A METHODOLOGY FOR DESIGNING AND MANAGING CONTEXT-AWARE WORKFLOWS

S. Modafferi[1], B. Benatallah[2], F. Casati[3], and B. Pernici[1]

[1] *Politecnico di Milano*
Dipartimento di Elettronica e Informazione
P.zza L. Da Vinci 32 - 20133 - Milano Italy

[2] *University of New South Wales,*
CSE, Sydney NSW 2052, Australia

[3] *Hewlett-Packard, Palo Alto, CA, 94304, USA*

Abstract: The increased availability of context information and the widespread adoption of more and more powerful devices creates the opportunity and desire for context-aware applications. In this paper we focus on a specific but important type of applications: workflow applications. Just like other applications, workflows too require context-aware capabilities, that is, require the capability of modeling business logic that is sensitive and varies depending on the users' context. In this paper, we propose a methodology for context-sensitive business processes development. We extend existing process modeling languages to allow modelling context sensitive regions (i.e, parts of the business process that may have different behaviours depending on context). We also introduce *context change patterns* as a mean to identify the contextual situations (and in particular context change situations) that may have an impact on the behaviour of a business process. Finally, we propose a set of transformation rules that allow generating a BPEL-based business process from a context sensitive business process. This allows using existing process engines to support context-sensitive business processes.

Keywords: User Context, Context-aware Workflow, Adaptive Workflow, Context Sensitive Regions, Dynamic Workflow Execution

1. INTRODUCTION

Capturing and managing user context is becoming more and more important in business applications. With recent advances in mobile technologies that provide several ways for identifying user contexts (e.g., identifying user location) (Capra et al., 2003; Chakraborty and Lei, 2004;

Dey and Abowd, 2001) and for leveraging context-based applications, the need and opportunity for delivering customized services to users in different situations is becoming prominent (Roman et al., 2000).

Generally speaking, *context* refers to information that characterizes the situation of a person, place, or object, that is relevant to a given system. In our work, we focus on context information that is relevant to provide personalized services to users based on their environment and needs. This includes information such as user location, current user device, or network bandwidth. Existing work in delivering personalized services focused mostly on capturing and representing context information (Chakraborty and Lei, 2004; Dey, 2001). This is clearly an interesting and necessary step as it provides for recognizing context as a separate abstraction, and fosters the development of tools and techniques for context management.

However, delivering personalized services to users requires methodologies and techniques that not only allow developers to capture and manage context, but that also facilitate the creation of context-aware applications. As a particular and very important kind of (context-aware) applications, in this paper we consider *workflow* applications, and more in general applications developed using composition technologies (also called process technology in the following). Workflow applications are rapidly gaining popularity, especially with the advent of Web services and the push towards service-oriented computing, as the availability of homogeneous components (services) makes it easier to develop applications by composing existing building blocks.

Specifically, the aim of this paper is to identify techniques that facilitate the development of context-sensitive processes, including in particular the ability to manage context change, a key issue in any context-aware application development. These techniques are based on what we believe to be simple but essential extensions to "traditional" process models. Indeed, the main challenge in this work has been that of identifying the process modeling concepts that could capture the essence of context-sensitive applications, or at least a wide variety of them, while avoiding to unnecessarily making process modeling more complex.

Our work builds upon existing techniques for managing context information and extends them by providing a methodology and an architecture for developing context sensitive processes. We propose the concept of *context-sensitive region* to localize parts of the process that have different behaviors according to context. A set of *context change patterns*, that classify and capture typical context changes is defined. Context-sensitive regions include the description of how to react to a context change defined in a context change pattern. The proposed model will

be used as a basis to generate the BPEL process and exception handlers to manage context-sensitive processes.

The paper is structured as follows: in Section 2, we discuss the integration of context in business process models. Section 3 presents our approach for modeling context-sensitive business processes. The runtime aspects are presented in Section 4. Section 5 presents related work, while Section 6 gives conclusions and future directions.

2. CONTEXT-SENSITIVE BUSINESS PROCESS MODELING

2.1 Context Definition

Delivering personalized services requires a precise definition of user needs and user environments. Traditional service provisioning relies on a relatively static characterization of the user and the user's context, because the changes in the user environment are relatively limited and because of the inability to dynamically and automatically capture context and context change (Roman et al., 2000).

Several definitions of context have been proposed in the literature. In (Dey, 2001), context and context-aware computing have been defined as:

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*
>
> *"A system is context-aware if it uses context to provide relevant information and/or services, where relevancy depends on the user's task."*

In general, the user context can consist of many different aspects, and different context models include different properties as elements of a context. The most common and typical context attribute is the geographical location, which can be expressed at different levels of granularity (XYZ coordinates, city, state, etc). Other context information may include a "logical" location (e.g., "in a meeting" or "at home"), the present occupation, the weather at the user's location, and many other attributes.

For the specifics of the context model, the present paper is based on the one developed within the MAIS project (Cappiello et al., 2005). In particular, besides generic location-related user attributes, the MAIS model focuses on user access devices (e.g., PDAs, laptop, or others) and quality of network connectivity. However, it should be noted that while we consider a specific context model as example, the concepts presented in the paper are generally applicable to virtually any context model. In fact, even if in our approach workflow design is influenced by consideration of context, the use of high level representation of context

decouples the problem of managing workflow schema modification and of context model evolution.

2.2 Context-Sensitive Process Models

There are many ways in which processes can leverage context information to create context-sensitive applications. One is to make routing decisions. For example, information about skiing conditions can be delivered to users that are in a mountain area, but not to others. Another usage of context information is to configure a given service invoked as part of the process, or to select a specific provider among the ones able to provide a given service. For example, a news delivery service may be configured to send videos of different quality based on the network bandwidth or user screen resolution. Finally, an important aspect is context change management, which involves the ability of the process to alter its course based on new context information about the user to which the process is delivering a service.

In prior work, there is a strong relationship between application and context model and very often they are developed together.

In our approach, the user context is considered as first class citizen in business processes. Each process has an implicitly defined set of data elements (process variables) that capture the user context. These variables are automatically populated and maintained by the infrastructure. For example, the user context may include the notion of "user device", which can assume the values of PDA and PC. The context is monitored by a *context monitor*, which measures context variables according to what is called *Low level context model* (see Fig. 1). We use the term "Low level" to denote that it is typically at a low level of abstraction (e.g., a location can be expressed in xyz coordinates). For this a conventional context representation model is used (i.e., (Cappiello et al., 2005; Chakraborty and Lei, 2004)). This model provides means to represent user context attributes and user context changes (also called context change events), such as decrease in throughput or change in location. The low level context is then mapped to high level context *dimensions*, which form the high level user context (or simply user context hereafter). This mapping is necessary to abstract from the details of the context (and from context changes) that would be too finegrained and detailed for most workflow applications. Each dimension can have a set of predefined values (High, Low; PDA, PC), and correspond to the process variables that are implicitly included in each workflow models.

The Context Mapping module is in charge of translating measured context values into values for the dimensions at the higher level, accord-

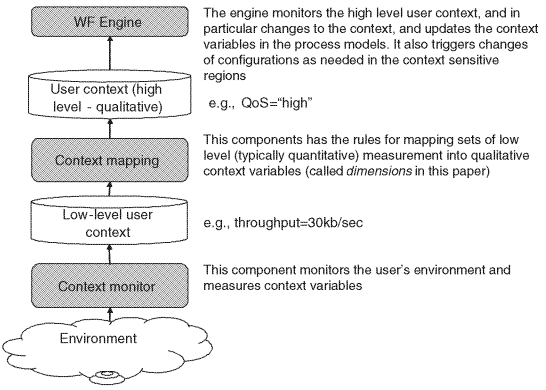

Figure 1. Relationship between workflow and context

ing to a set of Context Rules. For instance, if in the low level context the variable representing the throughput of the net decreases, this fact is transparent to the High Level User Model until a given threshold is reached, after which a rule in the Context Mapping component determines that the throughput dimension has now the value"Low Qos".

To facilitate context-sensitive process modeling we also propose in the workflow model the concept of *context-sensitive* region. A context-sensitive region is a part of the business process that may have different behaviors (e.g., different flow structures) depending on the context. A region is associated to several configurations - essentially corresponding to subprocesses - that represent the different behaviors. Whenever the region is instantiated, a specific configuration, among the possible ones, is selected based on the context.

The designer has to define an "entry condition" for each region. It is a boolean condition defined over the user context. The set of entry conditions of configurations for a region must define a partition over the user context. An"otherwise" condition can also be specified to capture all cases in which no entry condition is verified. Note that if the entry conditions do not define a partition, then the behavior of the system is non deterministic (the system currently does not enforce that conditions create a partition of the user context).

The problem of capturing and managing context change is at the heart of context-sensitive applications. The key here is to devise a model that makes it easy for developer to manage simple context changes, possibly with minimal or no modeling, while giving also the possibility to manage

complex context change requirements. To this end, we propose an approach that is based on schema evolution techniques and migration rules. Schema evolution is the default context change behavior provided by the model, and the nice aspect of it is that it requires no modeling from the user. Briefly, it works as follows: as mentioned, the system takes care of automatically monitoring the context. This also happens when a workflow instance is within a CSR. At some point, it may be that the system detects that the entry condition for the CSR in which an instance is (called "initial configuration") turns out to become false, and the entry condition of another configuration (called "target configuration") becomes instead true (one entry condition must be true as conditions identify a partition). In this case, the problem is handled analogously to what is done in workflow schema evolution: the execution of the instance is rolled back until a point in the CSR is reached where the execution of the instance in the initial configuration "can be seen as" an execution of the instance in the target configuration, at which point the instance is migrated to the target configuration and continues as prescribed by the process flow in the target configuration. As a simple example, the initial configuration can be composed by a sequence of tasks A,B,C,D, while the target configuration is a sequence of tasks A,B,X,Y. If the context change occurs while the instance was performing task D, then the execution of D is aborted, and the execution of C is compensated. At this point, the execution of the instance in the initial configuration (A,B) can be seen as an instance of the target configuration (which also starts with A,B). Hence the instance is migrated and the next task to be activated is X, according to the flow in the target configuration. In the worst case, the compensation of the initial configuration continues to the start of the CSR. We do not detail schema evolution further as it has been the subject of many papers (although in different scenario - our contribution here is its application to context change management and the idea to use this approach to simplify process definition and maintenance). The interested reader is referred to (Casati et al., 1998; Reichert et al., 2003).

Schema evolution proposes a default behavior that may be inadequate in some situations, for example due to the excessive loss of work due to compensation of completed activities. To handle this cases, our approach proposes the notions of *context change patterns* and of *migration rules* to respectively capture the interesting change events and manage the change with ad hoc behavior Context change patterns capture the different ways in which a context can evolve and help defining how the change is relevant from a process model perspective. Patterns help characterizing and classifying the different types of context changes and as such simplify the definition of how to handle a change within a

context-sensitive region. Indeed, the combination of context regions and context change patterns allows for flexible and easily maintainable process models, which are even robust to the evolution of the context model itself. A Transition is an instantiation of a Pattern. It causes changes within active workflow instances, and specifically in the context-sensitive regions sensible to the related context-change pattern. For a given transition in the User Context, the user may associate a Transformation in the Workflow model. A transition is analogous to an ad hoc migration. It includes the explicit identifications of points in the workflow where an instance should migrate from the initial to the final configuration (regardless of whether at that point the execution in the initial configuration can be seen as an instance of the target configuration), along with rules to manage the migration (e.g., data transformation).

2.3 Context Change Patterns

To enhance the design of context-aware workflows, it is important to determine patterns that capture common behaviors followed by the users to which the workflow provides a service. We identified four patterns which cover several interesting situations as well as a framework for building new patterns. As stated in the previous paragraph, patterns in our model are the basis for context state transitions (e.g. for the pattern "Device dependency" the transition can be "PDA to Pc" and "PC to PDA"). It is possible to define new interesting dimensions and then new pattern as transitions in this dimension. The identified patterns are:

Device dependency. It represents the situation when a user changes device, but is still involved in a business process. For example, the user uses a PDA initially, being out of office, and then connects through a desktop later on, when in the office, continuing to interact with the same process. The specificity of this pattern is that it could provide for data movement; in fact during a process some information can be stored only on the client side and, changing the client device, it is necessary to perform a migration of this data to the new device.

Qos driven choice. It represents the possibility of having different business process configurations according to different QoS levels in the client side. A typical situation is the abrupt decrease of available bandwidth that requires a substitution/reconfiguration of some services on the server side.

Location driven choice. It represents a situation where business process configurations are associated to given user locations. For example, the selection of a restaurant from a list will be different depending on the current city and country, offering different selection services to

the user and, if the location based choice is enough grained, a user can change location within a business process several times.

User on-line/off-line. Due to resource consumption constraints, user may prefer to work off-line. This pattern represents the typical working mode of a user connected through a mobile device. The interaction of wfms engine with the user is not continuous and during the user absence the process can show a different behavior waiting for his return.

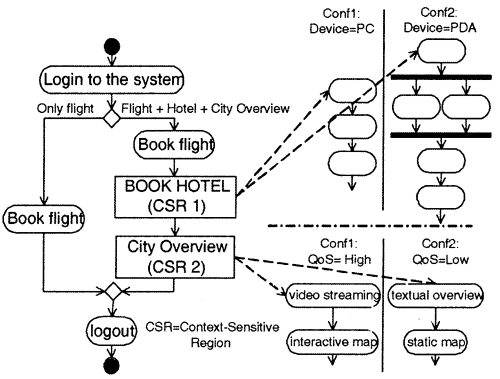

Figure 2. Wf with regions

2.4 Example

We illustrate the proposed approach using a simple example, taken from the travel reservation domain. In the sample process, a user can login in the system, and is then faced with the following two possibilities: i) book a flight; ii) book a flight, a hotel, and also get information about the destination city.

Assume that the flight booking task is context independent while the other tasks are context-sensitive (i.e., they have different behaviors according to the current user context). Fig. 2 shows the model of such business process.

The "Book Hotel" Region (CSR1) is sensible only to the Pattern *Device Switching* and thus the interesting Dimension is only "Device"; in the user context model there are two states "PC" and "PDA" and in the workflow model two configurations for the region. The difference between configurations are mainly that if the user device is a PDA the server will send in parallel part of the information to the PDA and part to a different client: a mailbox.

The "City Overview" Region (CSR2) is sensible only to the Pattern *Qos driven choice* and thus the interesting Dimension is only "QoS"; in the User Context Model there are two states "High QoS" and "Low QoS" and in the Workflow Model two configurations for the Region. The difference between configurations are that if the QoS is high the services are interactive (e.g. dynamic map), otherwise they are simple static information (e.g. textual information). Further details about the example will be presented in Section 3.2.

3. MODELING ABSTRACTIONS

In this section we will introduce our model. The Workflow model is composed of Context Sensitive Regions and each region is composed by different configurations. Also context model aspects are discussed.

3.1 Workflow Model

To describe the workflow model we define a workflow as directed graph. Our definition is based on traditional definition of workflows as graphs and then only specific constructs are presented here. In our model a Workflow W is composed by a set of i) Traditional coordination C and task T nodes; ii) Regions Reg; iii) Directed flow arcs F_W.

Let $Wgraph =< NW, FW >$ be a graph where NW: finite set of nodes, FW: directed arc (flow relation) $FW \subseteq NW \times NW$

$\forall nw \in NW, Nodetype : nw \rightarrow \{Coordinator, Task, Reg\}$

$NW = C \cup T \cup Reg, \quad (C \cap T = \emptyset) \wedge (C \cap Reg = \emptyset) \wedge (T \cap Reg = \emptyset)$

where C: set of coordinator nodes (e.g. switch, loop), T: set of Task nodes, Reg: set of Context Sensitive Region.

Context Sensitive Region. A Context-sensitive region (hereafter CSR) is a subprocess of the workflow that may have several configurations exporting different behavior according to specific conditions (i.e. user context).

A region is composed of alternative configurations linked with particular arcs called *migration arcs.* A migration arc is associated with instructions on how to migrate a workflow instance from one configuration to another.

A typical example of information associated to a migration arc is the data items that need to be moved when a context change occurs.

For example the *Device Switching* pattern provide for the change of the device and this fact can imply the necessity to move some data from the old device to the new one (see also the example in Section 3.2).

Configuration. A Configuration is a subprocess of the workflow related to a set of states in the User Context Model (i.e. $User\ Device = PDA$). A Configuration $Conf_i$ is composed by: i) an entry condition EC; ii) normal coordination C and task T nodes; iii) a set of Starting Migration Points MPs; iv) a set of Ending Migration Points MPe; v) a set of directed Configuration Arcs FC.

The entry condition EC is an expression used to define when the configuration has to be entered.

To handle the change of configuration there is a default behavior. This default behavior requires no modeling and no effort by the user. The system migrates the workflow from a configuration C1 to another C2. Migrating essentially means rolling back the execution of C1 up to the point where the instance can be seen as an instance of C2, as for the work in Workflow evolution (Casati et al., 1998). At that point the instance is migrated and the process continues. This also means that each task in this model has to be associated to a (possibly null) compensation action.

For the most complex cases, the user can explicitly specify migration rules. These rules are modeled using migration points and migration arcs. A starting migration point is a point in a configuration $C1$ where a direct link, that is a migration arc, to a new configuration is available. Assume the flow is in a configuration C1 and has to migrate to a configuration C2, if in the rolling back process a migration point and a migration arc from C1 to C2 is found then if is followed to perform the transformation, otherwise the default behavior is exploited. The definition of migration point is similar to the notion of safe-point in the WIDE project (Grefen et al., 1999).

Migration points and migration arcs provide high expression power to specify migration rules. For example they allow the designer to define business equivalent behaviors only by connecting two points and, in case, associating a process to the arc. This power has a cost in terms of what the designer has to specify, so we give this possibility in our model, but we expect the default behavior to apply often and we do not realistically expect a CSR to have a large number of configurations. Hence, this model, and in particular the combination of automated (default) migration along with the possibility of specifying migration rules for special cases is able to combine the need for ease of modeling/manageability with the possibility of defining specific migration semantics for the most complex cases. Now a formalization of the configuration is provided.

Let $Conf =< EC, NC, FC >$ be a graph where EC: the expression for the entry condition for the configuration, NC: finite set of nodes, FC: directed arc $FC \subseteq NC \times NC$

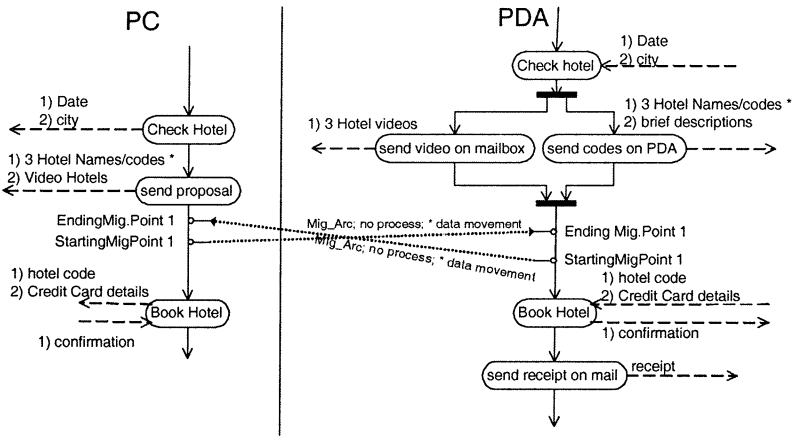

Figure 3. Region Book Hotel configurations for device switching pattern

$\forall nc \in NC, Nodetype : nc \rightarrow \{Coordinator, Task, StartingMigration Points, EndingMigrationPoints\}$

$NC = C \cup T \cup MPs \cup MPe$,
$(C \cap T = \emptyset) \wedge (C \cap MPs = \emptyset) \wedge (C \cap MPe = \emptyset) \wedge (T \cap MPs = \emptyset) \wedge (T \cap MPe = \emptyset) \wedge (MPs \cap MPe = \emptyset)$

where C: set of coordinator nodes, T: set of Task nodes, MPs: set of Starting Migration Points, MPe: set of Ending Migration Points.

Formalization of CSR. Let $Reg_i =< \bigcup_i Conf_i, MA >$ be a graph where $Conf_i$: a Configuration, MA: directed arc $MA \subseteq MPs \times MPe \times Op \times Proc$ where Op: set of optional Operations, $Proc$: set of optional Processes.

3.2 Example

Let us suppose to design a process starting from scratch. The first step of our methodology is the definition of the high-level workflow already presented in Section 2.4 and shown in Fig. 2. This step provides a high-level summary of the entire workflow and allows considering in the further steps each region as a stand alone part. The second step is the definition of each CSR.

Book Hotel CSR (see Fig. 3). This CSR is sensible to "Device Switching" pattern. It shows two different behavior according to the used device.

If the user is using **PC**, after the reception of a query, the server will send him information about some hotels and also a video. Then it will

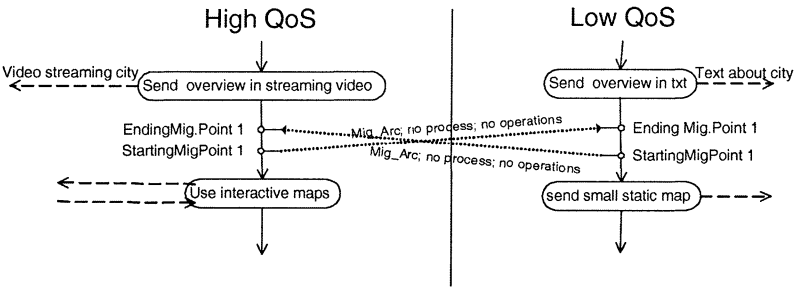

Figure 4. Region City Overview configurations for QoS driven pattern

provide the task for booking according to codes provided by the user. All operations are in sequence.

If the user is using **PDA**, after the reception of a query, the server will send him on the PDA some brief information about hotels and will send on the user email the video of the hotels. These operation are carried out in parallel. Then the server will wait for the hotel code, provided by the user, to make the reservation. Finally it will send a receipt on the user email. There is not a task for receipt in PC configuration because we suppose that the interaction is by web-browser and so the printout of the page is enough as receipt. It is less probable that a PDA is connected to a printer and so we provide a receipt on email box.

In Fig. 3 there are a pair of migration arcs. Whichever is the transformation (PC to PDA or PDA to PC) starting from MPs1 (Starting Migration Point 1) the migration arc is associated with an operation for "data movement". This operation allows moving data on the client side (in this case the hotel codes marked with a * in the figure). In fact, even if, the process is running on the server side, it is possible to have data movement in the client side. The optional process associated with the migration arc represents a process related to the configuration change and it is defined considering the two configurations and the starting, respectively ending, migration point.

The default exception management is used if the user context changes before the flow meets a migration point, that is schema evolution rules are applied. In the other situations the migration is driven from the migration arc starting from the last migration point. In Fig. 3 the migration arcs expresses the business equivalence of operation executed until the migration point. The system can not determine automatically this equivalence, in charge of the business logic, and thus the designer had to explicitly define it.

City Overview CSR (see Fig. 4). This CSR is related to the pattern "QoS driven choice". We assume that the user device configuration is sensible to different level of QoS (i.e. he is using an UMTS connection). With **High QoS** the first task is used to send a video streaming about the city. Then the user can use interactive city maps.

With **Low Qos** the first task is used to send some small texts about the city. Then the server will send to the user a small static city map.

In this CSR all the migration arcs are not associated with processes or other operations. Here the migration arcs are used to express the equivalence in terms of behaviours. Differences are in terms of realizations of the same behaviours according to different contexts.

4. RUN-TIME ARCHITECTURE

The output of the design phase will be a WS-BPEL process annotated with context abstractions to represent context-sensitive regions. . Each region is modeled as a stand-alone part. We propose transformation approach to translate a context sensitive WS-BPEL process into a conventional WS-BPEL process with the appropriate handlers.

We assume that a service oriented architecture is used to manage user context (this kind of approach is followed in several systems, e.g (Chakraborty and Lei, 2004)). In our approach the context manager is designed as a web service. Context changes are conveyed to the WS-BPEL engine as messages (context change event, with parameters) from the context manager. Context changes are managed in the WS-BPEL as exceptions and therefore the unified BPEL is composed of a normal behavior (conditional according to the context) and a context change handler to manage context exception.

The handler checks if an automatic migration (following the traditional schema evolution methods) is available. If it is not available the handler performs a reverse path compensating each task until reaching the last starting migration point, then it performs the actions and processes associated with the migration arch leading to the new configuration and, eventually, it restarts the normal flow from the ending migration point (in the new configuration) reached with this migration.

Fig. 5 shows the Ws-BPEL unified schema (with specific operation for region switching) that is the output of the CSR shown in Fig. 3 after the automatic generation process. This schema is referred to a standard WS-BPEL and the corresponding behavior is context-dependant according to the design presented in the previous sections.

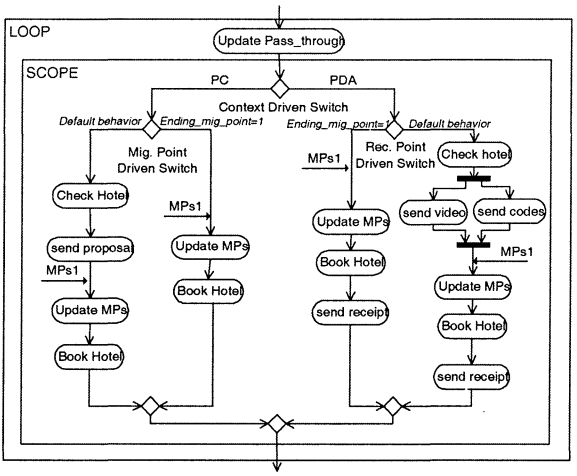

Figure 5. Context Sensitive Region of Fig. 3 as unique WS-BPEL schema

5. RELATED WORK

The possibility of modifying workflows to increase their flexibility is widely studied (Casati and Shan, 2001; Müller et al., 2004; Reichert et al., 2003; Shan et al., 2005). The reason of workflow modification are various and not strictly related to the user "context" issue. The context-aware workflow issue concerns many and different aspects that are now being linked together. The most common definition of context is provided in (Dey, 2001); generic context models like (Chakraborty and Lei, 2004; Dey and Abowd, 2001) can be the basis of context-aware applications; different middleware have been proposed to interpret the context providing useful information to the applications (Bellavista et al., 2003; Capra et al., 2003). A way followed to build context aware applications can be the definition of self-contained systems like (Zariskas et al., 2001). They have their own definition of context, are mobile-oriented and context-aware, but they do not focus workflow systems. Other systems are workflow-based, but they focus on single context-sensitive tasks (Long et al., 2004; Patil et al., 2004; Sheng et al., 2004).

The work presented in (Binemann-Zdanowicz et al., 2004) proposes an approach to context-awareness for Web Information Systems that distinguishes among the various kinds of contexts, but it is not clear how it manages together workflow execution and its rich context model.

Some recent papers propose the extension of workflow languages and models with aspect-oriented software design (Charfi and Mezini, 2004). They suggest the usage of aspect-orientation as a complementary tech-

nique for workflow modelling and specification presenting a hybrid approach for realizing the integration of business rules (modelled as aspects) with a WS-BPEL orchestration engine by using aspect-oriented programming techniques. Our definition of high level context can be viewed as an Aspect and the proposed solution is how to exploit these different aspects by using a standard, even if annotated, workflow model and a standard workflow engine.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a methodology for designing and developing Context-sensitive business processes.

By identifying patterns for capturing context changes, our approach provides a loose coupling between workflow definition and context model. By means of Context-Sensitive Region, a high level construct for modelling context changes is provided and, eventually, by annotating an existing workflow model a solution based on the currently available WS-BPEL for the run-time management is provided.

We are studying a more general use of regions defining region-level properties that are specific to a context, but not necessarily related to the fact that a context may change. In fact even if the context is always the same from start to end in a given workflow execution, it is possible to define region-level properties that depends on the context, e.g., attributes such as data transfer rates or resolution, used by all activities in the region.

ACKNOWLEDGEMENTS

This work is partially funded by the Italian MURST-FIRB MAIS Project (Multi-channel Adaptive Information Systems) and by a visiting research grant from School of Computer Science and Engineering of the University of New South Wales (UNSW) Sydney.

References

Bellavista, P., Corradi, A., Montanari, R., and Stefanelli, C. (2003). Dynamic binding in mobile applications: A middleware approach. *IEEE Int. Computing*, 7(2):34–42.

Binemann-Zdanowicz, A., Kaschek, R., Schewe, K., and Thalheim, B. (2004). Context-aware web information systems. In *Proc. of Asia-Pacific Conference on Conceptual Modelling*, pages 37–48, Dunedin, New Zealand. Australian Computer Society.

Cappiello, C., Comuzzi, M., Mussi, E., and Pernici, B. (2005). Context management for adaptive information systems. In *In Proc. of Int. Workshop on Context for Web Services (CWS)*, Paris, France.

Capra, L., Emmerich, W., and Mascolo, C. (2003). Carisma: Context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering*, 29(10):929–944.

Casati, F., Ceri, S., Pernici, B., and Pozzi, G. (1998). Workflow evolution. *Data Knowl. Eng.*, 24(3):211–238.

Casati, F. and Shan, M. (2001). Dynamic and adaptive composition of e-services. *Information System*, 26(3):143–163.

Chakraborty, D. and Lei, H. (2004). Pervasive enablement of business processes. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PERCOM)*, Orlando, (Fl), USA.

Charfi, A. and Mezini, M. (2004). Aspect-oriented web service composition with AO4BPEL. In *Proc. of European Conference on web Service, (ECOWS)*, pages 168–182, Erfurt, Germany. Springer.

Dey, A. (2001). Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7.

Dey, A. and Abowd, G. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction Journal*, 26(2-4):97–166.

Grefen, P., Pernici, B., and (Eds), G. S. (1999). *Database Support for Workflow Management - The WIDE Project*. Kluwer Academic Publishers.

Long, Y., Lam, H., and Su, S. (2004). Adaptive grid service flow management: Framework and model. In *Proc. of IEEE Int. Conf. Web Services (ICWS)*, pages 558–565, San Diego, Ca, USA.

Müller, R., Greiner, U., and Rahm, E. (2004). AGENTWORK: A workflow-system supporting rule-based workflow adaptation. *Data and Knowledge Engineering*.

Patil, A., Oundhakar, S., Sheth, A., and Verma, K. (2004). Meteor-S web-service annotation framework. In *Proc. of Int. Conf. WWW*, pages 553–562, New York, NY, USA.

Reichert, M., Rinderle, S., and Dadam, P. (2003). ADEPT workflow management system. In *Proc. of Int. Conf on Business Process Management BPM*, pages 370–379, Eindhoven, The Netherlands. Springer.

Roman, G., Picco, G., and Murphy, A. (2000). Software engineering for mobility: a roadmap. In *In proc. of Inter. Conf. on Software Engineering (ICSE) - Future of SE Track*, pages 241–258, Limerick, Ireland.

Shan, E., Casati, F., and Dayal, U. (2005). Adaptive process management. *to appear in Int. Journal on Business Process Management*.

Sheng, Q., Benatallah, B., Maamar, Z., Dumas, M., and Ngu, A. (2004). Enabling Personalized Composition and Adaptive Provisioning of Web Services. In *Proc. of Int. Conf. on Advanced Information Systems Engineering (CAiSE)*, pages 322–337, Riga, Latvia. Springer.

Zariskas, V., Papatzanis, G., and Stephanidis, C. (2001). An architecture for a self-adapting information system for tourists. In *Proc. of the Workshop on Multiple User Interfaces over the Internet: Engineering and Applications Trends (in conjunction with HCI-IHM')*, Lille, France.