International Journal of Software Engineering and Its Applications
Vol. 9, No. 2 (2015), pp. 151-162
http://dx.doi.org/10.14257/ijseia.2015.9.2.13

SLA Aware Dynamic Reconfiguration for the Safe Building Security System

Vallidevi Krishnamurthy and Chitra Babu

SSN College of Engineering
Department of Computer Science and Engineering
Anna University, Anna University
{vallidevik, chitra} @ ssn.edu.in
http://www.ssn.edu.in

Abstract

Service-Oriented Architecture (SOA) facilitates development of Business-to-Business (B2B) applications, by composing suitable services, either statically during design-time or dynamically at run-time. Dynamic reconfiguration in the service-oriented approach, involves replacing the statically composed services with some other ones, during run-time, only when it is necessary. This necessity might arise due to the need to accommodate change in user requirements, sudden unavailability of services or to keep up the non-functional requirements agreed in the Service Level Agreement (SLA).

Dynamic composition is another approach, where each service is selected and composed at run-time. This approach provides more flexibility in selecting appropriate services during run-time as opposed to the static composition. However, there is a possibility of an additional overhead, compared to dynamic reconfiguration. This paper discusses an approach for SLA aware dynamic reconfiguration, that happens for the same instance of execution and it has also been captured as a pattern named I-RAILS. The various ways of dynamically reconfiguring the services has been illustrated with a case study application namely the Safe Building Security system. This approach of dynamic reconfiguration of the services is compared with the dynamic service composition.

Keywords: *Dynamic Reconfiguration, Design Patterns, Service Oriented Architecture, Monitoring*

1. Introduction

Services that adapt themselves, either to the changes in the user requests or to avoid potential violations in the agreed SLA, are known as self adaptive services. Dynamic reconfiguration is a mechanism that facilitates such an adaptation. It can also be utilized to achieve high availability of the services in the composition.

Composition of web services has received much interest, to enable businesses to interact with each other and to facilitate seamless Business-to-Business (B2B) application integration. Business Process Execution Language (BPEL) [3, 10] is the de facto standard, which is designed for composing the web services together.

BPEL workflow logic contains a series of predefined primitive activities. Basic activities (receive, invoke, reply, throw, wait, pick) represent fundamental workflow actions which can be assembled using the logic supplied by structured activities. Pick is a structured BPEL activity and is part of the process definition, and it contains *OnMessage* and *OnAlarm* activities. Upon reception of a message, the control transfers to the *OnMessage* activity. Similarly, in case of a timeout, the control goes to the *OnAlarm* activity. In both cases, the events that are described under that activity take place.

ISSN: 1738-9984 IJSEIA
Copyright ⓒ 2015 SERSC

Fredj, *et al.,* [12] had proposed the SIROCO middleware platform, enabling the run-time, semantic-based service substitution. In this work, service substitution is realized by simply exchanging one atomic service with another at run-time. Further, dynamic reconfiguration achieved in [12, 14, 15], is only for the next instance.

Canfora, *et al.,* [13] discuss the QoS aware re-planning of the composite web services for the same instance. Whenever *re-planning* is triggered, a part of the service workflow, which is to be executed is re-planned. However, when this has been realized in implementation, human intervention is needed to change the bindings of services according to the re-planned workflow.

Bai, *et al.,* [1] had proposed the DRESR framework, which helps to achieve dynamic reconfiguration, in a service-oriented application, for the same instance. However, this work, achieves dynamic reconfiguration, by redirecting the control, to the execution of an alternative service that provides the same functionality, based on a predefined workflow.

Leitner, *et al.,* [6], realize dynamic reconfiguration, with the Windows Workflow Engine [7], along with aspect oriented technology [4, 5] by exchanging one composite service with another, during the same instance of execution. Even though, this approach provides run-time reconfiguration for the same instance based on the prediction of SLA violation, it does not continue monitoring after the composite service is replaced. However, in certain mission-critical applications, it becomes significant to monitor the individual services again, even after that point. In order to meet such stringent SLA requirements, this paper proposes a two-level dynamic reconfiguration approach namely SWEAR- Satisfying the **S**LA by changing the execution flo**W** of the composit**E** service **A**t **R**untime.

These two levels deal with:

1. Redirecting the control to an appropriate alternate workflow based on regular monitoring of services in the original workflow after a decision point.
2. Choosing suitable alternate services, in the redirected workflow, whenever necessary, through continuous monitoring of services in the redirected path.

This two-level reconfiguration approach has been captured as a pattern named I-RAILS. In order to demonstrate the proposed methodology for dynamic reconfiguration of a service oriented application, SOA based Safe Building Security (SBS) System has been developed as a case study. Windows Workflow Engine has been used for the implementation of dynamic reconfiguration. The proposed SWEAR approach and I-RAILS pattern; help to provide increasing levels of security for buildings which use this Safe Building Security System.

The remainder of the paper is structured as follows. Section 2 provides background of the Windows Workflow Engine. Section 3 explains the related work. Section 4 describes the SLA aware dynamic reconfiguration, *SWEAR approach for the composite services* in general and also specifically in the SBS system. In Section 5, the proposed pattern namely I-RAILS IS explained. Section 6 concludes and provides future directions. Pattern names are specified in small caps in this paper for the better understanding.

2. Background

The terminologies that are used in the Windows Workflow Engine are briefly explained as follows.

Windows Workflow Foundation (WF) [7, 8] is a Microsoft technology that provides an API, an in-process workflow engine. A series of distinct programming steps or phases could be defined as workflows. Each step is modeled in WF as an Activity. Custom activities can also be developed, for additional functionalities. Activities can be assembled visually into workflows, using the Workflow Designer. The designer can also be hosted in other applications. Encapsulating programming functionality into activities allows the developer to create more manageable applications; the *sequential workflow* style is

 Copyright ⓒ 2015 SERSC

straightforward and useful for repetitive, predictable operations that are always the same. The *state machine workflow* style consists of a set of event-driven states. The *data-driven* style relies on data, to determine, whether or not certain activities will run based on a local data state.

3. Related Work

The related work is discussed under two different broad categories.

1. Dynamic Reconfiguration without using Patterns:

- Li, *et al.,* [9], discuss the dynamic reconfiguration for the composite service by replacing one atomic service with another atomic service. Further, whenever, that too leads to a violation, some other atomic service that could be suitable for the requirement is searched. Whenever such an atomic service is not found, two services are replaced together, and it continues in this manner, till the maximum number of services that could be replaced together. In this case, initially, the searching time to identify the appropriate service for replacement is more. Added to that, the reconfiguration in this case, takes place only for the next instance.
- There are several works in the literature, which discuss about dynamic reconfiguration through aspects. These works address reconfiguration of the application, for the running instance itself. AO4BPEL [2] supports this kind of dynamic reconfiguration. However, AO4BPEL was built, to weave the cross-cutting concerns in the normal execution of the application. It was not built to dynamically reconfigure the services, in order to satisfy the SLA. Moreover, in these works, dynamic reconfiguration could be achieved with only a single option, for an alternative procedure.

2. Dynamic Reconfiguration using Patterns:

To overcome the limitation of a single choice for replacement in [2], Leitner, *et al.,* [6] used Windows Workflow Engine (WF), which focused on adaptation, by providing multiple choices for an alternative procedure, through the service composition fragments. Additionally, WF also helps to achieve dynamic reconfiguration with multiple choices of pre-defined paths. This is, a means to adapt the problematic part of the original static composition, in the main execution path. Nevertheless, there is always a chance that, there could be some unexpected delay, due to a service being overloaded or due to service unavailability. Hence, there is a continued need to monitor the individual services even in the adapted service composition. The *SWEAR* approach proposed in this paper, focuses on this sustained monitoring even after the changed workflow. Further, it also deals with the replacement of violating services in that workflow, with some other services, that provides the same functionality, whenever necessary. Weber, *et al.,* [11], have proposed two different types of patterns namely *adaptation patterns* and *patterns for predefined changes*. In [6], adaptation for the problematic composition instance, to prevent violation was addressed with the Windows Workflow Engine along with a few *adaptation patterns* [11]. The adaptations were realized with the 9 adaptation patterns of structural changes like, INSERT PROCESS FRAGMENT, DELETE PROCESS FRAGMENT, MOVE PROCESS FRAGMENT, REPLACE PROCESS FRAGMENT, SWAP PROCESS FRAGMENT and so on.

On the other hand, in the *SWEAR* approach, proposed in this paper, adaptation for the part of problematic composition has been realized, by using one of the patterns proposed under the type, *patterns for predefined changes* with two other patterns from the *adaptation patterns*. In the LATE SELECTION OF PROCESS FRAGMENTS pattern [11], the part of composition of services, in the main workflow is selected at run-time based on some predefined rules or user decisions. Likewise, with the proposed I-RAILS pattern which is

Copyright ⓒ 2015 SERSC

explained in Section 5, the part of composition is determined at run-time. Further, there are also two additional features:

a) insertion of service, in between the selected service execution, whenever needed, based on INSERT PROCESS FRAGMENT pattern.

b) replacement of a service, by some other service based on REPLACE PROCESS FRAGMENT pattern which in turn needs DELETE PROCESS FRAGMENT pattern too internally.

With these two features, the second level of dynamic reconfiguration discussed in Section 1 could be achieved. On the other hand, this additional level of dynamic reconfiguration is not achieved in [6].

In Section 4, the proposed approach for the dynamic reconfiguration, of the composite service is explained in detail.

3. SWEAR - Satisfying the SLA by Changing the Execution floW of the compositE Service At Runtime

Figure 1 shows the proposed *SWEAR* approach through which dynamic reconfiguration of composite services is achieved. In this approach, a composite service is statically composed using the Windows Workflow Engine. The *Predictor* identifies the prediction point for the composite service, beyond which the services are more likely to violate the SLA. There are many prediction point algorithms that are available and hence, the *Predictor* algorithm is not explained here, as it is beyond the scope of this paper. Once the predicton point in the workflow execution is reached, the *Predictor* invokes the *SLA monitor* which starts monitoring the services from that point onwards. Whenever, the *SLA monitor* compares the monitored value with the agreed SLA, and identifies the violation of QoS, specified in the global SLA, it triggers the *Windows Workflow Reconfigurator*. The *reconfigurator*, identifies a suitable workflow from the *alternative workflow repository*. This selection of the alternative workflow is based on two different criteria:

i. The number of services that remain to be executed in the current flow of execution.

ii. The QoS that needs to be met to satisfy the global SLA by the alternative workflow.

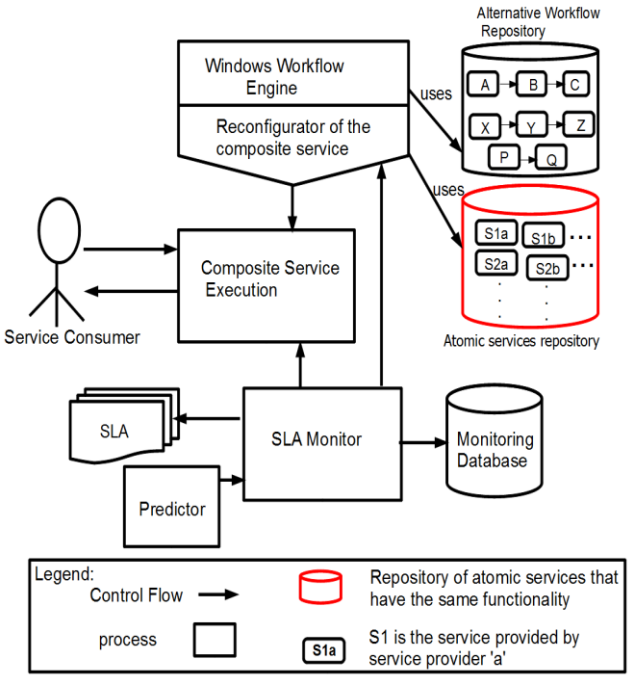

Figure 1. SWEAR Approach for the Composite Service

Copyright ⓒ 2015 SERSC

The workflows are also grouped together in the *workflow repository*, based on these two criteria. Even after this redirection in the workflow, the SLA monitoring is continued and any further violation predicted by the *SLA monitor*, triggers the *Windows Workflow Reconfigurator*. This *reconfigurator* replaces the next service to be executed, by another service that provides the same functionality with a better QoS, which is selected from the atomic services repository. In order to facilitate this, the services that have the same functionality with varying QoS are maintained as a group, in the atomic services repository as shown in Figure 1. The monitored values are also logged in the *Monitoring database* by the *SLA monitor*.

In Section 5, the proposed pattern I-RAILS is discussed in more detail.

5. I-RAILS - INSERTION AND REPLACEMENT IN LATE SELECTION OF PROCESS FRAGMENTS Pattern

Context

The service provider offers a highly critical composite service. Satisfying the agreed SLA between the service provider and the service consumer is very crucial. The provider has identified some potential violation of the atomic services which in turn will affect the global QoS of the composite service. Hence, the control has been redirected to an alternative path of service execution based on certain predefined rules.

Problem

How to ensure that the global QoS of the composite service that was agreed in the SLA, will not be violated even after redirecting to an alternative path of service execution?

Also known as

Profitable Partial Dynamic Composition

Forces

1. The service provider is offering a mission-critical service for the service consumer, where he has a few remedial measures to prevent the SLA violation by the prior prediction process. However, there are chances of these preventive measures being failed.
2. The service consumer has a choice to switch over to another service provider, if the QoS specified in the SLA of the original service provider is not satisfied. However, the original service provider is not willing to loose his potential customers. Instead, he prefers to make an additional investment to prevent any possible violation of SLA, even in the changed workflow

Solution

1. Based on the current monitored performance of the services that are executed after the prediction point, the workflow is redirected to an alternative path. The services in the alternative path are selected based on rigorous stress testing of services.
2. The services in the alternative path are still monitored. The monitored performance values are compared against the values specified in the SLA. Based on the result, decision is taken whether to execute the next service in the alternative path, or to replace it with some other functionally equivalent service with a better QoS.

Structure

The proposed pattern I-RAILS is shown in Figure 2b. In the selected path, there are three services namely, P, Q and R. Once, an alternative workflow path is selected, the service execution time is monitored. The service named M is the monitoring service that is

Copyright ⓒ 2015 SERSC

inserted using the INSERT PROCESS FRAGMENT pattern. Whenever any violation of the QoS specified in the SLA is predicted, the next service to be executed is replaced with an alternative service that could potentially compensate the delay in the service execution caused by the previous service. This replacement is realized with the REPLACE PROCESS FRAGMENT PATTERN.

Applicability

This pattern is applied, only when there is a critical need to satisfy the SLA agreed between the service consumers and the service providers at any cost.

Application Scenario

A Safe Building Security (SBS) system has been the chosen case study application, which was implemented, in the service-oriented architectural style. Initially, the users should register their finger prints, with the system. Such registered users, logging into the system is a normal scenario. On the other hand, if there is an unregistered user trying to login, continuously for the third time, to enter the building, the system should raise an alarm to alert the public living nearby. This system is also extended based on some critical factor as the deciding factor, to have some additional functionality. If the service consumer requests for the *Level-One* security, then, he could be provided the basic level security shown in Figure 3a, that has functionalities such as alerting the public by raising an alarm, logging service and informing the police as well as the owner, by sending an SMS to them simultaneously.

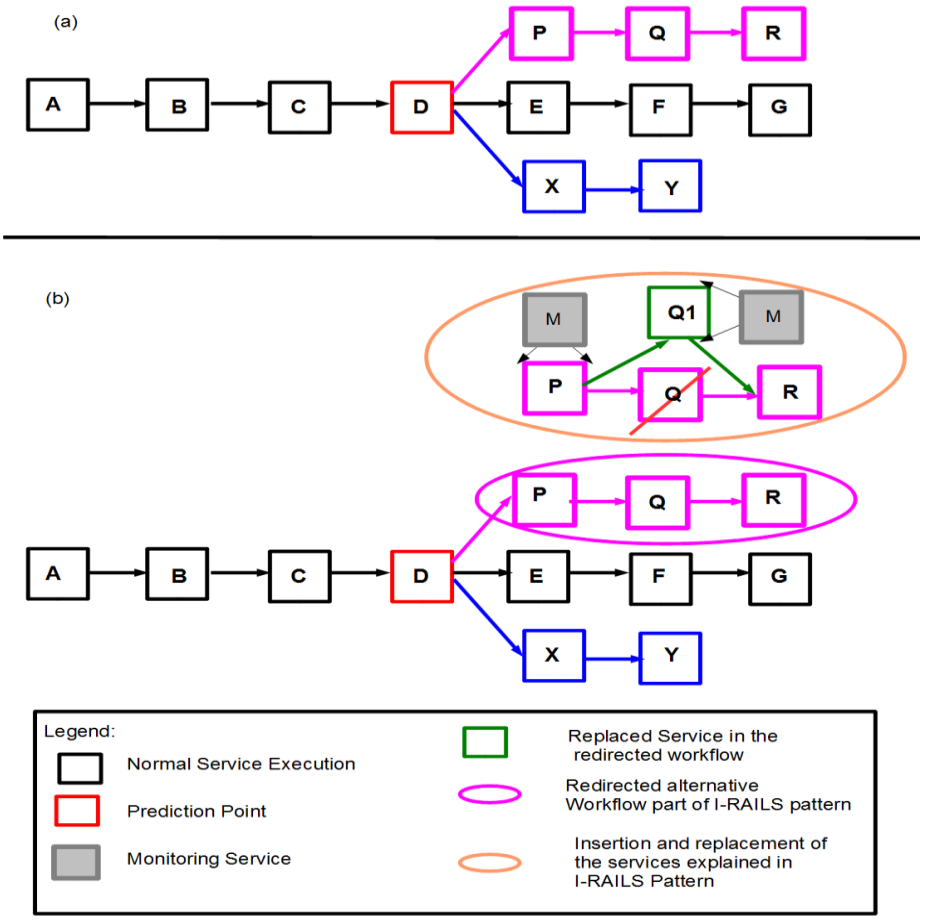

Figure 2. (a) Different Workflows with the Prediction Point D (b) Proposed I-RAILS Pattern

Copyright ⓒ 2015 SERSC

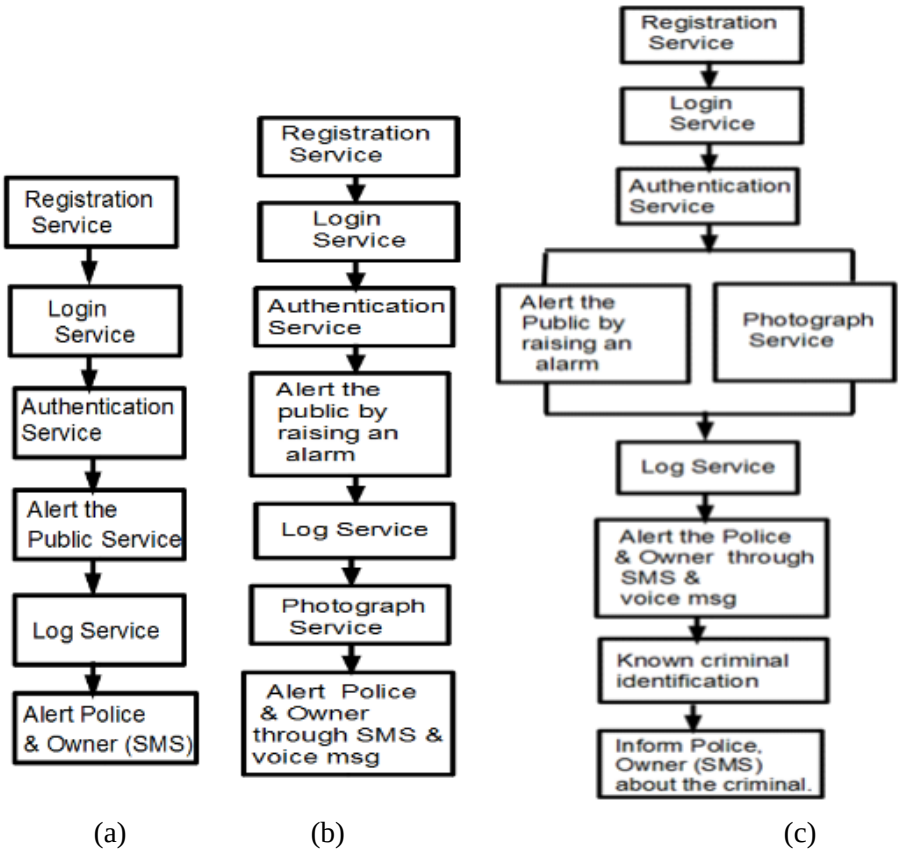

Figure 3. SBS System with Different Security Levels - (a) Level-One (b) Level-Two (c) Level-Three

Level-Two security, shown in Figure 3b, has *alerting the public service* by raising an alarm followed sequentially by two other services namely, *log service* and *photograph service*. *Alert the police and owner service*, which follows the *photograph service*, executes by sending a message to both of them simultaneously, through SMS and voice mail. The *photograph service* has the functionality of triggering the camera, to take a photograph of that place.

In the case of *Level-Three* security, shown in Figure 3c, *alerting the public service* and the *photograph service* are executed in parallel which is followed sequentially by the four services namely, *log service*, *Alert the police and owner service*, *Known criminal identification* service, *Alert the police and owner about the criminal service*. Buildings such as banks, which need more security than an individual house, could go with higher level of security, for which the critical factor could be more. Based on that critical factor, the service functionalites and QoS parameters agreed in the SLA differ. Depending on the change in user needs, the *SWEAR for the composite service*, could even select any level of security in the SBS system at run-time.

Implementation

Dynamic reconfiguration achieved with this pattern, could be compared to dynamic composition as monitoring happens, between every service execution, after the prediction point. Based on the monitored QoS value, the decision is taken, for the replacement of the current workflow, with a new one. Hence, as there is always a possibility to select an alternative workflow and again an alternative service too in that workflow, this approach is compared to the dynamic service composition. Once, the prediction point is identified by the *Predictor*, the services which are to be executed after that point, are in the

Copyright ⓒ 2015 SERSC

problematic zone. Till the prediction point, the services are executed in the same order as in the static composition. Only after the prediction point, the redirection in the path happens if needed. Additionally, dynamic replacement and deletion of services are also realized in the redirected path, in order to satisfy the global SLA that was agreed.

Application Scenario - Revisited for I-RAILS

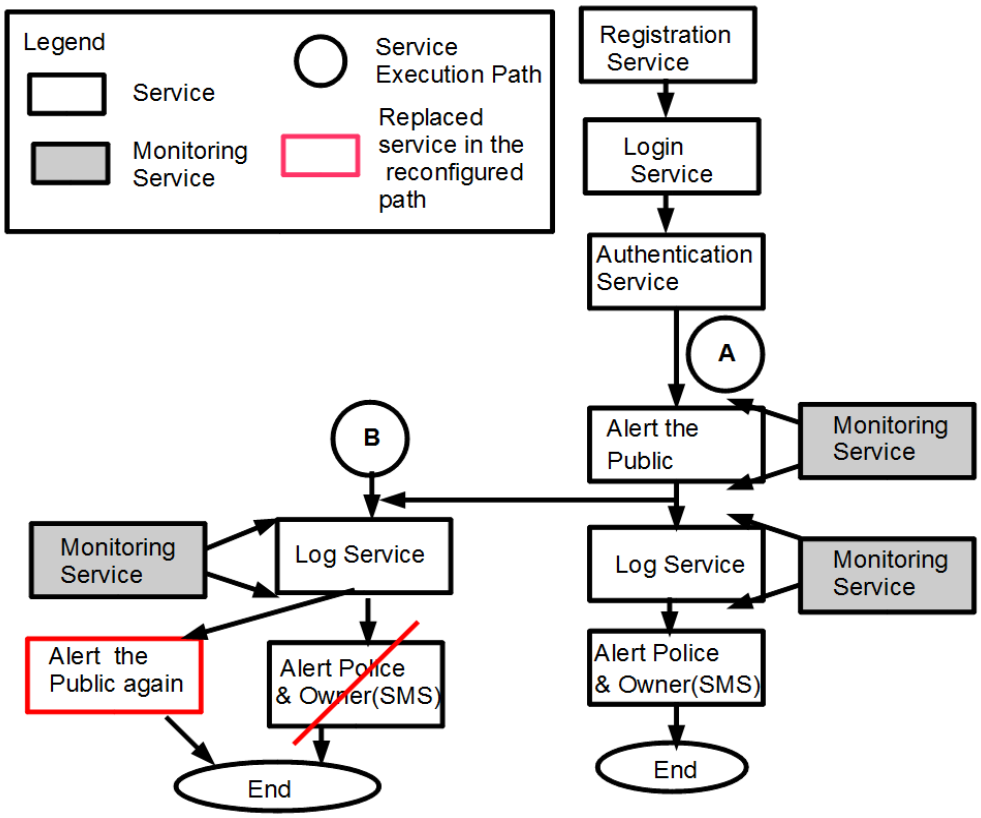

Figure 4. Level-One Security During Dynamic Reconfiguration

Level-One security with dynamic reconfiguration is shown in Figure 4. Here, there are two execution paths namely, A and B. Path A, is the flow corresponding to the normal execution of the *Level-One Security*. In path A, the services are continuously monitored, and there is no SLA violation, till the end of the application execution. Path B is another scenario, where there is no workflow replacement. Rather, in the main workflow itself, one of the services is replaced by another service. Monitoring service is executed in parallel to the execution of all the services after the *authentication service* which is the prediction point. In Figure 4, due to the temporary unavailability of the *SMS service*, the service to alert the public again is substituted instead of the service to alert the police and owner through SMS. The temporary unavailability of the *SMS service* is identified by the *SLA monitor service* which in turn triggers the service substitution in the original workflow.

Copyright ⓒ 2015 SERSC

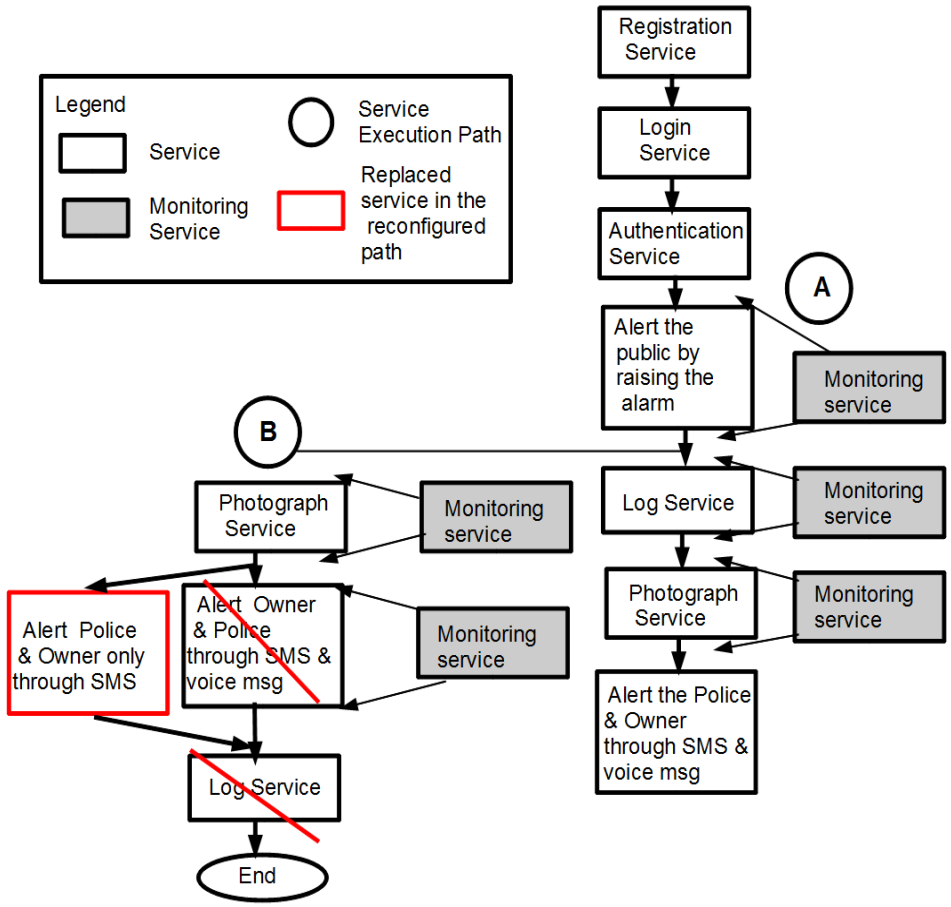

Figure 5. Level-Two Security During Dynamic Reconfiguration

Similarly, *Level-Two security* with dynamic reconfiguration is shown in Figure 5. Here, there are two execution paths namely, A and B. Path A, is the normal flow of execution corresponding to the *Level-Two Security*. In path A, all the services are continuously monitored, and there is no SLA violation, till the end of the application execution, even after the prediction point. This is one scenario. It is also possible that, SLA violation is predicted, after the execution of the *alert the public service* in path A, in which case, an alternative workflow B is chosen. In this path, there is a *photograph service*, followed sequentially by two other services namely *alert the police and owner service* and *log service*. *Monitoring service* is inserted, in parallel to the execution of these three services. Based on the monitored value of the *photograph service*, the next alert service in the queue for execution is replaced by another service, where the police and the owner are sent alert messages only through SMS. Similarly, the *log service*, is either included or eliminated based on the monitored performance value of the previous service.

Copyright ⓒ 2015 SERSC

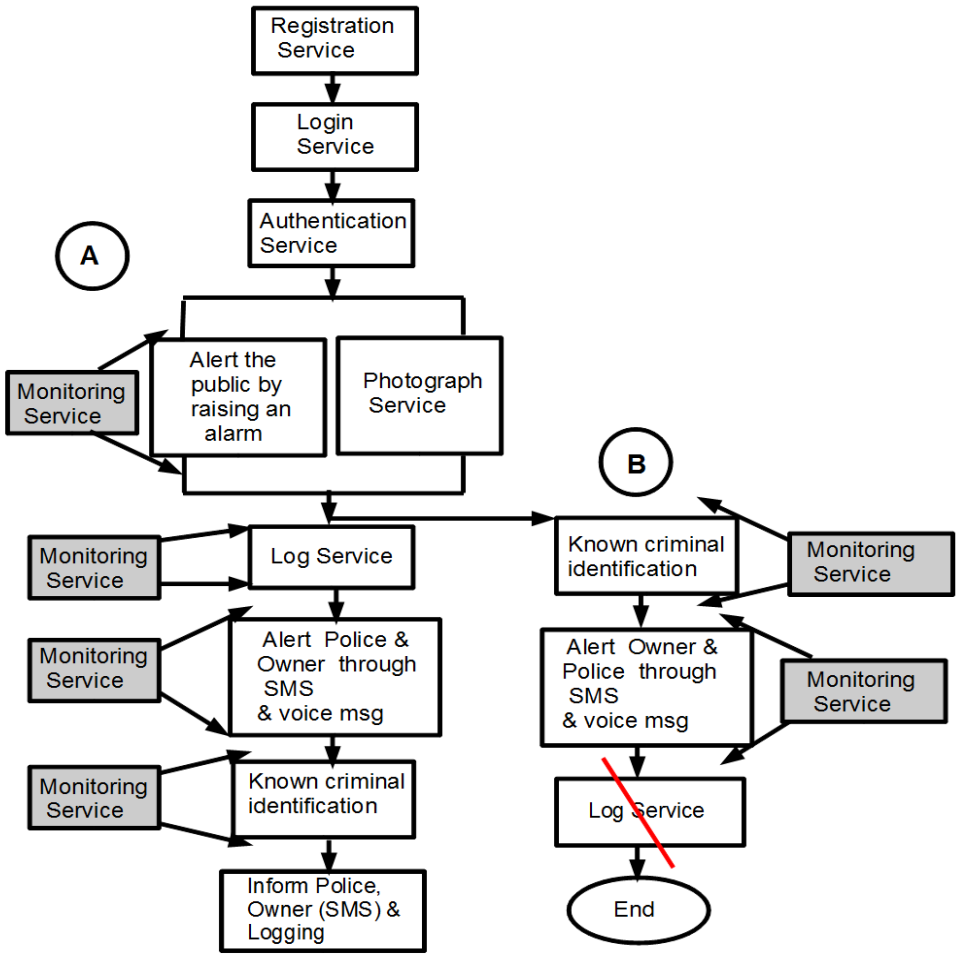

Figure 6. Level-Three Security During Dynamic Reconfiguration

Likewise, *Level-Three security* with dynamic reconfiguration is shown in Figure 6. Here, there are two execution paths namely, A, and B. Path A, is the normal execution of the *Level-Three Security*. In a normal scenario, the services in path A, are continuously monitored and the QoS violation is not predicted till the end of the application execution. On the other hand, if SLA violation is predicted after the parallel execution of the two services, workflow corresponding to path B shown in Figure 6 is substituted. In the new path, the first service is the *known criminal identification service*. The service to alert both the police and the owner simultaneously, by sending SMS and voice mail follows this. *Monitoring service* is inserted in parallel to the execution of these services. Based on the monitored value, decision is taken whether to execute the original alert service in the redirected workflow or replace that service with another, which has the feature of sending only the SMS. In Figure 6, the replacement of the *alert service* is not shown; rather, it shows the deletion of the *log service* based on the monitored value, in order to prevent the violation of QoS, specified in the global SLA.

To summarize, initially, the application starts its execution, with the available service composition that was decided during design time. After the prediction point, the *monitoring service* is inserted in parallel to all the services till the end of the application execution. Based on the monitored value, whenever any violation of QoS, specified in the global SLA is predicted, the existing workflow is replaced by a new appropriate workflow. The services in this new redirected workflow are also continuously monitored. Whenever any potential violation of QoS is foreseen in the new redirected workflow, then the next service in the execution queue is replaced by another service that has the same

 Copyright ⓒ 2015 SERSC

functionality and better QoS. In some cases, if the next service to be executed is the log service and the delay in the execution time is anticipated, then the *log service* is skipped and the execution proceeds to the service that follows the *log service*. The extended monitoring and replacement of services even in that redirected path is the distinguishing feature of the SWEAR approach and this also has been captured as a pattern named I-RAILS.

6. Conclusions

Dynamic reconfiguration of services, in a service-oriented application, is achieved by the proposed approach *SWEAR* for the composite service. The steps involved in this approach, are also captured as a pattern named I-RAILS. This *SWEAR approach for the composite service* has been applied to the *Safe Building Security* system, that helps to provide increasing levels of security to the building which uses this application. The Safe Building Security (SBS) system provides utmost security only when it responds quickly to the entry of criminals, by raising the alarm and by alerting the police, without any violation in the QoS specified in the global SLA of the composite service. Since, the proposed *SWEAR* approach, helps the SBS system, to respond quickly to the criminal entry, within the expected period of time, it helps to provide the increasing levels of security for the buildings that use this SBS system.

Further, the *SWEAR* approach for the composite web service, can be compared with the *dynamic composition of web services*.

SLA aware dynamic reconfiguration (SDR) achieved by the *SWEAR* approach, is likely to consume less time, for the entire service execution of a composite service, compared to the execution time of *SLA aware dynamic composition (SDC)* for the same set of services. The reason is that, there is a need to select the next service to be executed, at run-time, from the registry, for all the functionalities in the workflow, in the case of SLA aware dynamic composition. On the other hand, only whenever there is a need for the alternative service, the SDR selects the service from the registry.

A prediction point is first fixed, only after which the parallel monitoring of the service that is under execution is initiated. As the composite service execution, happens in the statically composed manner, till the prediction point, the searching time to find out either an alternate workflow or an alternative service in the reconfigured workflow, is entirely eliminated. Hence, the SDR achieved by SWEAR approach is likely to execute quickly, compared to the *SLA aware Dynamic Composition*.

The future plan is to employ the *SWEAR* approach on more case studies and do rigorous performance analysis. This will help, in gaining better insight with respect to the claimed reduction in the performance overhead, and the improved security levels, compared to the SLA aware dynamic composition.

References

[1]. X. Bai, J. Xie, B. Chen and S. Xiao, "DRESR: Dynamic Routing in Enterprise Service Bus", In Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE '07), **(2007)** October 24-26, Hong Kong, China.
[2]. A. Charfi and M. Mezini, "Ao4bpel: An Aspect-Oriented Extension to BPEL", World Wide Web, vol. 10, no. 3, **(2007),** pp. 309-344.
[3]. T. Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall PTR, **(2005)**.
[4]. R. Filman, T. Elrad and S. Clarke, "Aspect-Oriented Software Development", Addison-Wesley, **(2005)**.
[5]. L. Ramnivas, "AspectJ In Action: Practical Aspect-Oriented Programming", In Action Series, Manning Publications, **(2003)**.
[6]. P. Leitner, B. Wetzstein, D. Karastoyanova, W. Hummer, S. Dustdar1 and F. Leymann, "Preventing SLA Violations in Service Compositions Using Aspect-Based Fragment Substitution, Springer-Verlag Berlin Heidelberg, LNCS, vol. 6470, **(2010),** pp. 365-380.
[7]. http://windowsworkflowfoundation.eu/.
[8]. http://msdn.microsoft.com/en-us/library/vstudio/ms733615\%28v=vs.90\%29.aspx.

Copyright ⓒ 2015 SERSC

[9]. Y. Li, Y. Lu, Y. Yin, S. Deng and J. Yin, "Towards QoS-Based Dynamic Reconfiguration of SOA-Based Applications", In Proceedings of the IEEE Asia Pacific Services Computing Conference, IEEE Computer Society, **(2010),** pp. 107-114.
[10]. R. Schmelzer and T. VanDersypen, "XML and Web Services Unleashed", Sams, Indianapolis, IN, USA, **(2002)**.
[11]. B. Weber, S. Rinderle and M. Reichert, "Change Patterns and Change Support Features in Process-Aware Information Systems", In Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE'07), John Krogstie, Guttorm Sindre, and Andreas Opdahl (Eds.). Springer-Verlag, Berlin, Heidelberg, **(2007).**
[12]. M. Fredj, N. Georgantas, V. Issarny and A. Zarras, "Dynamic Service Substitution in Service-Oriented Architectures. In Proceedings of the IEEE Congress on Services - Part I (SERVICES '08), IEEE Computer Society, **(2008)**, September 23-26, Beiging, China.
[13]. G. Canfora, M. D. Penta, R. Esposito and M. L. Villani, "QoS-Aware Replanning of Composite Web Services", In Proceedings of the IEEE International Conference on Web Services (ICWS '05), IEEE Computer Society, **(2005)** July 11 -15, Florida, USA.
[14]. W. Dong, "Dynamic Reconfiguration Method for Web Service Based on Policy", In Proceedings of the International Symposium on Electronic Commerce and Security (ISECS '08), IEEE Computer Society, **(2008)** August 3 -5, China.
[15]. W. T. Tsai, W. Song, R. Paul, Z. Cao and H. Huang, "Services-Oriented Dynamic Reconfiguration Framework for Dependable Distributed Computing", In Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC '04), **(2004)** September 28-30, Hong Kong.

 Copyright ⓒ 2015 SERSC