A COLLABORATIVE FRAMEWORK FOR DYNAMIC SERVICE INTEGRATION OF LONG TERM COMPOSED SERVICES USING FSM

A Thesis

Submitted to Pondicherry University in partial fulfillment of the requirements for the award of the Degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

S.TIROUMALMOUROUGHANE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PONDICHERRY ENGINEERING COLLEGE (AUTONOMOUS)
PUDUCHERRY – 605 014
INDIA
MARCH 2018

Dr. P.THAMBIDURAI. M.E, Ph.D., F.I.E., LMISTE., LMCSI.
Professor of Computer Science & Engineering and Principal
Perunthalaivar Kamarajar Institute of Engineering & Technology (PKIET)
(Union Territory of Puducherry Institution)
Karaikal – 609 603.

CERTIFICATE

Certified that this Thesis entitled **"A COLLABORATIVE FRAMEWORK FOR DYNAMIC SERVICE INTEGRATION OF LONG TERM COMPOSED SERVICES USING FSM"** submitted for the award of the degree of **DOCTOR OF PHILOSOPHY in COMPUTER SCIENCE AND ENGINEERING** of the Pondicherry University, Puducherry is a record of original research work done by **Mr. S. TIROUMALMOUROUGHANE** during the period of study under my supervision and the Thesis has not previously formed the basis for the award of the candidate of any Degree, Diploma, Associateship, Fellowship or other similar titles. This Thesis represents independent work on the part of the candidate.

Place: **(Dr. P. THAMBIDURAI)**

Date: **Supervisor**

DECLARATION

I hereby declare that the Thesis entitled **"A COLLABORATIVE FRAMEWORK FOR DYNAMIC SERVICE INTEGRATION OF LONG TERM COMPOSED SERVICES USING FSM"** submitted to the Pondicherry University, Puducherry, India for the award of the Degree of **DOCTOR OF PHILOSOPHY** in **COMPUTER SCIENCE AND ENGINEERING** is a record of the original research work done by me under the supervision of **Dr. P. THAMBIDURAI,** Professor of Computer Science & Engineering and Principal, Perunthalaivar Kamarajar Institute of Engineering and Technology, Karaikal and the work has not been submitted either in whole or in part for any other Degree / Diploma / Certificate or any other title by an university / Institution before.

(S.TIROUMALMOUROUGHANE)

ACKNOWLEDGEMENT

With immense pleasure and deep sense of gratitude, I would like to place on record my grateful thanks to all those who have contributed to the successful completion of this research work. It is my pleasure to express my profound sense of gratitude to my supervisor **Dr. P. THAMBIDURAI,** Professor and Principal, Perunthalaivar Kamarajar Institute of Engineering and Technology, Karaikal for his invaluable guidance as well as his timely advice during the period of the research work. His consistent encouragement and personal attention are accountable for the successful completion of this Thesis.

I am grateful to express my sincere requital to my Doctoral Committee members **Dr. N.P. GOPALAN**, Professor in Department of Computer Applications, National Institute of Technology, Trichy and **Dr. M. ARAMUDHAN**, Associate Professor and Head, Department of Information Technology, Perunthalaivar Kamarajar Institute of Engineering and Technology, Karaikal for providing valuable comments and constructive suggestions to improve the quality of this research.

I express my sincere thanks to **Dr. D. GOVINDARAJALU,** former Principal, **Dr. P. DANANJAYAN,** present Principal and **Dr. K.VIVEKANANDAN**, **Dr. N. SREENATH** and **Dr. D. LOGANATHAN** former Heads of Departments and **Dr. M. SUGUMARAN** present Head of the Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry for their wholehearted support and for allowing me to make use of the facilities in the college for the research work.

With deep sense of gratitude, I wish to express my sincere thanks to **Dr. V. GOVINDASAMY,** Associate Professor, Department of Information Technology, Pondicherry Engineering College, Puducherry for the moral support and timely help rendered during the course of this research work. I also extend my sincere gratitude to my friend **Dr. M. THIRUMARAN**, Assistant Professor, Department of Computer Science and Engineering, Pondicherry Engineering College for his valuable help throughout my research.

I express my sincere thanks to my parents **Mr. S. SANTHANAM** and **Mrs. A. JAYALAKSHMI** and to my brothers, sisters and other well-wishers especially to my wife **Mrs. N. GEETHA** and my son **T. ANISH,** for their love, support and encouragement rendered during the period of the research work. Their continuous encouragement and support throughout this research has helped in the successful completion. Finally, I thank the Almighty, for making this a part of my destiny.

S. TIROUMALMOUROUGHANE

ABSTRACT

Today's challenging circumventions are pushing organizations to spread out their attention and to collaborate with their professional adherents to convey their inevitabilities. The Web services are the evolving technology in the field of business processes where the services offered by the organization are managed through the framework called Change Management Framework (CMF). By this framework any changes for the service offered by the organization can be added, modified or replaced in the form of the service request presented by either the customer or other business parties. Thus it is helpful for the organization to develop itself by satisfying the customer requirements in an autonomous manner.

Service Integration has turned out to be a serious issue in the fulfillment of user request. Whenever an enterprise discovers a better requisite to conventionally interact with their professional adherents and apportion their business logics to convey the essentialities, the features of their existing services have to be improved. But integrating Web accommodation logics of functionally different organization leads to exponential quandary which lets developers to grasp the whole Web service and to determine appropriate technique to integrate the Web services. This is a difficult and time-consuming task. Therefore, the current interest is to have an automatic system that could investigate the Business logics and distribute the felicitous style to integrate them. There is no regular archetypal framework to carry out these concerns and so this Thesis proposes a framework which inspects the Business logics independently and recommends proper structure to merge them.

As service logics are loosely coupled, integration of two or more logics might affect many other parts of the services. Moreover, since services from different enterprises are extracted and collaborated, accessibility issues for the required services need to be solved. The present demand is to automate all these processes and develop complex logics dynamically. Hence, the research focuses on a novel dynamic schema-driven service integration mechanism using Turing Machines to automate the task of service integration. Finite State Machine (FSM) is used to analyze the service business logic in terms of business rules, functions and parameters and also to evaluate the properties and dependency relation existing in the business logic.

The proposed advanced model is Policy Evaluation System which examines the service logics and discovers the policies while integrating professional policy assessment of the services and which also examines the Policy Detection Point (PDP) and Policy Enforcing Point (PEP) to deal more specifically with the policy. The proposed Integration model also uses ontology for domain specific knowledge to deal with policy, policy description and policy priority. This model integrates the required Web services without break through Service Level Agreement (SLA). In case of policy violation the reputation of the service is measured through (RM) Reputation Measurement by considering malicious intentions.

The proposed service integration system accomplishes this task through disparate agents such as Service Discovery Agent (SDA), SLA Negotiation (SN) Agent, Dependency Analyzer (DA) Agent, Property Evaluation (PE) Agent and Service Integration (SI) Agent. Hence, the collaborative service integration environment automates the task of service integration through dependency analysis and property evaluation using agents.

The developed algorithms for policy detection and violation detection are evaluated and comparison is made to differentiate the performance of Turing Machine and the Alternating Turing machines with the number of policies detected and violated. The performance of the integrating services is evaluated in terms of Reputation scores, computational time and accuracy of Reputation calculation. The results show that the overall computational time required for the business policy violation detection, Reputation Measurement and the integration of service which has better reputation is 5.087 seconds and the accuracy in calculating the Reputation score is an average of 86.75% which is a very high achievement when compared to service integration without Reputation Measurement. Hence, the proposed system is well suited for automatic Web service integration for a Change Management Framework.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ACRONYMS	**ABBREVIATIONS**
AHP	Analytic Hierarchy Process
ATM	Alternating Turing Machine
BLIS	Business Logic Integration Schema
BLPS	Business Logic Property Schema
BPEL	Business Process Execution Language
CMF	Change Management Framework
CMP	Change Management Practices
eOCS	Extended Online Charging Systems
EPC	Evolution Packet Core
FSM	Finite State Machine
KMC	K-Median Clustering
LCS	Long Term Composed Services
LIDS	Linked Data Services
LTE	Long Term Evolution
OBIEE	Oracle Business Intelligence Enterprise Edition
ONLI	Ontology Based Natural Language Interface
PCC	Policy and Charging Control
PCRF	Policy Control Rules Function
PDP	Policy Detection Point
PEP	Policy Enforcement Point
QoS	Quality of Services
QoSMOS	QoS Management and Optimization of Service based system
RM	Reputation Measurement
SOAP	Simple Object Access Protocol
SPL	Software Product Line

TM	Turing Machine
UDDI	Universal Description Discovery and Integration
WSC	Web Service Composition
WSCML	Web Service Change Management Language
WSDL	Web Service Description Language
WSML	Web Service Modeling Language

CHAPTER 1
INTRODUCTION

1.1 Overview

Web Service Technology is booming technology that has gained interest for the past few years. Single Web service is not a substantial solution for any single user request in the current era. Service integration is anticipated to enable the endeavor to achieve coordination development by coordinating between various associations' applications among partners. The present market scenario does not fulfill the user needs with integrating resources or services as a complete solution. It involves coordinating Web service logics at useful level or business control for assorted essential suddenly. The chapter tries to give a concise presentation of Web services, the parts of Web services and the requirement for Web service coordination. Importance of Business Policy is also listed. The issues of the research work, research objectives and the contributions towards the research are also discussed in this chapter.

1.2 Motivation

Long Term Composed Service comprises of a few self-sufficient outsourced Web services that go about as an essentially sound single element. Business substances, as Web services, are frequently topographically conveyed and furthermore authoritatively free with each other. Before integrating the service to the business process, the policy of the newest request will be verified against the existing defined policies and if the policies are not matched at different levels, it will be thrown to the exception handler but the incoming Web service request may be of an efficient one coping with future facilities of the clients accessing the service from the particular Website. This may lead to the improvement of the business and hence the cross-checking of the request is necessary to make the system completely dynamic for the changing environment. Also while checking the quality of the service request the act of malicious intentions also need to be taken into consideration as this may lead to the integration of service that is completely not necessary for the business process. This has led to a research question of "how service composition could be atomized completely with Reputation Measurement of composing services without Policy

Violations and thus the focus of research has been specific to automate the whole system of Web service integration without policy violation and to measure the reputation of composing services.

1.3 Web Services

Numerous organizations utilize different programming frameworks to deal with their obligations. Programming frameworks that are diverse frequently need to trade their information with each other for different activities. A Web service can be considered as a strategy for correspondence that permits two distinctive programming frameworks on various stages to trade the information over the Internet. Assorted programming may be worked with the assistance of various programming dialects. Accordingly, there is a requirement for a standard technique for information trade that does not rely on a specific programming dialect. Relatively extraordinary kinds of programming can translate XML labels. This is depicted in Figure 1.1.Web services generally use XML files for data exchange. A Service Oriented Architecture (SOA) is fundamentally an accumulation of a few services set up together. These services speak with each other to trade their information to satisfy client needs. The correspondence for the most part includes straightforward information passing or it can likewise be at least two services organizing with each other and sharing their information to play out some action. Along these lines there is a requirement for methods for associating services with each other.

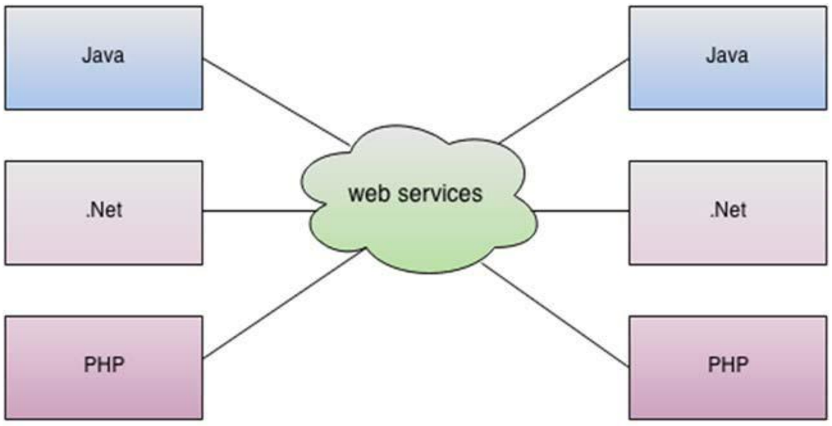


Figure 1.1: Structure of Web Service

Web services are self-describing and self-contained applications which are invoked from anywhere across a distributed system. They are modular applications which are published by various providers and located anywhere across the Web. A source of data as if they come from a single source. Also, the Software-as-a-Service (SaaS) worldview incorporates information in light of cloud that could without much of a stretch speak with the current frameworks to guarantee that clients can get to exact, finish, state-of-the-art information at whatever point and wherever they require it. Infrastructure-as-a-Service (IaaS) is more adaptable, versatile and reusable way to deal with combination where the core integration that performs incorporation conveyed from the Web as a service are functions, for example, semantic intercession, information movement, network, and other integration facilities.

A Web service which is composed of many Web services is always on demand. They are the coordinated effort of self-ruling Web services that give an esteem included administration powerfully. The autonomous Web services are specialized in their own domain and hence provide an increased quality of service with reduced cost to other business entities and their customers. The composed Web services can be categorized into (i) Short Term Composed Service and (ii) Long Term Composed Service.

1.3.1 Short Term Composed Services

The Business objectives and partnership between the Web services are temporary and for a limited time period for a Short Term Composed Service. The cooperation among the services is broken once the business objectives are accomplished. A Trip Planner could be said as an example for Short Term Composed Service. The planner could be a collaboration of Travel Service, Hotel Service and Taxi Service. Once the planning process is complete, the partnership among the services is dissolved.

1.3.2 Long Term Composed Services

Long Term Composed Services are services that are created by combining various services that are available online to provide a value added service to the customers. Many business entities are trying to advertise their functionalities now-a-days on the Web through Web services. LCSs have turned out to be prominent and have quickly embraced as another speculation for business related exercises. These LCSs encourage adaptable and on-request relationship between various business elements to trade their information. This basically centers on the wide application areas of LCSs that incorporates logical registering, tourism ventures, PC enterprises, and so on. There are several important benefits of LCSs that are summarized below:

- Utilization of Web services can extremely decrease the capital required to begin any business. Web services are promptly accessible substances that could be coordinated and organized. Therefore, ventures made by organizations individuals might be reused to a more prominent degree and procure more benefit. Provisioning of Web services additionally lessens an opportunity to showcase. Since new Web services can be created from the effectively existing administrations, the market intensity and notoriety will profit the LCSs that outsource them.
- Dynamic selection of partners of LCSs can be made that relates to individual domains. Web services provide APIs that are machine-processable and which could empower themselves to be summoned and arranged consequently as stated by J. Hendler (J. Hendler 2001). Thus, the LCSs arrange an on-request and venture driven organization together between various business elements attempting to speak with each other's information. Another preferred standpoint of LCS is that there is no land limit that could confine the choice of business accomplices. Along these lines, LCSs gives a wide-coordination of business elements from the “global village”.
- The several number of business elements have distributed their functionalities on the Web, it will be conceivable that numerous specialist co-ops rival each other to offer a similar sort of service with various client driven quality (Q. Yu 2008) "Best Service" be chosen from those suppliers to frame LCSs. In this manner, the end clients of LCSs will be profited from the open rivalry between organizations.

The lifetime of the long term composed service (usually known as LCS) is more when compared to Short Term Composed Service. They have the open ended life time. They have long term business goals and commitments. Until an external event occurs, the partnership among the components of the composed service remains stable. Many value-added services can be provided by these kinds of composed services for the cross organization collaboration. Such collaborative services are adaptive in nature in the dynamic environment. According to the frequent occurrences of changes, the adjustment has to be done.

Different stages are associated with the life cycle of LCS. It begins from beginning to its end. There are four stages in the lifecycle: planning, composition, orchestration, and dissolution. The main stage is the planning phase where an abstract level of description is done. This stage is activated when the proprietor of the LCS finds a requirement for another business goal to be accomplished (Reid et al. 1996). The composition stage is the stage that spotlights on coordinating the chosen Web services (Medjahed et al. 2003). After integration, the chosen Web services are organized to give the esteem included service as required by the clients. The dissolution stage happens when the proprietor of the LCS chooses that the orchestrated services have to no longer be integrated with each other and they need to be dissolved.

1.4 Change Management Framework

The advantage of LCSs can be fully realized only when they help to improve their compliance to the dynamic environment, i.e., when they are ready for the changes during their lifetime. Since the Web service infrastructure is dynamic in nature, changes should be treated as the rule and should be managed in a structured and systematic way (Khoshafian 2006). Changes are usually introduced when a new market interest arises and when there are new business regulations. There may also be a need to upgrade the technologies and the need could arise. A requirement is always associated with such changes to focus on the modification of a LCS. The change might respect the usefulness that it gives, the way it plays out its functions, the composition to which they are made out of, and the execution it conveys. Once the requirement for a change happens, the LCS needs to rapidly control itself to satisfy the prerequisite presented by the change. The modifications must be performed in a dynamic and a programmed way. A LCS can expand the market premiums it pulls in thusly. It can likewise streamline the way it outsources its usefulness. This prompts support of its aggressiveness among its companions. Changes in a LCS are for the most part categorized into top-down changes and bottom up changes. Top-down changes are those that are started by the proprietor of the LCS and Bottom-up changes are those that are started by the service providers.

Top-down changes

These changes are typically raised as the consequence of business approaches, business controls, or laws. The LCS outsources its usefulness from services in the travel domain to offer a total travel package can be taken for instance situation for top down changes. Contingent upon the market report, the LCS may require that the administration be extended with new business needs by including another nearby movement benefit that may incorporate cabin, directing and so on.

Bottom-up changes

These changes are raised as the aftereffect of a bug, an exception, or a shift in a Web service space. This is experienced when these services are being used. Web services now and then change their functionalities freely without the authorization of the LCS that uses their highlights. For instance, a taxi booking service may wind up inaccessible because of a system disappointment. Another case could be an aircraft reservation service supplier may change the usefulness of the service to its clients by including another activity for checking a flight status, or a movement service. Supplier may choose to expand the desirable charge of the service.

The Thesis focuses on the changes that are triggered by the owners of the services i.e., top-down changes. These changes could incorporate an arrangement of progress initiators, which can be business strategies, changes in business directions (or laws), and the LCS's proprietor necessities.

1.4.1 Service Integration

Service integration has been adapted as a major tool that allows the endeavor to reach its maturity to integrate between various organizations' applications among companions. The present market asks for as a rule is satisfied by incorporating solicitations or services as an element. There are three key challenges at this level of Integration. First, the Degree of coupling among them should be maintained whenever a business partner wants to keep a tie up or merging with others. Second, the need for business logics be positioned properly and should be maintained properly when integrating. Third, the recovered logics should be integrated strongly as interoperability issues are reached. It is indeed a difficult task for the developers to recognize the whole services and detect a best way for integration of the services. So it is required to build up an automated framework to incorporate the Web services progressively as changes emerge. The goal of the proposed work is to make the service integration process automated which enables organizations to easily exchange their service logic added dynamically and accurately with their profitable network partners by considering the service policies and Service Level Agreement (SLA).

1.4.2 Service Level Agreement (SLA)

A Service Level Agreement (SLA) is a kind of understanding that is made between a service supplier and the end-customer; this describes the level of service that the customer foresees from a service supplier. The motivation behind SLAs is yield based which implies their motivation is especially to describe what the customer will get. SLAs don't describe how the service itself is given or passed on. Figure 1.2 illustrates how the SLA of Internet Service Provider (ISP) will provide their customer through a basic example of the services that it will provide.

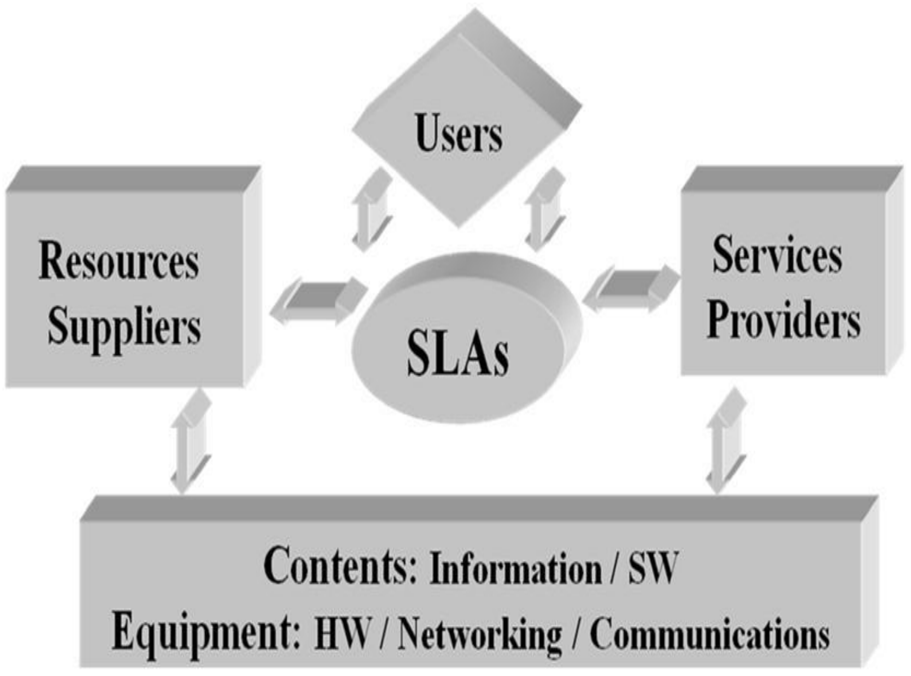

Users
Resources Suppliers
SLAs
Services Providers
Contents: Information / SW
Equipment: HW / Networking / Communications

Figure 1.2: SLA of Internet Service Provider (ISP)

1.4.3 Business Policy

Business Policy characterizes the extent of an association inside which the choices can be made by the individuals and the undertakings that can be done. It allows the lower level management to take decisions for certain problems without consulting the top level management all the time. They are the rules created by association to represent their exercises. They characterize the constraints of the choices that could be made and the goals that should be achieved by the organization. While developing services, the developers also frame the policy of the organization and the resource they hold. Once the policies are framed, care is taken in order to not violate the policies at any point of time.

1.4.4 Business Policy Features

Business policies framed by an enterprise should hold the following features:

- Policy ought to be particular and unequivocal. It ought to be sure and particular for an undertaking; if not the usage will wind up troublesome.
- Policy must be free from confusions. It ought to keep away from utilization of languages and meanings. There ought to be no false impressions in the policy.
- Policy must be sufficiently uniform for the subordinates to tail it.
- Policy ought to be proper for the satisfaction of the hierarchical objective.
- Policy ought to be straightforward and effectively comprehended by all in the association
- Policy ought to be far reaching to make utilization of it in a wide range.
- Policy ought to be adaptable in activity/application.
- Policy ought to be steady so no vulnerabilities happen while settling on choices with its rules.

Figure 1.3 illustrates the life cycle of a business policy framed for an organization. The policy development is first initiated to develop new advancement of organizations. Then they are properly developed using any of the policy specification language and then they are sent for the approval and communicated with people in the concern. The implementation phase is then done and the policies are deployed to make decisions. Reviewer reviews the policies periodically to monitor the alterations done and that the organization runs based on the framed policy. The proposed system tries to detect these policies while integrating various services and checks that there are no policy violations being made while integrating the services provided by various service providers.

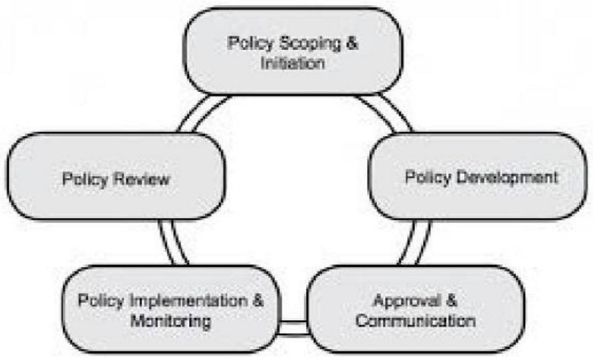

Policy Scoping & Initiation
Policy Review
Policy Development
Policy Implementation & Monitoring
Approval & Communication

Figure 1.3: Life Cycle of a Business Policy

1.4.5 Challenges in Change Management Framework

The genuine research issue in Change Management in LCSs is not quite the same as the ones in work process frameworks. This is a direct result of the distinction in settings and suppositions that are discussed:

- LCSs need to guarantee the empowering of adaptable cooperation between various business elements. They are fit for adjusting to the dynamic condition in a fitting way. The change of the cooperation is relied upon to be visit by this nature and needs a methodical help. Though, the association between various substances of work process frameworks is generally inside an association. The condition that the frameworks cooperate with is moderately "static". Changes on the portrayal are dealt with as "exemptions" and for the most part performed physically.
- LCSs for the most part contract out their functionalities from free service providers. No focal control component is available that could be utilized by a LCS to screen and oversee changes. With regards to substances in work process frameworks, they are inside an association. This empowers a brought together way to deal with screen and deal with the changes.

1.5 Malicious / Fake Services

Web services give cross-stage and language independent access to information and applications over the Internet. Web services follow some significant standards such as SOAP, UDDI, and WSDL. With the assistance of these standards, the Web services can progressively connect with each other. In this manner, they can execute complex business shapes. Such interoperability drives a creating example to make complex and esteem included Web services by fusing the present ones. Two issues must be steered to enable Web service combination in the midst of a couple of free Web service applications and service providers. A versatile Web service disclosure methodology is basic for customers to pick the Web services that can ensure their necessities and Quality of-service (QoS) essentials. Furthermore, an appropriate care to perceive that the services being incorporated isn't a phony one. Thus notoriety service component is important to separate service providers in view of their notoriety in giving reliable Web service.

1.5.1 Malicious Behaviour

Among quite a lot of threats that can occur in Web service integration; the main kind of danger is the least complex type of risk which is the Malicious Services. These threats are frequently seen inside Web service integration situations. Noxious service providers consistently advance that they would give services that match customers' favorable position. They get distinctive services into the integration structure while genuine solicitations can't be reacted. The secured services may not be unsafe, but rather at times, they could infuse viruses or Trojan horses that are equipped for overwhelming basic harm to the end client.

1.5.2 Distributing Dishonest Opinions

As the service show allows its clients to contribute their transaction experiences with a service provider and create proposals, some malicious clients may exploit this trademark. They give the wrong impression about less educated looks into think about false data about different providers. Moreover, it might lead clients to pick less mindful providers, perhaps malicious ones, and get undesirable services which might be injurious. The other variety of such misconduct is boosting inconsistent providers' reputation unreasonably, potentially as a piece of conspiracy, to trap purchasers into unquestioning deceitful providers. For this situation, it is conceivable that purchasers confront brutal issues in utilizing such fake services.

1.6 Finite State Machine

Finite State machines are crude however more intense and helpful computational model for equipment and certain sorts of programming. Finite state machines can also be called as Finite State Automata.

1.6.1 Structure of FSM

A well known form of Finite State Machine is the Turing Machine. Turing machine consists of state controller with a versatile read/compose head and an unbounded stockpiling tape. The input for a Turing Machine is a set of symbols and those symbols that are written by the tape could be the output as shown in Figure 1.4. There are several applications of Turing machines which include simple form of pattern matching and for modeling several sequential logic circuits and many device controllers.

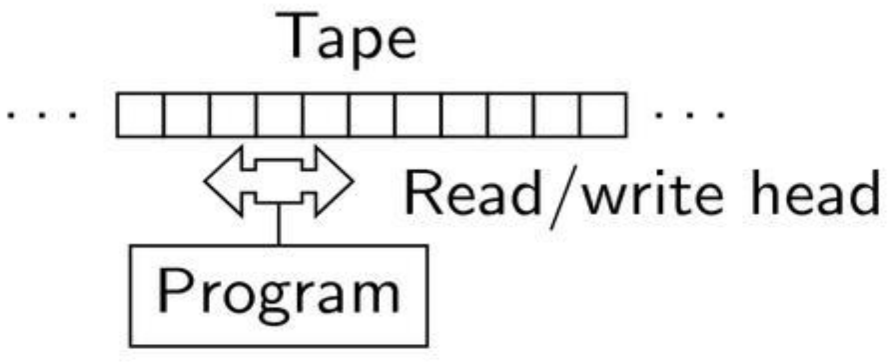


Figure 1.4: Finite Control of a Turing Machine

1.6.2 Working of Turing Machines

Turing machines can be simulated in variety of types based on the tapes in the Turing machine and the heads in it. By these varieties they are classified as follows:

a. One dimensional tape TM
b. Two dimensional Tape TM
c. Multi Tape TM
d. Multi Head TM

For the proposed work, a Multi-tape and multi-head Turing machines are simulated. Multi-tape Turing Machines have multiple tapes in which each tape is accessed with a separate tape head. Each tape head is capable of moving independently of the other heads. The input symbols are at tape 1 initially and the other tapes are kept blank. The machine then reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads further as depicted in Figure 1.5.

A Multi-tape Turing machine can be formally described as a 6-tuple (Q, X, B, ä, q_0, F) where,

- **Q** – ser of finite set states
- **X** – alphabet tape
- **B -** blank symbol
- **ä** - relation between states and symbols where

$\delta: Q \times X^k \rightarrow Q \times (X \times \{Left\_shift, Right\_shift, No\_shift\})^k$ Where there is **k** number of tapes.

- **q_0** - initial state
- **F** - set of final states

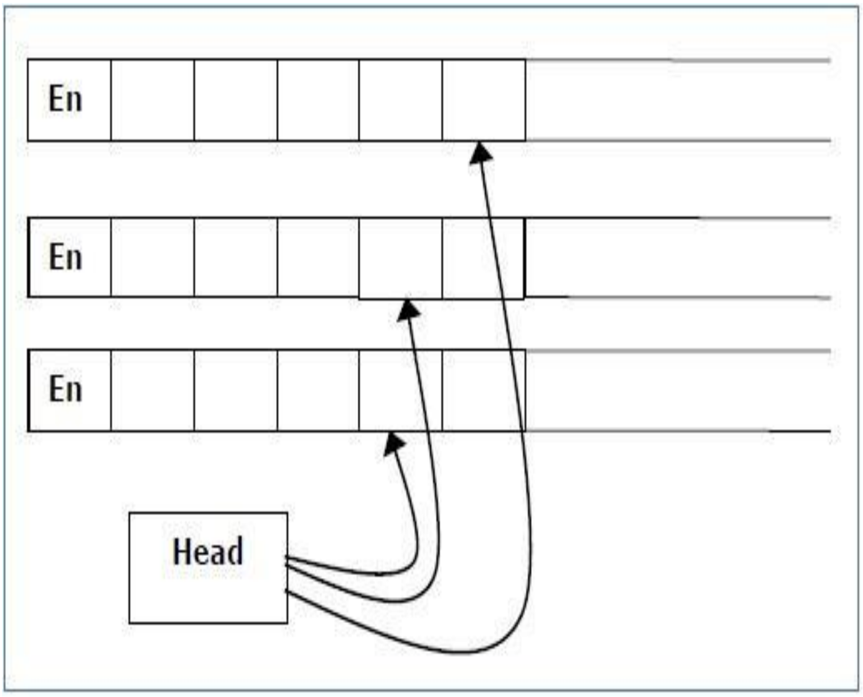


Figure1.5: Finite Control of a multi- tape Turing Machine

For the research work, a multi-tape Turing Machine has been constructed that could detect the business policies of different services framed by different service providers and to detect the policy violations while integrating the services.

1.6.3 Applications of Turing Machines

A large number of practical applications are available for Turing Machines. Some important applications are as follows:

Check Decidability

On the off chance that a Turing Machine can't take care of an issue in countable time, at that point it can be reasoned that it can't be solved utilizing any algorithm. The issue is so not decidable. If there should arise an occurrence of a choice issue if the TM ends in countable time for all finite length inputs, at that point the problem is solvable in countable time.

Classify Problem

TM helps in the grouping of decidable problem into classes of Polynomial Hierarchy. In the event that it is discovered that the problem is decidable, at that point the objective turns into the subject of how effectively we can comprehend it. The productivity can be ascertained in number of steps, additional space and length of the code/size of the FSM.

Design and Implement Algorithm for Practical Machines

TM transmits the possibility of an algorithm in other constant machines. After the effective finish of issue characterization and decidability, the pragmatic gadgets and PCs can be utilized to outline and execute an algorithm.

1.6.4 Challenges in Construction of Turing Machine

All mathematical yet not just mathematical languages have restricted expressiveness. This data achieves a few essential choices. For any language, there dependably exist certain problems that can't be spoken to utilizing it. The Second actuality is that there dependably exist problems identifying with inquiries regarding the matter of these problems can be planned in a language however the language is excessively feeble, making it impossible to express the coveted answer. Indeed, even the exactness of the appropriate response that is gotten is deficient for the few viable needs. Next is at any given moment of time for each current problem there exist a finite number of languages that can be utilized to assault the problem. Finally, the most effective language among them characterizes computational breaking points for the problem that exist both for physical and nonexistent Turing machines. In both the cases a maximal finite or a maximal infinite number of emphases are considered.

1.7 Scope of the research

Several challenges arise when trying to compose the long term composed services. The system extracts the Policies from the already existing Long Term Composed Services (LCSs). These services may have different policies at a different level of abstraction by constructing two multi-tape Turing Machines. Then the system tries to detect the Policy Violations that occur in the services of the two LCSs at various levels such as Rule level, Function level, Parameter level and Dependency level using the Multi-Tape Turing Machines. A state transition table is drawn to note the violated policies. When a change request arises, policy violation is detected and the measurement of the reputation of services through RM is done by considering malicious intentions. This phase is operated only if a policy violation is detected. Reputation measurement is done for Ballot Stuff Attack and Sybil attack.

To Handle Multiple Change Request dynamically and to completely automate the system a multi-agent environment is created. OWL Methodologies are adopted to handle-domain specific change request emerging from the users of such LCSs.

1.8 Organisation of the Thesis

The objective of this thesis is to develop a collaborative framework for dynamic service integration by policy violation detection of integrating services without fake services and to automate the whole system by using Finite State Machine. The rest of the chapters are organized as follows:

Chapter 2 presents the literature survey regarding the need for Web Service Integration and the tools and techniques for Web services integration. It also gives a brief overview of Change Management Framework. This chapter also summarizes the Business policies that exist and the need to consider Business Policies while integrating the Web services.

Chapter 3 presents an overall idea of the proposed work. Besides the research objectives, challenges in developing the system are also briefed. This chapter also discusses the Experimental Setup to develop the proposed system and the evaluation metrics used to evaluate the performance of the proposed work.

Chapter 4 deals with the construction of Turing Machines. The advantage of using Turing machines for Web service integration is summarized in this chapter. The results proved that policies detected via Turing machines are more accurate and less prone to error.

Chapter 5 briefly discusses the construction of Alternating Turing Machine to handle multiple change requests coming into the system. ATM is employed to detect the policy points with acceptance and rejection status and then the policy detection points are obtained in terms of Business, Function, and Parameter. Domain specific knowledge is also given to the system using Ontology. A comparison is also made to show that the accuracy of Web service integration is more powerful when ATM is employed.

The Reputation Measurement System is introduced in chapter 6 to calculate the malicious and fake services advertised by some service providers. The system works by considering the malicious intentions such as ballot-stuffing attack and fake identities which are intentionally performed to credit the service by giving inaccurate credit labels. The results show that the computational time increases if the Reputation Measurement is done prior to Web service integration.

In chapter 7, a collaborative framework for dynamic Web service integration is introduced. The main functionality of this framework is to perform Dependency analysis between the integrating services. A new Turing machine is constructed to perform pattern matching of the patterns that are extracted from the integrating services. Similarity measure is likewise furnished alongside Output Generative Power (OGP). Similarity measure gives the quantity of administration requests dealt with properly when integration with no exceptions.

A case study of Travel Agency LCS (Composite Web Service) is made in Chapter 8 to prove the efficiency of the research. A Travel LCS is composed with other LCSs, namely, the Hotel service and the Vehicle Service. Results prove that the integrated services are free from business policy violations.

Chapter 9 concludes the thesis by highlighting the findings that assisted to accomplish the objectives. A summary of research contribution and the scope for future improvement are also incorporated in this chapter.

CHAPTER 2
LITERATURE REVIEW

2.1 Overview

An exhaustive literature has been collected relating to the proposed work and the literature has been thoroughly evaluated and reviewed critically. The review includes the challenges in Change Management Framework. It has been identified that whenever a need for change management occurs, a thorough care for policies framed by different service providers should be given. Proper study has been made for the methodologies adopted to understand the business policies and the techniques for identifying the business policy violations and the need for dynamic Web service integration. The research challenges from the extract of the literature review have been presented at the end of this chapter.

2.2 Importance of Business Policies in Web Service Integration

Whenever there is a need for change in the already existing Web services, there arises a need to integrate Web services that could be given as a value-added service to the customers. Every service provider and the services provided by them have its own business policies framed during development. The term 'business policy' refers to the process and procedures of an organization. It is important for a business owner to carefully deal with the business policies which forms the specific methods for running the day-to-day activities of an organization. Policies are defined with care and policies are specific for a specific business. When Web services are considered, business policies should be dealt very carefully. These business policies are framed in the Services Level Agreement (SLA). A clear study is made to understand the importance of business policies and the effects of business policy volitions in Web service integration.

It is essential for service providers to keep track of the SLA violations to be detected properly to enhance the business process and customer satisfaction. Philipp Leitner (Philipp Leitner et al. 2010) proposed an approach for predicting the Service Level Agreement (SLA) at runtime. The measured facts i.e., the QoS parameters and their values are taken as inputs for the proposed system. The model also uses a machine-learning regression technique for SLA violation detection.

Service integration and composition system should be able to detect even the Web service failures and performance degradations due to SLA violations. A solution for dynamic policy violation was proposed by Adina (Adina Mosicat et al. 2011). A runtime monitoring system for degradation detection, diagnostics and repair of service composition was developed. The solution uses a lightweight monitoring technique that is compatible with any standard.

QoS parameters are usually included in the Service Level Agreements of each service. These parameters may also be useful for detecting the policy violations of the composing services. A general monitoring system and reference model for analysis was proposed by C. Muller (C. Muller et al. 2012) that could notify the clients with the violations and the causes for violations in an easy-to-understand manner. Li (Li et al. 2012) had exhibited a dynamic policy and charging control (PCC) system. This structure controls and upgrade network utilization and to give network data measure assets to their supporter. It additionally works as per the client's continuous profile and membership data of the client and to stay away from or limit administrator's network congestion. It likewise helps in optimizing the administrator's current and potential network speculation. The creators predicated the key estimation of streamlining that depended on the concurrent skill of the rating and charging capacities on knowledge benefit streams and the endorser account profile and lifecycle.

SLA compliance is most important for the providers because the violation of the policies in those SLA could lead to customer dissatisfaction. Therefore a system that could predict the violations in prior was proposed by Philippe (Philippe Leitner et al. 2013). This system allows the operators to take timely remedial actions. The system is also designed to detect violations before they occur. Instance-level prediction and forecasting are the two techniques proposed by the authors. An incremental SLA violation recognition framework with time affect investigation was proposed by Azlan Ismail (Azlan Ismail et al. 2013) to automatically produce an effect locale in light of negative time affect condition. In light of the assessments made by the analysts, the approach could diminish the measure of service change inside significant recuperation execution time.

A Petri Net-Based Model proposed by Yanhua Du (Yanhua Du et al. 2014) addressed the issue of detection of temporal violations in a holistic manner. The proposed work not only compares the Web services by adding a prediction net to predict message mismatch but also checks the compatibility of the composing services using a modular-timed state graph.

The traditional model of service composition tries to improve customer satisfaction by improving the QoS parameters. But mere improving the QoS parameters alone does not lead to an improved service composition. Therefore an On-Demand strategy for quality improvement is proposed by Quanwang Wu (Quanwang Wu et al. 2014). A service broker is introduced in the strategy. The third party broker collects the services from the service providers and follows the SLA of each service and picks up the best services for composition.

A SLA of an association incorporates an arrangement of terms that guarantee to ensure that it must satisfy amid the provisioning of administration to the clients and keeping in mind that utilization of the services. The violation of such certifications will prompt use of potential punishments. Therefore it is important to give surety that the Sevice Level Agreement meets its expectations. Marcos Palacios (Marcos Palacios et al. 2015) have proposed a practical approach to test SLA-aware SBAs. This approach identified the test requirements that represented the situations that are most appropriate to be tested. To address this issue, the authors have defined an evaluation technique that is a four values logic which evaluated both the individual guarantee terms and the logical relationships between them.

Table 2.1 represents the detailed summary of the various research works addressing business policies and the importance of business policy violation detection. Different aspects have been addressed by the researches as shown in the Table.

From Table 2.1, it is evident that research contributions have described various techniques to identify the Business Policies and to further keep track of the business policy violations for proper Web service composition and integration. Further the literature has been narrowed to identify the various aspects to be considered to meet the need for a change in business process and to integrate the Web services without policy violations.

Table 2.1: Contributions towards SLA violation (Business policy violation)

Researchers	Contributions	Addressed Aspects
Philipp Leitner et al. 2010.	The model uses a machine learning regression technique for SLA violation detection.	Business Policy Detection, Dynamic Integration.
Adina Mosicat et al. 2011.	A runtime monitoring system for degradation detection, diagnostics and repair of service composition was developed.	Service Composition, Policy Violation Monitoring.
C.Muller et al. 2012.	Developed a system that notifies the clients with the violations and the causes for violations in an easy to understand manner.	Policy Detection, Violation Detection.
Li et al. 2012.	A dynamic policy control and charging control (PCC) system that controls and upgrade organize utilization. It additionally gave arrange data measure belonging to supporters.	Dynamic Policy Violation Detection, Dynamic Integration.
Philippe Leitner et al. 2013.	This system allows the operators to take timely remedial actions when a policy violation is detected.	Policy Detection, Completely Automated.
Azlan Ismail et al. 2013.	Based on the evaluations made by the researchers, the approach could reduce the amount of service change within considerable recovery execution time.	Static Policy Violation Detection.
Yanhua Du et al. 2014.	Looks at the Web services by adding an expectation net to anticipate message confound and furthermore checks the similarity of the composing services utilizing a measured planned state chart.	Service composition, Static Policy Detection.
Quanwang Wu et al. 2014.	A service broker is introduced to check the compatibility of SLA of each service and pick up the best services for composition.	Completely Automated, Business Policy Detection, Dynamic Integration.
Marcos Palacios et al. 2015.	Four-esteemed logic that permits assessing both the individual assurance terms and their logical connections.	Business Logic, Dynamic Composition.

2.3 Methodologies for Long Term Web Service Composition

Semantic Web and Web Services are the developing advancements for the future age Web applications. Ontology is characterized as the building squares of numerous Web applications. The Ontology Web Language for Services (OWL-S) is a most helpful arrangement of mark-up language that is built for depicting the properties and capacities of Web services in a reasonable and PC interpretable way. OWL-S depends on Ontologies of items and the concepts identified with a domain that is characterized utilizing the Ontology Web Language (OWL). The main objective of Web service is to provide a distributed computation by automatic discovery, composition and execution of service. But these technologies do not include explicit description of the functionalities about the domain. Hence ontology provides a better solution for this problem.

The retrieval of specific Web service can be done by Web Service Modeling Ontology (WSMO) where the ontologies are formed to organize the services so that Web service matching is performed through the reasoning approach Pierluigi (Pierluigi et al. 2009). Keeping in mind the end goal to join information from various APIs and information sources, we require a paste code that is regularly not shared to all and can't be reused. (Sebastian Speiser et al. 2011) have proposed Linked Data Services (LIDS), a general, formalized way to deal with coordinate information furnishing services with Linked Data. This is exceptionally a famous system for information distributing that encourages information integration and permits decentralized information distributing. The creators have introduced interchanges for benefit get to interfaces that work on Linked Data standards, and have proposed theoretical paltry administration portrayal framework. Creators have created calculations that make utilization of LIDS portrayals to make interfaces automatically amongst services and existing informational indexes. Jun (Jun Shen et al. 2010) developed an algorithm to perform a fine grained match of the service profiles at all levels of service composition.

An ontology-based framework proposed by Kehagias (Kehagias et al. 2010) is a system which works to collect content from various Web services about their functionalities. It is a service alignment tool which enables the service providers to map their services according to the present domain ontology. Ontologies can also be a helpful tool to judge the overall composition of Web services. Freddy (Freddy et al.

2011) showed specific interest in combining semantic and non-functional criteria such as QoS to evaluate the composed services. The system measures the semantic quality with a QoS metric to rank and optimize the Web service composition.

An important component of Web service composition is discovery of relevant services the user requires. A system proposed by Paulraj (Paulraj et al. 2012) uses an algorithm to perform a fine grained match of the service profiles at all levels of service composition. It is a framework for business processes integration, which can be applied to Web services defined in OWL-S. Information contained in the service profile is sufficient for composite semantic Web services discovery. The authors have also proposed a solution for composite semantic Web services composition. An integrated system called PORSCE II was proposed by Ouransa (Ouransa et al. 2013) which performs automatic Web service composition with the help of AI techniques. The system provides an improved composite service by obtaining the semantic description obtained by a domain-independent planning system.

To advance the process of integration, Somluck (Somluck L-Ongsri et al. 2015) proposed that it is important to improve the quality of demonstrating formalisms and to ensure that these varieties of extra communicative formalisms continue to be semantically complete. The authors state that theoretical demonstrating procedures would work semantically better-off if with the help of intelligence to prompt the semantics of a requester on concrete presentation scenarios. Here the author have examined that the incorporation of ontologies into theoretical demonstrating methodologies would work better by giving the OntoER model, OntoORM and OntoUML class diagram.

The service oriented architecture is an independent and standardized self describing unit made of several services. A new architecture was proposed by Son (Son et al. 2014) for building an automation system adapting SOA paradigm with device profiles for the Web services. In this system, context information is collected, processed and sent to the composition engine to coordinate with appropriate devices and its adaptable services in order to give a more value- added service based on the context, composition plan and pre-defined policy rules.

Table 2.2 represents the detailed summary about the various research works carried out by the researchers to address the issue of Web service integration with domain knowledge.

Table 2.2: Contributions towards Methodology for Web service composition

Researchers	Contributions	Addressed Aspects
Pierluigi et al. 2009.	Web Service Modeling Ontology (WSMO) where the ontologies are formed to organize the services.	Domain Knowledge, Service Integration.
Kehagias et al. 2010.	Service alignment tool which enables the service providers to map their services according to the present domain ontology	Ontology, Service Alignment.
Freddy et al. 2011.	Semantic and non-functional criteria such as QoS to evaluate the composed services.	Service Composition, Semantic Knowledge.
Sebastian Speiser et al. 2011.	Formalized approach for integrating data-providing services with Linked Data.	Service Integration, Linked Data Knowledge.
Jun Shen et al. 2010.	Algorithm to perform a fine grained match of the service profiles at all levels of service composition.	Service Composition, Dynamic Integration.
Paulraj et al. 2012.	Proposed a business process integration framework which is applied to Web services that are defined in OWL-S.	Web Ontology, Dynamic Integration.
Ouransa et al. 2013.	Automatic Web service composition by AI techniques.	Completely Automated, Business Policy Detection, Dynamic Integration.
Somluck L-Ongsri et al. 2015.	The procedures of consolidating ontologies into hypothetical approachs, giving the OntoER model, OntoORM and OntoUML class chart.	Domain Knowledge, OWL Methodologies.
Son et al. 2014.	Automation system adapting SOA paradigm with device profiles for the Web services.	Partially automated service Integration.

Table 2.2 summarizes the different aspects in which the ontology is used for Web service composition. Several research contributions have been made to enhance the degree of Web service integration with domain knowledge. Further the literature has been narrowed to identify the various aspects to be considered to meet the need for a change in business process and to enhance the process of integration in an effective manner.

2.4 Approaches to Change Management Framework

Change Management is the caretaker of one’s controlled environment. It is stimulated to protect everything that could impact services. An optimized change management process usually results in a fewer problems and also does a huge improvement on processing the responses to requests for services. In today’s competitive environment, it is very important to easily and appropriately handle the changes. The balance between flexibility and stability should always be maintained by the business people. Therefore, it is necessary for the IT organizations to make use of the best practices for the entire, end-to-end, change management life cycle. The IT organizations that hold the closely controlled approach to Change Management will only be able to move their operational alertness essential for service fineness.

“A novel engineering for Change Management Framework” proposed by Wen-Hsiung Wu (Wu et al. 2012) supported Configuration Management II (CMII) standards and industrial applications. The framework considered the accumulation of full-track or quick track procedures to fathom the basic item common sense or quality issues among a particular exchange. The adequacy of the change management framework proposed was through the utilization of the framework for a situation investigation of a genuine Taiwan motorcycle maker. At long last they displayed an execution assessment plot in view of the consequences of that framework and furthermore talked about the cost of its social control tips and suggestions.

Matthias Weske (Matthias Weske et al. 2012) remarked that all communities associated with the business need a typical comprehension of the disparate parts of business process management. A complete business process life cycle is detailed in this work. It starts from the modeling phase to the process performance phase. The significance of change and considering every single diverse partner included is additionally included here. Beginning with an introduction of general establishments and deliberation models, the work clarifies the ideas like process orchestrations and movements. The process properties and information conditions were also addressed here. As a final point, traditional and advanced business process management architecture was dealt that covers the imperative angles, for example, work process management frameworks, benefit situated designs, and information driven methodologies. The standards like SOAP, WSDL and BEPL are also brought into picture to fit the proposed framework.

A framework for effectively managing changes in Long-term Composed Services (LCSs) was presented by Xumin Liu (Xumin Liu et al. 2010). As it is normal that LCSs outsource their handiness from self-overseeing other Web service providers, there is a need to pick the best Web service substitutions when changes are provoked at the LCS level. Along these lines a framework has been proposed where managing changes in LCSs is modeled as a twofold service question change process. In the principle organize; reputation is used as a trust framework. In the second stage, the important necessities are addressed furthermore confine the game plan of reputable Web services.

A framework for change management was proposed by Salman Akram (Salman Akram et al. 2010) which maintained taxonomy for change. A bottom-up approach for change management with a combination of ordinary petri nets and reconfigurable petri nets were developed. To address the change management issues in Long Term Composed Services(LCSs), a system called Evolution of long term composed services (Ev-LCS) was proposed by Xumin Liu (Xumin Liu et al. 2013) which tended to a dynamic cooperation between independent Web services that by large give an esteem included service. A prototype system was simulated to demonstrate the effectiveness of the system.

A Web Service Change Management Language (SCML) to manage the top-down changes in LCSs was developed by Xumin Liu (Xumin Liu et al. 2014). In light of the scientific categorization, an arrangement of progress administrators was characterized that could determine distinctive sorts of changes formally. Aminesh (Aminesh Chathurvedi 2014) performed a regression test for the services that are composed for Long term Composition. An automated system was developed to manage the changes that arise in a dynamic environment. Trust is an important property in the process of composition of various services. A Trust Analysis model was developed by Shanshan Qui (Shanshan Qui et al. 2012) to analyze the trust of operations performed on composite services. Hayes (J. Hayes et al. 2014) in his research gave a clear idea of the theory and the practices to be followed in handling change management process. His complete study gives the tools that can automate the whole system.

Programming frameworks are turning into a vital module of businesses in the aggressive world today. The change usually arise when a development of new features to be incorporated in the existing software. A Software Requirement Change Management framework proposed by Arif Ali Khan (Arif Ali Khan et al. 2012) gives a clear picture of the change management done for geographically distributed software systems. Table 2.1 represents the detailed summary about the various research works addressing change management process.

A mindful versatile Web service arrangement system is introduced by Zhiying (Zhiying Cao et al. 2015). In this system, BPEL was utilized as an instrument to portray Web service arrangement process. Setting services were characterized to support setting mindfulness Agent innovation was taken as a helping apparatus to empower setting service to effectively see and process settings of Web service and Web service creation for an entire change administration.

Table 2.3 depicts the contributions of researchers on the process of Change Management Framework and the mechanisms adapted for managing such change. The Table also describes the different aspects that are addressed by the researchers.

Table 2.3: Contributions towards Change Management process

Researchers	Contributions	Addressed Aspects
Wu et al. 2012.	This structure considered the collection of full-track or quick track procedures to unravel the basic item common sense or quality issues in a particular exchange.	Structured Change Management.
Matthias Weske et al. 2012.	A complete business process life cycle starting from modeling phase to process enactment phase is detailed in this work.	Completely Automated system, Business Process.
Xumin Liu et al. 2010.	A system has been proposed where overseeing changes in	Domain Knowledge, Dynamic Integration.

	LCSs is demonstrated as a double service query enhancement process.	
Salman Akram et al. 2010.	A bottom-up approach for change management with a combination of ordinary petrinets and reconfigurable petrinets.	Service Integration on errors and Exception.
Xumin Liu et al. 2013.	Dynamic coordinated effort was started between self-ruling Web services that could on the whole give an esteem added service to its clients.	Dynamic Service Integration, Completely Automated.
Xumin Liu et al. 2014.	A Web Service Change Management Language (SCML) was proposed to deal with the top-down changes in LCSs.	Dynamic Service Integration, Top-Down approach.
Aminesh Chathurvedi et al. 2014.	Gives an insight on the computation of changes and an algorithm for construction of reduced regression test suite for AWSCM.	Static Service Integration.
Shanshan Qui et al. 2012	A Trust Impact Analysis display is proposed to investigate the crash of two sorts of improvement activity performed on composite services.	Trust Model, Service Composition, Composite Services.
J.Hayes et al. 2014.	Gives the need and techniques to address change management process. Also talks about the reflection of change management process in organizational practices.	Top-Down Service Integration.
Arif Ali Khan et al. 2012.	A Framework is proposed for programming prerequisite change administration (RCM) in geologically circulated programming frameworks. It depicts the correspondence dangers required amid RMC and Global Software Development (GSD).	Completely Automated, Domain Knowledge.
Zhiying Cao et al. 2015.	BPEL is utilized to clarify Web service piece strategy and a sort of extraordinary Web service called the setting service.	Web service Integration.

From Table 2.3, it is clear that research contributions have described various frameworks for change management in Long Term Composed Service composition. Further, the literature has been narrowed to identify the various aspects to be considered to meet the need for a change in business process.

2.5 Identification of Malicious Services

Web services that develop different associations and figuring stages are generally made to make new service-situated applications all the more intensely. In any case, some Web services may act vindictively. The crucial necessity is to guarantee a productive system in prescribing solid services for clients. The suggestions of Web service are useful for clients when at least two Web services have the comparable activity however unique speed, throughput performance, etc. Hence, the different Web service with similar functionality can be rated by using user feedbacks. The reliability of the cumulative result of the feedback for commercial benefit can be degraded by malicious feedback ratings. Malicious users can produce a numerous malicious feedback ratings to threaten the reputation system of Web services.

Distributed computing has taken a new paradigm with the advent of Web services and Web service composition. Web services are used in a wide range of business integration process. In today's business competition an adequate security should be provided to the Web service for Web services composition. A complete survey has been made by Sumitha.T (Sumitha.T et al. 2011). The research focuses on some important vulnerabilities and attacks on Web services and Web service composition. Reputation is a method of trust mechanism followed for Web services to measure its trust worthiness. This is usually calculated by using the feedback rating by the users. A novel reputation measurement method is proposed by Wang (Wang et al. 2011) that employs two phases in the process of reputation measurement. In the first phase, the malicious ratings are detected by cumulative sum method. In the second phase, Pearson correlation coefficient method is used for rating adjustment. Experimental results show the efficiency of the system.

Selection of useful and accurate Web services for composition is a bottleneck for every researcher in the field of Web service integration and composition. The Web services can be selected based on the peer experience of the user. A system developed by Zakiria (Zakiria et al. 2012) uses three components that could aid proper discovery of accurate service. A composer, executer and monitor help in finding the relation between user and the Web services and execute the process of composition. A prototype was developed to show the efficiency of the system.

QoS value prediction is an important research area for Web service composition. Selection of proper services and trustworthy services are always needed. Weiwei (Weiwei et al. 2013) proposed QoS-mindful esteem forecast approach that ascertains the notoriety of the client in view of their commitment esteem. Then the process of rating is done to find the untrusted services. The work proposed by Szu-Yin (Szu-Yin et al. 2014) is a dependable two stage Web service revelation component in view of QoS and community oriented separating which finds the most appropriate Web service. The system also solves the problem of incorrect QoS information by identifying the QoS parameter provided by the Web services.

User preferences are the key factors for finding the top k data services that could be composed to form a value added services. A new user-preference model using fuzzy sets was developed by Karun Benouaret (Karun Benouaret et al. 2014). The system matches the fuzzy constraints for the relevant services to those of the user query. The matching degree of the service set is then ranked based on fuzzification of parent domain name method. The authors have also introduced a new method for increasing the diversity of returned top K compositions with high score. The profit organizations which are using the service platform for their business growth will gather information about the users from the social Websites but there exist two risks as the privacy of the users and the fake identities Katharina (Katharina Krombholz et al. 2014).

Another versatile separating method was proposed by Ing-Ray (Ing-Ray Chen et al. 2015) to decide the most ideal approach to consolidate coordinate trust and circuitous trust progressively. This limited joining time and trust estimation inclination within the sight of noxious hubs that perform artful service and agreement assaults. For adaptability, the creators exhibited a plan that utilized a limit restricted hub that keeps trust data of a subset of hubs that are of intrigue and the framework performs least calculation to refresh trust.

Table 2.4 summarizes the different contributions of the researchers on the identification of Fake identities given by the services to decrease the functionality of the long-term composition that is required for a change request.

Table 2.4 Contributions towards Identification of Malicious Services

Researchers	Contributions	Addressed Aspects
Sumitha. T et al. 2011.	Spotlights on some imperative mischievous activities and assaults on Web services amid Web service organization.	Reputation mechanism
Wang et al. 2011.	A novel reputation measurement method with two techniques; cumulative sum method and Pearson correlation coefficient method.	Completely Automated, Domain Knowledge, Malicious Services
Zakiria et al. 2012.	A composer, executer and monitor help in finding the relation between user and the Web services.	Web service Integration
Weiwei et al. 2013.	A new QoS aware approach of value prediction that calculates the reputation of the user based on their contribution value.	QoS prediction, Web Service Integration
Szu-Yin et al. 2014.	A two stage Trustworthy Web service disclosure component that depends on QoS and communitarian sifting procedure to find the most reasonable Web service.	Completely Automated, Domain Knowledge, Dynamic Integration
Karun Benouaret et al. 2014	A new user preference model using fuzzy sets.	Fuzzy Sets for Trust Management
Katharina Krombholz et al. 2015.	Identified that there exists two risks as the privacy of the users and the fake identities for composing services.	Risk Identification, Service Composition
Ing-Ray Chen et al. 2015.	To combine direct trust and indirect trust dynamically a Filtering technique that could minimize convergence time to determine trust inference prejudice in the presence of malicious nodes.	Dynamic Detection, Malicious Nodes

From table 2.4, it is revealed that many researchers have contributed on the process of Identification of Malicious Services during change management in Long Term Composed Service composition. Further the literature has been narrowed down to identify various aspects to be considered to meet the need for integrating different Web services to form a value added service.

2.6 Web Service Integration Techniques

Web services are applications that are accessible over the Internet to give some sort of service that are either automatic or educational that are identified with different applications. Web services are diverse shape Web applications somehow like; they for the most part include application-to-application correspondence. Likewise they are not planned to be gotten to by means of a Web browser. In a Web service situation, customer applications can be composed in any dialect that backings HTTP and SOAP. To transmit a message, a remote method call is activated to a Web service, which processes on the message and returns a reaction back to the customer. UIs are not made for Web services, it is for the most part up to the customer to process info and show yield.

Eleni (Eleni Stroulia et al. 2013) proposed an Intelligent-Agent Architecture for Flexible Service Integration on the Web. They for the most part gave the particular dialect of the mix workflow and the intelligent astute agents that translate this dialect at run time. The determination dialect utilized here is XML. They are additionally actualizing the incorporation design through construction dialect. Assignment agents and the wrapper agents are created, where errand agents build up the workflow for the combination process and the wrapper agents go about as the delegate between the undertaking agents and the current Web applications which comprise of the services to be incorporated. They exchange the composition related info and yield to http solicitations to the service side applications.

Web Service Technology and Service Oriented Architecture have gained attention in recent years. When there arises a need for a new service request, the change management framework does the task of integrating several Web services that suit the user needs. Integration and composition are the key factors of change management framework. Several researches have been carried out for efficient Web service integration and composition. An A* algorithm was proposed by Mier (Mier et al. 2012) to take care of the issue of semantic info and yield message structure coordinating for Web service piece. The calculation is utilized to locate the insignificant piece that fulfills the client ask. A dynamic optimization technique is used to improve the performance of the search process.

Web service composition is an NP hard problem and one of the challenging issues in Web service paradigm. A new combined Web service composition and integration model was developed by Hassan (Hassan et al. 2012). Both Static and Semantic Web service composition techniques were proposed by the authors. The orchestration approach and the choreographic approach were discussed in detail. A Service invocation mechanism which is event-driven was proposed by the authors. A comparative analysis between all the proposed models is also done.

A dynamic Web service composition framework based on OWL-S and HTN was proposed by Tang (Tang et al. 2013). The authors proposed a service integration algorithm based on K-Means algorithm. A two stage service model is done. The HTN planning approach is based on the service type of the service instance. A representative state transformation approach was proposed by Hussain (Hussain et al. 2013) that uses a Uniform Resource Identifier (URI) and hyper links for service access. An idea of RESTful service was developed by integrating Web service that can satisfy the end user requirements. Most of the Web service composition methods include semantic contents. Alex (Alex alevski et al. 2015) introduced the concept of an extended logistics enterprise and to explore the software engineering issues underlying the development of such complex systems.

Tristan (Tristan Glatard et al. 2008) proposed dynamic service coordination in matrix foundation with the assistance of service wrapper which has a workflow motor which performs mix utilizing 3 service systems - association, piece and substitution of services. Despite the fact that methodologies have been viewed as, still system similarity check (i.e. regardless of whether association or piece or lessening should be possible on the asked for rationales) isn't made on rationales which may prompt disgraceful incorporated rationale. Lu Liu (Lu Liu et al. 2012) outlined a service joining framework by incorporating lower level services to shape the capacities. New engineering is proposed for service joining where developments of abilities are considered; as customers may change their prerequisites much of the time yet scientific models were not utilized for automation.

Jiachen (Jiachen Hou et al. 2016) proposed another work process administration framework to meet the necessities of dynamic business changes utilizing service joining. The Analytic Hierarchy Process (AHP) approach makes a reasonable depiction of the issue, supports group basic leadership and aides in organizing which ventures asked for service logics can be separated. Thus the work concentrates only till the discovery level. An intelligent multi agent system was developed by Eleni (Eleni Stroulia et al. 2013) for coordinating Web construct application services based with respect to an extensible incorporation particular language. To interpret and execute integration specifications defined in the language, a reflective intelligent agent was applied.

Thirumaran (Thirumaran et al. 2015) had proposed a framework for change collision analysis for the Long term composed services with the following features. The framework made the business people implement by themselves to implement the changes by means of their analysts. The cost of the framework was reduced by eliminating the need for the IT developers once after the services relating to the application was developed and delivered by them. After ensuring the efficient incorporation of the changes, finite state automata was used by them to verify the compatibilities made during the runtime, change assessment, and probabilistic cell machine was utilized to affect examination and expectation. They demonstrated that the learning picked up by the experts over the service logics and the consolidations of changes were expanded by methods for the likelihood measures and in addition the episode coordinating.

Table 2.5 illustrates the contribution of the researchers as to various tools and techniques used for Web services composition and integration. The researchers have addressed the various aspects of Web service composition when a change request arises.

Table 2.5 Contributions towards Web Service Integration and Composition

Researchers	Contributions	Addressed Aspects
Mier et al. 2012.	A dynamic optimization technique is used to improve the performance of the search process.	Completely Automated, Domain Knowledge.
Hassan et al. 2012.	Both Static and Semantic Web Service Composition techniques were proposed.	Domain Knowledge, Web Service Composition.
Tang et al. 2013.	OWL-S and HTN Planning - A dynamic Web service composition framework.	OWL Methodologies.
Hussain et al. 2013.	An idea of RESTful service was developed by integrating Web service that can satisfy the end user requirements.	Dynamic Web Service Integration.
Alex Talevski et al. 2015.	Made a case study on the concept of an extended logistics enterprise. Also made a study to explore the software engineering issues that are fundamental in the development of such complex systems.	Concept understanding, OWL.
Tristan Glatard et al. 2008.	Dynamic service reconciliation in network framework with the assistance of service wrapper which has a work process motor which performs incorporation utilizing 3 service systems - association, organization and diminishment of services.	Dynamic Service Integration.
Lu Liu et al. 2012.	Planned a service coordination framework by incorporating lower level services to shape the capacities.	Dynamic Web Service Composition.
Jiachen Hou et al. 2016.	New work process administration framework to meet the necessities of dynamic business changes utilizing service mix with Analytic Hierarchy Process (AHP) approach.	Domain Knowledge, Malicious Services, Dynamic Integration.
Eleni Stroulia et al. 2013.	An extensible integration-specification language that is used for integrating Web based application services.	Domain Knowledge, Dynamic Integration.
Thirumaran et al. 2015.	The cost of the framework was reduced by eliminating the need for the IT developers once after the services relating to the application was developed and delivered by them.	Business Policy Detection.
Ying Huang et al. 2015.	Knowledge Agent Architecture for adaptable Service Integration on the Web.	Business Policy Detection, Domain Knowledge, Dynamic Integration.

Table 2.5 shows that many researchers have taken place to bring a value-added service by integrating several services in the Change Management Framework. Even though most of the Web services are built using SOAP and WSDL, the scaling of those services would be difficult with the absence of certain degree of automation. The retrieval of specific Web service can be done by Web service Modeling Ontology (WSMO) where the Ontologies are formed to organize the services so that Web service matching is performed through the reasoning approach. The services offered by the organization through the specific Web portal can be added or replaced without the need to change the entire implementation of the developed system.

2.7 Approaches to Finite State Machine

State Machines (FSMs) are mathematical formalism for depicting forms with a limited number of conceivable states and successive state advances. Finite state machines are used for various applications in many fields. Whenever there are state transitions involved in the system, finite machines come into picture. Many research works have been carried out to show the importance of Finite Stare Machines. Deepak (Deepak Chenthat et al. 2010) have proposed FSM-based modeling scheme for Web service specifications. The authors have shown employments of FSM (Finite State Machine) based models being developed of Web services and furthermore have built up an instrument HUMSAT to show service particular in FSM and to produce executable codes in BPEL and WSDL.

Sung (Sung-Shik et al. 2012) have introduced a compositional development of Web Services, utilizing Reo and Constraint Automata as the principle "stick" fixings. Reo is a graphical and exogenous coordination language in view of channels. Creators have proposed a structure that, taking as information the behavioral portrayal of services (as Constraint Automata), their WSDL interfaces, and the depiction of their collaboration in Reo, produces all the essential Java code to arrange the services one after the other.

A standout amongst the most well known coordination languages for Web services synthesis is Web Services Business Process Execution Language (WS-BPEL). Despite the fact that the taking an interest free service may work accurately, a few unexpected issues might happen amid execution of composite Web service. It is normally hard to distinguish such blames and spread on it. Ching (Ching-Seh Wu et al. 2013) display a method of Model-Based Testing (MBT) to improve testing of cooperations among the Web services. The method joins Extended Finite State Machine (EFSM) and UML grouping outline to create a test display, called EFSM-SeTM. Creators have additionally characterized different scope criteria to produce legitimate test ways from EFSM-SeTM demonstrate for a superior test scope of every single conceivable situation.

Examination of the consistence of a business procedure execution as for an arrangement of controls is a huge issue in a few settings. To defeat this confinement (Riccardo De Masellis et al. 2014) have introduced a novel checking approach that tracks floods of process occasions (that maybe convey information) and confirms whether the procedure execution is passive with an arrangement of information mindful business requirements. These imperatives not just allude to the transient development of occasions, yet additionally to the fleeting advancement of information. The system depends on the formal detail of business imperatives as far as first-arrange direct fleeting logic rules. On process, these rules are converted into limited state automata for powerfully thinking on incomplete and advancing execution follows.

Another approach for Web service show that isolates service practices into operational and control practices has been proposed by (Quan Z.Sheng at al. 2014). The association of operational and control practices at runtime is activated by conversational messages. The creators have proposed a mechanized service confirmation approach in light of emblematic model checking utilizing FSM. The proposed approach extricates the checking properties utilizing fleeting logic recipes and from control practices. These recipes consequently confirm the properties in operational practices utilizing the NuSMV display checker. The approach introduced in this work has been executed utilizing various best in class innovations.

Table 2.6 gives the contributions of several researchers on the use of Finite State Machines for Web services. The use of FSM in many aspects shows that it plays an important role in making any system an automated one.

Table 2.6 Contributions towards FSM in Web Services

Researchers	Contributions	Addressed Aspects
Deepak Chenthat et al. 2010.	FSM-based modeling scheme for Web service specifications.	Completely Automated.
Sung-Shik et al.2012.	A system that takes as info, the behavioral depiction of services (as Constraint Automata), their WSDL interfaces, and the portrayal of their cooperation.	Web Service Integration, Completely Automated.
Ching-Seh Wu et al. 2013.	A strategy of Model-Based Testing (MBT) to improve testing of associations among the Web services.	Testing Tools, Web Service Integration, Completely Automated.
Riccardo De Masellis et al.2014.	Formal particular of business limitations as far as first-arrange straight transient logic rules.	Formal Representation.
Quan Z.Sheng et al. 2014.	An approach that concentrates the checking properties, as worldly logic equations, from control practices, and naturally confirms the properties.	Formal Specifications.

Table 2.6 shows that contributions have been made by several researchers to use the Finite State Machine for the purpose of Automation. We extend the research to automate the process of Business Policy Extraction using FSM.

2.8 Comparative Analysis of Existing Research

The following section briefs the comparative analysis of the research works carried out by the researchers towards the Web service integration with policy detection and identification of fake identities of the Web services.

Table 2.7 Comparative Analysis of Existing Research

Contributors	**Aspects of Research in terms of Automation and Domain Knowledge with Policy Detection**				
	Completely Automated	**Business Policy Detection**	**Domain Knowledge**	**Identification of Malicious Services**	**Dynamic Integration**
Xumin Liu et al. 2010			✓		✓
Arif Ali Khan et al. 2010	✓		✓		
Philipp Leitner et al. 2010		✓			✓
Philippe Leitner et al. 2013	✓	✓			
Quanwang Wu et al. 2014	✓	✓			✓
Ouransa et al. 2013	✓	✓			✓
Wang et al. 2011	✓		✓	✓	
Szu-Yin et al. 2014	✓		✓		✓
Mier et al. 2012	✓		✓		
Jiachen Hou et al. 2016			✓	✓	✓
Eleni Stroulia et al. 2013			✓		✓
Ying Huang et al. 2015		✓	✓		✓

2.9 Limitations of Existing Research

Organizations utilize different specialized strategies to share the data and assets between the heterogeneous frameworks. Among the different techniques available, the Web Services are the most ordinarily utilized framework to share administrations which are more adaptable and a successful answer for the undertaking community oriented business frameworks. Administration reconciliation is the procedure to empower the associations to advance incorporation development between various endeavors. Additionally an undertaking joint effort framework stage takes the general supervision on all data accomplishments. The collaboration process is responsible for the registration of different enterprise resources and also for checking the database present in the system, information sharing and exchanging data by releasing certain information. The present market situation isn't with incorporating applications or Web service overall yet to soak to pull out fundamental bit of logic from the whole service. This develops an entire administration by coordinating these uncovered logics. Incorporation at this level has numerous difficulties. Expected logic to determine the service integration request for must be situated between the edgy services of different accomplices.

Found service logic must be separated and examined with its needy parts, and after that it must be incorporated in a legitimate mode. In addition, openness level to use the found logic must be confirmed before going into the incorporation procedure. In spite of the fact that interest for dynamic service coordination develops high and furthermore has numerous focal points, for example, business benefit mechanization, reusability, business readiness, and so on, these issues make it minor.

In the related works, different approaches were provided for service integration and its workflow. The greater part of the proposed thoughts utilized pattern level integration or code level integration and one of them incorporated the different coordination methods like union, composition and substitution. The typical form of coordinating administrations happens thusly – the service demands are acquired, the required construction are extricated and they are consolidated powerfully utilizing a particular method lastly gave as a support of the end-client. The real issue here is the point at which a piece of a construction or code is extricated from a whole administration they are not finished and limited. They will be interconnected to different parts of the code through capacity calls. At the point when these kinds of deficient services are coordinated, the combination will end in a contradictory stage. Consequently, another system has been proposed for benefit joining of business logic with the interoperable objectives, for example, Computability, Completeness and Configurability and alongside it to diminish the time and cost for integration.

The construction is driven inside a numerical model, investigating the stream of business logic by examining its parts like capacities and parameters. Mostly for benefit joining, the elements if the business logic needs to interestingly examined, this procedure of breaking down is called as property assessment. The property assessment is finished utilizing the Finite State Machine in this way expanding the level of automation. Limited state machine is by and large utilized for taking care of dialect calculability hypothesis. It has its own particular essential end functionalities meaning a specific demand as 'reasonable' or 'unsolvable' called as decidability. This decidability of limited state machine is here utilized for breaking down completeness, finiteness, and traceability and configurability properties of rationales and furthermore for the early expectation of treatment of administration demands.

From the survey, the following limitations are identified in the existing research:

- Extraction of Policy and Policy violation detection are not automated.
- Handling multiple requests is a complex and time consuming task.
- Domain Specific Knowledge for integration should be improved.
- Malicious Service labels may cause fake identities of services which may lead to integration conflicts.
- The service integration system is not automated.

2.10 Research Objectives

Service integration has been developed to allow the enterprise to reach integration maturity by integrating the organizations’ applications among their companions. Though service integration is useful and popular, the present market need does not get fulfilled with incorporating solicitations or services as a whole substance. It involves incorporating Web service logics at useful level or business run for fluctuated need quickly.

Several important challenges are available that should be faced at the level of Integration.

First and foremost is the amount of loosely coupled system that should be maintained mainly when the business partners want to tie up with others or merge their functionality with others.

Secondly lays the requisite logic that must to be positioned properly and have to be taken out from the entire service.

Third and the most essential thing is that the recouped logic must be coordinated capably as the interoperability issues don't emerge. This is a troublesome undertaking that expects engineers to comprehend the entire services and recognize a superior route for combination. In this way it is required to devise a mechanized framework to incorporate the Web services powerfully.

Change management with respect to LCSs comes up with a set of research issues. An LCS tries to outsource its functionality from different independent service providers. In order to monitor and manage these service providers, there are no central control mechanisms. And so, the challenge of administrating the changes lies in providing an end-to-end framework that could manage the service providers in a top-down change manner that could reacts to the change in an effective way. The summary of the major research issues in managing changes in LCSs are as follows.

Change model

To understand and process the Changes, they have to be first captured and identified. Changes may vary from each other as far as their inspirations, necessities, and practices, and so on, which normally triggers a need to separate changes where distinctive procedures can be sent. Conventional approaches of related to top-down changes are temporary and informal. Those techniques involve exhaustive human interventions. Hence, change ought to be demonstrated formally with the end goal that it is justifiable and process capable by machines.

Change reaction

Top-down changes are always connected to a new requirement on an LCS. These changes may necessitate the LCS to put in new features. They may also be triggered to improve privacy protection of services. On responding to such a change, an LCS should change the services that it outsources. The way that the outsourced administrations coordinate with each other may likewise be changed. It would present a high cost if the procedure is performed physically. Computerizing the procedure of progress requires the data about the outsourced administrations and their joint effort be comprehended by machines. Only on proper understanding of the required change, modification on an LCS can be performed to fulfill the requirements that can fulfill user need by the change.

Semantics support

The process of automation of change management can be improved with the help of software agents that can understand the problems they are dealing with. They follow several approaches to do the process of automation. In particular, they need to have adequate knowledge about LCSs and the services outsourced by them. In databases, the automation of data queries and updates relies on meta-data (i.e., data types and data schema). A meta-information similar to that of databases is also required to make available a semantic support for the process of change management.

Several analyses should be done in order to meet the challenges in the research. The important analyses are:

Request Analysis

The request for new service integration may be encountered previously and by analyzing its previous incidence on comparing with the service ID and the decision for its integration can be made with less effort and time.

Domain Analysis

If the request is a new one and not been encountered previously, then, it is analyzed to which domain it belongs to such as travel, airlines etc., and then categorized to the corresponding domain and the corresponding policy control is generated for that service.

Business logic Analysis

In this phase only the business logic of the request is analyzed such that it copes with the existing logic of the business and this can be accomplished by policy detection and the level of policy violation of the respective service. The detection of policy points is performed with the help of the ATM and this is explained in detail in the following section.

In order to make the complete process of service integration dynamic and automatic one, the proposed system should meet the following objectives:

- To Extract the Policies from the already existing Long Term Composed Services (LCSs). These services may have different policies at a different level of abstraction. These differences in Business Policies are to be detected by constructing two Multi-Tape Turing Machines.
- To detect the Policy Violations that occur in the services of the two LCSs at various levels such as Rule level, Function level, Parameter level and Dependency level using the Multi-Tape Turing Machines. This Business Policy Violation detection has to be done for multiple change requests arising at a time. To detect policies of multiple services for multiple change requests, Alternating Turing Machines (ATM) are constructed.
- To Adopt OWL Methodologies to handle domain specific change request emerging from the users of such LCSs.
- To measure the reputation of services through RM by considering malicious intentions. This phase is operated if a policy violation is detected. Reputation Measurement is done for Ballot Stuff Attack and Sybil Attack.
- To handle Multiple Change Request dynamically and completely automatic. In order to fulfill this objective, a multi-agent system is designed.

2.11 Summary

This chapter presents a thorough study of theoretical foundations for dynamic service integration which help us understand that the enterprises need to give their business logics in a more complex and secured route to their accomplices. Beside performing mix, avocation for not performing joining is similarly found. The nonattendance of potential is seen in the property appraisal, through which the specialist co-ops can begin taking a shot at tackling those issues for better coordinated effort of their services. A complete study has been made to identify the issues in the existing researches on Web service integration for a new long-term integrated service. On proper analysis of the issues, the challenges that lie ahead of this research are identified. The following chapters demonstrate further research that constitutes automatic Web service integration without policy violations and explain its aspects.

CHAPTER 3
PROBLEM DEFINITION AND METHODOLOGIES

3.1 Overview

This chapter gives an overall idea of the proposed work. The challenges in the Change Management Framework have been studied and to completely automate the system of integrating the already composed long-term composed services, the Finite State Machine has been constructed to accomplish the task of integrating the services to a value added service without violating the business policy rules framed by the service providers. The ATMs (Alternating Turing Machines) are employed to perform the task of detecting the policies dynamically when a request for change is encountered. The objectives of the research are formulated here. Hotel and Travel services are taken as datasets and composing these services is accomplished and the results are produced.

3.2 Research Problem

Whenever there is a change request for a new LCS, the proposed system tries to overcome the following research problem:

- ***Construction of Turing Machine for Policy Detection***

Business policy plays a major role in the process of service integration. Hence a careful model is to be developed to take care of the business policy. An automated system is also required since the process of service integration is dynamic in nature. Therefore, multi-tape Turing machines are developed to detect the business policies of different services. A Framework which uncovered the necessities as business rules and concentrates the required logics in runtime through business logic supervisor must be created.

- ***Matching the policies between the two LCSs***

Service Integration is a process of making two or more different services of different service providers to collaborate with each other and share their data in order to provide the customer with the best service they need. While collaborating with each other, the policy decisions of the services being integrated should be considered.

Hence an automatic Business policy detection and matching of policies with each other is done with the help of a newly constructed Turing Machine.

- ***Handling Domain Specific change request using Ontology***

  Web service Ontology comprises of an arrangement of service concepts. A service concept characterizes a sort of Web benefits inside a space. It catches the normal highlights of Web services. A Web service concept can be seen as a dynamic one, which can be instantiated by solid Web services. Using the service concepts of the Ontology, Web service functionalities are defined in a way that is clear and unambiguous to software agents.

- ***Reputation measurement against Malicious Attack***

  It has been found from the study of Web services that the service provided by some service providers does not fulfill the user requirement but still their credit scores are shown high. Because of their fake identity, owners who want a change tend to make use of those services and end up with an improper integration. In order to avoid these kinds of problems we have come up with a solution by making use of Reputation Measurement against malicious attacks to find out the fake identities with the help of user credits.

- ***Multi-Agent Architecture for collaborative service integration***

  Web service Integration is a complex process which involves various complex interfaces to locate the required service. It requires a strong authentication and access control mechanism that could be used and utilized by the partner's logic at business logic schema level. Also the service level agreement negotiation process should be carried out in an effective manner. To handle such various important tasks, several agents might be applied. Hence multi-agent architecture is developed for collaborative service integration.

3.3 Overall Architecture of the system

As listed in the above section, there are several objectives to be achieved in order to give a complete solution to dynamic Web service integration without any policy violation and identification of fake services. The overall architecture of the proposed system is as shown in Figure 3.1. The whole process of service integration has three folds. Initially the incoming user requests are to be analyzed. After a thorough analysis, the services that are to be composed to form a value-added service are then identified and the business policies of those services are matched for policy violations. After this process, the fake services are identified and the reputation is measured.

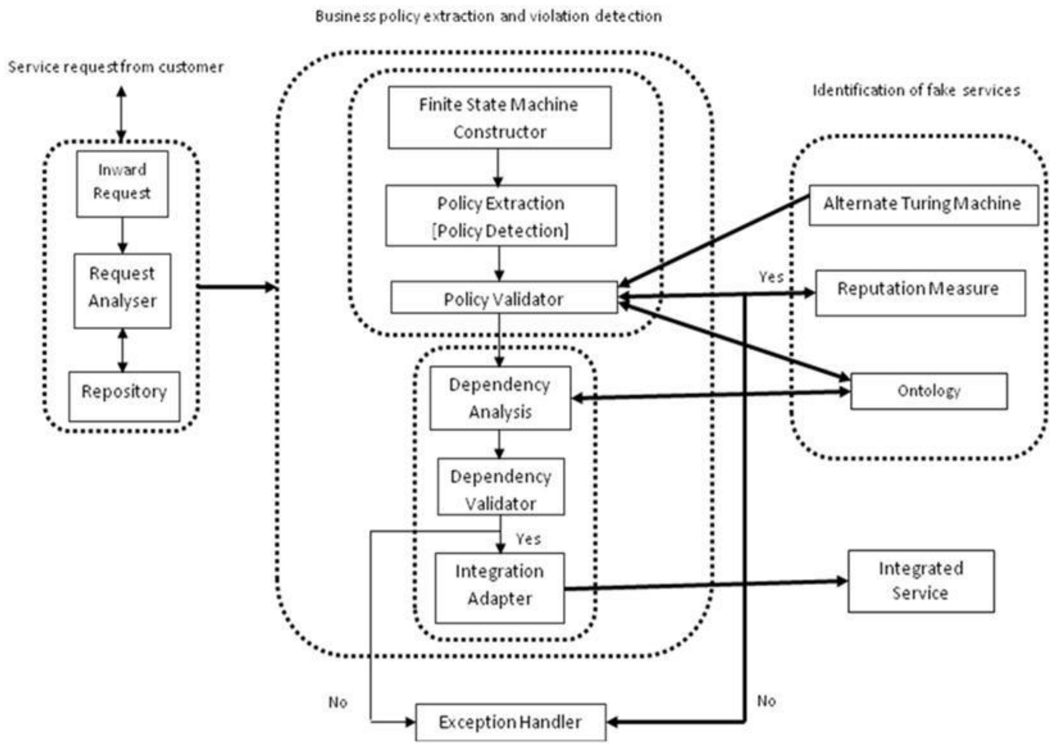


Figure 3.1: System Architecture

3.3.1 Analysis of User Request

Service discovery is typically considered as an inquiry looking for a service and consequently discovering a service from the database that matches the parameters introduce in the question. Questions are the client asks for that emerges for a change. To make a last arrangement, the established arranging diagram calculation utilizes a retrogressive pursuit which is the most tedious piece of the current systems. Rather than enhancing the retrogressive inquiry, endeavors are placed into evacuating the excess Web services amid the foundation of Web service disclosure. It is still ensured that no less than one arrangement will be kept after we evacuate these excess Web services. Once a user request arrives, the system first analyses the request and checks to see if the request can be fulfilled by the services available in the service repository. If a single service is capable of fulfilling the request, it is given to the user. Or else, if there is a need to integrate the available services to give a value added service to the customer, then the next module progresses.

3.3.2 Business Policy Extraction for Requested Services

Web Services Policy (WS-Policy) Framework provides a general purpose model for Web service applications. It also defines the comparing language structure to depict and impart the arrangements of a Web Service with other Web services. WS-Policy is characterized as an arrangement of builds that is utilized and stretched out by other Web Services details to depict an extensive variety of service necessities, inclinations, and abilities. These policies are therefore used at the time of composition of Web services. The two commonly used parameters in policy definition are:

WS-Policy Assertions specifies a set of frequent assertions of message policy that are specified within that policy.

WS-Policy Attachment determines three particular extra components for utilizing policy articulations. This is finished with existing XML Web Service advancements. Specifically, they characterize how to connect policy articulations with WSDL write definitions and elements of UDDI. They additionally characterize how to associate usage particular policy with the entire or a piece of a WSDL port write when they are uncovered from a particular execution.

3.3.3 Identification of Mismatched Business Policies

The crucial objective is to identify the policies during the integration as well as the violations of the business policy appraisal of the services during the process of service integration. The CMF automates these changes directly without any need for sanction from the analyst. Policies of integrating services are extracted and checked for violation using FSM and the results are checked for Reputation Measurement and fake identities.

3.3.4 Identification of Fake Services

The impact of the uncertain and malicious service nodes on the Web service composition (WSC) presentation is usually a serious issue in the Internet. And so the problems of services selection for Web service composition cannot be completely solved in the perspective of performance. A trust model with reputation computing should be built through interactive services composition submitted by the parties, and then a complete reputation evaluation model should be formed. The communication between two service entities might be either direct or indirect. After these processes of communication are completed, the services are given to the integration module for integration. An exception handler handles the exceptions raised in the system.

3.3.5 Service Integration

Service integration is the final process of the proposed methodology. Several Agents are deployed to perform the process of integration. At every level the property evaluator agent and the pattern matching agent are used to check the properties of the integrating services and to create a new value-added service.

The functionality of the system is illustrated with an example.

Consider a Long Term Composed Service (LCS) that offers services for a Tour Package. This LCS consists of Airline Service, Hotel Booking Service, Taxi Service and Payment Service. Suppose a user requests for a service for tour package with Airline service, Hotel service and a Car Rental service instead of Taxi Service, there arises a need for change. This new user request is first analyzed to see if any such LCS is available in the repository to offer such service with car Rental instead of Taxi service. If such a service is available, it is fetched from the repository and the user request is fulfilled. Otherwise, the existing Taxi service has to be removed from the Tour Package LCS and the Car Rental service is added to the LCS. To do this the policies of the composing services have to be checked for policy violation. For example consider that in the existing Tour package LCS, there exists a Business Policy for Payment Mode. This policy says that the payment should be only through Credit Cards. In such a case the composing service ie., the Car Rental service should also have the same Payment Mode Policy. Only then may these services be composed. In order to accomplish this task, we employ Turing Machine. The Turing Machine automatically detects the policies of the composing services and checks whether any violation occurs or not. In our case, the Turing Machine checks whether both the Car Rental Service and the existing Tour Package LCS have the same Payment Mode policy. If so, the Car Rental Service is chosen for composition with the Tour Package LCS. To speed up the process and to handle multiple requests at a given time, Alternating Turing Machines are employed.

Once the Car Rental Service is chosen for composition, it has to be analyzed for Fake identity. Services may sometime be malicious and it may cause serious effects on performance when they are composed with other services. To increase the usage of a service, there are possibilities that the credit rates of those services may be purposely increased by attackers. Hence to avoid such malicious activities, the service chosen for composition i.e., the Reputation of Car Rental Service is measured. This is done by two methodologies in our system namely i) Rater's Creditability Evaluation and ii) Majority Opinion by K- Median Clustering. As a result of these approaches, the reputation of the Car Rental Service is measured. The measurement values decide whether the service is a reliable one or not. If the Car Rental Service is a reliable one, then it is sent to the integration module where it is composed with the existing Tour Package LCS.

In the integration module, several agents play a vital role in composing the Car Rental Service to that of the Tour Package LCS. The property Evaluation Agent checks the important properties like completeness, computability and configurability of the composing services. The SLA Negotiation Agent takes care of the Service Level Agreements of the composing services. The Dependency Analyzer Agent analyses the dependency that exists among the composing services. In the example, users do not necessarily need to provide the information for each service when they are combined into a single service. The input of some services can be derived from other services by the dependency relationship. Finally the services are integrated using the Service Integration Agent to give a value-added service to the user.

3.4 Experimental Setup

The proposed business policy violation detection with Reputation Measurement is implemented in the framework of Visual Studio by means of DOTNET and the implementation details, results obtained with the proposed methodology and its efficiency when compared with other existing works are presented. The experimental results are given both in a tabulated form and analytically through comparison graphs. The proposed business policy violation detection in the CMF based on the reputation of each of the Web services is implemented using DOTNET in the framework of Visual Studio.

3.5 Output Metrics

To assess the proposed Web Service Integration Framework, the accompanying measurements are utilized: Precision and Recall are figured with the assistance of the policy data.

Precision:

Precision is a metric used to figure the recouped strategies that are considerable. That is the quantities of proper arrangements that have been returned by the structure to satisfy the client ask for from the quantity of strategies that are recovered. The equation for precision is given as,

$P = (tp) / (tp + fp)$

$(tp) =$ No of policy enforcing point handled

$(tp + fp) =$ No of PDP (Policy Detection Point) detected.

Recall:

Recall is the part of related approaches that are recovered. Recall is communicated by the proportion of Number of significant polices returned by the structure and the aggregate number of polices for the specific services. The equation for recall is given as,

$R = (tp) / (tp + fn)$

$(tp + fn) =$ Total no of mandatory policies

to evaluate the process of identification of fake services

3.6 Summary

An overall functionality of the system being developed is summarized in this chapter. The challenges that arise in developing the system are focused clearly. An experimental setup is done to check if the proposed system fulfills the objectives of the research. The datasets are chosen so that it is suitable for the proposed model. To evaluate the results produced by the proposed model, suitable evaluation metrics, namely, Precision and Recall are used.

CHAPTER 4
CONSTRUCTION OF TURING MACHINE FOR BUSINESS POLICY VIOLATION DETECTION

4.1 Overview

Policy is a XML document that depicts numerous restrictions for substituting, consolidating and composing the Web service logic to achieve interoperability objective line. After the business approaches are separated from the SLA, it will be given as a contribution to policy evaluator. The policies are present in the rules and functions of the business logic also given as an input to the policy evaluator. Then policies are compared with one another and then the result is sent to the policy evaluator. Module I aim to construct an FSM (Finite State Machine) to detect the policies of the composing services. An algorithm is designed to accomplish the task of finding the Policy detection points in each service that is to be integrated. Several evaluation metrics are used to check the performance of the algorithm.

4.2 Turing Machine

A Turing Machine is a unique computing device that consists of a scanner otherwise called as the read/write head along with a tape passing through it. The tape is divided into several squares. Each of these squares bears a single symbol either '0' or '1'. This tape is the machine's general purpose storage medium that serves as a medium for both input and output as well. It also acts as a memory area for storing the results of intermediary steps of the computation.

A finite number of symbols are given as an input to this computational model on its tape. However, the tape is of unbounded length. The read/write head in the computational model is programmable. Hence different tape heads can be employed to perform tasks simultaneously.

To perform calculation with the device, we can program on it. We check the contribution on the tape in parallel or decimal code. At first the tape head must be set toward the starting square and on the furthest left image. At that point the machine can be set to movement. After the calculation is finished, the machine will stop gaze with the head situated over the square containing the furthest left image of the yield.

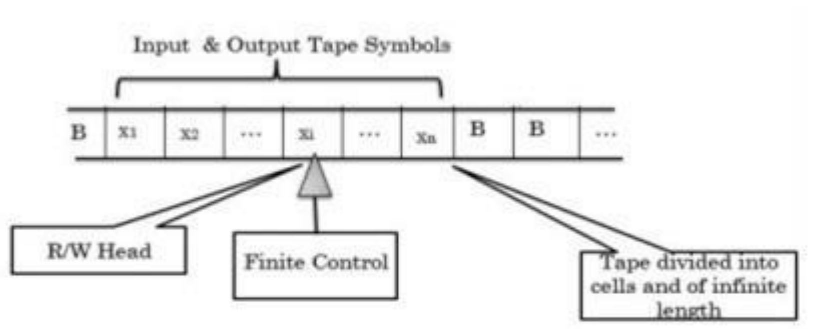

Input & Output Tape Symbols
B
x1
x2
...
xi
...
xn
B
B
...
R/W Head
Finite Control
Tape divided into cells and of infinite length

Figure 4.1: Turing Machine

4.2.1 States of Turing Machines

The Head contains another sub-component called the indicator. This is another form of working memory. The indicator can be set to any of the quantities of positions. This position marker is known as the State of the machine. This position is set once around then. To give a basic case of the marker's capacity, it might be utilized to monitor whether the image last experienced was '0' or '1'. On the off chance that '0', the pointer is set to its first position, and if '1', to its second position.

4.2.2 Atomic operations

There are six fundamental operations that a Turing Machine can perform in its course of computation. Those are listed below:

a. Read or recognize the symbol that is currently under the head.
b. Write a symbol on the square under which the head is pointing. This is done by deleting the symbol that is already written.
c. Move the tape left by one square.
d. Move the tape right by one square.
e. Change the state.
f. Halt.

4.3 Construction of Turing Machine

Two Turing machines M1 and M2 are constructed which correspond to two Long-term Composed sets. M1 and M2 are multi-tape Turing machines. The Tape T1 in the Turing machine represents the Business Rule Tape which exhausts all the rules in the corresponding composed service such as R1, R2…...Rn, T2 represents the Function Tape which exhausts all the functions in the corresponding composed service such as F1, F2.…Fn, T3 corresponds to Parameter Tape which exhausts all the parameters in the corresponding composed services such as P1, P2….Pn and T4 stands for Dependency Tape as shown in Figure 4.2. In response to a request for service integration (Integrating LCS1 & LCS2) the compatibility among the services for integration is verified. This is elucidated in Figure 4.2 by inspecting rule R1 in tape T1 of LCS1 and rule R1 in tape T1 of LCS2 and verifying the legitimacy of the business policies.

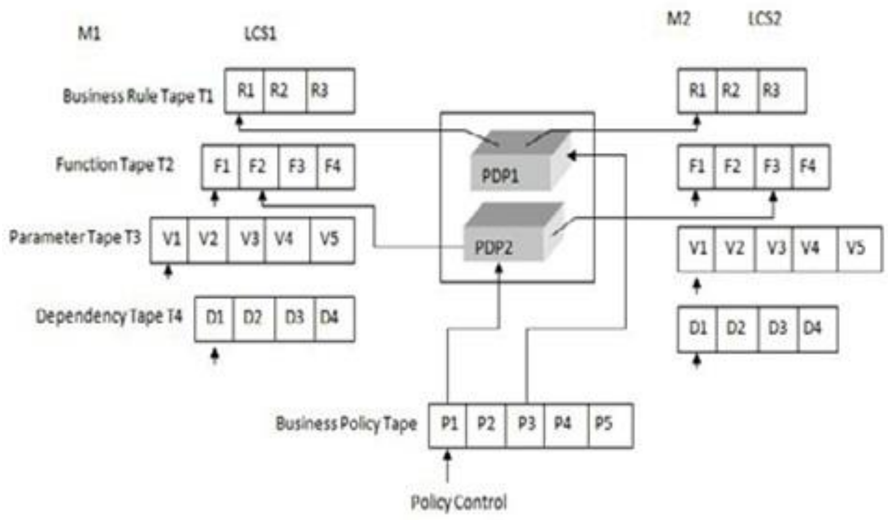


Figure 4.2: Multi Tape Turing Machine

On successful verification, this process is repeated for the remaining tapes until the policy detection points in all the tapes are found to be in agreement. This process is explained in detail in the subsequent sections.

4.3.1 Architecture for Policy Detection

Figure 4.3 depicts the working of policy detection in service integration. The idea of service combination is envisioned as big business' undertaking to stretch out its business procedures to its business accomplices to impart their business logics to each other to perform out the essential activity. To remove the required logics precisely and productively and coordinate the required business logics from the quickly developing business services, engineers must consider the entire service and should settle on legitimate way to deal with combine them. This is a complex and tedious undertakings. The Changes which can be automatically done without any code injection comes under service integration. Before and after integration the output generative power should be alike which indicates that the integration is successful.

4.3.2 Major components for policy detection

- Service Registry
- LCS Set
- Service Profile
- Request Analyzer
- Domain Analyzer
- BL Analyzer
- Property Evaluator
- Constraint Analyzer
- Complexity and Problem Analyzer
- Audit Log

Each of the components listed plays an important role in the system. The working of each component is elaborated in the following section. Since the proposed system is an automatic one, whenever a need for a change occurs, the service repository is first checked for an existing service to fulfill the user request. If it exists, the request is responded and if not, the process starts with the components that are discussed below.

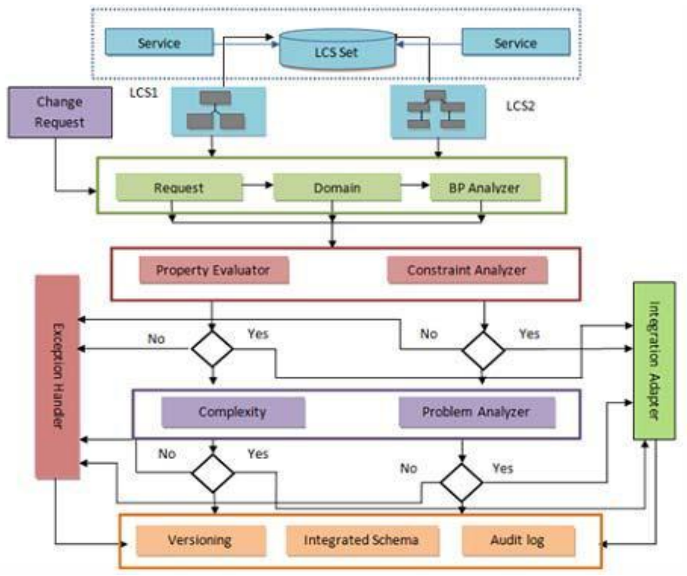


Figure 4.3: Data Flow for Policy Violation Detection

A. Service Registry

Service Registry is an accumulation of services and its analogous details are managed by an Enterprise. It is a local registry of services. The service registry contains numerous services categorized under various domains. Each domain can be further classified into sub-domains and each of the sub-domains includes Business Process which can be for either a single service or a set of services. Whenever changes are being reflected in the services, they get updated in UDDI instantly.

B. *LCS Set*

The LCS set consists of various Long-term composed services which is a composition of independent services extracted from the service registry in diverse arrangements. For instance, this arrangement is represented in the diagram assets LCS1 and LCS2. The LCS is composed to fulfil the business goal for a long run. It is merely integration and orchestration of autonomous services readily available in Web services pool.

C. *Service Profile*

The service profile comprises information about each service that is stored in the service registry such as service Id, service name, and service location, functional and non-functional description.

D. *Request Analyzer*

The new change request is completely analyzed in order to verify whether there is any incidence existing in the incident service repository or is a completely new request and then taking the appropriate decisions. Request analysis refers to analysis of how instantaneously the change request has occurred and under which circumstance the change request has been raised.

E. *Domain Analyzer*

Once the request is analyzed according to the context the domain analysis part is being carried out. The Domain analyzer identifies to which particular domain the change request belongs. The change request is being analyzed extensively and categorized into domains according to the type of the change and its being matched with the entire set of incidents previously.

F. *BL Analyzer*

The change request analysis phase involves processes such as determining the type of change request, the priority of the change request, verifying the feasibility of change, etc. In BL analysis, the required business logic will be extracted and the Business Logic Set will be constructed with all dependent business logic entities. The dependency analysis needs to be done at this stage in order to identify the relationship existing among the entities. The FSM equivalent to the BL Set is constructed with reference to predefined State Transition Table of target service logic and then the Business Logic Schema will be generated from FSM.

G. *Property Evaluator*

Once the request analysis phase is completed, properties of the service chosen for integration are evaluated in order to reveal the fact that the services are eligible for service integration. Property Evaluator acts as a prerequisite for the change management and is performed before and after the change. There are various issues influencing the properties of service. The properties such as dependency, computability, configurability, accessibility and traceability are evaluated and if all the properties are found to be set true, i.e., if all the properties are encountered to be fulfilled and manageable, then the subsequent process of constraint analysis is carried out.

H. *Constraint Analyzer*

After the properties are evaluated, the constraint factors such as Synchronization, Sequence, Branch, Parallel and Iteration are evaluated.

I. *Complexity and Problem Analyzer*

Complexity analyzer analyses whether the problem can be completed (i.e. service integration process) within the expected time. Then the problem analysis is done to find out whether the problem is solvable or unsolvable (i.e. the feasibility of the problem is ensured).

J. *Version*

As per the difference in change requests requirements put forth by different set of consumers, versioning will be carried out accordingly. Through Versioning, the transparency is provided to the customers due to whom the customers will not be able to notice the changes that have been made. In case of any failure due to the changes, versioning can be used to roll back to the previous state.

K. *Audit Log*

Audit log is an archive catalog that stores change history and correct area alongside the data about the space, sub-area, business process and the service where the change happened. This is like a log document which comprises of the time and date at which the change has happened. The insights about the proprietor who has the rights on that specific square of the business logic and the insights in regards to the changes are additionally put away. It likewise holds data about the business procedure in which the change occurred. The whole changes that have happened and special cases that have been tossed are recorded in audit log for recurrence coordinating later on.

L. *Integration Adapter*

In Integration adapter the services desired for the integration is chosen from the service repository and blended using any one of the integration methods as per the requirement of the change request. Properties serves as a prerequisite for service integration and these properties are evaluated before and after integration. Every time after the integration is being carried out, a new WSDL file will be generated.

M. *Exception Handler*

If any one of the analysis such as Property evaluation, Constraint analysis, Complexity and Problem analysis fail, the corresponding exception along with the reason for the root cause of the exception is exhibited.

4.3.3 Algorithm for Business Policy Extraction

The algorithm for policy detection shown in Algorithm 4.1 is for finding the Policy detection points in each service that is to be integrated. In the LCS set the transitions in the service are verified whether a transition like this $\delta * (q_n , S) \rightarrow P_n$ exists, if so the policy detected points are stored in PDPn[] array correspondingly for each service. The transitions are stored in the state transition table as given in Table 4.1.

Table 4.1 State Transition Table

Start State	Transition	Next State	PDP
Q_0	$\delta^*(q_n, S_j) \rightarrow P_j$	q_3	P_1
q_0	$\delta^*\left(q_n, \{S_1, S_2, S_3\}\right) \rightarrow P_j$	q_3	P_1
q_1	$\delta^*\left(q_n, [S_1]\{S_2, S_3\}\right) \rightarrow P_j$	q_3	P_1
q_2	$\delta^*\left(q_n, \{r_1, r_2, r_3\}\{S_1, S_2, S_3\}\right) \rightarrow P_j$	q_3	P_1
q_3	$\delta^*\left(q_n, [r_1]\{r_2, r_3\}\{S_1, S_2, S_3\}\right) \rightarrow P_j$	q_6	P_1

The policy points are found at rule level and function level so that the reason for policy violation can be known in case violation occurs. Then these PDP arrays serve as the input for policy violation detection algorithm in Algorithm 4.1. And then all the PDP arrays of the services to be integrated are compared with each other. If all the values in a array are same the values in the next PDP array is checked and the status is incremented which is initially assigned zero. The algorithm generates a State Transition table with the help of the LCSs as input and in each state it considers the policy at rule level, functional level and parameter level. When a successful transaction is generated, then it means that a policy has been detected from the LCSs that are taken as input from the Turing Machine.

Algorithm 4.1: Algorithm for Business Policy Detection

Algorithm Policy Detection (LCS[], State Transition Table)

```
//Input :LCS[] ← {LCS1[],LCS2[]LCS3[],……………LCSn[]}
//n ← The number of services to be integrated.
//i ←{S,R,F,Pr,Dp}
// S ←{S1, S2, S3, ……… Sn} [S is the set of services for service integration]
//R ←{R1, R2, R3, ……… Rn} [R is the set of rules in the services]
//F ←{F1, F2, F3, ……… Fn} [F is the set of functions in the rules]
//Pr ←{ Pr 1, Pr 2, Pr 3, ……… Pr n} [Pr is the set of parameters]
//Dp ←{Dp1, Dp2, Dp3, ……… Dpn} [Dp is the set of dependency logic]
// Output: PDP[] –policy detection point (i.e.) policy detected points will be
stored in this array for the each service.
Let P ← {P1, P2, P3, Pn} be the business policy set defined for each service
```

```
BEGIN
While LCSn[i] != empty && $\delta^*(q_n, S) \rightarrow P_n$Pj
do
Extract STTn for LCSn[]
//All the transitions in the services are stored as transitions in the State transition table. The values stored in STT are state, input, next state and policy detection point (PDP).
For each j = 1 to n DO
IF ($\delta^*(q_n, S_j) \rightarrow P_j$) then //Detecting the policy at service level
For each k = 1 to n DO
IF ($\delta^*(q_n, R_k) \rightarrow P_k$) then//Detecting the policy at rule level
For each l = 1 to n DO
IF ($\delta^*(q_n, F_l) \rightarrow P_l$) then//Detecting the policy at function level
PDPn[]← Sj(Rk(Fl)) //Storing the policy detection points in PDPn[] for each service
End If
End For
End If
End For
End If
End For
End While
END
```

If at any point any one of the policies of one service mismatches with that of the policies of the other service, then policy violation is detected. Or else if status value is equal to n (i.e.) for all n services, it has been verified that there is no policy violation. This task is accomplished by the algorithm designed for Business Policy Violation.

4.4 Business Policy Violation Detection

As discussed in the previous section, the services that are to be composed to fulfill a change request have different levels of policies. The policies of each service are first detected using the Turing machine that has been constructed. The Next phase is to detect whether any violations exists in the composing service’s policies. This is discussed in the following sections.

4.4.1 Business Policy Violation

A Business policy expects to represent and oblige the conduct of services and business circumstances. For instance, a policy of supply chain stock may oblige restricts on the scope of stock levels for the assembling procedure in light of the benefits focus of the undertaking. These significant arrangements can be formulated and connected to various parts of Service arrangements. To perform approval to oversee target business arrangements and assets, the extent of monitor business arrangements and assets, and service-level understandings can be considered as cases. Policy is characterized as an official responsibility that wins between a service supplier and a service customer. Scrupulous parts of the service quality, accessibility, duties are settled on the service supplier and the service client. These business approaches are contained in the Service Level Agreement (SLA). At the point when such approaches of various services are not tuned in to each other i.e., when there is an infringement in the services that are to be made, serious impacts may emerge.

Neglecting to meet SLAs could bring about genuine monetary results for a supplier. Along these lines, service providers are indicating enthusiasm for picking up a decent comprehension of the connection between what they can guarantee in a SLA and what their IT framework is equipped for conveying. It is along these lines important to distinguish the policy infringement preceding the services being coordinated to defeat a portion of the genuine results. An algorithm has been designed that could be simulated via the Turing Machine to detect such violations automatically which is illustrated in Algorithm 4.2.

4.4.2 Algorithm for Business Policy Violation Detection using TM

The policy violation detection algorithm takes the PDP array which is generated at the policy detection level as input which consists of the policy points in the services of each of the integrating services. Each of the PDP is checked with other PDPs to check if there is any state which does not match with the policy points. Algorithm 4.2 illustrates this process.

Algorithm 4.2: Algorithm for Business Policy Violation Detection

```
Algorithm Policy Violation Detection(PDP_n[])
  //Input: PDP_n[] – Policy detection point array where the policy points detected in
  the services are stored for the each service that is to be integrated.
  //   PDP_n[]← { PDP_1[],PDP_2[],PDP_3[],…………PDP_n[]}
BEGIN
status ←0
While PDP[] != empty
For each i= 2 to n do
If (PDP_{i-1} == PDP_i) then
status ← status+1;
End If
End For
If (status ==n ) then
  //   If status < n (i.e.) if policy is violated even at least once then there is policy
  violation
policy_violation =0 //There is no policy violation
Else
policy_violation =1//There is policy violation.
END
```

As shown in Algorithm 4.2, PDP arrays serve as input for policy violation detection algorithm. The PDP arrays of services that are to be integrated are compared with each other. The corresponding rules, functions, parameters, dependency set and policy set are retrieved. By this the change specification is checked whether it violates any policy. If there is any violation, the analyst is informed as the change cannot be made due to the violation of policy. Otherwise the change is successfully included in the existing logic set L. If the values are same, the status pointer is incremented which is initially assigned to zero. At any point the policy of one service mismatches with the policy of other service and then it returns a policy violation is detected.

4.5 Evaluation Metrics

To evaluate the proposed Web service integration system Precision and Recall is calculated with the help of the policy information.

Precision:

The number of appropriate policies that have been returned by the framework to fulfil the request from a set of policies that are retrieved.

$P = (tp) / (tp + fp)$

(tp) = No of policy enforcing point handled

$(tp + fp)$ = No of PDF detected

Recall:

Recall is the fraction of related policies that are reclaimed. It is expressed by the policies returned by the framework and total number of policies.

$R = (tp) / (tp + fn)$

$T =$ Total no. of mandatory detected

4.6 Experimental Results and Discussion

Consider two travel agencies LCS WS_1 and WS_2 which include services such as airlines, hotel booking and cab service. The rules present in the airlines service S_1 of LCS WS_1 are R_1 and R_2 and similarly rules present in the airlines service S_4 of WS_2 are R_3 and R_4 respectively. Since the performance of R_4 of service S_4 is better than the performance of R_3 of service S_1; a change request arrives for the integration of the first rule R_1 from service S_1 and the second rule R_4 of the second service S_4. First, the policies are detected from those services and stored in arrays PDP_1[] and PDP_2[].The state transition table for the two services S_1 and S_4 are shown in Table 4.2 and table 4.3 respectively. Once the policies detection step is over policy violation check is done. If it is identified that the policies detected are found to be different, it implies that the above mentioned rules R_1 of service S_1 of first LCS and rule R_4 of service S_2 of second LCS are not compatible to be integrated because of business policy violation. Suppose there had been no policy violation, then those two rules would have been preceded for integration without any hindrance.

Table 4.2: State Transition Table for LCS1

LCS	Service	Rule	Transition	Policy
WS1	S1	R1	$\delta^*(q_1,[r_1]\{r_2\}\{S_2,S_3\}) \rightarrow P_1$	P_1
WS2	S2	R2	$\delta^*(q_8,[r_2]\{r_1\}\{S_2,S_3\}) \rightarrow P_5$	P_5

Table 4.3: State Transition Table for LCS2

LCS	Service	Rule	Transition	Policy
WS1	S1	R3	$\delta^*(q_1,[r_1]\{r_2\}\{S_2,S_3\}) \rightarrow P_1$	P_1
WS2	S4	R4	$\delta^*(q_8,[r_2]\{r_1\}\{S_2,S_3\}) \rightarrow P_5$	P_5

As illustrated in Table 4.2 and Table 4.3, when services of two Long Term Composed Services LCS1 and LCS2 are to be integrated their policies are identified to determine which policy matches with the other policy in two long term composed services. The crucial objective is to detect the policies during the integration as well as the violations of the business policy assessment of the services during the process of service integration. This leads to increase in the degree of automation of this framework which is facilitated by property pre-evaluation.

Table 4.4 illustrates the evaluation results of the policies of services that are returned by the Turing machine with the help of the algorithm designed to do it. Precision and Recall are also calculated to evaluate our proposed system. The table additionally demonstrates the aftereffects of the service reconciliation. The service section demonstrates the services (S1, S2) to be incorporated. The quantity of PDP and PEP which are distinguished by the policy evaluator helps in ascertaining the precision and recall.

Table 4.4 Evaluation Results of Policy violation Detection

Service Name	Total no. of Mandatory Policies	No. of PDP detected	No. of PEP handled	Precision	Recall
[S1]: Travel [S2]: Hotel	5	5	4	0.8	0.8
[S1]:Retail [S2]:Payment	10	9	7	0.7	0.7
[S1]:Hospital [S2]:Payment	6	3	2	0.6	0.3
[S1]:Bank [S2]:Customer	5	5	5	1.0	1.0
[S1]:Register [S2]:Login	6	5	5	1.0	0.8

From Table 4.4, it is clear that the system is able to detect all the mandatory policies at a given time automatically. It is also observed from the table that the integration time reduces if the number of PDPs is equal to the number of PEPs.

4.7 Summary

The detection of business policies in the Long Term Composed services and the detection of policy violation help us to avoid the occurrence of abortion of transactions in services in advance. In case there is no detection of policy violation prior to the process of integration, the services would be integrated without any warning and then during the run-time of the service the transaction gets aborted in the middle. And also the detection of business policy at service level, rule level, function level gives us additional knowledge about why and where exactly the policies have been violated. The consumers will not be able to observe much difference in user interface characteristics in the corresponding versions before and after the integration. The change request for service integration is done automatically by the Change Management Framework without receiving any approval from the analyst thereby leading to the increase of degree of automation of the Change Management Framework.

CHAPTER 5
ALTERNATING TURING MACHINES FOR BUSINESS POLICY VIOLATION DETECTION

5.1 Overview

In order to detect the policy violation at first of the policies that are satisfied by LCS have to be identified and this is evaluated by the policy detection points. Here the system uses ATM to detect the policy points of the service which is requested by the user. Initially the proposed system has detected these policy points through multi-tape Turing Machine but it was noticed that by combining universal and existential conditions it became possible to detect the policy points by their status as either the existing case, universal, acceptable or subject to rejection.

5.2 ATM in detecting policy points

Alternating Turing Machine (ATM) is a kind of Non-Deterministic Turing Machine (NTM); generally Turing Machines are used for the purpose of computation. The ATM can be defined mathematically with some parameters and conditions and this is presented as follows:

Definition: An ATM A is the seven-tuple machine as given by

A= (t, Q, $\prod$, Ψ, Υ, q^0, h)
Where, t = Number of Work tapes
Q = Finite set of states
Q = Existential $\cup$ Universal = E $\cup$ U
$\prod$ = Finite input alphabet
Ψ = Finite work tape alphabet
Υ = Transition Function corresponds to position of Left, Right and Stationary
q0 ϵ Q is the initial state
h: Q $\rightarrow$ { $\wedge$, $\vee$, accept, reject}

As given in the definition of ATM, h specifies its output and if h(q) = V, then q is said to be universal(q) = W; then q is said to be existential, h(q) = accept; then q is said to be accepting state; h(q) = reject; and q is said to be rejecting state. The working principle of policy detection with ATM is given in Figure 5.1. As given in figure 5.1, the ATM will detect the policy detection points from N number of LCS using N number of ATMs. If the request to integrate N number of LCS is received, it means that the compatibility among those services is verified for all the tapes as specified. In this case, there are four possible outcomes such as universal, existential, accepting and rejecting, and based on this their compatibility is decided. The working principle of ATM is similar to that of a Turing Machine and the difference is in definition of transition function and this is given in the following equations.

Transition Function of Turing Machine,

$$\gamma : Q \times \Pi \times \{L, R\}$$

The Transition function of ATM is given

$$\gamma : P\left(Q \times \Pi \times \{L, R\}\right)$$

Thus the acceptance and rejection results for ATM is decided by the transition function based on the universal and existential states and the corresponding conditions are as follows:

Condition for Acceptance:

1. If the tape T is existential and some $t \in$ accept then $T \in$ accept
2. If the tape T is universal and all $t \in$ accept then $T \in$ accept

Condition for Rejection:

1. If the tape T is existential and all $t \in$ reject then $T \in$ reject
2. If the tape T is universal and some then $t \in$ reject then $T \in$ reject

In Figure 5.1, M1 and M2 mean the Turing Machine which compares to various Web services WS1 and WS2. These are Multi-Tape Turing machines. The Tape T1 in the Tuning machine signifies the Web service Tape which relates to asked for service to be coordinated. Tape T2 shows the business logics tape. More often than not for a Web service there might be at least one business logics which might be removed from the Web services. Tape T3 means a Business govern tape which utilizes every one of the rules in the predictable service, for example, R1, R2, R3, and R4. T5 means the Function Tape which expends every one of the capacities in the predictable service, for example, F1, F2, F3, and F4. Tape 4 and Tape 6 are the critical tapes. They are called Policy tapes where T4 holds arrangements for rules and T6 holds strategies for functions.

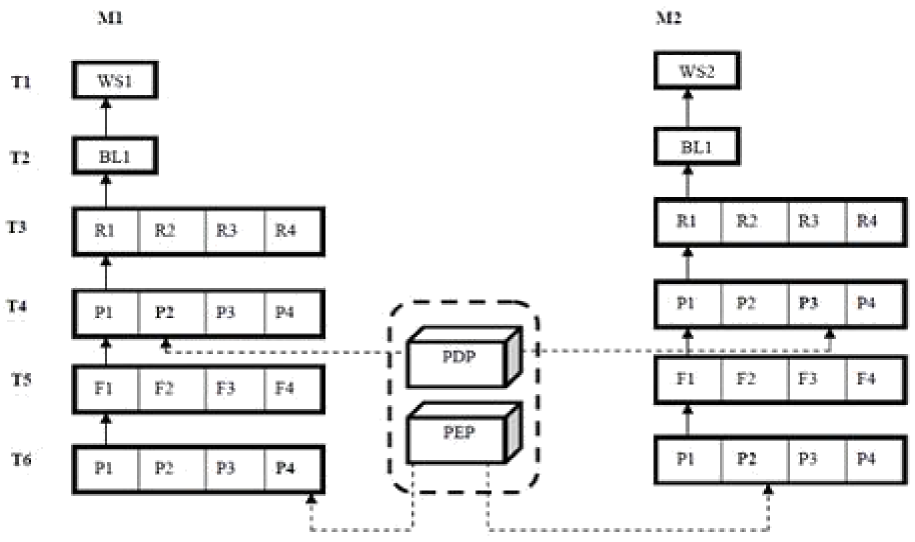


Figure 5.1: Policy Detection Points with ATM

There are two arrays shown here called PEP and PDP. At the point when the machine peruses the tape one by one it will coordinate the approaches of one Web service to other service. When it finds the proper match then it is said to be a Policy Detection Point. In specific cases the specific policy might be obligatory, that is, which has higher need and can't be overlooked. Now of time, Policy Enforcing Point raises and tosses the expected data to the client. In some different cases the tape needs semantic reference and it alludes to ontology for area particular information for fitting location of policy.

5.2.1 Policy Detection Point

Policy Detection Point (PDP) is a special data structure is used in the proposed system. This is a special type of array. When the machine reads the tape one by one, it will match the policies of one Web service with that of other services. When it finds a Policy match with other similar service, then it is said to be policy detection point, these policies may or may not be mandatory.

5.2.2 Policy Enforcement Point

A Policy Enforcement Point (PEP) is a component that serves as the gatekeeper and “front door” to a Business Policy. When a user tries to access a file or other resource on a computer network or server, the PEP will depict the policy’s attributes to the Policy Decision Point (PDP), request a security decision which will then enforce that decision. PEPs are compatible with Web servers, portals, legacy applications, LDAP Directories, SOAP Engines (e.g. Apache AXIS), and similar resources. Services can be configured to communicate with any PEP through an interface.

5.3 Algorithm for Business Policy Violation Detection using ATM

In order to perform the task of policy violation detection in an efficient manner and to respond to multiple requests at a time, an algorithm has been designed for Alternating Turing Machine. The algorithm takes the PDPs in the PDP array of each Turing Machine present in the multiple tapes. The tape is at the respective level (Business rule, Function, Parameter and Dependency logic) that contains number of blocks and for the corresponding t^{th} block the transition is verified. Similarly the state called Deadline will appear at the time when no transition rule is applied to the tape. Hence, the deadline for universal is accepted state and the deadline for existential is 'rejected state' for the tape.

Based on the acceptance, the policy detected points (PDP) are obtained once after the application of transition rules and the corresponding PDPs and the associated transitions are stored in the state transition table and used in the process of policy violation detection. The table is very much similar to that presented in our first work Tiroumalmouroughane (Tiroumalmouroughane et al. 2015). The policy points are detected at the function and rule level and thus lead to effective policy violation detection. Since in the previous work ATM was employed to detect the policy points, no separate algorithm is necessary for the detection of policy violation. The process of ATM in detecting policy violation can be depicted as in the Algorithm 5.1.

Algorithm 5.1: Algorithm for Policy Violation Detection using ATM

```
    Algorithm: Policy Violation Detection by ATM
    Input: Transition function =Policy control
    (Set of Defined Policies) = Tape ( )
    Output:
    For
    If, then service is accepted at Rule Level
    Else if, then is accepted at Function Level
    Else if, then is accepted at Parameter Level
    Else if, then is accepted at Dependency Level Else
Return
```

Figure 5.2 describes the organization of Tapes and the PDP Arrays for policy violation detection using ATM.

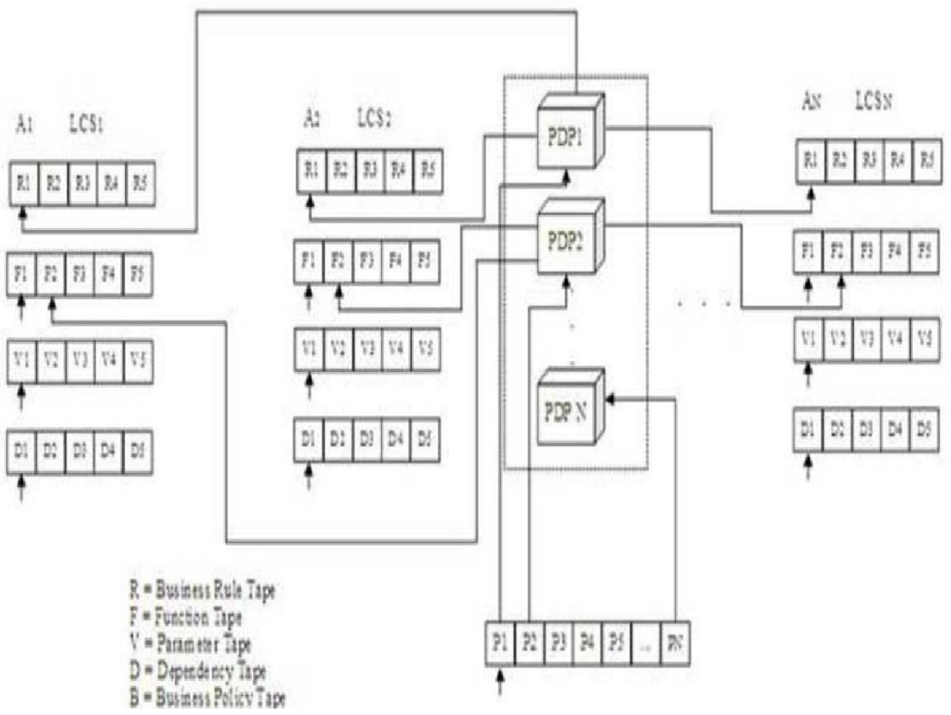


Figure 5.2: Policy Violation Detection with ATM

After detecting the policy violation using the Algorithm 5.1, if the violation is occurred at more than one level means the Reputation Measurement of the corresponding service will be carried out. This is done to find out its necessity through Reputation Measurement and if the service yields better reputation, then it will be integrated to the CMF. Otherwise it will be thrown to the exception handler. The Reputation Measurement of the service which violates the policy is explained in the following chapters.

5.4 Comparison on the performance of ATM with that of TM

The performance of the proposed methodology is proved by comparing the results produced by it with the existing techniques and here the performance of the proposed methodology is compared in terms of Business Policy Violation detection by the number of services with ATM and TM. The comparison results are tabulated and depicted in Table 5.1. The process of the business policy violation detection is given in the form of the comparison graph which is discussed in the following sections.

Table 5.1: Business Policy Violation Detection with ATM and TM

Number of Requests	**Number of policy violated**	
	ATM	TM
1	0	0
2	0	0
3	1	0
4	2	1
5	3	2

From Table 5.1, it is clear that the performance of the employed ATM for the policy violation detection detects more services when compared with the TM and hence the integration of valid services becomes improved. As the number requests increases, the Alternating Turing Machine detects more number of business policies.

5.5 OWL Methodologies for Policy Violation Detection

WSDL (Web Service Description Language) is an available Web service portrayal language that depicts the useful data of services, for example, input parameters, yield parameters and service providers. The service areas are additionally portrayed and have limits at the bottom of the disclosure, execution, organization and interoperation of Web services. As WSDL can't portray semantic data of Web services, an ontology-based Web service interface technology is required. This can depict the language structure as well as the semantics of services. The services incorporate not just the essential of static data through Web locales, yet in addition activities, for example, offering items and driving physical hardware. To use Web services, agents must be utilized. At that point the services must be semantically depicted with the goal that product agents can translate and process them self-ruling.

Web service ontology comprises of an arrangement of service concepts. A service idea characterizes the importance and sort of Web services inside an area. It catches the normal highlights of Web services. A service idea can be seen as a dynamic service, which can be instantiated by solid Web services. The Web services can be characterized into a few classifications in view of their functionalities. Utilizing the service concepts show in the ontology, Web service functionalities are characterized in a reasonable and unambiguous approach to programming agents.

5.5.1 Ontology in Business Policy Violation Detection

The ontology provides support for both semantic and query. The semantic support is used to provide machine-understandable description of Web services. This enables the services be automatically located and orchestrated. An efficient way to retrieve the semantics from the proposed ontology is the query support. The work focuses on the semantic support. Once the policies are extracted from the SLA, it will be given as input to policy evaluator. Policies are compared with one another and if any semantic reference is needed the PE refers ontology.

5.5.2 Architecture of Violation Detection using Ontology

Ontology based Web Service Composition includes two main steps. The first step is a filtering process. Filtering of the domain service ontology aims at reducing the Web services research's space is the first step. The second step is the generation process of an abstract workflow. The ultimate goal of the proposed work is to develop a semantic approach of composition based on ontology called Domain Services Ontology. The system also suggests a set of novel rules in order to automatically select candidate's abstract Web services for the composition process. This process will enable an automatic generation of an abstract workflow based on concepts and semantic links transformation.

Figure 5.3 represents the architecture of Ontology in Policy violation detection. As illustrated in the Figure 5.3, major components play important roles. The Major components in the architecture and their description are discussed in detail.

a. Request Analyzer
b. LCS Repository
c. Policy Extractor
d. Policy Evaluator
e. Ontology Repository

Every user request goes through all the above said components to detect the violations in the integrating services. The functions of each component are discussed below.

a. Request Analyser

This component is used to analyze the request and coordinate all users' requests for all the nearby hosted Web services. Its main functions include dispatching all incoming requests to be parsed. The other is to invoke the particular method on the requested Web services accordingly and generate the Web service responses to the clients. After which, if the requested Web service(s) require security, the "Request Processor" coordinates with the "Security Agent" for verifying and validating the required security parameters.

b. LCS Repository

On analyzing the incoming request if the Request analyzer finds the service that could satisfy the user request, it retrieves the service from LCS repository. This repository is a place where the already composed Long-Term Composed Services are present.

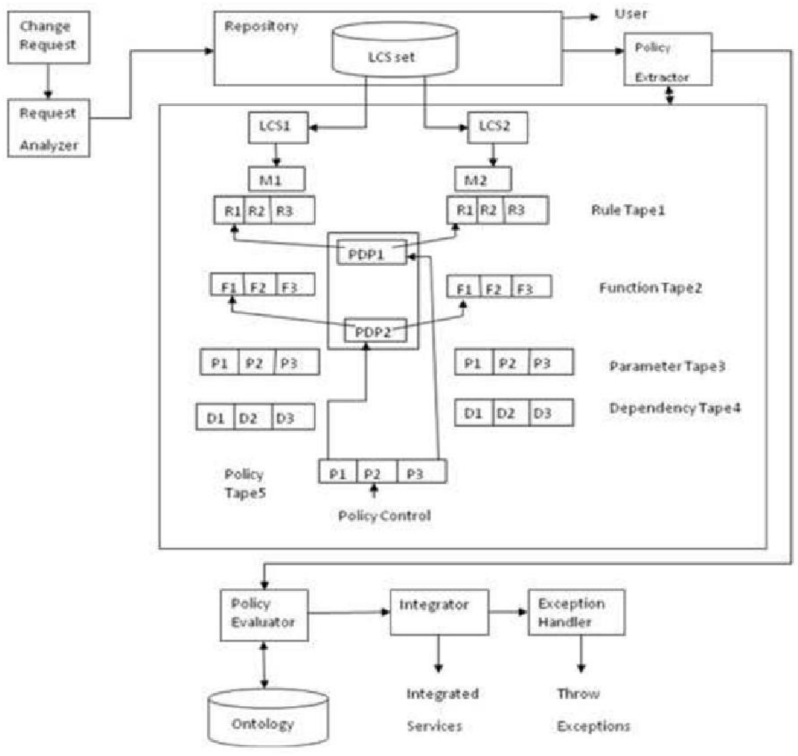


Figure 5.3: Policy Violation Detection using Ontology

c. Policy Extractor

If the available LCS could not fulfill the user request, the services from the already existing services have to be composed to form a value-added service to be offered to the user. The Policy Extractor sends the policies of services to be integrated to the ATMs to detect the business policies and to check for any violations that arise on the policies of the composing services. If there are no violations found then the extractor sends the services to policy evaluator to evaluate the domains of the services being integrated.

d. Policy Evaluator

The extracted policies are fed into the policy evaluator to check the domains of the composing services are similar or not. The repository for Ontology is used in this phase to check the domains of the services. The context information is collected, processed and sent to the composition engine to coordinate with appropriate LCSs for appropriate services.

e. Ontology Repository

Ontology repositories are usually more specific than semantic Web search engines. Their navigation and search interfaces can vary to a great extent. They offer many tools that may be specific to the type of applications for which the repository was designed. A Repository that consists of domain specific knowledge that is required for the Web service and composition system to find the best service for integration.

Apart from these major components there are two other components called service integrator and exception handler. After a clear analysis of the policies and to confirm that there are no policy violations, the proposed system tries to integrate the services that chosen for composition. Policies are checked at all levels namely the rule, function, and parameter and dependency level. At the time of policy violation detection, if there arises any exception, the exception handler component will take care to handle the exceptions. The integration part of the proposed system is further extended to make it a dynamic and a most effective one in the following chapters.

During the process of policy violation detection, there are also possibilities for services to make fake identities about them and increase their credits. There are chances for the fake services to pass the policy violation detection and enter into the integration phase. Hence care should be taken to see that no such fake services enter into integration process. Though the policies of the fake services are desirable, they may cause severe effect on offering service to the customers. Some kind of mechanism has to be followed to identify such problems. The next module concentrates on this issue. A new mechanism called reputation measurement is followed in order to stop the process of integration if a service is found to have a fake identity.

5.6 Experimental Results and Discussion

With the help of the algorithm designed for policy detection using ATM, multiple Turing Machines were simulated. With the same set of LCSs, experiments were conducted to monitor the performance of a single Turing Machine and ATMs. The results proved that the performance of ATM is high when compared to single TM. The number of requests handled and the number of policies detected with violation using Alternating Turing Machine are more when compared to using single a Turing Machine. This is very well plotted in Figure 5.4.

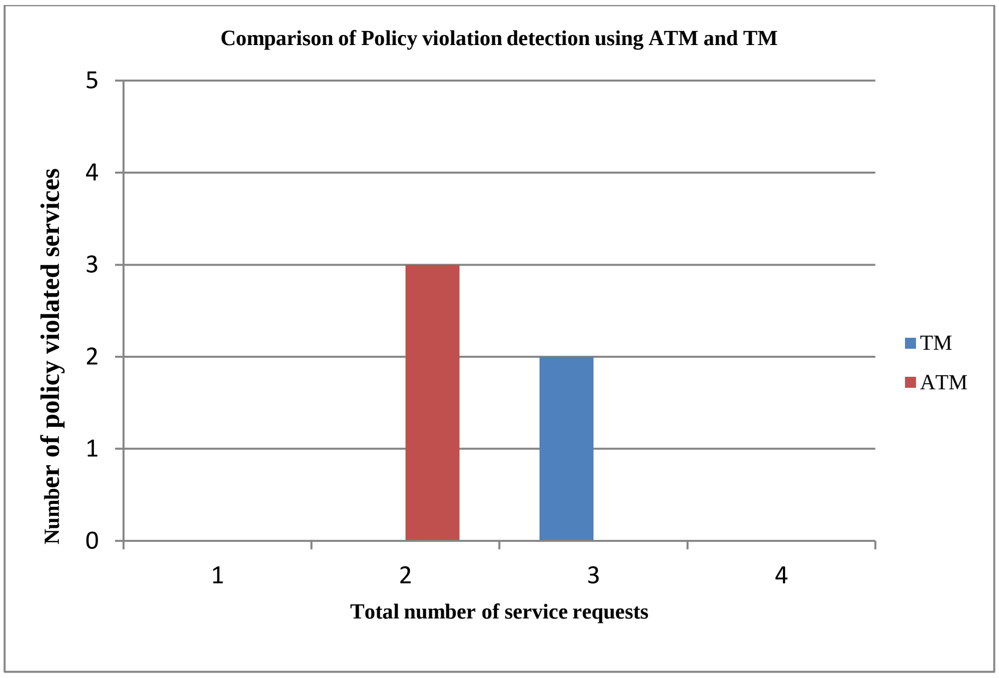


Figure 5.4: Comparison graph of policy violation detection using TM and ATM

Experiments were also conducted to compare the performance of policy violation detection with and without reference Ontology i.e., Domain Specific knowledge of services to be integrated. The same LCS set were used to experiment the result. The results are tabulated as shown in table 5.2.

Table 5.2: Business Policy Violation with reference Ontology and without reference Ontology

Domain	No of Integrating Services	No of Policy Violation Detection	
		With reference Ontology	Without reference Ontology
Travel	10	4	6
Medicine	12	3	7
Hotel	8	3	6
Billing	10	5	6
Security	7	3	5

5.7 Summary

In this chapter, a methodology has been proposed for business policy violation detection in the Change Management Framework based on ontology and the domain of the service requests. Initially the incoming requests are analyzed in terms of its previous incidence, domain and the business logic. In business logic analysis of the request, Alternating Turing Machine is used to detect the policy violation points of the requested and integrates them to CMF if there is no policy violation; otherwise termination of services of those policy violated services is carried out. The experimental results confirmed more the efficiency of the proposed methodology than the conventional techniques.

CHAPTER 6
REPUTATION MEASUREMENT

6.1 Overview

This chapter emphasizes the Reputation Measurement of the services to check whether the service is reliable one to be integrated to the CMF and eventually to progress the development of the framework. The Reputation Measurement of the respective service is done based on the credits collected from the personalities who may be business partners or some other individuals. The risk associated with malicious motivations is that some personalities will provide higher credits for the service which has no future scope that affects the development of the business and this can be avoided by evaluating the credits. The malicious activities considered here are the Ballot Stuff Attack and Fake Identities.

6.2 Reputation Based Business Policy violation detection

The detection of business policy violation in the Change Management Framework (CMF) is the main stage to integrate the service which is requested by another business party or the customer. Hence the service integration has to be done in a manner that is adaptable for future trends in business era. In the previous chapters the work focused on the policy detection by means of Turing Machine which is capable of performing one action at a time such as either acceptance or rejection of services for integration with the policies associated with them. A framework for Business Policy Violation Detection based on the Reputation of Web service by considering into account the malicious intentions is now proposed.

Reputation Measurement

The Reputation speaks to an agreeable impression of the clients in the gathering of individuals about a Web benefit, that is, the notoriety of a given service is an aggregate criticism rating of the clients who have associated with it or utilized the service before. Input rating is the acumen of every client about consumed services. Notoriety could be a solitary esteem speaking to a general assessment or a vector representing a value for each QoS attributes for a Web service, for example, a response time, unwavering quality, and accessibility.

At whatever point a client asks for a support of the proposed framework, a Service Level Agreement (SLA) between a client and a specialist organization is confirmed and afterward the client chooses a Web service that fulfills his/her QoS prerequisites. The service is then consumed. Once the service is devoured by the client, the client reports a feedback rating for the web service about the execution of the Web servicet. At last, the proposed framework gathers the feedback rating and other feedback appraisals from different clients with a Data Collector, ascertains the notoriety (scores). The gatherer at that point refreshes these scores in a QoS repository, and gives the scores while prescribing those services to the clients.

In the proposed methodology, after detecting the policy points the policy violation detection is performed by analyzing the change request in terms of previous incidence, domain as well as business logic of the request. After this analysis the third stage called Reputation Measurement is carried out where the Reputation of the service is calculated to integrate it to the business policy framework on trust basis by considering the malicious intentions such as ballot stuffing attack and fake identities which are intentionally performed to credit the service by giving inaccurate credit labels. By levitating such attacks and based on the Reputation Measurement the service can be integrated to the business policy before thrown to the exception handler and results in a complete dynamic CMF. The functional block diagram of the proposed methodology is depicted in Figure 6.1.

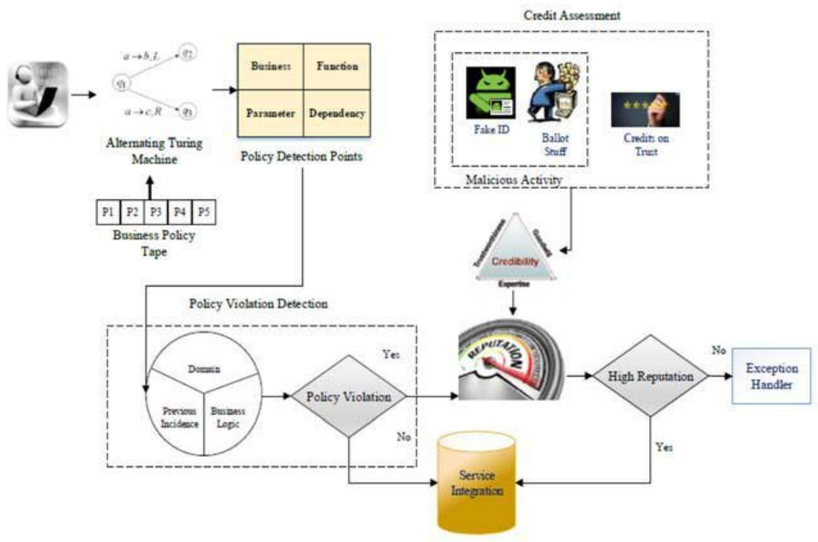


Figure 6.1: Architecture of Business Policy violations Detection with Reputation Measurement

Reputation of Web services is also measured in some existing works like one by Zaki Malik (Zaki Malik et al. 2009) where the Reputation is measured by the service consumer to select the reliable service provider. In the present work, Reputation Measurement is introduced to calculate the Reputation of the Web service before its integration to the CMF. The Reputation of each Web service is measured based on the quality parameters associated with it called Quality of Web Service (QoWS) which includes security, privacy preservation, response time, availability, reliability etc., and by the credits provided by the peer reviewers (personalities).

Normally before integrating the service to the business process, the policy of the newest request will be verified against with the existing defined policies, and if the policies are not matched at different levels it will be thrown to the exception handler. But the incoming Web service request may be of an efficient one coping with future facilities of the clients who are accessing the service from the particular Website and may lead to the improvement of the business. Hence the cross checking of the request is necessary to make the system completely dynamic for the changing environment. Also while checking the quality of the service request the act of malicious intentions also need to be taken into consideration as this may lead to the integration of service that is completely not necessary for the business process.

6.3 Reputation Measurement against Malicious Intentions

In this phase the Reputation of the services in LCS is measured to check whether the service is reliable one to be integrated to the CMF and eventually results in the development of the framework. This is done in a case when at times the defined policies in the framework is the common one and the request for the new service integration will have policies that is different at some level to the existing one but has the future scope that leads to the satisfaction of the users and subsequently the business development. The Reputation Measurement of the respective service is done based on the credits collected from the personalities who may be business partners or some other individuals. The risk associated with this sometimes with malicious motivations some personalities will provide higher credits for the service which has no future scope and results in no development of the business and this can be avoided by evaluating the credits. The malicious activities considered here are the Ballot Stuff Attack and Fake identities, and these attacks are defined in the following section.

6.3.1 Ballot Stuff Attack

This is the attack in which the personality with malicious intention will give more number of credits to a particular service exceedingly over to his limit such that the maximum number of credit level is exhausted. This averts other legit clients to give acknowledgment for the individual service. For this situation the valuable service will lose its opportunity to be incorporated to the CMF. To evaluate the possibility of a Ballot Stuffing assault by methods for produced services, the way the copy Ballots are recognized by the service Providers and the way they are taken care of amid benefit combination ought to likewise be considered.

These processes have to be done when trying to integrate that service with CMF. These attacks can boost the reputation of a malicious node by providing first-class recommendations to the services so that it increases the chance of that bad service to be selected as a part of LCS. This is a form of Collusion Attacks that combines with other bad nodes to boost the Reputation of one another.

6.3.2 Fake ID (Sybil attack)

Sybil attacks have the property that could damage trust management system. In these kinds of attacks end-users have purposely different identities. Therefore they can provide contradictory ratings about the same Web services. The existing approaches depend on random choices to sort out Sybil users and reduce their attacking capabilities. With this attack the identities are created with fake profiles and will provide higher credits for the useless services that are completely violated all the policy levels. This results in the integration of useless service to the CMF. By considering these attacks the valuable services which have to be integrated to the CMF can be determined.

In the proposed work, Reputation Measurement is employed to calculate the Reputation of the Web service before its integration to the CMF. The Reputation of each Web service is measured based on the quality parameters associated with it, called Quality of Web Service (QoS) which includes security, privacy preservation, response time, availability, reliability etc and by the credits provided by the peer reviews (personalities). Thus the Reputation Measurement for the service LCS*pv* from the peer review is approximated using the equation given below:

$$\text{Reputation, } R\left(LCS_{pv}, J\right) = \underset{X \in i}{\wedge}\left(PE_m^x pv\right)$$

Where, PE = Evaluation obtained from each peer review i J = the
entity measuring reputation
m = QoWS parameters

Avoiding the malicious activities in Reputation Measurement credibility of raters is done by considering the attacks such as the Ballot Stuff Attack and the Sybil attack and this is performed as given in the following section.

6.3.3 Raters creditability Evaluation

Generally in some frameworks the trust of the entity from where the service accessed is measured through the feedback systems and in those systems the credits are assumed to be reliable one and in a practical world some malicious activities may be associated. Hence to obtain reliable credits, the evaluation of credits is mandatory and by including this Reputation Measurement where the previous equation may be rewritten as

$$\text{Reputation, } R\left(LCS_{pv}, J\right) = \frac{\sum_{X=1}^{i}\left(PE_m^{X\ pv} * C(x, j)\right)}{\sum_{x=1}^{i} C\left(x, j\right)}$$

Where, c(x, j) = Credibility of the peer reviewer for the service LCS_{pv}.

In credibility evaluation, the credibility of the rater is evaluated with the majority opinion and if the rating provided by the rater is agreed with that opinion then credibility becomes increased, otherwise it will get decreased. However, the rating provided by them will not get discarded. The majority opinion is obtained with the clustering technique and in the existing Reputation Measurement technique k-means clustering method is used. The drawback associated with this is the error value is somewhat higher while clustering the data and hence the system employs K-Median Clustering (KMC) technique. This is explained as follows:

6.3.4 Majority Opinion by K-Median Clustering

The advantage of using KMC is that the error associated with this clustering becomes reduced and the technique will consider the median of the neighbors instead of mean value. The input for the clustering technique is the ratings of the reviewers. The process of the KMC algorithm is detailed as in Algorithm 6.1.

Algorithm 6.1 K-Median Clustering Algorithm

```
Algorithm: Pseudo code for K Median Clustering algorithm
Input: Ratings of reviewers, CR = {crn}in=1
Output: K number of Clusters, {Ck}kk=1
Step 1: Let C01, C02 . . . C0k be the initial cluster centers
Step 2: For the credit crn assign it to the nearest calculated by the following equation:
    Manhattan Distance, $D_M = \sum_{k=1}^{k} \sum_{n=1}^{i} \left| C_k^0 - cr_n \right|$
Step 3: Center updated for every cluster C0k by the median of the points crn within it.
Step 4: Repeat steps 2 and 3 for maximum iterations
Step 5: Return final cluster centers as {C1, C2 . . . Ck}
```

The adjustment of the rater's credibility is done by calculating the Euclidean distance between and the reported rating, and the resulting credibility adjustment is given in the following equation:

$$A(cr_n) = \begin{cases} \text{Modified Reputation, } R_M\,(LCS_{pv}, j) \\ = \dfrac{\sum_{x=1}^{i} \left(PE_m^{x\ pv} * A(c(x,j)) \right)}{\sum_{x=1}^{1} A(c(x,j))} \end{cases}$$

After adjusting the credibility of the raters, the modified reputation of the service is calculated as given in the following equation:

$$\text{Modified Reputation, } R_M\,(LCS_{pv}, j) = \frac{\sum_{x=1}^{i} \left(PE_m^{x\ pv} * A(c(x,j)) \right)}{\sum_{x=1}^{1} A(c(x,j))}$$

From the above equation the reliable reputation of the Web services which are violating the policy levels are calculated, and based on that the decision for the service integration is made. This is discussed in the following chapters.

6.4 Service Integration Decision to Change Management Framework

The modified Reputation Measurement is done for each of the service which comes through the request to be integrated to the CMF and based on the list of measured Reputations the threshold value is defined, called Reputation threshold, R_{Th}.

Then the decision for Web service integration is produced as given in the following equation:

$$R_M = \begin{cases} \geq R_{Th}, LCS_{pv} \rightarrow S_i \\ \leq R_{Th}, LCS_{pv} \rightarrow EH \end{cases}$$

In the above equation, if the Reputation of the service is greater than or equal to the threshold value means it will be considered to be integrated to the CMF as denoted by Si, else it will be labeled as the service *EH* to be thrown to the exception handler. The performance of the proposed methodology can be evaluated in terms of different performance metrics that perfectly evaluate the system. In the proposed work, the performance is evaluated through the parameters called reputation Measurement of the service, Computational Time, and Accuracy. The definition of these performance metrics is also given in the following section.

6.5 Experimental Results and Discussion

The Reputation Measurement is the reputation of the Web services measured using the equation given in the section 6.3.4. Table 6.1 represents the description of service request. This value of Reputation is calculated for the number of services and the evaluated values are given in the Table 6.2 and the Reputation score is plotted in Figure 6.2 for the violated service requests. It is noted that the Reputation threshold is fixed to the value of 10(i.e., R_{Th} = 10).

Table 6.1: Description of the service request

Parameter	Value
Number of services	20
Service ID	1-20
Service Domain	Travel, Airlines, Hotel
Description of the service	WSDL, XML, HTML, UDDI and UML
Number of True Peer reviewers	20
Number of Malicious Peer reviewers	5

Table 6.2: Reputation Measurement for number of services

Request ID	Reputation
1	8
5	11
8	20
12	22
17	23

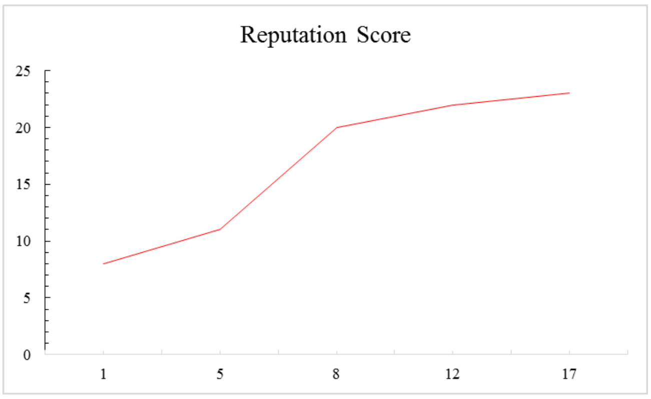


Figure 6.2: Reputation measurement for the policy violated services

The amount of time required to do different processes involved in the proposed methodology is also an important parameter and the result is given in the succeeding section.

Computational Time

This is the time required for completing the process of Policy Violation Detection, Reputation Measurement and Service Integration. These values are also tabulated in Table 6.3 and the values are plotted in Figure 6.3 with a different variety of services from various domains.

Table 6.3: Computational time for policy violation detection, reputation measurement and service integration

Process	Time (seconds)
Policy Violation Detection	2.00
Reputation Measurement	2.58
Service integration	0.507

Accuracy

The last performance parameter to measure the efficiency of the proposed methodology is the accurate calculation of the Reputation for the policy violated services and here the accuracy is calculated as the ratio of total number of services whose Reputation is the correct one to that of the total number of services and this is calculated using the following equation.

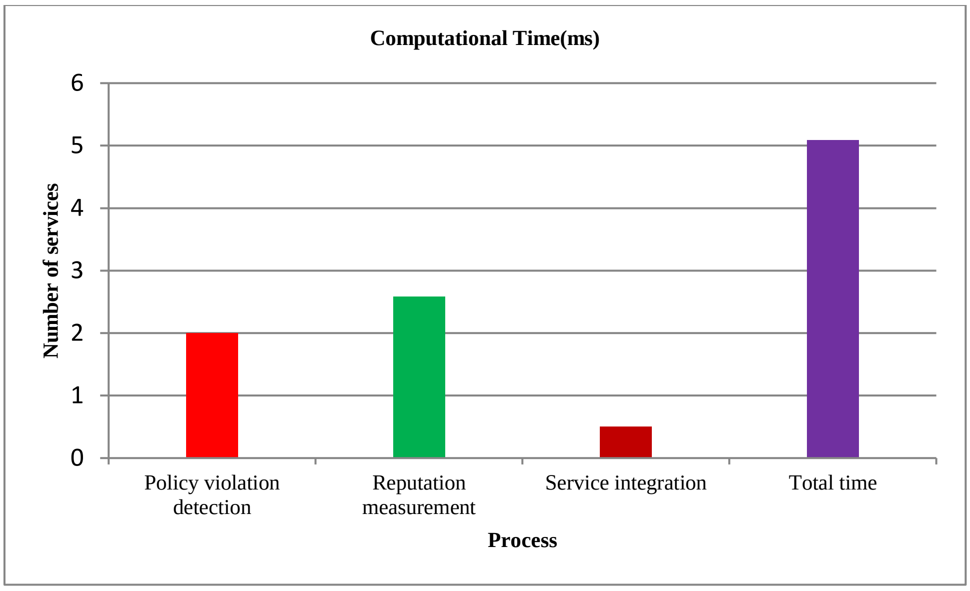


Figure 6.3: Computational Time

$$Accuarcy = \frac{T_R(LCS_{pv})}{\sum LCS_{pv}}$$

Where, the numerator T_R (LCS_{pv}) denotes the number of services whose Reputation is high as predicted and the summation term in the denominator denotes the total number of services whose Reputation is to be measured. The calculated accuracy for different number of services is tabulated in the following Table 6.4 and is plotted in Figure 6.4.

Table 6.4: Accuracy for number of Web services

Number of service requests	Accuracy (%)
5	82
10	84
15	89
20	92

The accuracy of the proposed methodology is given in the form of a graph in Figure 6.4 and from the graph it is evident that when the number of service requests increases, the accuracy of the system also gets increased and it which shows that the proposed system will be able to produce accurate Reputation score even for more than one number of service requests arrived

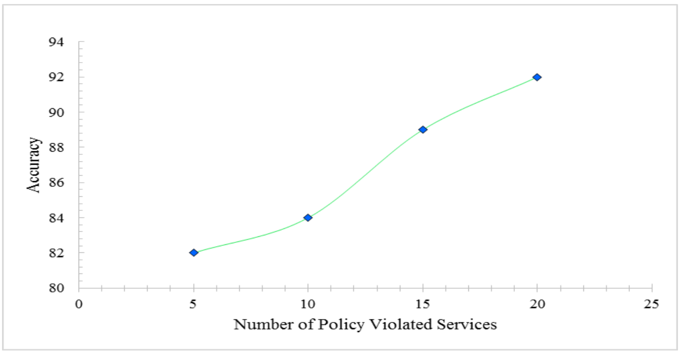


Figure 6.4: Accuracy of reputation measurement

Reputation Measurement is also done by several enterprises that make use of feedbacks and Qos parameters of the services to be integrated. This enterprise level Reputation Measurement uses some kind of filtering techniques to find the trustworthiness of the services. The study shows that these systems contribute less level of prediction of malicious services. The experiments conducted using Reputation Measurement with Raters credibility and K-Median clustering proves that the accuracy of service integration is improved by 92%.

6.6 Summary

This chapter has introduced a methodology for Business Policy Violation Detection in the Change Management Framework based on the Reputation of the service requests. Initially the incoming requests are analyzed in terms of its previous incidence, domain and the business logic. In business logic analysis of the request, Alternating Turing Machine is used to detect the policy violated points of the services and integrate them to CMF if there is no policy violation. Otherwise Reputation Measurement of the policy violated services is carried out. The Reputation of the service requests is evaluated based on the credits provided by the peer reviewers but there may be the possibility that the reviewers may believe in malicious intentions. So the credibility of the provided ratings is verified against K-Median Clustering technique. Based on the calculated Reputation level of the policy violated services, Reputation threshold is fixed and if the threshold exceeds, it becomes integrated to the CMF or it will be thrown to the exception handler. The experimental results proved that the efficiency of the proposed methodology is greater than that of the conventional techniques.

CHAPTER 7
COLLABORATIVE AGENTS FOR DYNAMIC WEB SERVICE INTEGRATION

7.1 Overview

The collaboration process is responsible for the registration of different enterprise resources and also for checking the database present in the system, information sharing and exchanging data by releasing certain information. Integration at this level has numerous difficulties. Expected logic to determine the service integration demand must be situated between the demanding services of different accomplices. To address the above said issues, a Collaborative Multi-Agent based Dynamic Service Integration system is proposed. This system facilitates the process of integrating the service logics automatically.

7.2 Dynamic Service Integration Framework

Service Integration is a procedure that uses a complex interface to put the required business logic in the service complete and to coordinate them formally. It additionally requires confirmation and access control component to check and make utilization of the partner's logic at outline level. To handle and work out the above said tasks, the proposed system employs various agents that could help in the process of integration. The agents are Discovery Agent (DA), Evaluator Agent (EA), Negotiation Agent (NA), Dependency Checker Agent (DCA) and Integration Agent (IA). Figure 7.1 illustrates the overall system architecture for the proposed Dynamic Service Integration.

Schemas that are produced from the business logics extracted from TM are adaptable to supervise. They are typically utilized at property assessment levels. These patterns are then incorporated and created to shape new services. Discovery Agent is the essential component in the framework that looks into the required service logics progressively through its intricate business space situated searching ability. The Negotiation Agent is another noteworthy component in the framework. This operator is utilized for guaranteeing that the requirements existing between business strategies of the teaming up service is likewise affirmed by the approval level to neglect the business logics.

Several constrains may be associated with every service. These constraints may cause dependencies between several functionalities within the service. To analyze these dependencies, Dependency Checker Agent is employed. After these agents complete their tasks successfully, the Integration Agent dynamically integrates the services that are checked for property, business policy evaluation and dependency. To perform the task of Dependency Checking and Business Policy evaluation, the TMs are utilized.

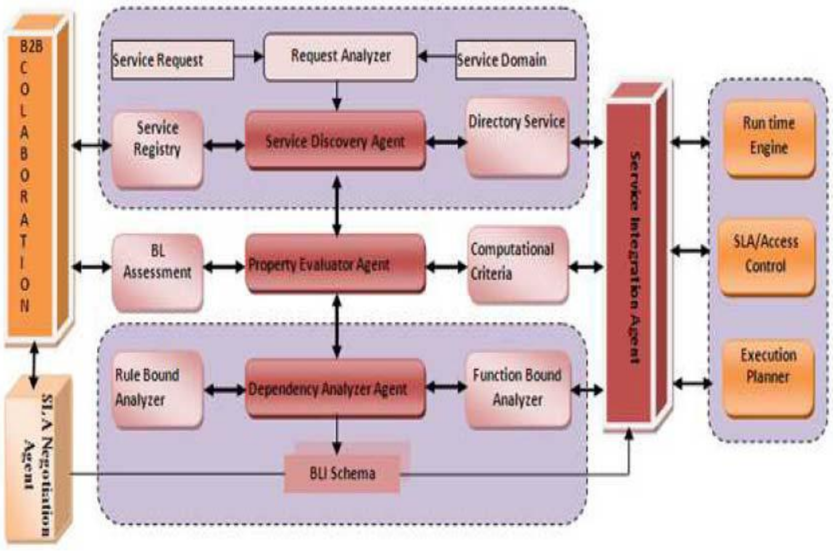


Figure 7.1: Dynamic Service Integration

7.2.1 Discovery Agent

When a user request arrives for service integration from the service request block, it is sent to the request analyzer component in the system. The request analyzer analyzes the domain of request. It checks the domain registry for this process. If the service domain is a registered one, then the request is sent for processing. If the domain of service is not registered, the request is denied and the process of integration is stopped at this level itself. Once the registration confirmation process is over, the directory service is triggered. Here the locations of the services are identified and then its corresponding service logics and schema are extracted as a single set. The Business Logic (BL) holds the rules, functions and parameters associated with the services. The BL set will then be given to the Evaluator Agent to evaluate the properties. When the schemas of all the services are extracted, then the level of collaboration is checked in a B2B collaboration component.

In future when the request of the same type enters the system, the BL schemas are not sent to the Evaluation Agent or the Dependency Checker Agent. It is directly sent to the Integration Agent. This is because the knowledge base stores the details the details that are collected from the DCA and the EA regarding the evaluation results. When the request was denied previously due to any accessibility issues then, the whole process of integration is stopped and therefore the execution time is reduced drastically.

There may also be situations where service requested be found reliable but, the service requester may be an unreliable one. Also there might be policy violations among the requester and the providers. In such cases, they are verified at the primary level be the DA itself. Only when the collaboration process is satisfied, the process of integration begins.

7.2.2 Evaluator Agent

There may be defects in the parts of business policies extracted from several sources of services. The Evaluator Agent investigates such defects and verifies the functional and non-functional assessment of the services that may be needed to decide whether the integration logic fulfills user request absolutely. Functional assessments can be made with some important properties that include computability, completeness and configurability. These properties guarantee the effectiveness and execution of the logic. This operator has the synergistic nature to speak with other service providers and occasionally informing the required properties. Other key properties like Computability and Completeness are also verified at Rule and Function level. Turing Machines are employed at this level. The logics are first converted into its equivalent FOL before processing.

7.3 Property Evaluation using PE Agent

In the process of service integration, the proposed system determines the important properties such as completeness, configurability and computability with the help of several Agents in the framework. This is done to make sure that no run time issues occur in the logics on the flow and after process of integration. These properties are evaluated and the results are discussed in detail in the following sections.

7.3.1 Computability

In spite of the fact that the framework coordinates the service logics progressively, it is basic to confirm that created complex logic is calculable inside a period restrict. In order to make the business logic and the business function to be a computable one, they must be handled with clearly defined and a finite length instructions. The complete description of the functions to be performed is therefore present in the corresponding function's finite program.

Consider if the business function is represented in a k-tuple which is denoted by x in the domain of f, then the function should terminate after a finite number of discrete steps and should produce the function f(x). The process of business function proceeds with a sequence of steps following a accurate rule to denote what has to be done at each step of the calculation. It is a defined rule that only finite steps should be carried out and the value of the function should be returned. If the function that was represented in k-tuple denoted by x is not in the domain of f, then the function executes continuously without halting and caused sensitive exceptions. There are also possibilities that it might get stuck at any point with because of some basic type of exception. At any point of time the system must not act as if it has produced a value for f at x. Therefore, at any point of time if a value for f(x) is found, it should be the correct one. Only then the business function will be completely computable. To perform these tasks, the system uses a finite amount of storage. Even if the storage exceeds, the space can be extended. Additional storage space can be given to the function whenever there is a need.

7.3.2 Completeness

The completeness property is utilized to demonstrate that the business rules are finished as for its related functionalities and furthermore they are semantically substantial. Just when the arrangement of guidelines that are related with that of the business logics are finished and legitimated semantically, targeted business logic is said to be complete. The business logic set is a four Tuple set that has rules, functions, parameters and dependency connection which are identified with each other. Any semantically valid contention can thus be caught by the formal evidence. The choice of rules for conclusion is to be made in earlier the procedure of integration over the logic set. This procedure is effectively performed when the framework is in a static stage. Be that as it may, at runtime, a few issues identified with the info/yield parameter mapping, calculations done iteratively and restrictive variable changes are to be overseen with no naming clash. The run-time condition is intended to deal with the exceptions when these factors are adjusted by guaranteeing the sort, degree and scope of the factors which are subject to change. The procedure ends up convoluted in light of these confinements need to ensure the soundness of the rules and due to the status of the factors.

Theorem 1. The Business Logic ‘BL’ is complete if and only if the associated rules, functions, parameters and the dependency relations are complete. Here, ‘BL’ is a 4-Tuple Business Logic Set. B L = (R, F, P, D), Where R is the Rule Set, F is the Function Set, P is the Parameter Set and D is the Dependency relation.

Proof:

Parameters are complete if the input and output variables are bound with distinct values.

Info Parameters < I1, I2, I3, .. In> are bound with discrete values, at that point return Boolean esteem (True). By experiencing the limit check work B (p), a condition for Business Logic is encircled as underneath:

$$B(p) = \begin{cases} 0 \text{ If } I_i \text{ is false for all } i <= n \\ 1 \text{ If } I_i \text{ is true for all } i <= n \end{cases}$$

Every one of the parameters including the temporary variables is acquired and included it in the parameter list. Let‘t’ be the temporary variable. This variable is substantial just if f(x) is in bound with f(x). This is the process used for finding the relationship between the business variable 'x' and temporary variable ‘’t’. To process the mapping between the brief variable't' and the business factors x, y and z, we outline a condition:

$$f(t) = \int_0^n [\, f(x), f(y) \,] \in t \rightarrow P\,(Z)$$

Subsequently the parameters are finished if the point of confinement check for all parameters including the temporary variable is seen to be valid. For every brief variable't' there exists a mapping limit f(x) related with the business parameter 'x', by then t is useful else it is pointless A Finite State Machine is developed for the parameter list by thinking about the successive connection from the source to ending purpose of a Business Logic. This procedure is rehashed for finding new business arrangement. At each state the reproduced FSM stores the parameters of the Business Policy and the present value.

7.3.3 Configurability

Configurability is a property for establishing and maintaining consistency of performance of a composed service throughout its life time. Before the process extracting and developing service logic from the services to be composed, the resources required to build them have to be verified. They can be executed only if the service logic is available in the system. On the off chance that specific asset isn't accessible, it can be then hunt down an elective way and do changes as needs be. Framework receives this change just when it is reliable. Business Logic is an arrangement of code fragment 'cs' that holds an arrangement of framework data identified with design and support. The logics related to configurability are also contained in the Business logic points. It is written with a set of methods that include drivers, connectors, resource types, a set of credentials, access methods, etc. Attempting to do any modification in the system logic, a proper verification should be done such that it does not affect associated business logic. It should also be checked whether updating the business logic is trustworthy or not. In order to do any modification, system logic is initially expressed in the form of First Order Logic.

After modification, the modified logic is also expressed in FOL and compared with the original one. Configurability is achieved only when original FOL and the modified FOL are same. Consider CL and CL′ to be the configuration logic before and after modification. Verification is done to see if FOL (CL) = FOL (CL′). The business logic should also be checked before and after modification. Consider BL and BL’ to be business logics before and after modification. Verification is done to see $BL \cap BL' = \varphi$. After each modification, the code segment is verified to examine whether the update is useful or not. Even the Database connection type should be of the same type and not with network connection different type. If f(x) is the original resource in the CL and g(x) is the modified one, then it is verified that $f(x) \cap T(x) = c$. Value c is some constant. Table7.1 illustrates FOL conversion for configurability and Algorithm 7.1 describes the working of configurability in the integration process.

Algorithm 7.1 Algorithm for computing Business Functions

Algorithm PrimitiveBusinessFun (BusinessLogicSet)

Input: BusinessLogicSet [L]

Output: Total or partial evaluation for computability

Method: PRF to confirm that logic is computable or not

```
//ßr [ ] : Business Rules (ßr [ ] = ßf [1..n])
//ß f[] : Business Function(ßf[] = Ißf[] Cßf[])
//Ißf[] : Initial Business Function
//Cßf[] : Composite Business Function
//Pßf[] : Primitive Business Function (Pßf[] = Pf[Ißf[],Cßf[]])
Begin
(i) Input the BusinessLogicSet (L)
//BusinessLogicSet is a 4-tuples consisting of Business Rules, Functions associated with the BusinessRules, Business Parameters and Dependency Relations that exist between the first 3-tuples.
L= R, F, P, D
        L = ßr1, ßr2...ßfn, ßf1, ßf2...ßfn, P1, P2...Pn, D1, D2...Dn
(ii) Evaluation of InitialBusinessFunction (Ißf)
For each BLn[where n > = 1] in BL
:Ißf(n) = Calculate Initial_function(BLn)
If BLne[Zßf (y ) , Prßf(y), Ußf (y),  Pßf (y),Idßf(y) , Sßf (y),, Cßf
(y) ] then Ißf(n) = BLn  where in BLin is said to be computable
within the polynomial time.
//Initial Function contains Zßf (y ) = 0 | Ußf (y) = 1 | Ißf(y) = y |
Psßfnn (y1, y2, y3, ... yn)
= yn | Sßf (y) = y + 1 | Pßf (y) = y – 1 | Cßf (y)
= y’ End if
End loop
(iii) Evaluation ofComposite Business Function Cßf
For each Ißf(j) [ where j > = 1] in Ißf(j)
: Cßf(j) = Composition is applied over InitialBusinessfunction Ißf(i)
If Cßf (x,y ) = Ißfh (Ißfg1 (x,y ), Ißfg2 (x,y)) from defined functions Ißfg1,
Ißfg2, Ißfh. Where Ißfg1, Ißfg2, Ißfh are generated InitialBusinessFunction
from the first phase.
Cßf(i) = [Ißfg1, Ißfg2, Ißfh] where in Cßf(i) is computable within the
polynomial time.
 End if
End loop
```

```
(iv) Evaluation of  PrimitiveBusinessFunction
(Pßf) For each Ißf(j) [ where j > = 1] in Ißf(j)
: Pßf(j) =Recursion is applied over InitialBusinessfunction Ißf(j)
If Pßf (x, 0) = Ißfg1 (x) and Pßf (x, y + 1) = Ißfh (Ißfg2 (x, y),
Ißf(x,y))
From the initial business functions  defined earlier Ißfg1,
Ißfg2, Ißfh
Where Ißfg1, Ißfg2, Ißfh will be generated by InitialBusinessFunction
Rßf(i) = [Ißfg1, Ißfg2, Ißfh] where in Pßf(i) is computable within the
polynomial time.
End if
End loop
(v) Calculate Computability factor
If (Input_BusinessLogic (BL)) is different in all the three phases then BL is totally computable
Otherwise  BL  is  partially
computable
End if
End
```

Table 7.1: First Order Logic Conversion

RULE	CODE	FIRST ORDER LOGIC
R1	get(String account) Select * from withraw where account no like {$Keyword}	∃a[account(a)]→[account (a)] Λ [account_no(a)]

R2	get (user id,pin no)
	if(user.equals(user id) &&pass.equals(pinno)){
	out.println("Login Successful");}
	else
	out.println("Login Failed,Please try Again") } }
		∃y[userid(y)] Λ z[password(z)]

7.4 Accessibility through Negotiation Agent

Approval and validation are basic issues as these are the procedures of sharing business logics among the accomplices. So availability is a basic issue as it is utilized to depict how much the requestor can get to the business logic. Before getting into the incorporation procedure, contract made between the community undertakings ought to be confirmed, appropriately framework must enable them to utilize. Verification and access control between the service accomplices are controlled by NA.

It additionally decides the openness level of every last business tenets, capacities and parameters in relationship with their business arrangements. Keeping in mind the end goal to play out this, FOL and FSM must be built and the FOL (First Order Logic) for the removed code portion (slice), through which the subject, asset and condition bound with the business logic are effortlessly recognized and their relating access control highlights are mapped.

Algorithm 7.2: Algorithm for Accessibility

```
Algorithm Accessibility (BusinessLogic, AccessPoint, AccessControl)
//  Input Business Logic is divided into set of Rules(R), set of Functions(F), set of Parameters(P) and set of Dependency Relation(R),
//  Here the BL consist of 4-tuples (R,F,P,D)
Each Rule is associated with Access Point for the business rule code segment and this confirms the accessibility of the modifiable business logic that is based on the Access Control Type w.r.t. business policy
Begin
For each AccessPointDef
For j = 1 to k // where 'k' is the no. of access point Devide[i] =
DoCodeSegment(BusinessLogic, AccessPoint)
For each slice in Devide[n] do
//convert the slice into first order logic (FOL)
  //  FOL[i] =ConvertFOL (slice)
//AccessControlType is of different categories such as subject, resource and environment ObjectU[] = Subject(FOL[j])
//Resources may include Database, File System, Commerce Server.
 ObjectR[] = Resource(FOL[j])
//  Environment are Data center, Organization, Business Logic Server.
ObjectE[] = Environment(FOL[j])
Acc_Cont_Type = {ObjectU, Object R ,ObjectE}
// Finite State Machine is constructed for the complete business logic
Construct_FSM(FOL[i])
Begin
Let 'M' be the Finite State Machine that is a 5-tuple set
M = (Q, ∑, ä, q0, F)
Where
Q is the finite set of states which are stored in the form of converted FOL.
Σ  is a finite set of symbols, the input alphabet that form the access control specification.
ä: Q X ∑ Q is a transition function that makes the machine move from one state to another
q0ºis the initial state which is the entry point of FOL[i], where i = 1.
```

```
F ⊆ Q is a set of final states. Exit point of FOL[i], where i = n.
// The fact that ä is a function implies that every state has an outgoing transition for each
member of ∑. An extended transition function can also be defined as ä* as ä*: Q X ∑*
And transition function such as ∑ - r1 represents negation of the input access → specification
for r1.
A State is created for  each FOL[i] with the slice
A State Transition Table (STT) is constructed  for the recognized states and transition between
the states are maped.
End
Do{
Signal→DoTraversal(FSM, FOL[i]
 AccessControl, AccessControlType, STT)
} While (Not FinalState(FSM) && FOL[i]!=NULL)
If Signal returns 1 then Acc= TRUE for all set of Access Point
Else
Acc=FALSE for the Current Access Point ‘x’ with exception.
  End
```

FOL examines the code portion altogether and restore the Boolean value 1 if all related standards of the business logic meets the access control strategies. Convert_to_FSM inserts this FOL as limit in every area of TM since each state in TM is allocated for the business control as demonstrated by their execution arrange. The advance between two states happens exactly when the FOL related with the state returns 1.In the event that TM ends inside a response time, the requestor is approved to get to the specific piece of logic which is distinguished for incorporation. Table 7.1 illustrates the FOL conversion of business rules. Algorithm 7.2 represents the algorithm for Accessibility.

7.5 Dependency Checker Agent

Dependency checking is fundamentally used to investigate the dependency relation between the business rules, functions and parameters in the business logic that were extricated from the TM. DC Agent does this assignment successfully by teaming up with different specialists in the framework. As an initial step, it classifies business logics into rule set, function set and parameter set. Next, it creates Business Logic Schema as its tag sort business rules, functions and parameters and it delineates the spill out of one business lead to different business control, capacity, parameter and the other way around. Turing machine determines the transition and the dependencies between every business rules, function and parameter and the other way around. This section details the mapping between business logic and BL schema. An example of Credit Card Validation logic is shown in Listing 7.1.

BL schema that is generated by DC Agent is shown in Listing 7.2 which is a complete translation of logic in Listing 7.1 and describes the flow of above logic absolutely. The tags of BL development arrange the logics into business rules, functions and parameters. It furthermore depicts the degree of each business rule, function and parameter and dependence with other. It is given as contribution to the TM Simulator which repeats the TM to separate the advance starting with one segment then onto the next and produces BL outline fittingly. The BL design delineates the dependence between each grow skillfully with their record.TM simulation for BL schema is represented in Listing 7.2 is imparted in Figure 7.2 and BL pattern corresponding to the TM is generated in Table 7.1. The multi-tape Turing machine appeared in Figure 7.2 is developed by different parts (rule, function and parameter) at various levels so as to discover the blunders effectively and investigate the logic proficiently. It likewise examines the reliance and watches the rightness of the business rationale. The request of execution of business logic can be controlled skillfully by the business logic design which is appeared in Table 7.2.

```
importjava.sql.*;
importjava.util.*;
public class creditcardvalidation implements creditcardvalidIF{
        public String validation(String cardno, Date expdate){
                String status="";
                //Validate card length
                if (cardno.length() < 13 || cardno.length() > 16)
                        status+="Invalid Cardno: Out of range";
                  // Verify Expiry date
                  Calendar curdate = Calendar.getInstance();
                   Date todate=curdate.getDate();
                if(todate>expdate)
                        status+="Sorry! Credit card expired";
                   //*** Validatecardno
                intval=0,val1=0;
                for(int i=1;i<=cardno.length;i++)
                        if(i%2==0)
                                val+=Integer.parseInt(cardno.charAt(i));
                        else
                                 val1+=Integer.parseInt(cardno.charAt(i));
                        if(((val+val1)%10)!=0)
                                status+="Invalid creditcard";
```

Listing 7.1: Business Logic for credit card validation

With the assistance of transition capacity of Turing machine, the DC agent dissects the dependency that exists between the rules, functions and parameters and furthermore inspects the accessibility property. Figure 7.3 demonstrates the portrayal of capacity and BL pattern ordering in memory together with the BL mapping. The data structure used here is stack joint with linked list. Business logic, rules, functions and parameters are secured freely in each stack as showed up in Figure 7.3. Once the required service is arranged in charge document, it sends the relating BL design in the repository and takes after the isolated pieces. These parts toward the end joined to outline the BL Schema for the perceived business rules.

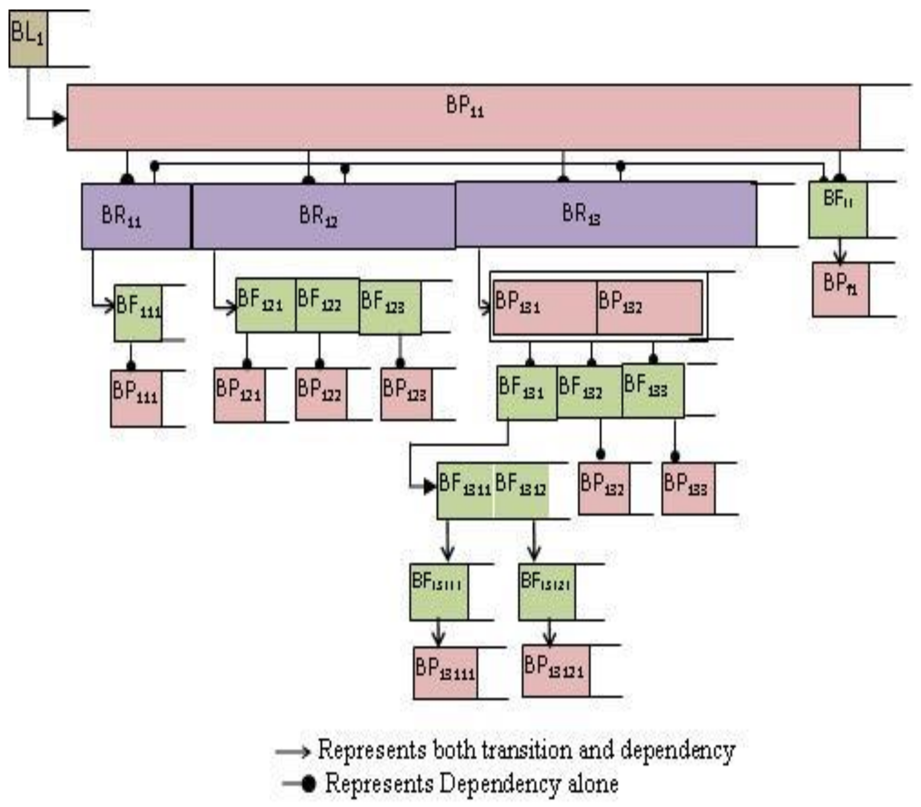


Figure 7.2: ‘Credit card Validation ‘using TM

For example, if the demand is to 'build up an administration for supporting charge card no', the Discovery Agent (DA) parts up the demand into single separate request and finds related services for every last one of its requestors. With the assistance of NA, it monitors the support level of each found services. From the quick overview of the found services, DA uncovers a right one to process the demand. In like way it recognizes the quick overview of business gauges to process the demand. The requested service can be worked from the above integration plan of exercises through the Business logics related with Business Rule BR13. DA looks into reliance between the required Rule BR13 with different business rules, business functions and parameters and finds through BL configuration set away in memory.

The DCA first recognizes the BL pattern associated with separated business rules to process the request. Thus, it discovers the relevant bits of code and finds coordinating schema each symbol in the pattern. It creates isolate BL schema with the followed tags in the current schema.

Table 7.2: BL Pattern for the above Turing Machine

$S \rightarrow BL_1 \mid BL_2$	$BR_{12} \rightarrow BF_{121} \mid BF_{122} \mid BF_{123}$	$BF_{1311} \rightarrow BF_{13111}$
$BL_1 \rightarrow BP_{11}BR_{11} \mid BP_{11}BR_{12} \mid BP_{11}BR_{13}$	$BF_{121} \rightarrow BP_{121}$	$BF_{13111} \rightarrow BP_{13111}$
$BR_{11}\ BR_{12}\ BR_{13} \rightarrow BF_{11}$	$BF_{122} \rightarrow BP_{122}$	$BF_{1312} \rightarrow BF_{13121}$
$BF_{11} \rightarrow BP_{f1}$	$BF_{123} \rightarrow BP_{123}$	$BF_{132} \rightarrow BP_{132}$
$BR_{11} \rightarrow BF_{111}$	$BR_{13} \rightarrow BP_{131}BP_{132}BF_{131} \mid BP_{131}BP_{132}BF_{132} \mid BP_{131}BP_{132}BF_{133}$	$BF_{133} \rightarrow BP_{133}$
$BF_{111} \rightarrow BP_{111}$	$BF_{131} \rightarrow BF_{1311}BF_{131}$	$BF_{13121} \rightarrow BP_{13121}$

BL schema generated for credit card validation is shown in listing 7.3. As the schema and business logics are firmly coupled, it determines business logic associated with the separated pattern and builds up logic. Presently with BL design it plays out a switch procedure to develop Turing machine for this new logic as it is important to follow the bugs and examine effectiveness of the logic. With the uncovered BL designs in rule, function and parameter level, it reveals the portion related with them and new TM is mimicked with the identified section. Locating required portion for the spotted pattern and newly constructed Turing machine for our request is shown in Figure 7.3.

Several new patterns are generated with the extracted business logic. These patterns are fed into the Turing Machine. The TM examines these patterns. One of the amazing features of the Turing Machine is the pattern matching ability. Given any pattern in the business logic, it examines it by constructing a transition table. When there are different states in the transition table and the system halts at any one state, then the pattern is said to be recognized. An XML representation of the Business Logic Schema is represented in the following sections.

```
<?xml version="1.0" encoding="UTF-8" ?>
-<BL name=”validation” includes="cardno,expdate"includetype="xs:String,xs:Date"excludestype="xs:String"
id="M1">
 <BPtype="String" name="status"minoccurs="null" id="BP1" />
-<BR name="Validate length" id="BR11">
-<BF id="BF11" type="if" test="cardnogt 13 and cardnolt 16">
 <BPname="status" value="Invalid cardno: Out of range" id="BP11" />
 </BF></BR>-<BR name="Verify Expiry date" id="BR12">
-<BF type="func"accessmodeule="getInstance()" use="calendar" id="BF121">
 <BPname="curdate" type="Calendar" id="BP121" />
 </BF>-<BF type="func"accessmodule="getDate()" use="curdate" id="BF122">
 <BPname="todate" type="Date" id="BP122" />
</BF>-<BF type="if" test="todategtexpdate" id="BF123">
 <BPname="status" value="Sorry! Credit card expired" id="BP123" />
</BF> </BR>-<BR name="Validate Entered digits" id="BR13">
 <BPtype="int" name="val"minoccurs="0" id="BP31" />
 <BPtype="int" name="val1"minoccurs="0" id="BP32" />
-<BF type="for"minoccur="1" step="+1" test="i le cardno.length" name="BF131">
-<BF type="if" test="i mod 2" id="BF1311">
-<BF type="func" name="In" accessmodule1="charAt()"accessmodule="parseInt()" use="cardno"
id="BF13111">
 <BPop="add" par1="val" par2="In" name="val" id="BP13111" />
 </BF> </BF>-<BF type="else" id="BF1312">
-<BF type="func" name="In" accessmodule1="charAt()"accessmodule="parseInt()" use="cardno"
id="BF13121">
 <BPop="add" par1="val" par2="In" name="val1" id="BP13121" />
 </BF> </BF> </BF>
-<BF type="if" test="va1+val1 mod 10" id="BF132">
 <BPname="status" value="Invalid cardno" id="BP1321" />
 </BF> </BR>
-<BF type="if" test="status=null" id="BF11">
                              <BPname="status" value="Card valid" id="BPf1" /></BF> </BL>
```

Listing 7.2: Business Logic Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
 -<Implementation>
  <packagename="sql.*" property-access="true" id="p1" />
  <packagename="util.*" property-access="true" id="p2" />
 -<class name="creditcardvalidation" implements="creditcardvalidIF" id="c1">
 -<validation includes="cardno,expdate"includetype="xs:String,xs:Date"excludestype="xs:String" id="M1">
  <BPtype="String" name="status"minoccurs="null" id="BP1" />
 -<BR name="Validate Entered digits" id="BR13">
  <BPtype="int" name="val"minoccurs="0" id="BP31" />
  <BPtype="int" name="val1"minoccurs="0" id="BP32" />
 -<BF type="for"minoccur="1" step="+1" test="i le cardno.length" name="BF31">
 -<BF type="if" test="i mod 2" id="BF32">
 -<BF type="func" name="In" accessmodule1="charAt()"accessmodule="parseInt()" use="cardno"
        id="BF33">
  <BPop="add" par1="val" par2="In" name="val" id="BP33" />
      </BF> </BF>
 -<BF type="else" id="BF34">
 -<BF type="func" name="In" accessmodule1="charAt()"accessmodule="parseInt()" use="cardno"
        id="BF35">
  <BPop="add" par1="val" par2="In" name="val1" id="BP34" />
      </BF> </BF> </BF>
 -<BF type="if" test="va1+val1 mod 10" id="BF35">
  <BPname="status" value="Invalid cardno" id="BP35" />
      </BF> </BR>
             </validation>
```

Listing 7.3 BL Schema for validating credit card no

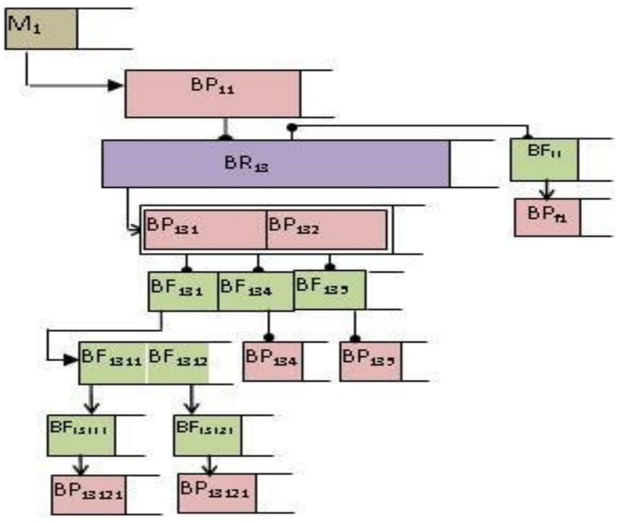


Figure 7.3: New Turing Machine for the extracted logic

7.6 Pattern Recognition using Integration Agent

In the wake of discovering the required logics with its needy sections and examples to incorporate, the services are coordinated. Here Integration Agent achieves this assignment through the demand organized by DA and TM built by DC Agent. The services and the providers are produced by means of different programming platforms, for example, Java Web Service Developer Pack (JWSDP), Tomcat Web server, BPEL Engine and with Jade module. At first, Integration Agent shapes found out logics into a total service and convey the formed service into Tomcat Server joined with JWSDP. At that point utilizing BPEL Engine, it makes the created services in an example perceived by DC Agent. Four conceivable joining designs are examined beneath in detail with delineations. In the wake of coordinating the services, with the assistance of Evaluation Agent, it looks at whether the incorporated service is productive one. Assuming this is the case, it sends the service into the server through Runtime Engine and produces WSDL document. Yield Generative Power (OGP) of the coordinated administration is likewise found and showed. The OGP decides if there is any adjustment in the useful or non – useful viability of the business logics when integrated. The OGP of the incorporated logics in spite of the example utilized for joining ought to be continuing as before or should change in a positive way after combination. The incorporated logic ought not to confront any misfortune either practically or non-practically. This is resolved and shown underneath for each example. Comparability measure is likewise given along OGP. Similitude measure gives the quantity of service with careful attention on of properly when integration is processes with no exemptions. The similarity measure amid our test is around 95% to 96 % which is appeared in the diagram for each example.**7.7**

7.7 Closure Properties of Integrated Services

7.7.1. Union

If BL1 and BL2 are two business logics and both BL1 and BL2 are accepted by TM M, then BL1UBL2 should also be accepted by M. Hence we say the integrated services are closed under Union.

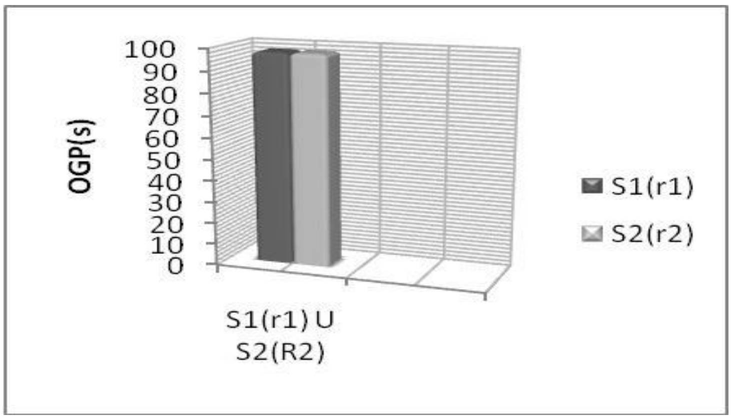


Figure 7.4: OGP of Integrates Services

7.7.2 Composition

Let ô (BL) be the OGP of composed logic L under composition where BL= $BL_1 \circ BL_2$ such that ô(BL) = ô($BL_1 \circ BL_2$). At the end we say when two logics are closed under composition then the OGP of BL_1 is mapped into BL_2 in terms of their rules, functions or parameters.

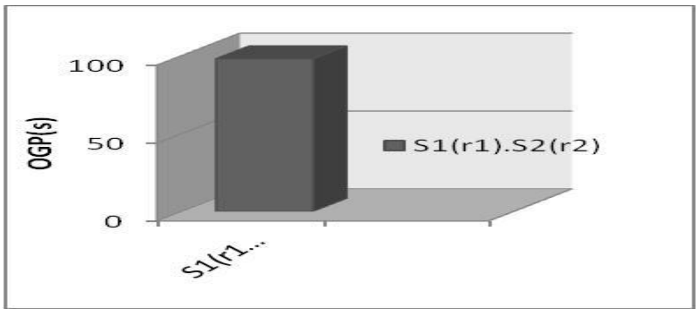


Figure 7.5: OGP of Composed Services

7.7.3 Substitution

Let BL = (R, F, P, D). For each Rule r_1 in ∑ , there is L_{Rule1} that includes the rule, functions, parameters and dependency relation associated with rule1 in L. The new logic set BL'=(R', F', P', D') which includes the substituted logic in terms of rules, functions and parameters, where ô(BL') be the output generative power of integrated logic BL' under substitution.

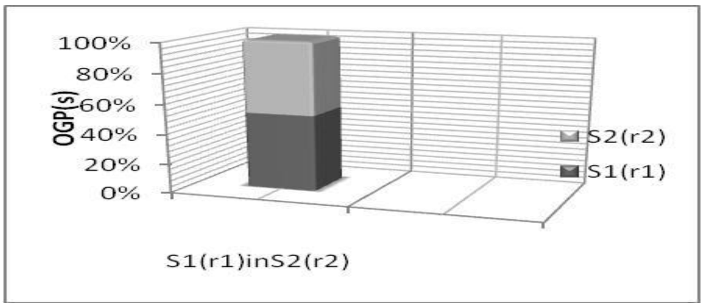
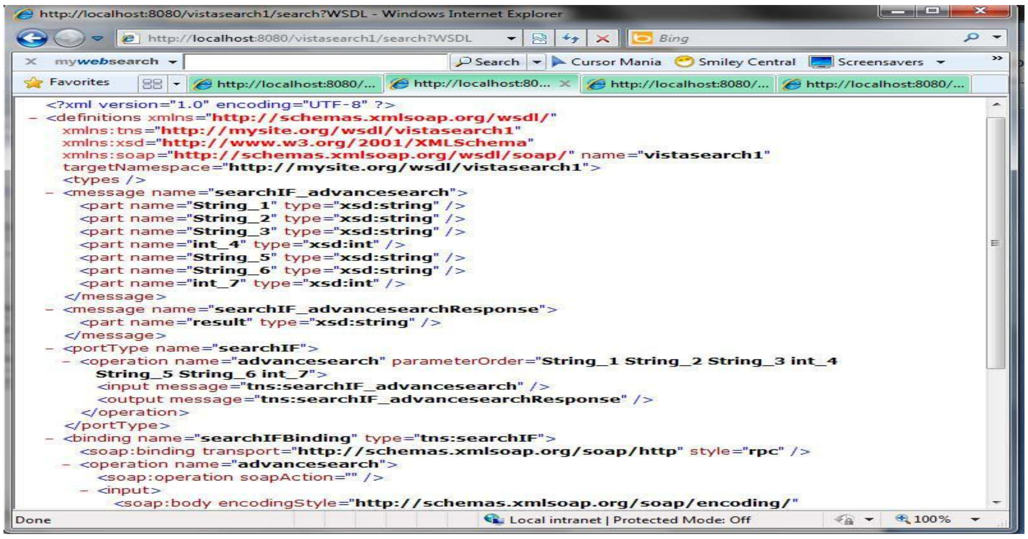


Figure 7.6: OGP of Substituted Services

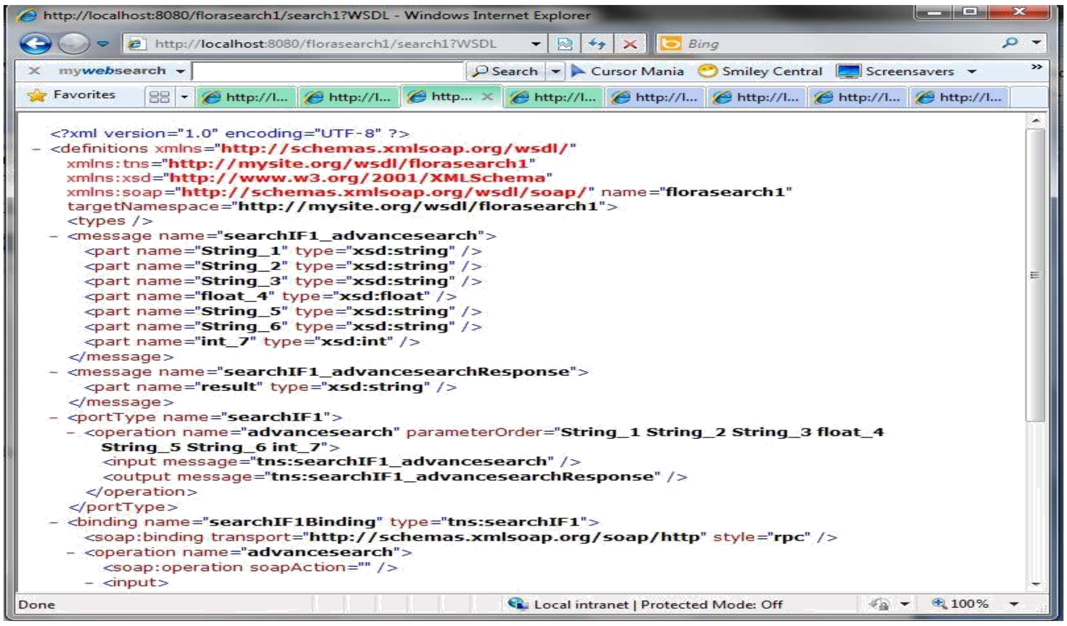
7.8 Experimental Results and Discussion

A service integration demand needs search service which seeks both by content and record. Give Vista a chance to seek be a book look benefit that contains 'look by record compose', date of distribution and search a business rules. Thus, let Flora seek is item search service that contains business principles to 'seek by content kind', cost and brand name. Consequently for the request received, the logic from the Vista service and Flora services must be considered. At first, business logic for the required standards are extricated and created as isolated services. WSDLs of the developed services are shown in Figure 7.7 and Figure 7.8. Subsequently, FSM is constructed to interpret the flow of logic. Then the services are integrated using union and the results are shown in Figure 7.8. Similarity measure between requests is shown in Figure 7.9.


http://localhost:8080/vistasearch1/search?WSDL - Windows Internet Explorer
http://localhost:8080/vistasearch1/search?WSDL
Bing
mywebsearch
Search
Cursor Mania
Smiley Central
Screensavers
Favorites
http://localhost:8080/...
http://localhost:80...
http://localhost:8080/...
http://localhost:8080/...
Done
Local intranet | Protected Mode: Off
100%

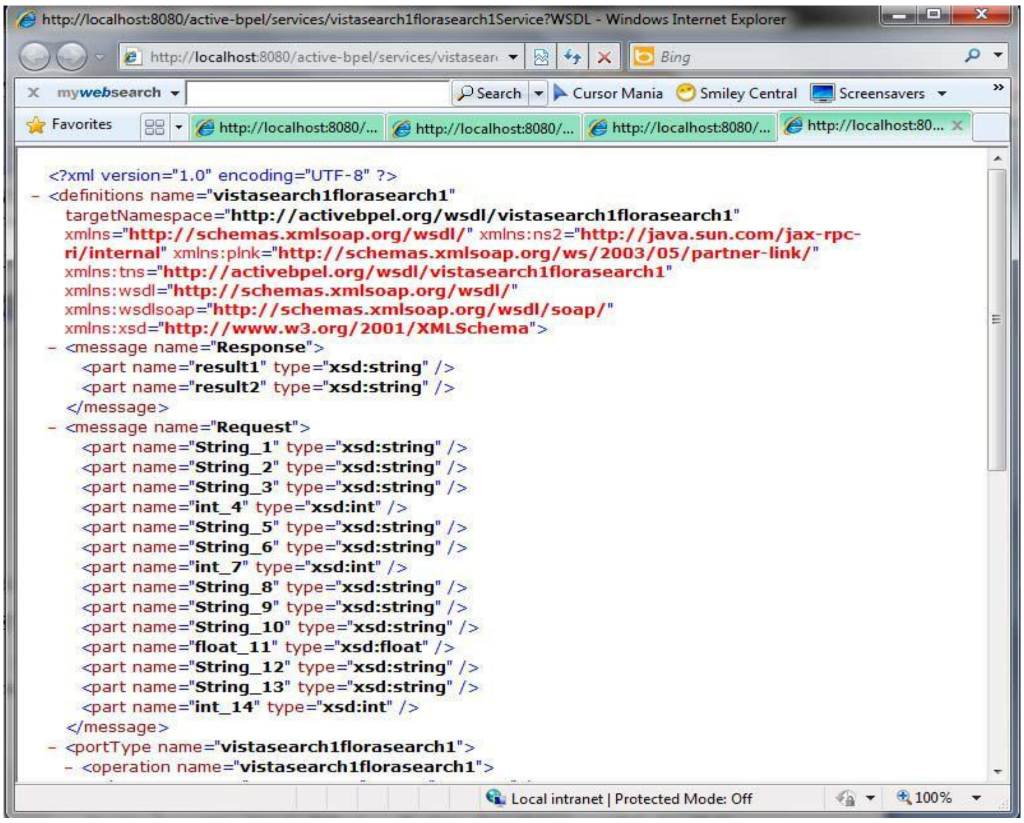
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://mysite.org/wsdl/vistasearch1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="vistasearch1"
    targetNamespace="http://mysite.org/wsdl/vistasearch1">
    <types />
  - <message name="searchIF_advancesearch">
      <part name="String_1" type="xsd:string" />
      <part name="String_2" type="xsd:string" />
      <part name="String_3" type="xsd:string" />
      <part name="int_4" type="xsd:int" />
      <part name="String_5" type="xsd:string" />
      <part name="String_6" type="xsd:string" />
      <part name="int_7" type="xsd:int" />
    </message>
  - <message name="searchIF_advancesearchResponse">
      <part name="result" type="xsd:string" />
    </message>
  - <portType name="searchIF">
    - <operation name="advancesearch" parameterOrder="String_1 String_2 String_3 int_4
        String_5 String_6 int_7">
        <input message="tns:searchIF_advancesearch" />
        <output message="tns:searchIF_advancesearchResponse" />
      </operation>
    </portType>
  - <binding name="searchIFBinding" type="tns:searchIF">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
    - <operation name="advancesearch">
        <soap:operation soapAction="" />
      - <input>
          <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

Figure 7.7: WSDL of logic extracted from Vista Search


http://localhost:8080/florasearch1/search1?WSDL - Windows Internet Explorer
http://localhost:8080/florasearch1/search1?WSDL
Bing
mywebsearch
Search
Cursor Mania
Smiley Central
Screensavers
Favorites
Done
Local intranet | Protected Mode: Off
100%

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://mysite.org/wsdl/florasearch1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="florasearch1"
    targetNamespace="http://mysite.org/wsdl/florasearch1">
    <types />
  - <message name="searchIF1_advancesearch">
      <part name="String_1" type="xsd:string" />
      <part name="String_2" type="xsd:string" />
      <part name="String_3" type="xsd:string" />
      <part name="float_4" type="xsd:float" />
      <part name="String_5" type="xsd:string" />
      <part name="String_6" type="xsd:string" />
      <part name="int_7" type="xsd:int" />
    </message>
  - <message name="searchIF1_advancesearchResponse">
      <part name="result" type="xsd:string" />
    </message>
  - <portType name="searchIF1">
    - <operation name="advancesearch" parameterOrder="String_1 String_2 String_3 float_4
        String_5 String_6 int_7">
        <input message="tns:searchIF1_advancesearch" />
        <output message="tns:searchIF1_advancesearchResponse" />
      </operation>
    </portType>
  - <binding name="searchIF1Binding" type="tns:searchIF1">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
    - <operation name="advancesearch">
        <soap:operation soapAction="" />
      - <input>
```

Figure 7.8: WSDL of logic extracted from Flora search


http://localhost:8080/active-bpel/services/vistasearch1florasearch1Service?WSDL - Windows Internet Explorer
http://localhost:8080/active-bpel/services/vistasear
Bing
mywebsearch
Search
Cursor Mania
Smiley Central
Screensavers
Favorites
Local intranet | Protected Mode: Off
100%

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions name="vistasearch1florasearch1"
    targetNamespace="http://activebpel.org/wsdl/vistasearch1florasearch1"
    xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:ns2="http://java.sun.com/jax-rpc-
    ri/internal" xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
    xmlns:tns="http://activebpel.org/wsdl/vistasearch1florasearch1"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <message name="Response">
      <part name="result1" type="xsd:string" />
      <part name="result2" type="xsd:string" />
    </message>
  - <message name="Request">
      <part name="String_1" type="xsd:string" />
      <part name="String_2" type="xsd:string" />
      <part name="String_3" type="xsd:string" />
      <part name="int_4" type="xsd:int" />
      <part name="String_5" type="xsd:string" />
      <part name="String_6" type="xsd:string" />
      <part name="int_7" type="xsd:int" />
      <part name="String_8" type="xsd:string" />
      <part name="String_9" type="xsd:string" />
      <part name="String_10" type="xsd:string" />
      <part name="float_11" type="xsd:float" />
      <part name="String_12" type="xsd:string" />
      <part name="String_13" type="xsd:string" />
      <part name="int_14" type="xsd:int" />
    </message>
  - <portType name="vistasearch1florasearch1">
    - <operation name="vistasearch1florasearch1">
```

Figure 7.9: WSDL of logic extracted after integration (Vista and Flora)

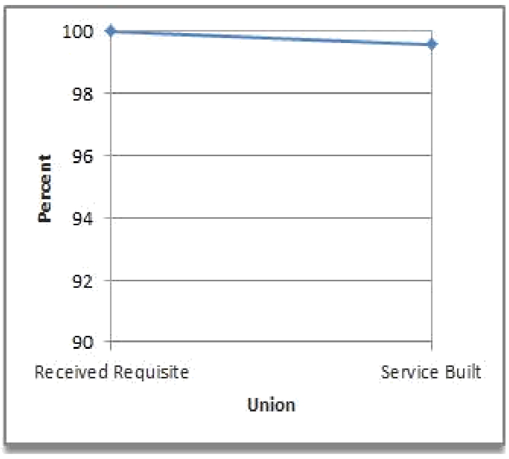
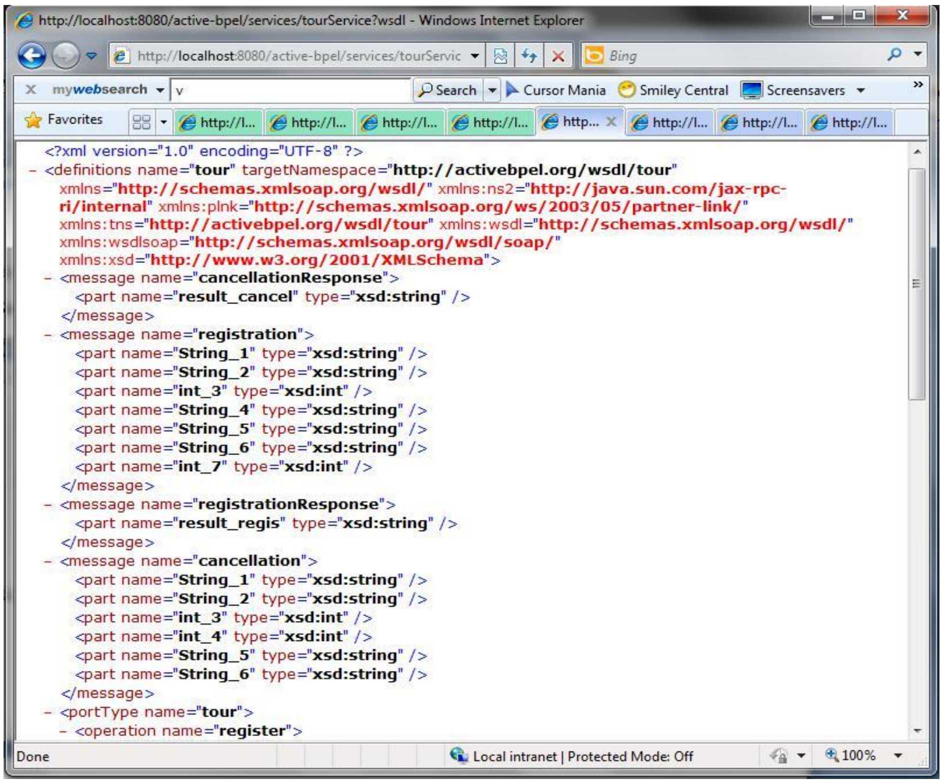


Figure 7.10: Similarity Measure between given Request & Service Built

Demand for service integration may require visit reservation service that could save both for settlement and voyages consequently. Hotel Service incorporates logic and rules for convenience service and travel area contains logic and rules for movement service. Hence the required logic is removed from lodging space and travel area and is composed together. WSDL of the composed service is shown in Figure 7.10. The similarity measure is given in Figure 7.11.


http://localhost:8080/active-bpel/services/tourService?wsdl - Windows Internet Explorer
http://localhost:8080/active-bpel/services/tourServic
Bing
mywebsearch
Search
Cursor Mania
Smiley Central
Screensavers
Favorites
Done
Local intranet | Protected Mode: Off
100%

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <definitions name="tour" targetNamespace="http://activebpel.org/wsdl/tour"
    xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:ns2="http://java.sun.com/jax-rpc-
    ri/internal" xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
    xmlns:tns="http://activebpel.org/wsdl/tour" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <message name="cancellationResponse">
      <part name="result_cancel" type="xsd:string" />
    </message>
  - <message name="registration">
      <part name="String_1" type="xsd:string" />
      <part name="String_2" type="xsd:string" />
      <part name="int_3" type="xsd:int" />
      <part name="String_4" type="xsd:string" />
      <part name="String_5" type="xsd:string" />
      <part name="String_6" type="xsd:string" />
      <part name="int_7" type="xsd:int" />
    </message>
  - <message name="registrationResponse">
      <part name="result_regis" type="xsd:string" />
    </message>
  - <message name="cancellation">
      <part name="String_1" type="xsd:string" />
      <part name="String_2" type="xsd:string" />
      <part name="int_3" type="xsd:int" />
      <part name="int_4" type="xsd:int" />
      <part name="String_5" type="xsd:string" />
      <part name="String_6" type="xsd:string" />
    </message>
  - <portType name="tour">
    - <operation name="register">
```

Figure 7.11: WSDL of tour service (developed by composing hotel and travel service)

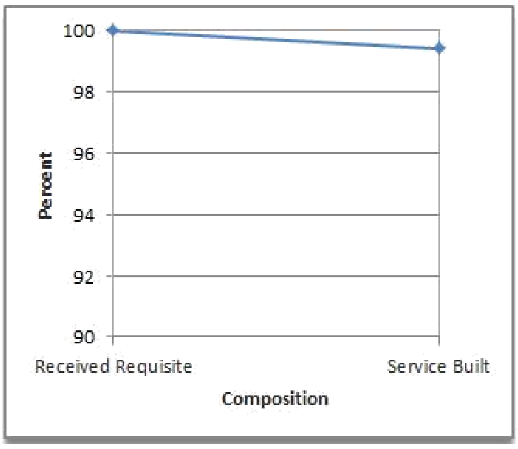
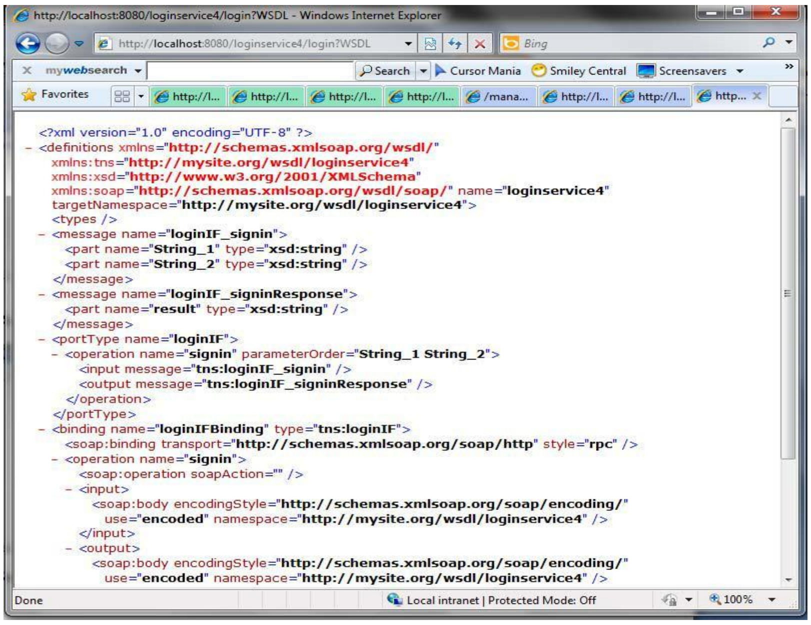


Figure 7.12 Similarity Measure between given request and service built

Service Integration that needs validation services to be stretched out alongside the encryption service. The security benefit contains the encryption part. This piece of the logic is extricated and added to the verification code by association technique or it can be substituted in any required place. Then the extended service is built and deployed. WSDL of the deployed service is shown in Figure 7.12 and the similarity measure is given in Figure 7.13.



http://localhost:8080/loginservice4/login?WSDL - Windows Internet Explorer
http://localhost:8080/loginservice4/login?WSDL
Bing
mywebsearch
Search
Cursor Mania
Smiley Central
Screensavers
Favorites
http://l...
/mana...
http...
Done
Local intranet | Protected Mode: Off
100%

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://mysite.org/wsdl/loginservice4"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="loginservice4"
    targetNamespace="http://mysite.org/wsdl/loginservice4">
    <types />
  - <message name="loginIF_signin">
      <part name="String_1" type="xsd:string" />
      <part name="String_2" type="xsd:string" />
    </message>
  - <message name="loginIF_signinResponse">
      <part name="result" type="xsd:string" />
    </message>
  - <portType name="loginIF">
    - <operation name="signin" parameterOrder="String_1 String_2">
        <input message="tns:loginIF_signin" />
        <output message="tns:loginIF_signinResponse" />
      </operation>
    </portType>
  - <binding name="loginIFBinding" type="tns:loginIF">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
    - <operation name="signin">
        <soap:operation soapAction="" />
      - <input>
          <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://mysite.org/wsdl/loginservice4" />
        </input>
      - <output>
          <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            use="encoded" namespace="http://mysite.org/wsdl/loginservice4" />
```

Figure 7.13: WSDL of authentication service (after encryption service substituted into it)

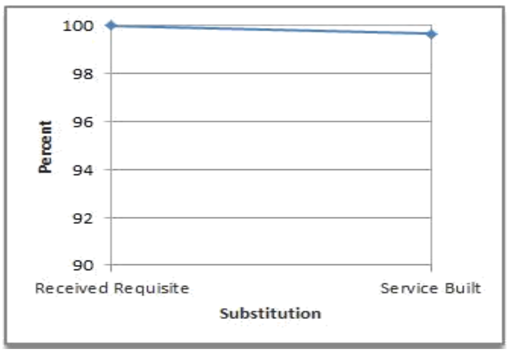


Figure 7.14: Similarity Measure between Request and Service Built

7.9 Summary

This chapter has presented a collaborative agent-based framework along with its theoretical foundations for dynamic Web service integration that could assist organizations to contribute their business logics in a more modern and secured path with their competitors. The introduced system plays out the confirmation of required business logics without the engineer's inclusion at any middle level of service integration. Agents proposed in this section isolate the general work among them and in this way increment the level of automation. They successfully distinguish trust that exist between the Business logics and encourage coordinating them conceivably. The DA agent plays a main role by helping in composite service integration which reduces the overall integration time for about 50 %.

CHAPTER 8

CASE STUDY

COMPOSITE SERVICE INTEGRATION

8.1 Introduction

A collection of native packaged public interfaces are called the composite services. These are otherwise called as ‘Native Services’. These services belong to a specific product or a particular family of products. These services are available in the Integration Repository. These native services serve as building blocks for constructing a sequence of business activities for Composite services without flaws. When a need for change arises, this integration repository serves as a source for integration.

8.2 Composite Service Integration Analogy

A composite Service integration is done in the way described here. At the point when in excess of 2 organizations will work together in excess of one type of example, at that point composite reconciliation methods can be embraced. More than 2 logics are integrated using multiple integration patterns. Thus more than 2 enterprises involve in collaboration of their services. There are two possibilities of integration. First, three new logics may come in the service integration request or a third new service may come in. In both the cases, request is to be integrated with an already integrated Long Term Composed Service. Since one part of request is already available as a service has already been scrutinized, additional integration is done in short time. This is because the property evaluation is already done by the PE Agent. It is the DA that retrieves the available patterns from the repository for new integration. These available patterns and the new logic are then given to the integration agent. The Dependency and the property set of the effectively incorporated services are accessible with the DA. The evaluated results of composite integration are shown in Table 8.1.

The system considers two enterprises A and B which offer search services. For each of the services they offer there are several policy rules associated. The file type of each service may differ and the input validation, encrypt and decrypt types may also differ.

Table 8.2: Performance Evaluation results of Service Integration

Enterprise A Service (S1)	Property Evaluation (S1) Com-Computability Cmp- Completeness				Enterprise B Service(S2)	Property Evaluation (S2) Con-Configurability Acc - Accessibility				Applicable Service Integration Methods across S1 and S2			
Quick Search	Com	Cmp	Con	Acc	Advance Search	Com	Cmp	Con	Acc	Union	Compo sition	Substit ution	Reduct ion
r1: Search by file type	1	1	1	1	r1: Search by content type	1	1	1	1	[r1,r1]	[r1,r1]		
r2:Search by date of pub	1	0	1	0	r2:Search by price range	0	1	1	1				
r3:Search by language	1	1	1	1	r3: Search by keyword	1	1	0	1	[r3,r1]	[r3,r1]		S1[r1]
Security Service1					Security Service 2								
r1: Input validation	1	1	1	1	r1: AES Encrypt-Decrypt	1	1	1	1	[r1,r2]			
r2:RSA Encrypt-Decrypt	0	1	1	1	r2:DES Encrypt-Decrypt	1	1	1	1	[r1,r2]		[r1,r2]	
Online Shopping					Mail Service								
r1: Manage& add items	1	1	1	1	r1: Mail compose	1	1	0	0				
r2: Calculate amount	1	1	1	1	r2: Send msg to mobile	1	1	1	1	[r2,r2]	[r2,r2]		
Registration Service					Authentication Service								
r1: Get contact, edu info	1	1	1	1	r1: Input validation	1	0	1	1				S1[r1]
r2: Get personal info	1	1	1	1	r2:RSA encryption	1	1	1	1	[r2,r2]			
Billing Service					Banking Service								
r1:Manage customer info	1	1	1	1	r1: Credit &debit on acct	1	1	1	1	[r1,r1]	[r1,r1]		
r2: Calculate bill	0	1	1	1	r2: Fund Transfer	1	1	0	1				
r3: Dispatch bill details	1	1	1	1	r3: Balance enquiry	1	1	1	1				S1[r3]
r4: Pay amount	1	1	1	1	r4: Report generation	1	0	0	1	[r4,r1]	[r4,r1]		
Travel Service					Accommodation Service								
r1: Get customer detail	1	1	1	1	r1: Get customer detail	1	1	1	1				
r2: Airline Reservation	1	1	1	1	r2:Hotel Reservation	1	1	1	1	[r2,r2]	[r2,r2]		
r3: Cancelation	1	1	1	0	r3:Cancelation	1	1	1	0				
Login Service					Access Control Service								
r1: Get user credentials	1	1	1	1	r1: Get role and rights	1	1	1	1				S1[r1]
r2:login verification	1	1	1	1	r2: Role based access	0	1	1	1			[r2,r1]	
Medicine Service					Find Doctor Service								
r1: Get disease details	1	1	1	1	r1:Gets disease details	1	0	1	1	[r1,r1]			
r2: Return list of medicine	1	1	1	1	r2: Return list of doctor	1	1	1	1	[r12,r2]			

From Table 8.1 plainly the distinctive services incorporations, property assessment is done and in the same manner the integration design is actualized. Obviously if the openness is 1, at exactly that point the service logics are taken care of for incorporation. Two Enterprise level coordinated effort is assessed in Table 8.1, where r1, r2.. rn are rules having a place with various business logics of Enterprise A and Enterprise B. The method of coordination is chosen by the raised demand once when the property assessment process gets over.

After the integrating service logics is analyzed and after developing a complex logic, SI Agent evaluates the performance of the integration process, compares with its past integrations. The performance of the integration system is measured by service integration time, Logic discovery time, Negotiation time and Dependency Checking time.

Service Integration Time

Service Integration Time is characterized as the time slipped by to achieve the entire coordination process. As such, it is the aggregate time required for each bit of system to execute. Service Integration time is registered by adding up logic disclosure time, service alignment time, and schema generation time, mapping time and service deployment time.

Logic Discovery Time

Logic Discovery Time (T_{ldt}) is the time spent by the Discovery agent to receive and process the request, find out the exact business logic. T_{ld} = Time taken to process the request (T_r) + Time taken to locate the rule (T_l). Let n be number of located logics and T be the time taken to identify the rule, then

$$T_{ldt} = n * T * T_{ld}$$

Negotiation Time

Negotiation Time (T_{sla}) is time taken by the Negotiation Agent to verify the policy in the service level agreement for all the service logics found by DA. It is the summation of collaboration time T_{col} and time taken to do verification for each logic Txp.

$$T_{sla} = \sum_{i=0}^{n} \left(T_{col(i)} + T_{xp(i)} \right)$$

Dependency Analysis Time

Tda means add up to time slipped by for DA Agent to achieve reliance examination process in the entire service advance. Tda is summation of time taken to distinguish the required as to business standards to process the demand, time taken to investigate reliance at business rule, function and parameter. If there exist a request to accept Tαr is time taken to distinguish rundown of principles to process the demand and Tgr is time taken to find setting free language structure in memory for relating tenet, capacity or parameter. Td be add up to time taken to learn reliance at rule, function and parameter level for each found service and n is number of logics found by DA.

$$T_{dv} = (T_{dr} + T_{df} + T_{dp}) + d(t_{dr} + t_{df} + t_{dp})/dt$$

Here T_{dr}, T_{df} and T_{dp} refer time taken to observe dependency at rule, function and parameter level respectively.

$$\begin{aligned} T_{dr} &= T_{gr} + (r * T_{dr}) + \left(f * T_{df} \right) + \left(f * T_{df} \right) \\ T_{df} &= T_{gr} + \left(f * T_{df} \right) + \left(p * T_{dp} \right) \\ T_{pf} &= T_{gr} + \left(p * T_{dp} \right) \end{aligned}$$

t_{dr}, t_{df} and t_{dp} implies time taken to inspect dependency at rule, function and parameter level to manage the changes at runtime respectively.

$$\begin{aligned} t_{dr} &= t_{gr} + (r * t_{dr}) + \left(f * t_{df} \right) + \left(f * t_{df} \right) \\ t_{df} &= t_{gr} + \left(f * t_{df} \right) + \left(p * t_{dp} \right) \\ t_{pf} &= t_{gr} + \left(p * t_{dp} \right) \end{aligned}$$

Total time taken for the whole process of RBA is

$$T_{da} = T_{ar} + (T_{dr} + T_{df} + T_{dp}) + d(t_{dr} + t_{df} + t_{dp})/dt$$

Service Integration Time

Service Integration Time is time taken by the SI agent to integrate the service logics extracted by DC Agent. It is the total of the time taken for the agent to recognize the pattern to integrate the service logics (T_p), time taken to interact with DC Agent to obtain the extracted service logic ($T_{cld)}$ and time taken to deploy the service T_{sd}. T_p is time taken to out the dependencies between the logics and ascertain the proper pattern. Service Aggregation is time taken to aggregate the service with the ascertained mode (T_{agg})

$$T_p = \left(T_1 * T_{rba} \right) + T_{agg}$$

Table 8.2: Performance Evaluation results of Service Integration

Enterprise A	Enterprise B	Integrated Service	T_{Id}	T_{Sla}	T_{da}	T_{PE}	T_{SI}	T_{TI}
Service (S1)	Service (S2)		(ms)	(ms)	(ms)	(ms)	(ms)	(ms)
Quick Search r1: Search file type	Advanced Search r1: Search by content type r2: Search by price range r3: Search by keywords	Compound Search Union: S1(r1) U S2(r1)	0.954	1.132	0.567	0.674	0.395	3.722
Security Service1 r1: Input validation r2: RSA Encrypt-Decrypt	Security Service 2 r1: AES Encrypt-Decrypt	Security Service Substitute:S1(r1) in S2(r2) [Perform input validation on Encryption and Decryption]	0.876	1.354	0.678	0.8	0.563	4.271
Online Shopping r1: Manage & add. items r2: Calculate amount	Mail Service r1: Mail compose r2: Sendmsg to mobile	Alert Service compose: s1(r2)°S2(r2) [Calculate amt, if not paid, sent alert message]	0.979	0.996	0.623	0.710	0.482	3.79
Registration Service r1: Get contact, edu info r2: Get personal info	Authentication Service r1: Input validation r2:RSA encryption	OptRegistration Service S=Reduce(S1(r1)) [S1(r1) is reduced to getedu info only]	0.785	1.046	0.637	0.659	0.475	3.904
Billing Service r1:Manage customer info r2: Calculate bill r3: Dispatch bill details r4: Pay amount	Banking Service r1: credit & debit on acct r2: Fund transfer r3: Balance enquiry r4:Report Generation	Online Billing Compose: S1(r4) ° S2(r1) [Payment debited on corresponding customer acct]	0.851	1.267	0.598	0.692	0.495	3.903
Travel Service r1: Get customer detail r2: Airline reservation r3: cancellation	Accommodation Service r1: Get Customer detail r2: Hotel reservation r3:cancellation	Travel Package Compose:S1(r2) ° S2(r2) [If customer reserve air ticket then also book hotel]	0.912	1.160	0.620	0.786	0.477	3.955
Login Service r1: Get user credentials r2:login verification	Access Control Service r1: Get role & rights r2:Role based access	Login Service S=Reduce (S1(r1)) [S1(r1) is reduced to get username & password only]	0.865	0.899	0.712	0.683	0.491	3.65
Medicine Service r1: Get disease details r2: returns list of medicines	Find Doctor Service r1; Get disease details r2: returns list of specialized doctor	E-Medical Union:S1(r1 &r2) U S2(r2) [Get disease details and suggest medicine and doctor]	0.785	0.963	0.617	0.672	0.483	3.52

Property Evaluation Time

Property Evaluation Time (TPE) is time spent by Evaluation Agent to achieve the property evaluation in the service integration process.

Total Integration Time

$$(T_{TI}) = T_{ldt} + T_{sla} + T_{da} + T_{pa} + T_{si}$$

The efficiency of the work done by the Integration Agent is evaluated with various service logics using various constructs that produced several outcomes. The results are shown in Table 8.2 that shows the effectiveness of the integration process.

Table 8.3: Evaluation results of composite service integration

Enterprises	**Service Logic (Set of Rules)**	**Composite Service Integration Methods**	**Available Service Integration Patterns**	**Service Integration time (ms)** (without pattern analysis)	**Service Integration time (ms)** (with pattern analysis)	**Reduction In Time (ms)**
A	r1: search by keywords	1 U r2 U r3	1 U r2	4.215	2.201	2.014
B	r2: search by file type	r1 U r2) ∘ r3	1 U r2	4.369	2.262	2.107
C	r3: search by language	(r1 U r2) in r3	1 U r2	4.122	2.099	2.023
A	r1: Airline reservation	1 ∘ r2 ∘ r3	1 ∘ r2	4.333	2.102	2.231
B	r2: Hotel reservation	1 U r2 U r3	1 ∘ r2	4.122	2.291	1.831
C	r3: Cab reservation					
A	r1: 8-bit Encode	(r1∘ r2 U (r3∘ 4)	r1∘ r2) , (r3∘ 4)	4.886	2.902	1.984
A	r2: 8-bit Decode	(r1U r2 ∘ r3 U 4)	r1U 2) ,(r3 J r4)	4.964	2.966	1.998
B	r3: 16-bit Encode					
B	r4: 16-bit Decode					

The assessment of the proposed methodology is finished by the ManageEngine Applications and WAPT instrument and the execution are spoken to graphically. The ManageEngine is utilized to decide reaction time, execution time, accessibility properties of the administrations when combination. The WAPT device is utilized for deciding the service integration time, property assessment time i.e., the general execution measure of the considerable number of operators are done by means of this device. The execution assessment charts are appeared in the following Figures.

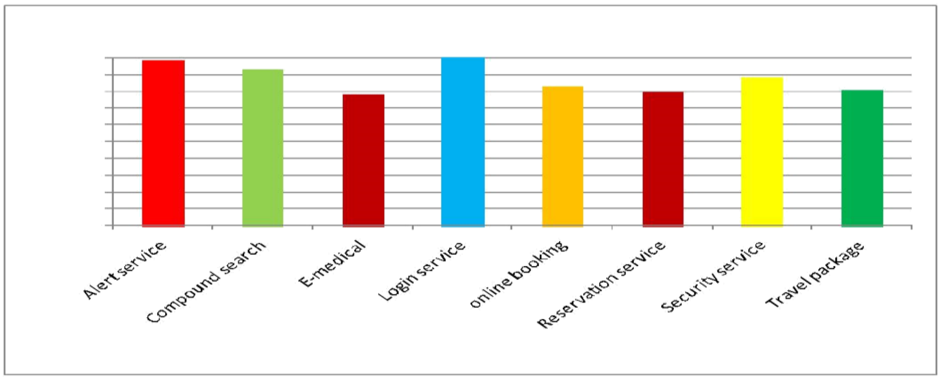


Figure 8.1: Logic Discovery Time

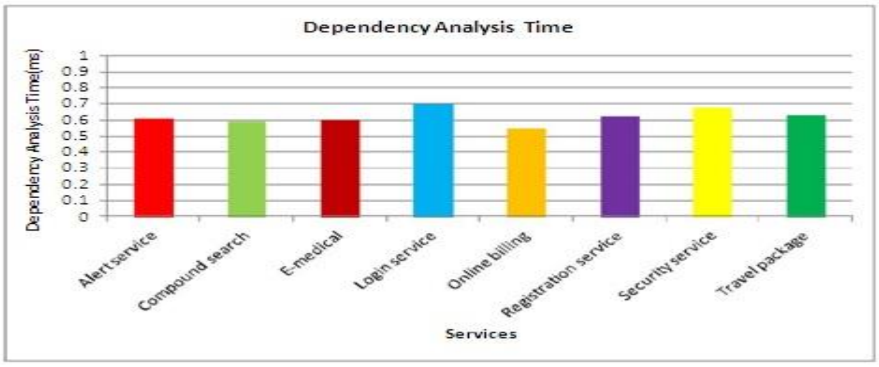


Figure 8.2: Dependency Analysis Time

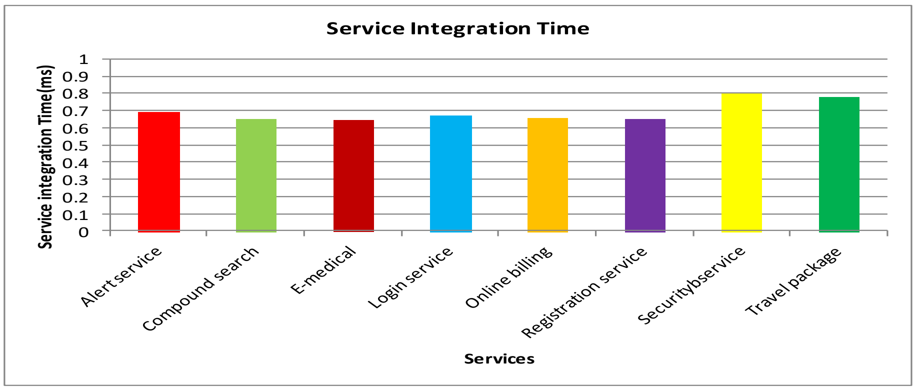


Figure 8.3 Service Integration Time

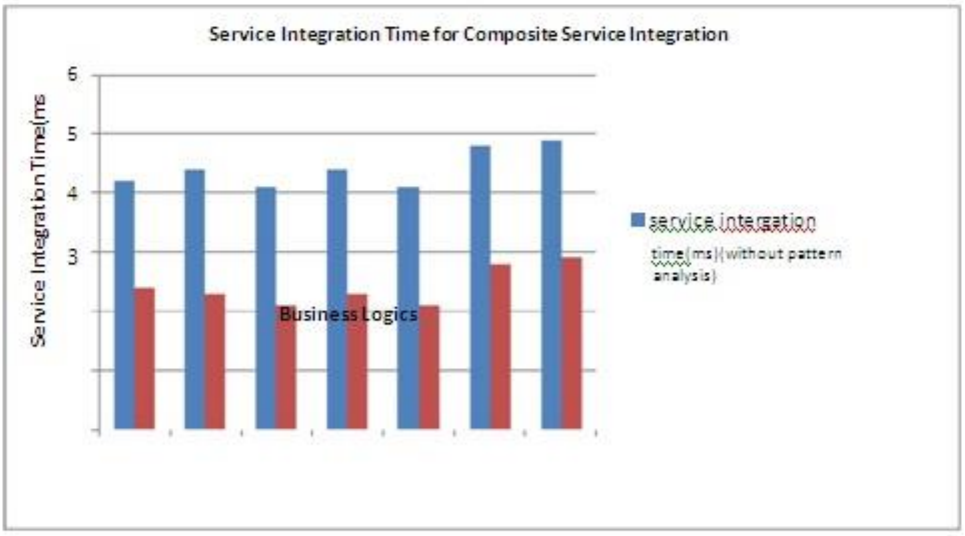


Figure 8.4: Service Integration Time for Composite Service Integration

8.3 Summary

Thus from the case study, the degree of automation is improved by doing theoretical and computational analysis that is important for service integration. FSM is employed as mathematical model to increase the benefit of automation. By accomplishing the complete process with the help of agents cost and the operational risk is reduced to a greater extent. Substantial increase in direct and computational processing is targeted that leads to succeeding reduction in operational risk. A set of services of different enterprises are examined for integration and the performance is measured that shows significant improvement. Especially on composite services that has been taken for study has been scrutinized. The integration time for composite services that involve pre-evaluated logics is comparatively less than those of service integration which does not involve the process of evaluating the business logics. Therefore, only the first request takes time to integrate. The subsequent requests are responded by the agents in a faster manner because of the knowledge of similar integration technique is known by the agent. Thus the overall time taken for service integration drastically reduced to about 50-60 percentage for the requests which have pre-evaluated integration patterns. This cost-effective and easy to adopt methodologies enable the organization to have a high level operational integrity.

CHAPTER 9
CONCLUSION AND FUTURE WORK

A standard mechanism that the applications use to publish and subscribe to other software services over an Internet or the Intranet is the Web services technology. Business Policies provide the service providers with a specification of the verifiable quality characteristics that the service will provide. These Business Policies play a very important role in Change Management Framework. These technologies have opened several opportunities for composing different autonomous services on demand. As a result of this composition several research issues arise for managing the changes during service composition throughout its lifetime.

In this research, an end-to-end framework is proposed that models, implements, verifies, and optimizes the top-down changes in a Long Term Composed Services. The major contributions towards the research are summarized below.

9.1 Research Findings and Level of Objectives Achieved

Turing Machine for Policy Violation Detection

Business policy plays a major role in the process of service integration. An automated model is required to take care of the business policy of integrating services when a need for change arises. The traditional approaches like Domain Specific Service Integration and Semantic Service Integration fail to check the business policies at rule, function and parameter levels. During Service Integration done by these approaches, only 60% to 70% performance level was achieved. Therefore, multi-tape Turing machines are developed to detect the business policies of different services. Once the Business Policies are detected, the Turing Machines also check for any Business Policy Violation that exists between the two integrating services. The performance measure of the Business Policy Violation Detection using Turing Machine was calculated by experiments and the results proved that the process of service integration is improved by 80% when compared to the existing traditional techniques that do not employ Turing Machines for Policy Violation Detection.

ATM for Multiple Change Request

The developed Business Policy Violation System using single Turing Machine could handle a single change at a time. But, Service Integration is a process of making two or more different services of different service providers to collaborate with each other and share their data in order to provide the customer the best service they need. Therefore many change requests may arise at a time. While collaborating with each other, the policy decisions of the services being integrated should be considered. To improve the performance of Policy Violation Detection, the Alternating Turing Machines were constructed to handle multiple service policies at a time. Hence by experimental results, it is proved that Business Policy Violation Detection through ATMs is more accurate and the number of Policy Violation Detected is improved by 90%.

Domain Specific change request

Web benefit Ontology comprises of an arrangement of service concepts. A service integration idea characterizes a kind of Web services inside a space. It catches the normal highlights of Web services. A service integration idea can be seen as a theoretical administration, which can be instantiated by solid Web services. Utilizing the service concepts of the Ontology, Web service functionalities are characterized in a way that is clear and unambiguous to programming agents. The Service Integration system understands the service request better when the Domain Specific Service concepts are described using OWL. Service Integration is improved by 95% when compared to integrating Web services based on extensible integration-specific language.

Reputation Measurement against Malicious Attack

The services provided by some service providers do not fulfill the user requirement but still their credit scores are shown high. Due to their fake identity, owners who want a change tend to make use of those services and end up with an improper integration. In order to avoid these kinds of problems the research has come up with a solution by making use of Reputation Measurement against malicious attacks to find out the fake identities with the help of user credits. Rater's creditability is measured to check whether the integrating services are trustworthy. By doing so, the overall accuracy of the process of Service Integration is improved by 92% when compared to Enterprise Service Integration.

Multi-Agent Architecture for collaborative service integration

Web service Integration is a complex process which involves various complex interfaces to locate the required service. It requires a powerful authentication and access control mechanism to view and utilize the partner's logic at business logic schema level. Also the service level agreement negotiation process should be carried out in an effective manner. To handle such important tasks and to make the service integration with Policy Violation Detection an automated one, several agents might be applied. Hence multi-agent architecture is developed for collaborative service integration. Several agents are developed to take care of the property evaluation. The patterns of the policies to be matched are extracted and are done by the Multi-tape Turing machine. When various agents are employed to different levels of Service Integration, BL patterns are generated and services are integrated through union, composition and substitution. Therefore the overall integration time is reduced to about 50%.

9.2 Future Research Work

Since Web services get updated as the technology develops, there is always a need to improvise its performance. The dynamicity of service integration has to be maintained and policy detection has to be further unrehearsed. Change specification should contain unambiguous and sufficient information about a change so as to respond to the change in an efficient manner.

The research can be further enhanced with following features.

- Meta Heuristic Approach can be applied to the policy detection algorithm to optimize the obtained solution.
- Security policy may be further considered after the detection of policy violation to enhance the security constraints of the composing services.
- The constructed Turing Machine may also be used to verify the SLAs of services for Web service Choreography and Orchestration.

Research might be done to do early forecast of response time i.e., regardless of whether the requested integration can be set aside a few minutes in this way deciding hardness and fulfillment of the service integration request.

REFERENCES

1. **Adina Mosicat et al. (2011)** “Automated Maintenance of Service Compositions with SLA Violation Detection and Dynamic Binding”, *International Journal on Software Tools for Technology Transfer,* 13(2), pp. 167–179.
2. **Alex Talevski et al. (2015)** “Reconfigurable Web Service Integration in the Extended Logistics Enterprise”, *IEEE Transactions on Industrial Informatics,* 1(2).
3. **Amiri et al. (2010)** “QoS aware Web service composition based on genetic algorithm”, *In Telecommunications (IST), 5th International Symposium,* 502 – 507.
4. **Andr´es, B.R (2011**) “Change management practices: Impact on perceived change results”, *Journal of Business Research*, 64(3):266–272.
5. **Animesh Chaturvedi (2014)** “Automated Web service change management AWSCM-A Tool”, *IEEE 6th International Conference on Cloud Computing Technology and Science*.
6. **Arif Ali Khan et al. (2012**) “A Purpose Frame work for Requirement Change Management in Global Software Development”, *International Conference on Computer & Information Science (ICCIS),* 2.
7. **Azlan Ismail et al. (2013)** “Incremented service level agreement violation handling with time impact analysis”, *Journal of Systems and Software*, 86(6).
8. **Bao et al. (2012)** “Massive sensor data management framework in cloud manufacturing based on Hadoop”, *in Proc. IEEE 10th Int. Conf. Ind. Inf.,* pp. 397–401.
9. **Belqasmi et al. (2014)** “A Case Study on IVR Applications, “Provisioning as Cloud Computing Services”, *IEEE Network,* 28: 33–41.
10. **C. Müller et al. (2012)** “SALMonADA: A Platform for monitoring and displaying violations of WS-Agreement Compliance Document”, *Proceedings of the 4th International Workshop on Principles of Engineering Service-Oriented Systems*, pp. 43–49.
11. **Camlon et al. (2010)** “Towards a flexible service integration through separation of business rules”, *14th IEEE International Enterprise Distributed Object Computing Conference*.

12. **Chang Lee et al. (2003)** “Business value of B2B electronic commerce: the critical role of inter-firm collaboration”, *Electronic Commerce Research and Applications 2, Elsevier B.V.* pp.350–361.
13. **Deng et al. (2010)** “A Study and Design of SOA-based Service Integration for Logistics Customs clearance”, International Symposium on Parallel and Distributed Processing with Applications.
14. **Dominique et al. (2010)** “Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of Web services’, *IEEE Transactions on Services Computing*, 3(3).
15. **Eleni Stroulia et al. (2013)** “An Intelligent Agent Architecture for Flexible Service Integration on the Web”, *IEEE Transactions on Systems, Man and Cybernetics*, 33(4).
16. **Freddy et al. (2010)** “Seeking Quality of Web Service Composition in a Semantic Distribution”, *IEEE Transactions on Knowledge and Data Engineering*, 23(6).
17. **Hadad et al. (2010)** “TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition,” *IEEE Trans. Service Comput.*, 31:73– 85.
18. **Hassan et al. (2012**) “Web Service Composition: Models and Approaches”, *Proceedings of IEEE International Conference on Multimedia Computing and Systems*.
19. **Hayes J. et al. (2014)** “The Theory and Practices of Change Management”, *Palgrave Macmillan*.
20. **Hendler J. (2001)** “Agents and the Semantic Web”, *Intelligent Systems, IEEE,* 16(2):30–37.
21. **Hennie (2006)** “On-Line Turing Machine Computations”, *IEEE Transactions on Electronics Computers*.
22. **Hui Ma et al. (2012)** “A formal model for the interoperability of service clouds”, *Service Oriented Computing and Applications*, 6(3):189–205.
23. **Hussain et al. (2013)** “RESTful Web Service Integration using Android Platform”, Proceeding of IEEE Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT).
24. **Ing-Ray Chen et al. (2015)** “Trust Management for SOA-Based IoT and Its Application to Service Composition”, *IEEE Transactions on Services Computing*, 9(3).

25. **Inico (2014)** Inico Technologies [Online]. Available: http://www.inicotech. com/. [Accessed: 05-Mar-2014].
26. **Jiachen Hou et al. (2016)** "Integration of Web Services technology with business models within the total product design process for supplier selection", *International Journal of Computers in Industry,* 57(8–9).
27. **Jun Shen et al. (2010)** "Analysis of business process integration in Web service context", *International Journal of Future Generation Computer Systems,* 23(3).
28. **Karim Benouaret et al. (2014)** "Web Service Composition with Fuzzy Preferences: A Graded Domain Name Relationship Based Approach", *ACM Transactions on Internet Technology (TOIT),* 13(4).
29. **Katharina Krombholz et al. (2015)** "Fake identities in social media: A case study on the sustainability of the face book business model", *Journal of Service Science Research,* Springer, 4(2).
30. **Kehagias et al. (2010)** "An ontology based framework for Web Service Integration and Delay to Mobility Impaired users", World Summit on Knowledge Society Part of the Communications in Computer and Information Science book series (CCIS, volume 111).
31. **Khoshafian (2006)** "Service oriented enterprises", Auerbach Publications, Boston, MA, USA.
32. **Li et al. (2012)** "Dynamic service integration for reliable and sustainable capability provision", *International Journal of Systems Science,*43(1).
33. **Lin et al. (2010)** "The design and implementation of service process reconfiguration with end-to-end qos constraints in soa", *Service Oriented Computing and Applications,* 4(3):157–168.
34. **Losup et al. (2011)** "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, 22(6).
35. **Lu et al. (2015)** "Ontology-based knowledge modeling for automated construction safety Checking", *Safety Science,* volume 79.
36. **Lu Liu et al. (2012)** "Dynamic service integration for reliable and sustainable capability provision", *International Journal of Systems Science,* 43(1).
37. **Marcos Palacios et al. (2015)** "Coverage-Based Testing for Service Level Agreements", *IEEE Transactions on Services Computing* ,8(2).

38. **Mario et al. (2015)** "ONLI: An ontology-based system for querying DBpedia using natural language paradigm", *Expert Systems with Applications*, 42(12).
39. **Matthias Weske (2012)** "Business Process Management Architectures", *In: Business Process Management.* Springer, Berlin, Heidelberg.
40. **Medjahed et al. (2003)** "Composing Web services on the semantic Web," *Journal on Very Large Databases*, 4:333–351.
41. **Medjahed et al. (2003)** "Composing Web Services on the Semantic Web", *The VLDB Journal, Special Issue on the Semantic Web*, 12(4).
42. **Mier et al. (2012**) "Automatic Web Service Composition with a Heuristic Based Search Algorithm", *Proceedings of IEEE International Conference on Web Services*.
43. **Ouransa et al. (2013)** "The PORSCE-II Framework: Using AI Planning for Automated Semantic Web Service Composition", International Journal of Knowledge Engineering for Planning and Scheduling, Volume 28.
44. **Pablo et al. (2010)** "Composition of Web services through genetic programming", *Evolutionary Intelligence*, 3:171–186.
45. **Paulraj et al. (2012)** "Process Model based Automatic Service Discovery and Composition of Composite Semantic Web Services using Web Ontology Language for Services (OWL-S)", *Journal of Enterprise Information Systems,* Volume 6.
46. **Philipp Leitner et al. (2010)** "Runtime predictions of Service Level Agreement Violation for composite services", *Service-Oriented Computing,* Springer-Verlag Heidelberg.
47. **Philipp Leitner et al. (2013)** (2013) "Data-Driven and Automated prediction of service level agreement violations in service composition", International Journal of Distributed and Parallel Databases, 31(3):447–470.
48. **Pierluigi et al. (2009)** "URBE: Web service retrieval based on similarity evaluation", *IEEE Transactions on Knowledge and Data Engineering*, 21(11).
49. **Puttonen et al. (2013)** "Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems", in 2013 *IEEE 20th International Conference on Web Services (ICWS).*

50. **Q. Yu (2008)** “A Foundational Framework for Service Query Optimization”, PhD thesis, Virginia Tech.
51. **Quanwang Wu et al. (2014)** “Broker Based SLA-Aware Composite service provisioning”, *Journal of Systems and Software,* 96.
52. **Radu et al. (2011)** “Dynamic QoS management and optimization in service-based systems”, *IEEE Transactions on Software Engineering,* 37(3).
53. **Reid et al. (1996)** “An Integrated Management Model for Virtual Enterprises: Vision, Strategy and Structure”, *IEEE Engineering Management Conference,* pp. 522–527.
54. **Salman Akram et al. (2010)** “A Change Management Framework for Service Oriented Enterprises”, *International Journal of Next-Generation Computing,* 1(1): 91–112, 22p.
55. **Sebastian Speiser et al. (2011)** “Integrating Linked Data and Services with Linked Data Services”, Proceedings of European Semantic Web Conference, volume 6643.
56. **Shah et al. (2009)** “A Qos perspective on exception diagnosis in service-oriented computing”, *J. UCS,* 15(9):1871–1885.
57. **Shanshan Qui et al. (2012**) “A Trust Impact Analysis Model for Composite Service Evolution”, 19th Asia-Pacific Software Engineering Conference, Vol. 2.
58. **Sheng-Yuan Yang (2013)** “Developing an energy-saving and case-based reasoning information agent with Web service and ontology techniques, Expert Systems with Applications”, 40(9).
59. **Somluck L-Ongsri et al. (2015)** “Incorporating Ontology-based semantics into conceptual modeling”, *Journal of Information Systems,* Volume 52.
60. **Son et al. (2014**) “Semantic context-aware service composition for building automation system”, IEEE *Transactions on Industrial Informatics,* 10(1), Mar 2013.
61. **Sun et al. (2011)** “Master-slave parallel genetic algorithm based on Map Reduce using cloud computing,” Appl. Mech. Mater., 121–126: 4023–4027.
62. **Sunitha T. et al. (2011)** “Survey of Potential Attacks on Web Services and Web Services Composition”, *Proceedings of IEEE International Conference on Electronics Computer Technology,* Volume 6.

63. **Szu-Yin et al. (2014)** “A Trustworthy QoS based Collaborative Filtering Approach for Web Service Discovery, *Journal of Systems and Software*, Volume 93.

64. **Tang et al. (2013)** “Dynamic Web Service Composition Based on Service Integration and HTN Planning”, *Proceedings of IEEE 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*.

65. **Tao et al. (2012)** “Modelling of combinable relationship-based composition service network and the theoretical proof of its scale-free characteristics”, *Enterprise Inf. Syst*., 6(4):373–404.

66. **Thirumaran et al. (2011)** “Security Model to Incorporate Add-On Security for Business Services”, *International Journal of Computer Applications* 22(2):1–10.

67. **Thirumaran et al. (2015)** “Collaborative Web Service QoS Prediction with Multi-Criteria Decision Making Using CBNIMF”, (IJCSIT) *International Journal of Computer Science and Information Technologies*.

68. **Thirumaran et al. (2016)** “A formal approach for change impact analysis of long term composed services using Probabilistic Cellular Automata”, *Journal of King Saud University - Computer and Information Sciences*, 28(2).

69. **Tristan Glatard et al. (2008)** “A Service-Oriented Architecture enabling dynamic service grouping for optimizing distributed workflow execution”, Journal of Future Generation Computer Systems, 24(7).

70. **Viriyasitavat et al. (2012)** “SWSpec: The requirements specification language in service workflow environments”, *IEEE Trans. Ind. Inf.,* 8(3):631–638.

71. **Wang (2012)** “Editorial advances in information integration infrastructures supporting multidisciplinary design optimization”, Enterprise *Inf. Syst.,* 63:265.

72. **Wang et al. (2011)** “Evaluating Feedback Rating for Measuring Reputation of Web Services”, *International Conference on Services Computing*.

73. **Weiwei et al. (2013)** “Reputation Aware QoS Value prediction of Web Services”, *IEEE International Conference on Services Computing*.

74. **Wu et al. (2012)** “A novel CMII-based engineering change management framework: an example in Taiwan’s motorcycle industry”, *IEEE Transactions on Engineering Management*, 59(3).

75. **Wu et al. (2013)** "Predicting quality of service for selection by neighborhood-based collaborative filtering", *IEEE Transaction on System Management*, Cybern: Syst., 43(2):428–439.
76. **Xumin Liu et al. (2013)** "SCML: A change Management Language for Adaptive Long Term Composed Services", *Advanced Web Services* pp. 225–252, *Springer*.
77. **Xumin Liu et al. (2011)** "Efficient change management in long-term composed services", *Service Oriented Computing and Applications*, 5(2), pp. 87–103.
78. **Xumin Liu et al. (2011)** "Ev-LCS: A System for the Evolution of Long-Term Composed Services", *IEEE Transactions on Services Computing*, 6(1).
79. **Yan Hu et al. (2015)** "Time Aware and Data Sparsity Tolerant Web Service Recommendation Based on Improved Collaborative Filtering", *IEEE transactions on services computing*, 8(5).
80. **Yanhua Du et al. (2014)** "Timed Compatibility analysis of Web service composition: A Modular Approach based on Petri Nets", *IEEE Transactions on Automation Science and Engineering*, 11(2).
81. **Ying et al. (2003)** "A Web services-based framework for business integration solutions", *Electronic Commerce Research and Applications* 2, Elsevier Science B.V., pp. 15–26.
82. **Ying Huang et al. (2015)** "A Web services-based framework for business integration solutions, Electronic Commerce Research and Application", *International Journal of Electronic Commerce Research and Applications,* Springer 2(1).
83. **Zakatia et al. (2008)** "Toward Behavioral Web Services Using Policies", *IEEE Transactions on Systems Man and Aybernetics - PART A: Systems and Humans*.
84. **Zakiria et al. (2012)** "Towards a User Centric Social Approach to Web Service Composition, Execution and Monitoring", *Proceedings of International Conference on Web Information Systems Engineering*, Springer, volume 7651.
85. **Zeeb et al. (2010)** "WS4D: Toolkits for Networked Embedded Systems Based on the Devices Profile for Web Services", *39th International Conference on Parallel Processing Workshops (ICPPW)*, 2010.

86. **Zheng et al. (2012)** "An Intent based approach for service Discovery and Integration", *IEEE 17th International Conference on Computer supported co-operative work in design*.
87. **Zhiying Cao et al. (2015**) "A Context-Aware Adaptive Web Service Composition Framework", *IEEE International Conference on Computational Intelligence & Communication Technology*.
88. **Zhou et al. (2012)** "Guest editorial special section on enterprise systems", *IEEE Trans. Ind. Inf.,* 8(3):630.
89. **Zhu et al. (2012)** "Group role assignment via a KuhnMunkres algorithm-based solution", *IEEE Transaction on System Management, Cybern: Part A: Syst. Humans,* 42(3):739–750.
90. **Zhuoren et al. (2010)** "A research on multi-layer structure for dynamic service integration: an innovative model based on SOA", The 2nd IEEE International Conference on Information Management and Engineering.

LIST OF PUBLICATIONS

International Journals

1. Tiroumalmouroughane S, Thambidurai P, "Identification of Business Policy Violation in Service Integration of Long Term Composed Services", International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.75 (2015).

2. Tiroumalmouroughane S, Thambidurai P, "Agent Based Dynamic Service Integration Framework using FSM" Helix International Journal, ISSN 2277 – 3495, Vol.7 No.5, 2017.

3. Tiroumalmouroughane S, Thambidurai P, "Reputation Based Business Policy Violation Detection of Long Term Composed Services with Efficient Alleviation of Malicious Rating of Violated Service", International Journal of Ad-hoc and Ubiquitous Computing-**SCI (**Accepted for Publication**)**.

International Conferences

1. Tiroumalmouroughane S, Thambidurai P, "An Intent Based Web Service Composition and Integration through Active –XML Framework", South Asian Journal of Research in Engineering Science and Technology (SAJREST), ISSN (ONLINE): 2455-9261, VOLUME: 01 Special Issue: 01 (ICRDICT – 2016).

2. Tiroumalmouroughane S, Thambidurai P, "Automatic Policy Based Web Service Integration Using Ontology", ICIA-16, August 25-26, 2016, Pondicherry, India ©2016 ACM. ISBN 978-1-4503-4756-3/16/08.

3. Tiroumalmouroughane S, Thambidurai P, "A Collaborative Framework for Dynamic Service Integration using Agents", I2C2-17, June 23-24, 2017, Coimbatore. ISBN 978-1-5386-0373-4.