Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
2326-9865

Applying MiningTechniques to Analyze Evolution in SOA based Systems

Zeenat Parween[1] , R. B. S. Yadav[1]
[1] Department of Computer Science and Mathematics, Magadh University,
Bodh Gaya, Bihar 824234, India

Article Info
Page Number:11136 - 11156
Publication Issue:
Vol 71 No. 4 (2022)

Article History
Article Received: *15 September 2022*
Revised: *25 October 2022*
Accepted: *14 November 2022*
Publication: *21 December 2022*

Abstract

To maintain the SOA based systems, it is important to analyze the changes. In this paper, we analyze the changes in SOA based systems using mining techniques. We have first developed an algorithm to classify the evolution in these systems using the four types of changes which are addition, deletion, and modification.We have used WSDL files to identify these changes. Then, we have proposed formula toobtain the measure of evolution in these systems. We have proposed a model named SEMM (Service Evolution Management Model) to analyze the evolution in these systems. We have empirically validated the model using real world data and simulated data where real world data was not available.

***IndexTerms*—**Change,Service,SOA, evolution, Alter and Adaptive.

I. INTRODUCTION

Service-oriented architecture (SOA) is an architecture for designing and developing distributed systems and is used tofulfill business goals such as reduced expenses and agile adaptation etc. SOA based systems have services which are self-contained, modular, discoverable and dynamically bound. The service user of SOA based systems is not required to have the service implementation. Instead, the service is placed and destined at runtime. SOA systems are much different from other development systems such as the variety of service consumers and service providers, shorter release cycles etc. These are because of the ability of SOA based systems to rapidly adapt to the varying business requirements.

The SOA based systems uses a commonplace to accumulate the services and support thecontinuous evolution of services. The agile model of services in SOA based systemssupport the constantevolution and maintenance.The process of growth and maintenance of SOA based systems becomes difficult as the complexity increases and therefore, there is a need for developing an automatic management tool for these processes when changes occurs in services.

In this paper, we analyze the evolution which occurs in services in SOA based systems.The research in this paper consists of two data mining techniques. One of the data mining technique isAdaptiveminingorevolutionimpact analysis. This mining aims to determinethe evolutiondata. Another data mining technique is theAlter mining which aims to determinechange information.

Ourmethod of discovering the change information in SOA based systems uses dataminingtechniques.These SOA based systems are based on web. Usually, such web-mining techniques are classified as web content, web usage and web structure mining [13], [14]. In our research, web structure mininghas been takeninto consideration.This technique extractsdatafromserviceinterfacefiles i.e.WSDLfiles. The implementation files can also be used here.In our paper, we have taken real world data as well as simulated data and considered the data which changes over time. Then, we have applied data mining techniquesi.e. *Alter mining*and*Adaptive mining*.

The reason for usingAlter and Adaptive mining for SOA basedsoftware is to discovertransformation information as the time passes. Initially, we have used classification theory to categorize the changes by considering 2 versions of the interface files of the SOA based system. Then, we developservice metrics to measure the changebetween sets of different interface file versions. In this way, we aim to extract the differences between the services when changes occurs and then categorize these changes.Now, the challenge is to automate these tasks to extract the required information.

In the research, we use the WebServicesDescriptionLanguage (WSDL) files to discover the changes toselectsome ofthe test cases. This is done toaccomplish the regressiontesting of the services.With the use of regression testing and data mining techniques, in our research, we develop service metrics to measure the service evolution. We propose a new mining algorithm for service evolution.

To attain this, we use the data from the Alter mining basedonthe categorization of evolution proposed in the paper which are additions,deletions,andupdations. This datahelps in extractingseveralsegments of WSDL file. Then these segmentsarecombinedtodevelopa*WSDLsegment*.The subsets of the interface of a service is analyzed by this WSDLsegment.Now, because we do not need to test the complete service interface, but only parts of it, therefore we are also saving efforts forregressiontesting as well asreverse engineering.This segment is developed in very less time. This segment isbeneficial to both the one who provides the service as well as the one who uses the service. The one who provides the servicewill be able toregulateconsumption of a service with the support of WSDL segmentwhich aids inretrieving a subset of a service.The advantage to the service consumers is that they need to concentrate on the appropriatesubgroup of the operations of a service. A WSDL segment isbeneficial in research of effect of evolution in one operation on rest ofoperations in SOA based systems.

Then, in the research, we developvarious metrics for a service which aids inassessing the risk linked with the service in SOA based systems. The proposed metricscan be used in maintenance of currentservice versions as well as in creation of next serviceforms. These metrics may aid a servicecustomerto analyze theoperations provided by WSDL files. The regression testing is also supported by the help of the proposed metrics when changes occur in the services in in SOA based systems.

Next, wepresent a change analytic model toanalyze the various service versions from the logs. These logs are formed from the various service versions. The proposed model will help increating and sustaining Service Level Agreements (SLAs) and Quality of Service (QoS). Services evolve over time by the service providers so that they can enhance their services and meet new requirements of the market. But these changes can lead to SLA violation where an improper change results inimproperbehavior. Therefore, the Quality of Serviceneeds to be checked regularly so that it is ensured that SLA violation and QoS is maintained. When we make less work to be done in maintenance of the services then the work to be done for ensuring Quality of Service is also decrease [23, 24]. And after making less efforts for QoS, both time and cost are decreased for regression testing[17]–[20].

The major work done in the research are as follows.

In Section 2, an algorithm for classifying the evolution in theweb service is proposed. The algorithm is used for performing the Alter mining using the different versions of the web service.We consider two versions of the service in our algorithm. The data required to prepare the interface segment is the evolution data. This gives us the interfacesegment. The WSDL segmentis agroup of functionalities defined in the service interface which is present in the WSDL file. Also, to consider more than two versions of web service, we propose a method to perform Adaptive mining on them. We have used various metrics to perform the Adaptive mining.Next, in the Section 3, we discuss

the model which is the Service Change Analytics model. This model is used to perform Alter and Adaptive mining on theweb services. In next section, the experiments and resultsare elaborated. Here, the model is evaluated using real world data. Next, the literature review is described in Section 5.Lastly, research is wrapped up in last section.

2. PROPOSED MINING TECHNIQUES

In the currentpart, we have considered two scenarios.

a) To consider two versions of service, we have performedAlter mining

b) To consider more than two versions of service, we have performed Adaptive mining

The distributed system is used for the evolution data to perform Alter and Adaptive mining.In this section, we use below terminology for the series of versions of a web service and the time at which these versions are considered.

VS = {V1, V2, …, VN} is a series of different versions of a web service

{t1, t2, …,tN}represents the time at which the above versions are considered.

Next, we present the process of Alter mining of the two versions of a web service.

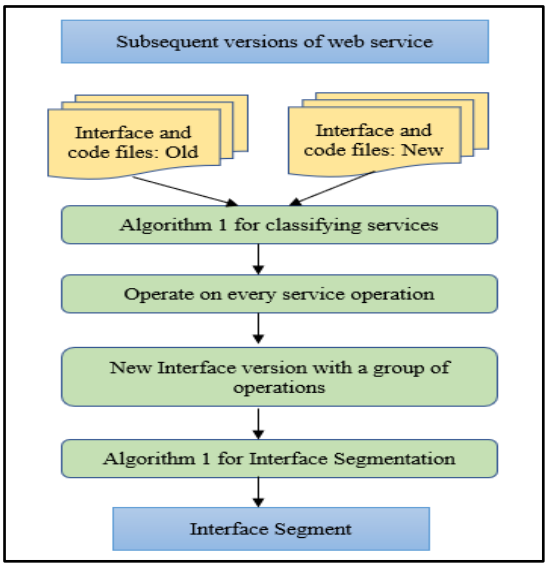


Fig.1.Alter mining process

The WSDL file and the Web service code is considered for the above process.
In the coming sub-sections, we will elaborate the two types of mining applied in the research. In the former sub-section, the categorization of evolution in the web services is explained. This categorization aims to develop the Interface segment.The later sub-section explains the metrics which we have developed in order to understand the changes in the web services in a distributed system. We have developed three metrics to measure changes in a series of WSDL files.

A. Two web service versions for Alter mining: Categorization

Here, two versions of a web service are considered to measure changes across two subsequent versions. We call the one in which changes are done is termed as previous version and the one which contains the changes is termed as new version of service. Algorithm considers theWSDLfile

as well as theWScode.

Figure 1 describes the developed algorithm in which there are two main parts.

The input to the *AlterminingOfService*are thepreviousandnewversions of a service.*Algorithm 1,ServiceAlterClassifier*sums up the changes mentioned under the labels in Table 1. These changes are done in WSDL and WS code of the latestservice version. The changes in the service areinsertion,deletion,andmodification.The operations in which these changes are done are then collected. Here, those operations are considered on which either insertion or modification changes are done. The algorithm deletes those operations which are under the label "*deleted"*. This is due to the fact that theoperationswhich have been deleted arenotexistinginthenewWSDL file and therefore, it is not worthy to include themin WSDL segment. Thesecollected operations are present under $\text{Subset}_{\text{operations}}$. Lastly, *Algorithm2* is calledto develop the WSDLsegment whichincludes the group of operations in the original WSDL file.

TABLE1
CHANGECLASSIFIERS

Change	File Type	Change Description	Interface subset
$\text{Addition}_{\text{WSDL}}$	Interface	Addition of service operation/schema type	DWSDL
$\text{Addition}_{\text{WSCode}}$	Web service code	Addition of service code	UWSDL
$\text{Deletion}_{\text{WSDL}}$	Interface	Removal of service operation from interface	None
$\text{Deletion}_{\text{WSCode}}$	Web service code	Removal of service operation from service code	None
$\text{Modification}_{\text{WSDL}}$	Interface	Modifying schema type	DWSDL
$\text{Modification}_{\text{WSCode}}$	Web service code	Modifying service code	UWSDL

TABLE2
EVOLUTION IN WSDL ELEMENTS

Service Element	Impact of service change on interface
Definition	Change in service name or namespace
Type	Change in simple or complex data type in XSD Schema file
Message	Name of message or name of message part may change
Port Type	Name of porttype or operation may change
Binding	Name of binding may change
Service	Name of service or binding address may change

ALTER MINING APPROACH

AlterMiningOfService (Old Interface, New Interface, Old Code, New Code)

1. opLabels = ServiceAlterClassifier(Old Interface, New Interface, Old Code, New Code)
2. Assign initial values to service lot_operations
3. For every pair in opLabels
   a. If label is equal to "addition" or label is equal to "modification" then
      $Subset_{operations}$ = lot_operations Union $operation_{Name}$
   End For
4. $Interface_{Segment}$ = Interface Segment (New Interface, $Subset_{operations}$)
   Return $Interface_{Segment}$

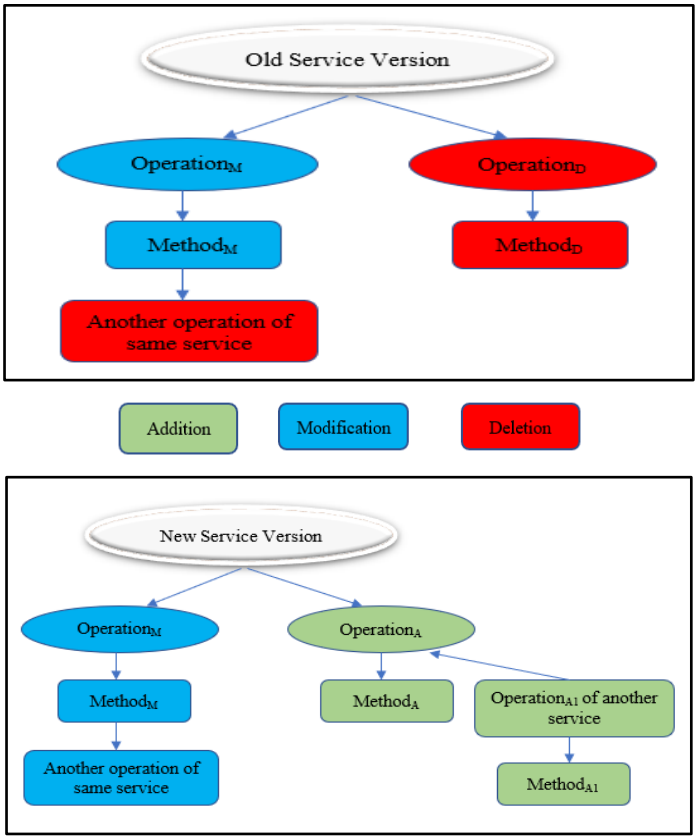


Fig.2.Service Evolution

The WSDL and WS code are subjected to change classification in first stage. The identified change types are summarised in Table 1. Each change is documented with the change's level (WSDL or WS code), its goal, and interface segment (described in the below part).

Six main components of a WSDL description are mentioned in Table 2. The definitions of services offered are given in the first section. The second one contains the XSD schema, which depicts structures for data for each operation's I/P and O/P. Message calling, the 3rd component, facilitates interaction between the customer and the operation. The port used for each operation is provided in the fourth part. The fifth step involves a binding that links each operation's input and output data types. The service that can be accessed through a URL is finally defined in the sixth section.

Classifying the components of the interface and WS code is aim of first stage. Any one of the following can be altered:the input-output parameters, the operation code, and the communication messages. Attaching classification labels to things is a data mining technique for classifying items.

By marking the three different service pieces, we can thereby determine which procedures have changed. Step 2 simply collects all operations with the labels "inserted" or "updated" (as subset operations).

The first two steps are demonstrated in the example in Fig. 2. To create the "Service version 2," the "Service version 1" is upgraded. A modified "$Operation_M$" in "Service version 1" calls a changed "$Method_M$" (which in turn invokes a deleted Operation). Additionally, "$Operation_D$" uses "$Method_D$," both were removed in"Service version 2." Changed "$Operation_M$" calls changed "$Method_M$" and modified operation." Further, inserted "$Operation_A$" invokes inserted "$Operation_{A1}$" and next inserted "$Method_{A1}$,"and added "$Method_A$." "$Operation_{A1}$" is delivered by another web service.

Algorithm 1, ServiceAlterClassifier

facts: new version of interface is termed asNew Interface, previous version of WSDL is termed as Old Interface, new version of Web Service code is termed as New Code, and previous version of Web Service code is termed as Old Code.

These six steps make up the algorithm.

1) Determine the semantic WSDL differences between New Interfaceand Old Interfacewith the help of semantic differencing. Keep the outcome in $Interface_{Alterations}$.

2) Use semantic differencing to determine the differences in the semantic code between New code and Old Code. Put the outcome in $Code_{Alterations}$.

3) Create labels for modifications between previous and latest versions by initializing list opLabels. Obtain the operation name change, which gives title of modified operation and its accompanying label (from $Interface_{Alterations}$and $Code_{Alterations}$).

4) When looking for operation modifications in the $Interface_{Alterations}$, make sure to:

a) assign the added label in opLabels to any operations that were insertedin latest version of interface.

b) delete any operations from previous form to producelatest form of interface and

c) if an operation from previous form is evolved while drifting to latest form of interface, then allocate it to evolved label in opLabels.

5) When checking$Code_{Alterations}$for operation modifications, do the following:

a) Assign any operation code lines that were added to the updated WS code with the label that was placed in opLabels.

b) Assign removed label in opLabels to someLOC that are deleted as part of an operation to build a latest form of the WS code.

c) Allocate the amended label in opLabels to someLOC in an operation that are changed during the migration to the latestform of the WS code.The sequencing of steps four and five go through the interface and the web service code. This is crucial in this technique since the 2^{nd}scan may cause a label from $1^{st}$to be reset. Therefore, if the $Interface_{Alterations}$analysis is completed before the $Code_{Alterations}$analysis, the web service code analysis will take precedence. The WSDL analysis, on the other hand, takes precedence if Steps 4 and 5 are switched. Additionally, either stage may be skipped, allowing a cloud service holder or its customer to use the method independently.

To distinguish the modificationsamong the 2forms ofweb service code and 2 forms of interface, a tool is utilizedwhich provides these modifications, in the implementation. The algorithm is as follows.

Algorithm 1: ServiceAlterClassifier (Old Interface, New Interface, Old Code, New Code)

1. $Code_{Alterations}$ = DiffSemanticCode(Old Code, New Code)
2. $Interface_{Alterations}$ = DiffSemanticInterface(Old Interface, New Interface)
3. Assign labels which do not change to opLabels and select opLabelChange from $Code_{Alterations}$ and $Interface_{Alterations}$
4. While every opLabelChange in $Interface_{Alterations}$
   a. If (opLabelChange is addition)
      Give label "addition" to operationName in opLabels
   b. If (opLabelChange is deletion)
      Give label "deletion" to operationName in opLabels
   c. If (opLabelChange is modification)
      Give label "modification" to operationName in opLabels
   End While
5. While every opLabelChange in $Code_{Alterations}$
   d. If (opLabelChange is addition)
      Give label "addition" to operationName in opLabels
   e. If (opLabelChange is deletion)
      Give label "deletion" to operationName in opLabels
   f. If (opLabelChange is modification)
      Give label "modification" to operationName in opLabels
   End While

Following step uses InterfaceSegmentation defined as follows to record the differences between previous and latest forms in an interface segment. In order to have a portion of original interface, InterfaceSegmentation pulls a group of the interface's functionalities from a given system. A subset of the operations in WSDL are contained in the subsequentInterfacesegment, which is known as anInterfacesegment. Bank service might, for instance, support 2 operations "deposit" & "withdraw." Occasionally, an interface segment will just have "deposit" feature.

AnInterfacesegment might help an engineer (for example, during testing) through concentrating their consideration on the pertinent group of service because it is an interoperable group of a service. As can be seen in Table I, our method uses the Difference Interface (DINTERFACE) and the Unit Interface(UINTERFACE) of the Interface[15]–[17]. Based on variations in the Interface, the DINTERFACE is created. On the basis of variations in the WS code, the UINTERFACE is created.

The WS change classification is used to determine the WS code discrepancies. This procedure first separates the operations and procedures of the previous and latestforms of the code, after which it finds alterations such as position (file and line) of everymodification. The DINTERFACE (group of the interface) is designed to record interface-level modifications, on one hand. One variety of InterfaceSegment is the DINTERFACE. As opposed to that, we nextrecognize activities that have inter-operational dependencies on the intra-operational and intra-procedural modifications.

Algorithm 2: InterfaceSegmentation (New Interface, $Operations_{subset}$)

Datatype String to Sdefinition, XSD, Message, Port, Bind, Ser, Edefinition
StringArray[] $InterfaceOperation_{New}$

1. $Operations_{subset}$ belongs to $InterfaceOperation_{New}$
   $Operations_{subset}$ is a collection of operations needed for InterfaceSegment
2. Sdefinition = BeginningDefinitionSegment (NewInterface)
   XSD = XSDSegment (NewInterface, $Operations_{subset}$)
   Message= MessageSegment (NewInterface, $Operations_{subset}$)
   Port= PortSegment (NewInterface, $Operations_{subset}$)
   Bind= BindSegment (NewInterface, $Operations_{subset}$)
   Ser = ServiceSegment (NewInterface)
   Edefinition = EndDefinitionSegment (NewInterface)
3. InterfaceSegment = Sdefinition + XSD + Message + Port + Bind + Ser + Edefinition

Return InterfaceSegment

The UINTERFACE, an interface segmentwhichcontainsevolution in web service code, is built using these modified techniques. As a result, the UINTERFACE can be used to access and carry out WS code changes. We record evolution in the below3 ways which dependupon the labels in opLabels.

1) First, both the DINTERFACE and the UINTERFACE record inserted operations.
2) Because they are not included in the newly released version of the service, removed operations are disregarded in the SegmentInterface.
3. Modified operations can be found in interfaceas well as web service code. Evolution in port, binding, and XSD (input, output, or both) are indicative of Interfacelevel modifications and are detailed in Table II. These changes are recorded in the DINTERFACE. The UINTERFACE contains information about changes made at the WScode level.
Interface of latest form and the specified group of operations are the final 2 inputs for Algorithm 2InterfaceSegmentation. These 2 are utilized to produce the interfacesegmentconsuming the functions segmentXSD, segmentMessage, etc. These functions take out a group of the interface. The interfacesegment is created by concatenating these subsets (which were captured as substrings).
A semantically significant portion of the functionality of a service is accurately captured by anInterfacesegment. It can be calculated without the code, as was already mentioned. But as Interfacegives users access to the WScode,Interfacesegment can also give them a way to a condensed form of the code when performing regression testing.

B. Mining on a Series of forms of web service

Here, 3 innovative service evolution metrics are used in this subsection to characterize Adaptive mining of a service. Number of operations, WSDL lines, messages, and operation code lines are the four key quantitative elements on which the metrics are based.

The form series $V_S = V_1 ... V_i ... V_N$ is quantitatively evaluated using these four attributes, where V_i denotes the i^{th} version of the service. Here,3 Service Adaptive Metrics are defined based on the fourattributes. Idea for these metrics is to aggregate attributes from several versions to provide information about web service evolution.

1) Lines Per Operation in Interface: This is proportion of interface lines (other than comments, blank lines) to Interface actions. $LINTERFACE_i$ stands for Lines per Operation in Version V_I.

$$LInterface_i = \frac{Numberofsourcelinesin Interface of V_i}{Numberofoperationsinversion i}.$$

$$This defines a set as LInterface = \{(V_1, LInterface_1) \ldots (V_i, LInterface_i) \ldots (V_N, LInterface_N)\}.$$

2) MessagesPerOperationintheInterface*:*This metric is the proportion of messages to operations in the Interface. Under typical circumstances, there are two messages sent for each operation, one for input and one for output. In the $Interface_i$ of V_i, the measure is known as M_i the Messages per Operation and is defined as follows: output, or both), as explained in Table II. These changes are recorded in the DINTERFACE. The UINTERFACE records modifications at the WS code level.

$$MInterface_i = \frac{Numberofmessages in Interface of V_i}{Numberofoperationsforversion i}.$$

As stated below, a set is $MInterface = \{(V_1, M_1) \ldots (V_i, M_i) \ldots (V_N, M_N)\}$.

3) CodeLinesPerOperationintheserviceCode*:*The last metric uses the Interface'srationale. It is proportion between the count of operations and the LOC. This illustrates how, from a business logic perspective, service operations have grown. $It is represented as CL_i the WSCode Line per Operation in WSCode_i of V_i$

$$\text{Ccode}=\frac{\text{NumberofcodelinesinWScodeof}V_i}{\text{Numberofoperationsinversion}i}.$$

$\text{Ccode}=\{(V_1,\text{Ccode}_1)\ldots(V_i,\text{Ccode}_i)\ldots(V_N,\text{Ccode}_N)\}$.

Together, these three unique measures can be stated as (V_i, LInterface_i, MInterface_i, Ccode_i). These three metrics are applied to the versions to provide 4 time series graphs which are helpful in the analysis of SOA based web service systems that are changing. Fig. 3 summarizes the Adaptive mining of a form series using Service Adatpive Metrics.

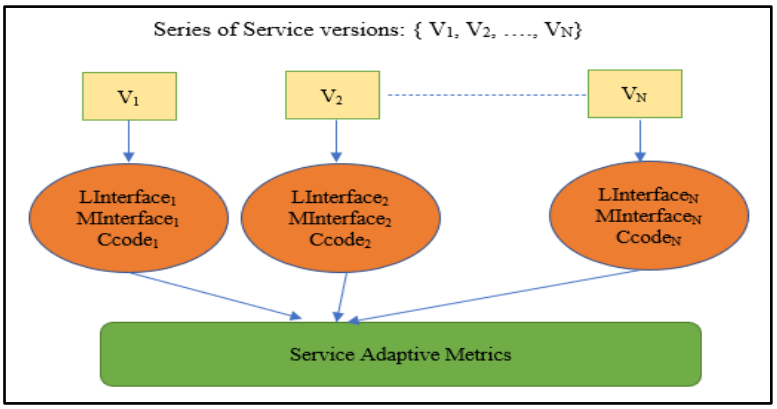


Fig. 3.Adaptive mining of version series

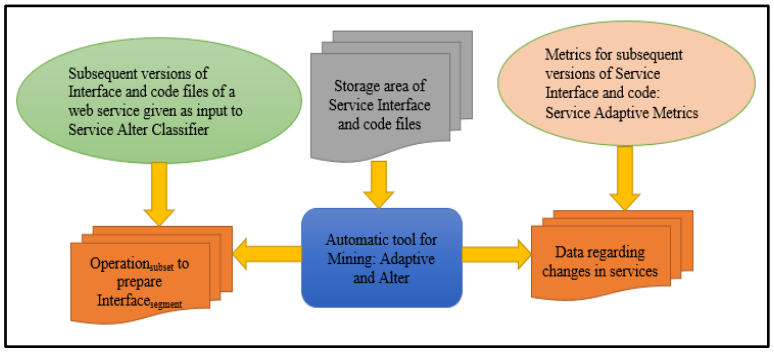


Fig.4.Evolutionmodel.

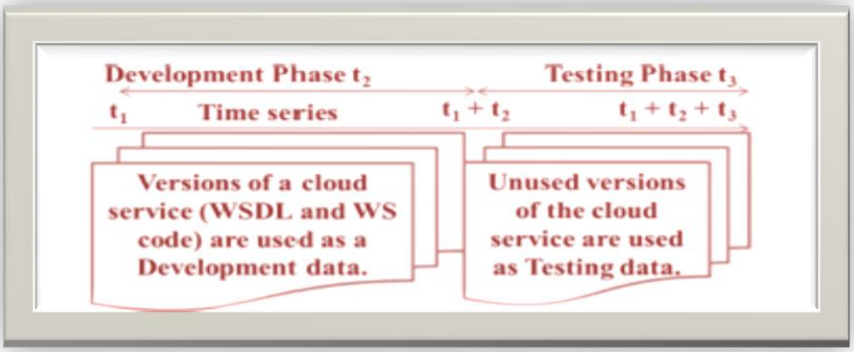


Fig. 5.phases of the tool's development and testing. Suppose the development phase lasts for time t2 and begins at time t1. Let the testing phase that follows begin at time t1 plus t2 and last until time t3. As a result, both phases come to a close at time t1 + t2 + t3.

3. SERVICEEVOLUTIONANALYTICS

We will now go over the model and tool for our Analytics. Fig. 4 uses version series' Interface and WS code. Fig. 5 shows that there is a growth phase (time t2) and a testing phase after tool creation (time t3).

The tool picks up knowledge from prefix of the service's version series during the development phase. The method is supervised learning; if the result is wrong, the tool's code is updated before Interfacesegment is computed again.

TABLEIII
INTERFACE FOR EVOLUTION ANALYSIS

Service Name	Difference Interface	Unit Interface
AWS	Yes	No
HotelService	Yes	Yes
BankService	Yes	Yes
KBB	Yes	Yes

There are two possible criteria for determining output correctness: software approval and human acceptance. An IDE (such as NetBeans and Eclipse) or testing framework (such as SoapUI and JMeter) must accept a portion of the WSDL as an interface segment in order for the software to be accepted. An engineer evaluates if the tool's output is appropriate for human approval. While creating a segment can be challenging, a developer (or tester) should have little trouble spotting the right results. By doing this, the tool picks up the proper interface segment from the developer. When the IDE or testing framework accepts the tool's output as the desired interface segment, the procedure is finished. The desired structure for facilitating communication between the interface's operations and the WS code should be present in the interface (WSDL) segment.

The created tool is then tested utilizing the remaining versions of the service throughout the testing process. The output of the tool is compared to the expected output, which contains the WSDL (interface) segments, to determine how well it performs.

AWSCM: We developed a new segmentation component for the programme AWSCM (Automated Web Service Change Management) iteratively using this paradigm [16]. AWSCM is an intelligent automated programme that tries to comprehend the WSDL and WS code's structure. By doing Alter mining on evolving web services, it creates Interfacesegments. It compares the semantic differences between two WSDL versions using the Membrane SOA Model [60]. It makes advantage of jDiff [59] to identify semantic differences between two versions of the WS code. Empirically, jDiff offers accurate coverage information for the single test case at 84.70% and the complete test suite at 77.81%, as indicated in [57], [58]. We created a number of functions, such as a function for operation separation and a function that separates different parts of the WSDL, to take into account these semantic variations. The segmentation code for AWSCM was continuously improved until the tool was capable of producing the proper Interfacesegment on its own for each system examined in the next section. The four recommended service evolution metrics listed in Section II-B were computed using an upgrade to AWSCM. A new module called "Service Evolution Metrics" has this new capability.

4. EXPERIMENTS

This section outlines three experiments that take into account two self-created example web services and two actual web services (see Table III). The usage of change classification in Interfacesegment building is presented in the first subsection.

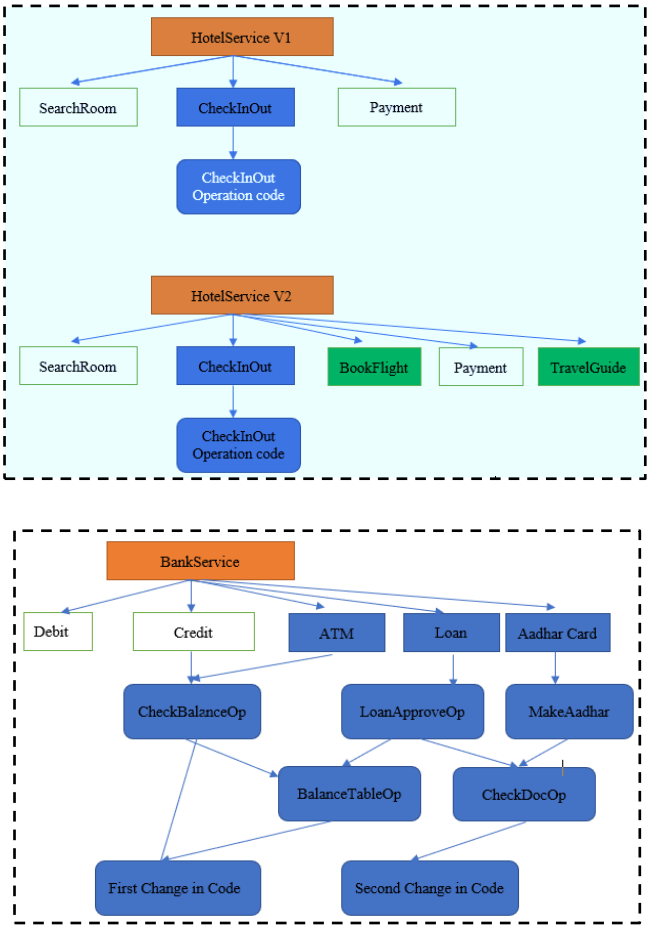


Fig. 6.Simple representation of the changes in two self-made web services,HotelService,andBankService.

This section outlines three experiments that consider two self-created example services and 2 actual web services. The usage of change classification in Interfacesegment building is presented in the first subsection.

A. *InterfaceSegmentConstruction*

AnInterfacesegment is computed in the first experiment using Algorithms 1, 2, and 3. To determine the change classification, Alter mining is applied to the online service. It then creates the corresponding Interfacesegment using this change classification. Table III displays the interfacesegments generated for everysoftware taken into consideration. The Difference Interfaceand the Unit Interfaceare the two Subset Interfacethat we take into consideration, as stated in Section II-A. A "Y" in the table denotes a group of WSDL was created, andsolo "N" denotes the 1 instance where we were unable to access the source.

The modifications we made to the HotelService and BankService web services are depicted in Fig. 6. The dependency graphs of evolution in 2 services are shown in the figure as their impacts. These are the changes that are involved.

1) There are three operations in HotelService version 1: SearchRoom, CheckInOut, and Payment. Version 1 was modified in the following two ways to produceformtwo. Initially, 2 brand-new DINTERFACE-captured operations are inserted: BookFlightand TravelGuide. Second, we changed the CheckInOutoperation's code recorded by UINTERFACE.

TABLEIV
EVOLUTION AMONG TWO INTERFACES

Changes in HotelService
HotelServiceV14.wsdl → HotelServiceV15.wsdl
Operation Change: PlayGameOp added ; TravelInstances added **Schema Changes: HSGameType has added; HSTravelType has added ;** **HSInstancType has modified**
HotelServiceV21.wsdl → HotelServiceV22.wsdl
Operation Change: GiveCreditNode deleted ; RemoveStaff deleted; TrainStaff added **Schema Changes: HSTrainStaffType has added; HSGiveCreditType has modified ;** **HSCreditType has deleted**

Changes in BankService
BankServiceV31.wsdl → BankServiceV32.wsdl
Operation Change: DeleteNode added ; LoanInstances added **Schema Changes: HSCounterType has modified; HSDimensionType has modified ;** **HSInstanceType has modified**
BankService441.wsdl → BankServiceV45.wsdl
Operation Change: CancelNode deleted ; DetachInstances deleted; AddCreInstance added **Schema Changes: HSConfigType has added; HSDeConnectionType has modified ;** **HSDeduceType has modified**

2) The BankService version 1 code is raised to version 2 with 2 changes that have an impact on the subsequent activities. 2 operations, BalanceTableOpand CheckBalanceOp, are impacted by First Code Change, and their parent nodes in the graph are also impacted. CheckDocOp is impacted by second Code Change, which also has an impact on their parent nodes in the graph. Overall, three of the fiveBankService processes are impacted by the two code changes.

AWS and KBB are 2 examples of 3categorization labels (added, removed, and updated) for 2fast growing SOA based cloud services shown in Table V. When we see the outcome, it is evident that the semantic differencing tool's accuracy affects the change detection's accuracy, and as a result, AWSCM depends on the tool's accuracy.

B. ServiceMaintenance:ReducedRegressionTesting

4 studies which take into account both recovery of InterfaceSegments and their application in decrease of test-case are included in Table V. We carried out two experiments for AWS. In the first, two operations were used to construct the DWSDL, and one operation was used to create the UWSDL. For the two, the test case reduction was 50% and 75%, respectively. We conducted two trials for BankService.

TABLEV
EXPERIMENTS FOR ALTER MINING-BASED WSDL SEGMENTSANDTESTCASERETRIEVAL

Name of Service	Consideration of Test Cases	Interface Segment operation count	% decrease in Test Cases
AWS	3 out of 7 operations are inserted and the proposed technique retrieves 3 out of 7 TCs	Difference WSDL has 3 operations	43
	2 out of 9 operations are inserted and the proposed technique retrieves 2 out of 8 TCs	Unit WSDL has 2 operations	75
HotelService	4 out of 5 operations are derived from three changes and the proposed technique retrieves 3 out of 6 TCs	Unit WSDL has 3 operations	50
	5 out of 6 operations are derived from four changes and the proposed technique retrieves 3 out of 6 TCs	Unit WSDL has 3 operations	50
BankService	2 operations are selected which leads to retrieval of 2 out of 25 TCs	Difference WSDL has 2 operations	92
	5 out of 25 service code operations are changed leading to retrieval of 5 out of 25 TCs	Unit WSDL has 3 operations	80
KBB	2 out of 22 operations are chosen leading to retrieval of 2 out of 22 TCs	Difference WSDL has 2 operations	90.9

We aim to have the easy-to-understand experiments and thereforewe have not taken into consideration the external code dependencies. The diagramsshowtwo projects are expanding steadily. A manager can use the metric values to inform decisions about tasks like service monitoring and regression testing that are part of the evolution of cloud services. An abrupt change (glitch) in a time series graph of changes in service measure may indicate an abnormality, like an improper improvement. In this situation, the head of the software project may decide to validate the anomaly's cause, which will aid in assessing the project's risk. As a result, every blip in the time series graphs in Figure 7 represents a departure from the general trend.

Take the following three changes from Table VI as an illustration. First, HotelServicefirst version was made available in Jan 2020. The first row reflects this. After that, in Apr 2020, there were two operations added. Table VI's 2nd row (Apr 2020), 3rd row (Jul 2020),4th row (Oct 2020), and 5th row all show these four increases (Jan 2021). These modifications can be seen in Fig. 7's MInterface time series for HotelService. The development was nearly consistent between (Nov 20) and (Nov 21). The quick rise comes in third glitch from Nov 2021 to Jan 2022. In conclusion, evolutionorganizationallowsHotelServiceto get two distinct regression test suites, everyone of them contains 60% of the original tests, and AWSCM to build a WSDL segmentwhich contains

operations altered.

We use the assumption of only 1 test case for each operation in order to make remaining two experiments, which made use of AWS and KBB, more understandable. Size of regresion test suite in AWS is reduced by 75% when the user chooses 3 of the 23 available operations. AWSCM for KBB shrinks the test suite by 90.9% using the generated DWSDL.

C. *ServiceEvolutionMetricStudy*

Examination of 2web services, HotelServiceand BankService is now presented. We have developed our central repository toanalyse our proposed metrics. Table VI lists number of line, WS code lines, and WSDL operations. The proposed metricsare displayed in Fig. 7, are computed from counts in the table.

From Fig. 7, the following is what is inferred. First, WSDL for HotelService has somewhat more lines per operation than WSDL for BankService. The average number of code lines for BankService is more than HotelService. The Adaptive mining gives details on a complete version series, which aids in an empirical comparison of 2 services. 2 mining techniques together yield diverse data that is useful for service evolution research.

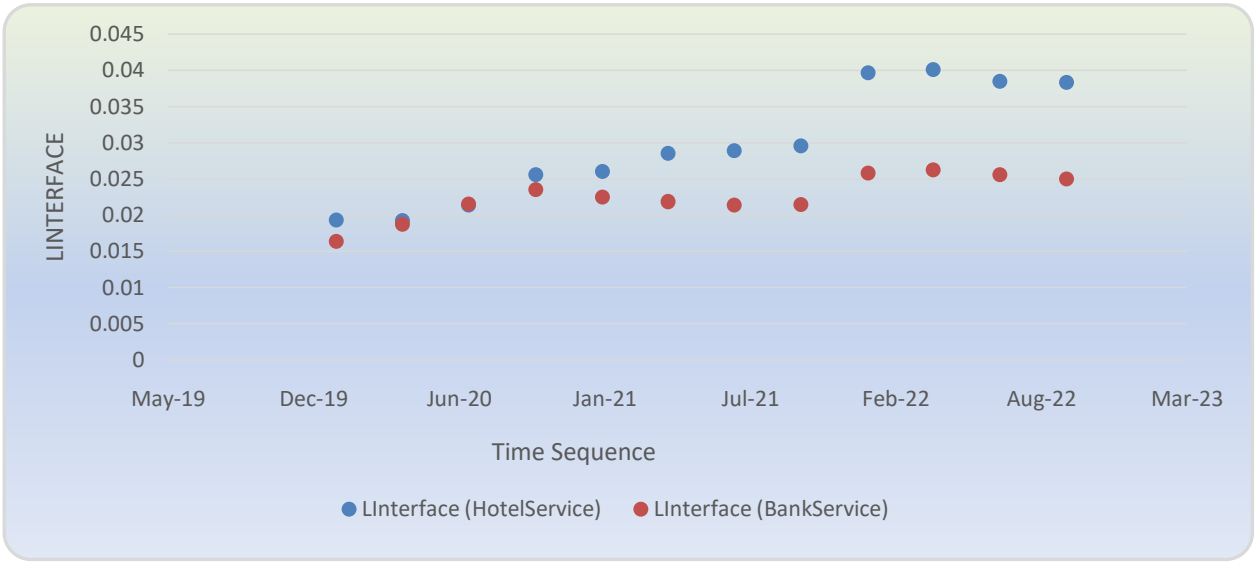
5. RELATEDWORKANDDISCUSSION

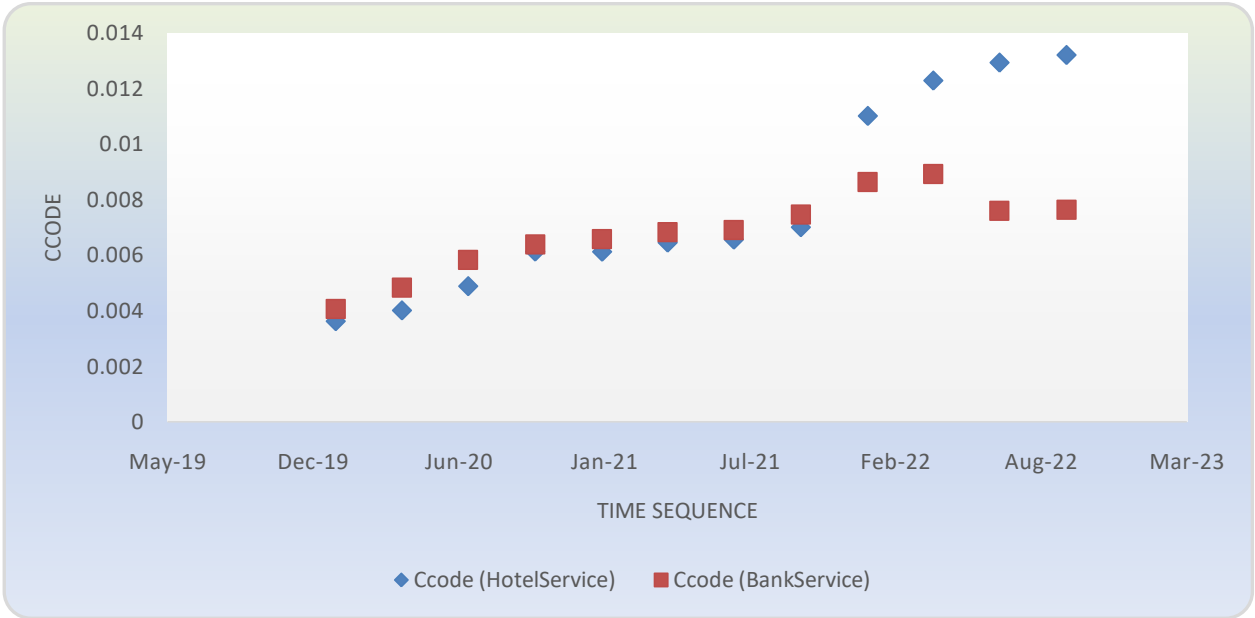
Literature which related to our proposed work are covered in this section. In the framework of change effect analysis, work on examinationof the software [27] and mining storage areas of software [28] includes the following, to start. For instance, Ying et al. [29] assessed the proposedresearch on code storage area for Mozilla and Eclipse before presenting a method to identify modification trends. Analyzing the KBB and AWS repositories helpedto assess the efforts. More recently, from an old repository, Negara et al. [30] discovered before undiscovered recurrent code modification patterns. After that, they examined a few of the high-level programme transformations and few of mined unidentified code modification patterns.

TABLE VI
INFORMATION ABOUT BANKSERVICE AND HOTELSERVICE TO CALCULATE SERVICE ADAPTIVE METRICS

Month and Year	HotelService				BookService		
	File version	Number of Lines in Interface file	Number of LOC	Number of Operations	Number of Lines in Interface file	Number of LOC	Number of Operations
Jan 20	1.0	518	2756	10	731	2956	12
Apr 20	1.3	624	2987	12	799	3111	15
Jul 20	2.0	701	3068	15	880	3264	19
Oct 20	2.1.3	822	3423	21	934	3444	22

Jan 21	2.3.2	845	3589	22	1022	3501	23
Apr 21	3.0	912	4026	26	1279	4100	28
Jul 21	3.1	934	4112	27	1355	4198	29
Oct 21	3.4	1016	4278	30	1489	4289	32
Jan 22	4.0	1235	4447	32	1512	4519	39
Apr 22	4.1	1298	4824	36	1599	4710	42
Jul 22	4.5.1	1467	5025	39	1716	5800	44
Oct 22	5.0	1516	5166	40	1799	5899	45





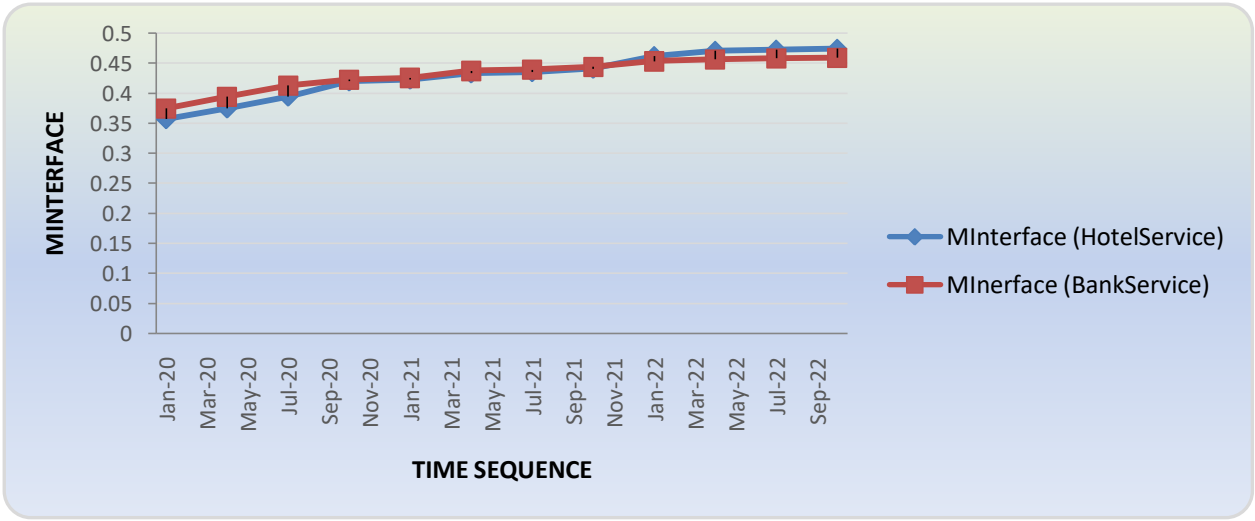


Fig.7.GraphstoshowService AdaptivemetricsforHotelService and BankService.

A method to calculate the price of process evolution in an architecture based on SOA was proposed by Xiao et al. [31] in their work on change analysis, mining, and learning of a developing service source. Change propagation was examined by Dam et al. [32] using SoaML, which is an extension of mostly used UML framework for Service Oriented Architecture. A method for managing the changesin services utilizing distinct shallow and deep alterations is presented by Papazoglou et al. in [33], [34]. On the basis of research topics, Alam et al. [35] offered a literature review of 60 pertinent studies of evolutionspread in Service Oriented Architecture and Business Process Management (BPM). A method called WSDLDiff was created by Romano and Martin [36] for gathering specific changes made in different iterations of a service, and this data was utilized to track development of 4 services. In order to increase the accuracy of the prediction of security assaults in dynamic, changing environments, Karn et al. [37] have developed an approach for selection and tuning of a machine-learning model.

Facebook researchers have created CT-Scan [40], which can help with regression testing [41], in support of testing mobile apps [39].

In order to determine the set of commits that are semantically connected, Li et al. [42] created CSLICER and formalised the semantic slicing of software version histories.

The following are some of the projects dealing with software and service evolution metrics. Six criteria were proposed by Constantinou et al. [43] to evaluate the architectural development and stability of software projects. Four service evolution patterns were proposed by Wang et al. [44] and used to examine service dependencies, identify changes to services, and calculate customer effect.

A service evolution metric, a service client-code evolution metric, and a service usefulness evolution metric were recently proposed by Kohar et al. [45]. In the same way, we take into account service evolution analysis using the three innovative service adaptive metrics, which yielded insightful conclusions.

FokaefsetalVTracker's technique for XML differencing was used in an actual study to give WSDL evolution analysis [46]. The system can identify and evaluate changes across different WSDL web services' later versions. WSDarwin is a specific differencing mechanism for WSDL and WADL (Web Application Description Language) that creates a set of rules. It was later introduced by Fokaefs et al. [47].

Li et al[48] .'s research on an empirical investigation into the growth of web APIs was followed by

recommendations for application developers. Our tool, AWSCM, may produce interface segments, which can execute portions of a cloud/web service, using change information as opposed to these works. Additionally, AWSCM may retrieve data on cloud/web service evolution that can be used to extract information about the development of a distributed system.

For a multitenant single instance, Mohamed et al. [49] proposed a SaaS evolution platform based on Software Product Lines (SPLs). The platform offers evolution rules based on feature modelling to guide evolution decisions. In their evaluation of the literature on cloud migration, Jamshidi et al. [50] included a comparison between SOA and cloud migration. The SelCSP framework was proposed by Ghosh et al. [51] to make it easier to choose a dependable and capable service provider. A trust management approach called CloudArmor, which is built on a reputation and credibility model, was recently described by Noor et al. [52]. Similar to this, our service daptive metrics produce empirical reports that could be beneficial for choosing a reliable service provider and for the Cloud/SOA migration. Our method can also be useful in maintaining services that support blockchain management [53, [54] and cloud management [55, [56] as they alter and evolve.

6. CONCLUSION

The methods for Alter mining and Adaptive miningof SOA based systems were discussed in the research. The first method assigned evolution labels to a web service action using a service change classifier algorithm before extracting anInterfacesegment. The second method examined the changes of services using three service adaptive metrics and inferred information from a version series. AWSCM, a paradigm and tool for Service Evolution Analytics that helpedboth types of mining of a developing SOA based system, was also covered in this article. To create interfacesegments of 4 services, we carried out four case studies. Using the most used cloud services AWS and KBB, service evolution categorization has been discussed. The capacity of our developed tools is to lower the price of regression testing, and service adaptive metrics. The outcomes showed how the technique can be applied to research the development of cloud services. In order to sustain the QoS demanded by a SLA, our Service Evolution Analytics model was very beneficial in subset regression testing. We intend to use modification and Adaptive mining to additional APIs in the future.

REFERENCES

1. S. A. Frost and M. J. Balas, “Evolving systems and adaptive key com-ponent control,” in *Aerospace Technologies Advancements*. Norderstedt,Germany:BooksonDemand,2010,ch.7,p.115.
2. T. Mens, A. Serebrenik, and A. Cleve, *Evolving Software Systems*. Berlin,Germany:Springer,2014.
3. M.Efatmaneshnik,S.Shoval,andL.Qiao,“Astandarddescriptionofthetermsmoduleandmodularity forsystemsengineering,”*IEEETrans.Eng.Manage.*,vol.67,no.2,pp.365–375,May 2020.
4. Fricke and A. P. Schulz, “Design for changeability (DfC): Principlestoenablechangesinsystemsthroughouttheirentirelifecycle,”*Syst.Eng.*,vol.8,no.4,pp.342–359,2005.
5. M. Ross, D. H. Rhodes, and D. E. Hastings, “Defining changeability:Reconciling flexibility,

adaptability, scalability, modifiability, and robust-ness for maintaining system lifecycle value," *Syst. Eng.*, vol. 11, no. 3,pp.246–262,2008.

6. M. A. Chaumun, H. Kabaili, R. K. Keller, F. Lustman, and G. Saint-Denis,"Design properties and object-oriented software changeability," in *Proc.4thEur.Conf.Softw.MaintenanceReeng.*,2000,pp.45–54.
7. H. P. Breivold, I. Crnkovic, and P. J. Eriksson, "Analyzing softwareevolvability,"in*Proc.32ndAnnu.IEEEInt.Comput.Softw.Appl.Conf.*,2008,pp.327–330.
8. S. V. Loo, G. Muller, T. Punter, D. Watts, P. America, and J. Rutgers, "TheDarwinproject:Evolvabilityofsoftware-intensivesystems,"in*Proc.IEEE Int.WorkshopSoftw.Evolvability*,2007,pp.48–53.
9. M.Böttcher,F.Höppner,andM.Spiliopoulou,"Onexploitingthepowerof time in data mining," *ACM SIGKDD Explorations Newslett.*, vol. 10,no.2,pp.3–11,2008.
10. M.Boettcher,"Contrastandchangemining,"*WileyInterdisciplinaryRev.:DataMiningKnowl.Discovery*,vol.1,no.3,pp.215–230,2011.
11. M.Takaffoli,F.Sangi,J.Fagnan,andO.R.Zäıane,"Communityevolutionmining in dynamic social networks," *Procedia—Social Behavioral Sci.*,vol.22,pp.49–58,2011.
12. Wu,X.Pei,J.Tan,andY.Wang,"Resumeminingofcommunitiesinsocial network," in *Proc. 7th IEEE Int. Conf. Data Mining Workshops*,2007,pp.435–440.
13. T. Srivastava, P. Desikan, and V. Kumar, "Web mining—Concepts, appli-cations and research directions," in *Foundations and Advances in DataMining*.Berlin,Germany:Springer,2005,pp.275–307.
14. H. Chen and M. Chau, "Web mining: Machine learning for web applica-tions,"*Annu.Rev.Inf.Sci.Technol.*,vol.38,pp.289–330,2004.
15. Chaturvedi,"SubsetWSDLtoaccesssubsetserviceforanalysis,"in*Proc.IEEE6thInt.Conf.CloudComput.Technol.Sci.*,2014,pp.688–691.
16. Chaturvedi,"AutomatedwebservicechangemanagementAWSCM—Atool,"in*Proc.IEEE6thInt.Conf.CloudComput.Technol.Sci.*,2014,pp.715–718.
17. Chaturvedi and D. Binkley, "Web service slicing: Intra and inter-operational analysis to test changes," *IEEE Trans. Serv. Comput.*, to bepublished,doi:10.1109/TSC.2018.2821157.
18. Chaturvedi and A. Gupta, "A tool supported approach to performefficientregressiontestingofwebservices,"in*Proc.IEEE7thInt.Symp.MaintenanceEvol.Serv.-OrientedCloud-BasedSyst.*,2013,pp.50–55.
19. Chaturvedi,"Reducingcostinregressiontestingofwebservice,"in*Proc.CSI6thInt.Conf.Softw.Eng.*,2012,pp.1–9.
20. Y. Du, L. Wang, G. Mu, and X. Li, "Dynamic monitoring of serviceoutsourcingfortimedworkflowprocesses,"*IEEETrans.Eng.Manage.*,vol.66,no.4,pp.715–729,Nov.2019.
21. Z.Li,G.Nan,andM.Li,"Advertisingorfreemium:Theimpactsofsocialeffects and service quality on competing platforms," *IEEE Trans. Eng.Manage.*,vol.67,no.1,pp.220–233,Feb.2020.
22. G. Canfora and M. Di Penta, "Testing services and service-centric systems:Challengesandopportunities,"*ITProfessional*,vol.8,pp.10–17,2006.
23. S. Linthicum, "The evolution of cloud service governance," *IEEE CloudComput.*,vol.2,no.6,pp.86–89,Nov./Dec.2015.

24. Binkley,“Theapplicationofprogramslicingtoregressiontesting,”*Inf.Softw.Technol.*,vol.40,no.11/12,pp.583–594,1998.
25. R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, “Web ser-vices description language (WSDL) version 2.0 part 1: Core language,”W3C Recommendation, Jun. 2007. [Online]. Available:https://www.w3.org/TR/wsdl20/
26. Binkley, “Source code analysis: A road map,” in *Proc. Future Softw.Eng.*,2007,pp.104–119.
27. E.Hassan,“Theroadaheadforminingsoftwarerepositories,”in*Proc.FrontiersSoftw.Maintenance*, 2008,pp.48–57.
28. T. Ying, G. C. Murphy, R. Ng, and M. C. Chu-Carroll, “Predictingsourcecodechangesbyminingchangehistory,”*IEEETrans.Softw.Eng.*,vol.30,no.9,pp.574–586,Sep.2004.
29. S.Negara,M.Codoban,D.Dig,andR.E.Johnson,“Miningfine-grainedcodechangestodetectunknownchangepatterns,”in*Proc.36thInt.Conf.Softw.Eng.*,2014,pp.803–813.
30. H. Xiao, J. Guo, and Y. Zou. “Supporting change impact analysis for ser-viceorientedbusinessapplications,”in*Proc.Int.WorkshopSyst.Develop. SOAEnviron.*,2007,p.6.
31. H. Khanh and A. Ghose, “Supporting change propagation in themaintenanceandevolutionofservice-orientedarchitectures,”in*Proc.Asia PacificSoft.Eng.Conf.*,2010,pp.156–165.
32. M.P.Papazoglou,V.Andrikopoulos,andS.Benbernou,“Managingevolv-ingservices,”*IEEESoftw.*,vol.28,no.3,pp.49–55,May/Jun.2011.
33. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, “On the evolu-tion of services,” *IEEE Trans. Softw. Eng.*, vol. 38, no. 3, pp. 609–628,May/Jun.2012.
34. K. A. Alam, R. Ahmad, A. Akhunzada, M. H. N. Md Nasir, and S. U. Khan,“Impact analysis and change propagation in service-oriented enterprises:Asystematicreview,”*Inf.Syst.*,vol.54,pp.43–73,2015.
35. D. Romano and M. Pinzger, “Analyzing the evolution of web servicesusing fine-grained changes,” in *Proc. IEEE 19th Int. Conf. Web Serv.*,2012,pp.392–399.
36. R. Karn, P. Kudva, and I. M. Elfadel, “Dynamic autoselection and autotun-ing of machine learning models for cloud network analytics,” *IEEE Trans.ParallelDistrib.Syst.*,vol.30,no.5,pp.1052–1064,May2019.
37. M. Harman and P. O’Hearn, “From start-ups to scale-ups: Open problemsand challenges in static and dynamic program analysis for testing andverification(keynotepaper),”in*Proc.IEEE18thInt.WorkingConf.SourceCodeAnal.Manipulation*,2018,pp.1–23.
38. J.Dunn,A.Mols,L.Lomax,andP.Medeiros,“Managingresourcesforlarge-scaletesting,”May24,2017.[Online].Available:https:
39. //code.facebook.com/posts/1708075792818517/managing-resourcesfor-large-scale-testing/
40. Reversat,“ThemobiledevicelabatthePrinevilledatacenter,”Jul.13,2016.[Online].Available:https://code.facebook.com/posts/300815046928882/the-mobile-device-lab-at-the-prineville-data-center/
41. Z.Mi,“Mobileperformance:Toolinginfrastructureatfacebook,”Apr.10,2015.[Online].Available: https://code.facebook.com/posts/924676474230092/mobile-performance-tooling-

infrastructure-at-facebook/
42. Y. Li, C. Zhu, J. Rubin, and M. Chechik, "Semantic slicing of softwareversion histories," *IEEE Trans. Softw. Eng.*, vol. 44, no. 2, pp. 182–201,Feb.2018.
43. Constantinou and I. Stamelos, "Architectural stability and evolutionmeasurement for software reuse," in *Proc. 30th Annu. ACM Symp. Appl.Comput.*,2015,pp.1580–1585.
44. S. Wang, W. A. Higashino, M. Hayes, and M. A. M. Capretz, "Serviceevolutionpatterns,"in *Proc. IEEEInt. Conf. Web Serv.*,2014, pp. 201–208.
45. R.KoharandN.Parimala,"Ametricsframeworkformeasuringqualityofawebserviceasitevolves,"*Int.J .Syst.AssuranceEng.Manage.*,vol.8,no.2,pp.1222–1236,2017.
46. M. Fokaefs, R. Mikhaiel, N. Tsantalis, E. Stroulia, and Alex Lau, "Anempiricalstudyonwebserviceevolution,"in*Proc.IEEEInt.Conf.WebServ.*,2011,pp.49–56.
47. M.FokaefsandE.Stroulia,"WSDarwin:Studyingtheevolutionofweb service systems," in *Advanced Web Services*, New York, NY, USA:Springer,2014,pp.199–223.
48. J. Li, Y. Xiong, X. Liu, and L. Zhang, "How does web service APIevolution affect clients?" in *Proc. IEEE 20th Int. Conf. Web Serv.*, 2013,pp.300–307.
49. Mohamed, M. Abu-Matar, R. Mizouni, M. Al-Qutayri, and Z. A.Mahmoud, "SAAS dynamic evolution based on model-driven softwareproductlines,"in*Proc.IEEE6thInt.Conf.CloudComput.Technol.Sci.*,2014,pp.292–299.
50. P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: A systematicreview,"*IEEETrans.CloudComput.*,vol.1,no.2,pp.142–157,Jul.–Dec.2013.
51. N.Ghosh,S.K.Ghosh,andS.K.Das,"SelCSP:Aframeworktofacilitateselectionofcloudserviceproviders,"*IEEETrans.CloudComput.*,vol.3,no.1,pp.66–79,Jan./Mar.2015.
52. T.H.Noor,Q.Z.Sheng,L.Yao,S.Dustdar,andA.H.H.Ngu,"CloudArmor: Supporting reputation-based trust management for cloudservices,"*IEEETrans.ParallelDistrib.Syst.*,vol.27,no.2,pp.367–380,Feb.2016.
53. Fortino,F.Messina,D.Rosaci,andG.M.L.Sarne,"Usingblockchain in a reputation-based model for grouping agents in the internetofthings,"*IEEETrans.Eng.Manage.*,tobepublished,doi:10.1109/TEM.2019.2918162.
54. M. Kuperberg, "Blockchain-based identity management: A survey fromtheenterpriseandecosystemperspective,"*IEEETrans.Eng.Manage.*,tobepublished,doi:10.1109/TEM.2019.2926471.
55. S. Singh and S. C. Misra, "Exploring the challenges for adopting the cloudPLM in manufacturing organizations," *IEEE Trans. Eng. Manage.*, to bepublished,doi:10.1109/TEM.2019.2908454.
56. R. C. Basole and H. Park, "Interfirm collaboration and firm value insoftwareecosystems:evidencefromcloudcomputing,"*IEEETrans.Eng.Manage.*,vol.66,no.3,pp.368–380,Aug.2019.
57. T. Apiwattanapong, A. Orso, and M. J. Harrold, "JDiff: A differencingtechnique and tool for object-oriented programs," *J. Autom. Soft. Eng.*,vol.14,no.1,pp.3–36,Mar.2007.
58. T.Apiwattanapong,A.Orso,andM.J.Harrold,"Adifferencingalgorithmfor object-oriented programs," in *Proc. 19th IEEE Int. Conf. Autom. Softw.Eng.*,2004,pp.2–13.
59. "JDiff,"Oct.2019.[Online].Available:http://javadiff.sourceforge.net/
60. "MembraneSOAModel,"Mar.2017.[Online].Available:http://membrane-soa.org/soa-model/
61. "AWSEC2,"Mar.2017.[Online].Available:https://github.com/chilts/awssum-amazon-ec2

62. “AWSEC2Release,”Mar.2017.[Online].Available:https://aws.amazon.com/releasenotes/Amazon-EC2
63. “Eucalyptus,”Mar.2017.[Online].Available:https://github.com/eucalyptus/eucalyptus
64. [Online]. Available: Jeff Barr, (August 25, 2006). “Amazon EC2 Beta”.
65. https://aws.amazon.com/blogs/aws/amazon_ec2_beta/. Link accessed on February 2019.
66. Jeff Barr, (October 16, 2007). “Amazon EC2 Gets More Muscle”. Link accessed on February 2019.
67. [Online]. Available: Jeff Barr, (May 29, 2008). “More EC2 power”.
68. https://aws.amazon.com/blogs/aws/more-ec2-power/. Link accessed on February 2019.