Service Oriented Product Lines - Managed Service Level Agreements for Better Quality of Service

Asanka Garusinghe, Indika Perera, Dulani Meedeniya

Abstract— **Service Oriented Architecture (SOA) is being used for developing service oriented applications as a set of business specific web services and gives more flexibility for the software development industry. However, systematic reusability for developing applications to fit customers' individual needs with high customization is significant to increase the productivity and reusability of such service oriented applications. Software Product Line (SPL) has the ability to prepare core sets of assets in an identifiable and reusable manner with manageable variable components. Thus, the combination of SOA and SPL has highlighted the term of Service Oriented Product Line (SOPL), which is used for increasing levels of flexibility and reusability. It helps to develop semantics of variability over identified service components. Likewise, Quality of Service (QoS) attributes play an important role in selection of web services in a SOA environment. Service Level Agreements (SLAs) provide the mechanism with a specification of the verifiable Quality attributes in web services. In this paper, we present our implementation approach of SOPL to manage Service Level Agreements (SLAs) in SOPL environments by monitoring Quality of Service (QoS) attributes in bundles of web service components. The design and development of service bundles for representing core sets of assets in SOPL are followed by the initial feature based analysis and identification of service components. The management of SLAs is handled by detecting the deviation between actual and acceptable predefined QoS metrics values in previously analysed web service components via Web Service Level Agreement (WSLA) language specified templates. The case study based evaluation results indicate the usefulness of research contribution.**

Keywords— **service oriented product lines, quality of services, feature model, web service level agreements.**

I. INTRODUCTION

Service Oriented Architecture (SOA) has become advantageous architecture for productivity and flexible reusability. Software reuse is a key factor for companies that intend to increase productivity, improve software quality and reduce the cost and time to market. Therefore, instead of developing everything from scratch, it could be better to reuse some assets to fit specific customer needs that are common across the application. Software Product Line (SPL) is playing a crucial role of preparing core assets and making them more reusable by defining with identified commonalities and variabilities.

Today, with the increasing complexity and size of software, maintenance and reusability has become a challenging task [1]. Moreover, flexibility plays an important role for such challenges and SOA delivers valuable aspects of components based flexibility and reusability. The integration of SOA and SPL together makes Service Oriented Product Lines (SOPL), which associate with main engineering goals of SOA and SPL and increase the flexibility and reusability of the application components to overcome challenges. However, reasoning about how to combine both SOA and SPL concepts within variability implementation and technological concerns is an important research area.

Researchers have already discovered the possibilities of combining such approaches in the software development process ([2]-[4]). However, they do not deliver actual level of implementations due to challenges consisting of implementation activities that become more complex, since it is important to understand concepts of variability and commonality analysis of SOPL to identify the right set of services. Web services are mainly used for satisfying the architectural behaviours of SOA. Therefore, we realize that for implementing SOPL based environment prepared with reusable core assets can be done by understanding the variability management among web service bundles. Service Level Agreements (SLAs) establish the contract between the service provider and the service consumer for meeting the conditioning requirement related to quality attributes in web services. The contracts establish standard specifications between two parties. Normal web services are following such standards to determine quality attributes shared between service providers and consumers. However, it is important to create a new process to manage and to monitor quality of service (QoS) in bundles of service components, which are following product line architectural specifications. Thus, managing and monitoring SLAs for web service bundles to support runtime variations in SOPL environment, becomes a challenging task.

In order to understand aforementioned key challenges, this paper provides the implementation approach of designing core assets in SOPL environment with the bundles of web services and introducing management process of SLAs by detecting the deviation between actual and acceptable pre-defined Quality of Service (QoS) attributes in those bundles. We also describe a case study that we have followed to implement SOPL platform. Within the case study, we have defined feature types and a feature view model for those feature types. That model represents the significant analysis of structural implementation of SOPL since it gives a complete picture of variability of service architecture and it is necessary to have such a model to understand the full view of dependencies of variability [3]. It suggests how to identify the commonalities and variabilities among distributed service component bundles in an application with a single view model. Moreover, Web Service Level Agreement (WSLA)

Manuscript received on 11th December, 2016. Recommended by Dr. T.M.H.A.Usoof on 17th August, 2017. This paper is an extended version of the paper, "Managing Service Level Agreements in Service Oriented Product Lines" presented on ICTer2016 Conference.

Asanka Garusinghe, Indika Perera and Dulani Meedeniya are from Department of Computer Science & Engineering, Faculty of Engineering, University of Moratuwa, Sri Lanka. (asaindunil@gmail.com, indika@cse.mrt.ac.lk, dulanim@cse.mrt.ac.lk)

language specified templates have been used for managing SLAs. This would help to understand some weakness points in different service components that are managed by the service providers. In that case, SLAs must be managed for such service providers for providing real QoS guaranteed metrics to the consumer of web service bundles [4].

The rest of the paper is organized as follows: Section II describes the background study for this research. Section III presents related work and similar studies. Section IV describes the case study, which is undertaken to support the research. Section V demonstrates our design and implementation approach. Section VI evaluates the results of implemented approach. Section VII interprets those results with the qualities and the reasoning of the implementation. Finally, section VIII concludes the paper and outlines our future work.

II. Background

This section covers the overview of key terms of architectural specifications and analysis related to our work. Since correct understanding of the background of SOPL is important, this section presents related literature and background research.

A. *Maintaining the Integrity of the Specifications*

Services in a SOA typically require the following characteristics in order to make it useful ([5], [6]) such as coarse-grained interfaces, loosely coupled, self- contained – modular, interoperable and location-transparent. On the other hand, the concept of a SPL is a paradigm that introduces product families leading to a great decrease in time, cost and effort in producing each member of the product family. We achieved this by reusing proper assets produced or merely used by the SPL [7]. SPL practices are focused on the planned structure and intentional reuse of components to achieve a number of benefits. SPL practices have a well-defined methodology to represent precisely the domain of the software. The maintenance of variation points in order to facilitate the exploitation of commonality and the management of variability among software features is a key principle of SPL. The authors of [8] describe the positive economic impacts of SPL adoption such as decrease the cost of developing software, reduce the time-to-market for software products, and increase the overall quality of software artefacts within adopting organizations. Organizations must consider how software can be made cheaper, quicker, and of higher quality in order to remain competitive. Despite initial barriers to adopting SPL practices, new methods continue to be devised and developed to enable SPL adoption into mainstream software development practices.

B. *Combination of SOA and SPL*

Medeiros et al [9] propose an approach to service-oriented product line architectures that combines SPL and SOA to achieve high customization for planned and systematic reuse to reach the desired benefits. Since there is a lack of variability management in SOA, product lines oriented services are becoming a popular method to handle case based services and variability of service-based systems [4]. The research presented in [10], introduces a line-oriented product with the development of service approach as well as the research in [11] discusses the context of service-oriented software product lines regarding change management. According to their work, we have stepped up to introduce the efficient implementation to prove their theories. As we have discussed in Section I, feature modelling can be used to do significant analysis of structural implementation in SOPL environment. Therefore, we follow some standard and guidance which can be applied for the context of SOA that has already been given in [12].

SPL implementations are usually internal closed systems. However, it is necessary to identify the service components while integrating both SPL and SOA with single domain. The linkage between them would clearly state with well-known proper model designs. According to presented work in [13], they are the feature models of product lines and the business process models in service orientation. The proper implementation of SOPL would lead to define variabilities and commonalities inside identified product features of individual service components.

C. *Architectural Styles and Frameworks of SOPL*

Traditional SPL approaches do not consider aspects of dynamic service provision. The work discussed in [10] clearly illustrates the idea with activity is being executed with directed flows, in which SOPL achieving corresponding analysis and management techniques. Therefore, we believe that architectural perspective of SOPL implementation is increasing the productivity of service providers if they are following such followed directed flows.

Reusing services can be improved by creating a product line that satisfies several variability requirements from different customer applications [18]. In this paper, we are following such variability management techniques to identify specific service components accurately. In Section V, we elaborate our approach of identification of service components with a case study.

In addition to aforementioned background study of SOPL architectural behaviours and implementations, we thought to present some frameworks those are actually most applicable for management of SLA. While authors of [4] are integrating the similar idea of [10], they propose a QoS-aware framework that provides automated runtime support for service discovery, negotiation, monitoring, and service provider rating, which lets consumers handle recovery from SLA violations, service failures, and runtime environment limitations by renegotiating and substituting problematic services.

QoS-aware framework facilitates system integration and deployment. Through this framework, dynamic-service consumers and providers supply the broker with templates specifying strategies for the SOPL services. In this paper, we present our implementation with a technique of proposed SLA management for better Quality of Service.

D. *Web Service Level Agreements (WSLA)*

In this paper, the process of SLA management takes place with designed SLA templates via WSLA. IBM introduced this standard language specification to define SLA documents [16]. A WSLA maintains an agreement between service provider and consumer to perform a service according to agreed-upon guarantees with the service parameters on the technical level. WS-Agreement is guaranteed for providing standard QoS specifications. Therefore, we design our templates based on WS-Agreement specification.

Aforementioned QoS attributes would refer to Service Level Objectives (SLOs) and Action guarantees in WSLA language specification [17]. Quality attributes play an important role in the selection of services in a SOA environment. SLA provide the majority of characteristics of verifiable QoS and will ensure that services are provided with several quality attributes such as high availability, security, performance and other measurable qualities [15]. Fig. 1 shows a conceptual architecture for SLA management process between two parties, which depicts an idea behind proper SLA management.

Managing and Monitoring SLAs via WSLA language based specification would prepare the mechanism of processing QoS attributes in service based components of SOPL architectural environment.

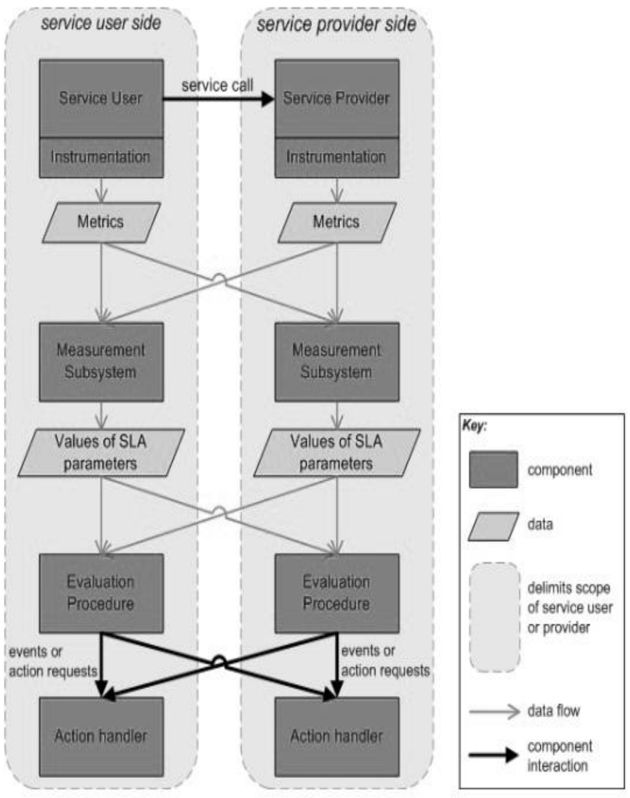


Fig. 1 Conceptual Architecture for Monitoring and Managing SLAs [15]

III. Related Work

Many researchers have described several challenges and proposed solutions for SOPL platforms indicating benefits by combining SOA and SPL. Though there are several related researches available to date, a proper way of implementing SOPL platform and managing quality of service components on such platforms is seldom found. There are two specific previous work, which indicate most related approaches for our work.

Heberth et al, have followed technology assessment techniques while implementing core assets in SOPL environment. They have presented some development practices that can be shared with the implementations of core set of assets and managed modularity with service level of bundles [18]. This work reflected on the definition of an approach to implementing core assets in a service-oriented product line. It has been pointed to the concepts of SPL and SOA and includes support for decision variability mechanisms applied under technology to control the life cycle of core assets. Their approach was the result of an investigation of how the current approach mentioned above in the problem statement, and resulting the assessment for combined with the mechanism of variability to achieve its objectives. They proposed their approach, which is built on top of SOPL environment. The approach has used OSGI as main technology and presented how variability mechanism can be done within OSGI. However, the presented implementation scope is not sufficient to show proper core assets management within service components and for managing better QoS attributes inside them. Our work reaches beyond the boundaries of the existing implementation which has been done in [18]. Section V denotes detailed explanation of our implementation approach. Though our work is related with the work in [18], the implementation for managing SLA specifications for better QoS in the corresponding platform is adding a significant factor for our work. Darwish and others have proposed an approach to monitor quality of web services via SLAs in [19]. They present some qualities behind web services that should be able to monitor while extracting service level objectives (SLOs) in SLA templates. By adding monitoring process to our approach, the presented idea has been taken with some improvements since they have proposed only the prototype for monitoring qualities on web services. Even there was not any implementation in [19], we show our implementation in this paper described in section V.

IV. Case Study

In order to illustrate the necessity of our approach with SLA management in SOPL environment, we use a case study related to a travel agency. Specific sample agency can be selected for satisfying this implementation with an approach of product line. Most of the travel agencies provide various types of reservations for customers who are willing to take the services from them. For example, Transport (Vehicle), Hotel and Airline reservations can be listed down. When customer makes one of the reservations, the payment component is involved in (credit card, debit card or cash).

We are taking this case study to deliver travel agency services as a corresponded service provider. It can be defined as SOPL with variability implementation according to customer specific needs. The customer may need only the selected reservation types and selected payment types. Service provider only needs to deliver whatever service requirements with identified commonalities and variabilities. Simple travel agency will provide itineraries to the customers according to selected reservations. Large agency will provide itineraries for Airline reservations with the accommodations such as hotel reservations. Likewise, service provider who is acting as a travel agency party can be highlighted different service components with highly customizations of variability.

There are also acceptable quality metric values to determine whether the agency is providing effective services to the customers. Customer and the Agency (Provider) should need to establish contract to measure QoS metric values between two parties. For example, customer will be satisfied for preparing reservation within specific timeframe without any delays from specified threshold value and response time of specific service bundle for processing number of user itineraries per request.

We present this case study to give high-level picture of our implementation and key techniques, which have been addressed for various challenges, faced by service providers and consumers when practicing high quality of service selections and performance. We are taking reservation component in detail for deriving our implementation approach of SOPL. Nevertheless, payment and notification bundles are playing mandatory key role to establish end-to-end service and client interactions.

V. Design and Implementation

Our design and implementation approach of this work mainly focuses on two specific implementations. One is for designing and implementing service components, which satisfies core assets in SOPL environmental platform based on aforementioned case study. Second is for implementing the mechanism to manage Service Level Agreements with pre-defined QoS attributes in each service bundles. There are few literatures ([3], [12], [19]), which describe several semantic analysis and implementations of service providers and consumers with the combination of SOA and SPL. However, none of them describes how to determine Service Level Agreements (SLAs) specifications to measure the qualities of service components and how accurately providing their services to customers. For understanding commonalities and variabilities, it is better to introduce component based analysis and identification of service components to map identified variants in different levels of granularity. In order to achieve such levels of activities, components can be depicted as features. The Open Services Gateway Initiative (OSGI) defines the architectural framework for deploying application with based on modularity aspects. Therefore, we decide to use OSGI to switch between one or more features and manage reusable assets by accepting component based granularity in SOPL platform. For this implementation, we have gone through both JAVA and Microsoft .NET as target environmental platforms.

The major responsibility of implementing SOPL environment falls under service provider perspective. In that case, our approach slightly defines roles, which can be used for the definition of the parties who are responsible with. We believe that developers and the domain architects should represent responsible parties. Therefore, Fig. 2 shows some steps of our design approach, which should be taken by developers and domain architects.

Goal: To design and implement the core sets of assets in SOPL environmental platform for managing Service Level Agreements (SLAs) with respect to Quality of Service attributes in each service components.

A. Feature Analysis

In this activity, relevant party who are representing either one of aforementioned roles must take a step to analyse features based on the standard criteria in product line architecture. For gaining a good high-level picture of variabilities in each service components, it is better to have a standard model to determine them in advance [3].

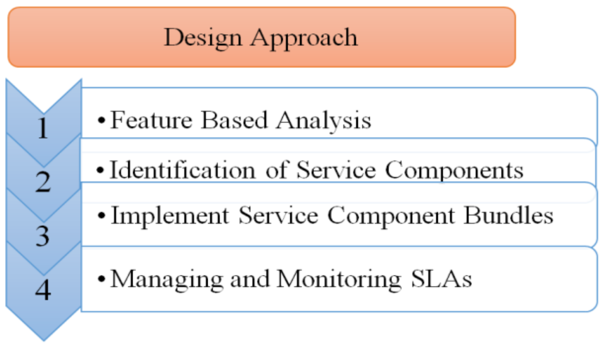

Design Approach
1
2
3
4
• Feature Based Analysis
• Identification of Service Components
• Implement Service Component Bundles
• Managing and Monitoring SLAs

Fig. 2 Steps of Design approach of SOPL

Feature modelling technique comes under feature based analysis activity. Here we specify it before implementing service components since it gives flexible approach for making service based analysis easily. As described in the case study, we apply feature-based analysis and modelling for sample travel agency. The same set of case studies have been followed by Heberth et al in [18] according to their assessment of making core assets in SOPL. In our approach, some refinements for this study has been added while implementing key features and structures. Fig. 3 shows the designed feature model for the case study, which is evaluated further.

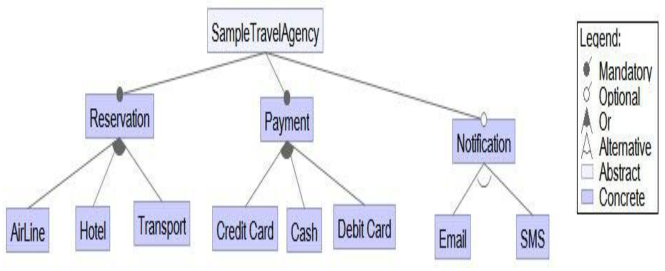


Fig. 3 Feature Model for Sample Travel Agency Case Study System

B. Identification of Service Components

Service components represent the assets of Service Oriented Product Line (SOPL) environment that can be managed in dynamic modularity. Each asset is named under defined feature types and is analysed those specified features in feature analysis section. Based on the specified feature model, we are gathering common set of features and making decision of what types of variation points can be described while identifying key service components. The selection of variability mechanisms should be based on a systematic analysis of specific context where the mechanism will actually be used. And also variability is supporting to detect corresponded variation point and the binding time. In our case study, we present some of identified variation points with specified decisions. Thus, for implementing such identified variation points with service components, we decide to take an appropriate service technology, OSGI framework.

Fig. 4 shows those identified service components and their variation points. Here, service bundle of Reservation is implemented under IReservation interface, which is acting as a variation point among Airline, Hotel and RetalCar (Transport) components of variants. During service invocation, those service components are trying to accomplish their goals in dynamic binding environment with different consumer parties who have requested different customization from specific service provider who has already published SOPL environment.

Fig. 5 shows a structural view of reservation variation point, which is selected relevant bundles for reservation according to customer specific need while consuming reservation bundle of services in SOPL environment. The service provider (travel agency) makes the full control of these variation points. Customer can consume selected agency bundle via implemented interfaces. Loosely coupled components increase the dependency management with high customizations according to customer specific needs.

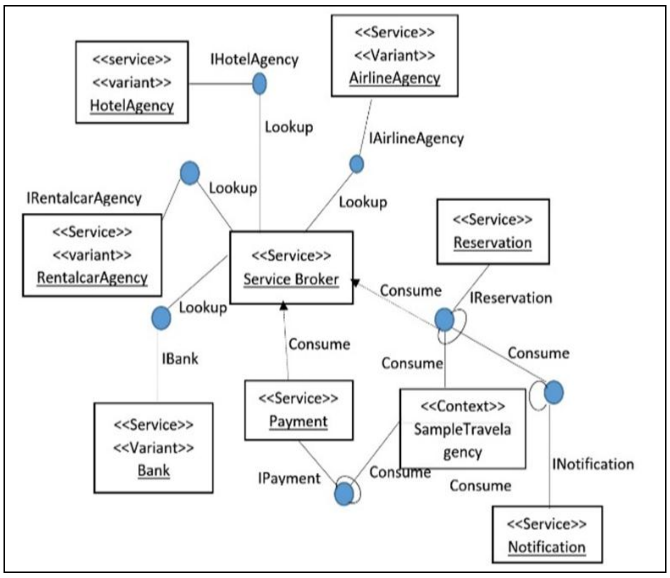


Fig. 4 Identified Service Components with the Variation Points

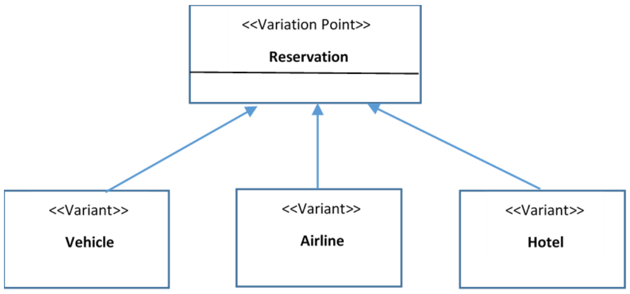


Fig. 5 Structural View of Variation Point

C. Implement Service Component Bundles

Once service components and their variation points are identified, this paper presents our next phase, which is the development of service components with dynamic modularity. OSGI Framework is well known architectural framework for designing such modular based application components as mentioned before. In this scenario, we are using OSGI to create bundles of web services for achieving SOPL platform. For this work, we have stepped forward for implementing this modularity in both JAVA and .NET platforms.

OSGI.NET was introduced for Microsoft .NET platform for supporting languages like C#. Eclipse and Visual Studio are two specific integrated development environments (IDEs), which were used throughout this implementation. Each component bundles can be activated via activator class, which is located under each service bundle. According to customer requirement, service provider can activate specific bundles like variability implementation. OSGI modularity has provided significant amount of flexibility for developers to design bundles of services. As mentioned in feature model, service bundles are representing each feature. Moreover, each bundle is communicating with defined variation points and analysed variability implementation. Fig. 6 shows the overview picture of source classes and interface declarations for reservation bundle in the solution explorer as such implementing each service bundle of Sample Travel Agency SOPL.

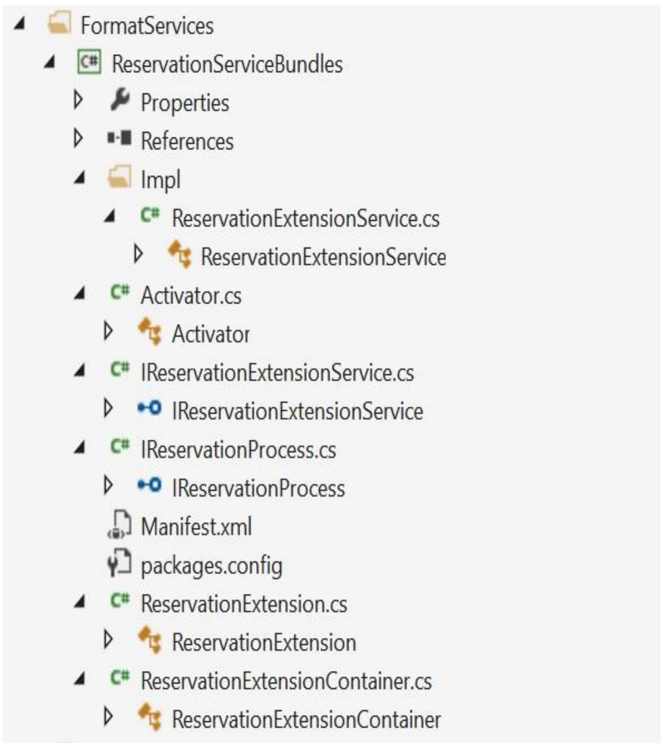

```
FormatServices
  ReservationServiceBundles
    Properties
    References
    Impl
      ReservationExtensionService.cs
        ReservationExtensionService
    Activator.cs
      Activator
    IReservationExtensionService.cs
      IReservationExtensionService
    IReservationProcess.cs
      IReservationProcess
    Manifest.xml
    packages.config
    ReservationExtension.cs
      ReservationExtension
    ReservationExtensionContainer.cs
      ReservationExtensionContainer
```

Fig. 6 Source code structure of Reservation Service Bundle

D. Managing and Monitoring SLAs

Aforementioned activities depict the implementation approach for SOPL with the steps of analysing features and identifying corresponding service components. Each component is managed like service bundles and they have been created with OSGI framework. For managing SLAs we have initialized expected and pre-defined quality attributes with the standard WSLA language specified templates. Fig. 7 shows the arrangement of WSLA templates for managing SLAs. Fig.8 and Fig.9 shows sample templates which is being executed.

The mechanism is integrated with both consumer (end-user) and service provider parties. In the service provider end, SOPL architecture contains implemented service bundles that are representing core assets and variability components. We maintain a shared WSLA language specific template among implemented service components and define specific Quality of Service attribute value metrics in that template. Once consumer consumes specific service component, it loads WS-Agreement specifications defined in the WSLA template and

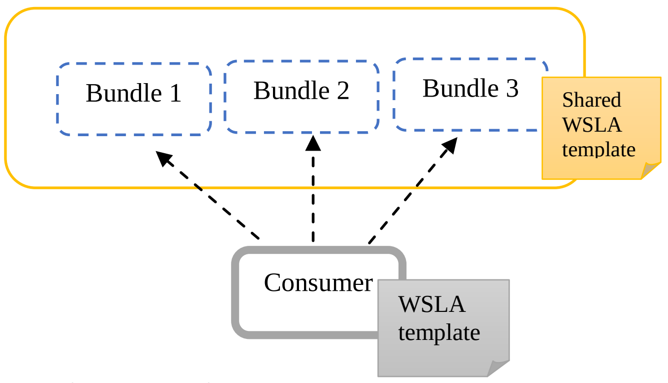


Fig. 7 Defining process of WSLA templates inside the SOPL environment

SLOs, which have guaranteed QoS attributes values. Fig 10 shows the mechanism of processing WSLA template in each service bundle. For evaluating corresponding QoS based metric value, the activator of the bundle was used. It evaluates XML based WS-Agreement document with consumer specified WSLA template. We are going to describe the template definition in going forward.

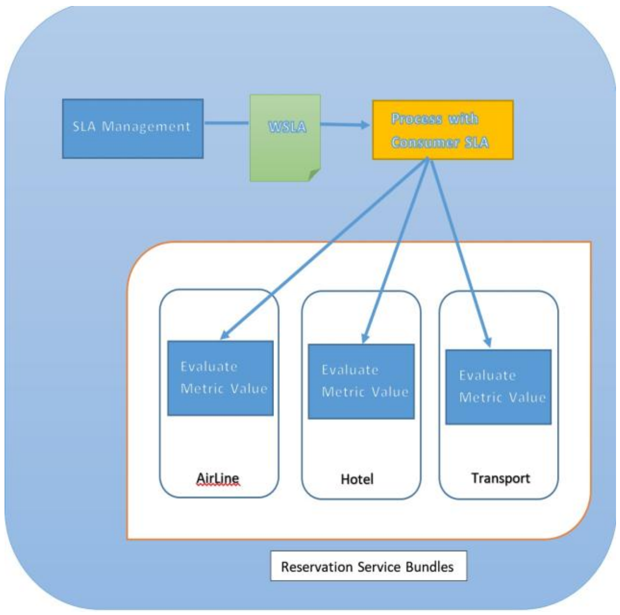
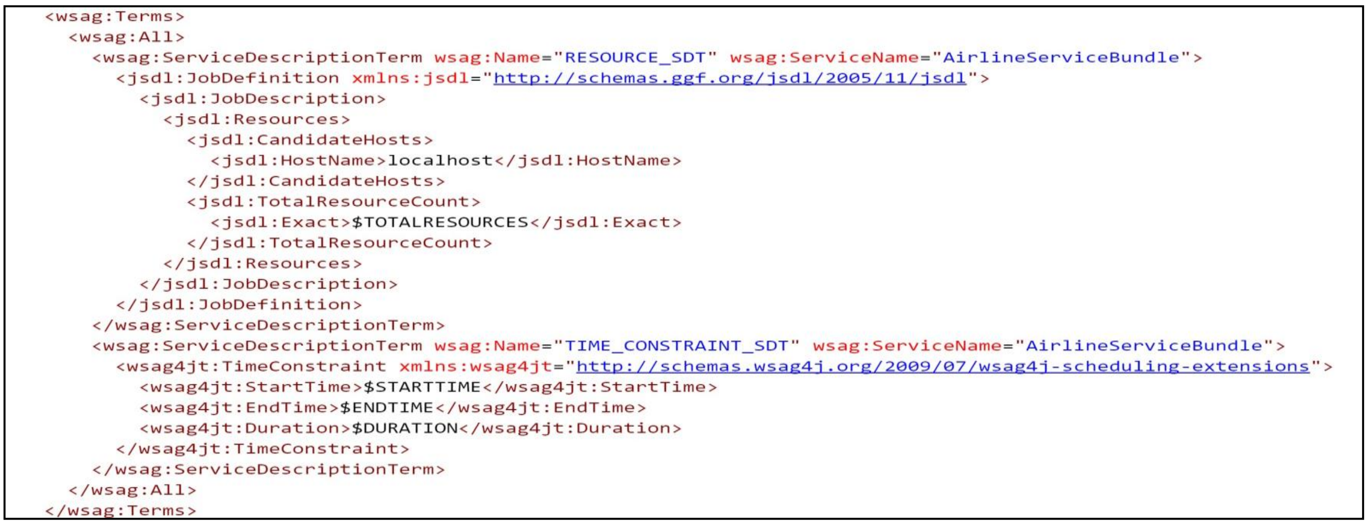


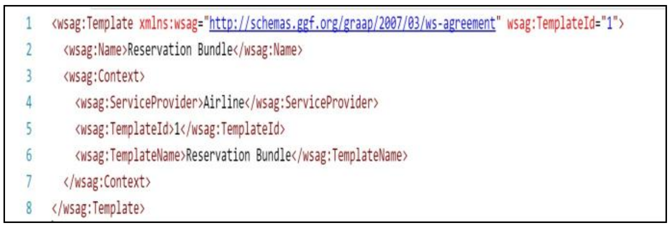
Fig. 10 Processing shared WSLA template inside service bundles



```
<wsag:Terms>
  <wsag:All>
    <wsag:ServiceDescriptionTerm wsag:Name="RESOURCE_SDT" wsag:ServiceName="AirlineServiceBundle">
      <jsdl:JobDefinition xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
        <jsdl:JobDescription>
          <jsdl:Resources>
            <jsdl:CandidateHosts>
              <jsdl:HostName>localhost</jsdl:HostName>
            </jsdl:CandidateHosts>
            <jsdl:TotalResourceCount>
              <jsdl:Exact>$TOTALRESOURCES</jsdl:Exact>
            </jsdl:TotalResourceCount>
          </jsdl:Resources>
        </jsdl:JobDescription>
      </jsdl:JobDefinition>
    </wsag:ServiceDescriptionTerm>
    <wsag:ServiceDescriptionTerm wsag:Name="TIME_CONSTRAINT_SDT" wsag:ServiceName="AirlineServiceBundle">
      <wsag4jt:TimeConstraint xmlns:wsag4jt="http://schemas.wsag4j.org/2009/07/wsag4j-scheduling-extensions">
        <wsag4jt:StartTime>$STARTTIME</wsag4jt:StartTime>
        <wsag4jt:EndTime>$ENDTIME</wsag4jt:EndTime>
        <wsag4jt:Duration>$DURATION</wsag4jt:Duration>
      </wsag4jt:TimeConstraint>
    </wsag:ServiceDescriptionTerm>
  </wsag:All>
</wsag:Terms>
```

Fig. 8 WS-Agreement specified template defined in reservation bundle

SLA template Creating and designing such formatted XML files is a challenging task without mitigating proper definitions of SLA parameters. Therefore, we use WSAG4J (WS-Agreement for Java) library to prepare WS-Agreement specified templates. For .NET implementation, we use customized WSLA templates in order to develop proper mapping of middleware to understand those templates. Fig. 8 and Fig.9 depicts WS-Agreement specified template, which were designed under WSG4J library.



```
<wsag:Template xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement" wsag:TemplateId="1">
  <wsag:Name>Reservation Bundle</wsag:Name>
  <wsag:Context>
    <wsag:ServiceProvider>Airline</wsag:ServiceProvider>
    <wsag:TemplateId>1</wsag:TemplateId>
    <wsag:TemplateName>Reservation Bundle</wsag:TemplateName>
  </wsag:Context>
</wsag:Template>
```

Fig. 9 WS-Agreement template main scope

<wsag:Template> is used to define the main scope of the template in order to *<wsag:Term>* is responsible for extensible attribute to define service specification with corresponding SLA parameters.

We can define specific timeframe for handling extraction of QoS values in the defined parameter section. According to Fig. 10 we process WS-Agreement specified template or designed WSLA based template with the defined template in consumer end.

Before finding the deviations of actual and acceptable pre-defined Quality of Service (QoS) attributes, which are defined on WSLA templates, each template has been loaded into bundle activators and while consuming the bundle of services, it extracts SLO values and verifies defined QoS attributes and find the deviations of such attributes between shared WSLA template and the consumer based WSLA template.

Fig. 11 depicts the flowchart showing the actual flow of execution of SLA management. Here, Service Level

Objective (SLO) values play a huge role to be a conceptual elements of an SLA for which SLA Parameters and the corresponding QoS Metrics can be defined. By extracting those SLO values, we can compute those values with pre-defined conditions and predicate values. To detect the variation between pre-defined and actual values, we located a specific process implementation to compare templates and the action guarantee defines a commitment to perform a particular activity if a given precondition is met. The precondition is defined as an expression on predicates that refer to the SLA Parameters defined in the Service Definition section of the SLA.

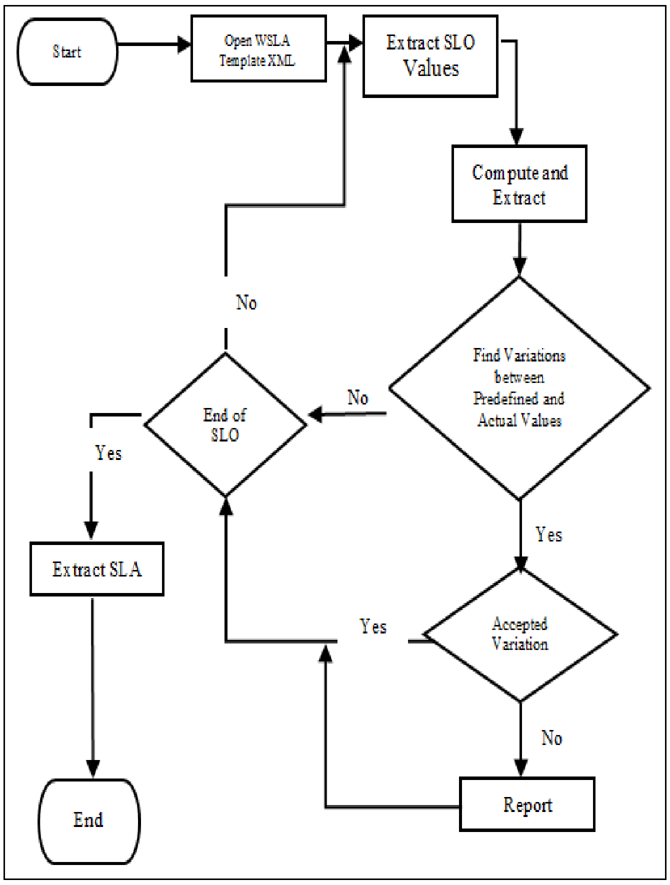
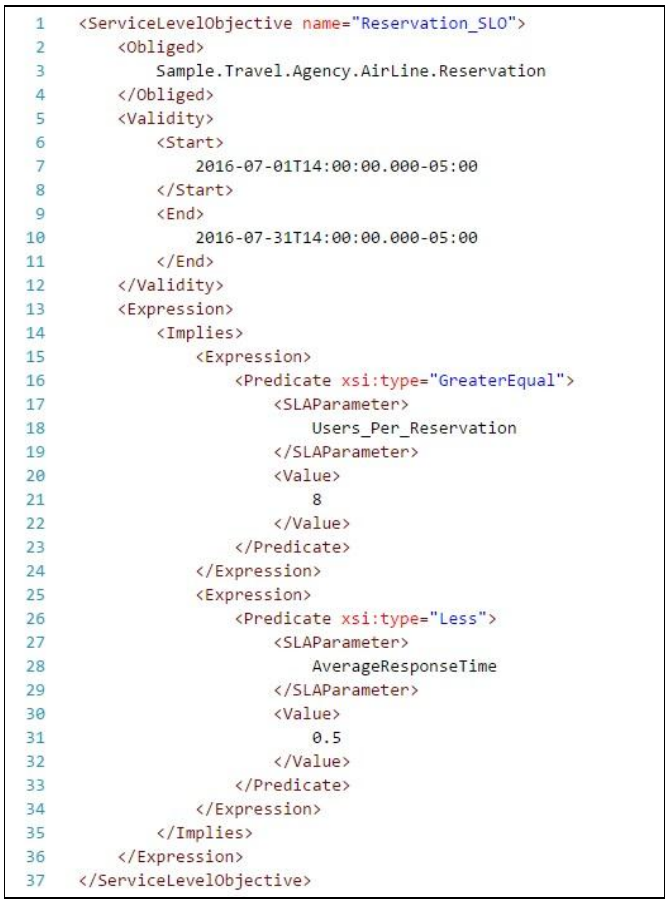


Fig. 11 Flowchart of Managing SLAs

SLA parameters [20] are properties of a service object. In our approach, we define some SLA parameters to specify QoS metric values, which are used to monitor SLAs, established between SOPL environment and customer. Each SLA parameter has a name, type and unit. SLA parameters are computed from metrics, which either define how a value is to be computed from other metrics or describe how it is measured. For this purpose, a metric either defines a function that can use other metrics as operands or it has a measurement directive that describes how the metric's value should be measured. Since SLA parameters are the entities that are surfaced by a Measurement Directive to a Condition Evaluation Service, we define AverageResponseTime and User per reservations metric value to WSLA specified template related to reservation bundle. Fig. 12 shows SLO specification of part of WSLA template. Here the minimum users per reservation is limited to 8 (the minimum value should be at least 8 to continue the service as expected). However, if that pre-condition is not met, the violation predicate has to be triggered. In our approach, we have separated service component bundles that are sharing template values. However, each bundle has its own QoS specified values. In this case, average response time of service bundle with specific time frame (0.5 ms).

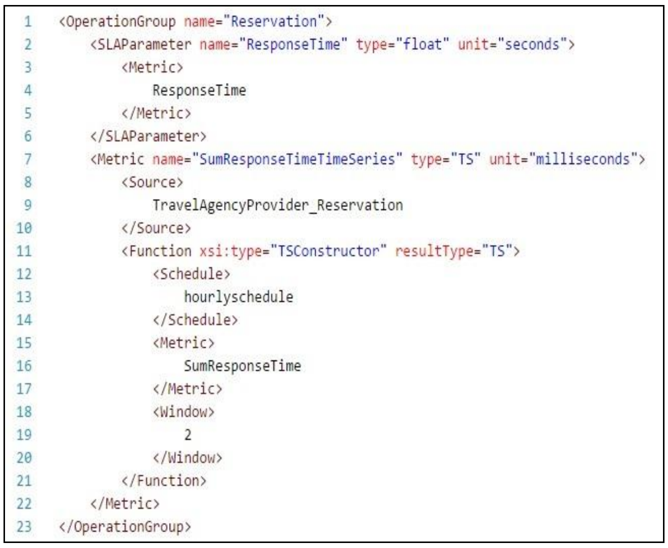


```
<ServiceLevelObjective name="Reservation_SLO">
    <Obliged>
        Sample.Travel.Agency.AirLine.Reservation
    </Obliged>
    <Validity>
        <Start>
            2016-07-01T14:00:00.000-05:00
        </Start>
        <End>
            2016-07-31T14:00:00.000-05:00
        </End>
    </Validity>
    <Expression>
        <Implies>
            <Expression>
                <Predicate xsi:type="GreaterEqual">
                    <SLAParameter>
                        Users_Per_Reservation
                    </SLAParameter>
                    <Value>
                        8
                    </Value>
                </Predicate>
            </Expression>
            <Expression>
                <Predicate xsi:type="Less">
                    <SLAParameter>
                        AverageResponseTime
                    </SLAParameter>
                    <Value>
                        0.5
                    </Value>
                </Predicate>
            </Expression>
        </Implies>
    </Expression>
</ServiceLevelObjective>
```

Fig. 12 Service Level Objective Scope

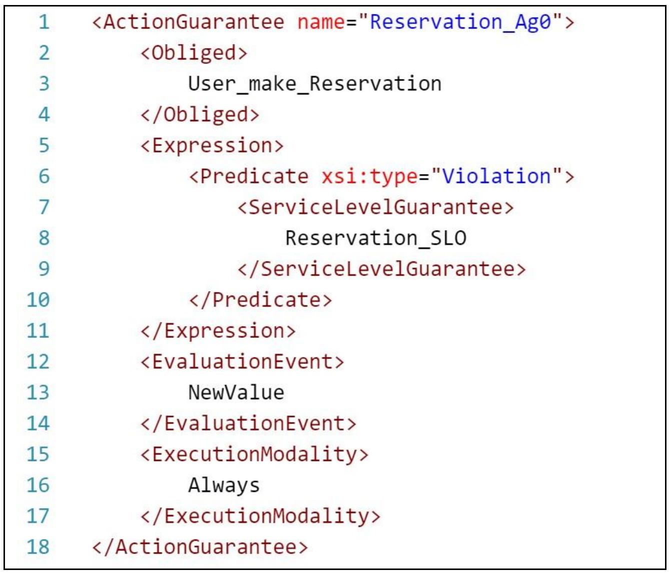
The purpose of a metric is to define how to measure or compute a value. Besides a name, a type and a unit, it contains either a function or a measurement directive and a definition of the party that is in charge of computing this value. Fig. 13 shows an example of composite metric "SumResponseTimeTimeSeries", containing a function "TSConstrctor". Thus, QoS value should be satisfied with pre-defined condition value while processing hourly schedule to monitor value changes of average response time defined as SLA parameter. Expression tag defined in SLO section is evaluated according to specific event. That event is created under action guarantee, which is shown in Fig. 14.

Managing and Monitoring SLAs via WSLA language based specification would prepare the mechanism of processing service based components lays on SOPL architectural environment. Therefore, we design these SLA templates with specified WSLA standards for defining Quality of Services (QoS) attributes within that. For reservation section, it is important to determine, which characteristic should be attached to SLA template. Therefore, operation is linked SLA with the service it refers to. Fig.13 depicts designed OperationGroup for managing SLA specification related to reservation. While describing our techniques that we are going to follow throughout this work, we believe to show sample WSLA template, which has specific action guarantee based on corresponding categories of sections.



```
<OperationGroup name="Reservation">
    <SLAParameter name="ResponseTime" type="float" unit="seconds">
        <Metric>
            ResponseTime
        </Metric>
    </SLAParameter>
    <Metric name="SumResponseTimeTimeSeries" type="TS" unit="milliseconds">
        <Source>
            TravelAgencyProvider_Reservation
        </Source>
        <Function xsi:type="TSConstructor" resultType="TS">
            <Schedule>
                hourlyschedule
            </Schedule>
            <Metric>
                SumResponseTime
            </Metric>
            <Window>
                2
            </Window>
        </Function>
    </Metric>
</OperationGroup>
```

Fig. 13 Defined operation group for response time



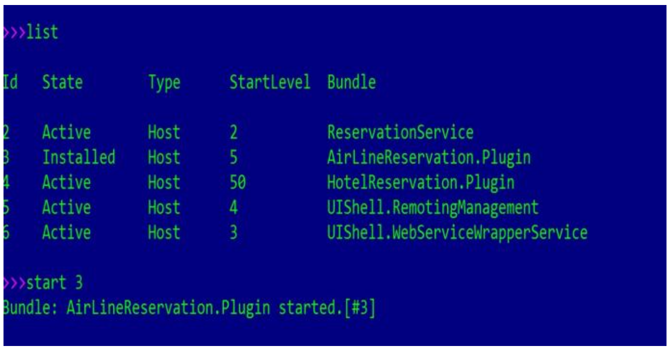
```
<ActionGuarantee name="Reservation_Ag0">
    <Obliged>
        User_make_Reservation
    </Obliged>
    <Expression>
        <Predicate xsi:type="Violation">
            <ServiceLevelGuarantee>
                Reservation_SLO
            </ServiceLevelGuarantee>
        </Predicate>
    </Expression>
    <EvaluationEvent>
        NewValue
    </EvaluationEvent>
    <ExecutionModality>
        Always
    </ExecutionModality>
</ActionGuarantee>
```

Fig. 14 Action Guarantee Scope of SLA Template Design

An action guarantee expresses a commitment to perform a particular activity if a given precondition is met. The precondition is defined as an expression on predicates that refer to the SLA Parameters defined in the Service Definition section of the SLA. Any party can be the obliged of this type of guarantee [21]. This particularly includes the supporting parties of the contract and the service customer. Fig. 14 depicts such action guarantee section with a predicate of SLA violation detection. Once a violation has been detected it will notify the corresponding service method.

VI. Results

Since our approach targets both Java and .NET based platforms, we are running implemented service component bundles with managed console that supports for OSGI Framework. Fig. 15 shows the list of activated and installed reservation bundles in runtime modularity. Specific service components can be added dynamically for arranging core assets and the component based assessment of variability in SOPL environment from specific bundle id. When service provider’s perspective, this enables the previllege to expose customized service components for consumer parties. If they are mainly focusing on dynamic modularity with reusable service components, OSGI implementation is the most relevant approach. The main idea is how we expose service components with identified variability management inside service provider. Here, travel agency’s reservation components are directed with specific commonalities and variabilities. For example, Airline reservation can be mandatory while Hotel reservation is not.



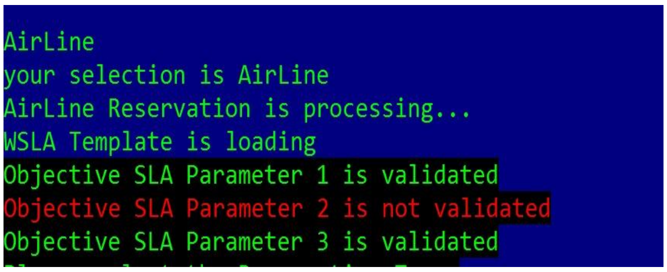
```
>>>list

Id  State      Type    StartLevel  Bundle

2   Active     Host    2           ReservationService
3   Installed  Host    5           AirLineReservation.Plugin
4   Active     Host    50          HotelReservation.Plugin
5   Active     Host    4           UIShell.RemotingManagement
6   Active     Host    3           UIShell.WebServiceWrapperService

>>>start 3
Bundle: AirLineReservation.Plugin started.[#3]
```

Fig. 15 List of reservation bundles

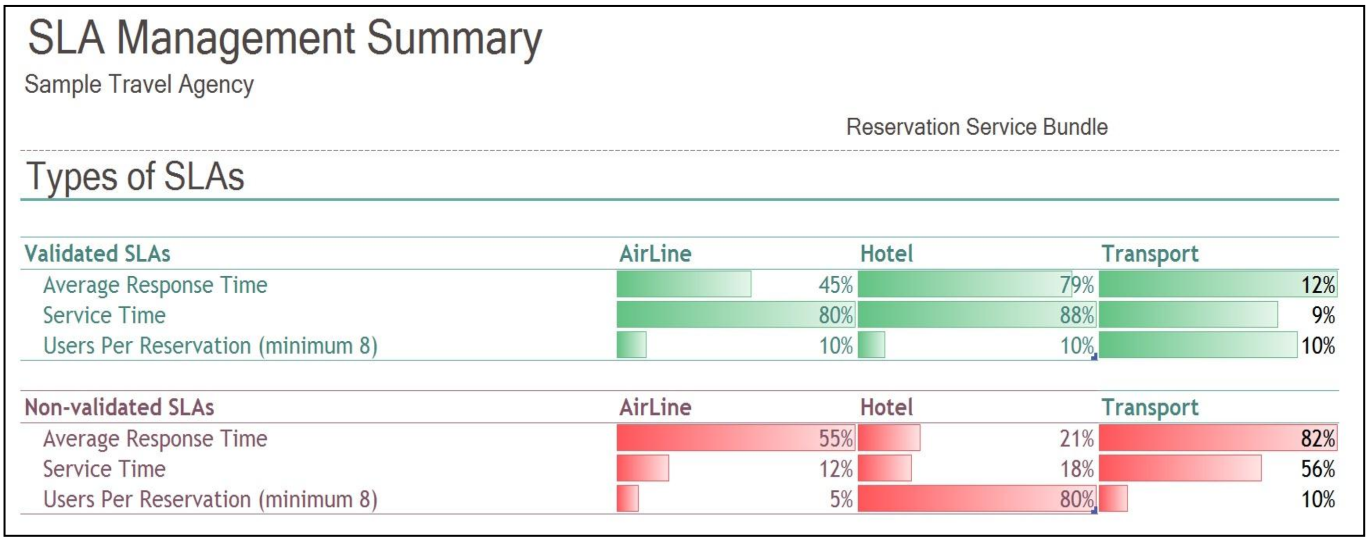


```
AirLine
your selection is AirLine
AirLine Reservation is processing...
WSLA Template is loading
Objective SLA Parameter 1 is validated
Objective SLA Parameter 2 is not validated
Objective SLA Parameter 3 is validated
```

Fig. 16 Processing SLA Parameters Airline component

Now, Airline reservation is selected as a component. While activating that bundle, corresponding WSLA template has been loaded to start the process of SLA management. Currently, we are selecting predefined SLA templates for activated bundle to do further SLA management process according to defined QoS attributes inside templates. Fig. 16 shows sample results with the deviations between actual and pre-defined QoS values defined like SLA parameter. That process is integrated with WSLA processing method which has been discussed earlier. If SLA parameter is validated, the evaluation expression must be satisfied with given predicate value. If SLA parameter is not validated, the predicate value is violated. Those results are logged with text files as well as one of data sources like MS Access or MS Excel. Those data will be used for further analysis to generate statistical presentation templates. Section VII will show how far we came up with results for proper evaluation.

Fig. 17 depicts the overall summary of SLA management for selected reservation service bundle. We validated three types of metric values in three reservation bundles. We took



SLA Management Summary

Sample Travel Agency

Reservation Service Bundle

Types of SLAs

Validated SLAs	AirLine	Hotel	Transport
Average Response Time	45%	79%	12%
Service Time	80%	88%	9%
Users Per Reservation (minimum 8)	10%	10%	10%

Non-validated SLAs	AirLine	Hotel	Transport
Average Response Time	55%	21%	82%
Service Time	12%	18%	56%
Users Per Reservation (minimum 8)	5%	80%	10%

Fig. 17 Summary for SLA Management

six different service requests from consumer-end and validated QoS values with predicate definitions. It is important to get quality of service distributions in each core bundle of web services to determine, which bundle has most violation rate and which bundle has less violation rate.

According to the summary, Airline reservation bundle has maximum level of violation rate 55% for the metric of average response time. Likewise, minimum violation rate 5% for the metric of number of users per reservation.

That means most of the reservations have been made for more than eight passengers. By using our approach, service providers who are providing travel agency services, and the customers who are taking the services from it can monitor SLA status in SOPL based environment more easily.

VII. Discussion

This work is evaluated with a case study system as described above. Thus considering productivity, scalability and the performance of our implemented approach indicates to realize that how far this work is successful.

1) *Productivity:* WSLA language specification to this running case allows for a significant amount of the necessary SLA management process to be created accurately from the feature oriented service component configuration to Service Oriented Product Line (SOPL). This accurate SLA management of SOPL environment can reduce error and development time for service providers who take the responsibility of making service components and their variability implementation. Creating individual components of Web services is a time consuming task without following the best practices of SPL. Thus without having proper mechanism for monitoring QoS compliance of Web services via SLAs would generate greater potential for error. Although implemented SOPL architectural environment is relatively simple to define core assets, the implementation process of both SOPL environment and the management process of SLAs would generate considerable productivity while service provider is providing maximum functionalities based on identified features and variabilities. Compared to the identified service bundles of components, it is nevertheless a potentially complex activity to systematically describe any features of web service components in terms of its capability and interface. When productivity comes, any assessment process of each asset, which can be provided to the service consumer, is useful. In addition, combination of well-placed commonalities and variabilities in identified service components generates well-established SOPL environment to manage Quality of Service (QoS) compliance on top of the platform architecture to create flexible service components with defined agreement specifications between two parties.

2) *Scalability:* Our approach is designed with more reusability and modularity. Therefore, expected systematic approach would be resulted with a specific solution that has ability to scale well in some aspects. The scalability is, in part, determined by the available framework containers support associated with the various formalisms. The OSGI can be used to implement SOPL with runtime modularity. As such, presented case study is based on Sample Travel Agency since it has three different service bundles. However, from the service provider end, it can be more scalable easily by producing Product Line Engineering paradigm with identified service components (core assets). The feature modelling practices proposed in this work can be applied accurately. Then the identification of service component happen as expected. However, extremely large feature models can be challenging to identify service components and the variation points with standard commonalities and variabilities, especially if they represent real-world services in a changing SOA environment. In terms of core assets management, scalability is extremely easy. If we think about the SLA management with defined QoS compliance attributes, scalability become challenging. The primary goal of our work is to determine whether defined SLA parameters are exist with applicable form of maintaining valid agreement between two parties. Thus, scalability is a significant factor in this case. We are evaluating some WSLA language based templates to verify whether some QoS attributes are changed or not according to scalable SOPL environment. We are successful by defining some SLA parameter definitions,

which process parameters in each individual component with specified Service Level Objectives (SLOs).

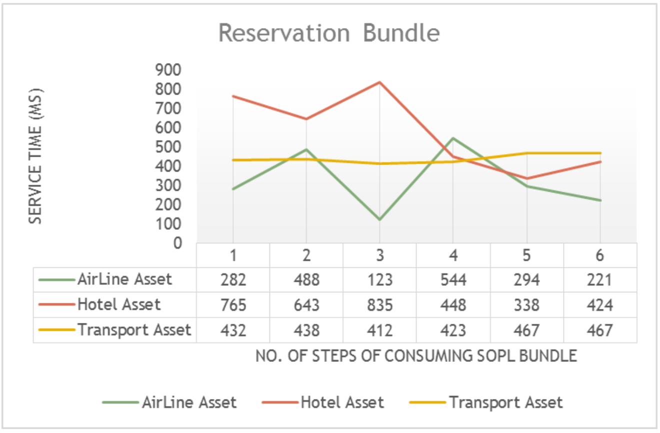
3) *Performance:* The activation process of bundles of services in SOPL takes mere seconds with the size of feature model used in this running case. For the demonstration core assets management and SLA management with the running case input, the time for bundle rearrangements takes approximately 200 – 700 milliseconds with the number of service components as 3. For the processing of WSLA template with defined SLA parameters in Reservation service bundle takes lower than 1000 milliseconds. Similarly, the extraction of the Service Level Objectives (SLO) from the output of WSLA template using our implemented approach of SLA management process is also relatively fast. Comparatively, the developer who is going to implement SLAs within SOPL environment would typically take far more time defining and configuring the SLA templates for each service components from development platform. Table 1 presents lowest service time of each bundles among 6 consecutives service requests, declared under two major service bundles (Reservation & Payment). It shows some deviations has happened with actual value. Like that way, SLA violation detection would be part of management process. Typically, successful execution of the running case composition for two service bundles in SOPL takes approximately 2937ms / 2634ms each or often less.

TABLE I
SLA MANAGEMENT OF TARGET SERVICE BUNDLES

Service Bundle	**Defined Metrics in WSLA templates**
	Service Time (Actual time with 1 sec)
Reservation	AirLine asset: 221ms Hotel asset: 338 ms Transport asset: 412 s
Payment	Credit Card asset 1252 s Debit Card asset 1228 s

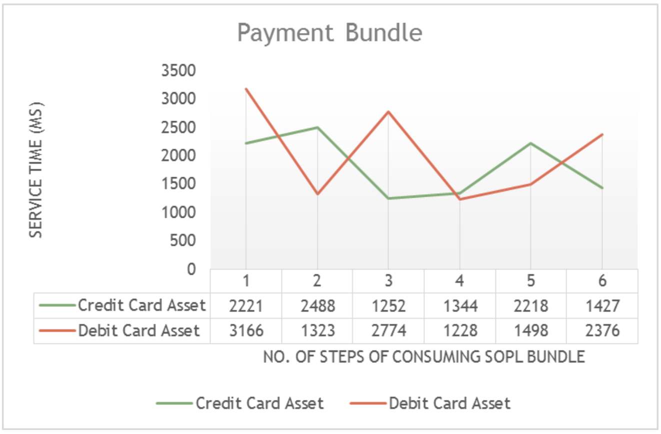
Fig. 18 and Fig. 19 show the measurements of service times for six consecutive service requests for reservation and payment bundles. By comparing two graph for reservation and payment bundle executions, service components inside payment agency bundle is spending more time than the service components inside reservation service bundle. However, if we analyse two charts, we can show Transport (Vehicle) reservation asset has small value changes for service time. Airline reservation asset has big value changes due to number of passengers.

Thus, the comparing the overall performance inside reservation bundle Transport asset gives average performance throughout each service requests. Hotel reservation has low performance initially, but after 4th, 5th and 6th requests is being processed, the performance is increasing. Payment bundle does not have better performance than reservation bundle. However, payment providers have to be initiated their parallel process to get maximum performance from it.



	1	2	3	4	5	6
AirLine Asset	282	488	123	544	294	221
Hotel Asset	765	643	835	448	338	424
Transport Asset	432	438	412	423	467	467

Fig. 18 Measurements of service time for Reservation bundle



	1	2	3	4	5	6
Credit Card Asset	2221	2488	1252	1344	2218	1427
Debit Card Asset	3166	1323	2774	1228	1498	2376

Fig. 19 Measurement of service time for Payment bundle

VIII. CONCLUSIONS

This paper aimed to present an approach of implementing Service Oriented Product Lines (SOPL), which has been used as key term of handling service components that can be shared with identified commonalities and variabilities. We implemented SOPL environment with the specific bundles of service components that maintain core assets. The second phase introduced management of Quality of Service (QoS) attributes of those service bundles with the defined set of terms and attributes in SLA templates, provided by WSAG4J Java framework tool and customized XML templates for WSLA in order to apply management technique for .NET based service applications. Our current research objective has already been reached at this stage. But, in future we are going to enhance our approach with better user experiences. This will support the developers who are representing service provision end and willing to manage service components accurately. Then they can easily identify the deviation results of pre-defined and actual quality metrics in service components. Applying the implemented SOPL environment with dynamically changing SLA conditions for evaluation at real time can be vital. To gain the true service orientation even the SOPL must be able to adapt to new SLAs while maintaining QoS levels agreed. One may explore how the proposed SOPL model can support for such needs.

REFERENCES

[1] C. Parra, D. Joya, L. Giral, and A. Infante. An SOA approach for automating software product line adoption. *In Proc. of the 29th Symposium on Applied Computing (SAC'14)*, Gjeoungju, South Korea, ACM, March 2014.

[2] R. Krut and S. Cohen. Service-oriented architectures and software product lines - putting both together, *in SPLC*, 2008, p. 383.

[3] M.-Matar, and H. Gomaa. Feature based variability for service oriented architectures IEEE *WICSA*, 2011, p.302-309, June, 2011

[4] Lee, J. and Kotonya, G. Combining Service Orientation with Product Line Engineering, IEEE Software, 2010.

[5] M.P. Papazoglou and W.-J. Heuvel, (2007) Service Oriented Architectures: Approaches, Technologies and Research Issues *The VLDB Journal*, vol. 16, no. 3, pp. 389-415, 2007.

[6] McGovern, J., Tyagi, S., Stevens, M. E., Mathew, S. (2003). Service-Oriented Architecture. In Java Web Services Architecture. (pp. 35-63). Elsevier.

[7] Papazoglou, M. P. (2003). Service-oriented computing: concepts, characteristics, and directions. *In Proceedings of the 4th International Conference on Web Information Systems Engineering* (WISE'03).

[8] Schmid, K., Verlage, M. (2002). The economic impact of product line adoption and evolution. IEEE Software 19, 4, 50-57.

[9] Medeiros, F. M.; Almeida, E. S.; Meira, S. R. L. Towards an Approach for Service-Oriented Product Line Architectures. *3rd Workshop on Service-Oriented Architectures and Software Product Lines (SOAPL) - Enhancing Variation, in conjunction with the 13th International Software Product Line Conference (SPLC)*, San Francisco, 2009.

[10] J. Lee, D. Muthig, M. Naab, “An approach for developing service oriented product lines”, in: *12th International Software Product Line Conference*, IEEE Comp. Soc., Limerick, Ireland, 2008, pp. 275–284.

[11] S. Cohen, R. Krut, Managing (2010) Variation in Services in a Software Product Line Context, CMU SEI, Pittsburgh, PA.

[12] M. Abu-Matar, H. Gomaa, M. Kim, A. Elkhodary, Feature modeling for service variability management in service-oriented architectures, in: *22nd International Conference on Software Engineering and Knowledge Engineering*, KSI, Redwood City, CA, 2010 pp. 468–473.

[13] Kang D,Baik D K. Bridging software product lines and service oriented architectures for service identification using bpm and fm[C]. *Computer and Information Science (ICIS),*2010 IEEE/ACIS 9th International Conference on. IEEE,2010: 755-759.

[14] V. Muthusamy, H. Jacobsen , P. Coulthard , A. Chan , J. Waterhouse , E. Litani, SLA-driven business process management in SOA, *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, October 22-25, 2007, Richmond Hill, Ontario, Canada [doi >10.1145/1321211.1321243]

[15] Bianco, P., Lewis, G.A., Merson, P.: Service level agreements in service-oriented architecture environments. Technical Report CMU/SEI- 2008-TN-021, Carnegie Mellon (2008)

[16] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer and A. Youssef. Web services on demand: WSLA-driven automated management. *In IBM Systems journal*, Vol. 43, No 1, 2004.

[17] Chrysostomos Zeginis, Monitoring the QoS of Web Services using SLAs - Computing metrics for composed services, *Master’s Thesis*, Heraklion, PP. 13-14, March 2009.

[18] Ribeiro, Heberth G. Braga, et al. An assessment on technologies for implementing core assets in service-oriented product lines. *Software Components, Architectures and Reuse (SBCARS),* 2010.

[19] Darwish, Nagy Ramadan, Rabab Emad Mohamed, and Doaa Hany Elsayed. A Proposed Approach for Monitoring Quality of Web Services Using Service Level Agreement. *International Journal of Computer Science and Information Security* 13.1 (2015): p.29.

[20] A. Keller and H. Ludwig. The WSLA Framework: Specifying and Monitoring of Service Level Agreements for Web Services. *IBM research report RC22456*, May 2002.

[21] Ludwig, H., Keller, A., Dan, A., King, R. P., and Franck, R. 2003. Web Service Level Agreement (WSLA) Language Specification. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.