HEIDELBERG LAUREATE FORUM

IIIT Dharwad ज्ञानेन विकासः

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

acm India Council

Pingala Interactions In Computing

Cloud Service Life-cycle Management

Dr. Animesh Chaturvedi

Assistant Professor: IIIT Dharwad

Young Researcher: Pingala Interaction in Computing

Young Researcher: Heidelberg Laureate Forum

Postdoc: King's College London & The Alan Turing Institute

PhD: IIT Indore MTech: IIITDM Jabalpur

KING'S College LONDON

The Alan Turing Institute

Indian Institute of Technology Indore
भारतीय प्रौद्योगिकी संस्थान इंदौर

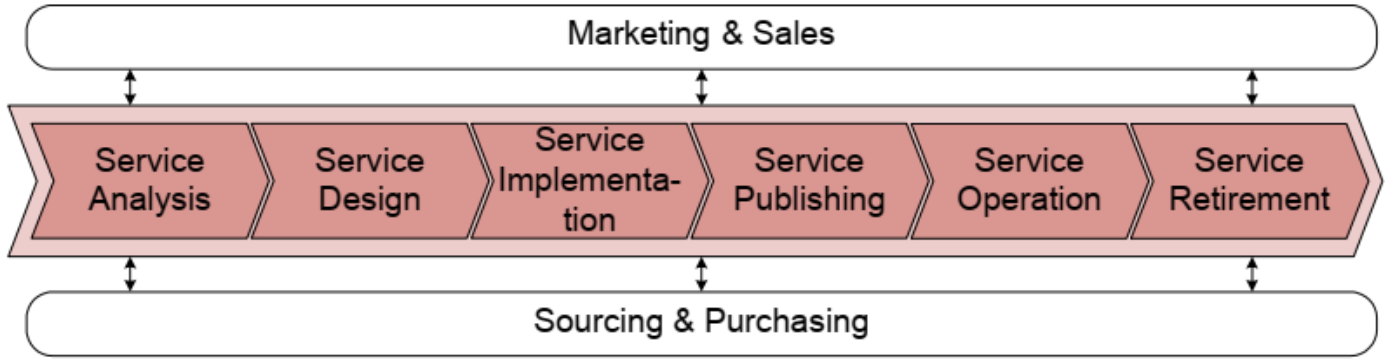
PDPM
Indian Institute of Information Technology, Design and Manufacturing, Jabalpur

Cloud Service Life-cycle

1. **Cloud Service Life-Cycle**
2. Cloud Deployment Scenarios
3. Cloud Service Development and Testing
4. Web Service Slicing for Regression Testing of Services
5. Cloud Service Evolution Analytics
6. Quality of Service and Service Level Agreement

Cloud Service Life-Cycle

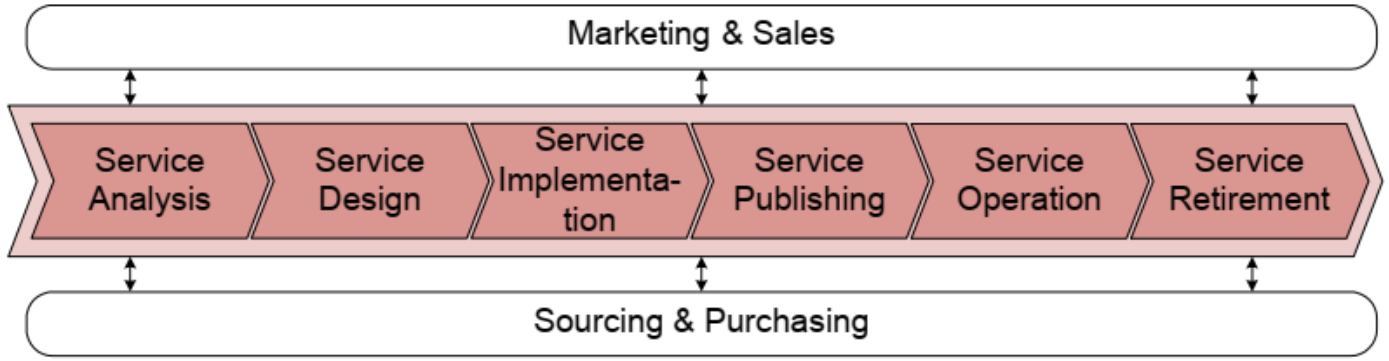
- Service Analysis: Identification and contextualisation - market requirements – Service Reuse
  - strategy maps, process models, data models, application diagrams
- Service Design:
  - detailed model of the service,
  - specification for development and reuse,
  - refining the service idea for implementation



Kohlborn, Thomas, Axel Korthaus, and Michael Rosemann. "Business and software service lifecycle management." *2009 IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2009.

Cloud Service Life-Cycle

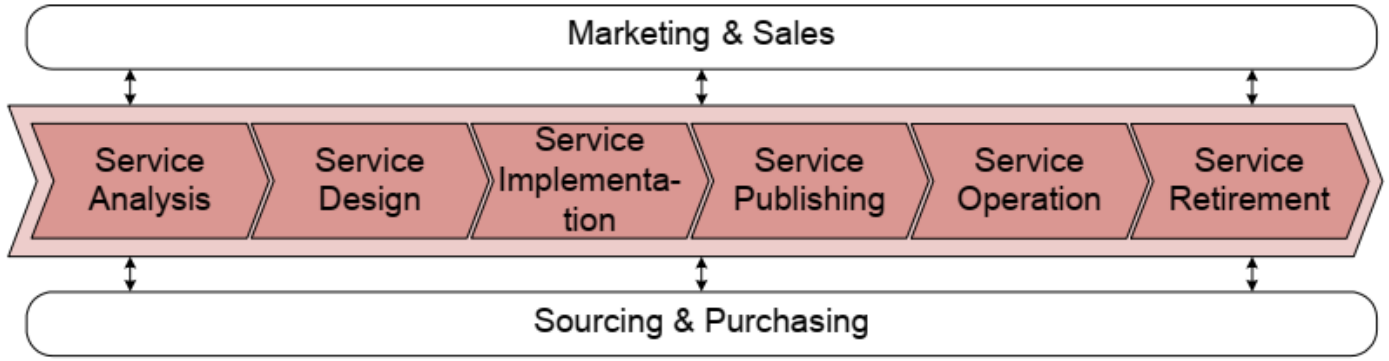
- Service Implementation: either software with technical service characteristics or executable non-technical service.
  - application package or supporting software services
  - traditional software implementation activities
- Service Publishing: dissemination of the service,
  - tasks of registering a service in marketplaces and repositories
  - access rights, costs, pricing models and SLAs



Kohlborn, Thomas, Axel Korthaus, and Michael Rosemann. "Business and software service lifecycle management." *2009 IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2009.

Cloud Service Life-Cycle

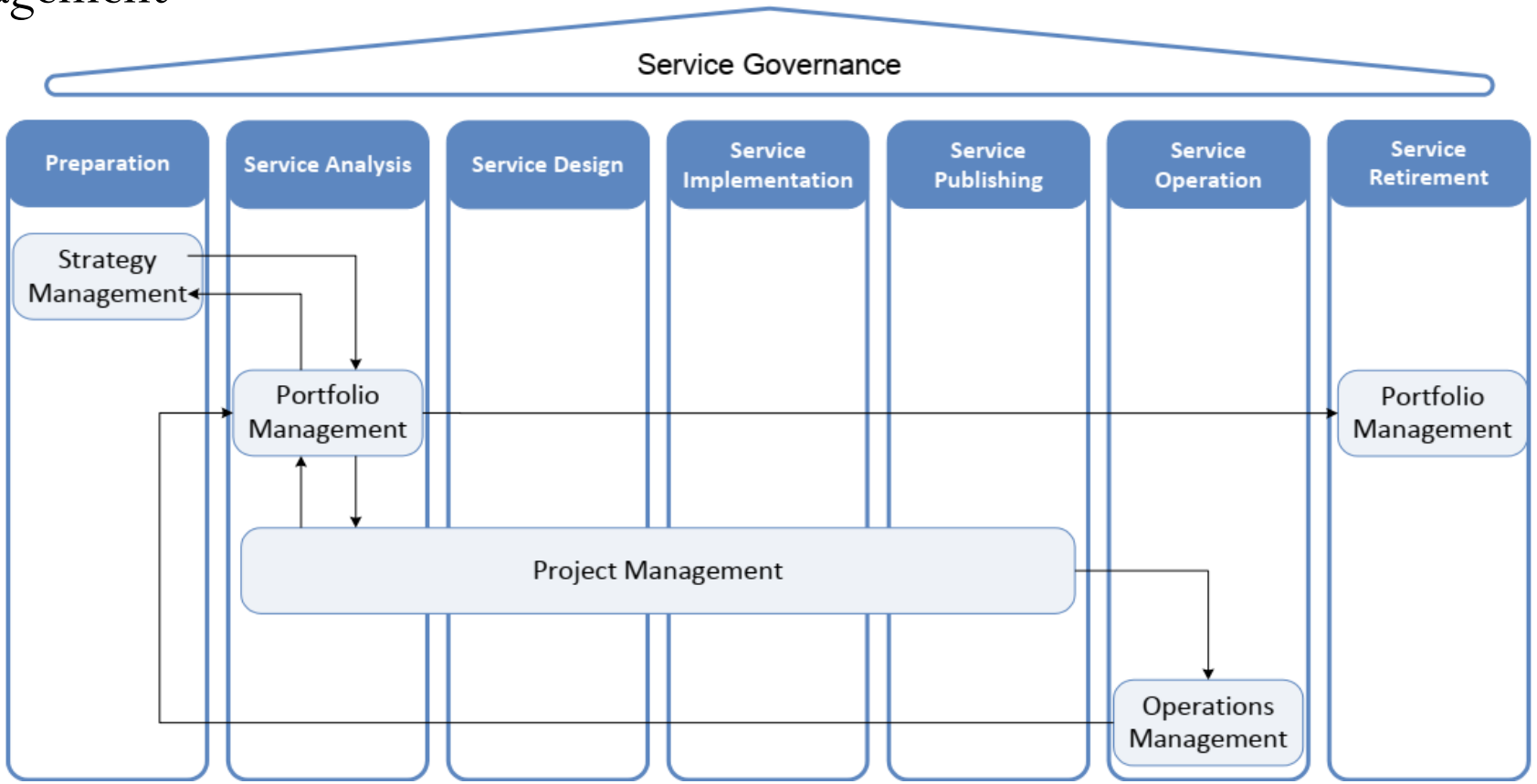
- Service Operation: monitored for contract management,
  - SLA compliance and billing, Service Performance Management,
  - Service Relationship Management, *Service Maintenance and Improvement*
- Service Retirement: reached end of economic or technical competitiveness, contractual activities and succession planning
  - new service to replace the retiring service?



Kohlborn, Thomas, Axel Korthaus, and Michael Rosemann. "Business and software service lifecycle management." *2009 IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2009.

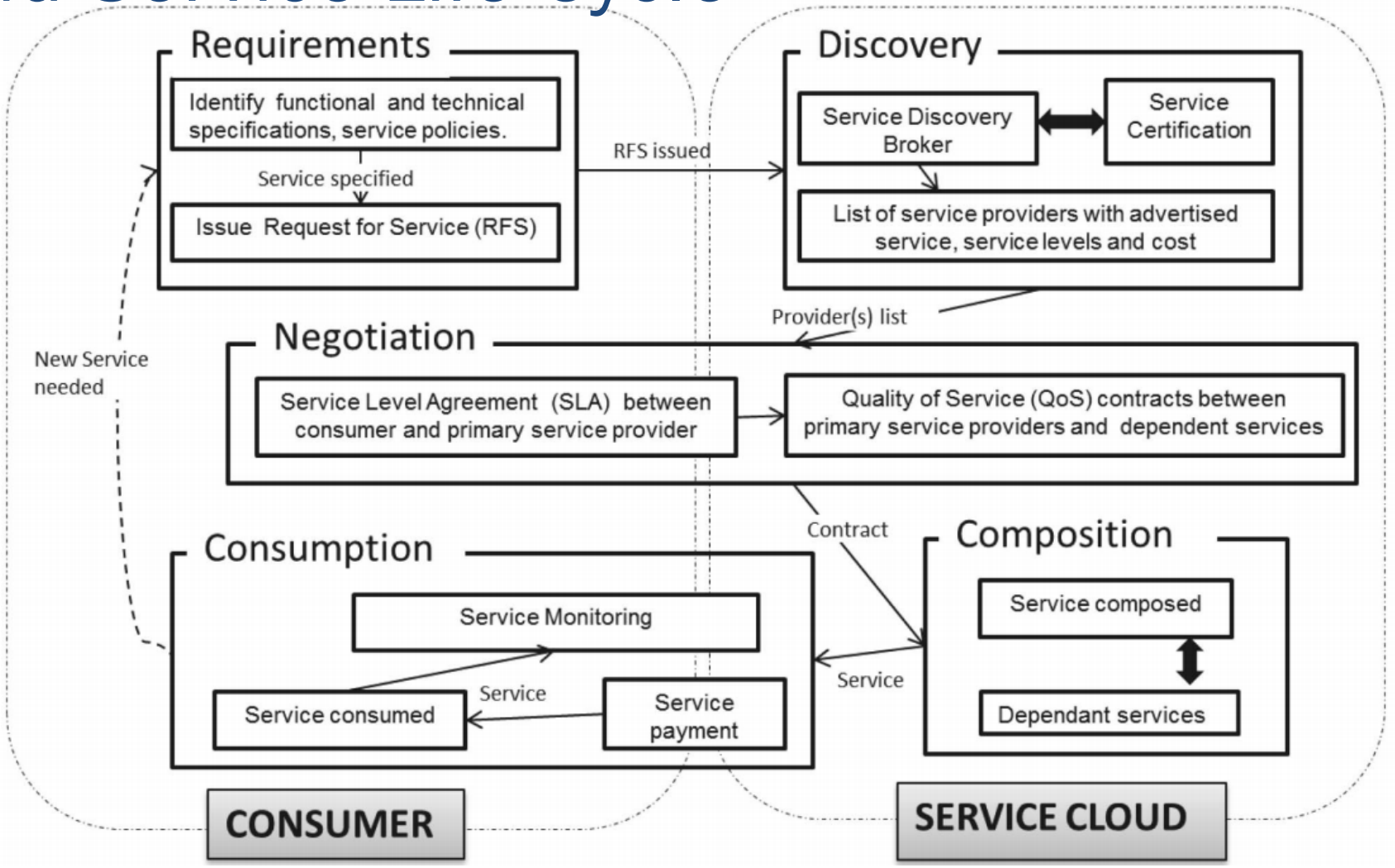
Cloud Service Life-Cycle

- Integration and mapping between the service lifecycle phases and the generic management



Kohlborn, Thomas, Axel Korthaus, and Michael Rosemann. "Business and software service lifecycle management." *2009 IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2009.

Cloud Service Life-Cycle



Joshi, Karuna P., Yelena Yesha, and Tim Finin. "Automating cloud services life cycle through semantic technologies." *IEEE Transactions on Services Computing* 7.1 (2012): 109-122.

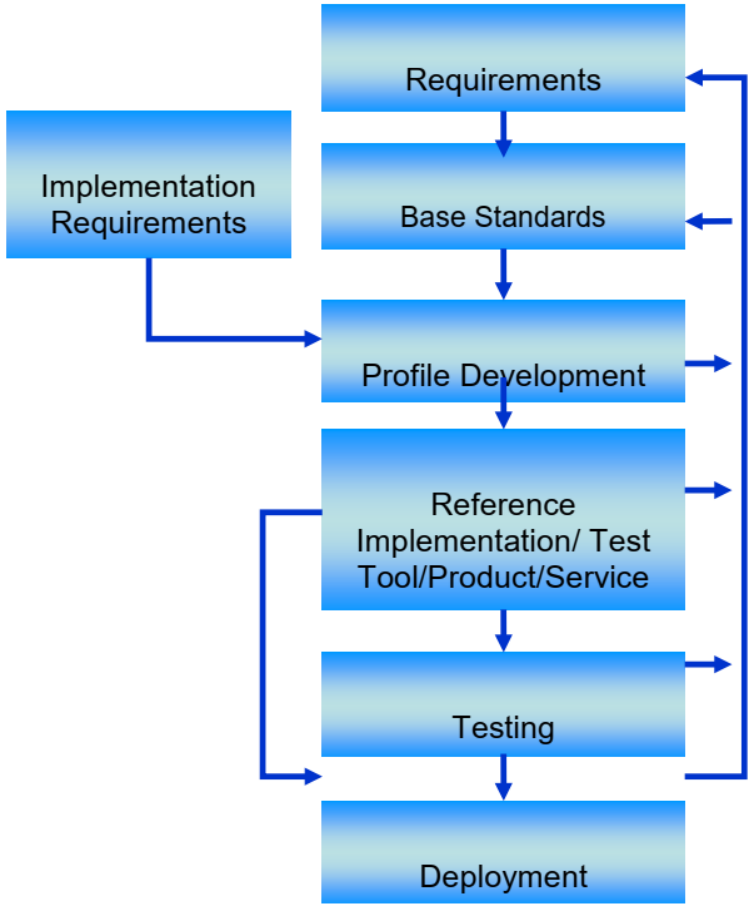
Cloud Service life cycle

- The processes and data flow of the five phases:
  - Requirements,
  - Discovery,
  - Negotiation,
  - Composition, and
  - Consumption
- Represent the concepts and relationships for each phase

Joshi, Karuna P., Yelena Yesha, and Tim Finin. "Automating cloud services life cycle through semantic technologies." *IEEE Transactions on Services Computing* 7.1 (2012): 109-122.

IT Standards Life Cycle

- High-level conceptualization of IT standards with development, products, processes, and services are deployed processes can occur concurrently



Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

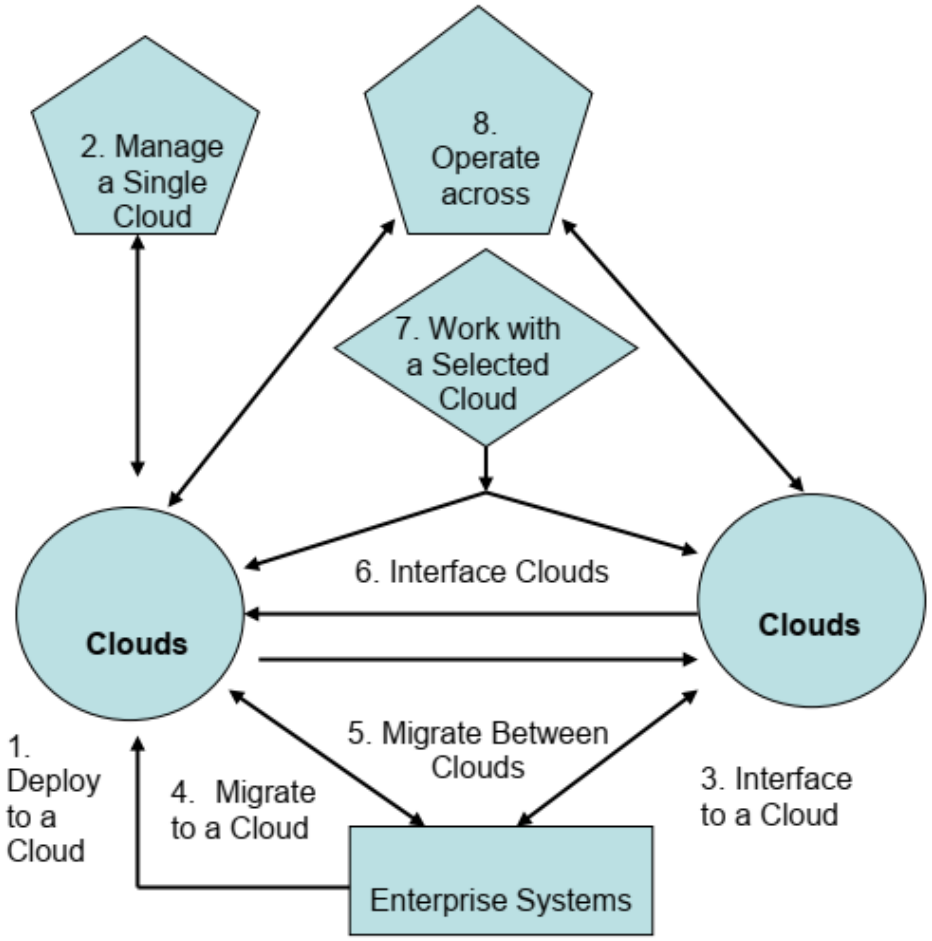
Cloud Deployment Scenarios

1. Cloud Life Cycle
2. **Cloud Deployment Scenarios**
3. Cloud Service Development and Testing
4. Web Service Slicing for Regression Testing of Services
5. Cloud Service Evolution Analytics
6. Quality of Service and Service Level Agreement

High-Level Generic Scenarios

Three scenarios from which interaction scenarios are derived

- Single Cloud System
- Multiple Cloud Systems
  - (serially, one at a time)
- Multiple Cloud Systems
  - (simultaneously, more than one at a time)



Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

High-Level Generic Scenarios

Three scenarios from which interaction scenarios are derived

- Single Cloud System
  - Scenario 1: Deployment on a single cloud system
  - Scenario 2: Manage resources on a single cloud system
  - Scenario 3: Interface enterprise systems to a single cloud system
  - Scenario 4: Enterprise systems migrated or replaced on a single cloud system
- Multiple Cloud Systems (serially, one at a time)
  - Scenario 5: Migration between cloud systems
  - Scenario 6: Interface across multiple cloud systems
  - Scenario 7: Work with a selected cloud system
- Multiple Cloud Systems – (simultaneously, more than one at a time)
  - Scenario 8: Operate across multiple cloud systems

Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

Deployment Cases for High Level Scenarios

Cloud computing Use-cases

- Centralized vs. Distributed, and
- Within vs. Crossing Trust Boundaries

	a.) Within Trust Boundary	b.) Crossing Trust Boundary
1.) Centralized i.e., one administrative cloud domain	Deployment Case 1A	Deployment Case 1B
2.) Distributed, i.e., crossing administrative cloud domains	Deployment Case 2A	Deployment Case 2B

Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

Deployment Cases for High Level Scenarios

Case 1: Each cloud provider → multiple cloud consumers with client-provider interaction with the provider.

Case 2: Cloud consumer application → distributed across two or more cloud providers and administrators simultaneously. Cloud consumer → consumer-provider interactions

	a.) Within Trust Boundary	b.) Crossing Trust Boundary
1.) Centralized i.e., one administrative cloud domain	Deployment Case 1A	Deployment Case 1B
2.) Distributed, i.e., crossing administrative cloud domains	Deployment Case 2A	Deployment Case 2B

Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

Deployment Cases for High Level Scenarios

Case 1A: private cloud within a single administrative domain and trust boundary

Case 1B: Commercial public cloud within a single administrative domain, outside of any trust boundary that a client could use to enforce policy and governance

	a.) Within Trust Boundary	b.) Crossing Trust Boundary
1.) Centralized i.e., one administrative cloud domain	Deployment Case 1A	Deployment Case 1B
2.) Distributed, i.e., crossing administrative cloud domains	Deployment Case 2A	Deployment Case 2B

Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

Deployment Cases for High Level Scenarios

Case 2A: Federated cloud of two or more administrative cloud domains, the cloud providers can agree to mutually enforce policy and governance – in a common trust boundary.

Case 2B: Hybrid cloud applications cross a private-public trust boundary, or multiple public clouds, where both administrative domains and trust boundaries are crossed.

	a.) Within Trust Boundary	b.) Crossing Trust Boundary
1.) Centralized i.e., one administrative cloud domain	Deployment Case 1A	Deployment Case 1B
2.) Distributed, i.e., crossing administrative cloud domains	Deployment Case 2A	Deployment Case 2B

Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

Scenarios and Technical Requirements

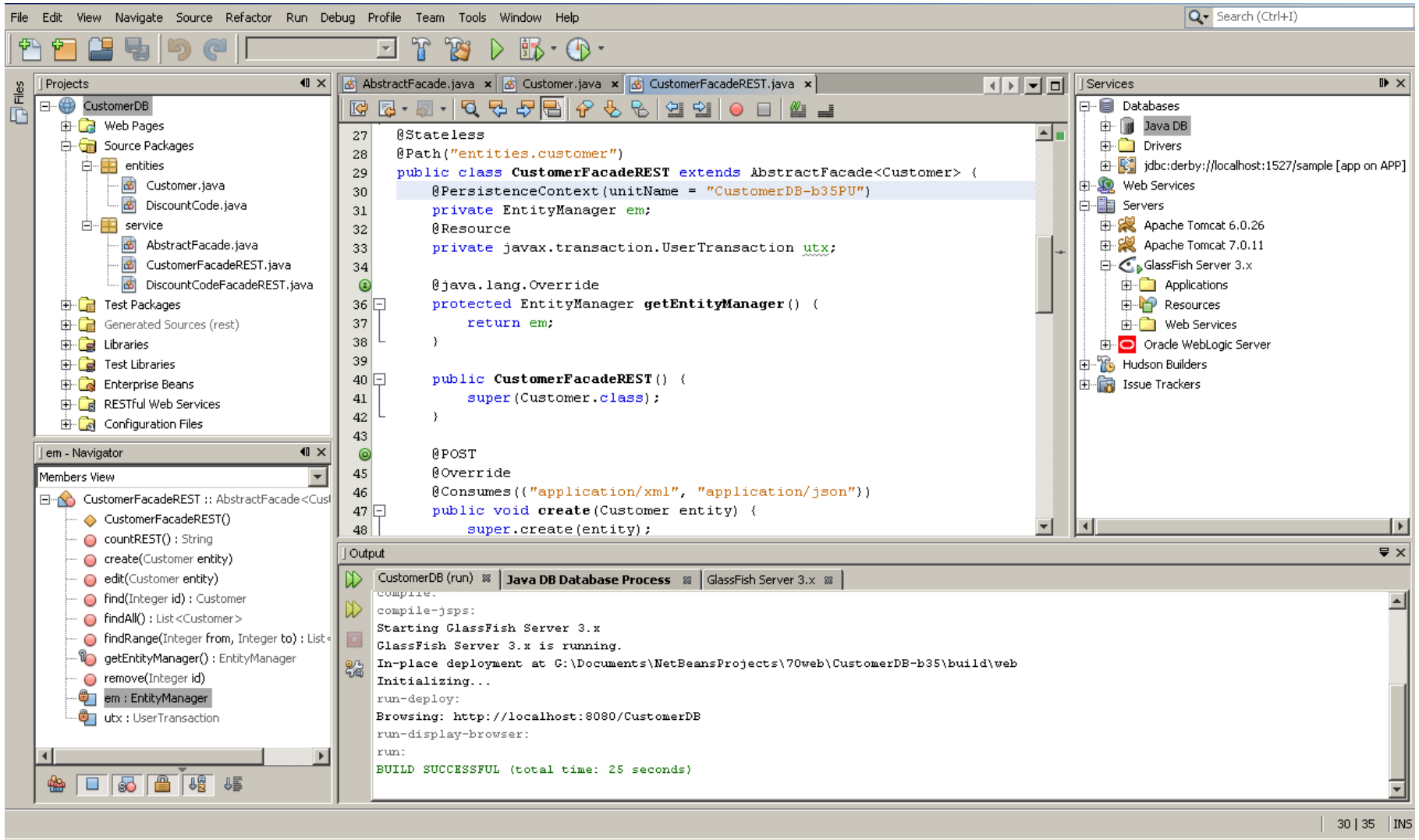
Scenarios	Technical Requirements
1.	Creating, accessing, updating, deleting data objects in cloud systems;
2.	Moving VMs and virtual appliances between cloud systems;
3.	Selecting the best IaaS vendor for private externally hosted cloud system;
4.	Tools for monitoring and managing multiple cloud systems;
5.	Migrating data between cloud systems;
6.	Single sign-on access to multiple cloud systems;
7.	Orchestrated processes across cloud systems;
8.	Discovering cloud resources;
9.	Evaluating SLAs and penalties; and
10.	Auditing cloud systems.

Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.

Cloud Service Development and Testing

1. Cloud Life Cycle
2. Cloud Deployment Scenarios
3. **Cloud Service Development and Testing**
4. Web Service Slicing for Regression Testing of Services
5. Cloud Service Evolution Analytics
6. Quality of Service and Service Level Agreement

NetBeans Web Services Development



https://netbeans.org/features/java-on-server/web-services.html

NetBeans Web Services Development

NetBeans IDE supports Web services standards from Java Enterprise Edition (Java EE), including the

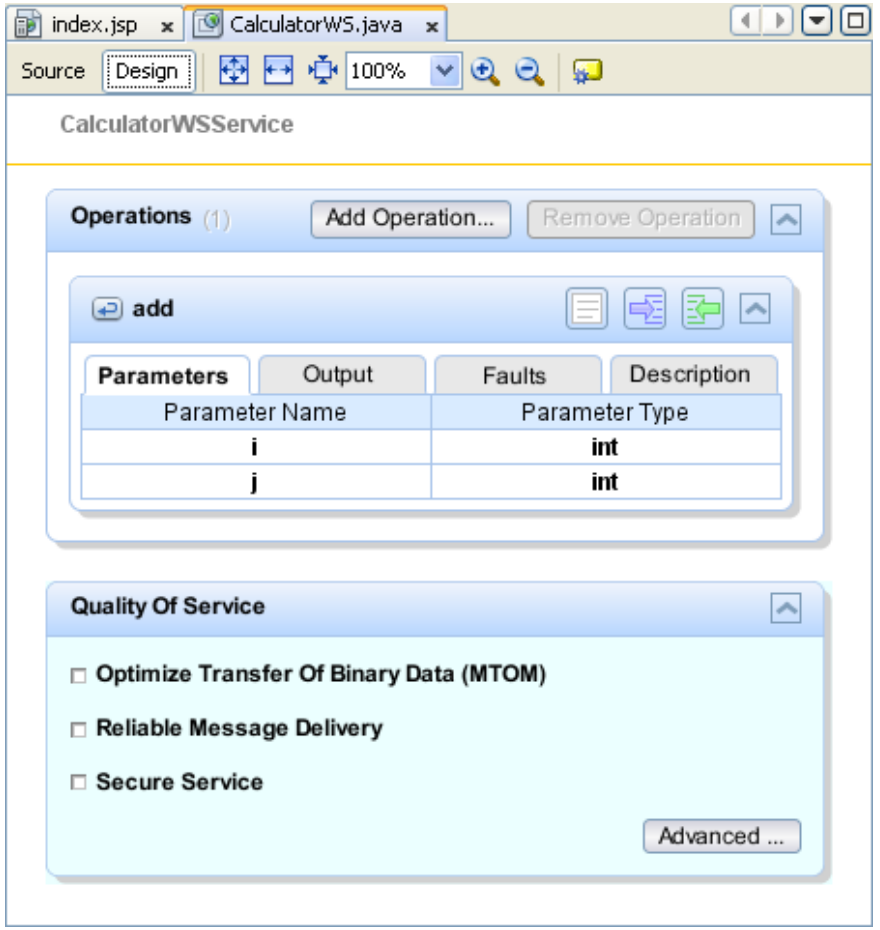
- Jakarta XML Web Services (JAX-WS),
- Jakarta RESTful Web Services (JAX-RS), and
- Jakarta XML Binding (JAXB) web service standards.

https://netbeans.org/features/java-on-server/web-services.html

NetBeans SOAP-based Web Services

Create and develop WSs from Java classes and WSDL files

- Java classes annotated with @javax.jws.WebService to automatically recognized as WSs in a project
- Convert SOAP based web services to RESTful service resources by using the action available in the web service node. Use the Web Service Customization editor to create asynchronous web service clients.

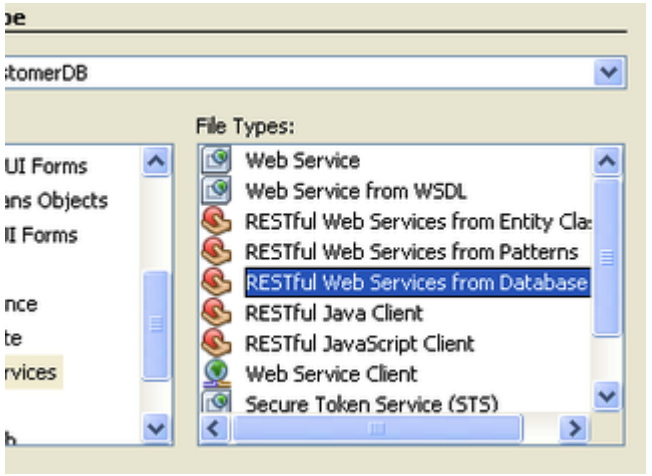


https://netbeans.org/features/java-on-server/web-services.html

NetBeans RESTful Web Services

(JAX-RS 2.0) RESTful web services from

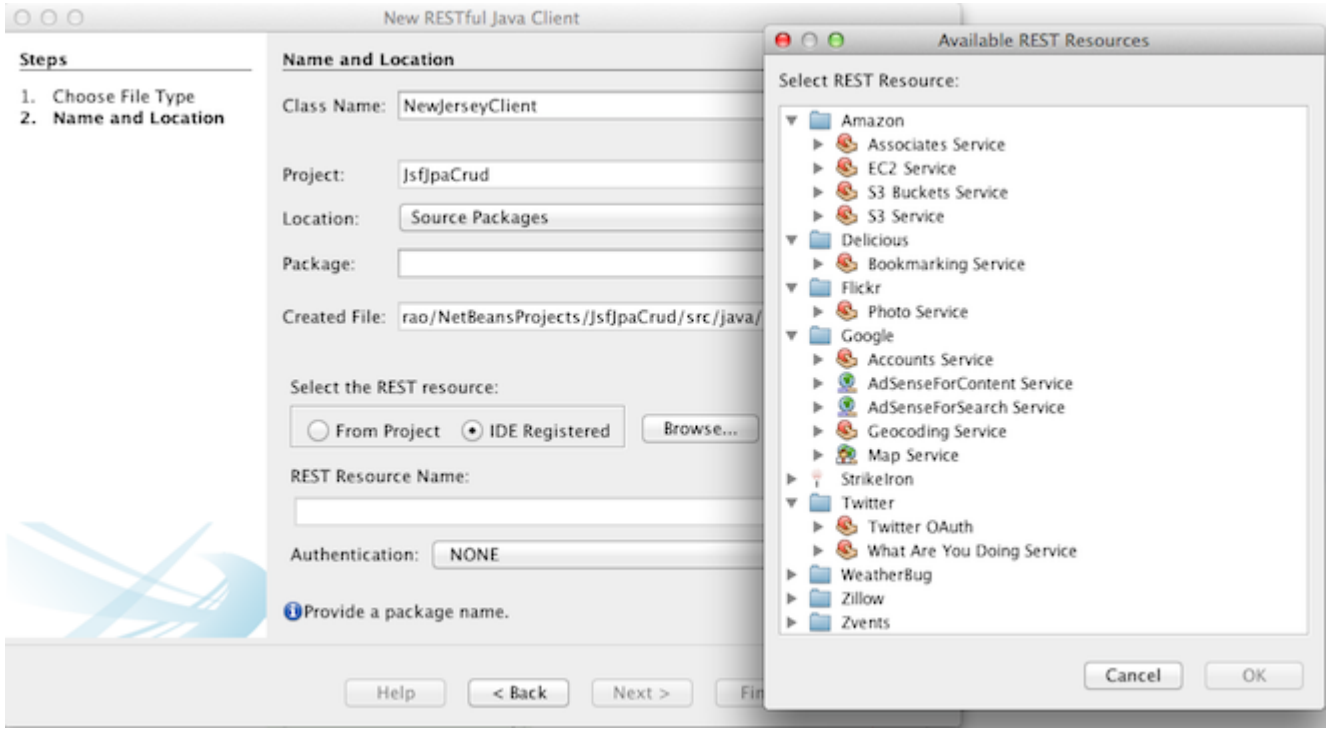
- Java Persistence API (JPA) entity classes and patterns, or even directly from a database
- RESTful WSs wrap entity beans and provide CRUD functionality



https://netbeans.org/features/java-on-server/web-services.html

NetBeans Web Service Clients

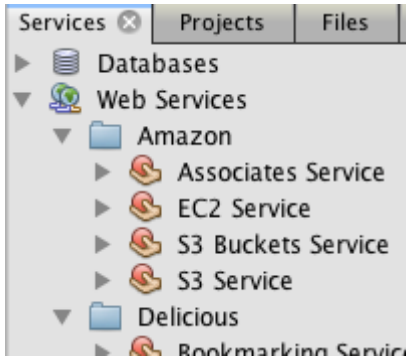
- Generate code for invoking WSs (both RESTful and SOAP-based), such as JavaScript client stubs from WADL.
- Generate RESTful Java clients in Java Web and Java applications
- Create server-side mash-up applications and add services from their web service descriptor files (WSDL or WADL).
- Drag and drop service operations into a POJO, Servlet, JSP, JSF, or PHP page, and the IDE will generate the access code.

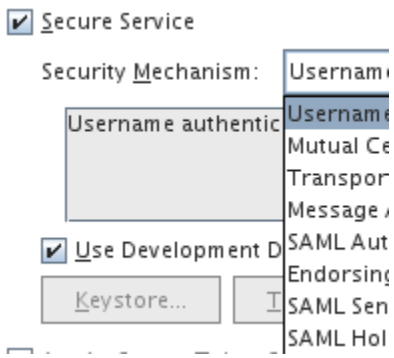


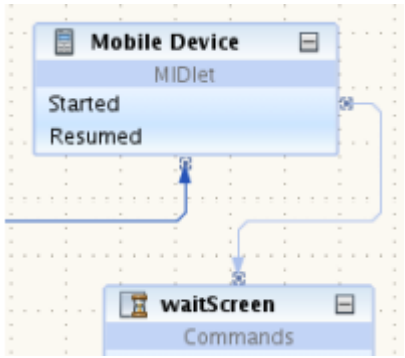
https://netbeans.org/features/java-on-server/web-services.html

NetBeans Web Service Clients

- Access Web Services: Web Service Manager to access popular RESTful Web APIs provided by Google, Facebook, Yahoo, flickr, Amazon, Twitter and many more.
- Web Service Interoperability Technologies (WSIT)
- Mobile Web Services







https://netbeans.org/features/java-on-server/web-services.html

Eclipse Web services Tools

- J2EE Standard Tools (JST) WSs component contains tools for
  - developing and interacting with Java Web services.
  - extensible WSs wizards for creating Web service and Web services client wizards for consuming Web service,
  - Web services Ant tasks for creating and consuming Web services,
  - wizard extensions for the Apache Axis v1.4 and Apache Axis2 Web service runtimes.

Integrated Development Environment (IDE)	Server support
NetBeans	GlassFish Server Open-Source Edition, Apache Tomcat etc.
Eclipse	Apache Axis, Apache Tomcat etc.

https://www.eclipse.org/webtools/ws/

Eclipse Web services Tools

The WST Web services component contains tools for Web services development which is not Java specific. It consists of:

- Web services preferences pages,
- Web services frameworks such as the creation framework and finder framework,
- Web Services Explorer, a Web application that let you discover and publish to UDDI, and invoke a WSDL/WSIL via native XML.
- WSDL model
- WSDL Editor
- WSDL and WS-I validator

https://www.eclipse.org/webtools/ws/

Apache JMeter Testing Framework

- Java application designed
  - to load test functional behavior,
  - measure performance,
  - testing Web Applications,
  - static and dynamic resources, Web dynamic applications,
  - simulate a heavy load on a server, group of servers, network or object to test its strength
  - analyze overall performance under different load types.
- Ability to extract data from HTML, JSON , XML or any textual format.

https://jmeter.apache.org/

Apache JMeter features

- Apache JMeter features include:
- Ability to load and performance test many different applications/server/protocol types:
  - Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)
  - **SOAP / REST Webservices**
  - FTP
  - Database via JDBC
  - LDAP
  - Message-oriented middleware (MOM) via JMS
  - Mail - SMTP(S), POP3(S) and IMAP(S)
  - Native commands or shell scripts
  - TCP
  - Java Objects

https://jmeter.apache.org/

Apache JMeter Testing Framework

- Apache JMeter test performance both on static and dynamic resources, Web dynamic applications.
- load test functional behavior and measure performance
- Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)
- SOAP / REST Webservices

https://jmeter.apache.org/

JMeter WebService Test Plan

File → *Templates* select Building a SOAP Webservice Test Plan

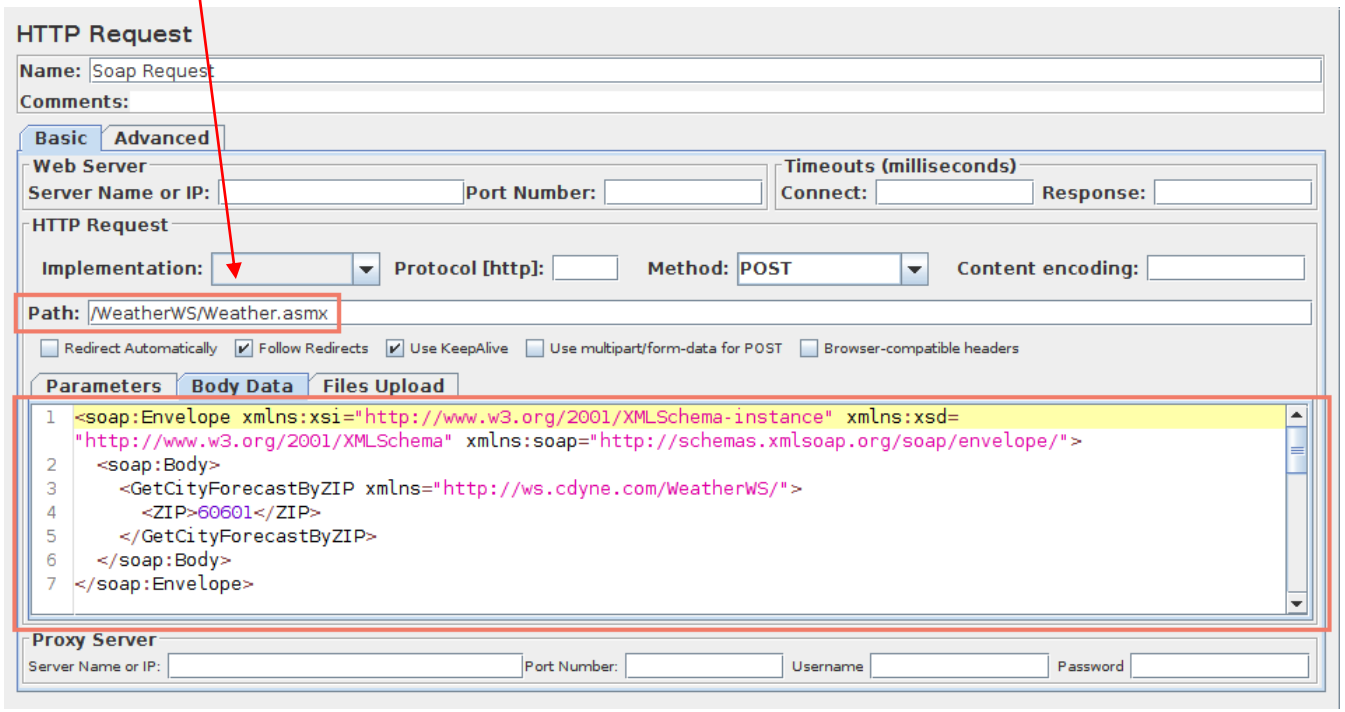
In "HTTP Request Defaults" change "Server Name of IP"

In "Soap Request", change "Path:“

Testing a REST Webservice similar modify in HTTP Request

Method: to select the one you want to test

Body Data: JSON, XML or any custom text



https://jmeter.apache.org/usermanual/build-ws-test-plan.html

SoapUI Testing Framework

- open source or commercial, SoapUI testing tools
- to create, manage, and execute end-to-end tests on REST, SOAP, & GraphQL APIs, JMS, JDBC, and other web services

For developers and testers

- to deliver REST & SOAP APIs,
- SoapUI Open Source is the simplest and easiest way
- API testing
- tool built for validation of REST, SOAP, GraphQL, microservices, and other back-end services.

https://www.soapui.org/

SoapUI API Testing Tool

Scriptless Functional Testing: create and run complex scenarios

Security Testing: tests and scans, protect services on websites from security vulnerabilities

Load Testing: existing functional API tests

API Mocking: Mimic Real Web Services

Protocol Support:

- Simple Object Access Protocol (SOAP)/Web Services Description Language (WSDL),
- Representational state transfer (REST),
- JMS (Java Message Service)

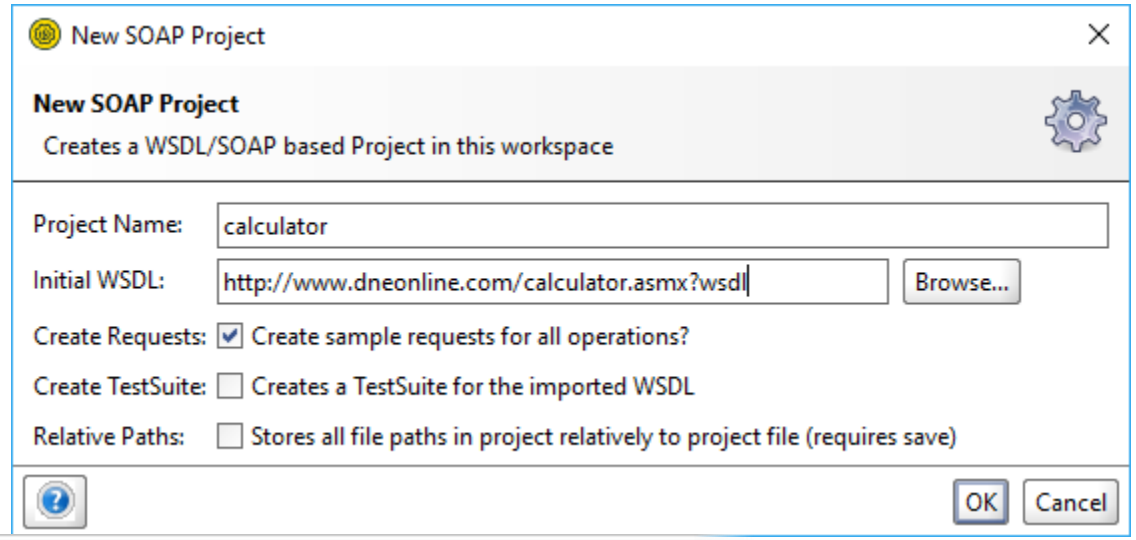
Vibrant Open Source Community Ecosystem

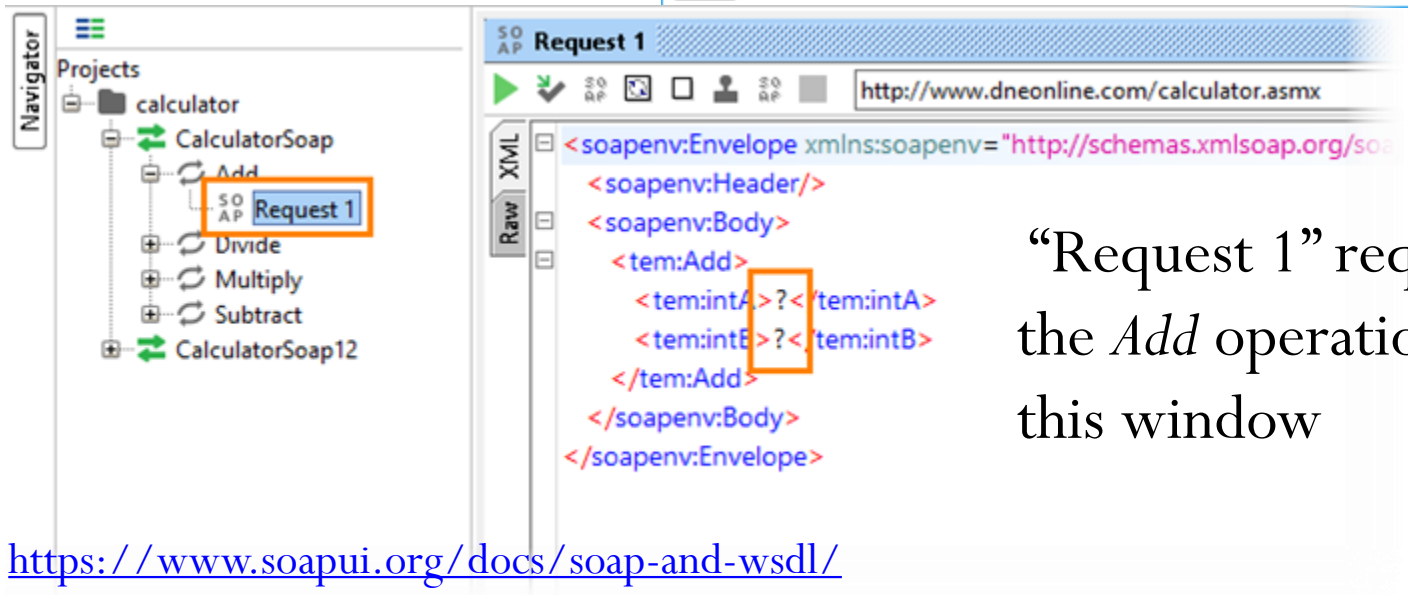
https://www.soapui.org/tools/soapui/

SoapUI Testing Framework

File menu option

- "New Project"
- import of WSDLs
- Or paste WSDL path
- Extract and parse WSDL





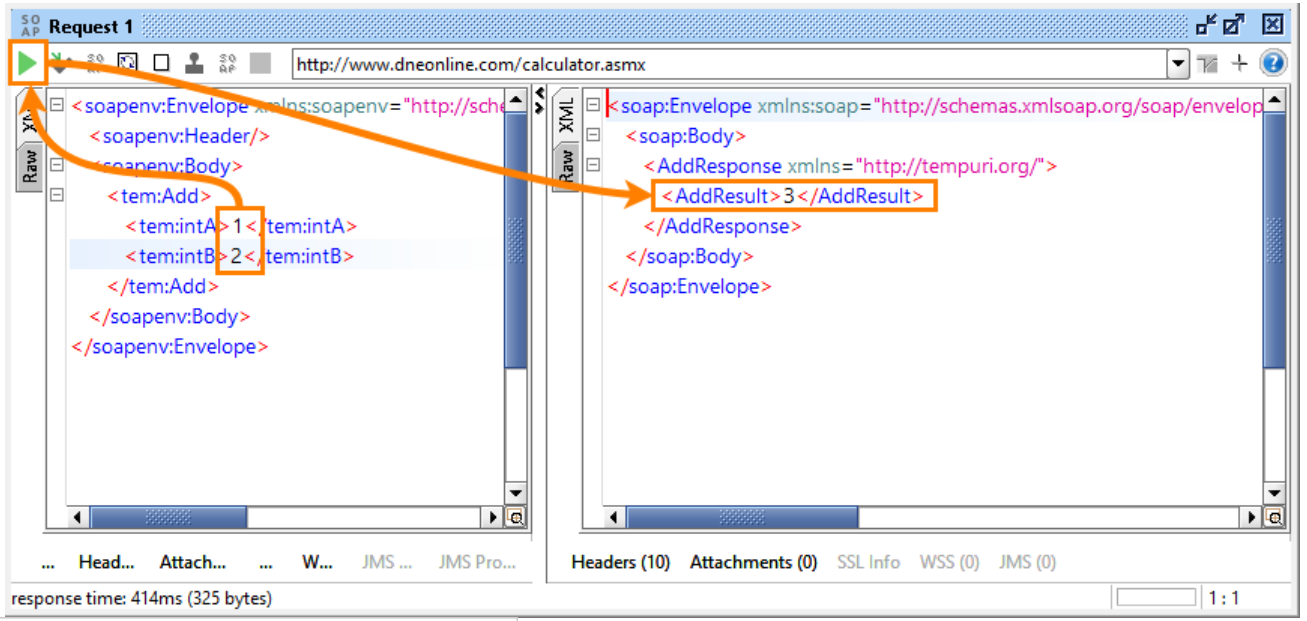
"Request 1" request generated for the *Add* operation and double-click it for this window

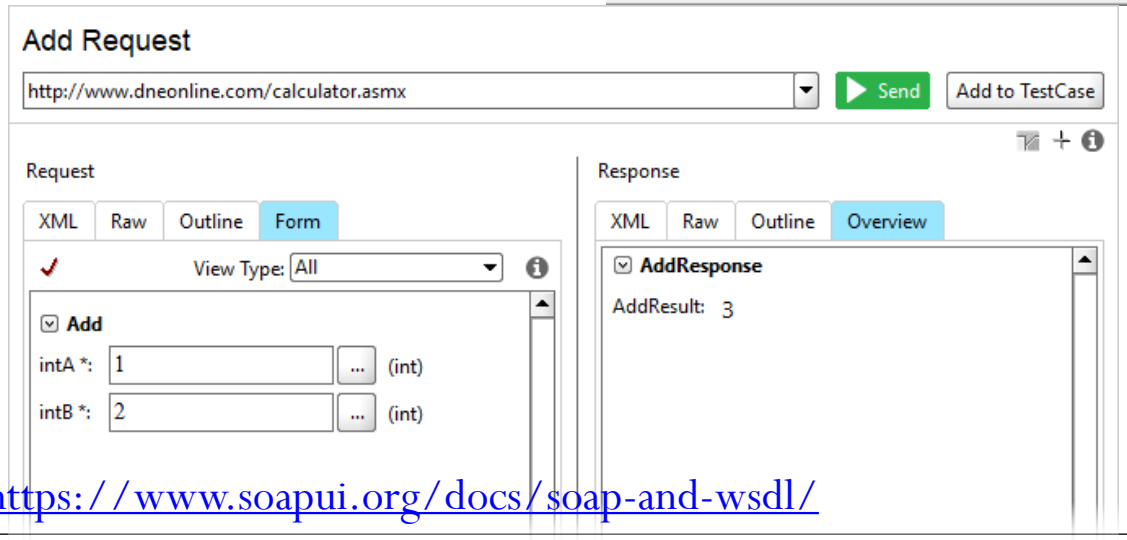
https://www.soapui.org/docs/soap-and-wsdl/

SoapUI Testing Framework

- enter two integer values (as request values 1+2) and press the green arrow

- return response 3





https://www.soapui.org/docs/soap-and-wsdl/

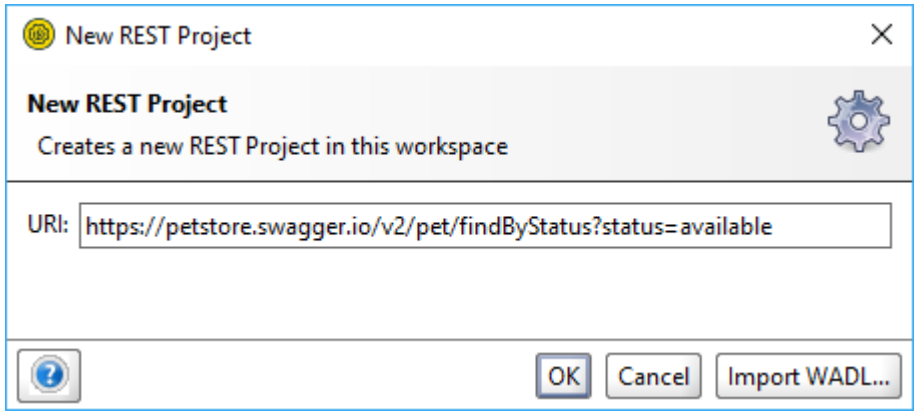
Alternate to XML syntax,

- Form view the request and Overview for the response

SoapUI RESTful API

File > New Rest Project.

paste URL and click **OK**



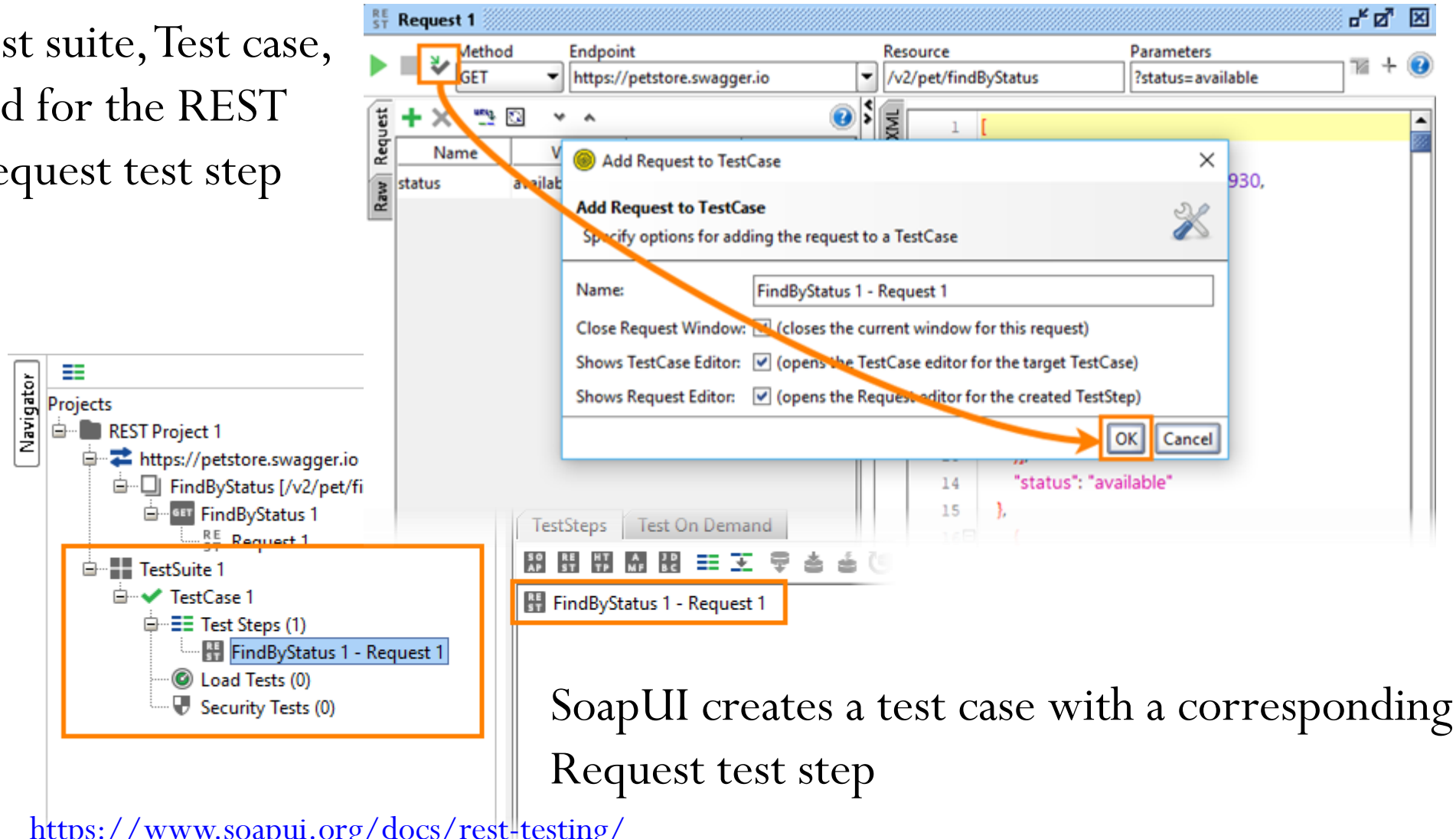
REST HTTP requests are:

GET	Read or retrieve data
POST	Add new data
PUT	Update data that already exists
DELETE	Remove data

https://www.soapui.org/docs/rest-testing/

SoapUI RESTful API

Test suite, Test case, and for the REST Request test step



SoapUI creates a test case with a corresponding REST Request test step

https://www.soapui.org/docs/rest-testing/

Web Service Slicing for Regression Testing of Services (Intra and Inter Operational Analysis)

1. Cloud Life Cycle
2. Cloud Deployment Scenarios
3. Cloud Service Testing
4. **Web Service Slicing for Regression Testing of Services**
5. Cloud Service Evolution Analytics
6. Quality of Service and Service Level Agreement

Animesh Chaturvedi, and David Binkley. "Web Service Slicing: Intra and Inter-Operational Analysis to Test Changes." **IEEE Transactions on Services Computing** (2018).

Web Service Slicing: Intra and Inter Operational Analysis to Test Changes

1. INTRODUCTION
2. PROPOSED DEFINITIONS
3. INTRA/INTER OPERATIONAL CHANGE ANALYSIS
4. OPERATIONALIZED AND PARAMETERIZED REGRESSION TESTING OF WEB SERVICES
5. EXPERIMENTS: CASE STUDIES ON WEB SERVICES
6. SUMMARY

INTRODUCTION

Analogy between

Object Oriented Programming → Service Oriented Computing

Functional unit of

API is a Class → Cloud Service is a Web service

Atomic unit of

Class is a Method (Function or Procedure) → Web service is an Operation

Method<variables> → Operation<parameters>

Abstraction unit of

Interface → Web Service Descriptive Language (WSDL)

Issues

1. Operational analysis for Large Web services.
2. Program slicing of Large Web Services.
3. Regression testing to Large Web services.

Contributions

- *Web Service Slicing*, a technique that captures a functional subset of a *large-scale web service* using an *Interface slice*
- *Interface slice* captured as a *WSDL slice* (a subset of a service's WSDL) provides access to an *Interoperable slice*
- *Intra-operational analysis:* inspects a single operation,
- *Inter-operational analysis:* inspects relationships between operations and procedures
- *Associative code-test mapping* for
- *Operationalized Regression Testing of Web Services* (ORTWS)
- *Parameterized Regression Testing of Web Services* (PRTWS)

PROPOSED DEFINITIONS

Definition 1: An ***intra-operational analysis*** is an intra-procedural analysis that considers analysis of code within the code of operation.

Definition 2: An ***inter-operational analysis*** is an inter-procedural analysis that considers analysis of *affected operations* depending upon other operations or procedures.

Definition 3: A ***subset service*** is a subset of a web service that provides only some of the original service's functionality.

PROPOSED DEFINITIONS

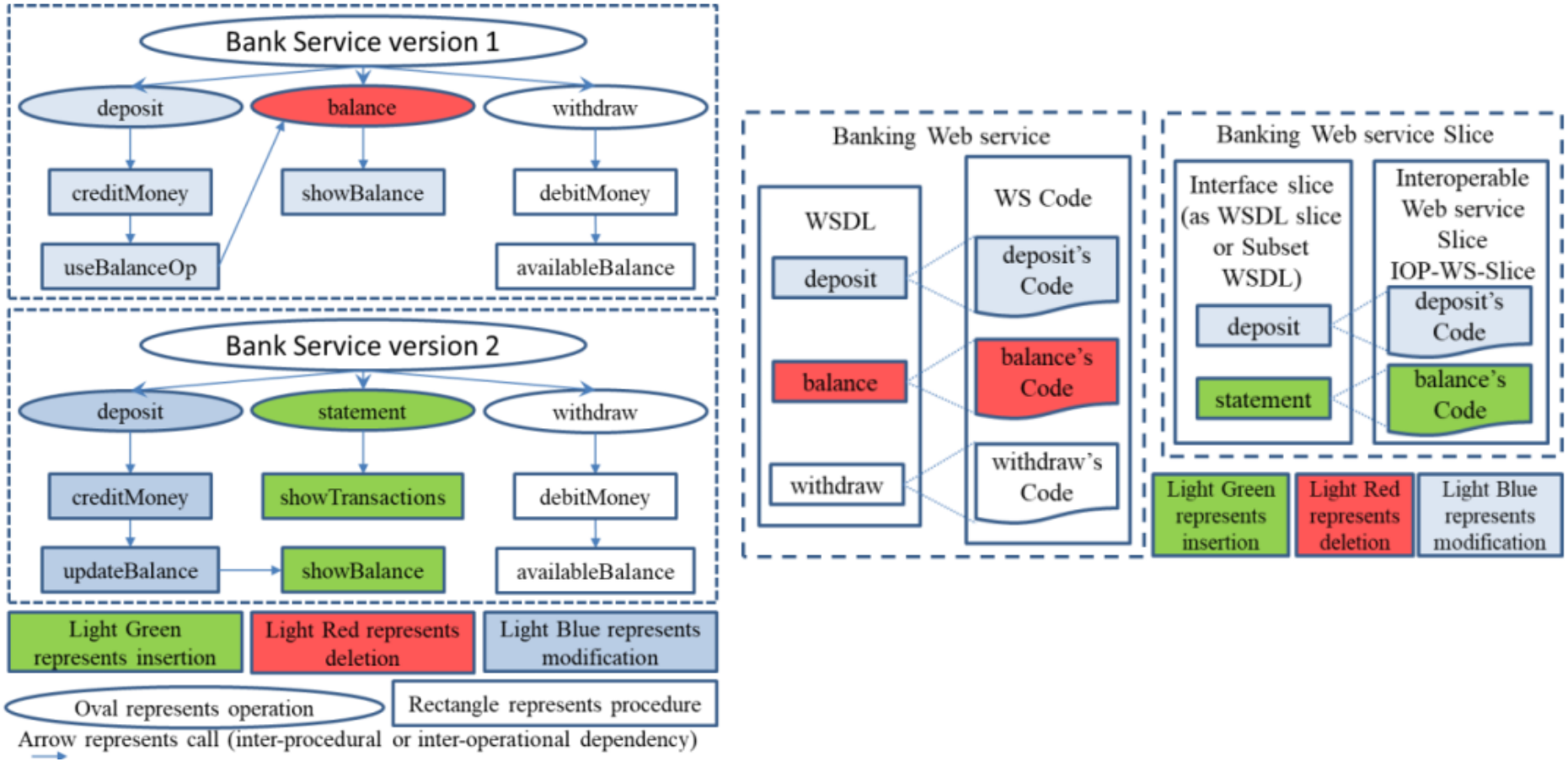
Definition 4: ***Web service slicing*** is a variation of program slicing that extracts a subset of a web service that provides a subset of the original service's behaviour. The resulting slice is referred as a ***web service slice***.

Definition 5: An ***interoperable slice*** is a portion of the web service code that can be accessed as a *subset service* from a remote client that needs a subset functionality.

Definition 6: ***Interface slicing*** is a variation of program slicing that extracts a subset of a given interface's operation that captures a subset of the original interface's behaviour. The resulting slice is referred as an ***interface slice*** containing a subset of interface's operations.

Banking Web service Slicing

- Operational Dependences for Web service Slice (Interface slice and Interoperable slice)



INTRA OPERATIONAL CHANGE ANALYSIS

It identifies *intra-operational changes*

- *Difference WSDL* (DWSDL) capture changes in the WSDL
- *Unit WSDL* (UWSDL) changes in the WS code
- *Reduce WSDL* (RWSDL), contains user selected operations
- These are combined to form the *Combined WSDL* (CWSDL) with unique operations.

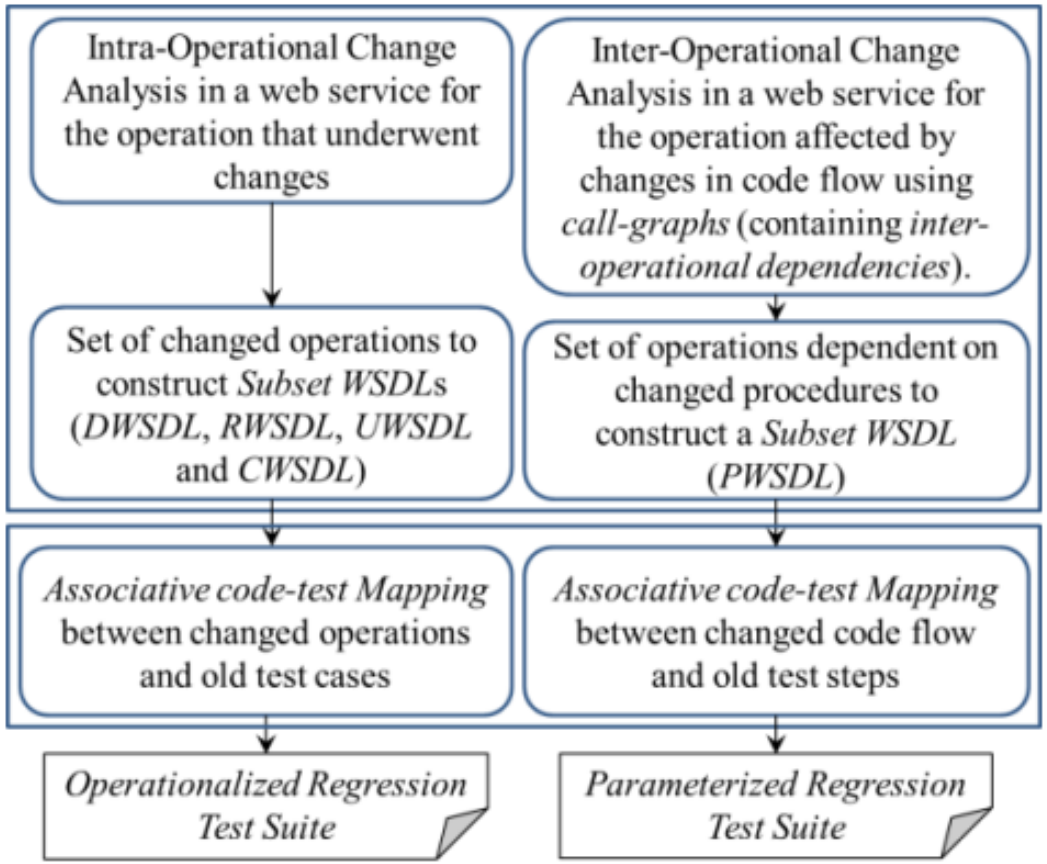
INTER-OPERATIONAL CHANGE ANALYSIS

It captures *inter-operational changes* that depend on changed operations and procedures.

- *Parameter WSDL* (PWSDL) captures code flow changes at the WS code level
- Analogy to Databases *attributes* → Web service Operation *parameters*,
  - input *query* → input *parameter values*
  - output *result* → output *result of the operation*
  - *key attributes* identifies other attributes → *key parameters* identifies other parameters
  - two or more *candidate primary key* → *candidate primary parameters*
  - *primary keys, non-primary keys* → *primary parameters*, *non-primary parameters*
- *Primary parameters* of an operation uniquely identify the other parameters

REGRESSION TESTING OF WEB SERVICES

An overview of ORTWS and PRTWS that use intra-operational and inter-operational change analysis, respectively.



REGRESSION TESTING OF WEB SERVICES

CHANGE BASED WEB-SERVICE REGRESSION TESTING

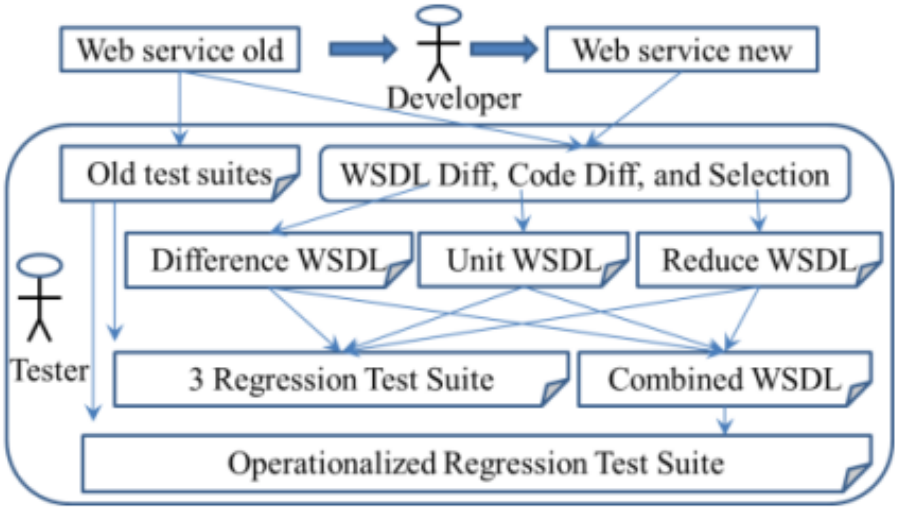
Type (or label)	Level	Location and purpose	Captured in Subset WSDL	Regression testing technique
Insertion	WSDL	Insertion of an operation in WSDL or Datatype in XSD	DWSDL RWSDL	Test case template require suit-able test data
	WS Code	Insertion of code for new operation	UWSDL PWSDL	
Deletion	WSDL	Deletion of opera-tion from WSDL to make it disfunc-tional to client	None, because these changes are rare and unu-sual.	Failure of test cases will test the deletion
	WS Code	Deletion of opera-tion from WS code to make it disfunc-tional at WSDL level		
Modifi-cation	WSDL	Modification of XSD	DWSDL RWSDL	Operational-ized testing
	WS code	Modification at code lines without affecting WSDL	UWSDL PWSDL	Code flow or Parametrized testing

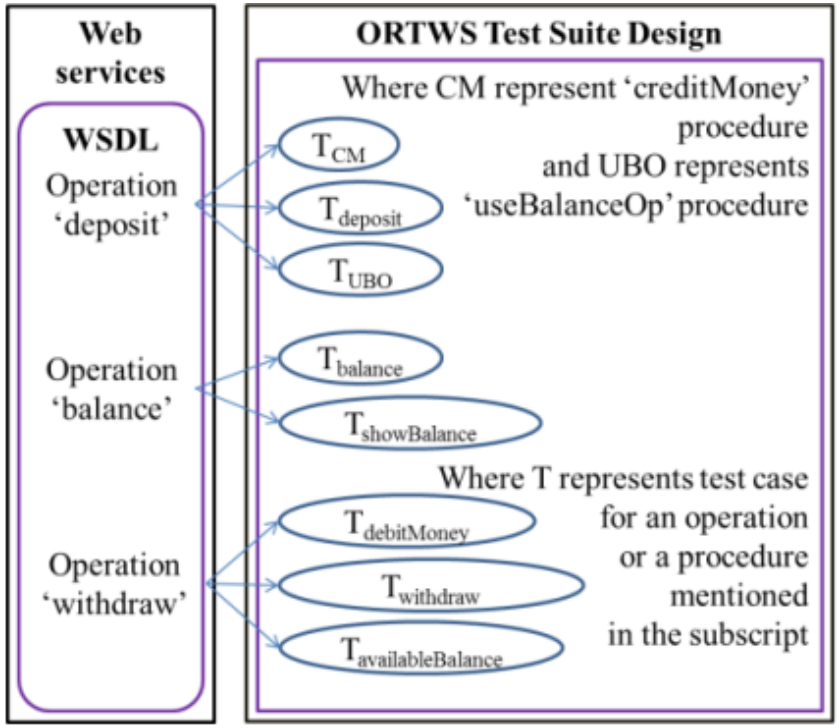
OPERATIONALIZED REGRESSION TESTING OF WEB SERVICES (ORTWS)

The *Intra-Operational Change Analysis* based ORTWS constructs

- four *Subset WSDLs* and
- an *Operationalized Regression Test Suite*

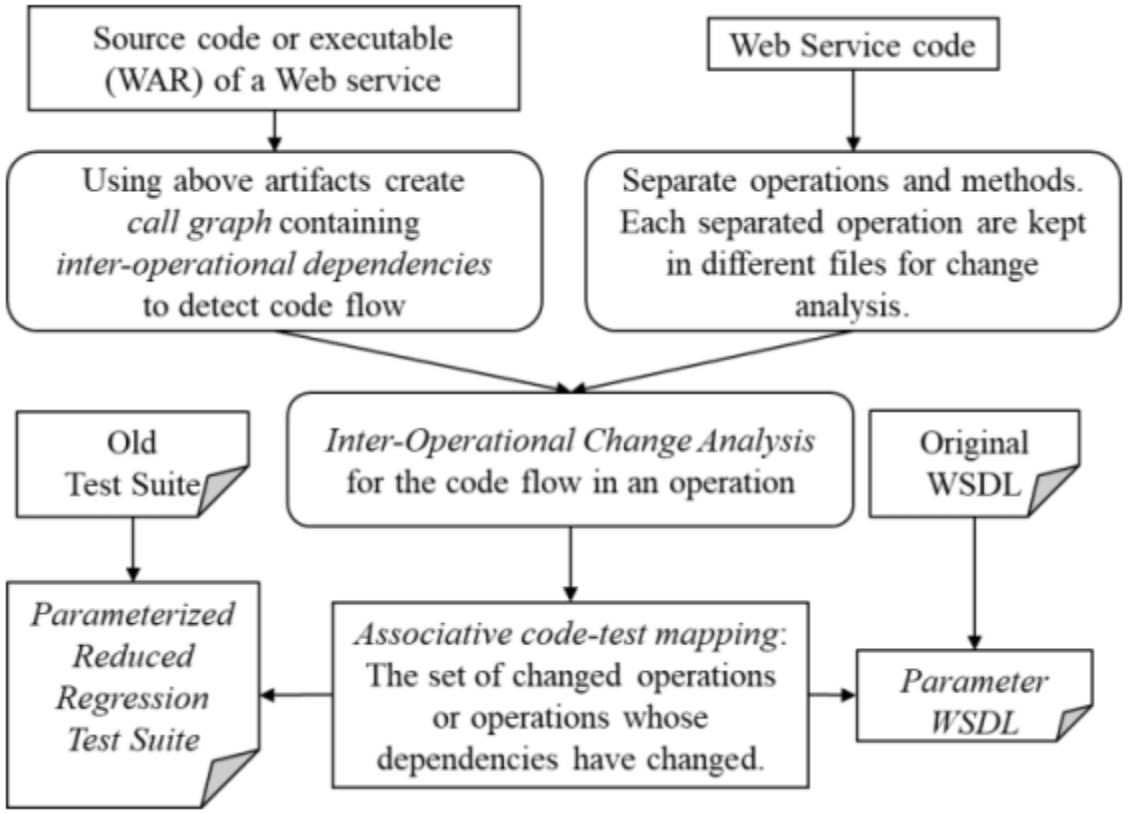
Example of *associative code-test mapping*





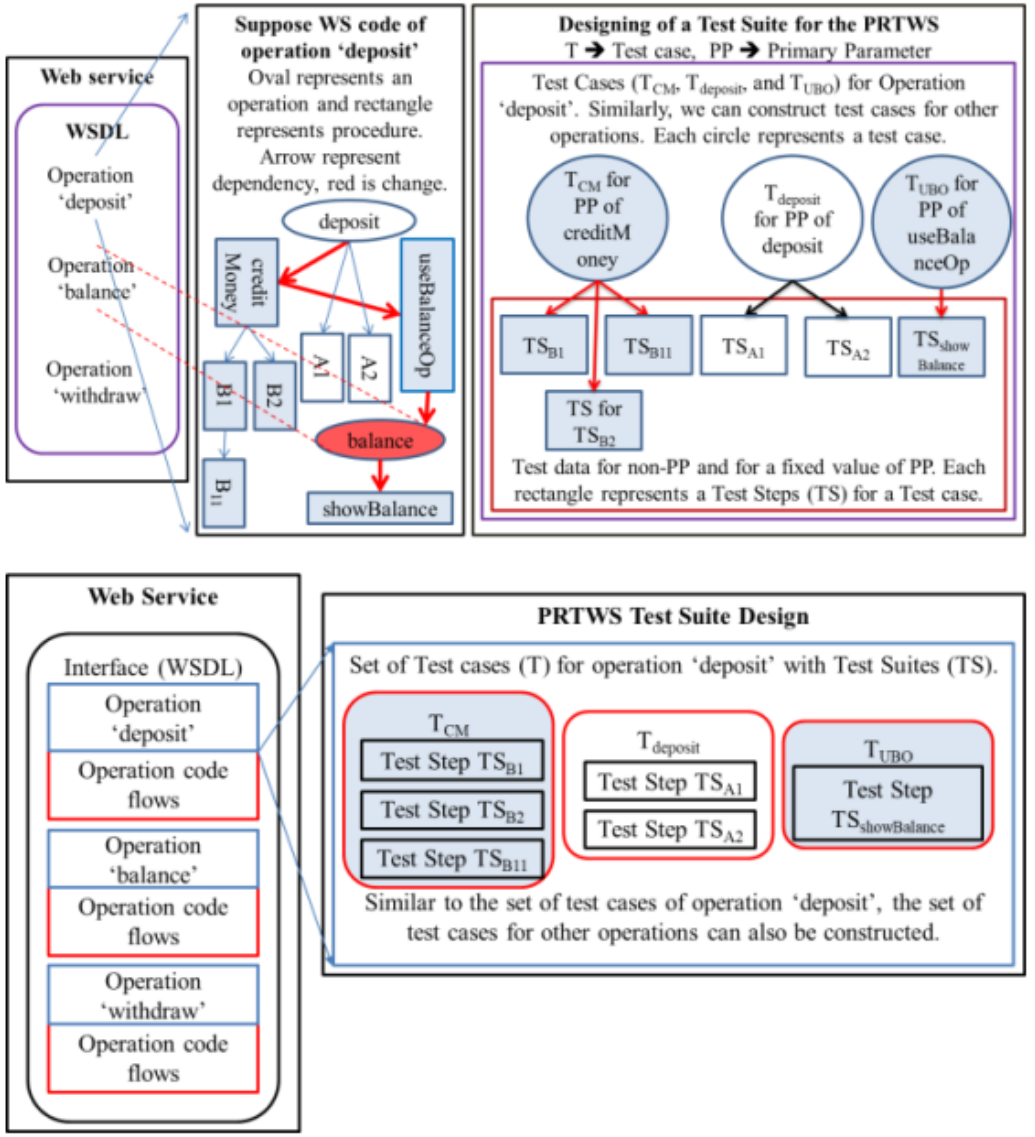
PARAMETERIZED REGRESSION TESTING OF WEB SERVICES (PRTWS)

The Inter-Operational Change Analysis based PRTWS constructs the PWSDL and the Parameterized Regression Test Suite.



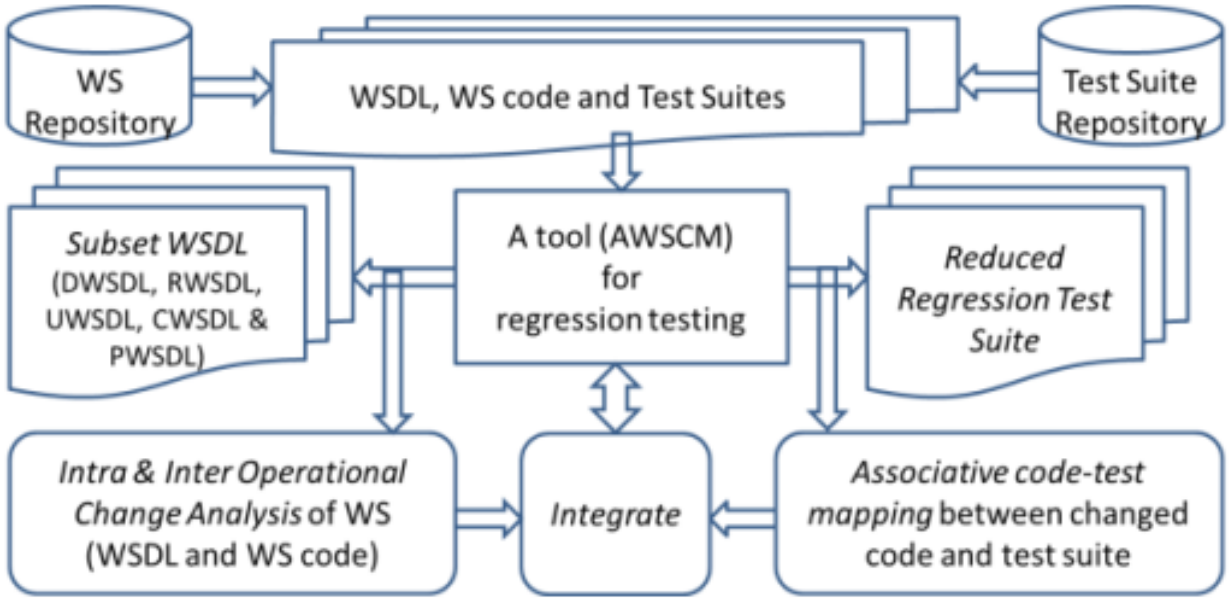
PARAMETERIZED REGRESSION TESTING OF WEB SERVICES (PRTWS)

An example of the associative code-test mapping used to design a PRTWS test suite, where each test case is designed for a fixed value of the Primary Parameter (PP).



AWSCM

Automated Web Service Change Management (AWSCM): a tool for change analysis based regression testing of web service.



EXPERIMENTS: CASE STUDIES ON WEB SERVICES

- *White box analysis* (where the internals of the WS code are known): ORTWS on *SaaS* and *Eucalyptus*
- *Black box analysis* (where they are unknown): ORTWS on *AWS* and PRTWS on *Sunset Sunrise*, *Bible*, *Currency conversion*, and *Global weather*

'Y' denotes the case study is performed'

Web service project	White Box Analysis	Black Box Analysis	Change source
Eucalyptus	Y	Y	GitHub
SaaS	Y	Y	Self-made
BookService	Y	Y	Self-made
AWS	Code is inaccessible	Y	Change analysis not performed
Sunset Sunrise		Y	
Bible		Y	
Currency conversion		Y	
Global weather		Y	

Evaluation of Intra-Operational Change Analysis with ORTWS

Web service project	Subset WSDL	Case study detail	Subset WSDL contains	Test cases (TCs) reduction
SaaS	*D WSDL*	5 operations of the new WSDL and 3 operations of the old WSDL. Here, 2 operations are added. Where 2 of 4 TCs were extracted.	2 operations	50%
	R WSDL	1 of 5 operations was selected. 1 of 4 TCs was extracted.	1 operations	75%
	U WSDL	1 of 5 operations underwent change at the WS code level. 1 operation that underwent change at code-level. 1 of 4 TCs was extracted.	1 operations	75%
	C WSDL	3 of 4 (2+1+1) operation are unique in DWSDL, RWSDL and UWSDL. 3 of 4 TCs were extracted.	3 operations	25%

CASE STUDIES FOR INTRA-OPERATIONAL CHANGE ANALYSIS BASED SUBSET WSDL AND ORTWS

CASE STUDIES FOR INTRA-OPERATIONAL CHANGE ANALYSIS BASED SUBSET WSDL AND ORTWS

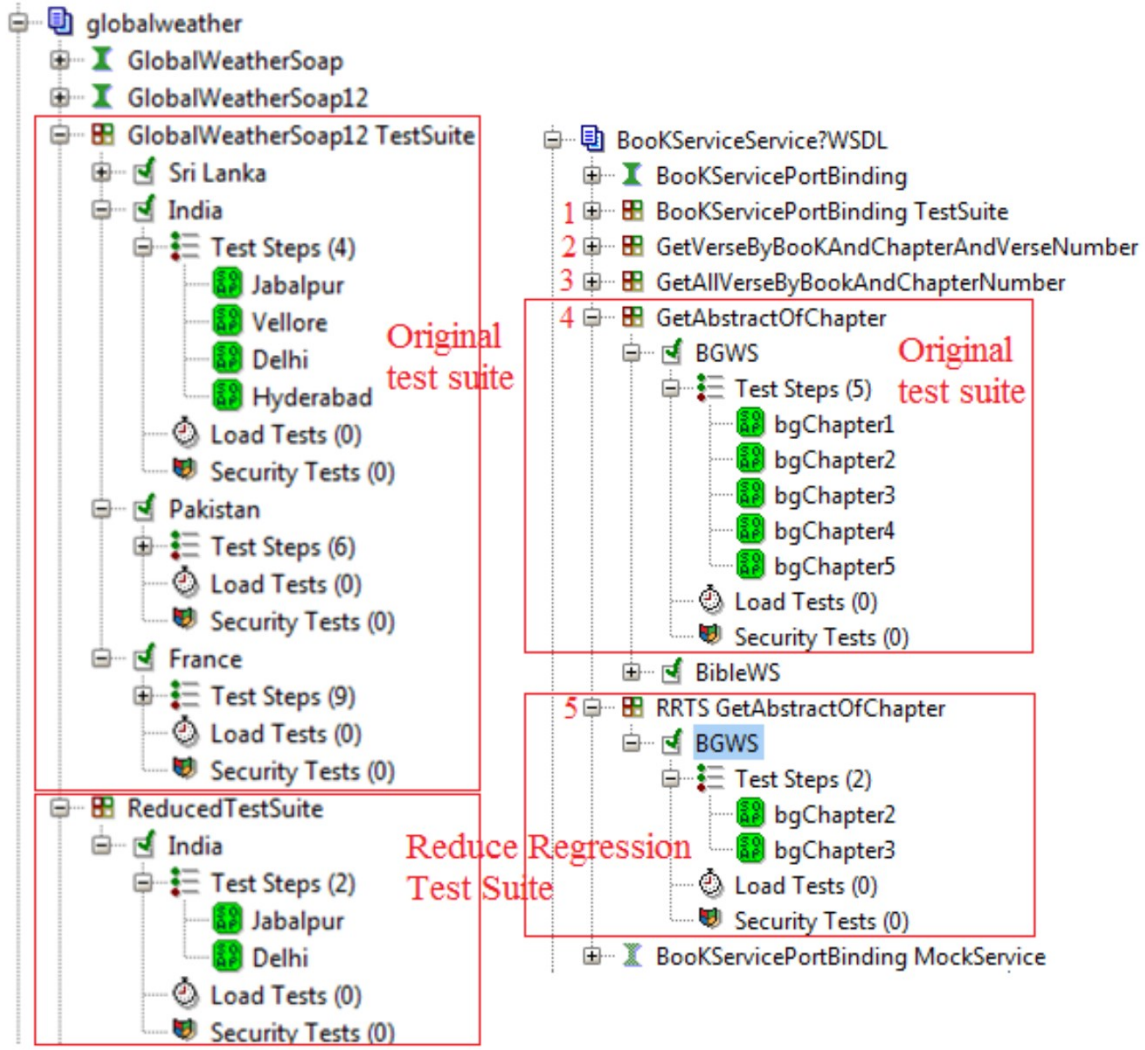
AWS	*R WSDL*	3 of 23 operations were selected. 3 of 23 TCs templates were extracted.	3 operations	86.95%
EucalyptusCC	*D WSDL*	26 operations of the new WSDL and 24 operations of the old WSDL. Here, 2 operations are added at WSDL-level. 2 of 26 TC templates were extracted.	2 operations	92.31%
	R WSDL	2 of 26 operations were selected. 2 of 26 TC templates were extracted.	2 operations	92.31%
	U WSDL	3 of 26 operations underwent changes at the WS code level. 3 of 26 TCs templates were extracted.	3 operations	88.5%
	C WSDL	6 of 7 (2+2+3) operations in the DWSDL, RWSDL and UWSDL were unique. 6 of 26 TCs templates were extracted.	6 operations	76.92%

Black box Inter-operational analysis with PRTWS

- *FromCurrency* is the *primary parameter* making *ToCurrency* the (only) *non-primary parameter*.
- *CountryName* is the *primary parameter* and *CityName* is the (only) *non-primary parameter*.
- *BookTitle* as the *primary parameter*, thus making *ChapterName* and *Verse* as the *non-primary parameters*.
- *Latitude* and *Longitude* as the *primary parameter*

Web service project	Primary parameter	Non Primary parameter	Case study detail	Test Case (TC) & Test Step (TS) reduction
Currency Conversion	From-Currency	ToCurrency	3 out of 9 TSs are selected in 2 out of 4 TCs	66 % reduction in TSs and 50% reduction in TCs
Global weather	CountryName	CityName	1 out of 2 TSs are selected in 1 out of 4 TCs.	50% reduction in TSs and 75% reduction in TCs
Bible	Book Title	Chapter Name	2 out of 4 TSs are selected in 1 out of 4 TCs	50% reduction in TSs and 75% reduction in TCs
Sunset Sunrise	Latitude & Longitude	SunSetTime, SunRiseTime, TimeZone, Day, Month, & Year	4 out of 8 TSs are selected from 1 (single) TC	50% reduction in TSs

Black box Inter-operational analysis with PRTWS



White box Inter-operational analysis with PRTWS

Operations (highlighted in bold) and procedures (in classes of BGVerse or BibleVerse) of BookService with its corresponding ID numbers.

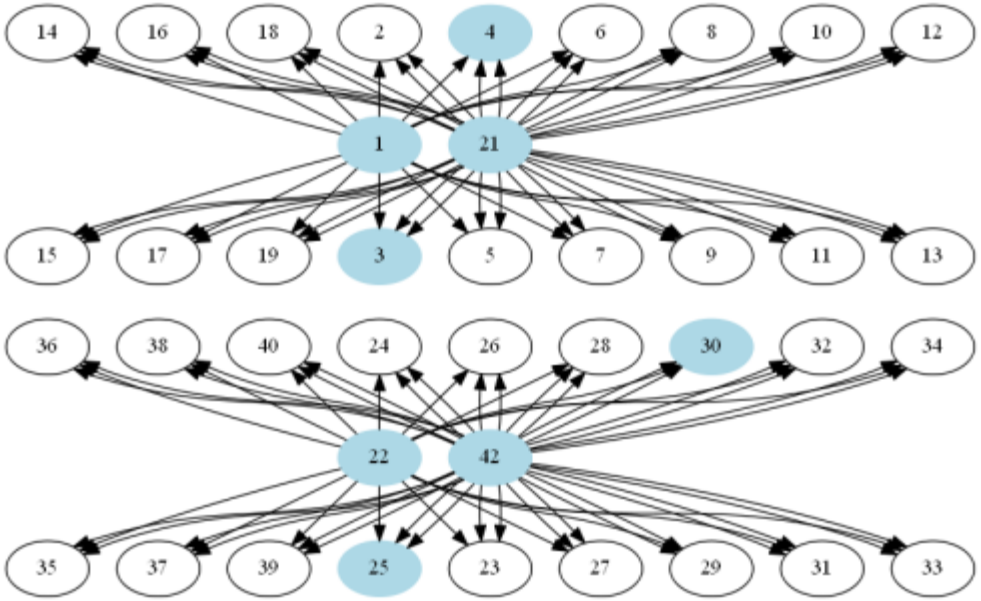
Operations	ID	Primary parameter	Non–Primary Parameter
findBookNumber	--	---	---
getVerseByBookAnd Chap-terAndVerseNumber	43	bookNumber	verseNumber & chapterNumber
getAllVerseByBookAnd ChapterNumber	44	bookNumber	chapterNumber
getAbstractOfChapter	45	bookNumber	chapterNumber

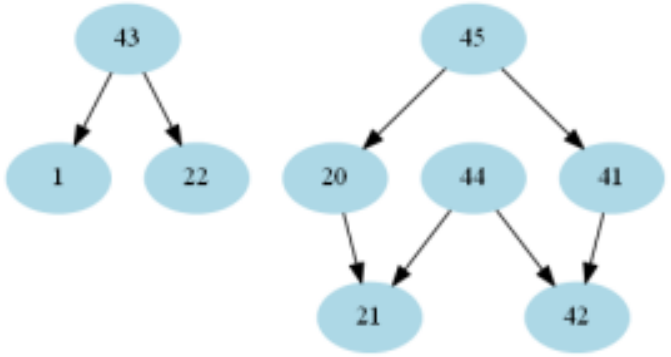
BookService.BGWS.bgOp	**1**
BookService.BGVerse.bgChapter1	2
BookService.BGVerse.bgChapter2	3
BookService.BGVerse.bgChapter3	4
BookService.BGVerse.bgChapter4	5
BookService.BGVerse.bgChapter5	6
BookService.BGVerse.bgChapter6	7
BookService.BGVerse.bgChapter7	8
BookService.BGVerse.bgChapter8	9
BookService.BGVerse.bgChapter9	10
BookService.BGVerse.bgChapter10	11
BookService.BGVerse.bgChapter11	12
BookService.BGVerse.bgChapter12	13
BookService.BGVerse.bgChapter13	14
BookService.BGVerse.bgChapter14	15
BookService.BGVerse.bgChapter15	16
BookService.BGVerse.bgChapter16	17
BookService.BGVerse.bgChapter17	18
BookService.BGVerse.bgChapter18	19
BookService.BGWS.bgOpAbst	**20**
BookService.BGWS.bgAllVerse	**21**
BookService.BibleWS.bibleOp	**22**
BookService.BibleVerse.bibleChapter1	23
BookService.BibleVerse.bibleChapter2	24
BookService.BibleVerse.bibleChapter3	25
BookService.BibleVerse.bibleChapter4	26
BookService.BibleVerse.bibleChapter5	27
BookService.BibleVerse.bibleChapter6	28
BookService.BibleVerse.bibleChapter7	29
BookService.BibleVerse.bibleChapter8	30
BookService.BibleVerse.bibleChapter9	31
BookService.BibleVerse.bibleChapter10	32
BookService.BibleVerse.bibleChapter11	33
BookService.BibleVerse.bibleChapter12	34
BookService.BibleVerse.bibleChapter13	35
BookService.BibleVerse.bibleChapter14	36
BookService.BibleVerse.bibleChapter15	37
BookService.BibleVerse.bibleChapter16	38
BookService.BibleVerse.bibleChapter17	39
BookService.BibleVerse.bibleChapter18	40
BookService.BibleWS.bibleOpAbst	**41**
BookService.BibleWS.bibleAllVerse	**42**
BookService.BookService.getVerseByBookAndChapterAndVerseNumber	**43**
BookService.BookService.getAllVerseByBookAndChapterNumber	**44**
BookService.BookService.getAbstractOfChapter	**45**

White box Inter-operational analysis with PRTWS

Calls of operations ‘43’, ‘44’, and ‘45’ in *BookService*.

- ‘1’ is *bgOp* and ‘22’ is *bibleOp*.
- ‘21’ is *bgAllVerse*, ‘42’ is *bibleAllVerse*, ‘20’ is *bgOpAbst* and ‘41’ is *bibleOpAbst*.

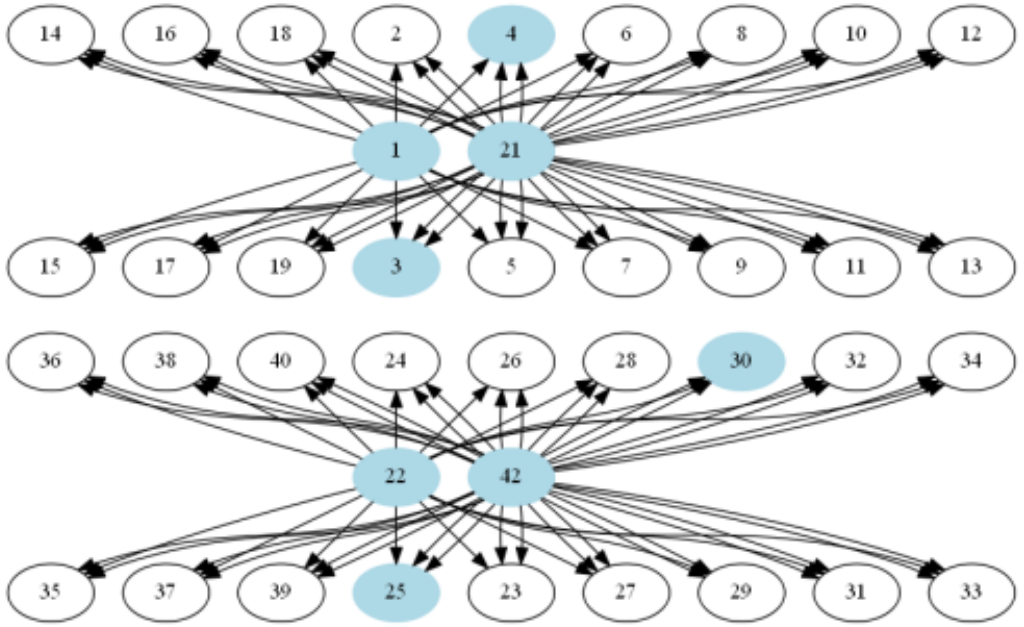




White box Inter-operational analysis with PRTWS

- Eighteen calls of operation *bgOp* (‘1’) and the thirty-six calls of operation *bgAllVerse* (‘21’) in BGWS.

- Eighteen calls of operation *bibleOp* (‘22’) and thirty-six calls of operation *bibleAllVerse* (‘42’) in BibleWS.



White box Inter-operational analysis with PRTWS

PRTWS CASE STUDY FOR *BOOKSERVICE*

Book Service detail	**Case study detail**	**Inter-operational Dependence analysis**	**PWSDL contains**	**Test case reduc-tion**
Original WSDL has 4 opera-tions. Figure 11 and 12 shows 116 calls.	'*bgChapter2*' & '*bgChap-ter3*' had gone through changes at code level	3 out of 4 operations in *BGWS* are dependent on the two changes	3 opera-tions	2 out of 5 test steps are ex-tracted 60% reduction
	'*bibleChap-ter3*' & '*bi-bleChapter8*' had gone through the changes	3 out of 4 operations in *BibleWS* is dependent on the two changes	3 opera-tions	2 out of 5 test cases are ex-tracted 60% reduction

Our Observations and findings

1. Change analysis on the WSDL and the WS code separately and may skip either analysis if it is not required.
2. The *Intra-Operational* and *Inter-Operational Change Analysis* of web services identifies subsets of the test cases that lead to reduced regression-testing cost.
3. Capturing changes are useful to make *Subset WSDLs*, which are helpful to identifying relevant test cases.

Our Observations and findings

4. The execution and analysis of an *interoperable slice* using an *interface slice* can reduce the number of test cases required for regression testing.
5. Systematic design of the test suites for a web service according to the *associative code-test mapping* makes it easier to conduct regression testing.
6. AWSCM proved helpful in testing the changes that occurred in a service.
7. AWSCM successfully generated accurate output (*Subset WSDLs* and *Reduced Regression Test Suites*).

SUMMARY

1. *Web-service slicing* exploits the combination of an *interoperable slice* and an *interface slice*.
2. *Web service slicing* for maintaining *interoperability* of subset services over a network.
3. *Intra-operational* and *Inter-operational* analysis gives rise to two the new regression-testing techniques for web services, ORTWS and PRTWS.
4. Enable the execution of a subset service (*interoperable slice*) using a subset of the test cases reduced *Regression Test Suites*.
5. Eight successful case studies for reduction in regression-testing effort.

More resources

Animesh Chaturvedi, and David Binkley. "Web Service Slicing: Intra and Inter-Operational Analysis to Test Changes." **IEEE Transactions on Services Computing** (2018).

DOI: 10.1109/TSC.2018.2821157

https://sites.google.com/site/animeshchaturvedi07/research/awscm

https://youtu.be/Thz8RZ7PZrA

https://youtu.be/qkXe3YEi264

Other related papers

- Animesh Chaturvedi, Aruna Tiwari, Shubhangi Chaturvedi, and Dave Binkley "Service Evolution Analytics: Change and Evolution Mining of a Distributed System", *IEEE Transactions on Engineering Management* (2020).
- Animesh Chaturvedi, "Subset WSDL to access Subset Service for Analysis", IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), 2014.
- Animesh Chaturvedi, "Automated Web Service Change Management AWSCM - A Tool", IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), 2014.
- Animesh Chaturvedi and Atul Gupta, "A Tool Supported Approach to Perform Efficient Regression testing of Web Services", 7th IEEE International Symposium on Maintenance and Evolution of Service oriented and Cloud Based Systems (IEEE MESOCA), 2013.
- Animesh Chaturvedi, "Reducing cost in regression testing of web service", 6th CSI International Conference on Software Engineering (6th CONSEG) on IEEE.

Cloud Service Evolution Analytics

1. Cloud Life Cycle
2. Cloud Deployment Scenarios
3. Cloud Service Testing
4. Web Service Slicing for Regression Testing of Services
5. **Cloud Service Evolution Analytics**
6. Quality of Service and Service Level Agreement

Animesh Chaturvedi, Aruna Tiwari, Shubhangi Chaturvedi, and Dave Binkley "Service Evolution Analytics: Change and Evolution Mining of a Distributed System", **IEEE Transactions on Engineering Management** (2020).

Service Evolution Analytics: Change and Evolution Mining of a Distributed System

1. INTRODUCTION
2. CHANGE AND EVOLUTION MINING OF AN EVOLVING DISTRIBUTED SYSTEM
3. SERVICE EVOLUTION ANALYTICS
4. EXPERIMENTS ON DISTRIBUTED SYSTEMS
5. CONCLUSION

INTRODUCTION

Distributed Computing models rely upon service frameworks

- Grid Computing: the mother, distributed collection of computing resources
- Cloud Computing: enables convenient, on-demand shared computing resources
- Utility Computing: on-demand, pay-as-you go billing
- Service Oriented Computing: the sharing of remote Web Service

Motivation

Evolving distributed system → *Evolving system* stored in a software repository

changeability → *change mining*
evolvability → *evolution mining*

uncover change and evolution information over time

CHANGE AND EVOLUTION MINING OF AN EVOLVING DISTRIBUTED SYSTEM

Version series of an evolving distributed system

$$\mathrm{VS} = \{V_1, V_2, \ldots, V_N\}$$

snapshots taken at times

$$\{t_1, t_2, \ldots, t_N\}$$

Change Mining of Two Versions: Service Change Classification

Two steps as summarized

- old and new versions as input

Then, invokes

- *Algorithm 1, ServiceChangeClassifier,*

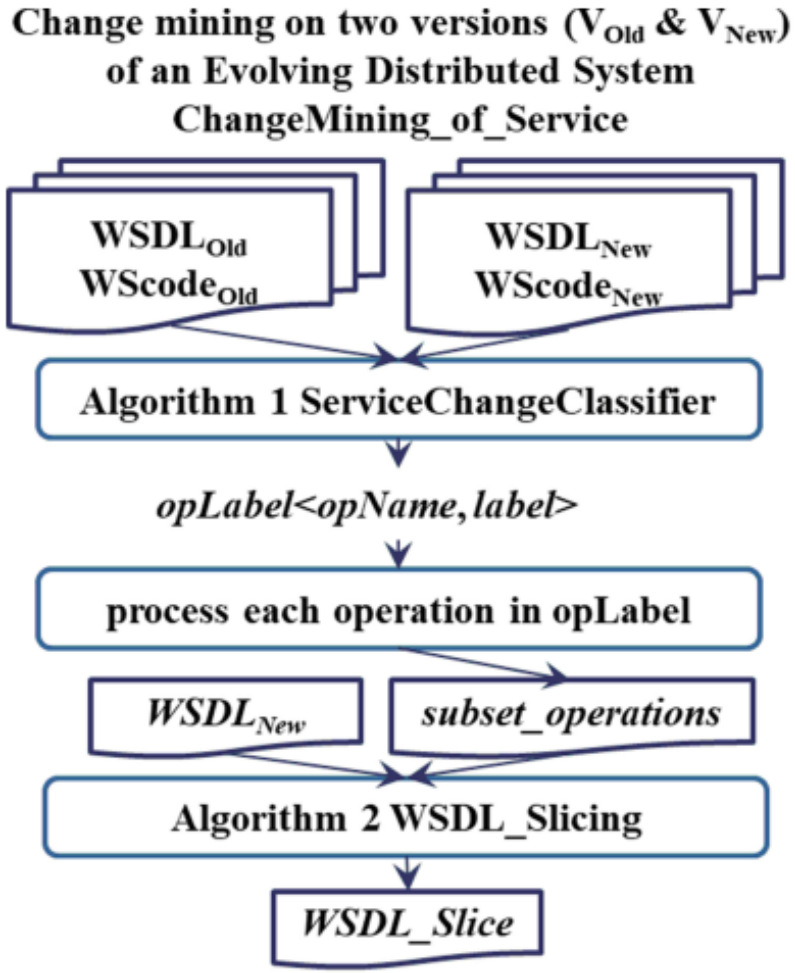
- *Algorithm 2,WSDL_Slicing*

ChangeMiningOfService ($WSDL_{Old}$, $WSDL_{New}$, $WScode_{Old}$, $WScode_{New}$)

1. *opLabels* = ***ServiceChangeClassifier*** (*$WSDL_{Old}$*, *$WSDL_{New}$*, *$WScode_{Old}$*, *$WScode_{New}$*)
2. Initialize an empty set *subset_operations*
3. **For each** pair < *opName*, *label* > in *opLabels*
   a. **If** *label* = = "*inserted*" or *label* = = "*modified*"
   subset_operations ← *subset_operations* ∪ *opName*

   End for
4. *WSDL_Slice* = ***WSDL_Slicing*** (*$WSDL_{New}$*, *subset_operations*)

Return *WSDL_Slice*



Change Mining of Two Versions: Service Change Classification

Overviews the types of changes identified

CLOUD SERVICE CHANGE CLASSIFIERS

Label of the change	Level	Purpose at location of change	Captured in Subset WSDL
Insertion I_{WSDL}	WSDL	Insertion of an operation in WSDL or Datatype in XSD	DWSDL
Insertion $I_{WS\ Code}$	WS Code	Insertion of code for new operation	UWSDL
Deletion D_{WSDL}	WSDL	Deletion of operation from WSDL to make its disfunction to client	None. Deletions does not affect, thus ignored [25].
Deletion $D_{WS\ Code}$	WS Code	Deletion of operation from WS code to make disfunction at WSDL	
Modification M_{WSDL}	WSDL	Modification of XSD	DWSDL
Modification $M_{WS\ Code}$	WS Code	Modification at code lines without affecting WSDL	UWSDL

Change Mining of Two Versions: Service Change Classification

A WSDL description has six major parts

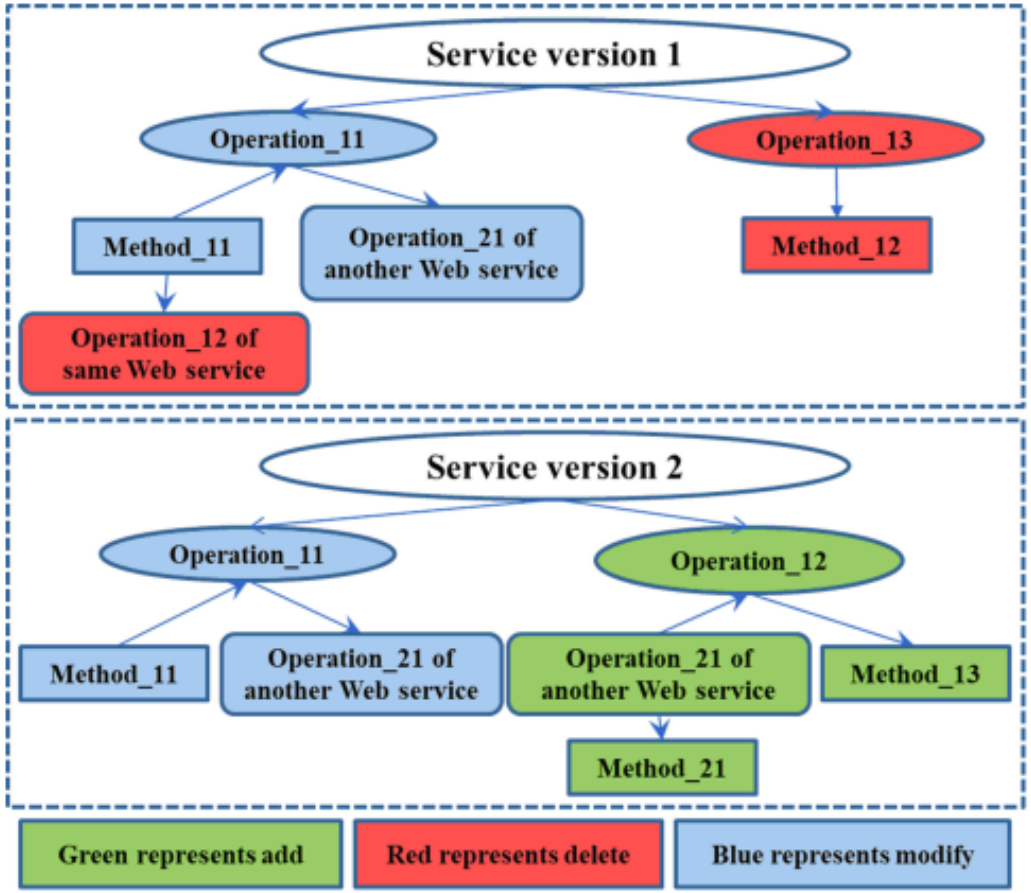
CHANGES IN WSDL PROPERTIES

Property	Effect of changes on the properties of WSDL
Definition change	Change in the xml version, name of service, xmlns or targetNamespace
XSD Type (schema)	The XSD includes sequence of two kind of data types in the form of XML elements: complex data types and simple data types. Changes may be in Simple Types or Complex Types, which are represented by elements, names, or data type. The simple data types contain only literal text, while the complex data types contain nested attributes and parameters.
Message	Changes may be in the message name of an operation. Changes may also be in the elements and part name of a message.
Port Type change	Change may be in the port type name or operation. Changes may also in the way message input and output were used.
Binding change	Change may be in the binding name and soap binding. Changes may also in the soap body (input - output).
Service change	Changes may be in the service name, binding address, or soap address location.

Change Mining of Two Versions: Service Change Classification

Simple representation of changes in a service

"Service version 1" is upgraded to make "Service version 2"



Change Mining of Two Versions: Service Change Classification

Algorithm 1: ***ServiceChangeClassifier*** ($WSDL_{Old}$, $WSDL_{New}$, $WScode_{Old}$, $WScode_{New}$)

1. $WScode_{Changes}$ = ***semanticCodeDiff***($WScode_{Old}$, $WScode_{New}$)
2. $WSDL_{Changes}$ = ***semanticWSDLDiff***($WSDL_{Old}$, $WSDL_{New}$)
3. Initialize array list *opLabels*<*opName, label*> with *unchanged* label, and retrieve *opName_change* from $WScode_{Changes}$ and $WSDL_{Changes}$ along with the change labels.
4. **For** each *opName_change* in $WSDL_{Changes}$
   a. **If** (*opName_change* has label *insert*)
      Assign label *inserted* to *opName* in *opLabels*
   b. **If** (*opName_change* has label *delete*)
      Assign label *deleted* to *opName* in *opLabels*
   c. **If** (*opName_change* has label *modify*)
      Assign label *modified* to *opName* in *opLabels*
   End For
5. **For** each *opName_change* in $WScode_{Changes}$
   a. **If** (*opName_change* has label *insert*)
      Assign label *inserted* to *opName* in *opLabels*
   b. **If** (*opName_change* has label *delete*)
      Assign label *deleted* to *opName* in *opLabels*
   c. If (opName_change has label *modify*)
      Assign label *modified* to *opName* in *opLabels*
   End For

Return *opLabels*<*opName, label*>

Algorithm 2: ***WSDL_Slicing*** ($WSDL_{New}$, *subset_operations*)

String *cStartDef*, *cXSD*, *cMsg*, *cPort*, *cBinding*, *cService*, *cEndDef*

Array of String[] $opWSDL_{New}$

1. [$subset\_operations \in opOfWSDL_{New}$ | *subset_operations* = operation required to construct *WSDL_Slice*]
2. *cStartDef* = **sliceStartDefinition**($WSDL_{New}$)
   cXSD = **sliceXSD**($WSDL_{New}$, *subset_operations*)
   cMsg = **sliceMessage**($WSDL_{New}$, *subset_operations*)
   cPort = **slicePort**($WSDL_{New}$, *subset_operations*)
   cBinding = **sliceBinding**($WSDL_{New}$, *subset_operations*)
   cService = **sliceService**($WSDL_{New}$)
   cEndDef = **sliceEndDef**($WSDL_{New}$)
3. *WSDL_Slice* = *cStartDef* + *cXSD* + *cMsg* + *cPort* + *cBinding* + *cService* + *cEndDef*

Return *WSDL_Slice*}

Evolution Mining of a Version Series: Service Evolution Metrics

Four novel *service evolution metrics*

The metrics are based on five important quantitative attributes:

- *number of operations*,
- *WSDL lines*, parameters,
- messages, and
- operation code lines.

Evolution Mining of a Version Series: Service Evolution Metrics

1) Lines Per Operation in the WSDL:

$$\text{LOWSDLi} = \frac{\text{Number of source lines in WSDL of } V_i}{\text{Number of operations in version } i}$$

$$LOWSDL = \{(V_1, LOWSDL_1) \ldots (V_i, LOWSDL_i) \ldots (V_N, LOWSDL_N)\}$$

2) Parameters Per Operation in the WSDL:

$$\text{PO}_i = \frac{\text{Number of parameters in WSDL of } V_i}{\text{Number of operations for version } i}$$

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

Evolution Mining of a Version Series: Service Evolution Metrics

3) Messages Per Operation in the WSDL:

$$\mathrm{MO}_i = \frac{\text{Number of messages in WSDL of } V_i}{\text{Number of operations for version } i}$$

$$\mathrm{MO} = \{(V_1, \mathrm{MO}_1) \ldots (V_i, \mathrm{MO}_i) \ldots (V_N, \mathrm{MO}_N)\}$$

4) Code Lines Per Operation in the WS Code:

$$\mathrm{WSCLO}_i = \frac{\text{Number of code lines in WS code of } V_i}{\text{Number of operations in version } i}$$

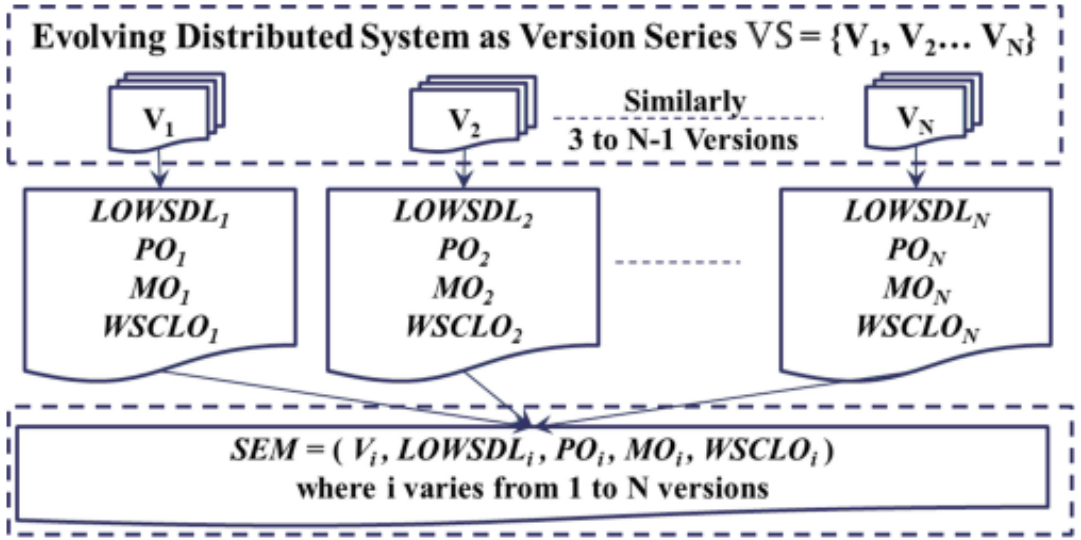
$$\mathrm{WSCLO} = \{(V_1, \mathrm{WSCLO}_1) \ldots (V_i, \mathrm{WSCLO}_i) \ldots (V_N, \mathrm{WSCLO}_N)\}$$

Evolution Mining of a Version Series: Service Evolution Metrics

- Summary of the evolution mining of version series based on service evolution metrics (SEM).
- Four novel metrics

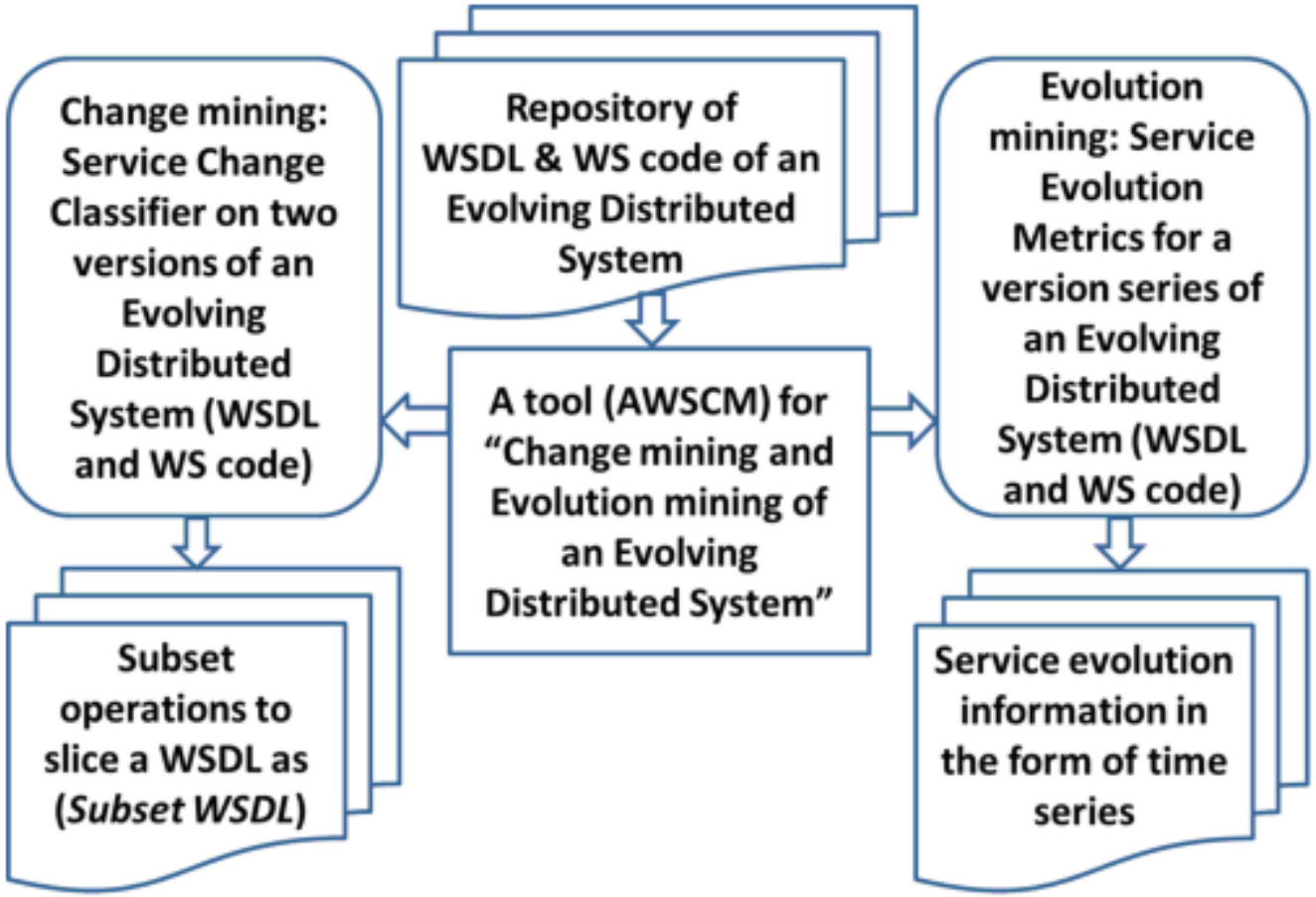
  $(V_i, \text{LOWSDL}_i, \text{PO}_i, \text{MO}_i, \text{WSCLO}_i)$

  for version V_i
- Create four time series graphs



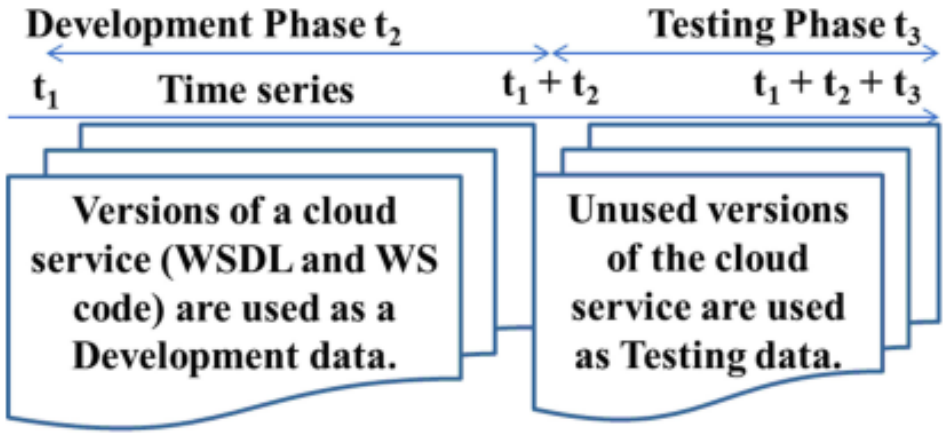
SERVICE EVOLUTION ANALYTICS

The WSDL and the WS code of a version series



SERVICE EVOLUTION ANALYTICS

- Time t_1 the *development phase* runs for time t_2
- Testing phase starts at time $t_1 + t_2$ and runs for time t_3
- Therefore, both the phases end at time $t_1 + t_2 + t_3$



SERVICE EVOLUTION ANALYTICS

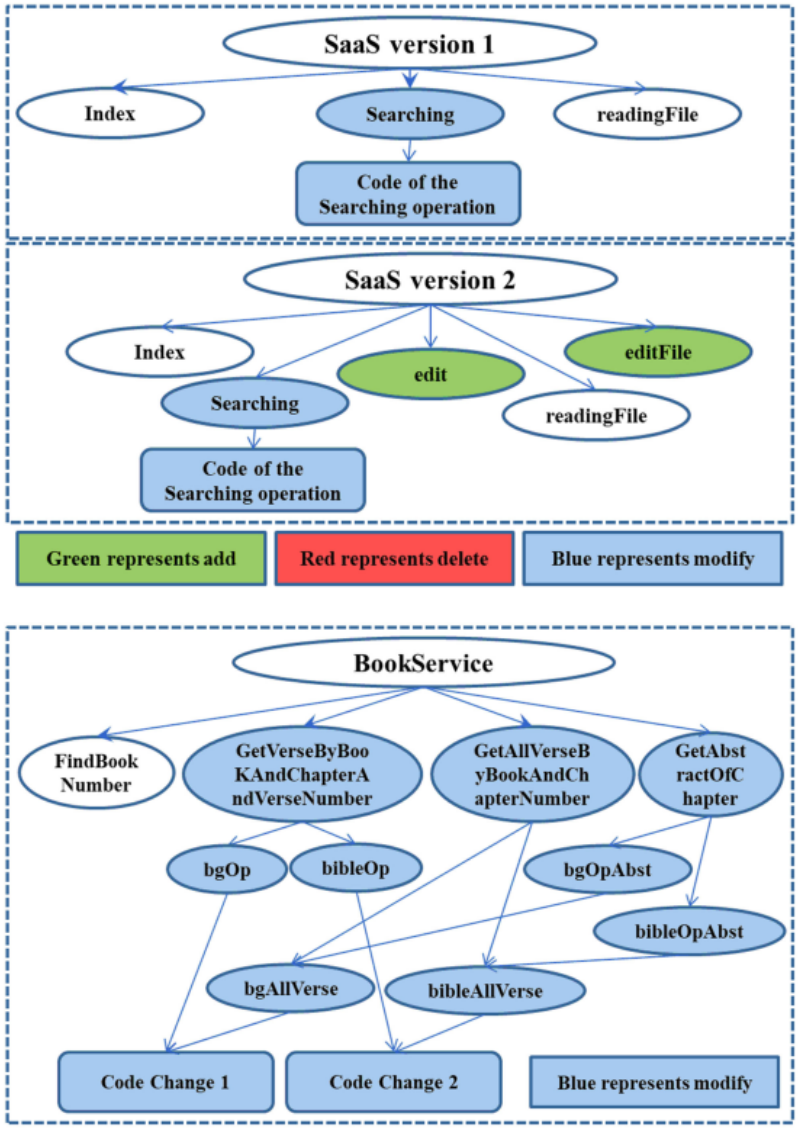
- Output correctness (Subset WSDL an interface slice)
  - *software acceptance:* an IDE (NetBeans and Eclipse) or testing framework (SoapUI and JMeter)
  - *human acceptance:* an engineer determines if the tool's output is satisfactory

EXPERIMENTS ON DISTRIBUTED SYSTEMS

- Web services
  - two self-made (*SaaS* and *BookService*)
  - two real-world (*AWS* and *EucalyptusCC*)
- Change classification in the WSDL Slice construction
- Empirical reduction in regression testing cost
- Experiments involving the Service Evolution Metrics.

WSDL Slice Construction

- Changes in *SaaS*, and *BookService*
- Effects of changes in the form of dependency graphs



WSDL Slice Construction

- Two Subset WSDLs:
  - the *Difference WSDL* (DWSDL) and
  - the *Unit WSDL* (UWSDL)
- SUBSET WSDL FOR CHANGE ANALYSIS

Web services	DWSDL	UWSDL	Available at
SaaS	Y	Y	Self-made
BookService	Y	Y	Self-made
Eucalyptus	Y	Y	GithHub
AWS	Y	N	GithHub

Changes in two WSDL

- Three classification labels (inserted, deleted, and modified)

Example for Eucalyptus-Cluster Controller (CC)
Feb-13.Eucalytpus-CC.wsdl ➔ Aug-13.Eucalytpus-CC.wsdl
Operation Change: ModifyNode inserted; MigrateInstances inserted.
Schema Change: CT ccInstanceType has modified; CT virtualBootRecordType has modified; CT metricCounterType has modified; CT metricDimensionsType has modified.
May-15.Eucalytpus-CC.wsdl ➔ Dec-16.Eucalytpus-CC.wsdl
Operation Change: DescribePublicAddresses deleted; AttachNetworkInterface inserted; DetachNetworkInterface inserted;
Schema Change: CT DetachVolumeResponseType inserted; CT ccInstanceType has modified; CT netConfigType has modified;

Example for AWS-Elastic Compute Cloud (EC2)
Feb-13.ec2.wsdl ➔ Aug-13.ec2.wsdl
Operation Change: DescribeReservedInstancesModifications inserted; ModifyReservedInstances inserted.
August-13.ec2.wsdl ➔ Oct-13.ec2.wsdl
Operation Change: AcceptVpcPeeringConnection inserted; CreateVpcPeeringConnection inserted; DeleteVpcPeeringConnection inserted; DescribeVpcPeeringConnections inserted; RejectVpcPeeringConnection inserted.

*where CT stands for ComplexType input-output data structure

*for inserted operations, it is obvious that their input-output data-structure were also inserted, thus we skipped its details.

*for deleted operations, it is obvious that their input-output data-structure were also deleted, thus we skipped its details.

Service Maintenance: Reduced Regression Testing

Four experiments that consider the retrieval of WSDL Slices as well as their use in test-case reduction

EXPERIMENTS FOR CHANGE MINING-BASED WSDL SLICES AND TEST CASE RETRIEVAL

Service	Test Cases (TCs) retrieval	WSDL Slice contains	TCs reduction in %
SaaS	2 out of 5 operations are added, the tool retrieves 2 of 4 TCs.	DWSDL has 2 Operations	50%
	1 out of 5 operations was changed in WS code. Thus, 1 out of 4 TCs is retrieved.	UWSDL has 1 Operation	75%
BookService	3 out of 4 operations in *BGWS* are dependent on the two changes. Thus, 2 out of 5 test steps are retrieved.	UWSDL has 3 Operations	60%
	3 out of 4 operations in *BibleWS* is dependent on the two changes. Thus, 2 out of 5 test cases are retrieved.	UWSDL has 3 Operations	60%
AWS	3 out of 23 operations are selected leading to retrieving 3 out of 23 TC templates.	DWSDL has 3 Operation	86.95%
EucalyptusCC	2 operations are added leading to 2 out of 26 TC templates being retrieved.	DWSDL has 2 Operation	92.31%
	3 out of 26 WS code operations are changed leading to 3 out of 26 TC templates being retrieved.	UWSDL 3 has Operation	88.5%

Service Evolution Metric Study

- Two cloud services in the
- Two large-scale evolving distributed systems:
  - Eucalyptus Cluster Controller (Eucalyptus-CC) and
  - Amazon Web Service – Elastic Compute Cloud (AWS-EC2)

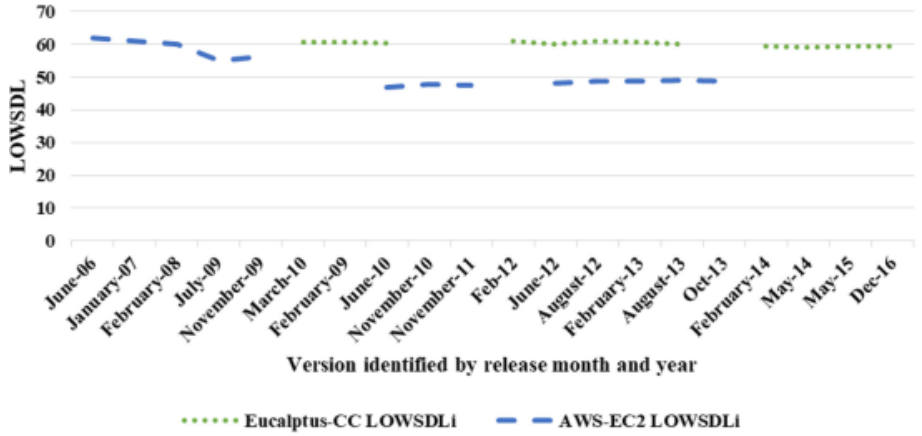
Service Evolution Metric Study

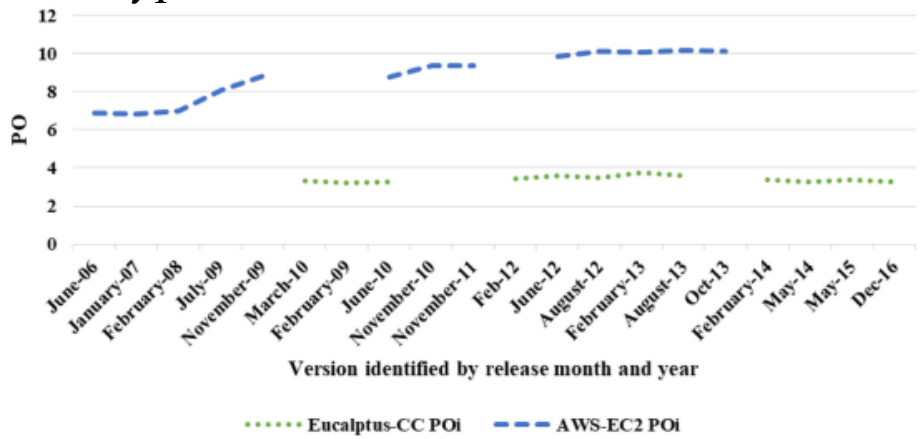
INFORMATION TO CALCULATE SERVICE EVOLUTION METRICS

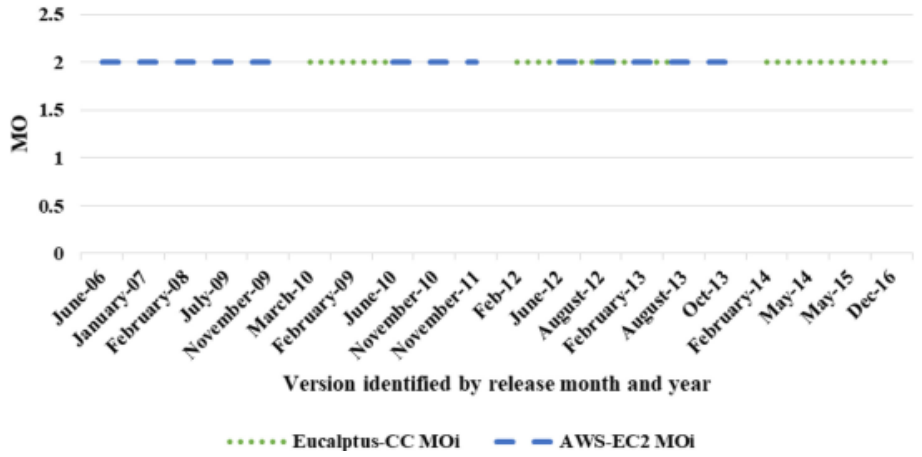
Time Sequence	Eucalyptus Cluster Controller (CC)						AWS-Elastic Compute Cloud (EC2)		
Time for last commit	Version	# Commits	# Lines of WSDL	# Parameter	# Code lines	# Operations	# Lines of WSDL	# Parameter	# Operations
June-06	The project was not published on GitHub during this period.						865	96	14
January-07							1157	130	19
Feb-08							1560	181	26
July-09							3568	523	65
November-09							4565	714	81
March-10	1.0	2612	912	50	3272	15	Data is not available for these months.		
Feb-09	1.5	512	848	45	2086	14			
June-10	2.0	4054	1087	59	3663	18	4077	762	87
Nov-10	Data is not available for these months.						4536	890	95
Nov-11							5637	1116	119
Feb-12	3.0	13647	1464	83	4822	24	Data is not available for this month.		
June-12	3.1.0	14252	1684	101	4870	28	6513	1359	137
August-12	3.1.1	14310	1465	84	4870	24	7021	1455	144
Feb-13	3.2.1	16770	1580	98	5606	26	7252	1501	149
August-13	3.3.1	19974	1684	101	6438	28	7392	1535	151
Oct-13	3.4	20621	1783	102	6672	30	Data is not available for this month.		
Feb-14	Data is not available for this month.						7612	1579	156
May-14	4.0	22130	1836	102	6869	31	Data is not available for these months.		
May-15	4.1.1	23208	1840	104	7066	31			
Dec-16	4.3.1		1905	105	6346	32			

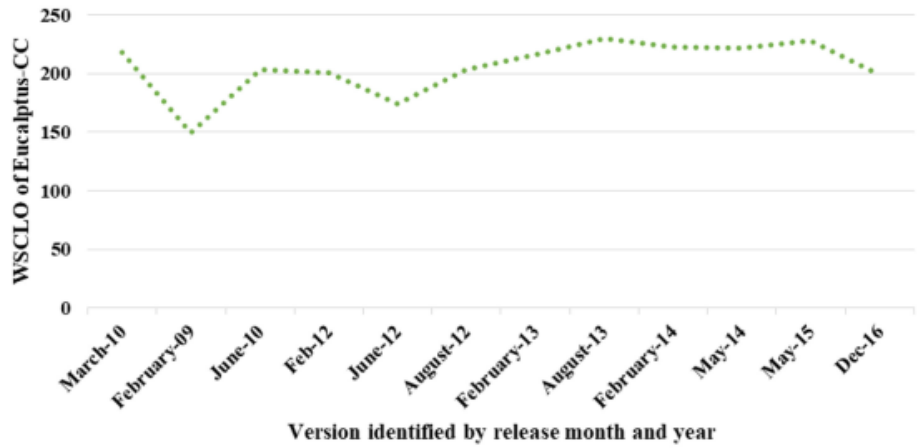
Service Evolution Metric Study

Four time-series graphs to show four service evolution metrics for AWS-EC2 and Eucalyptus-CC.









CONCLUSION

Change and Evolution mining of Evolving Distributed System

- *Service Change Classifier*: change labels to operations that extracts *WSDL slice*.
- *Service Evolution Metrics* from a version series of Cloud service
- Service Evolution Analytics model and tool (AWSCM)
- Case studies to construct WSDL slices, to reduce regression testing cost, and service evolution metrics
  - two self-made (SaaS and BookService)
  - two well-known cloud services: Eucalyptus-CC and AWS-EC2.
- Subset regression testing helps to maintain the QoS and SLA.

More resources

Animesh Chaturvedi, Aruna Tiwari, Shubhangi Chaturvedi, and Dave Binkley "Service Evolution Analytics: Change and Evolution Mining of a Distributed System", **IEEE Transactions on Engineering Management** (2020).

DOI: 10.1109/TEM.2020.2987641

https://sites.google.com/site/animeshchaturvedi07/research/awscm

https://youtu.be/Thz8RZ7PZrA

https://youtu.be/qkXe3YEi264

Other related papers

- Animesh Chaturvedi, and David Binkley. "Web Service Slicing: Intra and Inter-Operational Analysis to Test Changes." *IEEE Transactions on Services Computing* (2018).
- Animesh Chaturvedi, "Subset WSDL to access Subset Service for Analysis", IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), 2014.
- Animesh Chaturvedi, "Automated Web Service Change Management AWSCM - A Tool", IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), 2014.
- Animesh Chaturvedi and Atul Gupta, "A Tool Supported Approach to Perform Efficient Regression testing of Web Services", 7th IEEE International Symposium on Maintenance and Evolution of Service oriented and Cloud Based Systems (IEEE MESOCA), 2013.
- Animesh Chaturvedi, "Reducing cost in regression testing of web service", 6th CSI International Conference on Software Engineering (6th CONSEG) on IEEE.

Quality of Service and Service Level Agreement (Monitoring, & Resource Management/Provisioning)

1. Cloud Life Cycle
2. Cloud Deployment Scenarios
3. Cloud Service Testing
4. Web Service Slicing for Regression Testing of Services
5. Cloud Service Evolution Analytics
6. **Quality of Service and Service Level Agreement** (Monitoring & Resource Management/Provisioning)

Quality of Service (QoS)

Service performance measurement in

- telephony,
- computer network,
- cloud computing service,
- users of the network

Quantitatively QoS measure in network service

- packet loss,
- bit rate,
- throughput,
- transmission delay,
- availability,
- jitter, etc.

https://en.wikipedia.org/wiki/Quality_of_service

Service Level Agreement (SLA)

Contract between Service provider and Consumer.

- Quality,
- Availability,
- Responsibilities

Different levels SLAs:

- Customer-based SLA: an individual consumer group
- Service-based SLA: all consumers using the services
- Multilevel SLA: different set of consumers for similar services
  - Corporate-level SLA: all generic Service Level Management (SLM)
  - Customer-level SLA: all SLM issues of particular consumer group
  - Service-level SLA: all SLM issue relevant to the specific services

https://en.wikipedia.org/wiki/Service-level_agreement

Service Level Agreement (SLA)

Web Service Level Agreement (WSLA):

- SLA for Web service monitoring

SLA contract:

- QoS parameters;
- SLA negotiation;
- SLA monitoring;
- SLA violation detection; and
- SLA enforcement

https://en.wikipedia.org/wiki/Service-level_agreement

Performance Metrics

SLA contains service-performance metrics

- TAT (Turn Around Time): Time taken to complete a certain task.
- TRT (Total Resolution Time): Total time taken to complete a certain task.
- MTTR (Mean Time To Recover): Time taken to recover after an outage of service.
- Uptime: Network uptime, Power uptime, etc.

https://en.wikipedia.org/wiki/Service-level_agreement

Resource Management & Provisioning

Public cloud services can be used with three Cloud provisioning:

- **Consumer self-provisioning:** Consumer contract and pay as per usage for cloud services directly to provider,
  - e.g. Institute Google or Microsoft domain (email, form, docx, excel etc.).
- **Advanced provisioning:** Consumer contract and pay in advance for resources and services
  - e.g. online event management system
- **Dynamic provisioning:** Provider allocates resources as per consumer usage, then de-provisioning when resources are not in use.
  - Consumer pays as per usage
- **Provisioning and orchestration:**
  - create, modify, and delete resources
  - orchestrate workflows and management of workloads

https://en.wikipedia.org/wiki/Cloud_management

Elasticity and Resource Provisioning

Elasticity: provisioning and de-provisioning resources in an autonomic manner,

For Elasticity **Avoid**

- Over-provisioning: allocating more resources than required,
  - Issue: pay for the useless resources
- Under-provisioning: allocating fewer resources than required
  - Issue: poor service performance, e.g. slow or unreachable
  - Issue: loses customers

https://en.wikipedia.org/wiki/Elasticity_(cloud_computing)

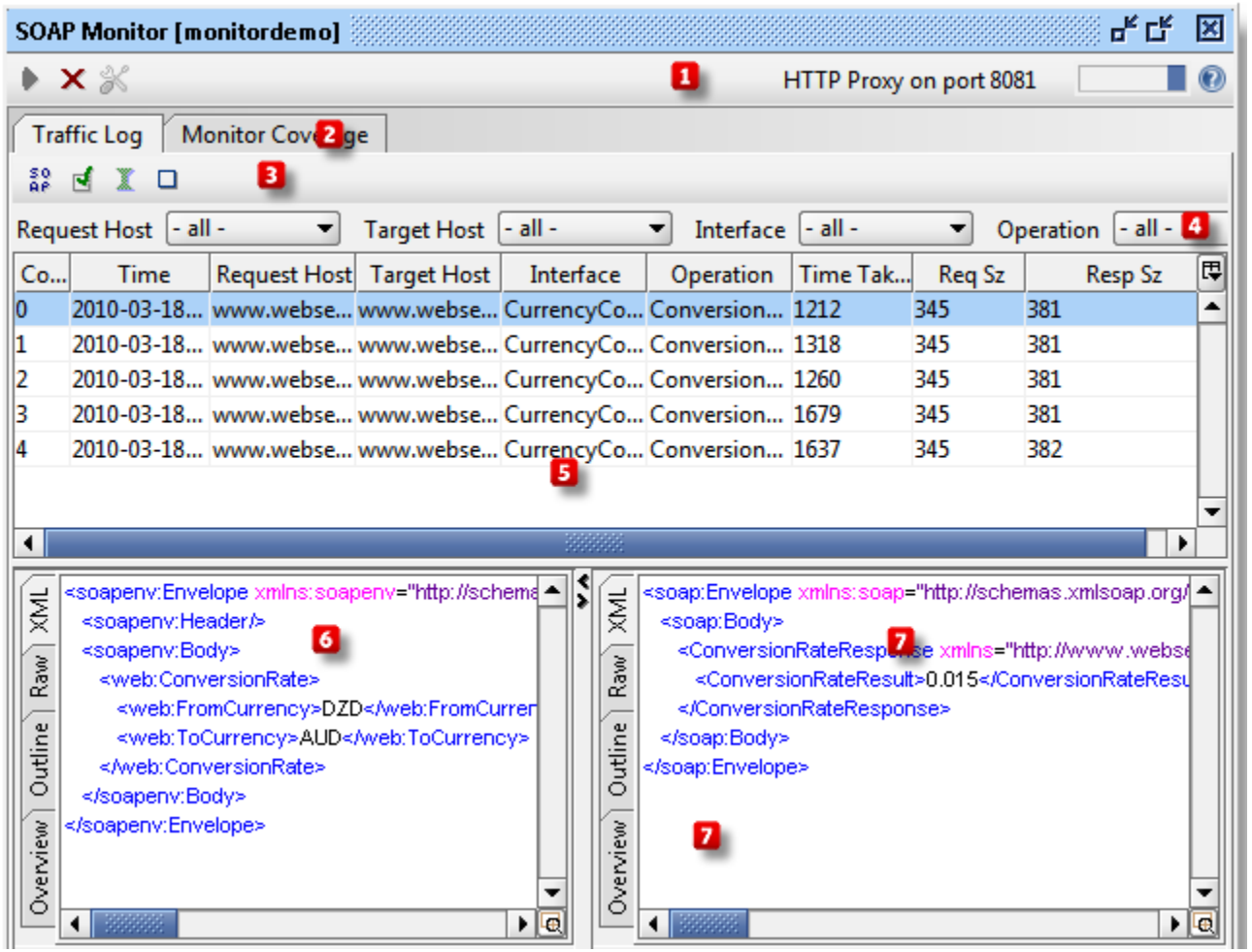
Monitoring

- Monitoring for REST and SOAP APIs
- Monitor Microservices and APIs that affect internal applications and decelerate innovation.
- Monitor internal APIs that serve
  - customers,
  - private network,
  - local network,
  - data center, or
  - private cloud.
- Monitoring APIs throughout the life-cycle
  - to detect and fix issues early on,
  - aiding to continuous development,
  - deployment, and
  - IT Operations.

https://www.soapui.org/docs/api-monitoring/

SOAPUI Monitor

1. Toolbar	Overall control of the Soap Monitor
2. Monitor Coverage Tab	Place where you can check Coverage
3. Traffic Log Tab Toolbar	control of the traffic Log
4. Filter Bar	filters for monitor
5. Trafic Log	Trafic Log panel
6. Request Tab	Request panel
7. Response Tab	Response panel



https://www.soapui.org/docs/http-recording/reference/http-monitor/

Advantage of Monitoring

Monitoring is useful for tasks

- prevents failures, e.g. by replacing a soon to be unavailable service with another equivalent;
- verifying that a service invocation meets given pre- and post- conditions; and
- triggering recovery actions when needed.

Canfora, Gerardo, and Massimiliano Di Penta. "Testing services and service-centric systems: Challenges and opportunities." *IT Professional* 8.2 (2006): 10-17.

QoS and SLA

- The QoS of services can vary over time with monitoring
- Testing to guarantee the SLAs
- Different stakeholders test individual services or service-centric systems to ensure or verify SLA
- Testing Levels
  - Service functional testing
  - Service non-functional testing
  - Integration testing
  - Regression testing

Canfora, Gerardo, and Massimiliano Di Penta. "Testing services and service-centric systems: Challenges and opportunities." *IT Professional* 8.2 (2006): 10-17.

Web Services Interoperability

- WS-I (http://www.ws-i.org) is an open industrial organization that promotes interoperability across
  - platforms,
  - operating systems, and
  - programming languages.
- WS-I helps define protocols for the interoperable exchange of messages between Web services.

Canfora, Gerardo, and Massimiliano Di Penta. "Testing services and service-centric systems: Challenges and opportunities." *IT Professional* 8.2 (2006): 10-17.

QoS and SLA: Service Maintenance

- The scope, quality, and responsibilities of a service provider regarding the service consumers.
- To maintain QoS, service providers monitor their web service continuously.
- Change analysis enables regression testing of web services to manage the QoS, which, in turn, helps to enforce SLA requirements.
- Intra-Operational and Inter-Operational Change Analysis based regression testing help in maintaining QoS and SLAs.
- Reduction in the effort of regression testing can lead to reduction in the effort of monitoring and maintaining QoS. Thus, it lowers the effort required to guarantee the SLA.
- Construct composite services while keeping satisfactory QoS.

Animesh Chaturvedi, and David Binkley. "Web Service Slicing: Intra and Inter-Operational Analysis to Test Changes." *IEEE Transactions on Services Computing* (2018).

QoS and SLA: Service Maintenance

- Service providers often update or enhance a service to meet new requirements, which may cause an SLA violation where an incorrect change leads to incorrect behaviour.
- Thus, QoS monitoring is required to check and re-establish the specific QoS described in the SLA.
- Reduced cloud/web service maintenance effort can further reduce the effort required to guarantee the QoS found in the SLA.
- Regression testing's goal is to maintain QoS and SLA; reducing the regression test suite can save effort, and thus reduce costs.

Animesh Chaturvedi, Aruna Tiwari, Shubhangi Chaturvedi, and Dave Binkley "Service Evolution Analytics: Change and Evolution Mining of a Distributed System", *IEEE Transactions on Engineering Management* (2020).

References Publications

1. Kohlborn, Thomas, Axel Korthaus, and Michael Rosemann. "Business and software service lifecycle management." *2009 IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2009.
2. Joshi, Karuna P., Yelena Yesha, and Tim Finin. "Automating cloud services life cycle through semantic technologies." *IEEE Transactions on Services Computing* 7.1 (2012): 109-122.
3. Sokol, Annie W., and Michael D. Hogan. *NIST Cloud Computing Standards Roadmap*. No. Special Publication (NIST SP)-500-291r2. 2013.
4. Animesh Chaturvedi, and David Binkley. "Web Service Slicing: Intra and Inter-Operational Analysis to Test Changes." IEEE Transactions on Services Computing (2018).
5. Animesh Chaturvedi, Aruna Tiwari, Shubhangi Chaturvedi, and Dave Binkley "Service Evolution Analytics: Change and Evolution Mining of a Distributed System", *IEEE Transactions on Engineering Management* (2020).

References links

1. https://netbeans.org/features/java-on-server/web-services.html
2. https://www.eclipse.org/webtools/ws/
3. https://jmeter.apache.org/
4. https://jmeter.apache.org/usermanual/build-ws-test-plan.html
5. https://www.soapui.org/
6. https://www.soapui.org/tools/soapui/
7. https://www.soapui.org/docs/soap-and-wsdl/
8. https://www.soapui.org/docs/rest-testing/
9. https://www.soapui.org/docs/http-recording/reference/http-monitor/
10. https://www.soapui.org/docs/api-monitoring/
11. https://en.wikipedia.org/wiki/Quality_of_service
12. https://en.wikipedia.org/wiki/Service-level_agreement
13. https://en.wikipedia.org/wiki/Cloud_management
14. https://en.wikipedia.org/wiki/Elasticity_(cloud_computing)


תודה רבה
Hebrew
Ευχαριστώ
Greek
Danke
German
Спасибо
Russian
धन्यवादः
Sanskrit
Merci
French
ধন্যবাদ
Bangla
நன்றி
Tamil
شكراً
Arabic
ಧನ್ಯವಾದಗಳು
Kannada
Thank You
English
നന്ദി
Malayalam
Grazie
Italian
ధన్యవాదాలు
Telugu
આભાર
Gujarati
多謝
Traditional Chinese
ਧੰਨਵਾਦ
Punjabi
धन्यवाद
Hindi & Marathi
Gracias
Spanish
多谢
Simplified Chinese
https://sites.google.com/site/animeshchaturvedi07
Obrigado
Portuguese
ありがとうございました
Japanese
ขอบคุณ
Thai
감사합니다
Korean