A SURVEY ON CONTEXT-AWARE WORKFLOW ADAPTATIONS
Sucha Smanchat, Sea Ling and Maria Indrawan
Faculty of Information Technology, Monash University
ssmanchat@acm.org, {chris.ling, maria.indrawan}@infotech.monash.edu.au

ABSTRACT
Recently, workflow has been employed in pervasive computing systems to orchestrate and provide services to the users. Since pervasive computing systems emphasise context-awareness and adaptability, these two features must be included in the workflow mechanism in order to utilise workflow in pervasive environment. In this paper we present various existing approaches to adaptation in context-aware workflow. These approaches are compared based on the common adaptation characteristics identified from our study. Lastly, we discuss and point out the future challenges of predictive adaptation for context-aware workflows.

Categories and Subject Descriptors
A.1 [Introductory and Survey]; H.4.1 [Information Systems Applications]: Office Automation – workflow management; H.1.0 [Models and Principles]: General

General Terms
Theory, Design

Keywords
Workflow, adaptation, context-aware, pervasive computing

1. INTRODUCTION
Context-aware systems have become prevalent research in the area of pervasive computing. They are systems that provide services to the users in a more powerful and user-centric way. Some of the context-aware systems have employed workflow mechanism to orchestrate the services for the users [10, 12] as workflow is a tool that enables the coordination between services available in heterogeneous environments [12]. In order to implement workflow in a context-aware system, the important features of context-aware system which are context-awareness and adaptability must be accommodated into the workflow. In this paper we focus on the various adaptation approaches in the context-aware workflow and identify the characteristics shared by these approaches.

From our study, we have found that the adaptations are usually included as a part of context-aware systems. However, many context-aware systems do not actually perform adaptation, they merely use context to select or provide the tasks, services, or workflow [13, 19]. According to the differentiation between context-aware systems and adaptive context-aware systems by Efstratiou [5], a system is considered adaptive if it changes its behaviour in response to the change of context. In this paper, we define context-aware workflow adaptation based on [5] as the action that modifies workflow in a way that changes workflow behaviour in response to the change of context considered relevant to that workflow.

The next section describes the adaptation characteristics we identify from our study. Section 3 describes, compares, and discusses the various approaches to context-aware workflow adaptations and section 4 is the conclusion of this paper.

2. CONTEXT-AWARE WORKFLOW ADAPTATION CHARACTERISTICS
This section describes a set of common characteristics of the adaptation in context-aware workflow identified from our study which helps compare the different adaptation approaches.

2.1. Objectives of Adaptations in Context-aware Workflow
There are many reasons for the adaptation in context-aware workflow which affects the mechanism of adaptations. Based on the categorization in [6], we group the objectives of context-aware workflow adaptation into three types which are customisation, correction, and optimisation. Customisation refers to the adaptation that is triggered by the change of context related to user’s need such as the need for new services or the replacement of an original service. The workflow is adapted to accommodate the new requirement or context.

During execution, a service that is bound to a task might become unavailable and the workflow needs to adapt to compensate the failure. This corrective type of adaptation deals with faults and usually involves rolling back the failed task and restarting a part of workflow [14]. The adaptation can also be employed to optimise the performance of workflow such as to improve workflow completion time or to detect and solve bottlenecks in the workflow [9].

2.2. Adaptation Strategies
Based on the adaptation strategies proposed by Müller et al. [12], the strategies for context-aware workflow adaptations can be reactive and predictive. The reactive adaptation takes place when the sensed context triggers the need for adaptation for the task that is about to be executed. On the other hand, a predictive adaptation
analyses the workflow further down the track from the currently executed task and adapts the affected part of workflow in advance which helps to avoid wasteful resource allocation [12].

2.3. Workflow Adaptation Levels

Context-aware workflow adaptations can take place in three different levels depending on the different adaptation techniques. The abstract workflow, employed by some approaches as the top level, defines the workflow construct that contains workflow tasks without the details of execution. Abstract workflow allows for adaptation in the workflow by determining the actual implementation of tasks in the abstract workflow at execution time based on the context [1, 2, 15].

Before or during execution, the abstract workflow may be mapped to a concrete workflow definition, the second level, with execution details [2, 9]. The concrete workflow is the definition that is executed by the workflow engine. The adaptation in this level involves the modification of workflow definition [4, 9] and the propagation of such change to workflow instances. Workflow instance, the last level, is an instance of concrete workflow definition for an execution. These adaptation levels imply different workflow management. Workflow instance adaptation can be handled easier while the adaptation in the concrete definition requires the handling of the running workflow instances.

2.4. Service Binding Time

In order to execute a workflow instance, the actual services must be assigned to the tasks specified in the instance. This process of service binding can happen before or during execution, also identified by earlier literature [16, 17]. The former means that the binding of all the tasks in a workflow is completed before the workflow instance is executed. This can cause failure if an assigned service becomes unavailable. The latter, also called late binding or late modelling, means that the tasks in the workflow are bound to the services as the execution proceeds to each task in the workflow instance. The late binding provides more flexibility but requires run-time service discovery and that the list of available services must be maintained [16].

2.5. Acquisition of Substitutes for Adaptation

The adaptation in a workflow usually involves adding, dropping, and replacing tasks in the workflow [12]. The substitute tasks for adaptation can be acquired at run-time or predefined before the workflow is executed. User might be able to manually add new services to the system for the adaptation at run-time and also a new service might be discovered and become available [4, 6]. This allows for more flexibility in the adaptation. On the other hand, the substitute tasks can also be predefined by the user to ensure that the adapted workflow will produce desirable results.

2.6. Granularity of Restarting Workflow

Adaptation may involve some degree of restarting the executed tasks. There might be no restarting at all in some approaches which usually employ late binding. Other approaches might restart zero of more tasks depending on the context. The entire workflow might be required to restart in some approaches that specifically consider a failure of a certain task as a failure of the entire workflow.

3. CONTEXT-AWARE WORKFLOW ADAPTATION APPROACHES

This section describes and discusses existing approaches to adapt context-aware workflow in various domains. Although some of these works do not directly use the workflow concept [3], their domains, such as web service composition [15], imply the use of workflow. Therefore they are also included and described here.

3.1. Adaptation Approaches

The context-aware workflow adaptations can be done in many approaches including specifying all the possible adaptations in the workflow definition itself, utilising abstract workflow concept, and adapting the workflow instance. Wieland et al. [18] use the simple approach by specifying all the possible adaptation choices as the execution paths in the workflow definition of factory production workflow; the selection is based on the context at run-time. This method is not flexible since the workflow definition must be changed every time a new adaptation is needed and is only suitable for a workflow that does not change often [18]. As a more flexible approach, Chaari et al. [3] propose a generic framework for context-aware adaptation in pervasive environment by employing a functional model. Each service in the functional model is attached with a set of service versions which can be added at run-time. During execution, either the original service is executed or is replaced by one of the available versions based on various types of sensed context and the predefined adaptation rules [3]. Therefore, the adaptation in this approach takes place at the instance level and the service is bound during execution. Choi et al. [4] propose another adaptation in pervasive environment to support modification of workflow at run-time. Each service is modelled as sub workflow which can be inserted into the main workflow. If the context conditions are satisfied, that service will be executed. In this approach, the adaptation takes place at the workflow definition level and is reflected in the running instance.

Many approaches utilise abstract workflow concept. Ardissono proposed abstract workflow in “CAWE” (Context-Aware Workflow Execution) framework [2]. In this approach, the abstract workflow contains one or more abstract activities which are replaced by one of the concrete implementation, which are predefined, to customise the service for the user. This limits the flexibility of this approach but it ensures that the adapted workflow gives the expected result and is suitable for rigid workflow such as the hospital workflow presented as their scenario. Adams et al. [1] propose the similar concept called “worklet” which is substitutable sub-process. The “parent” workflow or the main workflow can have the task within marked for worklet substitution which is selected from the available set of worklets during execution by using “Ripple Down Rule” (RDR) [1]. The additional worklets can be added during the execution; this provides more flexibility than the approach proposed in [2].

Narendra et al. [15] propose the adaptation in workflow in pervasive computing by using “variability point”. This point, similar to the worklet [1], defines a part of the workflow to which the web service is selected and bound during execution. Taking into account the resource context, this approach may require the executing task to restart if the resource becomes unavailable. In their later work, a corrective adaptation is also proposed to respond to a failure of task [14]. The pivot task, the crucial task whose failure means the failure of the entire workflow, is used to
determine the scope of the adaptation called “Sphere” [14]. A failed task, if not a pivot, can be retried, replaced, or abort.

Modafferi et al. [11] propose the context-aware workflow adaptation through the use of “Context-Sensitive Region” (CSR). The CSR is a part of workflow that allows the execution to choose the appropriate implementation during execution. The difference between this approach and the abstract activity [2] is that CSR contains “migration point” [11] that allows switching between predefined implementations which may involves rolling back executed tasks before migrating to another implementation [11]. The drawback is the limited flexibility since the implementations for CSR and the migration points must be predefined. Since the approaches in [1], [2], [11], and [15] implement the adaptation by the selection of substitute during execution, these approaches implement adaptation at the workflow instance level.

Context-aware workflow is also employed in grid environment as Lee et al. [9] use abstract workflow to adapt grid workflow. Their main aim is to improve the performance of the workflow by utilising context-awareness and “MAPE” (Monitoring, Analysis, Planning, and Execution) model. The resources in grid environment are monitored in order to detect the performance problems such as bottleneck and the opportunities to improve performance such as a new execution node becoming available. The adaptation in this approach involves the mapping between abstract workflow and concrete workflow, and the scheduling of tasks such as increasing concurrency [9]. This is the only approach in our study that proposes the adaptation in every workflow adaptation level.

In order to handle the failure in web service composition, Erradi et al. propose the middleware “wsBus” [6] located between the business process (workflow definition level), and the services (instance level). wsBus employs the late service binding to avoid the complexity in the business process definition; the actual web service is selected during execution. wsBus monitors the execution of web services and triggers adaptation to deal with web service failures and timeout to ensure the completion of business process. The adaptation action is determined by predefined policies and is done at the instance level to avoid the modification of the business process definition [6]. In their later work, “MASC” middleware [7] is proposed as an improvement over wsBus. MASC also enables customisation adaptation to respond to the change of user’s need [7]. Although wsBus and MASC use predefined policies to specify adaptation action, the substitute services might be acquired and changed at any time during the execution.

Müller et al. [12] propose workflow adaptation in the workflow management system “AgentWork”. The adaptation in this approach aims to customise the hospital cancer treatment workflow to suit each patient’s medical profile by adding and deleting tasks in the running workflow instance according to the predefined extended ECA rules [12]. The adaptation in this approach predicts the affected part of the workflow to adapt it in advance, and optimises the adapted workflow to make sure that the time constraints of the treatment are not violated. At a later time, “AdaptFlow” system [8] is developed based on AgentWork for real use. This approach is by far the only approach that suggests and implements predictive adaptation strategy.

3.2. Comparison and Discussion

From the comparison shown in table 1, the acquisition of substitute for workflow adaptation varies depending on the application domains. The substitutes should be predefined in the workflows whose domains are imposed with strict constraints such as production factory [18] and hospital [2, 8, 12], to ensure expected result and the standards or protocols are not violated [8]. On the other hand, to allow for flexibility, workflows in the domain such as pervasive computing and business workflow should bind the services and allow the substitutes to be added at run-time which also helps avoid failure due to a service becoming unavailable [1]. In many approaches, the adaptation actions are specified in the form of rules, such as ECA [12], RDR [1] and policies in [6]. These rules must be manually maintained making the systems unsuitable for non-technical users. Most of the adaptation approaches such as [2, 6, 7, 14] focus on the adaptation at the workflow instance level to avoid the complexity of handling the modification of workflow definition.

It is notable that only the approach proposed by Müller et al. [8, 12] suggests and implements predictive workflow adaptation (see Table 1). The other approaches adapt the workflow as it encounters choices or problems, and therefore are only reactive. In the area of pervasive computing, predictive adaptation in context-aware workflow can be valuable by predicting and adapting the workflow in advance to provide smooth services to the users. The predictive adaptation in pervasive environment might consider functional and non-functional dependencies of the adapted tasks such as the effect on the system environment and the context itself.

For example, a context-aware workflow for arranging transportation contains a task to book for flight ticket, a task to arrange airport pickup, and a task to make hotel reservation. Before the workflow is executed, the system is notified from the weather forecast service that there will be heavy snow storm on that day. The system predicts that the flight is likely to be cancelled and replaces the task with a task to book for a train ticket. As a result, the task to arrange for airport pickup becomes irrelevant to the workflow. The system should also be aware of this effect and recursively adapts by replacing this task with the task to arrange pickup at the train station. Hotel reservation might be postponed or cancelled. The workflow should be able to predict this adaptation by using the estimated time the train takes to reach the destination, the task replacing the airport pickup, and the availability of the accommodation.

4. CONCLUSION

In this paper we identify the characteristics of context-aware workflow adaptation which are used to compare various approaches to context-aware workflow adaptation. We also come to the conclusion that there is room for further research in predictive adaptation and introduce recursive prediction in context-aware workflow in pervasive environment as a possible solution. We aim to further investigate this possibility as it can be a powerful extension to the pervasive computing research area.
Table 1 The summary of comparison between context-aware workflow adaptation approaches

<table>
<thead>
<tr>
<th>Authors</th>
<th>Workflow Domain</th>
<th>Objectives of Adaptations</th>
<th>Workflow Adaptation Level</th>
<th>Service Binding</th>
<th>Adaptation Strategies</th>
<th>Acquisition of Substitutes</th>
<th>Granularity of Restarting</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10]</td>
<td>Production Factory</td>
<td>Customisation</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
</tr>
<tr>
<td>[3]</td>
<td>Pervasive computing</td>
<td>Customisation</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
</tr>
<tr>
<td>[4]</td>
<td>Smart home</td>
<td>Customisation</td>
<td>Concrete workflow</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Not specified</td>
</tr>
<tr>
<td>[2] CAWE</td>
<td>Hospital workflow</td>
<td>Customisation</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
</tr>
<tr>
<td>[1] workshop</td>
<td>Business process</td>
<td>Customisation</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
</tr>
<tr>
<td>[11] CSR</td>
<td>Business process</td>
<td>Customisation</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Zero or more tasks</td>
</tr>
<tr>
<td>[9] Grid computing</td>
<td>Optimisation</td>
<td>All levels</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Zero or more tasks</td>
<td></td>
</tr>
<tr>
<td>[6] wsBus</td>
<td>Business process</td>
<td>Correction</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Zero or more tasks</td>
</tr>
<tr>
<td>[7] MASC</td>
<td>Business process</td>
<td>Customisation, Correction</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Zero or more tasks</td>
</tr>
<tr>
<td>[15] Pervasive computing</td>
<td>Customisation</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Single task</td>
<td></td>
</tr>
<tr>
<td>[14] Business process</td>
<td>Correction</td>
<td>Workflow instance</td>
<td>N/A</td>
<td>N/A</td>
<td>✓</td>
<td>Single task up to entire workflow</td>
<td></td>
</tr>
<tr>
<td>[8, 12] Medical treatment</td>
<td>Customisation, Optimisation</td>
<td>Workflow instance</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

5. REFERENCES


