Support for business process flexibility in service compositions: An evaluative survey

Malinda Kapuruge, Jun Han and Alan Colman
Faculty of Information and Communication Technologies
Swinburne University of Technology,
Melbourne, Australia
{mkapuruge, jhan, acolman}@swin.edu.au

Abstract— **Service compositions provide a promising way to realize and coordinate automated support for business activities and processes. These business processes and their automated support need to survive in highly volatile market and technical environments. Consequently, many approaches have been proposed to address the issue of flexibility support in business process modeling and enactment. However, it remains a challenge for researchers and practitioners. In this paper we provide a systematic analysis of the requirements for process flexibility in the context of service compositions, and analyze the existing approaches against these set of requirements. Based on this analysis, we draw some general observations and point to some critical issues for future investigation into business process flexibility support in service composition.**

Keywords: Business process, process flexibility, automated process support, services composition, SOA.

I. INTRODUCTION

Business process management (BPM) provides automated support for business operations to achieve business efficiency and capture new business opportunities[1]. However, business requirements and the environment under which the business operations are carried out are subject to continuous changes. As such, the automated support for the business operations needs to evolve and reflect such changes. For example, business operations may need to be optimized in response to changing operational and market conditions. Capturing these changes effectively and reflecting them in business process management in a timely and non-disruptive manner, is critical and represents an ongoing challenge.

In recent years, service oriented systems that compose services in a loosely coupled manner, have emerged as a new paradigm to provide such automated support for business processes. These systems aim to capture new business opportunities in highly dynamic market places[2]. The survival of businesses is dependent on how flexible the service compositions are in adjusting to variability in their operating conditions. This requires that the techniques used to model and enact the business processes in service compositions be flexible.

Traditional workflow based techniques used in service compositions are usually graphical, based on imperative software programming principles, and explicitly specify the ordering of activities [3-5]. This explicit ordering of activities in a workflow makes it relatively easy to understand the coordination of activities even for a non-technical stakeholder. However, this imperative nature of workflow based business processes strictly prescribes the order of activities, and the activities are considered to be atomic [6-8]. This results in rigid and hard to change business process definitions. Such rigidness does not help to capture the volatile, contextual and ubiquitous nature[9] of business processes within service compositions.

The rigidness and imperative nature of workflow based process modeling techniques can be seen in WS-BPEL[10], which is considered as the de-facto standard for specifying business processes in web services compositions[11]. Consequently, many efforts [12-15] have attempted to overcome the WS-BPEL flexibility limitations. Furthermore, there are also efforts to come up with alternative ways with different perspectives to model and enact business processes, such as constraint-based, product-based, and event-driven paradigms [7, 12-16]. In general, the challenge of providing effective flexible process support remains. As such, there is the need to have a systematic analysis of the process flexibility issue and the existing approaches to further our understanding and effort in addressing this issue.

Several surveys have been carried out on business process modeling from different viewpoints. Surveys have variously focused on formal underpinnings [1]; conditional structures of workflow management systems [17]; and , artifact centric process models[18]. Other surveys have focused on process flexibility. An analysis of flexibility and other properties for graph-based and rule-based approaches to business process modeling is provided in[19]. A classification of flexibility in terms of applicable techniques is provided in[20], but without systematically evaluating existing approaches. A taxonomy of flexibility is provided in[21], but restricted to control flow concepts. In general, these existing surveys have not systematically considered the requirements for process flexibility in services compositions (and consequently evaluated existing approaches against such requirements), which is critical to understand the issue of process flexibility and provide corresponding modeling and enactment support. Furthermore, none of them have considered how the business and technical relationships inherent in a service composition constrain the flexibility of business processes that use those services.

In this paper we provide a systematic analysis of process flexibility in service compositions providing specific example scenarios. In particular, we identify a set of requirements for process flexibility from the perspectives of process definition, process instance and service relationship of a service composition. We further evaluate the existing approaches to process flexibility support against these set of requirements. We then draw some general observations and point to some critical issues for future investigation.

Section 2 presents a business scenario that we use to illustrate our discussion. Section 3 analyses and identifies the requirements for process flexibility. Section 4 evaluates existing approaches to process flexibility support against the flexibility requirements. In section 5 we summarize the evaluation and present some overall observations. In section 6 we point to some critical issues for future investigation before concluding the paper in section 7.

1530-0803/10 $25.00 © 2010 IEEE
DOI 10.1109/ASWEC.2010.27

II. SCENARIO

In order to motivate the discussion about business process flexibility requirements in a services composition, let us consider the following scenario.

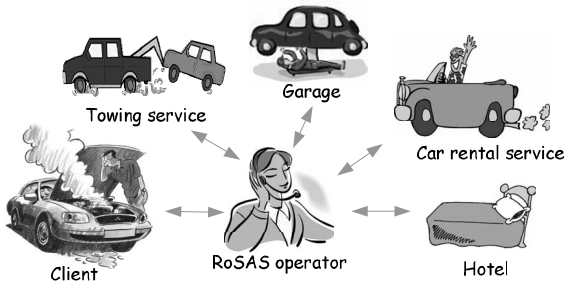

Figure 1: RoSAS - The Road Side Assistance Service

Assume there is a business organization called RoSAS (RoadSide Assistance Service) Pty Ltd., which assists its pre-registered clients when their cars breakdown. The services may include towing the car, finding a suitable Garage, providing an alternative car and even arranging accommodation. The type of services provided is dependent on the customer agreement, location of the breakdown, and the degree of damage. Therefore RoSAS needs to have a flexible modeling and enactment approach, so that they can change their business processes to optimize operations, broaden their business offerings and handle unexpected situations. For example, some specific requirements of RoSAS are,

1. Re-arrange activities, e.g. requesting the Tow Car before requesting the Rental Car and vice-versa.
2. Change service providers based on the location or availability, e.g. selecting nearest Garage, selecting an alternative Garage upon unavailability of the nearest.
3. Introduce new services. E.g. airlines, paramedic services upon business requirements.
4. Merge business processes to optimize business coordination in the long run. E.g. possibly merging car repair and car towing processes.

Moreover, it is necessary to do such alterations in a timely manner without interrupting the business operations.

III. REQUIREMENTS FOR PROCESS FLEXIBILITY

A business *process definition* needs to evolve over time to accommodate new business and technical requirements of a service composition. Multiple *process instances* are instantiated based on this definition to perform the actual work at runtime for each business situation of the same kind. While in operation, these instances have to deal with exceptions. Also process instances might need to be optimised.

Moreover, a service composition can be seen as a collection of *services relationships*. As an example, RoSAS composition is a collection of the relationships among composed entities, upon which the RoSAS business depends on. As an example, there are mutual obligations, constraints and performance levels of operation between the Garage and The Tow Car services. Similarly the Garage may keep relationships with other entities such as the Client and RoSAS operator. These relationships, which are defined by the composition, aggregate to classify the "position" of the Garage Service. Process definitions are defined on top of these relationships to coordinate the activities of these positions. And there are dependencies between the process definitions and services relationships, and both evolve over time. Consequently, the flexibility in process puts requirements on the flexibility of the service relationships and is constrained by the degree of flexibility in the service relationships.

In general, the process flexibility requirements in a service composition can be categorized into the above mentioned three levels: process instance, process definition and services relationship (see Figure 2). Process instances are created according to a process definition for different applicable situations, but may deviate from the definition. Process definitions are formulated within the constraints of the service relationships but pose flexibility requirements on these relationships.

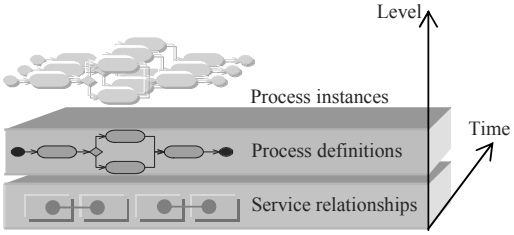

Figure 2: Levels of process flexibility

Overall, the flexibility allowed at these three levels should be realized in such a manner that it prevents the composition from evolving into states that violate business constraints. In other words, a business process modeling and enactment approach should be able to maintain business invariants, allowing changes in a controlled manner. For example the flexibility at the process instance level should not allow violations of any business regulations.

Furthermore, the change operations need to be applied in a timely manner, because slow and overdue change operations may lead to unsatisfactory situations, even though the change decision is appropriate.

In the following subsections we identify and analyse in detail the requirements that allow one to *model* and *enact* business processes in a flexible manner.

3.1 Process definition flexibility

Business process definitions are subject to numerous modifications during the analysis and design phases. This can happen even during the enactment phase as process definitions need to be refined to reflect the unpredicted changes. If the process model is not flexible enough, the designer will find it difficult to apply the changes correctly and in a timely manner. This would result in an outdated or inaccurate process definition. To provide the necessary flexibility support at the process definition level, we have identified the following set of requirements.

3.1.1 Configurable design: Process definitions should be expressed in such a way so as to make them as configurable as possible. To achieve this, it is necessary to identify the configurable points, aspects of a process definition, and there should be support from the modelling and enactment approach to make the identified points, aspects configurable. As an example, the RoSAS designer might need to make the crosscutting concerns (e.g. units of measurements) as configurable as possible.

3.1.2 Inbuilt context awareness: Operations and changes to them should be based on the context it is operated in. For example, in the RoSAS system factors like holidays, severe weather conditions and communication breakdowns can be

vital to making decisions during enactment. Therefore it is necessary to have built-in context awareness in the process definition constructs and the ability to readily redefine context conditions.

3.1.3 Ease of understanding: The representation of how activities are coordinated in a process definition needs to be easy to comprehend. This encourages users to constantly improve the business processes with fewer or no mistakes. For example, if the complex RoSAS business processes are scripted using their proprietary standard and no tool support given for visualizing, then the changes can be time consuming and error-prone.

3.1.4 Late specification: This is the ability to partially define process definitions to be completed later during enactment[21]. For example, a RoSAS process might partially specify the constraints and activities on handling client requests due to lack of knowledge or very complicated nature of the runtime. Such as the duration of repair, the activities involve to repair the car etc. But during the enactment, it should be possible to improve them as the knowledge become available.

3.1.5 Automated change verification: Changes may jeopardize the consistency and correctness of the business process representation. For example, a change in "pay using direct debit" shared-sub-process might be valid for "pay Rental Car" process but not for "pay Tow Car" process. If it is not possible to verify the correctness of the resulting definition, users might be reluctant to introduce due changes, and changes introduced may well contain inconsistencies. The effort and time needed for such verification should also not be excessive. Checking tools that use formal representations of the process definition are therefore required.

3.1.6 Merging Business Process Definitions: As the business evolves, it might be required to merge two process definitions. Suppose that RoSAS has two separate processes: one to book the Garage service and the other to request the Tow Car. Later the modeling team recognizes the mutual dependencies between the activities/decisions of the two processes. As an example, there could be a tendency that many Garages provide car towing. Manual merging might be tedious for the team as it requires mapping variables, identifying overlaps, and detecting conflicts etc. Therefore it is required to have formal mechanism and associated tool support to guide the merging procedure [22].

3.2 Process instance flexibility

Process instances are runtime entities passing through different states[23] during its lifetime. They derive behaviour from the process definitions. However, it is not possible to assume that the process instances can strictly adhere to the behaviour they derived from the definitions. The runtime environment might pose challenges for the survival of the instance, compelling for a change. Usually changes at the process instance level are applied in ad-hoc manner[7, 24]. This is mainly due to the impossibility of predicting the runtime behaviour during the design phase. Therefore it is essential to have some degree of flexibility at the process instance level as discussed below.

3.2.1 Process instance deviation: The process instance behaviour should be able to deviate from the defined process by applying some ad-hoc changes in a controlled manner[25]. The changes may include skipping activities, dynamically proposing/selecting alternative paths of execution etc. These kinds of changes are not intended to be repetitive and are applied to avoid failures or to optimize a process instance. For example, the specified process in RoSAS may indicate that the Tow Car should be requested before the repair. But, for a particular instance a car might breakdown in front of a Garage. In that case the "request Tow Car" activity should be skipped. However, this may not be reflected in the process definition since it is an in-frequent situation. For such a situation, a small deviation from the specified behaviour would be sufficient.

3.2.2 Process instance handling as a case: For some businesses it is required to treat each process instance (a customer requirement) uniquely and isolated manner more often than not. In our scenario, RoSAS might require treating each customer request in a unique way as the type of breakdown, location, available services and customer preferences can be critical to select the activities and the coordination. The process designers might find it tedious to include all the possible paths of operations to the definition as it would make the design very complex. Or in the worst case it might not be possible at all. Therefore it is easier to drive the process based on the information available for the process instances as a case[12] under certain business constraints.

3.2.3 Process instance migration: Process definitions get changed or replaced during runtime. At the moment of change or replacement, some of the already instantiated process instances may not be terminated. For example, RoSAS might change the "Hire Rental Car" process as customers have to pay upfront for bookings later than 10PM, due to new government regulations. However, at that moment there could be many instantiated processes running. Cancelling them might adversely affect the business. Therefore having the ability to make them migrate to the newly introduced process definition would increase the chance of survival of the process instance.

3.2.4 Ease of human understanding and intervention (if required): Although the automation of business processes can deliver many benefits, it is essential to have human intervention when required. As an example, RoSAS might have automated the "client registration and credit evaluation" processes but would like to let an officer in charge to intervene on some critical decisions. Therefore it should be possible for the officer to study and understand the process instance in terms of its state and associated data and then intervene.

3.3. Services relationship flexibility

As discussed above, a service composition can be seen as defining the relationships among the different entities of the composition. For example, at the design time RoSAS may define the relationships between the Garage and the Tow Car company, the Garage and the Client, etc. These relationships define the mutual obligations, accepted performance levels etc. The aggregation of all the relationships that the Garage has with other entities of the composition defines the requirements for the Garage service as shown in Figure 3. These requirements can be seen as the "position" or "abstract service description" for the specific Garage services. A potential Garage service will be selected and bound, according to the description, to be part of the business process upon agreement. The abstract service descriptions, inter-service relationships and the bindings of external services to abstract service descriptions

are shown in Figure 3. A service such as Tom's motor works bound to the Garage position is obligated to perform according to the relationships that the Garage position has with other positions of the composition. As the business processes are subject to changes in both the definition and instance levels, it is possible that these bindings, interfaces and contractual relationships are required to change. Conversely, the degree of flexibility in these bindings, interfaces and relationships constrain the flexibility that can be achieved at the process definition and instance levels.

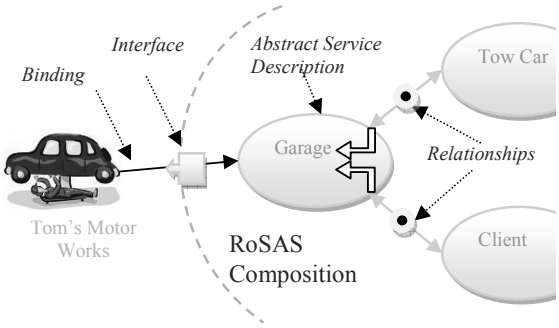

Figure 3: Service bindings, abstract service descriptions and relationships

3.3.1 Ability to change service interfaces: Services expose their abstract services descriptions via interfaces, which can be further categorized to functional, non-functional and behaviour interfaces. These interfaces provide the information about the method signatures and protocols of communication with the composition. RoSAS should be able to define and re-define these interfaces if necessary. For example, due to a change in the business process and upon inability to find a suitable Garage, the RoSAS should be able to modify or customize the Garage interface as the ultimate solution.

3.3.2 Ability to change service bindings: External services are bound to the service composition to do some useful task. However, at runtime these services may fail, under-perform or become inappropriate with respect to a process instance. In such situations the composite should be able to replace these services at runtime. For example, the RoSAS should be able to select and bind a Garage service depending on the severity/location of the breakdown.

3.3.3 Ability to change inter-service relationships: Service composition defines the relationships among different entities of the composition. An aggregation of such relationships results in set of abstract service descriptions as shown in Figure 3. However, circumstances might ask for changes in these relationships too. For example, RoSAS may define that the Tow Car should bring the car to the Garage within 5 hours, and the Garage should produce the quotation within 8 hours. In order to achieve higher process flexibility at the definition and instance levels, RoSAS should be able to change these terms of the relationships at runtime.

IV. Evaluation of Existing Approaches

In this section we analyse how the flexibility requirements discussed in the previous section are addressed by a selected list of approaches [8, 13, 14, 26-29] aimed at achieving process flexibility support. A large amount of work has been done in the past on this regard. Our selection of approaches was based on the evidence of implementation and also the uniqueness of the methodology, in order to highlight the diverse ways to achieve the above requirements. However, none of the approaches have addressed all the requirements mentioned above. Further we have extended our discussion, including some approaches where necessary, due to the important and specific contribution towards fulfilling a particular requirement.

4.1 Process definition flexibility

Configurability of the business processes is related to the software engineering principles of modularity and consistency. Many existing approaches to process flexibility modularize the configurable aspects and provide mechanisms to re-design a process by altering the allowed configurations. The AO4BPEL approach provides configurability by breaking down the business process into an abstract core-process and well-modularized business rules[29]. These business rules are woven in an aspect oriented way with the abstract processes and can be changed to modify the business processes in a consistent manner. However, the point-cuts where rules are interleaved with the process definition need to specified at design-time. Graml et al. also uses a similar approach by integrating business rules with such business process aspects as decision nodes, data constraints and control flow to make the process definition configurable and adaptable[28]. In the worklet approach the configurability is achieved by having an extensible repertoire of self-contained sub processes[27]. This repertoire and the selection rules improve the configurability of the definitions. In the event driven approach proposed by Alexopoulou et al., the events, actions and their causal relationships are configurable[14]. An Event Specification Language is being used to define the complex event action relationships. The declarative constraint-based approach by Pesic et al. [7] allows change the process behavior by reconfiguring the constraints and their mapping to process definitions. The approach uses two declarative languages, Condec[30] and DecSerFlow[31] to configure the constraints using constraint templates.

Inbuilt context awareness is provided to make the decisions more applicable to the operating environment. Having such capability in process definitions, improve the flexibility at runtime. Some approaches attempt to add that support explicitly in their process definition mechanism. Such an effort can be seen in the approach proposed by Graml et al.[28]. It allows business rules to access the "business process instance context" explicitly, in order to make the business processes context-aware. In the worklet approach[27] self-contained sub processes are selected contextually using Ripple Down Rules[32]. The data driven (case-driven, artifact driven) approaches guide process instances via the values and statuses of the data objects. Therefore there is no error-prone "context tunneling" [12] in these approaches as in control flow based approaches.

Ease of understanding of the process definition is predominantly achieved by visualizing the coordination. Many flexible process definition approaches provide tools to apply changes in a visual editor. Usually the imperative process definitions are easier to visualize showing the coordination among activities. Comparatively, the declarative approaches need to derive the coordination from a given process instance/case or to show the most probable coordination. Furthermore, visual tools are used to declare the constraints and data constructs. For example Pesic et al.[26] uses the Declare tool[33] along with the Condec, DecSerFlow languages[7, 30] to declare the constraints which eventually guide the process instances. Moreover, it is possible to visualize the coordination among activities with its YAWL integration[7]. The case handling paradigm

proposed by van der Aalst el. [8] utilizes the FLOWer[12] studio to specify the data objects, roles and forms. Similarly Graml et al. in their business rules integration[28] uses the websphere visual tools[34]. Also the hybrid approach by Charfi et al. uses the process view of the AO4BPEL engine to visualize the associated aspects for a given process[29].

Late specification is a well practiced technique in business process modeling when and where it is impossible to predict the possibilities, or the possibilities are too complicated to model. Adams et al. in their work, use abstract process definitions which can be further constructed using self-contained sub-processes or worklets[27]. The worklets repertoire is extensible allowing more worklets to be added at runtime. This allows specializing of existing sub processes or providing alternatives to them as the knowledge grows. The constraint model specified by Pesic et al. declaratively specifies constraints that should not be violated[26]. More constraints can be added/removed or softened/hardened at runtime as more knowledge becomes available. Similarly in data-driven approaches [12, 15] more data objects can be modelled at a later time. Other modelling patterns, such as "sub-process selection", as specified by Graml et al.[28], is also a practice the late specification technique. The approach allows defining process fragments at the runtime and also to modify the business rules to integrate them. The Events, actions and event-action relationships are allowed to be specified later by Alexopoulpu in their event-driven approach[14]. Somewhat interesting late specification can also be seen in the dynamic routing approach as the process is specified as the definition template is routed among actors[13]. Each actor specifies the next actions based on the outcome of their actions realizing the late specification technique. In this approach there is no predefined definition but the definitions get constructed at runtime as the control passes among actors. AO4BPEL approach also allows late specification of aspects[29]. However, the scope of late specification ability may be limited by the way the point-cuts is specified in the business processes.

Automated change verification can be provided as a tool if the modeling approach is based on a formal representation. Thus many flexible process modeling approaches use some kind of formalism. For example Declare[26] is based on Linear Temporal Logic(LTL)[35]. Declare models based on LTL expressions can be translated into automation and could be verified with model checking tools such as SPIN[36]. Worklets[27] too are checked out from the engine by evaluating worklets against the case data of a particular process instance. Thus modifications such as substitutions are verified automatically against the case data. Reichert et al. define a minimum set of operations to verify the control-flow modifications of running process instances[37]. They use a set of correctness properties and upon violation of those properties the proposed change gets rejected.

Business process merging is a rather complicated and error prone activity compared to small modifications as it involve mapping activities, decision points, states and data structures. Although there is no direct reference to merging business processes among our selected list of approaches, the constraint driven approach by Pesic et al.[26] provides an easy way to merge the business process definitions. In fact in their approach a constraint set can be seen as a process definition which specifies what shouldn't be violated. The constraint model mapping mechanism can be used to merge such two different constraint sets and thereby process definitions. Similarly two process definitions built with worklets can be merged rather easily due to their self contained-ness[27]. The merging can be enacted by modifying the selection rules. Apart from the above selected approaches, Sun et al.[38] discuss a number of concepts and methods for workflow merging. They have identified four categories of process merging, i.e. Sequential, Parallel, Conditional, and Iterative. An algorithm is also provided to carry out the merging process. Nemo et al.[22, 39, 40] present an approach that addresses issues like preserving the partial order of activities, unifying input/output data and detecting multiple calls to same services. However, the approach is not fully automated and requires semantic inputs to take certain decisions from the developer who interact with the merging tool, to avoid conflicts.

4.2 Process instance flexibility

Process instance deviation occurs because the business process definition cannot fully capture the complexity of the runtime in detail. As a solution to this, Pesic et al. allows changes to soft constraints for a process instance to deviate from the original behavior at runtime [7, 26, 30]. The constraints are specified declaratively using LTL allowing process instances to be progressed without violating the constraints. In the case handling approach, the process instance (case) is associated with data objects[12]. The case is entirely handled based on the data values. Also the type of the data objects (free/ mandatory/ restricted) and the case completion conditions can be altered for the instance. Barros et al. propose an approach that routes the control of the process instance among different actors(e.g. a role such as accountant) allowing them to add subtasks and refine constraints[13]. Adams et al. follows a rather different approach as the deviation is carried out by a rule based contextual selection of atomic process segments[27] to replace the existing segment at runtime for each instance. Graml et al. alter business rules that are integrated with the BPEL script to allow the process instances to deviate from its predefined behavior[28]. In the event-driven approach the Event-Action relationships can be altered to deviate a particular process instance[14]. The aspects are being used in AO4BPEL to deviate the instances based on outcomes of the rule evaluation[29]. However, the deviation may not be specific to a process instance. The approach rather provides a way to modularize the aspects common to process definitions.

Process instance handling as a case is achieved by many approaches to let the user focus on the state and data of a given process instance. van der Aalst et al. [8, 12] discuss the applicability of treating each individual process instance as a case and provide an approach to realize it. Vanderfeesten et al. [15, 41] use a similar model to realize product driven workflows. Their focus is not the control-flow but rather achieving the goal. The driving force is not the process but rather the data. The approach proposed by Barros et al. also routes the process instance as a case among different actors, dynamically, until the case is completed[13]. In the event-driven approach (by Alexopoulou et al. [14]) the initial "*event Id"* is propagated to all subsequent events to manage the process instance as a case. A rather different approach is taken by Pesic et al. in their constraint driven approach[26]. They allow mapping/re-mapping single process instances to Constraint Model (CM). However, the mapping is based on a process identifier which might not be enough to capture the data associated with the case or the process instance.

Process instance migration is handled in Pesic et al. [26] by re-mapping a process instance (P_{id}) to a constraint model (CM) via a constraint model identification (CM_{id}). The CM remapping is allowed only if there is no historical violation. In this way it is also possible to let multiple

instances migrate by a single operation, provided the multiple instances are mapped to a single CM_{id}. In the dynamic routing approach, the migration decision can be made by an actor who is involved with the instance[13]. This decision of migration is usually based on the outcome of the actor's action and the way the next tasks are delegated. Kradolfer et al. propose an approach that evaluates process instances and systematically handles instance migration in workflow based systems[42]. The authors provide taxonomy of schema modifications. The process instance migration is based on migration-conditions derived from execution invariants.

In order to realize *ease of human understanding and intervention* at the process instance level, it is required to represent the state of the instance and the data associated with it. One approach to address this is the approach of integrating business rule with BPEL as proposed by Graml et al.[28], which provides easy integration with the Websphere integration developer[34]. This allows users to understand the current status, associated data and thereby to manage the process instances. The FLOWer Case Guide is a client tool that allows each user to get access to cases. Here a case is a representation of a process instance. Case viewing capability allows a user to understand the current state of the process instance and possible actions to be executed.

4.3 Service relationship flexibility

The *ability to change service interfaces* is required to expose the changed internals of a composition to external entities. Manual change of interfaces is a difficult activity for a complex composite with frequent changing business processes. In the ROAD framework[43, 44], the services (players) are interacting with the composite by invoking the operations defined in its requirements description (roles). This interface gets changed as the contracts of the composite are changed. The behavior mismatches may too be avoided using adaptor composites. In the PAWS framework a mediator is used to transform the messages to manage the mismatches between given and required interfaces[45]. The mediator engine, which can be configured at runtime, is used to automate the mapping

The *ability to change service bindings* is supported by many approaches. The business rules can be used to dynamically bind the services as in [26] and [29]. Contextual worklet selection can be used to select the services during runtime[27]. But worklets that contains binding information need to be modeled before the actual binding take place. In data-driven approach[12] the actors of activity-roles can be changed at runtime. The event-driven approach [14] is also capable of changing the service bindings in a way such that, different bindings represented as actions. Based on the triggered event(s), a suitable action, thereby suitable binding, could be selected. Karastoyanova et al. use selection policies to change the service bindings[46]. The policies define the selection criteria based on QoS and semantics provided by the user. The authors introduce a <find_bind> construct to the existing WS-BPEL language, which can be used along with <invoke> to bind selection policies to activities. Robust-BPEL[47, 48] is a framework that provides autonomic handling of service invocation failures. The framework produces an adapt-ready[48] version of WS-BPEL scripts, which can handle partner service failures. Extending this work further, the authors propose TRAP/BPEL[49] that allows choices to be made over available services at runtime.

While much work has been done on how to represent and negotiate inter-service relationships, typically in the form of Service Level Agreements (e.g. WS-Agreement[50], Cremona[51]), less emphasis has been given to the ability to *change inter-service relationships* in a way that is consistent within a composition and does not adversely effects the processes running over that composition. In the ROAD framework[43] the terms of relationships between different entities ('contracts') can be changed or created at runtime to adapt the composition. These relationship contracts can express both business rules and technical obligations. The requirements descriptions ('roles') associated with these relationships are then dynamically aggregated and generated so that concrete services can be bound to those roles during the runtime. ROAD however does not provide a way to check that these dynamic relationship definitions are consistent with any processes that the composite defines. Likewise, the constraint driven approach by Pesic et al.[26] is capable of modeling the constraints of service relationships. However, while it provides the ability to add/remove or change the level (soft/mandatory) of those constraints it does not model or aggregate those relationships as abstract service descriptions as in ROAD.

Summary: Table 2 summarizes how various approaches support the flexibility requirements discussed above. Note that not all the approaches discussed above are included in the table, due to space limitations and some of them being only focusing on a very limited set of requirements.

V. DISCUSSION

From the analysis in the previous sections, we can make several observations on the attempts to address the process flexibility requirements.

First, the approaches for flexible business process modeling in service compositions can be categorized into two main groups. The first group of approaches consists of *extensions to WS-BPEL*. This is obviously due to the wide-spread use of the BPEL language for business process modeling. Some extensions [29, 48, 49, 52] require modifications to standard BPEL enactment engines (such as Active BPEL[53]), while others [28, 54] attempt to find solutions transparently to the enactment engine. The second group consists of *alternatives to WS-BPEL*[12-15, 26]. These alternatives describe the inherent limitations of WS-BPEL such as lack of modularity and runtime adaptability and show how they are capable of handling certain specific adaptability requirements.

Second, in alternative approaches (the second group mentioned above), the progress of the process instances is focused on some aspects other than the process flow. For example in the data-driven approaches the progress of process instances are modeled based on the values/states of the data objects or products. In event-driven approaches the next activity is triggered by the firing of a combination of events. In constraint driven approaches the progress of process instances is such that it does not violate the defined constraints. In dynamic routing, the definition gets changed by actors based on task delegation[13]. Table 1 shows the focal entities and gives sample approaches from our evaluation for each and every paradigm.

One common observed characteristic of these alternative approaches is that these alternatives avoid having a predefined sequence of activities. The idea is to improve the flexibility in the control flow to allow late modifications. However, there is an implicit flow of activities or a guide, even though it is not explicitly represented in the process model. As an example, in the event-driven approach the event-action relationships define this implicit flow. Consequently it is difficult to identify the flow of a process

instance at the instantiation time, which could be changed based on the changes of focal entities.

TABLE 1: ALTERNATIVE PARADIGMS TO WORKFLOW-DRIVEN APPROACHES

Paradigm	Focus	Example
Data-driven	Data-objects, Products, Business Artifacts	[12, 15]
Event-driven	Events	[14]
Constraint-based	Constraints	[26]
Dynamic routing	Task delegation	[13]

Third, there is no best approach *per se*, but rather a set of different approaches for different modeling purposes. The selection of a business process modeling approach should be based on the business requirements of the composition. For example, if the coordination of activities in RoSAS is more often than not dependent on the client/stakeholder data, then it is advisable to model the business processes of the composition using a data-driven approach. Interestingly, it is possible to combine multiple approaches from different paradigms. For example the main process description can be modeled using a workflow based approach for understandability, while a sub-process can be represented using a product driven approach for flexibility.

Fourth, the gap between handling of exceptions and expected changes is narrowing as dynamic runtime process changes become a norm. For example, the same mechanisms that are used to specify/change the constraints can be used to make a particular instance deviate because of an exception. In fact, it is conceivable that a flexible process system may not differentiate exceptions from normal behavior as exceptions are expected to be accommodated and dealt with. The system should always accept exceptions as part of the business operations and accommodate them in their change mechanisms.

Fifth, there is an inter-play between the three levels of process flexibility upon the occurrences of changes. For example a change in a process definition may require a change in the services relationships and vice versa. Such changes (at both of these levels) affect the process instances and trigger changes in them (e.g. cancellation, migration, deviation etc.). However, there are constraints that should not be violated in each of these levels. Thus the change mechanisms should take account of the constraints and dependencies so that a change at one level does not lead to a constraint violation at another level or the required change is accommodated appropriately at other levels.

Sixth, the flexibility of business processes depends on or is constrained by the way in which the service relationships are modeled. The ultimate target of the process flexibility is its survival over variable conditions. If the services relationship level has little flexibility, the business processes defined on them inherit those inflexibilities and may fail to survive. For example, if it is not possible to dynamically bind services, then a failure of a partner service would result in the failure of the whole business process instance. We observe that the support for services relationship flexibility in existing approaches is very limited, in particular in providing adaptable service interfaces and inter-service contractual relationships. This reflects the fact that services relationships are not considered "first-class" in most process modeling approaches, which limits the appropriate treatment of processes in both flexibility and stability.

Seventh, by analyzing existing approaches, we can see that different mechanisms and techniques are devised to achieve automated support for process flexibility. Consequently, the requirements for process flexibility support as discussed in section 3 pose technical requirements for a modeling and enactment approach to process flexibility in terms of realization mechanisms and techniques. Some of these technical requirements are management, monitoring, logging, and mining among others. While they are not direct requirements for process flexibility, the fulfillment of these technical requirements improves or enables the business process flexibility. For example, establishment of management communication channels between two compositions allow for a compromised solution upon an error. Monitoring and mining can be used to find past process instance deviations, and to regulate the processes in an autonomic manner.

VI. RESEARCH CHALLENGES

Based on the above observations we have identified a number of research challenges to achieve highly flexible business process modeling and enactment for service oriented systems.

First, service relationships, as exemplified in the section 3 above, need to be treated as first class entities in process modeling and be realized in a services composition in an adaptable manner. So far, limited attention is paid to representing service relationships in a clear, modularized and flexible way. Instead service relationships are represented as a monolithic block and usually intermingled with the orchestration/choreography logic. This is mainly due the unsatisfactory support for representing them in existing business process modeling and enactment technologies. Business processes enacted over such inflexibly modeled service relationships are difficult to maintain and may be has a less chance of surviving than those with flexibly modeled relationships.

Moreover, the fluctuating business and technical requirements make services relationships volatile. In a way the loosely coupled nature of the SOA tend to fuel this fragility as the entities are not tightly bound to the composition or the application. Also the ownership of the entities in the composition is distributed. Therefore the frequency of the changes in services relationships may not necessarily be synchronized with the changes in the business processes. As such, there is the requirement to automate the change to business processes based on the changes in the services relationships.

Also these changes in services relationships need to be interpreted with respect to each participating entity in order to alert them. Because, changes in service relationships such as the changes in mutual obligations, constraints, accepted performance levels and protocols of communication have different meanings to different entities. For example a client would be more interested than the rent car service about a change in the protocol to send diagnostic data of the breakdown The RoSAS operator would be interested about the operating hours of the newly bound Tow Car service, while a clerk/payment service need to know the payment method (e.g. Cash/Credit). Such changes could be specific to a particular process instance or could be common to all the running instances. Also those could be temporary or could be permanent. Thus the change interpretation and alert mechanism should be capable of correctly interpreting the change and communicate the interpreted change in timely manner. This would allow the participating entities too to determine the adaptation strategy if desirable.

Yet, the real challenge is not providing the business process flexibility or the automation, but achieving the business process flexibility in an automated setting in a controlled manner. The design and the realization mechanism should be able to preserve the business invariants during the changes.

TABLE 2: EVALUATION- HOW DIFFERENT BUSINESS PROCESS MODELLING AND ENACTMENT APPROACHES SUPPORT FLEXIBILITY REQUIREMENTS

Flexibility requirements	**Pesic et al. Constraint based[26]**	**van der Aalst et al. case handling[8, 12]**	**Adams et al. Worklets[27]**	**Alexopoulou et al. Event-driven[14]**	**Graml et al. Business rules integration. [28]**	**Barros et al. Dynamic routing[13]**	**Charfi et al. AO4BPEL[29]**
3.1.1 Configurable design	Constraints are configurable.	Case data model is configurable.	Rules and worklets are configurable.	E- A relationships are configurable	Configurability via Rules.	Uses configurable templates.	Configurability via Rules for predefined point cuts.
3.1.2 Inbuilt context awareness	-	Yes. Data objects to represent context.	Yes. The worklets selection is contextual.	-	Yes. BRs have access to BP instance context.	Yes. Actors have access to the case data.	-
3.1.3 Ease of understanding	Available with YAWL[55] integration[7].	Available with FLOWer studio[56].	Available with YAWL[55] editor.	-	Available with IBM Websphere[34] integration.	Possible. But parallel modifications may lead to confusion.	Available with the AO4BPEL editor[57].
3.1.4 Late specification	Possible. Constraints can be added at runtime.	Possible. Data objects can be modelled at runtime.	Possible. Worklet selection is at runtime.	Possible. Actions and events can be added, mapped at runtime.	Possible. But to which extent is depend on the pattern in use[28].	Possible. Actors specify/refine the templates at runtime.	Possible. But depend on the way point-cuts are specified.
3.1.5 Automated change verification	A formal verification[7] of LTL formulas is possible with Spin tool [36]	Available in FLOWer[56]	Possible. Rules are being used to verify the change.	-	-	-	-
3.1.6 Merging Definitions	Easy with Constraint Model [26] mapping.	-	Easy as worklets are self contained.	-	-	-	-
3.2.1 Process instance deviation	Possible. Deviations are allowed within specified constraints. %	Possible. Can alter completion conditions and type of the data objects. (mandatory/free) +	Possible. Worklets can be replaced to make an instance to deviate.	Possible. Event/Action relationships can be altered for a given process instance.	Possible. BRs can be altered based on BP instance context.	Possible. Actors can make instance deviate by modifying the received case constraints.	-
3.2.2 Process instance handling as a case	Possible. A single process instance can be mapped to a single CM.	Possible. The progress of the instance is determined by the state/value of the data objects.	Possible. Worklets selection and launching is done by the engine as a separate case.	Possible. The initial Id value is propagated to all subsequent events.	Possible. The progress of the instance is based on the BP instance context.	Possible. A case is routed among different actors until it is completed	-
3.2.3 Process instance migration	Possible. Via CM*id* mapping	-	-	-	-	Possible. Actor change definition.	-
3.2.4 Ease of human understanding and intervention if required	-	Possible. With FLOWer Case Guide [56]	-	-	Possible. With the graphical interface of Websphere [34]	-	-
3.3.1 Ability to change service interfaces	-	-	-	-	-	-	-
3.3.2 Ability to change service bindings	Possible Rules are used for services selection.	Possible. Actors of activity-roles can be changed at runtime	Possible. Via dynamic worklets that bind new services.	Possible. By defining new actions.	Possible. Autonomic choices made over available services.	Possible. Actors can change future participants.	Possible. BRs may be used to select participants at runtime
3.3.3 Ability to change inter-service relationships	-	-	-	-	-	-	-

-=Not being addressed specifically. CM=Constraint Model, BP=Business Process, BR=Business Rule.

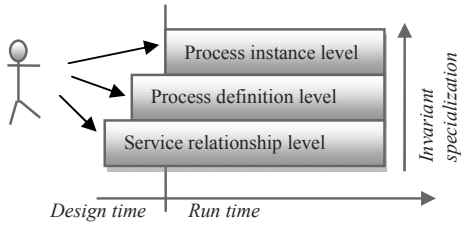

Figure 4: Invariants in all three levels of a services composition

All the three levels, shown in Figure 4 above, contain business invariants. The invariants at the instance level are defined during the execution time specifically for a particular process instance whilst the invariants in lower two levels could be defined at both design and execution time. As an example, a RoSAS process instance might be constrained by the communication protocol of the bound Garage, which is an instance level invariant and need to be defined at runtime when the Garage is bound to the process instance. Alternatively the composition may strictly define the communication protocol as an invariant in the process definition level, in which case the invariant is common to all the process instances. Usually the higher level invariants are a specialization of the invariants at lower levels. An invariant defined in the service relationships level would be specialized to define the coordination of the activities at the process definition level. Such coordination may be specialized by the runtime in order to prevent a particular process instance from going into an illegal state, due to an unforeseen runtime modification.

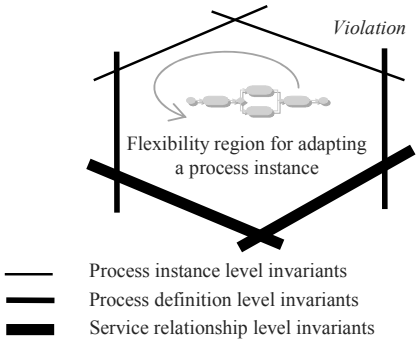

Figure 5: Boundaries of flexibility of a runtime process instance

Thus the boundary of flexibility of adapting a runtime *process instance* is defined by the invariants of all three levels as shown in Figure 5 above. Any change to a process instance, beyond the boundaries defined by the invariants will lead to a violation. Likewise the boundary of flexibility of adapting a *process definition* at runtime is defined by the invariants of the process definition level as well as of the service relationship level. Thus any change to a process definition should be done within the boundaries defined by the process definition level and service relationship level invariants.

In summary, while having a flexible business process modeling and enactment mechanism, the process model should be rich enough to clearly define the business invariants of the services composition. Based on this clear representation of business invariants, a decision making entity can formulate the strategies for business process adaptation either in an automated or semi-automated manner, so that the business invariants at all levels are preserved under the changing business requirements and operating conditions.

VII. Conclusion

In this paper we have systematically analyzed and identified the requirements for business process flexibility in service compositions. The requirements were categorized into those concerning process instances, process definitions and services relationships. We have analyzed and compared how existing process modeling and enactment approaches as to how they fulfill these requirements. Based on this analysis and evaluation, we have drawn a number of observations. One of the key observations is the lack of support to model service relationships, which describes the mutual obligations, constraints etc. in defining business processes of a service composition. Earlier, we discussed the importance of modeling business process definitions based on clearly and flexibly defined service relationships. Also the importance of steering the adaptation of business processes based on these relationships at runtime.

However, existing approaches for business process modeling and enactment neither treat these service relationships as first class entities nor model them in a flexible manner. Consequently, business processes defined over such an inflexible service relationships cannot withstand the inevitable change. Further the business processes defined over implicit service relationships have the potential of violating business invariants due to frequent runtime modifications. Finally, we pointed out the areas for future investigation in order to achieve full flexibility support for business process modeling and enactment in a service composition, in particular modeling and enacting the business processes over flexibly and clearly defined service relationships safeguarding the business invariants of the composition.

References:

[1] W. M. P. van der Aalst, A. H. M. ter Hofstede and M. Weske, "Business Process Management: A Survey," *Proceedings of the 1st International Conference on Business Process Management, LNCS*, vol. 2678, 2003, pp. 1-12.

[2] A. Barros and M. Dumas, "The Rise of Web Service Ecosystems," *IEEE Computer Society*, vol. 8, no. 5: September - October 2006, 2006, pp. 31-37.

[3] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski and A. P. Barros, "Workflow Patterns," *Distrib. Parallel Databases*, vol. 14, no. 1, 2003, pp. 5-51.

[4] M. Dumas, W. van der Aalst and A. H. ter Hofstede, *Process Aware Information Systems: Bridging People and Software Through Process Technology*, Wiley-Interscience, 2005.

[5] W. M. P. van der Aalst and T. Basten, "Inheritance of workflows: an approach to tackling problems related to change.," *Theor. Comput. Sci.,*, vol. 270(1-2), 2002.

[6] M. Adams, D. Edmond and A. H. M. ter Hofstede, "The Application of Activity Theory to Dynamic Workflow Adaptation Issues," *7th Pacific Asia Conference on Information Systems (PACIS), Adelaide, South Australia*, 2003, pp. 1836-1852.

[7] M. Pesic and W. M. P. van der Aalst, "A Declarative Approach for Flexible Business Processes Management," *Business Process Management Workshops*, 2006, pp. 169-180.

[8] W. M. P. van der Aalst and P. J. S. Berens, "Beyond Workflow Management: Product-Driven Case Handling," *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, vol. 2, 2001, pp. 42-51.

[9] M. Weiser, "The computer for the 21st century," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, no. 3, 1999, pp. 3-11.

[10] T. Andrews, et al., *Web Services Business Process Execution Language Version 2.0.* , January 2007.

[11] W. M. P. van der Aalst, et al., "Life After BPEL?," *Formal Techniques for Computer Systems and Business Processes*, 2005, pp. 35-50.

[12] W. M. P. van der Aalst, M. Weske and D. Gr¨unbauer, "Case Handling: A New Paradigm for Business Process Support," *Data and Knowledge Engineering*, vol. 53, 2005, pp. 129-162.

[13] A. Barros and G. Decker, "Dynamic Routing as paradigm for decentralized flexible process management," *Proceedings of the 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops*, IEEE Computer Society, 2006, pp. 27.
[14] N. Alexopoulou, M. Nikolaidou, Y. Chamodrakas and D. Martakos, "Enabling On-the-Fly Business Process Composition through an Event-Based Approach," *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, 2008, pp. 379-389.
[15] I. Vanderfeesten, H. Reijers and W. van der Aalst, "Product Based Workflow Support: Dynamic Workflow Execution," *Advanced Information Systems Engineering*, 2008, pp. 571-574.
[16] D. Müller, M. Reichert and J. Herbst, "Flexibility of Data-Driven Process Structures," *Business Process Management Workshops*, 2006, pp. 181-192.
[17] E. M. Bahsi, E. Ceyhan and T. Kosar, "Conditional workflow management: A survey and analysis," *Scientific Programming*, vol. 15, 2007, pp. 283-297.
[18] R. Hull, "Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges," *On the Move to Meaningful Internet Systems: OTM 2008*, 2008, pp. 1152-1163.
[19] R. Lu and S. Sadiq, "A Survey of Comparative Business Process Modeling Approaches," *Business Information Systems*, 2007, pp. 82-94.
[20] S. Nurcan, "A Survey on the Flexibility Requirements Related to Business Processes and Modeling Artifacts," *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, 2008, pp. 378-388.
[21] H. Schonenberg, R. Mans, N. Russell, N. Mulyar and W. M. P. van der Aalst, "Process Flexibility: A Survey of Contemporary Approaches," *Advances in Enterprise Engineering I*, 2008, pp. 16-30.
[22] C. Nemo, M. Blay-Fornarino, M. Riveill and G. Kniesel, "Semantic orchestration merging - towards composition of overlapping orchestrations," *International Conference on Enterprise Information Systems*, 2007, pp. 378-383.
[23] R. Fang, et al., "Dynamic Support for BPEL Process Instance Adaptation," *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 1*, IEEE Computer Society, 2008, pp. 327-334.
[24] S. Rinderle, M. Reichert and P. Dadam, "Correctness criteria for dynamic changes in workflow systems--a survey," *Data & Knowledge Engineering*, vol. 50, no. 1, 2004, pp. 9-34.
[25] B. Weber, S. Rinderle and M. Reichert, "Change Patterns and Change Support Features in Process-Aware Information Systems," *Advanced Information Systems Engineering*, 2007, pp. 574-588.
[26] M. Pesic, M. Schonenberg, N. Sidorova and W. van der Aalst, "Constraint-Based Workflow Models: Change Made Easy," *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, 2007, pp. 77-94.
[27] M. J. Adams, A. H. M. ter Hofstede, D. Edmond and W. M. P. van der Aalst, "Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows," *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, 2006, pp. 291-308.
[28] T. Graml, R. Bracht and M. Spies, "Patterns of business rules to enable agile business processes," *Enterprise Distributed Object Computing Conference*, vol. 2, no. 4, 2008, pp. 385-402.
[29] A. Charfi and M. Mezini, "Hybrid web service composition: business processes meet business rules," *Proceedings of the 2nd international conference on Service oriented computing*, ACM, 2004, pp. 30-38.
[30] W. van der Aalst and M. Pesic, "DecSerFlow: Towards a Truly Declarative Service Flow Language," *Web Services and Formal Methods*, 2006, pp. 1-23.
[31] M. Pesic, H. Schonenberg and W. M. P. van der Aalst, "DECLARE: Full Support for Loosely-Structured Processes," *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, 2007, pp. 287-287.
[32] T. Scheffer, "Algebraic Foundation and Improved Methods of Induction of Ripple Down Rules," *Procceedings of the Pacific Rim Workshop on Knowledge Acquisition, Sydney, Australia,*, 1996, pp. 23-25.
[33] M. Pesic, "DECLARE Tool. The Manual," *Book DECLARE Tool. The Manual*, Series DECLARE Tool. The Manual, ed., Editor ed.^eds., February 18, 2008, pp.
[34] "Websphere integration developer. IBM Software. http://www-306.ibm.com/software/integration/wid/," 2006.
[35] J. M. Couvreur, "On-the-fly Verification of Linear Temporal Logic," *FM'99 — Formal Methods*, 1999, pp. 711-711.
[36] G. J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston, Massachusetts, USA,*, 2003.
[37] M. Reichert and P. Dadam, "Adept_flex;Supporting Dynamic Changes of Workflows Without Losing Control," *J. Intell. Inf. Syst.*, vol. 10, no. 2, 1998, pp. 93-129.
[38] S. Sun, A. Kumar and J. Yen, "Merging workflows: A new perspective on connecting business processes," *Decision Support Systems*, vol. 42, no. 2, 2006, pp. 844-858.
[39] C. Nemo, M. Blay-Fornarino, M. Riveill and G. Kniesel, *Semantic orchestration merging - towards composition of overlapping orchestrations*, 2007.
[40] C. Nemo, T. Glatard, M. Blay-Fornarino and J. Montagnat, "Merging overlapping orchestrations: an application to the Bronze Standard medical application," *Services Computing, 2007. SCC 2007. IEEE International Conference on*, 2007, pp. 364-371.
[41] I. Vanderfeesten, H. A. Reijers and W. M. P. van der Aalst, "Case Handling Systems as Product Based Workflow Design Support," *Enterprise Information Systems*, 2009, pp. 187-198.
[42] M. Kradolfer and A. Geppert, "Dynamic Workflow Schema Evolution Based on Workflow Type Versioning and Workflow Migration," *Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems*, IEEE Computer Society, 1999.
[43] A. Colman and J. Han, "Using role-based coordination to achieve software adaptability," *Science of Computer Programming*, vol. 64, no. 2, 2007, pp. 223-245.
[44] A. Colman, "Role-Oriented Adpative Design," PhD Thesis. , Swinburne University of Technology, Melbourne, 2007.
[45] A. Danilo, C. Marco, M. Enrico, P. Barbara and P. Pierluigi, "PAWS: A Framework for Executing Adaptive Web-Service Processes," IEEE Computer Society Press, 2007 of Conference, pp. 39-46.
[46] D. Karastoyanova, A. Houspanossian, M. Cilia, F. Leymann and A. Buchmann, "Extending BPEL for run time adaptability," *EDOC Enterprise Computing Conference, 2005 Ninth IEEE International*, 2005, pp. 15-26.
[47] O. Ezenwoye and S. M. Sadjadi, "Enabling Robustness in Existing BPEL Processes," *In Proceedings of the 8th International Conference on Enterprise Information Systems*, 2006, pp. 95-102.
[48] O. Ezenwoye and S. M. Sadjadi, *RobustBPEL-2: Transparent autonomization in aggregate web services using dynamic proxies,* , Autonomic Comput. Res. Lab., Florida Int. Univ., Miami, FL;, 2006.
[49] O. Ezenwoye and S. M. Sadjadi, "TRAP/BPEL: A Framework for Dynamic Adaptation of Composite Services.," *In Proceedings of The International Conference on Web Information Systems and Technologies (WEBIST'2007), Barcelona, Spain, 2007*, 2007.
[50] Web Services Agreement Specification (WS-Agreement),A. Andrieux, et al.,citeulike-article-id:975488
https://forge.gridforum.org/projects/graap-wg/,
[51] H. Ludwig, A. Dan and R. Kearney, "Cremona: an architecture and library for creation and monitoring of WS-agreents," *Proceedings of the 2nd international conference on Service oriented computing*, ACM, 2004, pp. 65-74.
[52] Y. Wu and P. Doshi, "Making BPEL Flexible – Adapting in the Context of Coordination Constraints Using WS-BPEL," *IEEE International Conference on Services Computing*, vol. 1, 2008, pp. 423-430.
[53] "The ActiveBPEL Community Edition Engine . http://www.activevos.com/community-open-source.php."
[54] K. Geebelen, S. Michiels and W. Joosen, "Dynamic reconfiguration using template based web service composition," *Proceedings of the 3rd workshop on Middleware for service oriented computing*, ACM, 2008, pp. 49-54.
[55] W. M. P. van der Aalst, L. Aldred, M. Dumas and A. H. M. ter Hofstede, "Design and Implementation of the {YAWL} System," *Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE'04)*, 2004.
[56] "Pallas Athena : Case Handling with FLOWer: Beyond workflow ", Pallas Athena BV, Apeldoorn, The Netherlands., 2002.
[57] A. Charfi, "Aspect-Oriented Workow Languages: AO4BPEL and Applications - PhD Dissertation," Darmstadt University of Technology, Darmstadt, Germany, 2007.