in a page, given the user's device and font preferences.

4 Technical details

The CAWE prototype is developed on top of jBPM, a business process management system implemented in Java; see Koenig (2004). jBPM is based on the Graph Oriented Programming model, which complements Object-Oriented programming with a runtime model for executing long lasting workflows of activities, represented as graphs. The workflow specification language supported by jBPM is jPDL (jBPMProcess Definition Language).

The abstract activities are represented as jPDL `process-state` nodes. When a workflow instance execution gets in an abstract activity, a jPDL `Action` (associated to an "enter-node" event) is used to invoke the Workflow Adaptation Module and retrieve the subprocess to be performed.

The Dialog Manager module is implemented as a server-side program based on the Model View Controller pattern; see Seshadri (1999). A Java Servlet (Sun Microsystems (2008)) plays the role of the Controller, which intercepts the user's HTTP requests and selects the View to be displayed on the user's browser, given the user's request and of the workflow state. The interaction logic implemented by the Controller is described by the Finite State Automaton of the Dialog Manager. Each state of the automaton is associated to a View, representing the corresponding UI page, except for the final state. The Dialog Manager uses the `AbstractWizardFormController`, offered by the Struts library to manage the dialogues with the users; see Apache Software Foundation (2008).

The CAWE prototype exploits the JESS rule-based engine (Sandia National Laboratories (2009)) to handle the adaptation policies. JESS is a lightweight, rule-based engine implemented in Java, which supports the selection and execution of condition-action rules. In order to guarantee high performance in the selection of the rules to be fired, the engine applies an enhanced version of the Rete pattern-matching algorithm, initially proposed in Forgy (1982). In the CAWE prototype, JESS supports a very fast evaluation of the adaptation policies and introduces a marginal overhead on the performance of the workflow engine.

```
Role Model (RM):
  role: doctor
  currentUM: um05
  UMList: {um05}
  FeatureList:  {}

User Model (UM):
  ID: um05
  CMRef: cm11
  FeatureList:
    ID: 744
    phone: +3901188493523
    font-preference: 14
    ...

Context Model (CM):
  ID: cm11
  UMRef: um05
  FeatureList:
    device: PDA
    ...
```

Figure 7 Portions of the *doctor* RM, and of the UM and CM of an individual doctor.

5 The e-Health application

We exploited the CAWE framework to develop an e-Health prototype application supporting the management of a clinical guideline. Clinical guidelines are an excellent testbed for multi-user adaptivity and contex awareness. In fact, they are long-lived processes, which benefit from a workflow-based implementation. Moreover, they involve actors who play different roles (e.g., doctors and administration staff) and operate in dynamic environments, using desktop and hand-held devices.

The clinical guideline we selected specifies the activities to be performed in order to monitor the health state of patients affected by heart diseases. Such patients, who regularly stay in their homes, have to undergo periodical tests to check the fluidity of their blood and revise the therapy accordingly.

5.1 Context information

Within the e-Health scenario, we identified five roles: *patient*, *relative*, *nurse*, *doctor*, and *administration staff*. The top of Figure 7 shows the RM of the *doctor* role. The rest of the figure sketches the UM and the CM of an individual user playing that role. Specifically, the `FeatureList` field of the CM includes the device used to interact with the application (`device`).

Similar models describe the other roles and the involved actors. In particular, the user model of a patient includes the `movable` feature, which describes her/his mobility state. This kind of information influences the guideline execution because

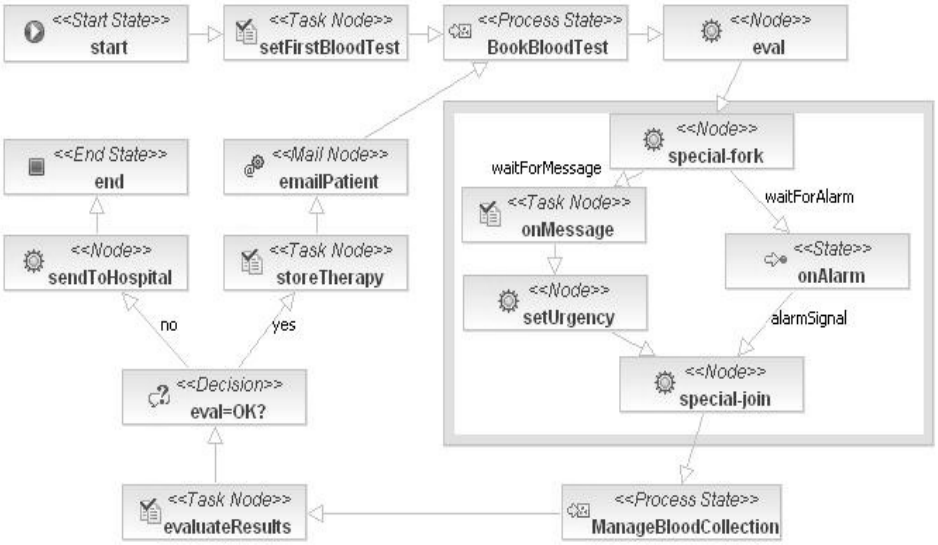

Figure 8 Abstract workflow of the e-Health scenario.

non-movable patients are entitled to receive home services and to use special transportation means.

The context information is retrieved from the human users involved in the management of the clinical guideline and from automated sources, such as sensors connected to the internet and the patient's clinical record. The information stored in the latter can be retrieved by invoking the Clinical Record Manager Web Service of the hospital.

5.1.1 Business logic

Figure 8 shows a portion of the context-sensitive workflow of the application. The types of nodes occurring in the workflow are auto-explicative; however, it should be noticed that, as described in Section 4, abstract activities are represented as process state nodes. The workflow can be roughly described as follows:

1. A doctor sets the date of the first blood test to be performed (`setFirstBloodTest`).

2. The application reserves a blood test with a lab at the specified date and evaluates the time interval before the test (`BookBloodTest`, `eval`).

3. If the patient's health state is good, (s)he waits until the date of the test (`onAlarm`). If any warning symptom occurs before that date (`onMessage`), the service sets the urgency of the case (`setUrgency`).

4. At the specified date, or after a warning symptom, a blood sample is taken from the patient and analysed (`ManageBloodCollection`). Then, a doctor evaluates the results (`evaluateResults`). If they are good, the doctor sets the therapy (`storeTheraphy`); then, the application notifies the patient

```
Rule 1:
  package: style-selection
  precondition: i-user.CM.device=desktop
  action: stylesheet file: large-UI.xsl
Rule 2:
  package: style-selection
  precondition: i-user.CM.device=PDA
  action: stylesheet file: medium-UI.xsl
```

Figure 9 Sample style selection rules (e-Health application). The rules are described in a simplified form.

(`emailPatient`) and the flow restarts from item 2. Otherwise, the patient is advised to go to the hospital (`sendToHospital`).

As a sample abstract activity, we analyse `BookBloodTest`. This activity has three context-dependent implementations, not shown for brevity:

- `WF10` handles the booking of the appointment for the blood test at the lab. As this implementation requires that the patient autonomously goes to the lab, it is suitable for patients which can be transported by car.
- `WF11` schedules the collection of the patient's blood at home (by means of a nurse) and it is suitable to handle non movable patients.
- `WF12` is an alternative solution devoted to non movable patients, when it is not possible to have a nurse at the patient's home. This implementation involves the booking of a special transportation means to carry the patient to the lab.

Figure 3 (page 7) shows the business logic adaptation rules associated to `BookBloodTest`. Their preconditions reflect the requirements described above; e.g., the precondition of `Rule 1` includes the `patient.UM.movable` condition.

5.1.2 Interaction logic

The interaction logic is defined by the Finite State Automaton of the Dialog Manager, and by the UI adaptation policies. We have already described the rules of the `layout` package; see Figure 6. The rules shown in Figure 9 belong to the `style-selection` package. In each rule, the precondition refers to the user's device (`i-user.CM.device`), as this is the main information needed for the stylesheet selection. The action part specifies the name of the XSL document to be applied.

Figure 10 shows a UI page generated during the management of a task assigned to a doctor: `evaluateResults()`. The page is targeted to a desktop device:

- The top bar displays the name of the application (eHealth), the username (house) and the logout button.
- The lower part of the page shows the input (Form area) and output (Information area) parameters of the task. Each parameter name is a link to its more specific information.

Figure 10 First dialogue turn in the management of task `evaluateResults`, tailored to a desktop device.

- The middle bar is organised 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 visualisation of the task description; the position of the interaction turn in the dialogue (Page 1 of 4); the Continue link (>>) taking to the next turn and a Cancel link to reset the user's inputs.

Figure 11 shows two UI pages, targeted to PDA. In order to cope with the smaller screen size, the dialogue is performed in more steps than in the desktop case.

5.2 Discussion

In the development of the e-Health application, the context awareness support offered by the framework was satisfactory for the following reasons:

- CAWE supports the dynamic definition of the business logic: the courses of action to be enacted are dynamically selected and composed, by evaluating possibly complex adaptation policies. The courses of action include performing a subprocess, invoking a service supplier, carrying out some computation or starting a task; thus, the application can exhibit very different behaviours. This flexibility is based on the specification of declarative adaptation policies which enable the developer to define fine-grained conditions for the selection of the activities to be performed. At the same time, such selection does

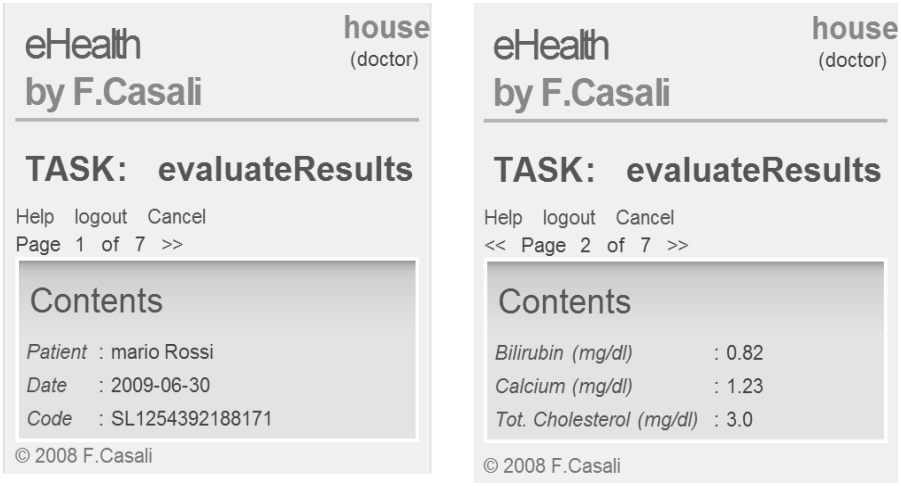

Figure 11 First two dialogue turns in the management of task `evaluateResults`, tailored to a PDA.

not significantly overload the system, as it is based on the execution of a lightweight and efficient rule-based engine.

- The framework also supports the management of a flexible interaction logic. In particular, it replaces the standard stylesheet selection with the evaluation of declarative adaptation rules. Moreover, it adapts and distributes the content to be visualised on the User Interface by taking various factors into account; e.g., the screen size of the user's device, her/his visualisation preferences and the applied layout. Furthermore, the framework supports the completion of tasks by presenting extra-helpful information about them.

The development of the application has also proved the applicability of CAWE in a real use case, as the application developer has to devote a limited effort to configure the knowledge bases of the system. Specifically:

- The specification of the features characterising the Role, User and Context Models is a standard task to be performed in a user-adaptive system and requires limited technical skills.

- The representation of the context-sensitive workflow, based on an hierarchy of processes, factors out the common courses of action. Therefore, it reduces the number of workflow paths, with respect to a flat representation. The advantages are obvious: first, the workflow is smaller and modular, and thus more readable; second, it is easier to modify, because the specification of the courses of action is not replicated. The hierarchical representation also facilitates the introduction of new courses of action, which can be added locally to the abstract activities. Furthermore, it supports the top-down development of the business logic.

Some more flexibility in the management of the interaction with the user might be achieved by enabling the developer to modify the structure of the Finite State

Automaton of the Dialog Manager. However, this kind of revision requires deep expertise in dialogue management systems; therefore, we decided to forbid it.

In order to test the generality of our approach, we instantiated the CAWE framework on another application domain. We selected a business travel scenario in a University and we identified the adaptation requirements emerging in such case. This analysis confirmed the need to apply different adaptation policies depending on the user's role (e.g., Ph.D students and faculty members are subject to different restrictions as far as conference participation is concerned), the device used to interact with the system (usually, a different one when the user is in her/his office, or is travelling), and individual user preferences (e.g., concerning the transportation means, or the User Interface layout). The specification of the context-sensitive workflow and of the adaptation policies for the business travel application confirmed the usefulness and the suitability of the adaptive features offered by the framework.

6 Related work

In Service Oriented Computing, some contributions extend standard Web Service composition languages with context-awareness features (e.g., C-BPEL; see Ghedira and Mezni (2006)), in order to comply with Quality of Service (QoS) requirements. For instance, Benlismane *et al.* (2005) monitor the service to prevent QoS violations; Ardagna and Pernici (2007) handle the personalised Web Service selection as an optimisation problem; Baresi *et al.* (2007) support the runtime binding and replacement of service providers for failure recovery purposes. These approaches are affected by the limitations of standard Web Service composition languages, such as WS-BPEL (OASIS (2005)) and its context-aware extensions (e.g., Context4BPEL, see Wieland *et al.* (2007)), which embed the adaptation logic in the workflow specification. On a different perspective, Charfi and Mezini (2007) propose using Aspect-Oriented Programming to improve the flexibility of the compositions; however, they focus more on modularisation than on adaptivity.

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

Our work also differs from the few workflow-based adaptive systems developed in the Adaptive Hypermedia research. For instance, CAWE personalises the workflow to the users and their context. Instead, in Holden *et al.* (2005) the system enacts the same workflow for all the users and contexts.

The Semantic Web research applies planning technology 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 McIlraith *et al.* (2001), Balke and Wagner (2003), Balke and Wagner (2004), Keidl and Kemper (2004). In particular, Qiu *et al.* (2007) propose a hybrid approach that integrates global planning and local optimisation, supported by an ontology-based context representation. Also Vukovic *et al.* (2007) use planning for the run-time service selection, in order to manage failure recovery. Furthermore, Horvitz and Subramani (2007) propose opportunistic planning to support the user in the efficient achievement of goals diverging from her/his main activities. However, planning technology

is not suitable to handle long-lasting services and processes because it does not offer persistence management. Therefore, up to now it has only been used to handle short-lived composition plans. As a matter of fact, relying on a standard workflow engine for the management of the business logic of an application has scalability and robustness advantages, which are not guaranteed by other technologies. 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 Mandell and McIlraith (2003) and Laures and Jank (2005).

An interesting approach to the design of Web-based applications is provided by the Web Engineering community; for example, Batini *et al.* (2007) propose a unified methodology for the design of multi-channel adaptive Web-based information systems. We believe that this work is complementary to our own.

Concerning the management of the interaction with the user, context-aware workflow systems only provide the adaptation of the User Interface to the user's device, in terms of stylesheet selection; e.g., see Keidl and Kemper (2004), and the extension to WebML to model multi-channel, context-aware Web applications proposed in Ceri *et al.* (2003). 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 User Interface and the interaction logic on an individual basis.

In the research about dialogue-based systems, some researchers employed scripts describing domain-level activities and linguistic behaviour to model articulated task-oriented dialogues; e.g., see Chu-Carroll and Carberry (1998). Moreover, planning technology was applied to manage short-lived interactions with the user; e.g., see Rich *et al.* (2002). Furthermore, plans and scripts were used to generate messages and explanations to the user; e.g., see Moore and Paris (1993) and Milosavljevic (1999). 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 behaviour.

7 Conclusions

This article has presented the Context Aware Workflow Execution framework for the development of context-aware composite Web applications. The framework enriches Service Oriented Architecture with:

- Adaptation techniques enabling the execution of context-sensitive workflows.
- Dialogue management capabilities supporting flexible user interactions.
- Context-dependent User Interface generation techniques aimed at presenting personalised 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. These adaptation capabilities are based on the declarative representation of the context variables to be taken into account, of the business, interaction and presentation logics, and of the policies steering the context-dependent selection of the system behaviour.

The analysis of two real-world application domains proved the usefulness and the suitability of the adaptive features offered by the framework. Moreover, the development of a prototype Web Application in the first domain confirmed the applicability of the framework to real-world use cases.

References

G.D. Abowd and E.D. 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.

Apache Software Foundation. Struts. http://struts.apache.org/, 2008.

D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Transactions on Software Engineering*, 33(6):369–384, 2007.

M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. Journal ad hoc and ubiquitous computing*, 2(4):263–277, 2007.

W.T. Balke and M. Wagner. Towards personalized selection of Web Services. In *Proc. of 12th Int. World Wide Web Conference (WWW'2003)*, Budapest, 2003.

W.T. 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.

L. Baresi, E. Di Nitto, C. Ghezzi, and S. Guinea. A framework for the deployment of adaptable web service compositions. *Service Oriented Computing and Applications*, 1(1):75–91, 2007.

C. Batini, D. Bolchini, S. Ceri, M. Matera, A. Maurino, and P. Paolini. The UMMAIS methodology for multi-channel adaptive web information systems. *World Wide Web*, 10(4):349–385, 2007.

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.

P. Brusilovsky, A. Kobsa, and W. Nejdl. *The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Vol. 4321*. Springer-Verlag, 2007.

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.

A. Charfi and M. Mezini. AO4BPEL: An aspect-oriented extension to BPEL. *World Wide Web*, 10(3):309–344, 2007.

J. Chu-Carroll and S. Carberry. Collaborative response generation in planning dialogues. *Computational Linguistics*, 24(3):355–400, 1998.

A.K. Dey and D. Abowd. Towards a better understanding of context and context-awareness. In *Proc. CHI2000 Workshop on the What, Who, Where, When and How of Context-Awareness*, The Hague, Netherlands, 2000.

P. Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004.

C.L. Forgy. Rete: A fast algorithm for the many pattern/ many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

C. Ghedira and H. Mezni. Through personalized web service composition specification: from bpel to c-bpel. *Electronic Notes in Theoretical Computer Science*, (146):117–132, 2006.

T. Gross. Cooperative ambient intelligence: towards autonomous and adaptive cooperative ubiquitous environments. *Int. Journal of Autonomous and Adaptive Communications Systems*, 1(2):270–278, 2008.

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

E. Horvitz and M. Subramani. Mobile opportunistic planning: methods and models. In *LNAI 4511: Proc. 11th Int. Conf. on User Modeling*, pages 228–237, Corfu, Greece, 2007.

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.

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

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

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)*, pages 227–241. Springer-Verlag, Sanibel Island, Florida, 2003.

M. Maybury and P. Brusilovsky, editors. *The adaptive Web*, volume 45. Communications of the ACM, 2002.

S. McIlraith, T.C. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

M. Milosavljevic. *The automatic generation of comparison in descriptions of entities*. PhD thesis, Macquarie University, Sydney, 1999.

J.D. Moore and C.L. Paris. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–694, 1993.

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

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

L. Qiu, L. Chang, F. Lin, and Z. Shi. Context optimization of ai planning for semantic web services composition. *Service Oriented Computing and Applications*, 1(2):117–128, 2007.

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.

Sandia National Laboratories. JESS, the Rule Engine for the Java-TM Platform. http://www.jessrules.com/, 2009.

G. Seshadri. Understanding JavaServer Pages Model 2 architecture - exploring the MVC design pattern. In *JavaWorld*, http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html, 1999.

Inc. Sun Microsystems. Java Servlet Technology. http://java.sun.com/products/servlet/, 2008.

M. Vukovic, E. Kotsovinos, and P. Robinson. An architecture for rapid, on-demand services composition. *Service Oriented Computing and Applications*, 1(4):197–212, 2007.

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.