CHAPTER 3
GOALS AND EXPERIMENTATION METHODOLOGY

3.1 INTRODUCTION

Service Oriented Applications (SOA) is found to be the prominent base than component based applications. The goals of SOA are to improve manageability, security, interoperability, integration and enterprise computing. As a consequence there arises a need for conversion of component based applications to service oriented applications satisfying the demand of large group of customers. Web services yield several technological and business benefits a few of which includes application and data integration, versatility, code re-use and cost savings. The scope of Web services is evolving regularly serving new set of people and the dynamic nature of web services had led to extravagant evolution. Henceforth there emerges an essential urge for a framework to manage this evolution of changes called Change Management Framework.

The experiments of the proposed research are designed such that to accommodate diversified features such as runtime management, change reaction management, proactive management in this framework through a standard model and thereby to validate the proposed metrics provided by the model on versatile applications. This chapter describes the goals of the research presented in this thesis along with the organization of the experiments carried out for the proposed research.

3.2 NEED FOR BUSINESS LOGIC EVALUATION MODEL

Due to the drastic evolution of the new web era the Business to Business (B2B) relationships has been transformed to Business to Customer (B2C) relationships. This in turn had made the Web services more prone to changes and hence the business functions have to be coordinated with their new expectations. Apart from the changes projected by the customers, there emerges a need for changes proposed by business people in order to improve their quality of service performance wise for the sake of competing with the other business parties and to maintain the reputation of the enterprise. This business logic evaluation model facilitates the business analyst to manage the changes effectively without the help of IT developer thereby contributing to reduced cost and time.
Even though it is critical to manage these changes since changes in a service might have side effects on the other services inter related to that service in which changes has been done and in case of emergency incorporation of certain changes the business logic evaluation model helps the business analyst to clearly decide whether to commit the changes or not by using the impact analyzer which predicts the consequences of a change. It also provides better run time support where the service logic can be predicted manageable with the help of property evaluator component. In addition to that, it also provides a standard procedure and methodologies for managing the changes.

3.3 GOALS OF THE PROPOSED RESEARCH

The proposed research is aimed to develop a Business Logic Evaluation Model for web service change management. In this respect, the goals of this research are derived such as to extend the support to the Business Analyst to manage the service logic and to assess and evaluate the changes effectively as follows:

Figure 3.1 Layered view of the Experimentation Methodology
I. To provide a standardized method and procedure for efficient and prompt handling of all changes over the web services in order to reduce the maintenance cost and time, and improve the service quality.

II. To support business analyst to manage the changes effectively in order to meet the expected business outcome.

The objectives listed below leads to attain the goals by using Finite State Machine:

(i) To evaluate the properties and change factors of service business logic through Dependency Analysis Approach using Finite State Machine (FSM).

(ii) To ensure all changes are assessed, implemented and evaluated in a controlled manner.

(iii) To extend the model for impact and prediction analysis over the evaluated properties and change factors using Cellular Automata.

Here the first objective provides run time support by evaluating various properties of business logic like computability, traceability, dependency, accessibility and configurability. If all these properties are met for target business logic then it is decided that the service logic is manageable. The second objective is meant for evaluating the various change factors such as Order of Execution, Similarity Measure, Code Consistency, Schema Validation, Business Policy Enforcement, Time Complexity, Space Complexity and Mapping Function. The third objective is supposed to verify the changes associated with the change request are successfully done. The results of first and second objectives are the source of fourth objective to perform impact and predictive analysis which in turn determines the risk associated with the changes and make Business Analyst comfort to attain the excepted business outcome.

3.4 EXPERIMENTATION METHODOLOGY

The Change Management Framework (CMF) comprises of storage subsystem and four layers namely: Analysis layer, Evaluation layer, Execution layer and Validation layer as shown in figure 3.1. The Storage Subsystem constitutes four components such as: Service Repository, BL (Business Logic) Source, Knowledge Base and Audit Log. The processes involved in analysis layer are: Request Analysis, Change Plan, Change Schedule and Business Logic Analysis. The processes concerned with evaluation layer are: Property Evaluation, Change Evaluation and Impact Analysis. The steps associated with execution layer are: Schema to
Service Translation, Integration Adaptor, Build & Deployment and Version & Audit. The processes in connection with validation layer are: Goal Assessment and Statistical Validation.

3.4.1 Storage Subsystem

The requirements for each layer in the CMF such as service repository, BL source, knowledge base and audit log are deposited in the storage subsystem.

3.4.1.1 Service Repository

Service Repository is an accumulation of services and its analogous details managed by an Enterprise. It is local repository of services. The service repository contains around 100 services categorized under various domains such as Banking, Insurance, Transport, Retail, etc. 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.

3.4.1.2 Business Logic Source

The Business Logic Set (BL Set) comprises of four subsets namely: Rule set, Functionality set, Parameter set and Property set. In a rule based system the rules which are necessary for a schema are elicited from the rule repository and then these rules are integrated to form the overall schema as per the logic of the change request. In addition to the above mentioned sets BL source has State Transition Table (STT) of the FSM for the logic associated with the change request and Change Measure Table (CMT). The CMT is a segment of STT but it has additional information about the changes such as details about the access permissions and the owner of the logic. CMT is instantly built when the changes are done and it exactly locates and projects only that part of the FSM where the change actually took place. The rules, functions and parameters of BL set are stored in the BL Source with their corresponding id, syntax, description and schema.

3.4.1.3 Knowledge Base

This component is majorly used to perform impact analysis in order to obtain proactive measures. The forthcoming consequences are predicted by the impact analyzer based on the
incident matching and behavioral analysis rendered on the services offered so far. The knowledge base comprises of incidence of changes that has been occurred so far and maintain it as cellular automata patterns. The knowledge base stores the entire incidence whenever the change event occurs regardless of the status of the change whether it is successful or failure. For a particular time slice we can get the incident patterns in various dimensions. Both pattern generation and incidence matching are required before committing the change and they are also useful for evaluating the proactive measure.

3.4.1.4 Audit Log

Audit log is a repository catalog for change history which stores the exact location along with the information about the domain, sub domain, business process and the service where the change occurred. It is similar to a log file which comprises of the time and date at which the change has occurred, the details about the owner who has the authority for that particular block of the business logic, the details regarding the changes and the information about the business process in which the change took place. The entire changes which have occurred and exception which have been thrown are recorded in audit log for incidence matching in the future.

Whenever a change request arises, services are been chosen using incidence matching from the service repository which comprises of service id, service name, service location, functional and nonfunctional description. Then we use BL source for extracting source code which comprises of BL set and its corresponding schema. For the impact analysis in proactive management we use the knowledge base component for predicting the forthcoming consequences. The incidence of the changes will be stored in CA patterns.

3.4.2 Analysis Layer

Changes are inevitable and CMF focuses on the management of the changes. Changes can be classified into two categories: top down changes and bottom up changes. The responsibility of analysis layer is to identify the type of changes, domain, context, process owner (analyst) and priority. The change request is being analyzed extensively categorized into domains according to the type of the change and its being matched with the entire set of incidents previously stored and the plan is derived on how the change is going to be
implemented depending upon its priority and in BL analysis all the possible similarity matches are found in between the new change request and incidences.

3.4.2.1 Request Analysis

The new change request is completely analyzed in order to verify whether there is any incidence exist in the incident stored in service repository or it’s a completely new request and then taking the appropriate decisions. The two types of change request analysis are: Domain analysis and Context analysis. Domain analysis refers to identification of which project domain the change request belongs to. Context analysis refers to how instantaneously the change request is occurred and which circumstance the change request has been raised.

3.4.2.2 Change Plan

The change requests are first grouped under two categories namely: Top down changes and Bottom up changes. Top down changes refer to those that are initiated by the owner of the service. They are usually the result of business policies and business regulations. Bottom up changes refer to those that are initiated by the consumers. They are usually the result of an error, an exception, or an alternation in a web service space. Change plan brings out the entire set of change requests similar to existing incidence and also identifies whether it’s a most frequently occurring change.

3.4.2.3 Change Schedule

Change Schedule perceives whether the change request can be completed within the appropriate desired time based on the past experience. The priority of the change will also be determined based on context and emergency.

3.4.2.4 BL Analysis

The request analysis of the change request involves processes such as determining the type of change request, the priority of the change request, verifying the feasibility of change, etc. Further the change plan, change schedule, whether the change request can be accomplished within the expected time will be determined. 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 exist among the entities. The FSM equivalent to the BL Set is constructed with reference to predefined State Transition Table of target service logic then the Business Logic Schema will be generated from FSM.

3.4.3 Evaluation Layer

As for now, Change Management Framework (CMF) facilitates maintenance in the form of all sorts of modifications and enhancement wherein it primarily emphasis on maintenance. So far CMF had been described as the framework dedicated for the management of evolution of services but by no means has CMF been characterized in layered approach. The proposed model has three core phases such as Property Evaluation, Change Evaluation and Impact analysis which are meant to address three key management issues such as Run Time Management, Change Reaction Management and Proactive Management. These issues are resolved by the three core phases of the model individually as well as collaboratively. Developer is obliged to view business logic; we are bringing business logic into analyst view but in schema level whereby the CMF strives to induce the analyst to carry out the changes due to better confidentiality. The privilege of schema is to compose meta information in concise manner intelligible by the analyst by which there is a proliferation in semantic range. Schema also yields change support (i.e.) despite the changes of Business alignment CMF should guarantee the corresponding requirements and specifications.

3.4.3.1 Run Time Management

Deciding if the extracted service logic is manageable or unmanageable resolves the run time issues such as computation failures, configuration failures, access permission denied, etc. This can be achieved by Property Evaluator which act as prerequisite for change management 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 change evaluation is said to be carried out. This process uses a manageability algorithm named CA calibration algorithm for evaluating the properties. These properties abode in the workflow whether the
logic is equivalent to the requirements and verifies whether the changes are manageable. The assurance of the above mentioned properties manifests interoperability since the characteristics of it are well established by the properties. More than 500 change requests are considered for change evaluation and all the requests will encounter the property evaluation phase in order to ensure that they are manageable. These change requests are associated with the services stored under various domains in service repository. Some particular change requests requires merely integration of logics existing earlier which emerges the need for Service Integration. Properties evaluated through the property evaluation phase serve as a prerequisite for service integration. The four service integration methods such as union, composition, substitution and reduction which demands infinitesimal effort for producing the new logic as per the change request without any sort of code injection.

**Dependency:** The different levels of dependency are: Input/ Output dependency, Calling sequence dependency, Mapping dependency (parameters mapping). In I/O dependency the input for one service might be the resultant output of another service where two services are dependent on each other. In calling sequence dependency a function present in a service calls a function present in another service. And in mapping dependency, the number of parameters and their data type which are requisite for a service should match.

**Computability:** The newly composed function for the new change request is rendered as computable only when it can be represented by initial function or composite function which is termed as primitive computable function. Most of the services today are developed using rule based approach. In this framework we hold on to a rule based approach which comprises of four sets: Rule set, Function set, Parameter set and Dependency set. In a rule based system the rules which are necessary for a schema are elicited from the rule repository and then these rules are integrated to form the overall schema as per the logic. To be more precise all the rules, functions, parameters and dependency relations are predefined and enclosed in their corresponding sets. For every change there are equivalent predefined tags present in the set. The logic for the change request comprises of rules which are extracted from the rule set. So when it is decomposed entire rules will be present in rules set and the entire functions will be present in functions set. A newly designed schema is said to be computable only when the tags enclosed within them are predefined tags and the user also has the privilege to define tags but it should
belong to the category of predefined tags. The drawback of the rule based system is that not all the change requests can be implemented using rules alone considering the fact that there are change requests which needs rules belonging to different processes to be merged where the merging process should be dealt logically. Even then it is beneficial when the changes are done using rule based system since most of the current enterprises publish services using rule based due to easy maintenance.

**Configurability:** The direct implementation of web service sometimes has business logic and system logic combined together. The need for configurability is to isolate the system logic from the existing business logic. The system logic is used for external connection to the database and other components. The changes in configuration logic should not affect the business logic and vice versa. The issue in configurability is that avoidance of certain services which involves direct implementation in system logic independent of business logic is mandatory. We use First Order Logic (FOL) to identify user, resources and environment in order to alter the configuration logic flexibly.

**Accessibility:** When a service needs to amalgamate with another service within the same enterprise accessibility issues doesn’t prevail and permissions are not indispensible. However when a service tries to access another service outside the enterprise, permissions are momentous. The services may be in need of access permissions like read permission, write permission and modification permission. The logic to be implemented as per the change request requirements are segmented into blocks wherein each block desires access permission to be acquired. Access points will be set for each block that insists access permission and the scope of it is from the beginning of the block to the end of the block.

**Traceability:** Errors prevailing in services can be easily traced and located. Using FSM we can trace the exact position in the workflow where the exception has been raised and also the reason for why the exception has been raised can be interpreted. Trace points are used to pinpoint the exact location of the exception and Trace path is used in identifying the exception path in the flow of execution. Due to dependency trace points can isolate a part of the source code for execution. Traceability is also facilitates to find the defects occurred in the property evaluation
(i.e.) to trace which property has failed and change evaluation (i.e.) to trace the root cause for the changes. We can get a complete navigation through the trace path for exploring exceptions.

### 3.4.3.2 Change Reaction Management

Change reaction management (CRM) proclaims what desirable operation is needed to be performed on the resource for the new change request and then the rules are framed and validated whether constraints are satisfied. The two types of changes for change request: Domain analysis and Context analysis. CRM determines who is responsible for handling the change request depending upon the types and severity of the change request. Once the properties are evaluated and detected to be manageable change evaluation will be carried out.

The two levels in CRM are: Functional and Non Functional. The functional level is further categorized as: Schema level and Source code level. Schema level consists of Schema validation, Order of Execution, Similarity Matching, Mapping Function and Business Policy Enforcement. The similarity measure of logical constructs along with code generative power is validated at schema level. The logical constructs comprises of iterative statements, control statements and input/output statements. It is merely a preliminary verification whether the schema has code generative power since the code is generated only after the changes are accepted that they are feasible and they can be committed. The reaction reflected after the change in order of execution is that the order of execution may vary from the previously existing order and every process is permitted to change its order of execution only when the change request specifies it. With respect to mapping function, I/O similarity mapping, output generative power mapping and the mapping between the parent rule and the derived rule are verified. In Business Policy Enforcement it is verified whether the implementation of the changes affirms the top down changes intimated by the owner of the service. The source code level of CRM ensures Code Consistency, Time Complexity and Space Complexity. Code consistency is validated at source code level consequently after the changes the implemented. The non-functional level in CRM constitutes response time, service interruption time, availability and reliability. The response time is the time taken by a service or a functional unit to respond to the user’s requests when the service is invoked. The service interruption time is the time taken by the CMF to manage the change successfully. The reliability of the service is determined by the availability and fault tolerance capabilities of the service and it will be
verified before and after doing changes over the service. The availability indicates whether the service is available to all consumers and end users during the course of change process and this can be achieved through versioning. The constructed FSM traverses to the accurate position of the error. FSM are accountable for all schema level and source code level changes. The Change Authority Board (CAB) fixes the threshold for all the identified change factors through the periodic review. For all change factors there exists threshold and priority consummated by CAB. Consequently after the change properties are evaluated using property evaluator the fluctuations from the absolute property are found and when there are no fluctuations from the actual threshold value fixed by the CAB, the changes are inevitably committed. When there are negligible deviations the framework inquires the analyst whether to commit the changes or not. Thus guarantee is proffered for automation of the changes by the CMF.

3.4.3.3 Proactive Management

Forthcoming consequences are predicted by the proactive management (PM) based on the impact and behavioral analysis rendered on the services offered so far. PM is basically triggered through incident matching (i.e.) for a new change request it is verified that whether the change is feasible and productive by comparing with the impact analysis on the recorded changes previously accomplished. PM also reports: Risk due to Functional variations, Risk due to non-functional variations. An additional responsibility of PM is business growth rate analysis by means of recognizing the constructive and destructive causes due to the changes. For example when the changes required by a small group of customers are accomplished as per the change request requirements and brought into existence and then after some time some process is being executed unnecessarily for those customers who are not in demand of that process. This leads to increase in response time in excess. And when the entire set of customers report the same problem mentioned above the entire system may crash. Hence these sorts of disruptions are predicted by this proactive management beforehand in order to prevent such complications. Some of the indications reported by the PM are subsidence of number of customers and the reason for the subsidence. Apart from functional change factor deviation the non-functional deviation like QoS parameters variations can also be observed through proactive management. PM is also used to intimate security disputes. And PM is also used for notifying the CAB in prior to the service crash in administrative and service consumption level. Along with FSM,
Cellular Automaton (CA) is also used for impact analysis. Every cell in CA is mapped to automata of each business process so it’s convenient to backtrack and identify the changes. The benefits of using CA are pattern matching and dynamism.

The ultimate goal of CMF is to render Runtime Management, Change Reaction Management and Proactive Management. In support of Runtime Management, the property evaluation will be performed. The properties such as dependency, computability, configurability, accessibility and traceability will be evaluated and if all the properties are encountered to be fulfilled and manageable then the subsequent process of change evaluation will be carried out. This process uses CA calibration algorithm for evaluating the properties. The CRM determines who is responsible for handling the change request depending upon the types and severity of the change request. The two major levels in CRM are functional and nonfunctional. The future consequences due to the changes will be predicted by PM using incidence matching.

### 3.4.4 Execution Layer

In this execution layer first the schema is translated to its corresponding service and those change requests that require mere integration can be implemented with help of integration adapter, in which changes which can be automated by the CMF directly without any code injection by using the service integration technique and any one of the integration methods such as: union, composition, substitution and composition can be made used for service integration. Finally the resource file, configuration file and executable file are built together as components and they are deployed.

### 3.4.4.1 Schema to Source Code Translation

Once the property evaluation, change evaluation and the impact evaluation processes in the evaluation layer are completed and suggestions for committing the changes are found to be constructive, the successfully rendered schemas will be converted to equivalent source code by the CMF.
3.4.4.2 Integration Adapter

Some particular change requests requires merely integration of logics existing earlier which emerges the need for Service Integration. Changes which can be automatically done without any code injection comes under service integration. Change evaluation and impact evaluation is not necessary for these changes that require purely integration. Before and after integration the output generative power should be alike which indicates that the integration is successful. Properties serves as a pre request for service integration and these properties are evaluated before and after integration. These are the other four methods union, composition, substitution and composition which demands infinitesimal effort for producing the new logic as per the change request without any sort of code injection. The change request for integration is carried out at schema level. The services needed 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. Each time after the integration is being carried out a new WSDL file will be generated.

3.4.4.3 Build and Deployment

Resource file will be generated and if the equivalent source code of the logic after compilation is error free, it is automatically deployed. The CMF checks whether the resource file, configuration file and executable file are available and builds them together as components and finally they are deployed. Subsequently after the deployment, the appropriate updation will be made in the registry.

3.4.4.4 Version

As per the difference in change requests requirements put forth by different set of customers, versioning will be carried out accordingly. Through Versioning, we provide transparency to the customers due to which the customers will not be able to notice the changes that have been made. In case of any failure due to the changes, we can use versioning to roll back to the previous state. In this layer the successfully rendered schemas are converted to equivalent source code and when there are change request that requires service integration, integration adaptor is used for this purpose. Resource file will be generated and if the equivalent source code of the logic after compilation is error free, it is automatically deployed and the
corresponding updation will be made in the registry. Versioning will be done as per the requirements of the customer and it is also used for roll back purpose in case of failures.

3.4.5 Validation Layer

The responsibility of the validation layer is to inspect whether the standard procedures (i.e.) the special components in the CMF is validated or not. Earlier in the previously existing framework there was negligible focus on property and change factors that are to be evaluated. Whereas in this CMF additional properties and change factors have been recognized and in addition to that this framework also supports impact analysis where the forthcoming consequences are predicted by the proactive management (PM) based on the impact and behavioral analysis rendered on the services stored in the service repository. But in the previously existing framework, it has achieved only incident matching level as far as impact analysis is concerned and they used petri nets. While the CMF uses Cellular automata patters for impact analysis in proactive management and they adopt FSM mechanism.

The logic can be recognized where it is manageable or not by FSM. Errors will be identified for every change in the logic and will be indicated by the FSM as Meta information. Meta information is gathered from impact analysis on the entire set of incidences using CA patterns. Earlier frameworks which used change log didn’t have the facility to exactly trace the location where the change occurred, whereas CMF uses a structured change log which uses CA patterns and FSM which help to trace the location of the change occurrence exactly. The CMF uses validation table which stores information such as the number of case studies and the number of time the changes had been implemented. When changes are done iteratively in each respective time there is rapid increase in the performance level which is the major advantage in CMF.

3.4.5.1 Goal Assessment

The ultimate goal of CMF is to provide standardized procedure in its every aspect, which is mainly due to benefit of FSM which has formalism nature by default.

- **Degree of Authorization:** Without the aid of analyst, the degree of authorization is improved by the CMF.
• **Degree of Automation**: The number of change requests which are automated directly by the CMF is termed as “Degree of Automation”. The CMF has the complete potential to automate evaluations in Change and Property evaluation processes.

• **Average Change Computation Time**: The time taken for the overall change evaluation process is called change computation time and which needs to be reduced.

Thus Degree of Automation, Degree of Authorization and Average change computational time are compared with existing models, validated and the CMF recommends the best change which has the beneficial impact in future.

### 3.5 SUMMARY

Thus the Change Management Framework (CMF) comprising of storage subsystem and four layers has been briefly explained. In Analysis layer, change plan stage, the change request will be analyzed based on the domain and the context and the category to which the change request belongs to is identified. The final outcome of this layer is the schema. In Evaluation layer, the primary goal is to render Runtime Management, Change Reaction Management and Proactive Management. In Runtime Management property evaluation will be performed. The properties such as dependency, computability, configurability, accessibility and traceability will be evaluated and if all the properties are encountered to be fulfilled and manageable then the subsequent process of change evaluation will be carried out. This process uses CA calibration algorithm for evaluating the properties. In Execution layer, Resource file will be generated and if the equivalent source code of the logic after compilation is error free, it is automatically deployed and the corresponding updation will be made in the registry. At last the responsibility of the Validation layer is to inspect whether the standard procedures are being followed or not. Then the following attributes Degree of Automation, Degree of Authorization and Average change computational time are compared with existing models, validated and the CMF suggests whether the changes has positive impact or not. The additional properties and change factors have been identified and impact analysis is carried out using CA based incident matching. It avails FSM technique due to its formalism nature by default and finds out all the errors when the change is being applied to the logic with help of FSM along with Meta information gathered from the CA based incident matching.