Controlled flexibility in business processes defined for service compositions

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

***Abstract*— Business process modeling and enactment approaches for services compositions provide a way to coordinate the activities performed by de-centralized entities such as web services. As business needs change, the defined processes supporting the business also need to change and adapt, giving rise to the need for flexible business processes. However, a service composition is a collaborative environment where the *service providers, consumers* as well as the *aggregator* have business goals to achieve. Safeguarding such goals can be a daunting task upon numerous runtime modifications to business processes, but it is necessary to ensure the viability of the composition with respect to the business goals of all the parties. Therefore the flexibility needs to be controlled but without unnecessary restrictions. In this paper we propose a novel architectural approach to model, enact and manage business processes and their changes based on explicitly represented service relationships of a service composition.**

Keywords- SOA, BPM, Flexibility, Controllability

I. INTRODUCTION

Service based systems allow decentralized collaboration among business partners with distrusted, autonomous software systems. Such business collaboration is modeled as a service composition to provide a value added service [1]. Business process modeling (BPM) and enactment approaches for service compositions are used to coordinate the business activities of such collaboration in an automated manner [2]. This usually results in having multiple and possibly, inter-related business process definitions in large service compositions. In a highly dynamic business environment, such defined processes need to be changed [3-5], for various reasons from technological advancement to renewed business strategies.

While improving the process support, the current standards for business process modeling and enactment (e.g., WS-BPEL[6]) however lack the *flexibility* to adapt to the process and service behavior variability requirements, both foreseen and unforeseen[5]. This is mainly due to the rigid and imperative nature of the modeling and enactment [2, 7]. In order to overcome such limitations of existing imperative approaches, a number of improvements to them have been proposed, e.g. [8-12]. In addition, many declarative process modeling paradigms have been introduced as alternatives, allowing a greater degree of flexibility, e.g. [13-15]. On the other hand, this higher flexibility can also causes problems to the system's runtime stability[16]. This includes violations of business goals of both the service composition and its constituent services amidst numerous modifications. Furthermore, these approaches do not help to identify the impact of business process changes on the collaboration or the impact of changes in the collaboration on the goals of business processes. Therefore it is necessary to accommodate flexible process changes in a systematic and tractable manner, to preserve the business goals of both the composition and its collaborators.

With these objectives in mind, we propose an architectural approach to capture the business constraints, allowable interactions and mutual obligations between service providers of a service composition via an explicit representation of *service relationships*. Based on such explicitly defined service relationships, we are able to define and modify business processes in a flexible but controlled manner. Runtime change requirements of business processes are well supported while respecting the business constraints of all parties in collaboration.

In Section 2 below, we present a motivational example to highlight the need for flexible but controlled process change support. The Section 3 provides details about our approach and the modeling language. Section 4 discusses how the approach provides the adaptation capabilities in a controlled and tractable manner. The Section 5 provides implementation details. Other related work are presented and evaluated in Section 6 while the paper is concluded in Section 7.

II. BUSINESS EXAMPLE

RoSAS is a business organization that provides road side assistance services to its registered members. Its business model involves its Members(MM) (i.e., motorists) using the services provided by a number of collaborative service providers (e.g., Garage(GR) to repair cars, Tow-Car(TC) for towing, Case Officer(CO) for handling requests, and Paramedics(PM) for medical assistance) as shown in Figure 1.

	Plat	*Gold*	*Silver*
Repair	+	+	+
Towing	+	+	-
Paramedic	+	-	-

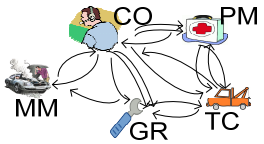

Figure 1. RoSAS business offerings and collaborators

RoSAS has defined different processes to provide different levels of services (Platinum, Gold, and Silver) to its members. These business processes share some common services while having differences between them. For example, activities related to car repairing are included

in all three business processes while the paramedic service is only available to platinum members.

Inevitably, these three business processes are subjected to changes at runtime. The changes may vary from process instance level changes (e.g., handling a special requirement of a member) to more evolutionary/definition-level changes (e.g., change in payment protocol for all garages)[5]. Also, in a collaborative environment, the change requirements can be originated from the collaborators, e.g., the service-service interaction protocol between TC and GR may change.

However, such changes should be carried out in a well-controlled and tractable manner. In this sense, the goals of the aggregator (RoSAS) as well as collaborators (underlying business services) need to be protected. For example, RoSAS may need to guarantee that any *platinum* process instance be closed within certain assured time boundary. On the other hand there are limitations of collaborators such as speed of execution, e.g., the time to process a request by the CO, time to acknowledge towing by TC. Therefore the modification to interactions needed to be controlled by the goals/constraints *of the aggregator* and goals/constraints *of the collaborators.* It follows that the *boundary for a safe runtime modification* is defined by the goals of the aggregator and the collaborators as shown in Figure 2. Interestingly, these goals also are subjected to adjustments over time. In this sense the boundary too is subjected to change, to make it either more relaxed or stringent.

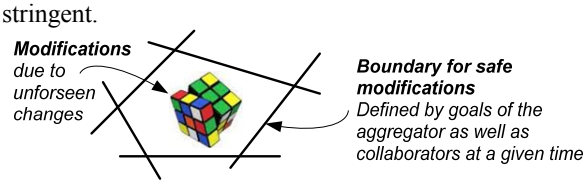

Figure 2: Boundary for safe modifications – An Illustration

The imperative approaches such as BPEL[6] fail to accommodate frequent modifications due to rigidity in activity sequencing, making the process support system a burden. On the other hand seamlessly flexible approaches without a proper control can harm the system integrity and thereby the confidence level of the designer who modifies the composition. This is very pertinent for service oriented business compositions because, *first*, there can be multiple but interrelated business processes defined in service compositions. *Second*, there may be many aggregated B2B (business to business) collaborations existing in a composition. If the business process modeling approach that is employed in service compositions fails to capture (a) *the service-service collaborations* (b) *the dependencies of business processes towards these collaborations*, adapting such a composition during runtime can be a tedious an error-prone task. Therefore RoSAS as a service aggregator is looking for following key requirements for a process modeling and enactment approach.

Req1. Flexible processes. As business changes, the business processes and their support need to be changed, including changes to individual process instances (*momentary/ad-hoc*) or to a process definition (*evolutionary*)[5]. The changes may be due to the collaborators or the aggregator's needs.

Req2. Controlled changes. Amidst numerous changes to business processes and their instances, (i) the business goals of *collaborators* as well as (ii) the business goals of the *composition* (RoSAS) represented by business processes need to be safe-guarded.

Req3. Predictable change impacts. The impact of a change (i) on business processes as well as (ii) on collaborators needs to be easily identified, so that remedial actions or subsequent changes can be taken if the change is to be realized.

III. APPROACH

A Service Composition is consisting of individual services collaborating with each other by exchanging messages and performing activities. Therefore it is necessary that a process modeling approach for service composition, capable of explicitly representing these service-service collaborations. Further, the aggregator defines and utilizes these service-service collaborations to achieve its goals. As an example RoSAS defines and utilizes the collaboration between CO and GR in all its business processes. Therefore it is necessary that the *dependencies of business processes on collaborations* are clearly represented. Further, the ordering, scheduling of activities need to be arranged and adapted at runtime, such that the goals of the aggregator as well as the collaborators are preserved.

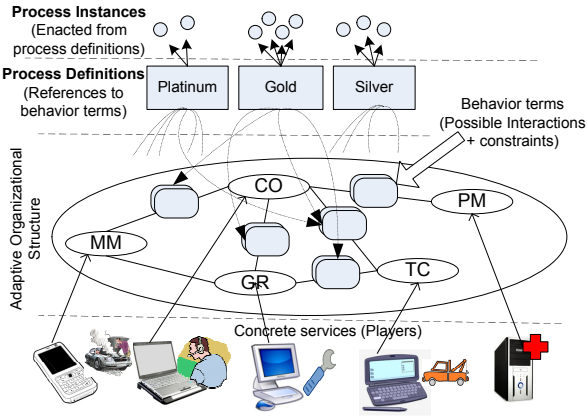

Figure 3: Business Processes based on an Adaptive Organizational Structure

Therefore, we model the service composition as an ***organization*** where the collaborations among partner services are explicitly represented. Partner services are represented as ***roles*** (CO, TC, MM, …) of the organization and collaborations are called ***service-relationships*** ($SR_{CO\text{-}TC}$, $SR_{GR\text{-}TC}$, …). As shown in Figure 3, we define the ***business processes*** (PD_{silver}, PD_{gold}, PD_{plat}) on top of these service relationships to represent the goals of the composition, e.g., how to serve *Gold* customers is represented by PD_{gold}. At runtime, ***process instances*** are enacted according to these definitions and executed on top

of the organizational structure. The ***players*** i.e., runtime entities such as *concrete web services* hosted in Garages and Tow Cars can play these roles during the runtime, contributing to the progression of enacted process instances.

The Role Oriented Adaptive Design (ROAD)[17] approach to adaptive software systems has such an organizational structure as the basis for service composition. However, it does not yet have support for defining business processes. In this research, we adopt ROAD and further extend it to realize the process support.

A. Capturing collaborations

During the runtime, partner services interact with each other. These partner services have certain obligations towards each other. As an example, "*CO has to request towing and TC has to acknowledge the request. Then TC has to perform the towing. Later, CO has to pay the TC"*. Also, partner services have *constraints on these interactions* that should not be violated upon modifications, e.g., "*the towing should eventually be followed by a payment"*. In our approach, a service relationship captures both the interaction and constraint aspects via one or more ***behavior terms***.

A behavior term groups relevant interactions and constraints together so that it is readily available to be referenced by one or more business process definitions as shown in Figure 3. As an example, both *Gold* and *Platinum* business processes need to provide towing in their business offering (Figure 1). Behavior Term CO-TC_b1 captures related interactions and constraints for towing. Therefore both PD_{gold} and PD_{Plat} can refer to the behavior term CO-TC_b1. These references represent the dependencies of business processes towards the collaborations in the service composition. Also the ability of having multiple behavior terms in a service relationship provides the required granularity to use only the relevant behavior terms in a business process.

The advantages of such architecture for a service composition are threefold.

1. The collaboration aspects, i.e., possible interactions and constraints among collaborating services are explicitly represented.
2. As the behavior terms (i.e. business interactions and constraints) evolve, those changes are automatically reflected in business processes.
3. The dependencies between business processes and collaborations are well-represented. This provides the basis to analyze the impact of a proposed change of business processes upon collaborations and vice versa.

In order to support above concepts we introduce the Serendip process modeling language. Given below are elements of Serendip language[1]. Here an *asterisk* (*) denotes multiplicity of zero or more, *plus* (+) denotes multiplicity of one or more, and *question mark* (?) denotes an optional element. Other elements are mandatory. The attributes are shown within parentheses.

```
Organization
   Role(id, binding)+
   ServiceRelationship (id, extends?)+
        RoleA
        RoleB
        BehaviorTerm (id)+
              Task(id)+
              Constraint(id) *
   ProcessDefinition(id)*
        BehaviorTermRef+
        Constraint(id) *
        CoT
```

Figure 4: Serendip language syntax tree

According to the above syntax, let us design the RoSAS composition as shown in Figure 5. Note that process definitions refer to the behavior terms of existing Service Relationships. For clarity we have shown only a partial description. Also, the *BehaviorTerms* and *ProcessDefinitions* will be described in subsequent sections with more details.

```
Organization RoSAS{
  Role CO{...}, TC{...}, GR{...}, MM{...}, PM{...};
  ServiceRelationship CO-TC{
    RoleA CO;
    RoleB TC;
    BehaviorTerm CO-TC_b1{...};
    BehaviorTerm CO-TC_b2{...};
  };
  ServiceRelationship GR-TC{
    RoleA GR;
    RoleB TC;
    BehaviorTerm GR-TC_b1{...};
    BehaviorTerm GR-TC_b2{...};
  };
  …
  ProcessDefinition PDgold{
   BehaviorTermRef CO-TC_b1, GR-TC_b1...;
   …
  };
   …
};
```

Figure 5: RoSAS organization

B. Modeling Behavior

Behavior term represents its interactions using ***tasks*** that should be performed by either one of the roles in collaboration. Naturally, tasks have dependencies with other tasks. Rather than directly linking tasks, we use ***events*** to show such task-dependencies in a declarative manner so that there is a loose coupling among tasks improving the flexibility (*Req1*).

Pre-conditions and Post-conditions of tasks are specified using event patterns, $EP_{pre,}$ and EP_{post} respectively. A task becomes doable when the EP_{pre} is evaluated *True*. Once a task is executed more events get triggered as per the EP_{post}. These events can become part of EP_{pre} of multiple other tasks enabling further execution. Using of events to define dependencies provides a loose-coupling between the tasks improving the ability to specify and adapt dependencies in a comprehensive manner.

Also tasks have obligated roles (R_{oblig}) and performance properties (P). As an example the task

[1] The prototype uses XML to describe organizations, service relationships, processes etc. But due to space limitations here we use an abbreviated syntax.

SendTowReq of behavior term CO-TC_b1 specifies "*Upon Towing is required the CO is obliged to request towing within 2 hours"*. Here *CO* is obligated Role (R_{oblig}) and *1 hour* is the performance property (P). For the clarity of presentation we will use time as the performance property but other properties such as *financial cost of executing tasks* can also be used.

The behavior term CO-TC_b1 of service relationship CO-TC (Figure 5) is described in Figure 6 below with explanation to follow.

```
BehaviorTerm CO-TC_b1{
   Task SendTowReq{
      EPpre TowReqd;
      EPpost TowReqSent & PickupLocKnown;
      PerfProp 2h;
      Roblig CO;
   };
   Task Tow{
      EPpre PickupLocKnown & DestinationKnown;
      EPpost CarTowed & TowAcked ;
      PerfProp 24h;
      Roblig TC;
   };
   Task PayTow{
      EPpre CarTowed & TowAcked;
      EPpost TCPaid;
      PerfProp 2h;
      Roblig CO;
   };
   Constraint (TowReqSent>0) ->[4h,24h](CarTowed>0);
   Constraint (CarTowed>0) ->(TCPaid>0);
};
```

Figure 6: Behavior Term CO-TC_b1

Explanation: First task definition (*SendTowReq*) states that when *TowReqd* event (=EP_{pre}) is fired, the player of the CO role (=R_{oblig}) is obliged to execute the task *SendTowRequest*. Here the expected performance (=P) is defined with time units, e.g. 2 hours. Once the task is completed events get triggered according to the post-event pattern, *TowReqSent & PickupLocKnown* (=EP_{post}). The two constraints are specified in CTL*[18]. If time properties need to be shown, we use the TCTL variant. The first constraint specifies *"if event TowReqSent triggered it should eventually be followed by CarTowed within 4 to 24 hours"*. The second constraint specifies *"if event CarTowed triggered should eventually be followed by TCPaid event"*.

C. Defining business processes

Business processes refer to the defined behavior terms (Figure 5). In order to make sure that the modifications in behavior terms do not contravene the goal of the business process, an additional set of constraints (CS_P) is defined in the process definition. Furthermore, a process definition also specifies a set of Conditions of Termination (CoT) which is again an event pattern similar to $EP_{pre/post}$. When CoT is evaluated to be *true*, a process instance is considered completed.

Let us consider the Gold process definition (PD_{gold}) which provides both the car repairing and towing services. Let's assume that PD_{gold} groups behavior terms CO-MM_b1, CO-GR_b1, CO-TC_b1, MM-GR_b1 and GR-TC_b1. An example for an additional process level constraint (CS_p) would be, *"A case should be closed within 5 days upon a complaint is received"*. The CoT could be satisfied upon event pattern (*CaseClosed OR CaseAborted)* is triggered. Following is the description of the PD_{gold}.

```
ProcessDefinition PDgold{
 BehaviorTermRef CO-TC_b1, GR-TC_b1...;
 Constraint (ComplaintRcvd>0) ->[0,5d](CaseClosed>0);...
 CoT CaseClosed | CaseAborted;
};
```

Figure 7: Process definition PDgold

During runtime the business process instances are enacted based on these definitions. As an example, the n^{th} instance of PD_{gold} will serve the n^{th} case/complaint of a gold customer. Each process instance has "its own" model or a copy of referenced behavior terms. This means, the modifications of interactions and constraints can be carried on a particular process instance too without affecting the other enacted instances or the original definition.

D. Meta-model

Summarizing the above introduced concepts we present our meta-model in Figure 8.

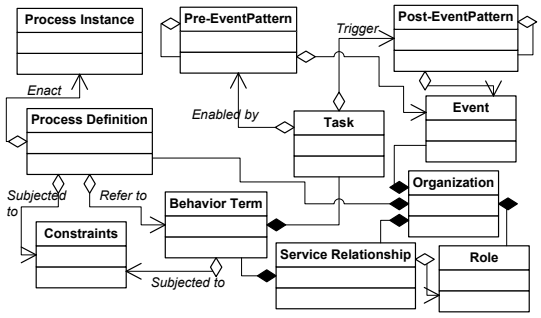

Figure 8: Serendip Meta-model

IV. CONTROLLED FLEXIBILITY

In this section we will first analyze the "change" in the organization. Then, we will see how the controlled flexibility is achieved giving examples.

A. Change dimensions in the organizational structure

A change in business requirement could trigger one or more changes in the adaptive organizational structure, such as adding/removing tasks, changing pre-/post-conditions, tuning performance properties etc (*Req 1*). Within the organizational structure a change can be classified along three dimensions as shown in Figure 9.

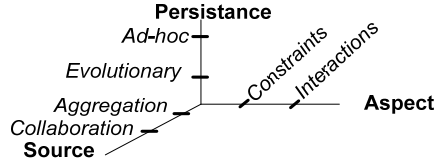

Figure 9: Change dimensions in the organization

Aspect: A change may occur in the defined *interactions* or the *constraints* of a given behavior term.

As an example a modification in CO-TC_b1 term could change the pre-/post-conditions of towing. Also as an example the first constraint of CO-TC_b1 given in Figure 6, can be modified to make it more relaxed, i.e., (TowReqSent>0) ->[4h,30h](CarTowed>0).

Source: A change could be proposed to modify the organizational structure due to a business level change in *collaborations*. As an example the collaboration between the TC and CO could be changed as new TC players might demand advance payments. Also, in order reflect the new requirements of the *aggregator*, i.e. RoSAS, the organizational structure could be changed, e.g., to speed up the *platinum* process might require tuning the performance property of task *Tow()*. Such changes are usually realized from the Process point of view.

Persistence: A change could be carried out in an evolutionary manner (on definitions) or in an ad-hoc manner (on a specific process instance) [7]. As an example the process instance $PD_{platinum}.031$ (i.e., 31st instance of process definition *Platinum*) could be modified to add an additional pre-event to the pre-conditions of task *tow()* to hold *towing* until Paramedics (PM) arrive at the scene. If such requirements are not common or unpredictable, usually those are not reflected on definitions and need to be implemented at process instance level.

B. Controllability and impact analysis

The organization structure aids applying changes in a controlled manner (*Req 2*). Also, it provides a platform to analyze the impact of a change (*Req 3*) on the aggregator as well as the collaborating services, prior to its realization. Such capabilities are vital for the viability of the business composition as pointed out in Section 2.

Let us analyze few example changes.

Change X (interaction, collab, ad-hoc): Behavior term CO-TC_b1 of process instance $PD_{gold}.021$ needs to be modified to provide more time for towing after the bound TC and CO players agree to this change.

Change Y (interaction, collab, evol): Collaborations among GR and TC need to be changed. TC players generally need to be paid an incentive by GR players. ConsequentlyThe task *PayIncentive()* need to be added to GR-TC_ b1 of service relationship GR-TC.

Change Z (constraint, aggreg, evol): RoSAS as a marketing campaign to attract more *gold* customers, require reducing the time to repair. So the process level constraint of PD_{gold} (Figure 7) is further restricted to the following. **(**ComplaintRcvd >0) ->[0,5d](CaseClosed >0)

Such changes are validated on the organizational structure to analyze the impact. Depending on the type of change, the impact on the organizational structure could be different. As an example, ***Change X*** has less impact compared to ***Change Y***. Because, the ***Change X*** is isolated to the process instance PD_Gold.021, whereas ***Change Y*** could affect all the processes definitions and (hence) instances that refers to GR-TC_b1. As the impact is low, the changes similar to ***Change X*** are less restrictive than ***Change Y***. So, it is necessary to identify the exact impact or the possibility of these changes.

In order to identify the impact, it is important to identify the *applicable set of constraints (CS)* for validation. During the runtime, both the interactions and constraints tend to evolve. Therefore for a given instance of time, we validate the current interactions against identified *CS*. If the validation produces a failure, the change is rejected sighting possible reasons for the rejection. If the validation is successful, change is realized on the organizational structure. A naïve approach to do this is to validate all the interactions against all the defined constraints in the composition upon a modification. However, such approach can be *computationally costly* as well as *unnecessarily restrictive*.

Instead we use the inter-dependencies between the processes and the service relationships that are well-represented via the organization, to determine the exact set of constraints that need to be validated. Let us take Change X and Change Y given above for a further clarification.

Change X, is carried out on process instance $PD_{gold}.021$ as shown in Figure 10 (a). In that case the constraints defined in the process definition PD_{gold} (CS_{gold}), Behavior Term CO-TC_b1 ($CS_{CO\text{-}TC\_b1}$) as well as new constraints (if any) introduced at/after the enactment on the process instance 021, (CS_{021}) need to be considered.

Therefore the applicable constraint set $=CS_{CO\text{-}TC\_b1} + CS_{gold} + CS_{021}$

Change Y is carried out on GR-TC_b1 of $SR_{GR\text{-}TC}$ as shown in Figure 10 (b). Let us assume that both PD_{gold} and $PD_{platinum}$ refers to GR-TC_b1 as both provides car towing (Section II). Therefore the constraints defined in the GR-TC_b1, ($CS_{GR\text{-}TC\_b1}$) as well as constraints defined in both definitions (CS_{gold}, $CS_{platinum}$) need to be considered.

Therefore the applicable constraint set = $CS_{GR\text{-}TC\_b1} + CS_{gold} + CS_{platinum}$

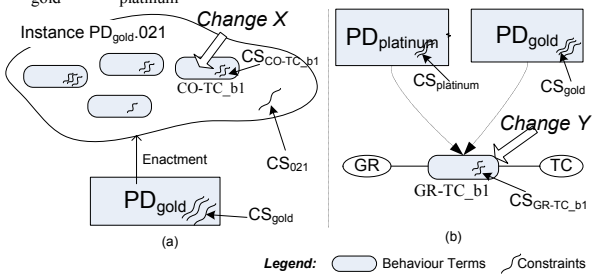

Figure 10: (a) Change X (b) Change Y

Therefore in general, if the ad-hoc change is in a Behavior Term **B** of a process instance **pi** of definition **PD**, the applicable constraint set is,

$=CS_B+CS_{pi}+CS_{PD}$

If the evolutionary change is in a Behavior term **B**, which is shared by *n* number of process definitions $\mathbf{PD_i}$(i=*1,2,3...n*), the applicable constraint set is,

$$=CS_B+\sum_{i=1}^{n} CS_{PDi}$$

As we see in the above examples, only the *applicable* constraint set is considered but nothing else. The same is valid when there is a change in defined constraints too. As an example in ***Change Z***, one of the constraints of PD_{gold}

is getting more restrictive. In this sense, the boundary for safe modifications gets shrunk (Figure 11). However, such restriction might or might not be possible with the currently defined interactions in service relationships. If the validation reveals it is possible to restrict, the RoSAS can go ahead with the ***Change Z*** and thereby with the marketing campaign. Otherwise either the change needs to be abandoned or facilitated by tweaking the relevant interactions defined in service relationships.

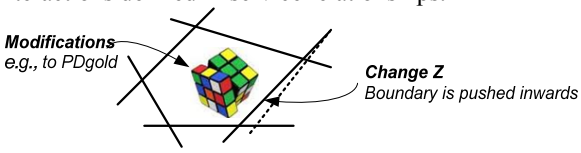

Figure 11: Restricting constraints – An Illustration

V. IMPLEMENTATION

In this section we will discuss how the Serendip framework is realized to model and enact business processes.

A. *Architecture*

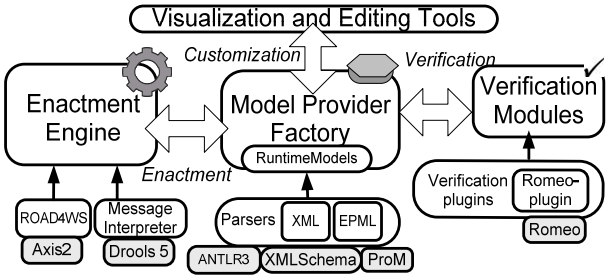

Figure 12: Serendip framework architecture

The high-level architecture of Serendip framework is given in the Figure 12. Core to the framework is the ***model provider factory*** (MPF) which creates runtime models based on behavior terms defined in parsed descriptions. These runtime models that are specific for enacted process instances interact with each other by triggering and listening to events progressing process instances (customer cases). The runtime models can be modified (e.g. a modification in CO-TC_b1 instantiated for PD_Gold.021) according to business needs. Such modifications are validated by the ***validation module***. The validation module can be further extended to add more domain specific *validator-plugins* if required. The ***enactment engine*** enacts business processes and maintains process instances. The messages are exchanged with external web services using the road4ws extension[19] developed for Axis2 web services engine[20]. The exchanged messages are interpreted as per the rules specified in the service relationships using Drools[21] to trigger events as per the EP_{post}. Both messages and events get associated with relavent process instances by the engine. The ***visualization and editing tools*** provide support to modify the runtime models in MPF and thereby adapt the interactions and constraints both in evolutionary and ad-hoc manner.

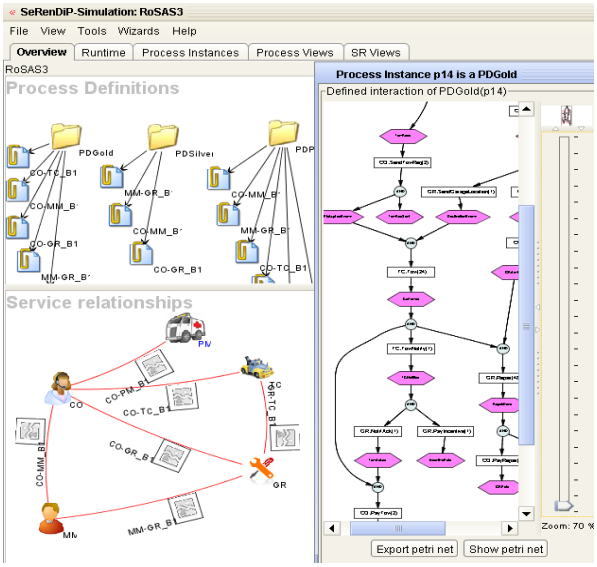

Figure 13: Serendip GUI

B. *Adaptation API*

Serendip provides a low-level adaptation API that can be used to adapt the runtime models as shown in the Figure 14. These operations are used by the editing tools to adapt the process definitions and service relationships to face unforeseen changes. Usually, these operations are used as a *batch command*, which is an ordered list of such lower level operations. Such batch commands get executed by higher-level graphical visualization tools using pre-defined wizards or simply via a command line interface by the expert users.

Task.	**Process Definition/Instance.**
Set_PreEP(String ep)	Add_BehaviorRef(String id)
Append_PreEP(String part)	Remove_BehaviorRef(String id)
Set_PostEP(String ep)	Add_Constriant(Constraint c)
Append_PostEP(String part)	Remove_Constraint(String id)
Set_Obligation(Role r)	SetCoT(CoT cot)
Set_PerfProp(Property p, value)	**Composition.**
Behavior Term.	Add_Event(Event e)
Add_Task(Task t)	Remove_Event(String id)
Remove_Task(String id)	Add_SR(SR sr)
Add_Constriant(Constraint c)	Remove_SR(String id)
Remove_Constraint(String id)	Add_PD(PD pd)
Add_Interpreter(String ruleId)	Remove_PD(String id)
Remove_Interpreter(String ruleId)	Add_Role(Role r)
Service Relationship.	Remove_Role(String id)
Add_Behavior(BehaiorTerm b)	Bind_Player(Role r, Player p)
Remove_Behavior(String id)	Remove_Player(Role r, Player p)

Figure 14: Serendip Adaptation API

C. *Validation via TPN and TCTL*

To validate a modification in a business process definition or of a process instance, we generate an EPC graph[22] by mapping the events defined in pre- and post-conditions. The generated EPC is then dynamically translated into a Time-Petrinet (TPN)[23] according to the translation rules introduced by van der Aalst et al.[22]. A TPN is a directed bipartite graph that can specify firing intervals of transitions. Places, transitions, firing intervals

in a translated TPN are analogous to the events, tasks and estimated time for completing tasks as shown in Figure 15.

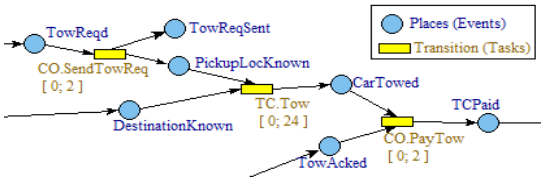

Figure 15: TPN Generated from CO-TC_b1

We use CTL*[18] with time properties(TCTL) to specify the constraints. It is shown[23] that TCTL properties can be used to validate TPN. After mapping the elements (i.e. events, tasks, performance properties) into the places, transitions and firing intervals of generated TPN, we validate whether the generated petri-net conforms to the *applicable set of constraints* (Section IV) as shown in Figure 16. For this, we implemented a wrapper for Romeo on-the-fly model checker[24] The wrapper dynamically generates the business constraints in TPN-TCTL format[23] and the models (behavior terms/business processes) as TPN to perform the validation via the Romeo plugin. The validation framework can be further extended to support other languages and model checking techniques if required.

Such validation is applicable for all types of changes mentioned in Section IV. If the validation is successful the change is permitted. Otherwise, the change is rejected by showing the violated collaboration constraints or aggregator goals, so that the designer can take further actions.

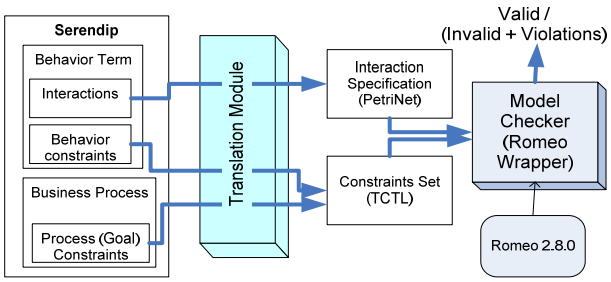

Figure 16: Validation of modifications

VI. Related Work

Early approaches to activity coordination of composed web services was done using programming languages such as *Java*, *C*, by writing code that glued together the invocations of service endpoints[25]. This was later replaced by more specialized workflow languages such as WS-BPEL[6]. Nonetheless these workflow based languages also inherit the imperative nature of the underlying programming paradigm, unnecessarily restricting the flexibility in process modifications[2]. Therefore many improvements [8-12] as well as alternative paradigms [13-15] to such a workflow approach have been proposed to overcome these limitations. The techniques include dynamically binding service endpoints[10], defining dynamic variation points[9], worklet selection[26], case driven[13] or event driven[14] enactment etc. Due to a large number of approaches proposed to improve process flexibility, we do not provide a complete overview of related work here but refer the reader to [3-4, 27].

While these approaches increase the flexibility of process modifications, little attention has been paid to controlling the flexibility so that both the goals of the composition as well as the collaborators are protected. Condec [7, 15] is a declarative constraint driven approach that provide a greater flexibility and control. However, the approach fails to specify the possible interactions making it difficult for the designer to visualize them. Such a visual aid is important for design and diagnosis stages of BPM[2]. Also Aspect oriented [25, 28] and policy based adaptation[11] can be seen as attempts towards a controlled flexibility in business processes. Likewise $ADEPT_{flex}$[29] allows handling ad-hoc changes in Workflow instances in a controlled manner. However none of those approaches support an explicit and well-modularized representation of service-service collaborations and their dependencies towards business processes. As such, the impact of a process change on collaborator relationships and *vice versa* cannot be easily understood. In collaborative environments, such as in service compositions, such a lack of representation may result in unpredictable consequences upon runtime modifications by violating the business objectives. Also such a lack of representation of dependencies might unnecessarily obstruct even the possible modifications.

The PartnerLinks[6] construct in WS-BPEL is a way to model a relationship between the composition (BPEL script) and the collaborators(e.g. RoSAS and a third party service provider). However, it cannot be used to represent the end-to-end relationships between collaborators (e.g., between GR and TC). On the other hand WS-CDL[30], defines the observable behavior of multiple service interactions. However WS-CDL is not declarative contributing to rigidity[31] and also it requires a mapping to another executable language such as BPEL[32] for execution. In contrast Serendip provides a language not only to model declarative service interactions, but also to enact the business processes. The Swimlane concept in BPMN[33] used to represent roles/participants of an organization. In comparison a Service Relationship in Serendip provides not only the representation of service-service interaction, but also act as an adaptable entity with a built in controllability, which could be safely modified to suit the runtime change requirements.

Moreover, in Aspect-Oriented and Template-based approaches [9, 12, 28] there is a *fixed* and a *volatile* part. As an example, *point-cuts defined business process* is the fixed part while *aspects/parameter* represents the volatile part. But in Serendip there is no such assumption. Any characteristic of the organization can be changed during the runtime, provided the modification leads the organization to a safe state. The safe state is determined by an automated validation process which considers the dependencies of business processes towards collaborations

and vice versa, clearly represented by the language. This provides a greater flexibility in modifying the composition without losing the control.

VII. Conclusion

In this paper, we have introduced a novel business process modeling and enactment approach based on explicitly defined service relationships. We showed how such an explicit representation is useful in realizing changes in a flexible but in a well-controlled manner. Also, the clear representation of the dependencies between processes and the collaborations allows a precise impact analysis before realizing the change. We have used a business example to explain the concepts. A prototype implementation of the Serendip framework is also introduced. As future work, the prototype will be further extended to provide a better user experience in modifying runtime models and also to support more constraint specification languages.

References

[1] 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,

[2] 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,

[3] S. Nurcan, "A Survey on the Flexibility Requirements Related to Business Processes and Modeling Artifacts," in Proceedings of the 41st Annual Hawaii International Conference on System Sciences, Year, pp. 378-388.

[4] H. Schonenberg, R. Mans, N. Russell et al., "Process Flexibility: A Survey of Contemporary Approaches," Advances in Enterprise Engineering I, 2008. pp. 16-30,

[5] W. M. P. van der Aalst, and S. Jablonski, "Dealing with workflow change: identification of issues and solutions," International Journal of Computer Systems Science and Engineering, vol. 15, no. 5, 2000. pp. 267-276,

[6] "BEA, IBM, Microsoft, SAP, AG, Siebel Systems, Business Process Execution Language for Web Services V1.1 specification ", 2003.

[7] W. M. P. van der Aalst, M. Pesic, and H. Schonenberg, "Declarative workflows: Balancing between flexibility and support," Computer Science - Research and Development, vol. 23, no. 2, 2009. pp. 99-113,

[8] D. Karastoyanova, A. Houspanossian, M. Cilia et al., "Extending BPEL for run time adaptability," in EDOC Enterprise Computing Conference, 2005 Ninth IEEE International, Year, pp. 15-26.

[9] K. Michiel, S. Chang-ai, S. Marco et al., "VxBPEL: Supporting variability for Web services in BPEL," Information and Software Technology, vol. 51, no. 2, 2009. pp. 258-269,

[10] O. Ezenwoye, and S. M. Sadjadi, "TRAP/BPEL: A Framework for Dynamic Adaptation of Composite Services.," WEBIST'2007, Barcelona, Spain, 2007, 2007,

[11] A. Erradi, P. Maheshwari, and V. Tosic, "Policy-Driven Middleware for Manageable and Adaptive Web Services Compositions," International Journal of Business Process Integration and Management vol. 2, no. 3, 2007. pp. 187-202,

[12] K. Geebelen, S. Michiels, and W. Joosen, "Dynamic reconfiguration using template based web service composition," in Proceedings of the 3rd workshop on Middleware for service oriented computing, Leuven, Belgium, Year, pp. 49-54.

[13] 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,

[14] N. Alexopoulou, M. Nikolaidou, Y. Chamodrakas et al., "Enabling On-the-Fly Business Process Composition through an Event-Based Approach," in Hawaii International Conference on System Sciences, Year, pp. 379-389.

[15] M. Pesic, M. Schonenberg, N. Sidorova et al., "Constraint-Based Workflow Models: Change Made Easy," in On the Move to Meaningful Internet Systems, Year, pp. 77-94.

[16] G. Regev, I. Bider, and A. Wegmann, "Defining business process flexibility with the help of invariants," Software Process: Improvement and Practice, vol. 12, no. 1, 2007. pp. 65-79,

[17] A. Colman, and J. Han, "Roles, players and adaptable organizations," Applied Ontology, vol. 2, no. 2, 2007. pp. 105-126,

[18] C. Baier, and J.-P. Katoen, Principles of Model Checking, 2008. The MIT Press

[19] "ROAD4WS : A realization of ROAD architecture using Web Services http://www.swinburne.edu.au /ict/research/cs3/road/road4ws.htm," CS3, Swinburne Universtiy of Technology.

[20] S. Perera, C. Herath, J. Ekanayake et al., "Axis2, Middleware for Next Generation Web Services," in Proceedings of the IEEE International Conference on Web Services, 2006, pp. 833-840.

[21] JBoss. "Drools : Business Logic integration Platform," http://www.jboss.org/drools/.

[22] W. M. P. van der Aalst, "Formalization and verification of event-driven process chains," Information and Software Technology, vol. 41, no. 10, 1999. pp. 639-650,

[23] H. Boucheneb, and R. Hadjidj, "CTL* model checking for time Petri nets," Theoretical Computer Science, vol. 353, no. 1, 2006. pp. 208-227,

[24] G. Gardey, D. Lime, M. Magnin et al., "Romeo: A Tool for Analyzing Time Petri Nets," Computer Aided Verification, 2005. pp. 418-423,

[25] M. Braem, K. Verlaenen, N. Joncheere et al., "Isolating Process-Level Concerns Using Padus," Business Process Management, 2006. pp. 113-128,

[26] M. J. Adams, A. H. M. ter Hofstede, D. Edmond et al., "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,

[27] M. Kapuruge, J. Han, and A. Colman, "Support for business process flexibility in service compositions: An evaluative survey," in Australian Software Engineering Conference- ASWEC. IEEE Computer Society. , Auckland, NZ, 2010, pp. 97--106.

[28] A. Charfi, and M. Mezini, "Hybrid web service composition: business processes meet business rules," in International conference on Service oriented computing, New York, NY, USA, 2004, pp. 30-38.

[29] M. Reichert, and P. Dadam, "ADEPT flex -Supporting Dynamic Changes of Workflows Without Losing Control," Journal of Intelligent Information Systems, vol. 10, no. 2, 1998. pp. 93-129,

[30] A. Barros, M. Dumas, and P. Oaks, "A Critical Overview of the Web Services Choreography Description Language," BPTrends, http://www.bptrends.com, 2005,

[31] W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede et al., "Life After BPEL?," Formal Techniques for Computer Systems and Business Processes, 2005. pp. 35-50,

[32] J. Mendling, and M. Hafner, "From Inter-organizational Workflows to Process Execution: Generating BPEL from WS-CDL," On the Move to Meaningful Internet Systems 2005: OTM Workshops, Lecture Notes in Computer Science R. Meersman, Z. Tari and P. Herrero, eds., 2005. pp. 506-515: Springer Berlin / Heidelberg,

[33] OMG, "Business Process Modeling Notation 1.2, 1st ed.," 2009.