University of Wollongong

Research Online

Faculty of Engineering and Information
Sciences - Papers: Part A

Faculty of Engineering and Information
Sciences

1-1-2013

Incremental service level agreements violation handling with time impact analysis

Azlan Ismail

Universiti Teknologi MARA, ai423@uow.edu.au

Jun Yan

University of Wollongong, jyan@uow.edu.au

Jun Shen

University of Wollongong, jshen@uow.edu.au

Follow this and additional works at: <https://ro.uow.edu.au/eispapers>

Part of the [Engineering Commons](#), and the [Science and Technology Studies Commons](#)

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Incremental service level agreements violation handling with time impact analysis

Abstract

This research addresses a critical issue of service level agreement (SLA) violation handling, i.e., time constraint violation related to service-based systems (SBS). Whenever an SLA violation occurs to a service, it can potentially impact dependent services, leading to unreliable SBS. Therefore, an SLA violation handling support is much required to produce a robust and adaptive SBS. There are several approaches to realizing exceptions and faults handling support for SBS, focusing on the detection stage, the analysis stage, and the resolution stage. However, the current works have not considered the handling strategy that takes the impact information into account to reduce the amount of change. This is essential to effectively handle the violation while consuming a reasonable recovery execution time. Therefore, in this research, we propose an incremental SLA violation handling with time impact analysis. The main role of the time impact analysis in the approach is to automatically generate an impact region based on the negative time impact conditions. Furthermore, the time impact analysis generates the appropriate time requirements. Both the region and the requirement are useful to support the recovery process. Based on a simplified evaluation study, the outcome suggests that the proposed approach can reduce the amount of service change within a reasonable recovery execution time.

Keywords

violation, handling, impact, service, analysis, incremental, agreements, level, time

Disciplines

Engineering | Science and Technology Studies

Publication Details

Ismail, A., Yan, J. & Shen, J. (2013). Incremental service level agreements violation handling with time impact analysis. *Journal of Systems and Software*, 86 (6), 1530-1544.

Incremental Service Level Agreements Violation Handling with Time Impact Analysis

Azlan Ismail^a, Jun Yan^b, Jun Shen^b

^a*Faculty of Computer and Mathematical Sciences,
Universiti Teknologi MARA (UiTM),
Selangor, 40450, Malaysia.*

^b*School of Information Systems and Technology,
University of Wollongong,
NSW, 2522, Australia.*

Abstract

This research addresses the handling issues of SLA violation i.e., time constraint violation related to Service-based Systems (SBS). Whenever a Service Level Agreement (SLA) violation occurs to a service, it can potentially impact dependent services, which lead to unreliable SBS. Therefore, an SLA violation handling support is much required to produce a robust and adaptive SBS. There are several approaches for realizing exceptions and faults handling support for SBS that focus on the detection stage, the analysis stage, the resolution stage. However, the current works have not considered the handling strategy that takes the impact information towards reducing the amount of change. This is essential to effectively handle the violation while consuming a reasonable recovery execution time. Therefore, in this research, we propose an incremental SLA violation handling with time impact analysis. The main role of the time impact analysis in the approach is to automatically generate an impact region based on the negative time impact conditions. Furthermore, the time impact analysis generates the appropriate time requirements. Both, the region and the requirement are useful to support the recovery process. Based on a limited scale of the evaluation study, the outcome suggests that the proposed approach can reduce the amount of service change within a reasonable recovery execution time.

Email addresses: azlanismail@tmsk.uitm.edu.my (Azlan Ismail),
jyan@uow.edu.au (Jun Yan), jshen@uow.edu.au (Jun Shen)

Keywords: impact analysis, recovery, Service-based Systems, Service Level Agreements, time constraints, violation handling

1. Introduction

A service-based system (SBS) contains a set of services which interact between each others to perform the application's goals. SBS is fundamentally designed and developed using service-oriented architecture (SOA) [39]. The current Web Service technology to express the process model of SBS is WS-BPEL [37].

Service Level Agreements (SLA) have been proposed as an essential mechanism to assess the enactment of SBS. Conceptually, an SLA contains a set of agreed terms related to the provisioning service. The terms represent the demands of service consumer and the constraints of the service provider. The service consumer may demand certain quality aspects of the service. The service provider may impose certain time constraints related to the service. The time requirements are expressed by the service consumer of the service and the time constraints are imposed by the service provider for the resource capability. Herein, the resource capability is a general term that refers to resources that a service provider is able to utilize to provide the service, such as CPU time, hard disk space, etc. Thus, the time constraints represent the time segments of the resource capability. For example, a time constraint might define that a service is available at the agreed quality from time A to time B.

In reality, the enactment of SBS might be dealing performance degrading or quality change due to exception or fault condition such as resource unavailability of a component service. As a result, the established SLA may be violated. Thus, an SLA violation handling support is much needed to support the execution of SBS to make the SBS becomes robust and adaptive.

Recent works e.g., [48], [40] has addressed a few issues in handling violation. As a result, several approaches have been proposed that include the aspect of handling patterns, the detection mechanism, the diagnosis mechanism, and the resolution mechanism. However, existing works have not adequately addressed the SLA violation handling requirements especially in three aspects:

- *Reducing amount of change* - The context of change can be varied. The change may involve the activity of changing the process model,

process instance, service instance and/or changing the service information. Furthermore, the change activity is motivated by the recovery mechanisms such as the service replacement, the SLA renegotiation, and recomposition. Thus, the amount of change refers to the number of changes needed for the sake of recovery. To simplify the discussion, the proposed model focuses on the change of service due to the service replacement. Within this focus, we believe that an SLA violation may require smaller region that contains less number of services e.g., two interrelated services to be replaced. If the region is not able to be recovered, it can be expanded gradually rather than considering the entire future services (the services that have not been executed) or the entire process. In addition, reducing the amount of service change can contribute to minimizing the time for executing the recovery process.

- *Identifying appropriate region* - This requirement is related to reducing the amount of change. The appropriate region refers to the region to be prioritized for the recovery process. One of the strategies is to analyze the impact caused by the violation which produces an impact region. The impact analysis has been addressed largely in the area of change management for the software evolution and maintenance. However, existing works have not adequately addressed the impact analysis for handling the SLA violation. More specifically, how to automate the impact analysis process that can produce and expand the region for supporting the incremental SLA violation handling.
- *Analysing time impact* - This requirement is related to analyzing the impact of the violation. The time impact has not been sufficiently addressed in the area of QoS-based on impact analysis. Herein, the time dimension views from multiple perspectives namely the time requirements expressed by service consumer, the time constraints imposed by service providers, and the estimated time before and after the violation. The time impact is useful for enabling the incremental SLA violation handling by determining the appropriate impact region and generating the regional requirements.

The initial work to address the about requirements have been proposed in [22]. This paper presents a comprehensive approach by focusing on the incremental SLA violation handling with time impact analysis. The proposed approach contains the modelling of violation handling with incremen-

tal strategy and the modelling of time impact analysis component. The main contributions of the proposed approach in this paper are twofold:

- *Incremental SLA Violation Handling Approach* - The proposed SLA violation handling comprises an impact analysis and the recovery analysis component to support the incremental strategy. The impact analysis is used to generate an appropriate impact region with a set of requirements to support the recovery process. Meanwhile, the recovery produces the appropriate recovery plan within the given region. The incremental strategy is classified into two types. The first type is the *initial increment* with the impact and recovery analysis for the new violation. The second type is the *subsequent increment* with the expansion of the impact and re-recovery of the existing violation and impact region. By applying the incremental strategy, it is believed that the implementation of SLA violation handling can potentially reduce the amount of service change and lead to minimizing recovery execution time.
- *Time Impact Analysis Approach* - The proposed time impact analysis can generate the appropriate impact region. Three classifications of impact regions are introduced, namely the initial impact region, the existing impact region and the expanded impact region. The identification of candidates for the impact region is based on the negative impact classifications, given as time inconsistent and unsatisfactory condition. The classification applies the rule-based approach. The rules are modelled by considering several information including the structure and time relation dependency, the build-time and run-time information, and the before-violation and after-violation information. The proposed time impact analysis also contains a set of estimations to produce the appropriate requirements of the generated impact region. The generated impact region enables the incremental violation handling strategy.

The detailed discussions of this research are organized as follows. Section 2 discusses the background information and related works that include the violation handling area and the time analysis. Section 3 introduces the proposed approach by providing the architecture of SLA violation handling support, the process flow for the incremental violation handling and the architecture of the impact analysis. Section 4 presents the modelling for the impact analysis in enabling the incremental violation handling. Section 5

provides the evaluation preparation and outcomes to assess the proposed approach. Section 6 concludes the paper and identifies the future works.

2. Related Works

This section reviews the existing works that contribute to realizing a violation handling approach for SBS. Then, we review the time modeling and analysis in Workflow Management Systems and SBS to position our work.

2.1. Violation Handling

2.1.1. Violation Context

Exceptions and faults are two common terms which can be associated with the violation of SBS. Both refer to the abnormal situation occurs to SBS that can cause unfulfillment of the system's goals. Several works have attempted to classify faults and exceptions. These classifications are essential towards handling the abnormal situation effectively and efficiently. Several classifications of faults or exceptions have been introduced in the literature. Some of them are interrelated. [43] views the exceptions of a workflow system, which includes item failure, resource unavailability, constraint violation, etc. [16] views the faults of SBS by classifying them into functional faults, operational faults and SLAs violation. [11] categorizes the faults of SBS from broad perspectives such as development faults and operational faults. In [1], they classify the faults of SBS based on the architectural perspectives such as Middleware level faults and Web service level faults.

In this research, we focus on the SLA violation of SBS. An SLA refers to the contractual document to govern the service provisioning between the service consumer and the service provider. It consists of a variety agreement terms including guarantee terms. The guarantee terms can include Service Level Objectives (SLO) to be fulfilled by the SBS. Example of SLO is the service guaranteed to be available from 8am until 5pm. Therefore, the SLA violation refers to the situation where the SLO is violated due to an unexpected situation causes by various faults such as resource unavailability.

2.1.2. Violation Handling Specifications

Several works have addressed the specifications for enabling automated violation handling. In [16], they proposed a policy language to express the handling logic between the exception and the recovery actions. [3] proposed

a supervision rule based on ECA to map between the monitoring specified using Web Service Constraint Language (WScOL) and recovery specified using Web Service Recovery Language (WSREL). In [47], a template based approach has been used to match the violation types with the respective recovery mechanisms. In [29], a rule-based language is proposed to support proactive exception handling.

2.1.3. Violation Handling Stages

Violation handling stages can be referred to the exception handling stages. There are mainly three stages: detection, diagnosis and resolution [44].

The detection stage involves the activity of identifying the problem occurs to the SBS. The detection can be done automatically with the monitoring approaches. Some of the existing approaches that enable the automated SLA violation detection are [32] based on EC-Assertion, [3] based on Web Service Constraint Language (WScOL), [2] by extending Active BPEL engine, [38] with VieDAME approach, [5] with the combination of Dynamo and Astro approaches. Detailed comparisons are provided in [5].

The diagnosis stage is needed to understand the deviation detected by the monitoring. Generally, the diagnosis aims to address the questions such as, which guarantee term caused the deviation, and which service caused the SLA deviation. A framework for enabling decentralized qualitative diagnosis has been proposed [13]. Other works address the diagnosis method in different contexts. In [45], they proposed QoS diagnosis for exception handling to identify the cause of the exception occurs in multi-agent systems and web services. In [9], the factor analysis approach is proposed to diagnose which service that causes the QoS violation of composite services. The work by [28] proposed a process quality factor analysis [48] to diagnose the influential factors for QoS violation of a service-based application. The work by [25] proposed a service dependency matrix to support the QoS diagnosis of SBS.

The resolution stage involves the activity of analyzing, planning and executing the appropriate recovery strategies and actions to recover from the violation. In [17], a planning-based repair approach is proposed to analyze for and plan a recovery of the service-based process. In the context of QoS or SLA, the recovery analysis can benefit from the adaptation literatures. In [20], they proposed a cost-benefit model based on the value of change information (VOC) proposed in [19] to decide when the adaptation (recovery) should occur. In [40], a QoS satisfaction approach is proposed to support the decision on the appropriate adaptation (recovery) strategy. The work by

[28] also proposed an adaptation (recovery) strategy selection approach.

The existing works provide the foundation to enable the SLA violation handling mechanism. However, current works do not study the significance of impact analysis in recovering from an SLA violation in the context of SBS.

2.2. Time Modelling and Analysis

The time dimension is one of the core problems of information systems, program verification, and other areas involving process modelling [24]. In the earliest work [24], the author discussed two types of time elements, time points and time intervals and proposed the interval-based algebra for representing time in natural language. These formalisms became the fundamental representation towards solving miscellaneous information system challenges. This includes the management of time properties in supporting the Lifecycle of a software system. In this section, some related works of time modelling and analysis are discussed from the workflow management systems and service-based systems.

- *Time Issues in Workflow Management Systems (WFMS)* - The workflow management research area has been the basis to enable the management of service-based system. In the time dimension, existing works have addressed the modelling and computation aspects [14] [33] [26] [8] [12]. In [14], the authors presented three types of time constraints, lower bound, upper bound and fixed-date constraint. Then, an approach to compute a timed workflow graph has been proposed. In [33], the concept of time duration with relative and absolute time is proposed. Other work such as [8] proposed different time granularity in scheduling autonomous agents. In [12], several classifications of time constraints are proposed. Firstly, the task-related constraints that refers to start time constraint or the end time constraint together with the duration time constraint. Secondly, the schedule-task constraints that refers to the time between the end time of a task and the start time of the succeeding task. Thirdly, the inter-task constraints that refer to combination time of a task with its successor. The work by [30] addressed the relation between time constraints and resource constraints. They proposed a verification approach to verify the time constraints of concurrent workflows which have resource dependency.
- *Time Issues in Service-based Systems (SBS)* - Although SBS shared some similar characteristics as WFMS, it has different challenges at-

tributed by the design principles based on SOA as well as the Web service technology standards. This requires different or innovative modelling and reasoning techniques to manage time elements. The related works can be viewed from three aspects namely, service composition, service procurement and service selection perspective. In the context of service composition, the work in [6] proposed time abstractions in the business protocols based on a finite state machine formalism. This modelling enables the compatibility and replaceability analysis for the business protocols. In [15], the authors modelled the time properties together with the workflow model and use it to check the conformance of timed orchestration against the global timed choreograph in the collaborative processes. In [18], they modelled timed properties based on the timed automata and use it for analyzing time compatibility of asynchronous services. The work by [46] modelled the timed properties based on Hierarchical Colored Petri Net and use it for analyzing time consistency in business collaboration. In the service composition based on the orchestration model, the work by [27] modelled the time properties and the time requirements of BPEL process based on the timed automata and duration calculus. This model enables time verification process against the time requirements based on model checking techniques. In [41], the authors modelled the time elements together with the process based on the workflow model to compute and predict time in avoiding deadline violations. From the service procurement, the time properties has been modelled as part of Quality Requirements Language (QRL) [42]. In [34], the authors utilized QRL to enable time consistency and time conformance analysis between the service request and the service offered by mapping to Constraint Satisfaction Problem. The time dimension is also being addressed in the context of time scheduling for SBS. The work by [31] proposed timed aware scheduling for transactional Web service composition to maintain atomicity. In the context SLA for SBS, the work by [36] proposed an extension of WS-Agreement with time elements.

Although time dimension has been addressed extensively in the literature, the time issues in the SLA violation handling introduce different challenges. The time elements need to be viewed from multiple perspectives namely, the service roles (i.e., service consumer and service provider), SLA life-cycle stages (i.e., establishment and enactment) and SLA violation stages (i.e.,

before and after violation). Furthermore, in the context of violation handling, the time reasoning can be used to understand the scope of the impact either for the dependent services or to the entire service composition. The generated elements are useful for the recovery process.

3. Incremental Handling Approach

3.1. Architecture of SLA Violation Handling Support

The architecture of SLA violation handling support is illustrated in Figure 1. It consists of SBS execution environment (SBSEE) and SLA violation handling components. In this research, the focus is on the impact analysis component that generates impact information for supporting the incremental SLA violation handling.

SBSEE contains the management console to enable deployment of BPEL process and the BPEL engine e.g., ActiveBPEL to enact a BPEL process. For the sake of violation handling, the BPEL engine has two core functionalities. First, it has the ability to provide the event data especially related to the time dimension such as the observed execution time and the violation time (Details are provided in Section 4.1.2 and 4.1.3). This can be applied using Aspect Oriented Programming (AOP) technology [4]. Second, it has the capability to implement the actual recovery execution based on the identified recovery plan. This is possible by embedding the recovery mechanisms into the BPEL engine [35].

The SLA violation handling support comprises of five key components, identified as follows:

- The *violation handling manager* acts as a central component to manage the interaction between SBSEE and internal components, and coordinate the process flow of internal components.
- The *monitoring and diagnosis component* are used to collect QoS information, detect and diagnose a violation. Essentially, the monitoring component contains two main modules. The first module is called an event receiver that is used to receive a set of events and identify its relation with the guarantee terms in the SLA. The second module is the monitor module that is used to estimate and check for any deviation. The diagnosis component contains a diagnoser to analyze the deviation data to decide for the occurrence of an SLA violation. If a violation

Figure 1: SLA Violation Handling Architecture

occurs, the diagnoser generates the respective violation information. In this research, we assume that the violation information can be provided by this component.

- The *impact analysis component* is used to generate an impact region with the region requirements. The outcome of this component enables the recovery process to prioritize the recovery scope. The component will re-execute whenever the recovery results in a failure status. In this case, the previous impact region will be incremented to consider more candidates for the next recovery stage. The component is the main focus of this paper.

- The *recovery analysis component* is used to plan the appropriate recovery mechanisms for the impact region. This component has the knowledge of potential recovery mechanisms such as renegotiation, replacement and recomposition. The appropriate recovery mechanisms can be selected by understanding the risks to the entire process in fulfilling the global requirements. If the appropriate recovery plan is found, the actual recovery will be implemented by SBSEE. Otherwise, a failure status is generated. In this research, we assume that the recovery status can be provided by this component to support the decision to re-execute the impact analysis.
- The *violation handling database* is used to store the data of four main aspects. First, a *process instance* that is used to store the graph and the time requirements information. Second, a *service instance* that is used to store the service and time constraint information. Third, a *violation* that is used to store the violation information. Fourth, an *impact* that is used to store the impact information. The detailed elements of each aspect are provided in Section 4.1.

3.2. Realizing Incremental SLA Handling Process

The proposed approach can automatically manage the SLA violation handling. The central component to enable the automated process is the violation handling manager (VHM). The Lifecycle of VHM begins as soon as an SBS is deployed to SBSEE. Upon the deployment, the VHM is initiated for a specific BPEL process model. Then, the VHM initiates a monitoring instance to monitor for a specific process instance. Once initiated, the monitoring instance interacts directly with SBSEE to collect the relevant data for violation checking and diagnosis purposes. The diagnosis task is triggered if there is a deviation found. The diagnosis may detect a violation with relevant information i.e., violation type and violation source. The violation and the process instance information are stored in the violation handling database (VHD). In addition, the monitoring instance notifies the VHM for the detected violation. Upon receiving a violation alert, the VHM notifies SBSEE to suspend the problematic process instance. All of these interactions are illustrated in Figure 2.

Based on the notified violation, the VHM will perform the incremental impact analysis, identified as follows:

Figure 2: Monitoring Stage of Violation Handling Manager

- *Initial Increment* - This stage is illustrated in Figure 3. This stage is the first time that the impact analysis is executed for the respective violation. Therefore, the VHM initiates the impact analysis instance and associate it with the violation. The analysis instance obtains the detailed violation and process information from the VHD to generate the impact information. The generated information is stored into the VHD. Then, the analysis instance forwards the impact status to the VHM. After that, the VHM initiates a recovery analysis instance and associate it with the respective violation and the impact status. The recovery instance obtains the violation, impact and process information from the VHD to identify the appropriate recovery plan. Then, the successful generation of the recovery plan is stored into the VHD. Besides that, the recovery instance notifies the VHM of the recovery analysis status.

If the recovery analysis results in an appropriate recovery plan, the VHM will notify SBSEE for the actual recovery implementation. The recovery plan is retrieved from the VHD and is forwarded to SBSEE. If the recovery analysis is unable to obtain the appropriate recovery plan, the VHM will execute the subsequent increment.

- *Subsequent Increment* - This stage is illustrated in Figure 4. This stage repeats the impact analysis for the same violation of a process instance.

Figure 3: Initial Increment of Violation Handling Manager

Therefore, the VHM re-execute the respective impact analysis instance. The analysis instance obtains the relevant information from the VHD to increment the impact region with additional information. The generated information is stored into the VHD. Then, the analysis instance forwards the impact status to the VHM. After that, the VHM initiates a new recovery analysis instance. The recovery instance obtains the relevant information from the VHD to identify the appropriate recovery plan. Then, the successful generation of the recovery plan is stored into the VHD. Besides that, the recovery instance notifies the VHM of the recovery analysis status. If the recovery analysis results in an appropriate recovery plan, the VHM will notify SBSEE for the actual recovery implementation as mentioned before. If the recovery analysis is unable to obtain the appropriate recovery plan, the VHM will re-execute the subsequent increment.

In the case where there is no more increment (unable to expand the

Figure 4: Subsequent Increment of Violation Handling Manager

impact) and the recovery analysis fails to generate the appropriate plan, the human intervention is required. However, this part is beyond the scope of this paper.

3.3. Architecture and Process Flow of Impact Analysis

In this research, the impact analysis component focuses on the time dimension. Architecturally, this component comprises three main modules, identified as follows:

- *Process Time Estimation (Module 1)* - This module aims to update the time attributes with the latest information especially after the violation has occurred. The estimation process performs two core activities. The first activity is to identify and confirm the workflow structure such as sequential and parallel structure of the process. Based on the identified workflow structure, three levels of reduction are applied. The

first level estimates the execution time of each service including the violated service. The second level estimates the start and finish time of each service while reducing the structure. Meanwhile, the third level estimates the total execution time and the total completion time of the reduced structure. The first and second activity is executed iteratively until the atomic service is found. The estimated values become the input for other modules.

- *Time Impact Region Generation (Module 2)* - This module aims to generate the impact region by analyzing the time impact. A region represents a subset of a process. There are two kinds of impact regions to be generated, namely the initial impact region and the expanded impact region. The initial impact region is needed for the initial increment of violation handling, while, the expanded impact region is needed for the subsequent increment of violation handling. The generation of the impact region is supported by the time impact condition analysis. There are two kinds of impact conditions, namely, the time inconsistency and the time unsatisfactory. The time inconsistency is concerned with the impact of the direct dependency relationship between services. The time unsatisfactory condition is concerned with the impact of global dependency relationship. The details are given in Section 4.3.1. The identified impact condition becomes the input to the region requirements generation module. It is also used to support the recovery process.
- *Region Time Requirements Generation (Module 3)* - This module aims to generate the time requirements related to the region. It comprises two core activities. The first activity is to identify the relevant requirements to be associated with the region. The identification is based on the filtering of the sets of requirements based on the violation category. The second activity is to estimate the values for each of the selected requirement parameters. The focused values are related to the minimum and maximum value. The outcome of this module is used to support the recovery process.

The process flow of this component is illustrated in Figure 5. The component will receive the current stage of the violation handling. If the cycle is at the initial increment (for a new detected violation), the estimation of

Figure 5: Process Flow of Impact Analysis

process time will be executed to update the time information especially after the violation has occurred. Then, an initial impact region is generated based on the time impact condition analysis. Finally, the region requirements are generated. In the case of subsequent increment, the existing estimation and impact region are taken into consideration. Thus, the component will generate an expanded impact region followed by the generation of region requirements. Both, the impact region and the region requirements are used to support the recovery process.

4. Time Impact Analysis Approach

4.1. Symbols Descriptions

The symbols related to modelling the time impact analysis are categorized into process instance, service instance, violation and impact. Each of them is explained as below.

4.1.1. Process Instance

It is modelled as a 4-tuple $PI = (G, WG, TR, TE_p)$ where:

- G is the process model represented as a graph $G = (N, R)$. N is a set of activities where each of them is implemented by a service. R is a set of flow to connect activities.
- WG is the sub process model represented as a sub graph $WG \in G$. A sub graph is based on a single workflow structure such as sequential structure, parallel structure or conditional structure.
- TR is a set of time requirement attributes associated with the process instance, given as $(mtex, dl)$ where $mtex$ is the maximum total execution time (relative) and dl is the deadline (absolute).
- TE_p is the estimated time set of a process p which comprises two time attributes, given as (tex, cpt) where tex is the estimated total execution time (relative) and cpt is the estimated completion time (absolute). Two classifications are introduced namely the estimated time set before the violation TE_p^{bf} (absolute) and the estimated time set after the violation TE_p^{af} (absolute).

4.1.2. Service Instance

It is modelled as a 4-tuple $SI = (ss, TC, TE_s, TO)$ where:

- ss is the execution state of a service which will be assigned to one of the following states, given as (ce, pe, ue) where ce is the complete execution state, pe is the partial execution state and ue is the unexecuted state.
- TC is a set of time constraints imposed by the service provider, given as (gex, gbv, gev, grc) where gex is the guaranteed maximum execution time (relative), gbv is the guaranteed begin availability (absolute), gev is the guaranteed end availability (absolute) and grc is the guaranteed recovery execution time (relative).

- TE_s is the estimated time set of a component service s which comprises three time attributes, given as (est, eft, eex) where est is the estimated start time (absolute), eft is the estimated finish time (absolute) and eex is the estimated execution time (relative). Two classifications are introduced namely, the estimated time set before the violation TE_s^{bf} and the estimated time set after the violation TE_s^{af} .
- TO is a set of observed times, given as $(oex, oexl)$ where oex is the observed execution time (relative) for the completed service and $oexl$ is the observed execution time left (relative) for the partial completed service.

4.1.3. Violation

It is modelled as a 3-tuple $VI = (vc, vt, vs)$ where:

- vc is the violation category of the detected violation in relation to QoS taxonomy [7].
- vt is the violation time (absolute) of the composite service which determines the time of the initial violation.
- vs is the violation source which identifies the violated service.

4.1.4. Impact

It is modelled as a 4-tuple $IM = (IR, IL, RR)$ where:

- IR is the impact region that refers to a snapshot of the process model, thus $IR \in G$. The first node refers to the violated service and the other nodes refer to the impacted services. Three classifications of impact regions are introduced, IR^{init} that refers to the initial impact region, IR^{exi} that refers to the existing impact region and IR^{exp} that refers to the expanded impact region.
- IL is a set of impact level associated with the composite service, given as $(vtex, vcpt)$ where $vtex$ is the impact level of the violation related to the total execution time and $vcpt$ is the impact level of the violation related to the total completion time. The computation of both parameters is presented in Equation 13 and 16.

- RR is a set of region requirement classifications $(C_1, \dots, C_m)$ associated with the impact region which can be based on the QoS classification [7]. RR^∂ is a set of region requirements that have been identified for the respective impact region with estimated values. In this research, the focus is on the time classification, given as $C_i = (r_x, r_y)$ where r_x is the required execution time of the impact region and r_y is the required completion time of the impact region. Each of the requirements is associated with a domain of values, given as $D_{r_x|r_y} = [mVal, \dots, MVal]$. In the domain, $mVal$ refers to the minimum value and $MVal$ refers to the maximum value.

4.2. Process Time Estimation

4.2.1. Process Reduction

The estimation approach proposed for this module is based on the QoS workflow reduction technique [10] [23]. The time-based reduction rules are adapted from our previous work [21] with some extension especially related to the run-time information. With this technique, the reduction problem aims to reduce the process model G into an atomic node G' . It begins by searching for the first sub process $WG \subseteq G$ that contains the first node until the last sub process that contains the last node. This is important since the focus is on the time dimension. For each reduction cycle, there are three reduction steps:

- *Pre-reduction Step* - This step estimates the current execution time $eex \in TE_s$ of each component service in the identified sub process. This information is important to support the actual reduction process. The estimation takes the guaranteed times imposed to the service $(gex, grc) \in TC$, the observed time information $(oex, oexl) \in TO$ and the execution state of service ss . The formulation is given as follows:

$$eex(s_i) = \begin{cases} oex(s_i) + grc(s_i) + oexl(s_i), & \text{if } ss = pe \\ gex(s_i) + grc(s_i), & \text{if } ss = ue \\ oex(s_i) + oexl(s_i), & \text{if } ss = pe \\ gex(s_i), & \text{if } ss = ue \\ oex(s_i), & \text{if } ss = ce \end{cases} \quad (1)$$

- *Reduction Step* - The estimation for the actual reduction stage aims to estimate the start time $es(s_i)$ and finish time $eft(s_i)$ of each component service to be combined with the identified workflow structure $WG \subseteq G$. The estimation is driven by the outcome of the component level estimation, the type of the workflow structure (i.e., the sequential structure, the split structure and the join structure), the time constraints $gbv \in TC$ and the violation time $vt \in VI$.

In the case of sequential structure such that $s_i \prec s_j$ the estimated start time $est(s_j)$ and the estimated finish time $eft(s_j)$ can be formulated as follows:

$$est(s_j) = \max(gbv(s_j), eft(s_i), vt) \quad (2)$$

$$eft(s_j) = est(s_j) + eex(s_j) \quad (3)$$

In the case of split structure such that $s_i \prec (s_j^1, \dots, s_j^k)$, the estimated start time of each $est(s_j^k)$ can be formulated as in Equation 2. Meanwhile, the estimated finish time of each $eft(s_j^k)$ can be formulated as in Equation 3. In the case of join structure such that $(s_i^1, \dots, s_i^k) \prec s_j$, the estimated start time of $est(s_j)$ can be formulated as follows:

$$est(s_j) = \max(gbv(s_j), \max(eft(s_i^1), \dots, eft(s_i^k)), vt) \quad (4)$$

Meanwhile, the estimated finish time of $eft(s_j)$ can be formulated as in Equation 3.

- *Post-reduction Step* - The estimation of post-reduction estimation aims to estimate the total execution time tex and the total completion time cpt of the composite service. The estimation takes the outcome of the reduction stage estimation into consideration. Therefore, assuming s_i is a composite service (not the component service), the estimation can be formulated as follows:

$$tex(s_i) = eft(s_n) - est(s_k) \quad (5)$$

$$cpt(s_i) = eft(s_n) \quad (6)$$

In Equation 5, the total execution time is depicted by computing the difference between the estimated finish time of the last service $eft(s_n)$ where n refers to the last service and the estimated start time of the first service $est(s_k)$ where k refers to the first service in the workflow structure WG . This strategy will be able to consider the potential waiting time that may occur due to the presence of time constraints [21].

In Equation 6, the total completion time $cpt(s_i)$ is depicted by taking the value of the estimated finish time of the last service $eft(s_n)$ where n refers to the last service. In the case where the reduction is not completed yet, the outcome of the post-reduction stage estimation will become the input of the pre-reduction stage estimation. Therefore, assuming s_i is a composite service, the following equation is applied:

$$eex(s_i) = tex(s_i) \quad (7)$$

In this equation, the total execution time of the composite services becomes the estimated execution time of a combined service.

4.2.2. Algorithm

The algorithm of the estimation process is shown in Algorithm 1. This process requires a set of input elements to produce a set of estimated information after the violation. The process begins with identifying the workflow structure of the composition model. If the structure is sequential (line 2), then the pre-reduction estimation is applied to each service (line 3 - 5). It is followed by the reduction estimation accordingly (line 6). In the case that the structure is AND-split or XOR-split (line 8), pre-reduction estimation is applied to each service (line 9 - 11). It is followed by the reduction estimation accordingly (line 12). In addition, if the structure is AND-join or XOR-join (line 14), pre-reduction estimation is applied to each service (line 15 - 17). It is followed by the reduction estimation accordingly (line 18). Then, post-reduction estimation is applied to a single workflow structure (line 20). The estimation is done repetitively until G becomes an atomic service (line 1).

Require: $(G, TE_p^{bf}, TE_s^{bf}, TC, TO, ss, VI)$
Ensure: TE_p^{af}, TE_s^{af}

```

1: while  $G$  is not atomic do
2:   if ( $struc(WG) == \text{sequential}$ ) then
3:     for each  $s_i \in WG$  do
4:       compute  $eex(s_i)$  based on Equation 1
5:     end for
6:     reduce  $WG$  with Equation 2 and 3
7:   end if
8:   if ( $struc(WG) == \text{AND-split or XOR-split}$ ) then
9:     for each  $s_i \in WG$  do
10:      compute  $eex(s_i)$  based on Equation 1
11:    end for
12:    reduce  $WG$  with Equation 2 and 3
13:   end if
14:   if ( $struc(WG) == \text{AND-join or XOR-join}$ ) then
15:     for each  $s_i \in WG$  do
16:      compute  $eex(s_i)$  based on Equation 1
17:    end for
18:    reduce  $WG$  with Equation 4 and 3
19:   end if
20:   compute  $tex(WG), cpt(WG), eex(WG)$  based on Equation 5, 6, 7
21: end while

```

Algorithm 1: Process Time Estimation Algorithm

4.3. Time Impact Region Generation

The aim of this process is to generate the impact region. In relation to the incremental strategy, there are two kinds of the impact region to be generated depending on the incremental stage of violation handling. Firstly, the initial impact region IR^{init} which is needed for the initial increment. Secondly, the expanded impact region IR^{exp} which is needed for the subsequent increment. The time impact condition is used to identify the candidates for both types of impact region. Thus, in this section, the time impact condition analysis is introduced, followed by the generation of the initial impact region and the generation of expanded impact condition.

4.3.1. Analyzing Time Impact Condition

There are two kinds of time impact classifications to determine the impact region candidates, identified as, time inconsistency and time unsatisfactory condition. Both classifications are related to the dependency relationships of services. Therefore, the context of dependency is defined first followed by the definition of the impact condition. The time inconsistency is concerned with the impact of the direct dependency relationship between services. The direct dependency refers to two interrelated services, given as $s_i \prec s_j \in G$ where s_i is the preceding service of s_j and s_j is the succeeding service of s_i .

Hence, the time inconsistency can be defined as follows:

Definition 1. *Time Inconsistency* refers to a condition where the dependent service is determined to violate its time constraints due to the violation occurs in the succeeding service.

Based on this definition, the time inconsistency, symbolized as T_{inc} is determined based on the following rules:

$$T_{inc} = \begin{cases} true, & \text{if } eft(s_i) + eex(s_j) > gev(s_j) \\ false, & \text{if } eft(s_i) + eex(s_j) \leq gev(s_j) \end{cases} \quad (8)$$

In this equation, eft and eex are the estimated finish time and the estimated execution time of a component service. The data for the estimated finish time and the estimated execution time are obtained from the previous estimation process. The parameter gev is one the time constraints in TC , identified as the guaranteed end availability.

The time unsatisfactory condition is concerned with the impact of global dependency. A global dependency refers to the relationship between a component service s_i and the entire process G , where $s_i \subseteq G$. This means, a time violation of s_i may cause a time violation of the entire process G .

Thus, the time unsatisfactory can be defined as follows:

Definition 2. *Time Unsatisfactory* refers to a condition where the entire process is expected to violate the time requirements due to the violation occurs in the component service.

Based on this definition, the time unsatisfactory, symbolized as T_{uns} is determined based on the following rules:

$$T_{uns} = \begin{cases} true, & \text{if } (tex(G) > mtex(G)) \text{ or } (cpt(G) > dl(G)) \\ false, & \text{if } (tex(G) \leq mtex(G)) \text{ and } (cpt(G) \leq dl(G)) \end{cases} \quad (9)$$

In this equation, (tex, cpt) are the estimated total execution time and the estimated completion time of the process instance. The data for these parameters are obtained from the previous estimation process. Meanwhile, $(mtex, dl)$ are the time requirements TR imposed to the process instance.

4.3.2. Initializing Impact Region

The basis to initiate the impact region IR^{init} is based on the process instance, service instance and violation information. The initialization requires two steps:

- The first step is to identify the first node of the impact region. For this reason, the violated source $vs \in VI$ is used to determine the first node of the impact region, given as $vs \in IR^{init}$.
- The second step to analyze the dependency. This is implemented by referring to the direct dependency and the global dependency as defined earlier. For the direct dependency, the violated service becomes the preceding service and the potential impacted service becomes the succeeding service. For the global dependency, the component service is referred to the violated service.
- The third step is to analyze for the time impact condition. This is implemented by referring to the time inconsistency and unsatisfactory as defined earlier.
- The fourth step is to decide for the impact region candidates. The decision is supported by a rule, identified as follows:

$$IR^{init} \in \begin{cases} s_j, & \text{if } (T_{uns} = true) \text{ or } (T_{inc} = true) \\ null, & \text{if } (T_{uns} = false) \text{ and } (T_{inc} = false) \end{cases} \quad (10)$$

4.3.3. Expanding Impact Region

The basis to expand the impact region IR^{exp} is based on the existing impact region IR^{exi} and the process model G . IR^{exi} is generated in the previous increment of violation handling.

The expansion requires three main steps, identified as follows:

- The first step is to identify the last service in the existing impact region IR^{exi} . This service becomes the starting point to expand the region. Given a service in the existing impact region, the last service is determined by the service which has no succeeding service.
- The second step is to analyze the dependency. This is implemented by referring to the direct dependency and the global dependency as defined earlier. For the direct dependency, the last service in the existing impact region IR^{exi} becomes the preceding service and the potential impacted service becomes the succeeding service. The potential direct dependency can be illustrated as in Figure 6. In the figure, there are two areas. The first area refers to the existing impact region. The second area refers to the potential direct dependency. Illustration (A) represents the situation of two last services in the existing impact region that has a similar direct dependency. Illustration (B) represents the situation where the last two services in the existing impact region that have different direct dependency. Illustration (C) represents the situation of the last service in the existing impact region that has a single direct dependency.

Meanwhile, for the global dependency, the component service is referred to the violated service.

- The third step is to analyze the time impact condition. This is implemented by referring to the time inconsistency and unsatisfactory as defined earlier.
- The fourth step is to decide the impact region candidates. The decision is supported by a rule, identified as follows:

$$IR^{exp} \in \begin{cases} s_j, & \text{if } (T_{uns} = true) \text{ or } (T_{inc} = true) \\ null, & \text{if } (T_{uns} = false) \text{ and } (T_{inc} = false) \end{cases} \quad (11)$$

Figure 6: Potentil Direct Dependency for the Existing Impact Region

4.3.4. Algorithm

The process of generating the impact region is shown in Algorithm 2. The algorithm requires a set of inputs to produce the initial impact region or the expended impact region with the expended status. The process begins with checking for the increment status (line 1) which is obtained from the violation handling manager.

In the case of the initial increment (line 1 - 13), the violated service is assigned to the initial impact region (line 2). Then the process of analysis of time impact condition begins with checking for the time unsatisfactory (line 3). In the case of true (time unsatisfactory), all services that are direct dependency is taken as part of the initial impact region (line 4 - 6). In the case of false (time satisfactory), all services that are direct dependency is analyzed for time inconsistency (line 8 - 12). The inconsistency relation is taken as part of the impact region.

```

Require:  $(G, TE_p^{af}, TE_s^{af}, TR, TC, IR^{exi}, VI, inc)$ 
Ensure:  $IR^{init}, IR^{exp}, exStatus$ 
1: if  $(inc == initial)$  then
2:    $IR^{init} \leftarrow vs$ 
3:   if  $(vs \prec G$  is time unsatisfactory based on Equation 9) then
4:     for each  $(vs \prec s_j, vs \in IR^{init}, s_j \in G)$  do
5:        $IR^{init} \leftarrow s_j$ 
6:     end for
7:   else
8:     for each  $(vs \prec s_j, vs \in IR^{init}, s_j \in G)$  do
9:       if  $(vs \prec s_j$  is time inconsistency based on Equation 8) then
10:         $IR^{init} \leftarrow s_j$ 
11:      end if
12:    end for
13:   end if
14: else
15:   if  $(vs \prec G$  is time unsatisfactory based on Equation 9) then
16:     for each  $(s_i \prec s_j, s_i \in IR^{init}, s_j \in G)$  do
17:        $IR^{exp} \leftarrow s_j$ 
18:     end for
19:   else
20:     for each  $(s_i \prec s_j, vs \in IR^{init}, s_j \in G)$  do
21:       if  $(s_i \prec s_j$  is time inconsistency based on Equation 8) then
22:         $IR^{exp} \leftarrow s_j$ 
23:      end if
24:    end for
25:   end if
26: end if
27: if  $(|IR^{init}| > 1) \text{ or } (|IR^{exp}| > |IR^{exi}|)$  then
28:    $exStatus = true$ 
29: else
30:    $exStatus = false$ 
31: end if

```

Algorithm 2: Time Impact Region Generation Algorithm

For the subsequent increment (line 14 - 26), the time unsatisfactory is reanalyzed (line 15). This is necessary since the existing violation might not impact the entire process. If the time unsatisfactory occurs, then all services

that are direct dependency to the last service(s) of the existing impact region are taken as part of the expanded impact region (line 16 - 18). If the time unsatisfactory does not occur, then each of the potential direct dependency is analyzed for the time inconsistency (line 20 - 24). All relations that are identified as time inconsistency are taken as part of the expanded impact region (line 22).

The success or failure generation of impact region is kept in a status variable to enable further decision by the violation handling manager. In the case of the successful generation (line 27), the status takes the true value (line 28), otherwise the status takes the false value (line 30).

4.4. Region Time Requirements Generation

The aim of this process is to produce the time requirements of the generated impact region. The requirements are useful to support the recovery process. The generation has two main steps. The first step is to identify the relevant requirements classifications. The second step is to fulfill the requirements with the respective values through a set of estimations. The details are presented in the following sections.

4.4.1. Time Requirements Identification

This process aims to identify the relevant requirements RR^∂ . It requires two core inputs namely the set of requirement classifications $(C_1, \dots, C_m) \in RR$ and the violation category $vc \in VI$. The direct solution to determine the appropriate classification is to map the violation category and the requirement classifications.

Since the focus is on the time dimension, the assumption is made that this process decides on a single classification that has two required parameters, given as $C_i = (r_x, r_y)$ where r_x is the required total execution time and r_y is the required total completion time. Each parameter is associated with a range or list of values, given as $D_{r_x|r_y} = [mVal, \dots, MVal]$ where $mVal$ refers to the minimum value and $MVal$ refers to the maximum value.

4.4.2. Time Requirements Estimation

This process aims to estimate the respective values for the identified requirements RR^∂ . For this reason, it requires additional information including the process model G , the estimated processing time information about before and after violation (TE_p^{bf}, TE_p^{af}) , the estimated service time information

before the violation TE_s^{bf} , the generated impact region IR and the violation time $vt \in VI$.

For the sake of simplicity, the discussion focuses on two types of time requirement parameters namely the required execution time of the impact region r_x and the required completion time of the impact region r_y .

Both parameters, r_x and r_y are associated with a range of acceptable values $D_{r_x|r_y} = [mVal, ..., MVal]$ where $mVal$ refers to the minimum value and $MVal$ refers to the maximum value. Thus, the aim of the generation process is to identify and compute the values of $(mVal_{r_x}, MVal_{r_x})$ and $(mVal_{r_y}, MVal_{r_y})$. The computation is explained as follows:

- *Required Execution Time* - The value of the minimum required execution time for the impact region $mVal_{r_x}$ can be easily set to 0. Meanwhile, the value for the maximum required execution time for the impact region $MVal_{r_x}$ can be obtained in several steps. The first step is to compute the execution time of the impact region before the violation occurs. It is formulated as follows:

$$tex(IR)^{bf} = eft(s_n) - est(s_k), \text{ where } (s_k, ..., s_n) \in IR \quad (12)$$

The second step is to compute the violation range of the entire process G . It is formulated as follows:

$$vtex(G) = tex(G)^{af} - tex(G)^{bf} \quad (13)$$

The third step is to compute the maximum required execution time for the impact region $MVal_{r_x}$ based on the previous steps. It is formulated as follows:

$$MVal_{r_x} = tex(IR)^{bf} - vtex(G) \quad (14)$$

- *Required Completion Time* - The value of the minimum required completion time for the impact region $mVal_{r_y}$ can be set to the violation time $vt \in VI$. Meanwhile, the value for the maximum required completion time for the impact region $MVal_{r_y}$ can be obtained based on the several steps. The first step is to compute the completion time of the impact region before the violation occurs. It is formulated as follows:

$$cpt(IR)^{bf} = eft(s_n), \text{ where } s_n \in IR \quad (15)$$

The second step is to compute the violation range of the entire process G . It is formulated as follows:

$$vcpt(G) = cpt(G)^{af} - cpt(G)^{bf} \quad (16)$$

The third step is to compute the maximum required completion time for the impact region $MVal_{ry}$ based on the previous steps. It is formulated as follows:

$$MVal_{ry} = cpt(IR)^{bf} - vcpt(G) \quad (17)$$

With these computations, the domain range of the refined requirements can be generated and used to support the recovery process.

4.4.3. Algorithm

The process of generating the region time requirements is presented in Algorithm 3. It requires a set of inputs and produces the classified region requirements with estimated values. In this algorithm, the violated time is set to time to support the assumption in this process (line 1). Then, the process begins with checking each requirement classification (line 2). If the required time requirement classification is found (line 3), then the respective formulations are applied (line 4 - 7). If the required requirement is another dimension (line 9), then a different set of formulations is applied which is beyond the scope of this research (line 10).

Require: $(G, RR, TE_p^{bf}, TE_p^{af}, TE_s^{bf}, VI)$
Ensure: RR^∂

```

1:  $vt = time$ 
2: for each  $C_i \in RR$  do
3:   if  $(C_i == vs)$  //where vs is a time dimension then
4:     set  $mVal_{r_x} = 0$ 
5:     compute  $MVal_{r_x}$  based on Equation 12,13,14
6:     set  $mVa_{r_y} = vt$ 
7:     compute  $MVal_{r_y}$  based on Equation 15,16,17
8:   end if
9:   if  $(C_i == vs)$  //where vs is another dimension then
10:    ....//depending on the dimension
11:   end if
12: end for

```

Algorithm 3: Region Time Requirements Generation Algorithm

5. Evaluations

The objective of the evaluation is to study the effectiveness of the proposed SLA violation handling approach with the impact analysis in reducing the service change. In addition, the evaluation aims to study the effect of reducing service change to the recovery execution time.

The evaluation is conducted based on the comparison and correlation study. A synthetic data is used to provide the core inputs to the study. The comparison study involves the SLA violation handling with impact analysis approach (the proposed approach) and the SLA violation without the impact analysis (the other approach). The focus attribute is the *number of service change* for both approaches. The attribute refers to the amount of service that needs to be recovered (or changed) in order to handle the violation (fulfill the time requirement). Ideally, the proposed approach contributes to reducing the service change as compared to the other approach.

The correlation study involves the time dimension in relation to the proposed approach. The focus attribute is the *average recovery execution time* for the proposed approach. This attribute refers to the time taken to recover the violation. By understanding the correlation, we can deduce the benefit of reducing the *number of service change* from the time perspective of the proposed approach.

5.1. Procedures

The evaluation procedure comprises three main steps, explained as follows:

- Step 1 - The preparation of data that covers three core models. The first model is related to the generation of process instances with respective time information. The second model is related to the injection of a violation to each of the process instances. The third model is related to the generation of service candidates to enable the recovery process.
- Step 2 - The execution of both approaches, namely, the proposed and the other approach based on the generated data. The proposed approach iterates between the impact analysis and the recovery process. The impact analysis is executed based on the model presented in section 4. The other approach proceeds with the recovery process. For both approaches, the replacement mechanism is applied.
- Step 3 - The data collection, analysis and findings. The data collection happens during the execution of both approaches. Two attributes of data are collected. The first attribute is the *number of service change* for supporting the comparison-based study. The second attribute is the *average recovery execution time* for supporting the correlation-based study. Based on these data, the analysis is conducted to determine the effectiveness of the proposed approach in reducing the service change and consuming reasonable recovery execution time.

5.2. Data Set Preparation

The evaluation contains three main data preparations. The first data set relates to the process instances based on specific process model. The process instances represent the potential BPEL process to be executed by the BPEL engine. For the evaluation, 400 process instances are generated based on the similar process model. For simplicity, the process model contains 6 nodes and interacts sequentially. The parameter settings are summarized in Table 1.

The second data set relates to the violation scenario. For this reason, each of the process instances is injected with a delay problem (longer execution time) to introduce the time constraint violation. The violation is associated with a component service that becomes the violated service. The

Table 1: Parameter Settings for the Process Model

Parameter	Value
Number of process instances	400 instances
Number of nodes per instance	6 nodes
Number of violated service per instance	1 node
Location of violated service per instance	1 or 2, ..., or 6 (randomly)

Table 2: Parameter Settings for the Violation

Parameter	Value
Type of violation	time constraints violation
Number of violated service	1 service per process
Source of violated service	Service 1 or 2, ..., or 6 (randomly)

source of violating service is selected randomly. The parameter settings are summarized in 2.

The third data set is related to the recovery process. To simplify the recovery analysis, the replacement mechanism is selected. For this reason, 6 groups of services are generated where each group comprises 1000 service candidates. For each component service, there are three time constraints imposed by the service provider namely the guaranteed begin availability, the guaranteed end availability and the guaranteed execution time. Meanwhile, the entire composite service is constrained with a fixed deadline, given as unit 90. The parameter settings are summarized in 3.

5.3. Outcomes

The chart in Figure 7 shows the comparison of the number of service change between both strategies. The lines represent the total number of instances for both strategies in relation to the number of service change

Table 3: Parameter Settings for the Service Candidates

Parameter	Value
Number of Service Group	6
Number of Service per Group	1000
Guaranteed begin availability	Unit 1 to 100 (randomly)
Guaranteed end availability	Unit 1 to 100 (randomly)
Guaranteed maximum execution time	Unit 1 to 60 (randomly)

Figure 7: Comparison of Number of Service Change for Both Strategies

needed for the sake of recovering the violation situation. As shown in the chart, there are a high number of instances needed to be changed with the number of service change is 1 for the *proposed approach* as opposed to the *other approach*.

However, the number of instances is decreasing with the number of service change greater than 1 for the *proposed approach* as compared to the *other approach*. This pattern is expected since the *proposed approach* strives to recover the violation as much as possible within the smallest scope. As contrary, the *other approach* will depend on the entire future services.

Furthermore, in average, the *other approach* requires 3 services to be changed, while the *proposed approach* requires 2 services to be changed for recovering purposes. We can conclude that the *proposed approach* contributes to reducing the number of service change as opposed to the *other approach* in recovering the violation.

The chart in Figure 8 shows the correlation between two attributes of the proposed approach namely the number of service change and the average recovery execution time. The circles represent the respective recovery time and instances of a specific number of service change. For instance, there are 236 instances which have at least 1 service that have been changed to the average of recovery time 6 ms. As shown by the circles, whenever the number of service change increases the average of recovery execution time increases as well. This pattern indicates a positive correlation between the service change attribute and the time attribute.

Correlation Between the Number of Service Change and the Average of Recovery Execution Time

Figure 8: Correlation between Service Change and Execution Time

In addition, the chart also shows that in the best case scenario where the number of service change is 1, the proposed approach takes a significantly less time with an average 6ms. Meanwhile, in the worst case scenario where the number of service change is 6, the proposed approach takes a reasonable time with an average 3000ms. Based on this observation, we can deduce that the proposed approach can efficiently recover and handle the violation in relation to the strategy of minimizing the number of service change.

6. Conclusion and Future Work

This research has addressed the issues of handling SLA violation for Service-based Systems. An SLA violation handling support for Service-based Systems Execution Environment is proposed to automatically handle the SLA violation and to overcome the identified issues. The proposed approach

includes the architecture of incremental SLA violation handling, the sequence diagram to realize the incremental handling process, the architecture and the process flow of impact analysis. The key contributions of this research are the incremental strategy and the time impact analysis classifications. The proposed approach can assist in reducing the amount of service change which leads to a reasonable execution recovery time.

The limitations of this work can be determined in three perspectives. Firstly, the proposed impact analysis rules are mainly concerned with the time dimension. Future works may consider other non-functional properties. Secondly, the proposed dependency analysis only examines the relationship of services at the same level. Future work may take hierarchical relationship of services. Thirdly, the SLA violation handling support is evaluated within a controlled experiment. A real world scenario can be conducted to evaluate the proposed approach.

Acknowledgment

The research especially for the first author is supported by the Ministry of Higher Education of Malaysia and Universiti Teknologi MARA, Malaysia. We would like to thank Mark Sifer for his valuable comments on the evaluation section. We also would like to thank the anonymous reviewers for their valuable comments and discussions of our work and this paper.

References

- [1] D. Ardagna, C. Cappiello, M. Fugini, E. Mussi, B. Pernici, P. Plebani, Faults and Recovery Actions for Self-Healing Web Services, in: 15th International World Wide Web Conference, WWW '06, pp. 189–196.
- [2] F. Barbon, P. Traverso, M. Pistore, M. Trainotti, Run-time Monitoring of Instances and Classes of Web Service Compositions, in: IEEE International Conference on Web Services, ICWS '06, pp. 63–71.
- [3] L. Baresi, S. Guinea, A Dynamic and Reactive Approach to the Supervision of BPEL processes, in: 1st India software engineering conference, ISEC '08, pp. 39–48.
- [4] L. Baresi, S. Guinea, L. Pasquale, Self-Healing BPEL Processes with Dynamo and the Jboss Rule Engine, in: International workshop on

Engineering of software services for pervasive environments, ESSPE '07, pp. 11–20.

- [5] L. Baresi, S. Guinea, M. Pistore, M. Trainotti, Dynamo + Astro: An Integrated Approach for BPEL Monitoring, in: IEEE International Conference on Web Services, ICWS '09, pp. 230–237.
- [6] B. Benatallah, F. Casati, J. Ponge, F. Toumani, On Temporal Abstractions of Web Service Protocols, in: Proceedings of the CAiSE'05 Forum, pp. 39–44.
- [7] S. Benbernou, I. Brandic, C. Cappiello, M. Carro, M. Comuzzi, A. Kertsz, K. Kritikos, M. Parkin, B. Pernici, P. Plebani, Modeling and Negotiating Service Quality, in: Service Research Challenges and Solutions for the Future Internet, 2010, volume 6500 of *Lecture Notes in Computer Science*, pp. 157–208.
- [8] C. Bettini, X.S. Wang, S. Jajodia, Temporal Reasoning in Workflow Systems, *Distributed and Parallel Databases* 11 (2002) 269–306.
- [9] L. Bodenstag, A. Wombacher, M. Reichert, M.C. Jaeger, Analyzing Impact Factors on Composite Services, in: IEEE International Conference on Services Computing, SCC '09, pp. 218–226.
- [10] J. Cardoso, A. Sheth, J. Miller, J. Arnold, K. Kochut, Quality of Service for Workflows and Web Service Processes, *Web Semantics: Science, Services and Agents on the World Wide Web* 1 (2004) 281–308.
- [11] K.M. Chan, J. Bishop, J. Steyn, L. Baresi, S. Guinea, A Fault Taxonomy for Web Service Composition, in: Service-Oriented Computing, ICSOC '07 Workshops, pp. 363–375.
- [12] C. Combi, G. Pozzi, Temporal Conceptual Modelling of Workflows, Springer-Verlag Berlin Heidelberg, 2003, pp. 59–76.
- [13] L. Console, C. Picardi, D.T. Dupre, A Framework for Decentralized Qualitative Model-based Diagnosis, in: 20th international joint conference on Artificial intelligence, IJCAI '07, pp. 286–291.
- [14] J. Eder, E.P. amd Michael Rabinovich, Time Constraints in Workflow Systems, in: Advanced Information Systems Engineering, CAiSE '99, pp. 286–300.

- [15] J. Eder, A. Tahamtan, Temporal Conformance of Federated Choreographies, Springer-Verlag Berlin Heidelberg, 2008, pp. 668–675.
- [16] A. Erradi, P. Maheshwari, V. Tosic, Recovery Policies for Enhancing Web Services Reliability, in: International Conference on Web Services, ICWS '06, pp. 189–196.
- [17] G. Friedrich, M. Fugini, E. Mussi, B. Pernici, G. Tagni, Exception Handling for Repair in Service-based Processes, IEEE Transactions on Software Engineering 99 (2010) 198–215.
- [18] N. Guermouche, C. Godart, Asynchronous Timed Web Service-aware Choreography Analysis, Springer-Verlag Berlin Heidelberg, 2009, pp. 364–378.
- [19] J. Harney, P. Doshi, Adaptive Web Processes using Value of Changed Information, Springer-Verlag Berlin Heidelberg, 2006, pp. 179–190.
- [20] Q. He, J. Yan, H. Jin, Y. Yang, Adaptation of Web Service Composition based on Workflow Patterns, Springer-Verlag Berlin Heidelberg, 2008, pp. 22–37.
- [21] A. Ismail, J. Yan, J. Shen, Verification of Composite Services with Temporal Consistency Checking and Temporal Satisfaction Estimation, in: Web Information Systems Engineering, WISE '09, pp. 343–350.
- [22] A. Ismail, J. Yan, J. Shen, Analyzing Fault-Impact Region of Composite Service for Supporting Fault Handling Process, in: IEEE International Conference on Services Computing, SCC '11, pp. 290–297.
- [23] M.C. Jaeger, G. Rojec-Goldmann, G. Muhl, QoS Aggregation for Web Service Composition using Workflow Patterns, in: 8th IEEE International Enterprise Distributed Object Computing Conference, EDOC '04, pp. 149–159.
- [24] F.A. James, Maintaining Knowledge About Temporal Intervals, Commun. ACM 26 (1983) 832–843.
- [25] Z. Jing, C. Yi-Chin, L. Kwei-Jay, A Dependency Matrix based Framework for QoS Diagnosis in SOA, in: IEEE International Conference on Service-Oriented Computing and Applications, SOCA '09, pp. 1–8.

- [26] E. Kafeza, K. Karlapalem, Temporally Constrained Workflows, Springer-Verlag Berlin Heidelberg, 1999, pp. 149–154.
- [27] R. Kazhamiakin, P. Pandya, M. Pistore, Representation, Verification, and Computation of Timed Properties in Web, in: Proceedings of the International Conference on Web Services, ICWS '06, pp. 497–504.
- [28] R. Kazhamiakin, B. Wetzstein, D. Karastoyanova, M. Pistore, F. Leymann, Adaptation of Service-based Applications based on Process Quality Factor Analysis, in: Service-Oriented Computing. IC-SOC/ServiceWave '09 Workshops.
- [29] K. Kim, I. Choi, C. Park, A Rule-based Approach to Proactive Exception Handling in Business Processes, Expert Systems with Applications 38 (2011) 394–409.
- [30] H. Li, Y. Yang, Verification of Temporal Constraints for Concurrent Workflows, Springer-Verlag Berlin Heidelberg, 2004, pp. 804–813.
- [31] A. Liu, H. Liu, Q. Li, L.S. Huang, M.J. Xiao, Constraints-Aware Scheduling for Transactional Services Composition, Journal of Computer Science and Technology 24 (2009) 638–651.
- [32] K. Mahbub, G. Spanoudakis, Monitoring WS-Agreements: An Event Calculusbased Approach, in: Test and Analysis of Web Services, 2007.
- [33] O. Marjanovic, Dynamic Verification of Temporal Constraints in Production Workflows, in: 11th Australasian Database Conference, 2000. ADC 2000. Proceedings, pp. 74–81.
- [34] O. Martin-Diaz, A. Ruiz-Cortes, A. Duran, C. Muller, An Approach to Temporal-Aware Procurement of Web Services, in: Service-Oriented Computing, ICSOC '05, pp. 170–184.
- [35] S. Modafferi, E. Mussi, B. Pernici, SH-BPEL: A Self-Healing Plug-in for WS-BPEL Engines, in: 1st workshop on Middleware for Service Oriented Computing, MW4SOC '06, pp. 48–53.
- [36] C. Muller, A. Ruiz-Cortes, P. Fernandez, Temporal-Awareness in SLAs: Why should we be concerned?, Springer-Verlag Berlin Heidelberg, 2009, pp. 165–173.

- [37] OASIS, Web Services Business Process Execution Language Version 2.0, OASIS, 2007.
- [38] M. Oliver, R. Florian, D. Schahram, Non-intrusive Monitoring and Service Adaptation for WS-BPEL, in: 17th international conference on World Wide Web, WWW '08, pp. 815–824.
- [39] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, Service-oriented Computing: State of the Art and Research Challenges, *Computer* 40 (2007) 38–45.
- [40] B. Pernici, S. Siadat, Adaptation of Web Services Based on QoS Satisfaction, volume 6568, Springer-Verlag Berlin Heidelberg, 2011, pp. 65–75.
- [41] H. Pichler, M. Wenger, J. Eder, Composing Time-Aware Web Service Orchestrations, volume 5565 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, 2009, pp. 349–363.
- [42] A. Ruiz-Cortes, A Semiquantitative Approach for the Automatic Management of Quality Requirements, Ph.D. thesis, 2002.
- [43] N. Russell, W. van der Aalst, A. ter Hofstede, Workflow Exception Patterns, in: Advanced Information Systems Engineering, CAiSE 2006, pp. 288–302.
- [44] S.W. Sadiq, M.E. Orlowska, On Capturing Exceptions in Workflow Process Models, in: Witold Abramowicz and Maria Orlowska (Ed.), BIS 2000.
- [45] N. Shah, R. Iqbal, K. Iqbal, A. James, A QoS Perspective on Exception Diagnosis in Service-oriented Computing, *Journal of Universal Computer Science* 15 (2009) 1871–1885.
- [46] H. Sun, J. Yang, w. zhao, Ticobtx-net: A Model to Manage Temporal Consistency of Service-oriented Business Collaboration, *IEEE Transactions on Services Computing* PP (2011) 1–1.
- [47] M. Wang, K.Y. Bandara, C. Pahl, Integrated Constraint Violation Handling for Dynamic Service Composition, in: IEEE International Conference on Services Computing, SCC '09, pp. 168–175.

- [48] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, F. Leymann, Monitoring and Analyzing Influential Factors of Business Process Performance, in: IEEE International Enterprise Distributed Object Computing Conference, EDOC '09, pp. 141–150.