Deployment of adaptive workflows in intelligent environments

Christos Goumopoulos, Ioannis Calemis
Research Academic Computer Technology Institute
N. Kazantzaki, 26500 Rio Patras, Greece
{gounop, calemis}@cti.gr

Achilles Kameas
Hellenic Open University & RACTI
16, Sahtouri Str, Patras, Greece
kameas@eap.gr

Abstract—Workflows have been used to model repeatable tasks or operations in a number of different industries including manufacturing and software. In this paper we examine the use of workflows to model the interaction of services that can be found in intelligent environments to support user tasks and goals. The deployment of such workflows needs to take care special design considerations, including context awareness, adaptation management, device heterogeneity, and user empowerment. In this paper, we present a framework for the deployment of adaptive workflows. The deployment infrastructure supports BPEL-like, design-time compositions that are complemented by mechanisms for the selection and binding of services at runtime. Workflow behaviour can also adjust dynamically in response to detected changes and unforeseen events by a suit of agents whose initial relationships are specified in the workflows.

Keywords- SOA; workflows; BPEL; dynamic compositions; ubiquitous computing; adaptation

I. INTRODUCTION

Intelligent environments (IE), like smart homes, offices and public spaces, are featured with a large number of devices and services that help users in performing efficiently various kinds of tasks. Combining existing services in pervasive computing environments to create new composite services is in line with the Service-Oriented Architecture (SOA) paradigm [1] and involves special design considerations, including context awareness, adaptation management, device heterogeneity, and user empowerment [2].

In many respects, a composite service can be modeled as a workflow [3]. The definition of a composite service includes a set of atomic services together with the control and data flow among the services. Similarly, a workflow is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules [4]. Workflows have been used to model repeatable tasks or operations in a number of different industries including manufacturing and software. In recent years, workflows have increasingly used distributed resources and Web services through resource models such as grid and cloud computing. In this paper, we argue that workflows can be used to model how various services should interact with each other as well as with the user in IEs depending on available resources, environment characteristics, user tasks and profile.

Web services are a key implementation technology of the SOA paradigm which is characterized by dynamism and flexibility. However, the main standards proposed to implement the SOA paradigm (i.e., WSDL, UDDI, SOAP) emphasize interoperability rather than the capability to accommodate seamless changes at runtime. Frameworks based on ontologies, such as METEOR-S [5], also lack flexible mechanisms for the distribution of information about services as they require the adoption of shared ontologies that impose the distribution policy. Regarding composition, BPEL (Business Process Execution Language [6]) is the de-facto standard. It takes a workflow-oriented approach to the coordination of cooperating services and provides a good solution for the design–time composition of heterogeneous components wrapped as WSDL services. However, runtime identification of partner services is not addressed and thus the degree of dynamism and flexibility is limited.

In the context of the EU funded R&D project ATRACO we are developing a conceptual framework and a system architecture that supports the realization of adaptive and trusted ambient intelligent systems [7]. Our approach is based on a number of well established engineering principles, such as the distribution of control and the separation of service interfaces from the service implementation, adopting a SOA model combined with intelligent agents and ontologies. Agents support adaptive task realization and enhanced human machine interaction based on a dynamically composed ontology of the properties, services and state of the IE resources.

In this paper, we focus on the service composition mechanism used in ATRACO and how the plan of activities that serve a user goal is specified as a workflow of resource and system services using a streamlined version of BPEL. Our approach is that since a workflow describes the relationship between services and if an agent is represented by such a service, then the relationship between the agents would be possible to specify. Following such an agent-based SOA approach, means that a workflow could be used to establish the initial relationships of the ATRACO multiagent system. Once the basic system has been deployed, the agents could be working proactively so they can adapt to unforeseen circumstances and automatically handle the extension of the workflow description. This leads finally to a decentralized workflow execution model that on the one hand does not suffer from the inflexibility of static workflows, and on the other hand can avoid the computationally expensive cost of frequent replanning of composite services, because of the agent-based proactive behaviour.
The paper is organized as follows. Section 2 gives a basic background on ATRACO concepts, architecture and the role of the main components. Section 3 contains the main contribution of this paper, a service composition framework for deploying adaptive workflows in IEs. We first give an example scenario, and then discuss the mechanisms developed, the BPEL extensions made in order to express ATRACO specific features in the workflows, and deployment issues. Related work is presented in Section 4 and our conclusion and future work in Section 5.

II. BACKGROUND

In ATRACO, we propose a combination of the SOA model with agents and ontologies (Fig. 1). We adopt SOA both at the resource level to integrate resources, such as devices, sensors and context in applications and at the system level to combine ATRACO services that provide adaptation and trust features into applications. We have defined the concept of an Activity Sphere (AS), to be both the model and the realization of the set of information, knowledge, services and other resources required to achieve an individual goal within an IE. ATRACO approach adopts a unique standpoint in modeling and realizing ASs. We assume that various IEs are already available; each of them hosting a dynamically changing set of heterogeneous and closed smart objects and components. They, nevertheless, contain heterogeneous descriptions of their capabilities and services that can only be accessed from but not modified by other components. Thus, these objects can collaborate in the realization of ubiquitous computing applications within the hosting IE, but the structure of these applications may dynamically change and their efficient operation depends on the orchestration of heterogeneous services. In order to achieve task-based collaboration amongst them, one has to deal with this heterogeneity, while at the same time achieving independence between a task description and its respective realization within a specific IE.

The ATRACO architecture consists of ontologies, active entities, passive entities, and the user who as the occupant of the IE is at the centre of each AS. Active entities are agents and managers. The role of the ATRACO agents is to provide task planning (Planning Agent or PA), adaptive task realization (Fuzzy systems based Task Agent or FTA) and adaptive human-machine interaction (Interaction Agent or IA). The PA encapsulates a search engine that exploits hierarchical planning and partial-order causal-link planning to select atomic services that form a composite service (workflow) [8]. One or more FTAs oversee the realization of given tasks within a given IE. These agents are able to learn the user behavior and model it by monitoring the user actions. The agents then create fuzzy based linguistic models which could be evolved and adapted online in a life learning mode [9]. The IA provides a multimodal front end to the user. Depending on a local ontology it optimizes task-related dialogue for the specific situation and user [10]. The IA may be triggered both by the FTA and the PA to retrieve further context information needed to realize and plan tasks by interacting with the user. On the other hand, ontologies complement agents regarding adaptation by tackling the semantic heterogeneity that arises in IEs by using ontology alignment mechanisms to generate the so-called, Sphere Ontology (SO). There are two main kinds of ontologies: local ontologies, which are provided by both active and passive entities and encode their state, properties, capabilities, and services and the SO, which serves as the core of an AS by representing the combined knowledge of all entities [11].

The Sphere Manager (SM) and Ontology Manager (OM) components are responsible for the formation, adaptation and evolution of the user applications (modeled in ATRACO as ASs) and will be further examined in this paper. In the current version of the system there is also a Privacy Manager (PM) that provides a set of privacy enhancing techniques in order to support privacy in an adaptive and individualized way. Finally, devices in the IE that may come from heterogeneous networks (e.g., LonWorks, ZigBee, Z-Wave, etc.) and services (e.g., Network Time, VoIP, Real Time Streaming, etc.) are accessed transparently through a service representation layer exporting them to the ATRACO clients as UPnP services. This layer is implemented in the Network Adaptation (NA) component [12].

III. SERVICE COMPOSITION FRAMEWORK

A. Example Scenario

In order to test our service composition framework and to illustrate how workflows can be used to fit user interaction with an IE, we give a simple scenario. This example corresponds to an AS that supports the realization of goal named “Feel comfortable upon arrival at home”.

Martha arrives at the door of her smart apartment. The system recognizes her, through an RFID card, and opens the door. On entering the space the system greets Martha by saying “Welcome home” and then when she has entered the living space the lights and A/C are switched on and brightness and temperature are automatically adjusted according to her profile, season, and time of day, to make her feel comfortable. Martha then sits at the sofa to relax and after a while, the system asks “Would you also like some music?” Martha responds positively and the music plays (according to predetermined preferences). Following this, the system asks “Would you like to view yesterday’s party photos?” Martha responds positively and a rolling slide show appears in a picture frame in front of her. After a while, Martha gets up, walks towards the window and opens it. Fresh air pours into
the actual human and physical resources is done at runtime, in keeping with service orientation. This dynamic binding is therefore dependent on the context in which the binding occurs.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>1</th>
<th>Task</th>
<th>PA responsibility</th>
<th>1</th>
<th>Service</th>
<th>1</th>
<th>Resource</th>
<th>1</th>
</tr>
</thead>
</table>

Figure 3. Conceptual model for dynamic service binding

After service binding the SM starts any interaction task in conjunction with the IA and also any FTA task and executes the workflow preserving the precedence constraints or the conditions that are specified in the workflow. At runtime a Workflow object aggregates a number of Task objects where each object represents a task in the workflow. The services that this task requires for running are divided into input and output services and are connected with the appropriate resources. The resources that are bound to the Task object can be either devices that the Task directly controls (i.e., input sensors and actuation devices) or agents, such as the IA or the FTA. In either case the Task object is informed on the status of the resource and operates according to the pattern specified by its type. The sequence diagram in Fig. 4 shows the basic interaction of the software components during the instantiation of the “Feel Comfortable” AS, which employs the dynamic service binding process mentioned earlier. In the diagram, this process is implemented by the methods used inside the two loops.

In addition, the SM handles exception events that affect the configuration of the AS. For example, exceptions during the execution of the workflow, such as disconnection or failure of devices trigger an adaptation of the workflow by rebinding services to alternative devices. Context changes during the execution of the workflow may invalidate preconditions that were valid during the workflow instantiation. For example, if the user changes location and a follow-me property has been defined for a display service, then the execution state needs to be updated and a new display service instance to be scheduled. In order to achieve workflow adaptation, replanning capabilities may be required by the PA. Replanning comes into play when the dynamic binding fails during workflow execution or update. When replanning is requested a new planning problem is defined with the services that are actually available, and the PA solves the problem and delivers a new workflow.

The planning problem can be stated as “discover an execution path of services (tasks) given some state of the world to achieve a goal”. In ATRACO, we use a library of abstract plans which model specific user goals. An abstract plan contains a sequence of abstract services which are actually ontological descriptions of service operations that cannot be directly invoked, but will be resolved by the SM during runtime. Having an abstract service workflow description, which is given in a BPEL-like language, the Dynamic Service Binding module of the SM applies a semantic-based discovery mechanism and uses information about available services and context, acquired by the SO through the OM, to discover suitable services or devices in registries able to perform each abstract service. The output of this process is an executable service workflow. In the execution management and control phase the SM executes and continuously monitors the deployed services and the termination condition of the workflow. Fig. 3 gives a conceptual view of the dynamic service binding process. A workflow is mapped into a number of tasks and a workflow task is mapped into one or more abstract services. In addition, each service would also require certain physical resources for its implementation. Mapping of the task to the services can be specified at design time by the PA as per users’ functional requirements. However, mapping of the service to the actual human and physical resources is done at runtime, in keeping with service orientation. This dynamic binding is therefore dependent on the context in which the binding occurs.

Figure 3. Conceptual model for dynamic service binding

After service binding the SM starts any interaction task in conjunction with the IA and also any FTA task and executes the workflow preserving the precedence constraints or the conditions that are specified in the workflow. At runtime a Workflow object aggregates a number of Task objects where each object represents a task in the workflow. The services that this task requires for running are divided into input and output services and are connected with the appropriate resources. The resources that are bound to the Task object can be either devices that the Task directly controls (i.e., input sensors and actuation devices) or agents, such as the IA or the FTA. In either case the Task object is informed on the status of the resource and operates according to the pattern specified by its type. The sequence diagram in Fig. 4 shows the basic interaction of the software components during the instantiation of the “Feel Comfortable” AS, which employs the dynamic service binding process mentioned earlier. In the diagram, this process is implemented by the methods used inside the two loops.

In addition, the SM handles exception events that affect the configuration of the AS. For example, exceptions during the execution of the workflow, such as disconnection or failure of devices trigger an adaptation of the workflow by rebinding services to alternative devices. Context changes during the execution of the workflow may invalidate preconditions that were valid during the workflow instantiation. For example, if the user changes location and a follow-me property has been defined for a display service, then the execution state needs to be updated and a new display service instance to be scheduled. In order to achieve workflow adaptation, replanning capabilities may be required by the PA. Replanning comes into play when the dynamic binding fails during workflow execution or update. When replanning is requested a new planning problem is defined with the services that are actually available, and the PA solves the problem and delivers a new workflow.

The planning problem can be stated as “discover an execution path of services (tasks) given some state of the world to achieve a goal”. In ATRACO, we use a library of abstract plans which model specific user goals. An abstract plan contains a sequence of abstract services which are actually ontological descriptions of service operations that cannot be directly invoked, but will be resolved by the SM during runtime. Having an abstract service workflow description, which is given in a BPEL-like language, the Dynamic Service Binding module of the SM applies a semantic-based discovery mechanism and uses information about available services and context, acquired by the SO through the OM, to discover suitable services or devices in registries able to perform each abstract service. The output of this process is an executable service workflow. In the execution management and control phase the SM executes and continuously monitors the deployed services and the termination condition of the workflow. Fig. 3 gives a conceptual view of the dynamic service binding process. A workflow is mapped into a number of tasks and a workflow task is mapped into one or more abstract services. In addition, each service would also require certain physical resources for its implementation. Mapping of the task to the services can be specified at design time by the PA as per users’ functional requirements. However, mapping of the service to the actual human and physical resources is done at runtime, in keeping with service orientation. This dynamic binding is therefore dependent on the context in which the binding occurs.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>1</th>
<th>Task</th>
<th>PA responsibility</th>
<th>1</th>
<th>Service</th>
<th>1</th>
<th>Resource</th>
<th>1</th>
</tr>
</thead>
</table>

Figure 3. Conceptual model for dynamic service binding

After service binding the SM starts any interaction task in conjunction with the IA and also any FTA task and executes the workflow preserving the precedence constraints or the conditions that are specified in the workflow. At runtime a Workflow object aggregates a number of Task objects where each object represents a task in the workflow. The services that this task requires for running are divided into input and output services and are connected with the appropriate resources. The resources that are bound to the Task object can be either devices that the Task directly controls (i.e., input sensors and actuation devices) or agents, such as the IA or the FTA. In either case the Task object is informed on the status of the resource and operates according to the pattern specified by its type. The sequence diagram in Fig. 4 shows the basic interaction of the software components during the instantiation of the “Feel Comfortable” AS, which employs the dynamic service binding process mentioned earlier. In the diagram, this process is implemented by the methods used inside the two loops.

In addition, the SM handles exception events that affect the configuration of the AS. For example, exceptions during the execution of the workflow, such as disconnection or failure of devices trigger an adaptation of the workflow by rebinding services to alternative devices. Context changes during the execution of the workflow may invalidate preconditions that were valid during the workflow instantiation. For example, if the user changes location and a follow-me property has been defined for a display service, then the execution state needs to be updated and a new display service instance to be scheduled. In order to achieve workflow adaptation, replanning capabilities may be required by the PA. Replanning comes into play when the dynamic binding fails during workflow execution or update. When replanning is requested a new planning problem is defined with the services that are actually available, and the PA solves the problem and delivers a new workflow.
Given the above requirements a variant of BPEL, called ATRACO-BPEL, was defined in order to provide those ATRACO specific features needed in order specify workflows. In the following we explain how using the ATRACO-BPEL formalism an example task is bound with the appropriate service(s). The task AdjustLights is associated with the partnerLink AdjustLightsPL as part of the orchestration logic section:

```xml
<bpel:invoke name="AdjustLights" partnerLink="AdjustLightsPL"/>
```

The partnerLink AdjustLightsPL has an input role called ATRACO:lightStatus and an output role (partnerRole) called ATRACO:triggerLight. The Continues type denotes that the execution of the activity is to be treated as a task that is running continuously, i.e., the workflow does not wait its termination.

```xml
<bpel:partnerLink name="AdjustLightsPL"
partnerLinkType="ATRACO:Continuous"
myRole="ATRACO:lightStatus"
partnerRole="ATRACO:triggerLight"/>
```

The input role ATRACO:lightStatus denotes the appropriate abstract service that must be bound to fulfill the role (Luminosity) along with any other application specific details that are needed for its operation e.g., the task will be monitored by an ATRACO agent for learning user behavior with respect to light adjustments and all found light devices are to be used.

```xml
<ATRACO:role name="lightStatus" type="input" Agent="yes" IAmode ="none">
  <ATRACO:service semantics="Luminosity" trigger ="Low" reset = "none" quantity = "all" rules="">
    </ATRACO:service>
</ATRACO:role>
```

The corresponding definition for the output role will be:

```xml
<ATRACO:role name="triggerLight" type="output" Agent="yes" IAmode ="withAgent">
  <ATRACO:service semantics="Actuate Light" trigger ="On" reset = "Off" quantity = "all" rules=""></ATRACO:service>
</ATRACO:role>
```

In ATRACO-BPEL each partnerLink role is specialized as an ATRACO:role which is a new definition in ATRACO-BPEL. In each ATRACO:role the attributes listed in Table 1 are defined.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the role.</td>
</tr>
<tr>
<td>type</td>
<td>Denotes the type of the role. Accepted values are input/output.</td>
</tr>
<tr>
<td>Agent</td>
<td>This attribute defines whether the task is monitored by an ATRACO agent or not. Accepted values are yes/no.</td>
</tr>
<tr>
<td>IAmode</td>
<td>Specifies the interaction mode with the ATRACO Interaction Agent. Accepted values are: none – no interaction is needed; pure – this value is used to indicate that a single interaction with the user through a dialog interface (spoken, tangible or software) needs to be provided either to provide a message or to receive an input for the system from the user in a form of...</td>
</tr>
</tbody>
</table>
Each ATRACO:role envelopes a set of services that are bound to it. Each role can have more than one abstract service. If the role type is input then the activity waits for all the services to deliver their result before proceeding. If the role type is output then, upon activity completion, all the services enveloped in this role are triggered. For each abstract service specific attributes are defined, providing the necessary support for device discovery and service operation. Table 2 summarizes the service-specific attributes in ATRACO-BPEL.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>semantics</td>
<td>The semantics of the service as a set of keywords – these are used to find the specific device that can be bound to this abstract service</td>
</tr>
<tr>
<td>trigger</td>
<td>input role: denotes a linguistic value that triggers the service</td>
</tr>
<tr>
<td>output role:</td>
<td>denotes a linguistic value passed to the service</td>
</tr>
<tr>
<td>reset</td>
<td>The reset state (linguistic value) that the service should apply in the case that the activity cannot be performed</td>
</tr>
<tr>
<td>quantity</td>
<td>A number that defines how many devices providing this service are needed for the specific activity. If the value is “all” then all found devices are used.</td>
</tr>
<tr>
<td>rules</td>
<td>Any special constraints need to be met for binding the corresponding device(s).</td>
</tr>
<tr>
<td>IAdlg</td>
<td>This attribute is associated with the direct or pure interaction modes with IA in order to give it the proper interaction dialog type. Examples of accepted values are: GreetingMessage, LightInstructions, GrantGuestAccess, MusicQuestion, MusicControl, PhotoFrameQuestion, SlideshowControl.</td>
</tr>
</tbody>
</table>
the nodes denote invocations to service operations, and the edges denote control-flow dependencies. When a service node is invoked, a search recipe is executed to select a reference to a specific service. Once a service is selected by the search recipe, the eFlow execution engine is responsible for performing the dynamic binding using metadata that it stores in the service repository. A Polymorphic Process Model has been proposed in [19] to model abstract workflows without immediately requiring the implementation details of each activity. The SWORD toolkit uses a rule-based expert engine for determining how to construct a composite Web service from primitive services [20]. However, these approaches are targeted towards automation of business processes in the internet and not towards supporting users to perform activities in intelligent environments. The use of workflows to describe user tasks and interactions in smart environments has been explored in [21]. However, the approach is based on static workflows that do not react to changes in the context of the environment.

V. CONCLUSION

Normally, workflow management systems have not been used for dynamic environments requiring adaptive behaviour. On the contrary, in ATRACO we require adaptive workflows which need to react to varying environmental conditions. Our general idea is that since a workflow describes the relationship between services and if an agent is represented by such a service, then the relationship between the agents would be possible to specify. Following such a combined agent-based and SOA approach means that a workflow could be used to establish the initial relationships of the multiagent system. Multiagent systems can be specified then first with a workflow description using ATRACO-BPEL that defines the most common scenario and fault conditions. Once the basic system has been deployed, the agents could be working proactively so they can adapt to unforeseen circumstances and automatically handle the extension to the workflow description. In addition, run-time adaptations of services and devices are possible since workflows specify abstract services that are bound dynamically. This gives the opportunity to handle events such as a device failure or using a high-quality service that can replace a service selected in the first-place. In that sense, we can see our workflows as adaptive workflows.

As an extension, ATRACO will handle in the near future the simultaneous deployment of multiple workflows in the same space which may cause synchronization problems where different workflows may compete for the same resources and perform conflicting actions. In the future we would like to test our system in public intelligent spaces like shopping malls, museums, airports, etc., to explore how such spaces can be enhanced with services and mobile devices to assist visitors in performing their tasks more efficiently.

ACKNOWLEDGMENT

The research described is partly supported by the ATRACO (ICT-216837) project.

REFERENCES