**WEB SERVICE USING EAPI FRAMEWORK WITH WEBLOGIC**

SEWP ZG628T DISSERTATION

By

##### **GODLY SURESH 2013HW70264**

Dissertation Work carried out at Wipro Technologies, Bangalore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN) INDIA

November 2017

SEWP ZG628T DISSERTATION

**WEB SERVICE USING EAPI FRAMEWORK WITH WEBLOGIC**

Submitted in partial fulfilment of the requirements of

M.S. Software Engineering Degree Program

By

##### **GODLY SURESH 2013HW70264**

Under the supervision of **Arun Kumar v Technologist**

Dissertation work carried out at Wipro Technologies, Bangalore

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN) INDIA

November 2017



##### **ACKNOWLEDGEMENTS**

I would like to thank my project supervisor Arun Kumar for his guidance, efforts and patience. He has been extremely supportive and co-operative in this whole effort. His valuable comments have been immensely helpful in enhancing the quality of work.

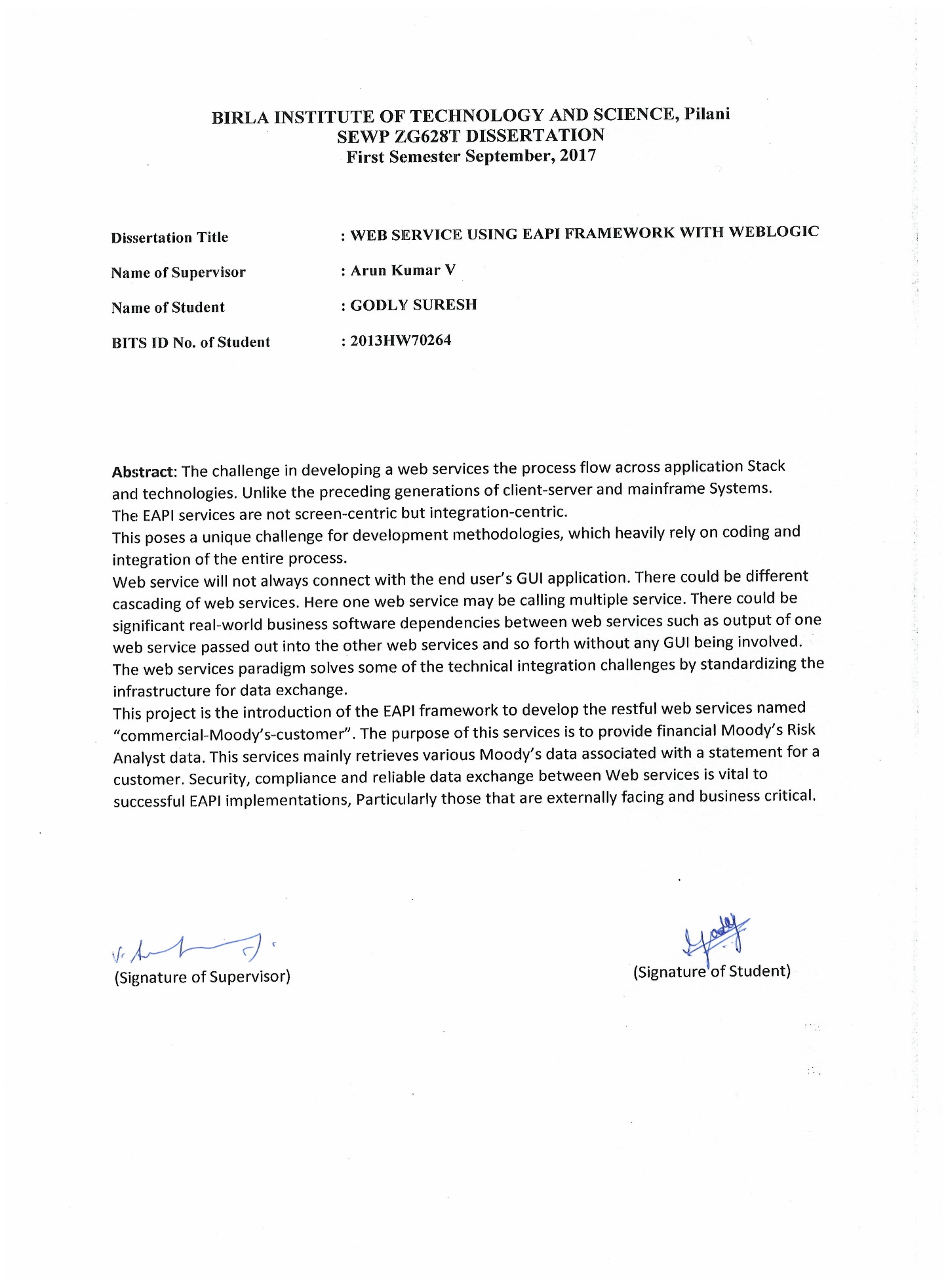
I would also like to thank my senior colleagues Manoj and Sathiya for the support and guidance extended to me during the dissertation.

I am also thankful to BITS, Pilani and the Training department for all their efforts in organizing this course.

My sincere regards and thanks to WASE department for the information and support.







**CONTENTS**

1. Web Service Development – Introduction and nuances 11
2. [Web Service Architecture 12](#_TOC_250015)
3. Benefits of using webservice 14
4. Web Service Terms 18
5. [Challenges in Web Service Development 19](#_TOC_250014)
6. [Types of Web Service Development 20](#_TOC_250013)
7. JAX-WS & eAPI Framework Comparison 21
8. [eAPI Framework Components 22](#_TOC_250012)
9. [Features of eApi 24](#_TOC_250011)
10. eAPI Project Layers 27
11. [Testing eAPI Projects 29](#_TOC_250010)
12. [Stub 30](#_TOC_250009)
13. [Junits 31](#_TOC_250008)
14. [Overview of Commercial-Moody’s Risk Analysis api 32](#_TOC_250007)
15. Activity Diagram for commercial-moodys-customers 33
16. [Soap UI project with request & Response 34](#_TOC_250006)
17. [SOA testing techniques 35](#_TOC_250005)
18. [Benefits of Testing SOA 39](#_TOC_250004)
19. [Conclusion 40](#_TOC_250003)
20. [Plan of work (16 Weeks) 41](#_TOC_250002)
21. [References 42](#_TOC_250001)
22. GLOSSARY 40
23. Candidate Project Details 41
24. [Checklist 43](#_TOC_250000)

## List of Symbols & Abbreviations used

WSDL—Web Service Definition Language SOA – Service Oriented Application

XML – Extensible Markup Language

UDDI - Universal Description, Discovery and Integration SOAP - Simple Object Access Protocol

RPC - Remote Procedure Calls

BPEL – Business Process Execution Language TDD – Test Driven Development

|  |  |
| --- | --- |
| **List of Tables Figures** |  |
| Framework Comparison | 20 |
| eApi framework components | 21 |
| **List of Figures** |  |
| Figure 2.0 - Web Service Architecture | 13 |
| Figure 3.0 - Rest Web Service | 17 |
| Figure 10.0 -Project Layers | 26 |
| Figure 12.0 - JAX-RPC API | 29 |
| Figure 15.0 - Activity diagram | 32 |
| Figure 16.0 - Soap UI Project | 33 |

## Introduction and nuances

To discuss the various challenges faced and methods followed by the Dev teams developing the business based in the environment. The best tools, solutions and practices involved in the Eapi specialization.

**Web Services What & Why?**

Web services are open standard (XML, SOAP, HTTP etc.) based Web applications that interact with other web applications for the purpose of exchanging data.

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.

A Webservice is available over the Internet or private (intranet) networks. It uses a standardized XML messaging system and it is not tied to any one operating system or programming language. They are self-describing via a common XML grammar as well.

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP.

The basic components of web services platform is XML + HTTP. All the standard web services work using the following components

* + - SOAP (Simple Object Access Protocol)
    - UDDI (Universal Description, Discovery and Integration)
    - WSDL (Web Services Description Language)

# Web Service Architecture

#### Web Service Roles

There are three major roles within the web service architecture:

##### Service Provider

This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

##### Service Requestor

This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

##### Service Registry

This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.

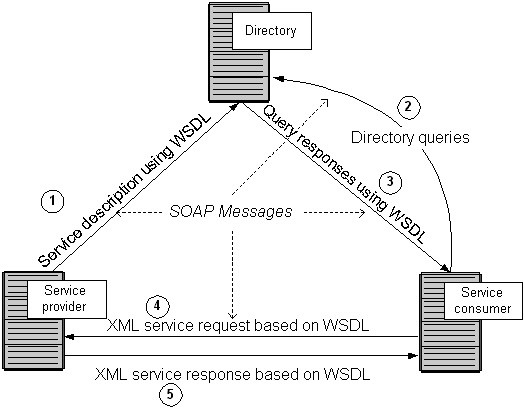


Fig 2.0

The Web Services Description Language (WSDL) forms the basis for the original Web Services specification. The following figure illustrates the use of WSDL. At the left is a service provider. At the right is a service consumer. The steps involved in providing and consuming a service are:

1. A service provider describes its service using WSDL. This definition is published to a repository of services. The repository could use Universal Description, Discovery, and Integration (UDDI). Other forms of directories could also be used.
2. A service consumer issues one or more queries to the repository to locate a service and determine how to communicate with that service.
3. Part of the WSDL provided by the service provider is passed to the service consumer. This tells the service consumer what the requests and responses are for the service provider.
4. The service consumer uses the WSDL to send a request to the service provider.
5. The service provider provides the expected response to the service consumer.

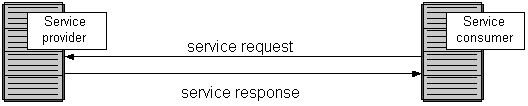


Fig 2.1

1. **Here are the benefits of using Web Services:**

* Exposing the Existing Function on the network

A web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. Web services allow you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.

### Interoperability

Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use the web services. For example, a VB or

.NET application can talk to Java web services and vice versa. Web services are used to make the application platform and technology independent.

### Standardized Protocol

Web services use standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack. This standardization of protocol stack gives the business many advantages such as a wide range of choices, reduction in the cost due to competition, and increase in the quality.

### Low Cost of Communication

Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.

### XML-Based

Web Services uses XML at data representation and data transportation layers. Using XML eliminates any networking, operating system, or platform binding. Web Services based applications are highly interoperable application at their core level.

### Loosely Coupled

A consumer of a web service is not tied to that web service directly. The web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

### Coarse-Grained

Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service.

Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic

### Ability to be Synchronous or Asynchronous

Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions.

Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

### Supports Remote Procedure Calls(RPCs)

Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.

Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms.

A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

* **Supports Document Exchange**

One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be as simple as representing a current address, or they can be as complex as representing an entire book or Request for Quotation (RFQ). Web services support the transparent exchange of documents to facilitate business integration.

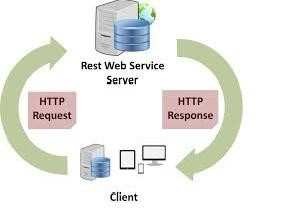


Fig 3.0

# Some pointers before going further

* + **SOAP** (**Simple Object Access Protocol)** is XML based protocol for accessing a Web Service. It allows the applications to exchange information over **HTTP**, **SMTP** or **TCP**.
  + SOAP is Platform independent and language independent i.e. it provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.
  + Extensible Markup Language, a new markup language published by the W3C to address the limitations of HTML
  + **XML** Schema describes the structure of an XML document. XML Schema language is also referred to as XML Schema Definition **(XSD)**
  + **XML namespaces** provide a way to distinguish deterministically between XML elements that have the same local name but are, in fact, from different vocabularies.
  + **Web Services** refers to the technologies that allow for making connections. Services are what you connect together using Web Services. Web Services can make your applications Web applications
  + **WSDL** (Web Services Description Language) is an XML -based language for describing Web services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes.
  + Representational state transfer (**REST)** is a style of software architecture for distributed hypermedia systems such as the World Wide Web.
  + Rest promotes and recommends generic operations on resource and uses HTTP methods: PUT GET POST DELETE. It Utilizes the caching mechanism
  + Web Services does not promote generic operations. The First generation only utilizes HTTP POST and has No caching capabilities
  + REST is over HTTP, but SOAP can be over any transport protocols such HTTP, FTP, STMP, JMS etc.

SOAP is using soap envelope, but REST is just XML.

# Challenges in Web Service Development

* + Loose coupling between services
  + The services are loosely coupled and are assembled at the run time
  + Heterogeneous computing environment
  + Tools, technology and platform at the consumer end and at the provider end can differ
  + Lack of total control on services
  + Boundaries of the service may span across the organization boundaries; transactions are distributed
  + Multiple points of failure
  + There may be multiple entry and exit points based on how the services are orchestrated
  + Performance, reliability & availability
  + Difficult to predict the future usage; hence ensuring performance, reliability and availability is a challenge
  + Security concerns - Security needs to be ensured at various points due to the exposure of the services to various threats
  + Asynchronous Services
  + Testing whether the asynchronous messages are processed in the right order
  + Integration Centric
  + Different from web applications as SOA is integration centric
  + Technical Skills Required for Testing

Testing team requires SOA knowledge for creating simulated environment, complex harness creation, script execution, XML validation

# Types of Web Service Development

Web Service can be developed in two ways SOAP and REST.

###### JAX-WS

Java API for XML Web Services (JAX-WS) is a technology for building web services and clients that communicate using XML. JAX-WS allows developers to write message-oriented as well as Remote Procedure Call-oriented (RPC-oriented) web services. In JAX-WS, a web service operation invocation is represented by an XML- based protocol, such as SOAP.

JAX-WS services use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Such systems often contain a machine-readable description of the operations offered by the service, written in the Web Services Description Language (WSDL), an XML language for defining interfaces syntactically.

###### JAX-RS

JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. REST is well suited for basic, ad hoc integration scenarios. RESTful web services, often better integrated with HTTP than SOAP-based services are, do not require XML messages or WSDL service–API definitions.

RESTful web services are built to work best on the Web. REST is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web.

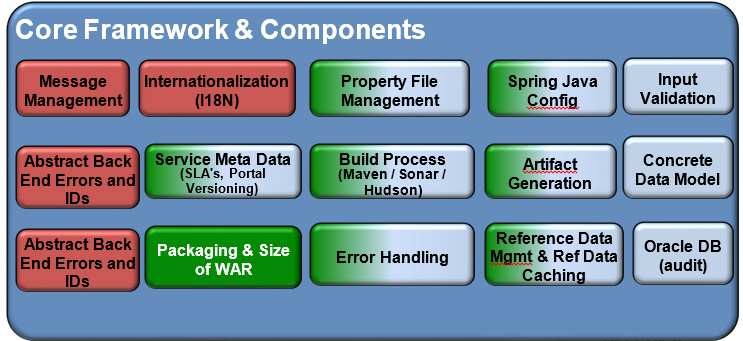
They have a lightweight infrastructure that allows services to be built with minimal tooling, developing RESTful web services is inexpensive and thus has a very low barrier for adoption.

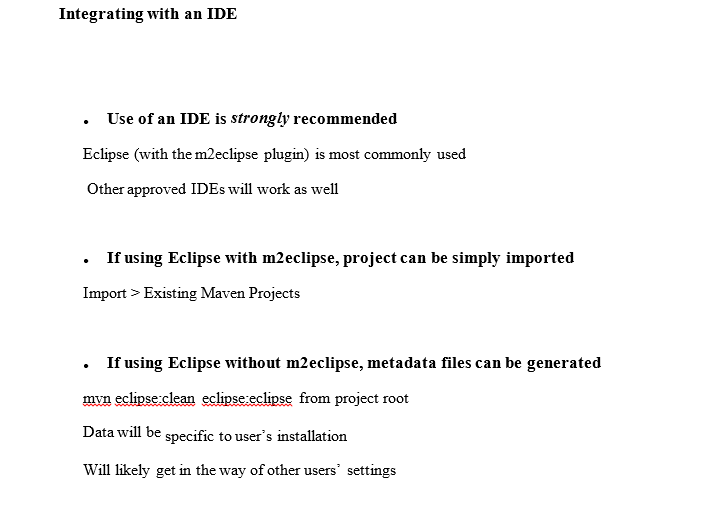
# Framework Comparison

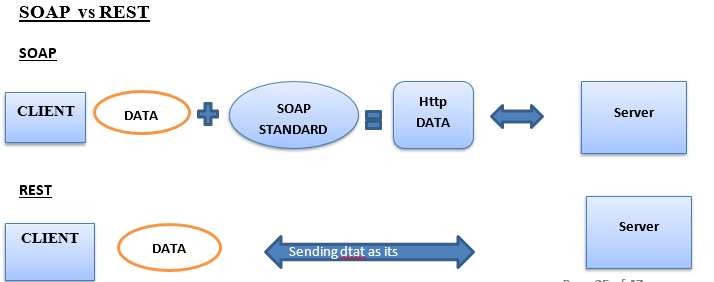
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature** | **Existing JAX-WS Framework** | | | | **eAPI Framework** | |
| Primary Protocol | SOAP | | | | REST | |
| Framework Solution | Weblogic Shared Libraries. EJB Support Required | | | | Container Independent. Tested Frequently in Jetty | |
| Spring Version | 2.5 | | | | 4.0 | |
| Reference Data | Custom solution with source code management inserted into database during deployment. | | | | New Client with Cache support. Restricted to Global Reference Data. | |
| Multi-Threading | Fork/Join | | | | Java Futures/@Async | |
| Code starter | Custom generator | solution | named | service | Maven Archetype | |
| Authentication | SSL Certificate | | | | RSA Token | |
| Authorization | SSL Certificate | | | | JAAS CLIENT and USER roles at resource level. | |
| Message Management | Not Supported | | | | Java Message (Internationalization Support) | Bundle |
| Caching | Not Supported | | | | Spring Caching | |
| Audit Logging | Not Supported | | | | UPF with MQ support added | |

# eAPI Framework Components

A software framework is a reusable software platform to develop software applications Software frameworks include application programming interfaces (APIs) that bring together all the different components to enable development of a project or solution. The primary protocol used in the eApi framework is REST. Framework Solution is Container Independent and tested frequently in Jetty server. Caching is done using Spring Caching.







# Features of eApi

##### Code Formatters and Templates

* **eAPI contains code formatters and templates for Eclipse** Ensures code follows a consistent format across all developers Removes false change notifications due to formatting differences Automates addition of standard Capital One footer

###### Use of formatters is mandatory for eAPI framework submissions

* **Projects developed with the eAPI archetypes can run on a number of application servers**

Oracle WebLogic & Eclipse Jetty Apache Tomcat (coming soon!)

###### Not all functionality available in Jetty

Less-used features like JMS are not supported currently Primarily intended to increase developer productivity

##### Encrypted Data

* + **Some data should not be exposed to external clients** Account Numbers are a classic example of this Data may still need to be referenced in future calls to the server

###### The Framework enables declarative encryption of parameters and attributes

Annotate resource parameters with @NonPublicInformation

Annotate model fields with @XmlJavaTypeAdapter(NonPublicInformationAdapter.class) Value will be decrypted when passed into class

Value will be encrypted when returned from class

##### Caching Data

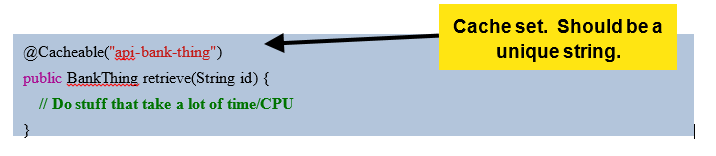
###### The Framework provides the ability to cache call results

Useful for performance Remote calls

Calls that require expensive processing Results stored in cache on first call Later calls return data from cache

Cache can be flushed automatically as well

###### To cache data on call, simply annotate method with @Cacheable



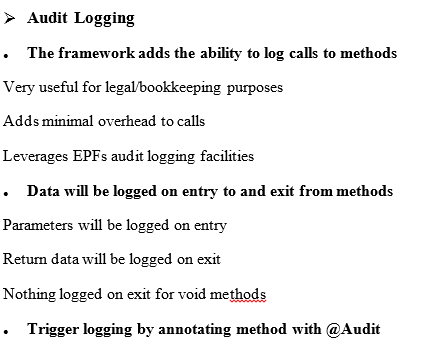
* Caching Limitations
* **Caching has the same limitations as Audit Logging**

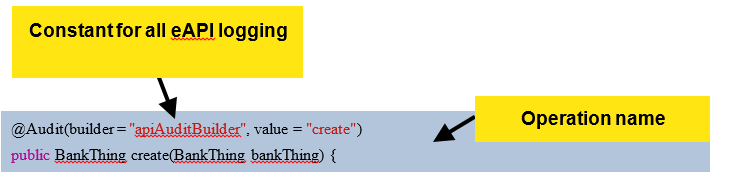
Can only cache public methods Methods must be defined in an interface Methods must be called by another class

###### If caching is needed outside these bounds, double-check your design

Code can usually be refactored to be cacheable

If not, contact framework team to discuss solutions

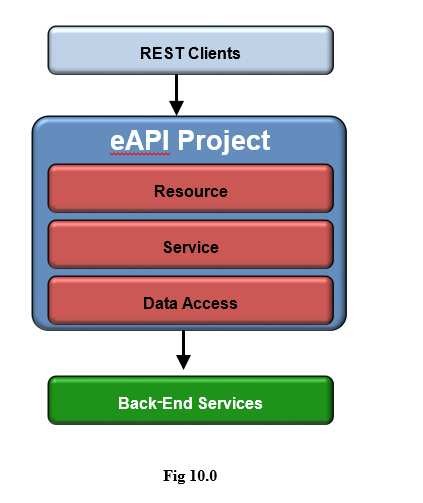




# Project Layers

###### eAPI archetype-generated projects split the application into layers

* Simplifies individual code
* Easier to unit test classes



* Resource Layer handles Requests and Responses

Service Endpoints and supporting logic

Only logic to translate data for business layer

##### Service Layer handles business logic

Orchestration and specialized processing Should not access back end services Most logic will be here

##### Data Access Objects handle external service access

Translates to and from back-end services (IS, Legacy services, etc.)

##### **T**he eAPI Framework handles conversion with Spring’s Conversion Framework

Isolates conversion from data access Allows for reuse of conversions Simplifies conversion on DAO side

# Testing eAPI Projects

###### Sample Unit and Integration Tests are generated by the eAPI archetype

* **All nontrivial code in an eAPI project should be covered by unit tests**

Testing single classes *only* ensures that tests don’t fail due to bugs in dependencies Use Mockito to mock dependencies as needed

Annotate test driver with @Category(UnitTest.class)

###### Integration Tests should also be added as appropriate

Useful for Resources

Annotate test driver with @Category(IntegrationTest.class)

###### Tests can be automatically be run via Maven

Unit Tests: run standard build from project root

Integration Tests: run with devint-test Maven Profile from project

# Stub

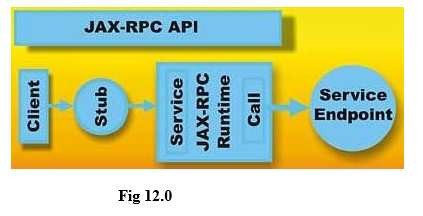
A stub is a small program routine that substitutes for a longer program, possibly to be loaded later or that is located remotely.

For example, a program that uses Remote Procedure Calls ( RPC ) is compiled with stubs that substitute for the program that provides a requested procedure. The stub accepts the request and then forwards it (through another program) to the remote procedure. When that procedure has completed its service, it returns the results or other status to the stub which passes it back to the program that made the request.

Since in many client side developments the developer may not have access to the database and will not be able to know whether the developed code is correctly working or not in this situation we to for a coding called Stub.

In case of web services we stub the expected response on the local system for unit testing. The Stub is where the normal call is routed to a particular mock response. Here we create the expected response in the form of XML which has the data same as that of the external database. In the method call, we call the StubDAbean instead of actual DAbean so that it hits the xml and fetches the response required for the user. We create all the scenarios that may arrive for testing the service and check it if the service is working properly or not.

The main purpose of using stubs is to check the flow of the code against the mock response created and also we arrive at a conclusion that the developed code is well to go for production without any issues.



# JUnits

JUnit is a unit testing framework for the Java Programming Language. It is important in the test driven development.

###### Why use JUnit?

* It’s free!
* It is simple and elegant to use.
* It is easy and inexpensive to write tests using the JUnit testing framework.
* JUnit tests checks their own result and provide quick visual feedback.  Tests can be composed into Test Suites.
* It is integrated into IDE’s like Eclipse and NetBeans.
* JUnit tests allow you to write code faster which increasing quality  JUnit is elegantly simple. It is less complex & takes less time.
* Junit shows test progress in a bar that is green if test is going fine and it turns red when a test fails.

**Some of the Junit Annotations:**

@Test- Mentioned at the beginning of each testcase.

@Before – Any method written under @Before is executed before @Test.

@After- method written run after every @Test method

@Before class –runs only once for the test class at the beginning

@Ignore—Tests will be Ignored at the annotation

# Overview of Commercial-Moody’s Risk Analysis api

The purpose of this service is to provide financial Moody’s Risk Analyst data. It retrieve’s various Moodys financial data associated with a statement for a customer.

The service name is “**commercial-moodys-customers**”. And the operation which we are going to implement is “retrieveCustomerStatementDetails”. It uses Http “Get” method.

This service in turn calls the MoodysRiskAnalysisISV3 service to fetch the response . The API takes three request parameters:

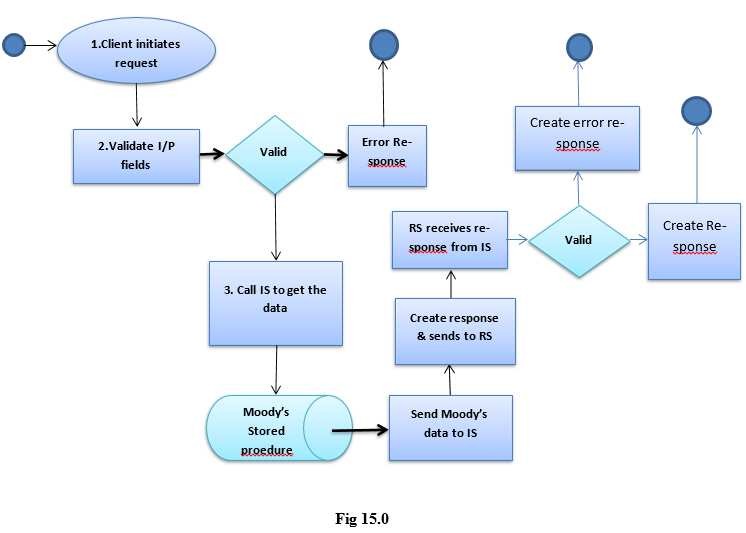
* moodysId- This is the customer id assigned by the Moody's system. This field is known by the user. Data Type: Number
* statementId- Each customer will have multiple statements. Each statement will have a unique id assigned by the system. Data Type: Number
* statementContext- A numeric value that identifies the historical vs projection context of the statements, which is unique within the customer. Data Type: Number

Request header parameters like User-Id, Api Key …and so on.

###### And the response parameters are:

* accounting Standard - Data Type: String(75)
* accountsPayableTradeAmount - Data Type: Decimal(26,6)
* adjustedCashFlowLeverageRatio - Data Type: Decimal(26,6)
* adjustedDebtAmount - Data Type: Decimal(26,6)
* adjustedLiabilitiesToAdjustedNetWorth - Data Type: Decimal(26,6)
* adjustedLiquidityRatio - Data Type: Decimal(26,6)
* adjustedNetWorth - Data Type: Decimal(26,6)
* adjustmentToRetainedEarningsAmount - Data Type: Decimal(26,6)





# Soap UI project with request & Response

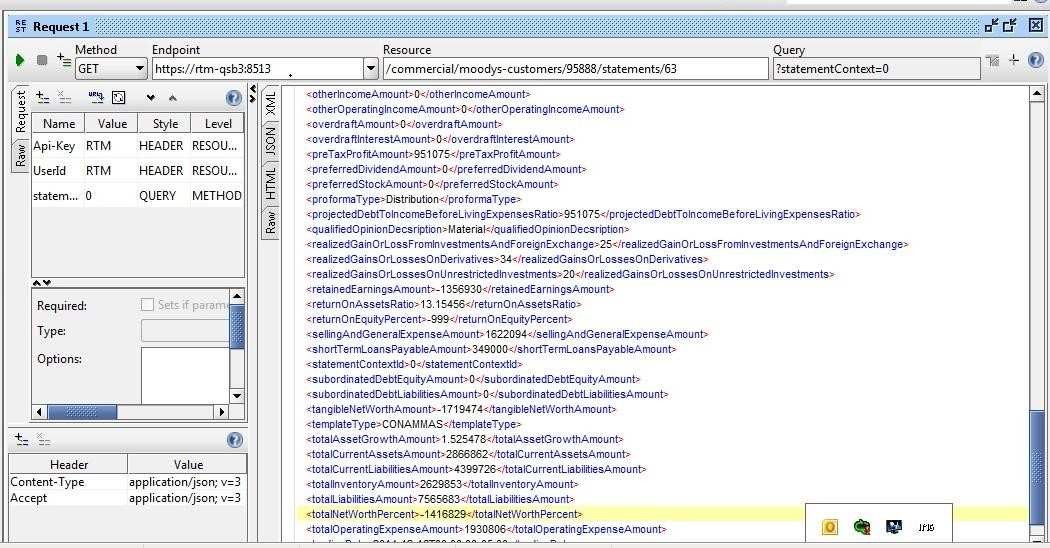


Fig 16.0

# SOA Testing Techniques

#### Different levels of SOA testing:

###### Service Agility Testing

* 1. Configurations
  2. Business Rules
  3. Policies

###### BPEL Level-1 Testing

* 1. Compensating transactions
  2. Service unavailability impact

###### BPEL Level-2 Testing

* 1. Workflow Testing
  2. Events Testing

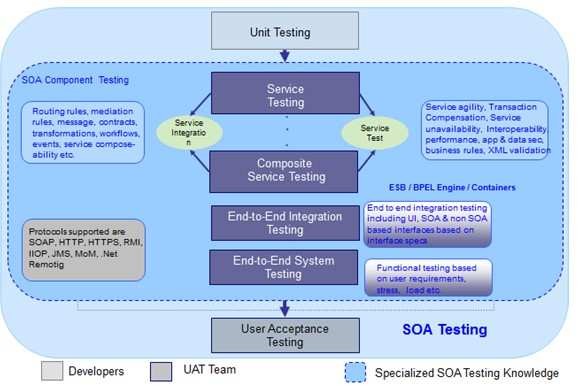
###### Security Testing

* 1. Denial-of-Service (DOS)
  2. Vulnerability
  3. Context propagation/Federation

###### Service Design Testing

* 1. Interoperability Testing
  2. Service-App Integration Testing
  3. Reuse Testing (Consumability and Composability)
  4. Service Data Testing

###### SOA Performance Testing



**Fig 17.0**

**Service Agility Testing:** This procedure requires complete understanding of the requirements for which the service is designed. The review should focus on identifying the volatile parts of the requirements, since agility is all about externalizing them into parameters via configurations, rules and policies. All of these parameters would change the behavior of a service or component in different contexts. Configurable items influence system functions, rules would influence business functions, and policies would influence security, monitoring, performance etc. of services. The reviewer should make sure that all the volatility is captured in either configurations or rules or policies

This essentially reinforces the fact that most of the iterative development models like TDD, SCRUM, and XP etc. are pushing the design and domain specialists to be more involved in testing and quality assessment activities.

Another important factor in agility testing is to measure how quickly service can be provisioned to accept a new consumer. Please refer to interface, implementation, data and packaging considerations of service for design guidelines.

One of the measures is to define certain test cases for data integration by changing some data entity semantics and changing the appropriate map to see how quickly the systems in question come back to normal operations.

###### BPEL Testing

In the case of composite services, two levels of testing are required i. Orchestration (BPEL) Testing ii. Process level testing.

**Level-1 Testing** is aimed at making sure that the sequence of service calls made to various systems will not leave any of the systems in an inconsistent state. This test also helps in Understanding the impact of one or more services in the chain not being available. Note that the more a service gets reused, the greater the impact associated with its downtime will be. Services are composed to form the business processes either partially or completely by orchestrating them in a defined sequence. These services may be hosted on different platforms with different underlying transactional engines. In the event of failure of any service invocation in the sequence, BPEL provides compensating transactions (service methods) to maintain the data in a clean state. All the compensating transactions have to be tested thoroughly with exhaustive test cases and data. It is essential to test the level of impact in case of service unavailability to the business process as a whole, and to check the remedial mechanisms provided and their effectiveness.

**Level-2 Testing** is aimed at testing the flexibility in the workflow design provided and also the Effectiveness of process visibility details captured via events.

Since SOA involves a lot of asynchronous communication across systems, it is possible that messages will not reach their destinations in an orderly fashion because of delays in processing at end-points (target systems). Some systems will respond quickly, and some slowly, depending on the load at any given point in time. Response messages must to be correlated centrally before they are dispatched to the target systems. Changes in workflow may result in changes in the delivery order of the messages from one system to another. Test cases should be aimed at testing this scenario to measure how quickly a workflow can be changed. The results are in fact a reflection on the extent of agility built in to the enterprise’s SOA.

Testing the above scenarios requires thorough involvement of the development team during the Testing process.

###### Security Testing

Typically, major SOA initiatives involve exposing services to and consuming services from the outside world. This fact opens up a host of vulnerabilities, such as DOS (denial-of-service) attacks, penetration, high volumes of spam data, etc. Typical security policies have to be enforced at the network level, middleware level and at the end-point level to create bullet proof SOA. Specific test cases aimed at targeting these policies need to be designed to fully test SOA security.

###### Service Design Testing

**Reuse Testing:** Knowledge of design decisions made and the utilization of data entities within various business processes would be validated by the domain knowledge of the business analyst to make certain that the interface or contract would work in all possible business scenarios, thus ensuring consumability and ability to compose.

**Interoperability Testing**: Since enterprise SOA development involves different technology platforms and development tools at the enterprise level, it is quite possible that developers use many different tools to generate the contracts (WSDL) and modify them in the process. Also, different runtime SOAP engines would interpret contracts differently, based on their implementations. Hence it is essential to make sure that all (web) services deployed are compliant with WS-I basic profile. Tools such as SOAPScope, SOAPTest and tools from WS- I org etc. Provide the necessary support in this testing process.

**Service – Application Integration Testing:** In a big SOA program, it is quite common that different services and applications are developed independently in parallel by multiple vendors in different geographical locations. Defining the contracts or interfaces at the beginning would help the development effort to go smoothly. But at the time of integration, it is useful to have an integrated test environment where services can be deployed, tested and integrated with Applications for overall testing.

**Data Level Testing:** For services, data must be tested for completeness, correlation across different consumers, separation of data for consumers, element level security as needed and ownership assessment by testing support services such as reporting and data archival mechanisms, etc.

# Benefits of Testing SOA

* Ensure the reliability, quality, security and interoperability of the Web service.
* Penetration testing integrated with functional testing for complete coverage.
* Uniform test suites can be rolled over from unit testing to functional testing to load testing to security testing.
* Prevent errors, pinpoint weaknesses, and stress test long before deployment.
* Verify data integrity and server/client functionality.
* Identify server capabilities under stress and load.
* Accelerate time to market.
* Leverage existing assets.
* Easier to integrate and manage complexity.
* More responsive and faster time-to-market.
* Reduce cost and increase reuse.
* Be ready for what lies ahead.

The last and best option is a fully integrated testing solution that enables testers to test at each point and end-to-end. This would include these key capabilities:

1. Creating and sending a message to a provider application;
2. Receiving and verifying a message to a consumer application;
3. Either sending or receiving messages to or from files when the applications aren't available or to isolate the infrastructure, and
4. Testing end-to-end through the user interface of the provider and consumer applications and the messages in between.

# Conclusion

In this report we examined types of Web services ,the eApi development process and the steps involved in it. The Key quality of a webservice development is to involve business analysts, executives and architects as a integral part of quality audits. The business analyst plays the very important role in gathering the requirement and sharing it to the developers of the team. They ensure the reuse and the required agility at the enterprise level. Going by the current technology trend in Webservice development where CICD (continuous integration and continuous deployment) where it makes easy for the testers to do the SOA testing hand in hand. I am concluding My dissertation report with few words quoting that “ Agile is all about adapting to changes. A core principle of agile is working at a constant pace, which in turn enables you to deliver at a constant pace, as opposed to working sporadically and delivering nothing.”

# Plan of work (16 Weeks)

|  |  |  |  |
| --- | --- | --- | --- |
| **Task #** **Tasks or subtasks to be done** | | **Planned** **duration** **in weeks** | **Status** |
| **1.**  **2.** | Requirement gathering | 3 | Done |
| Design Phase | 2 | Done |
| **3.** | Detailed study on Webservice | 3 | Done |
| **4.** | eApi framework | 3 | Done |
| **5.** | Testing the web services using SOA | 2 | Done |
| **6.** | Junits & stubs | 2 | Done |
| **7.** | Working on comments from mid semester and submission of final dissertation report to  the supervisor | 1 | Done |

# References:

1. Webservice terms:

<http://www.w3schools.com/xml/xml_services.asp>

1. Webservice Architecture:

<http://www.service-architecture.com/articles/web-> services/web\_services\_explained.html

1. Restful Web services:

https://docs.oracle.com/javaee/6/tutorial/doc/giqsx.html#gkcaw

1. knet website paper for Wipro:

An Introduction to SOA Testing Methodology - Dheeraj K, 2009

1. Stubs:

<http://whatis.techtarget.com/definition/stub>

1. Junits:

<http://junit.org/>

<http://www.tutorialspoint.com/junit/>

<http://wipro.udemy.com/>

1. Eapi Framework:

[http://pulse .kdc.capitalone.com/docs](http://pulse.kdc.capitalone.com/docs)

1. Capital One account: Client account in Wipro: Working experience in banking project on Web services

# 24.Checklist

1. Is the Cover page in proper format? Y
2. Is the title page in proper format? Y
3. Is the certificate from the Supervisor in proper format? Has it been signed? Y
4. Is abstract included in the Report? Is it properly written? Y
5. Does the table of Contents' page include chapter page numbers? Y
6. Is Introduction included in the report? Is it properly written? Y
7. Are the Paged numbered properly? Y
8. Are the Figures numbered properly? Y
9. Are the Captions for the Figures and Tables proper? Y
10. Are the Appendices numbered? Y
11. Does the Report have conclusions/Recommendations of the work? Y
12. Are References/Bibliography given in the Report? Y
13. Have the References been cited in the Report? Y
14. Is the citation of References/Bibliography in proper format? Y