Journal of King Saud University – Computer and Information Sciences (2016) 28, 211–229

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

A formal approach for change impact analysis of long term composed services using Probabilistic Cellular Automata

M. Thirumaran, M. Jannani [*], N. Sivakumar

Pondicherry Engineering College, Puducherry, India

Received 11 January 2014; revised 26 June 2014; accepted 23 October 2014
Available online 18 November 2015

KEYWORDS

Service oriented architecture;
Web services;
Change management;
Change impact analysis;
Finite State Machine;
Probabilistic Cellular Automata

Abstract Incorporating changes into the logics of composed services dynamically and successfully is a challenge for sustaining a business' image and profit in the society, especially when the change is expected to be made immediately at low cost. In this paper, we address this challenge by proposing a change impact analysis framework for long term composed services (LCS) which: (i) enables the business people to implement the changes by themselves through their analysts, (ii) reduces cost and time by eliminating the dependence on IT developers once the application services are developed and delivered, (iii) ensures effective incorporation of the changes made by using standard methodologies for evaluation – finite state automaton for verifying the runtime compatibilities and change evaluation and probabilistic cellular automaton for impact analysis and prediction. Through the evaluated probability measures and effective incident matching, the knowledge gained by the analyst over his service logics and the efficiency of incorporating changes are increased.

1. Introduction

Service Oriented Computing (SOC) is emerging as a new paradigm and is accruing the outsourcing of the required functionalities from third party web based providers especially through service composition. Composite services are built from aggregates of other autonomous services which collectively provide a value added service. In a long term composed service (LCS), the partnership among the services in composition and the business objective to be attained are for a long run. LCSs facilitate dynamic modification of the composition structure by choosing the best services and dissolving the services whose execution is no longer needed. They thus enable dynamic selection of the partners and aid the end users and consumers to be highly benefited from the open competition among the businesses. As the end users are always interested in latest techniques and technologies, the LCSs are more prone to changes, as time passes. Changes to an LCS can be of two types: top down changes or bottom up changes (Liu et al., 2011). In the former, the changes are initiated by the owners of the LCS and in the latter they are originated from the outsourced service providers. With the increase in the emanation of wide range of business competitors and the demands of

[*] Corresponding author.

Peer review under responsibility of King Saud University.

the end users, effective change management in LCSs has become vital. The dependence on the IT developers for the incorporation of these changes claims the investment of a considerable amount of time and cost which poses a significant threat to the business' income.

So there is a need for a framework which would enable the business people to make changes to the LCS through their analysts without the aid of the IT developers. Once a change occurs, the framework must enable the LCS to adapt itself quickly and automatically in order to satisfy the business need. However such changes should be incorporated systematically without any issues (Maamar et al., 2008) and hence change impact analysis which involves analyzing the impact which the incorporated change would have on the LCS, is essential and would increase the accuracy of the changes made. Though there are a lot of research works focusing on automatic composition and integration of the services, change impact analysis in web service change management using formal approaches has not been lime lighted much. The existing frameworks do not provide maximum runtime support and degree of automation and do not enable changes at the analyst level (Apostolou et al., 2010). They do not perform change impact analysis and incident matching which are very much essential for a timely and cost effective change management (Rovegard et al., 2008; Setzer et al., 2010; Chua and Aslam Hossain, 2012). There are frameworks for handling top down and for handling bottom up changes but not for both. There are no standard procedures and methodologies adopted in the existing works and even if the changes are made, there is no means to find the exact reason for the change being unmanageable in the current static scenario. In the current situation it is very difficult to incorporate the changes perfectly into the logics of the LCS. All these together have motivated our research question, "How can the change management process in long term composed services be enhanced such that the changes can be made by the analysts in a timely and cost effective manner and with reduced risk?".

In this paper, we have addressed this issue through the proposed change impact analysis framework which adopts formal methods and standard procedures for analyzing the impact of the changes through effective runtime support, change evaluation, constraint evaluation and QoS factor evaluation. When a request for making a change is received, the framework verifies if there are no runtime issues in the business logic extracted by property pre evaluation using Finite State Machine (FSM) and only then it allows the analyst to make the changes over the logic. After the changes are made, the change evaluation is done which verifies if the change made can be committed safely or not. This involves the checking over the deviation of the functionality of the logic after change through change factor evaluation, over the workflow of the logic through constraint factor evaluation and over the quality of the modified LCS using QoS factor evaluation for differentiating from competing service providers. All these evaluations are performed using FSM which provides the formalism required for verifying the changes made. The impact that the changes would have on the LCS is indicated through impact analysis and incident matching. The evaluated factors are compared with the threshold estimations and impact value estimations from the previous incidents and Cellular Automata (CA) is used to validate the evaluation and to generate the change factor, constraint factor and QoS factor patterns. The current changes are matched with the pre occurred, similar change request patterns followed by the behavioral analysis which is performed using Probabilistic Cellular Automata (PCA). PCA makes use of the knowledge extracted from these patterns and indicates the risk, degree of automation, accuracy and degree of incident matching involved which would aid the analyst in making the changes confidently and maximize the market attraction of the LCS.

2. Change impact analysis of LCS using Probabilistic Cellular Automata

In this section, we present our change impact analysis framework for managing changes in LCS using Probabilistic Cellular Automata. We first illustrate the working of the change impact analysis framework shown in Fig. 1 by means of the sequence of operations and activities involved in the change impact analysis and their dependencies depicted in Fig. 2. We then describe how the framework reduces the cost and time involved in effectively incorporating the changes with the help of a motivating example of a sample web service composition. The impact of a change, the construction of LCS schema, FSM and the associated PCA for the example are demonstrated manually. LCSs have attracted a lot of attention since they provide a powerful tool to offer value-added and customized services. These vividly contribute toward the need to standardize and fine tune the change management process for LCSs and to follow structured approaches to make error free changes in the functionality of the services without compromising the quality of the business process.

2.1. Overview of change impact analysis approach and framework

The importance of the change impact analysis approach can be adumbrated with the help of the following example. Consider a business analyst working on services in long term composition. In case of a situation where a change has to be made to the service logic immediately and at low cost, the analyst is put in a situation where he cannot wait for the development team and has to make the changes by himself. So, there are high chances for him to make a bug introducing change and inject incorrect statements into the logic of the LCS which might end with the changed LCS exhibiting an undesired behavior. Even if the changes are made as careful as possible, without analyzing the impact that the changes would have on the logic, it is not possible to assure a risk-free and accurate incorporation of the changes which in turn serves as a serious threat to meet the business outcome. This problem is addressed by the change impact analysis framework depicted in Fig. 1.

Initially the received request for change is sent to the change request manager to identify the part of the business process in which the change has to be made. The change request consists of the command to be executed i.e. the DML operations to be performed like an addition or a substitution etc., the resource to which the change is to be done expressed in terms of the process name and the condition to be followed i.e. performing the mentioned steps on the satisfaction of certain condition mentioned. Then, the source code of the composed service which can be in any language, to which the change is to be made, is extracted by analyzing the change

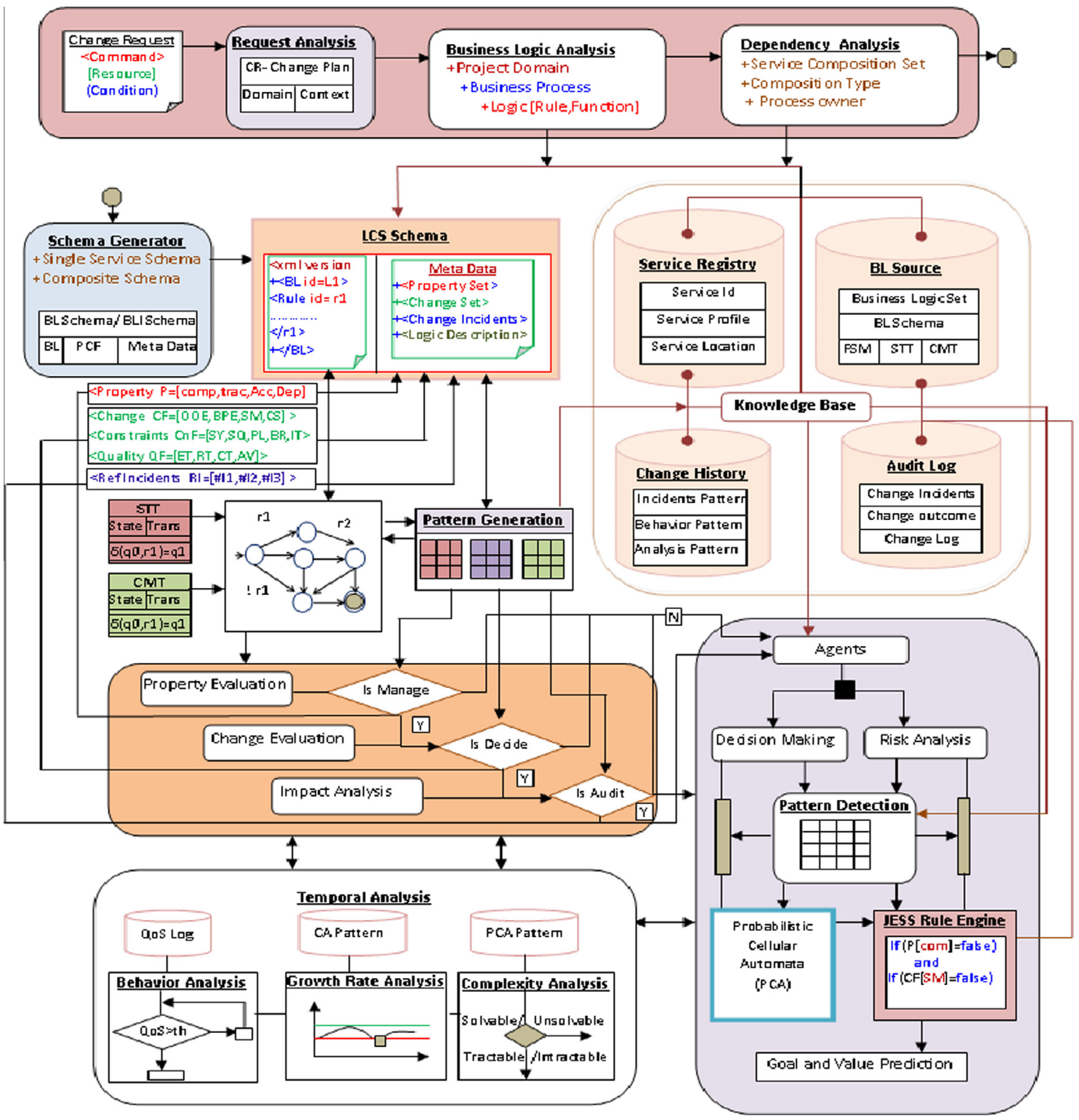
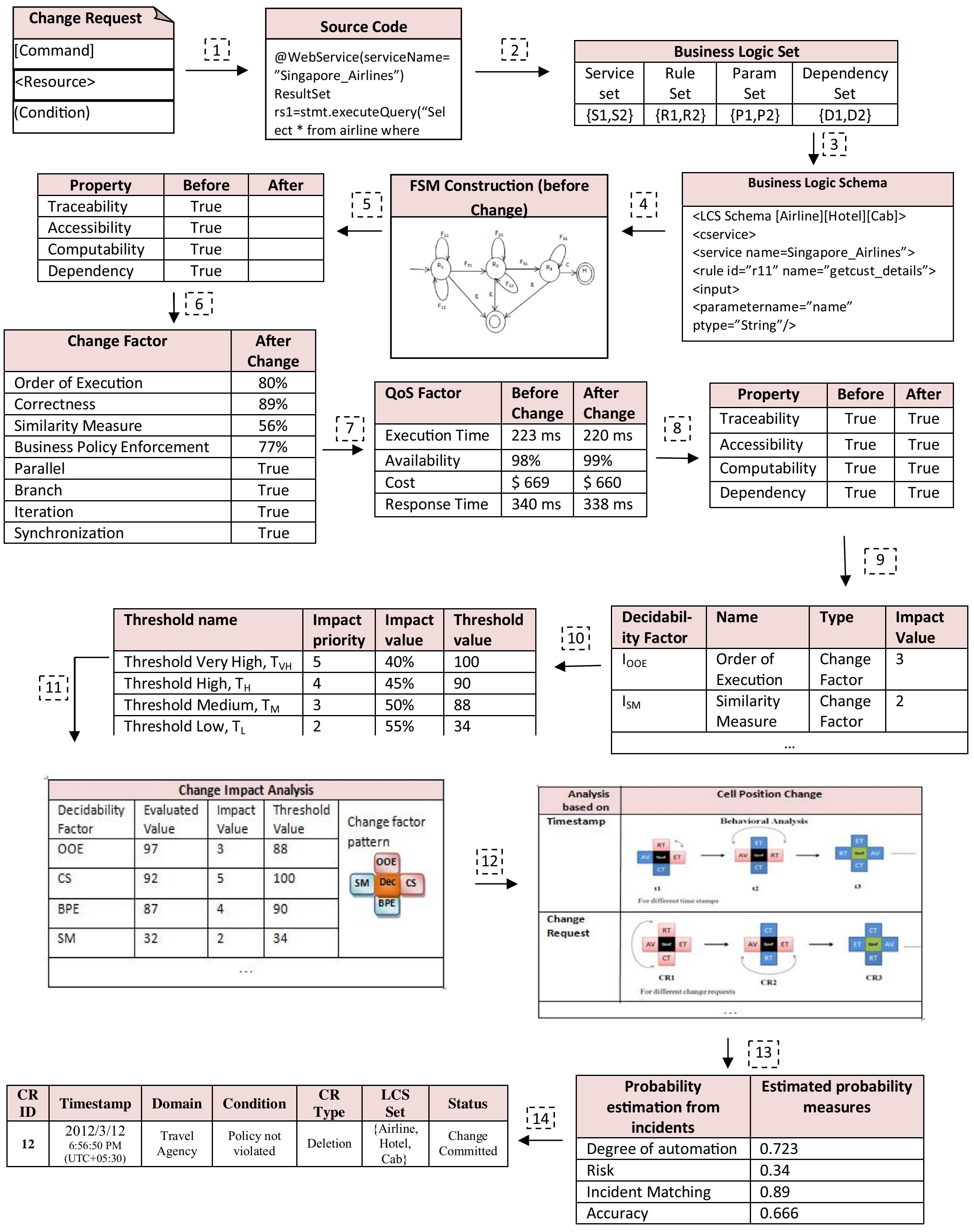


Figure 1 Change impact analysis framework.

request as shown in step 1 of Fig. 2. The domain analysis is performed to identify the business domain and the context analysis is performed to identify the business environment of the composed service. The business logic analysis involves the analysis of the project domain, business process and the business logic (rule, function).

This is followed by the dependency analysis. The entities are checked for any input dependency where two or more entities depend on the same input, output dependency where entities are expected to produce the same output, call sequence dependency, policy dependency where the entities are expected to abide by the same policy, mapping dependency where entities depend on the same data set for instance and existence dependency where the existence of an entity is meaningless in the absence of another particular entity. The service composition set, composition type and the process owner (policy dependencies are checked) are analyzed and checked. The service registry maintains information about the services including the service id, service profile and the service location. The business logic source maintains the business logic set, business logic schema and the corresponding Finite State Machine, state transition table and the change measure table.

After the dependency analysis, the rules, functions, parameters and dependency are identified and are placed in the business logic set as shown in step 2 of Fig. 2. Rules, their syntax and their description are identified. Functions are expressed in terms of primitive computable functions. Then the LCS schema before change is identified and generated by the schema generator as shown in step 3 of Fig. 2. Business logic is a group of the logic entities which includes rules, functions,



Change Request

[Command]

<Resource>

(Condition)

Source Code

```
@WebService(serviceName=
"Singapore_Airlines")
ResultSet
rs1=stmt.executeQuery("Sel
ect * from airline where
```

Business Logic Set

Service set	Rule Set	Param Set	Dependency Set
{S1,S2}	{R1,R2}	{P1,P2}	{D1,D2}

Business Logic Schema

```
<LCS Schema [Airline][Hotel][Cab]>
<cservice>
<service name=Singapore_Airlines">
<rule id="r11" name="getcust_details">
<input>
<parametername="name"
ptype="String"/>
```

FSM Construction (before Change)

Property	Before	After
Traceability	True	
Accessibility	True	
Computability	True	
Dependency	True	

Change Factor	After Change
Order of Execution	80%
Correctness	89%
Similarity Measure	56%
Business Policy Enforcement	77%
Parallel	True
Branch	True
Iteration	True
Synchronization	True

QoS Factor	Before Change	After Change
Execution Time	223 ms	220 ms
Availability	98%	99%
Cost	$ 669	$ 660
Response Time	340 ms	338 ms

Property	Before	After
Traceability	True	True
Accessibility	True	True
Computability	True	True
Dependency	True	True

Decidability Factor	Name	Type	Impact Value
I_{OOE}	Order of Execution	Change Factor	3
I_{SM}	Similarity Measure	Change Factor	2
...			

Threshold name	Impact priority	Impact value	Threshold value
Threshold Very High, T_{VH}	5	40%	100
Threshold High, T_H	4	45%	90
Threshold Medium, T_M	3	50%	88
Threshold Low, T_L	2	55%	34

Change Impact Analysis

Decidability Factor	Evaluated Value	Impact Value	Threshold Value	Change factor pattern
OOE	97	3	88	
CS	92	5	100	
BPE	87	4	90	
SM	32	2	34	
...				

Analysis based on	Cell Position Change
Timestamp	
Change Request	
...	

Probability estimation from incidents	Estimated probability measures
Degree of automation	0.723
Risk	0.34
Incident Matching	0.89
Accuracy	0.666

CR ID	Timestamp	Domain	Condition	CR Type	LCS Set	Status
12	2012/3/12 6:56:50 PM (UTC+05:30)	Travel Agency	Policy not violated	Deletion	{Airline, Hotel, Cab}	Change Committed

Figure 2 Workflow diagram of change impact analysis framework.

parameters and dependency. Every operation in a web service is rule based and is reflected by the schema. Every rule is converted into schema and this reduces the time taken in the generation of schema at runtime. The schema reflects the part of the logic under change only and hence is validated against the complete schema. After the schema validation is done, the corresponding FSM which provides better understandability and is language neutral is generated as shown in step 4 of Fig. 2. FSM being a conceptual and machine processable model provides enhanced focus on the part of the business process where the change has to be made by reflecting the schema. Any change made to the FSM is reflected back in the schema. While FSM consists of nodes, the schema consists of Meta data which are very useful when fetched during back-tracking. In order to fetch the Meta data, the information about the node is needed which is provided by the FSM. This provides better understandability to the business analyst and the analyst is facilitated to make the change which is then reflected back at the source code. For an analyst with little knowledge about the code, this facilitation would grant immense comfort. But still, the changes must be carefully evaluated, verified and only then reflected in the source code to avoid critical latent issues which is assured by the proposed framework. This is guaranteed by the framework by performing the Property evaluation, change factor evaluation, constraint factor evaluation, QoS factor evaluation and change impact analysis.

2.2. Runtime management

The property evaluator performs the process of evaluating the identified vital properties. The framework manages the changes dynamically without any exceptions and makes sure that the changes are made as expected, with the help of this property evaluator. When a part of the logic is extracted from an entire business process of the LCS, it tends to have compatibility issues to implement a change like its dependence on a rule, function, parameter of change. The extracted logic might not be complete with all required rules to implement the required change and some required rules or functions might be missing. The constructed FSM might reflect the rules, functions or even services which do not have the access permission to make a change or the rules might make a call to the rules of other services. All these issues need to be checked to set the manageability factor which indicates whether the extracted logic and the constructed FSM is complete reflecting the exact code to be changed and hence fit for a change process as shown in step 5 of Fig. 2.

Property evaluation is done all the time during the schema extraction, during the change process and even after the change is committed or discarded. The properties including computability, accessibility, traceability and dependency are evaluated over the FSM and the property evaluation tag is updated in the business logic schema. The evaluation of the computability property involves the check over the reachability and completeness of the logic reflected by the FSM. Let S_1, S_2, $S_3 \ldots S_n$ be the services in composition. Let M be the respective Finite State Machine associated with it. M is defined as $M = \{Q, \sum, \delta, q_o, F\}$. The transition function is given as, $\delta(Q_1, \sum_1) \rightarrow \{Q_2\}$, $\delta(Q_2, \sum_2) \rightarrow \{Q_3\}, \ldots \delta(Q_{n-1}, \sum_{n-1}) \rightarrow \{Q_n\}$, where $\sum_n$ is the set of input alphabets for the S_n for any n, such that Q_1 consumes the input set $\sum_1$, goes to Q_2 and finally Q_{n-1} consumes $\sum_{n-1}$ and reaches Q_n. If this is satisfied, reachability property is observed to be true. This verification makes sure that every node i.e. every rule or function represented in the FSM is reachable from every other rule or function. This identifies the lack of call to any rule or function of the logic under change. Completeness is associated with the single service wherein the transition function for any service S_n in M is given by, $\delta(q_n, \sum_n) \rightarrow F_n$, where $q_n \in Q_n$ and $F_n \subseteq Q_n$ is set of final states. This verification checks if there is a rule initiating the execution (start state) and for completing the execution (end state). The traceability property is verified in order to make sure that the business logic entity (service, rule, function or parameter) in the change request is traceable in the constructed FSM as the FSM reflects only the part of the logic under change. The services S_1, S_2, $S_3 \ldots S_n$ are said to be dependent on each other when the transition function is as $\exists n \quad \triangle(S_n, \sum_n) \rightarrow \delta(S_{n+e}, \sum_{n+e}) \bigcap \triangle(S_{n+e}, \sum_{n+e}) \rightarrow \delta(S_{n+d}, \sum_{n+d})$, where $e > 0$ and $d > 0$. The accessibility property is checked in order to verify that the business logic entities have the access permission to incorporate the change.

2.3. Change management

After the property evaluation before change is done, the analyst is allowed to perform functional changes to the logic. Then, the corresponding FSM after change is constructed and the change evaluation is done. The change evaluator performs the process of change factor evaluation, constraint factor evaluation and QoS factor evaluation. This assures that the changes made to the business logic are in accordance with the change request, meeting the expectation of the analyst and that the changes are injected correctly into the logic.

2.3.1. Change factor evaluator

Change factors are envisaged as the touchstones for the evaluation of the changes made. Any change in the business logic is influenced by the change factors either independently or in a group and they aid in making the changes effectively. The vital factors for change factor evaluation are listed in Table 1. Order of execution is given by $\mathrm{OOE} = \left[\sum\left(\frac{\Delta R_{id}}{R}\right) \cup \sum\left(\frac{\Delta F_{id}}{F}\right)\right] * 100$, where R_{id}, F_{id} are the unique ids which establish the order of the rule and the function within the logic L. Similarity measure is estimated by $\mathrm{SM} = \sum\left(\frac{\Delta I_P / \Delta O_P}{N}\right) * 100$, where ΔI_P and ΔO_P, refer to the number of states in which the inputs and outputs have varied respectively and N is the total number of states generated for L in FSM, M (Rovegard et al., 2008). Business Policy Enforcement is given by $\mathrm{BPE} = \sum\left(\frac{F_P}{P}\right) * 100$, where F_P is the states in which the business policies failed and P are the total states related to business policies in M. Correctness is given by $\sum\left[1 - \left(\frac{C_d}{T_s}\right)\right] * 100$, where C_d is the number of dangling states and T_s is the total number of states in M. The evaluation of these change factors indicates the deviations caused in the functionality of the business logic due to the change as shown in step 6 of Fig. 2.

2.3.2. Constraint factor evaluator

Constraint factors play a vital role in the evaluation of the changes made by evaluating the changes with respect to the invocation order and the work flow order which is significantly important in case of composed services. Synchronization, Sequence, Branch, Parallel and Iteration are the constraint factors considered for the functional changes made to the business logic. Let S_1, S_2, $S_3 \ldots S_n$ be the set of n services in composition comprising of rule sets such as $S_1 = \{r_1, r_2,$

Table 1 Evaluation factors and their purpose.

Evaluation phase	Component	Factors evaluated	Purpose
Runtime management	Property evaluator	Computability Traceability Accessibility Dependency	In order to provide runtime support by checking if the FSM reflects the exact logic under change and is free from compatibility issues before and after change
Change management	Change factor evaluator	Order of execution Correctness Similarity measure Business policy enforcement	In order to evaluate the changes made to the functionality of the business logic
	Constraint factor evaluator	Sequence Parallel Branch Iteration	To evaluate the changes made with respect to the invocation and work flow order
	QoS factor evaluator	Response time Execution time Cost Availability	In order to verify the impact that the change has over the quality of the business process
Change impact analysis	Behavior analyzer	Probability of risk Probability of accuracy Probability of incident matching Probability of degree of automation	To deduce the measures for decision making and analyze the behavior of the composed service after change using PCA

$r_3 \ldots r_p\}$, $S_2 = \{r_p, r_{p+1}, r_{p+2} \ldots r_q\}, \ldots S_n = \{r_{q+1}, r_{q+2}, r_{q+3} \ldots r_r\}$. These services are said to be synchronized with each other $(S_1 \cup S_2 \cup S_3 \ldots S_n)$ if and only if $S_1 \cup S_2 = \emptyset$, $S_2 \cup S_3 = \emptyset \ldots S_{n-1} \cup S_n = \emptyset$. The sequence is said to be observed in LCS if and only if, $\exists n\ \delta^*(S_n, \Delta S_n) => S_{n+e} \cap \delta^*(S_{n+e}, \Delta S_{n+e}) => S_{n+d}$ where $\delta^*(S_n, \Delta S_n) => S_{n+e}$ indicates the transition of S_n to S_{n+e} taking ΔS_n input in multiple steps, $e > 0$ and $d > 0$, ΔS_n represents the output of S_n. Let S_1, S_2, $S_3 \ldots S_n$ be n services possessing the states Q_1, Q_2, $Q_3 \ldots Q_n$. The branch is said to be observed in the LCS if and only if, $\delta^*(q_n, L_s) \rightarrow q_{n+d}$, $\delta^*(q_n, L_p) \rightarrow q_{n+e}$, where $s \cap p = \emptyset$, $d \cap^e = \emptyset$ and $d > 0$, $e > 0$. Let us consider two parallel services possessing the states Q_a and Q_b respectively and the parallel execution is observed between them if and only if, $\exists n\ \{[\delta(q_n, L_s) \rightarrow q_{n+d} \cap\ \delta(q_n, L_p) \rightarrow q_{n+e}] \cap [\delta(q_{n+d}, L_s) \neq\ > q_{n+e} \cap \delta(q_{n+e}, L_p) \neq\ > q_{n+d}]\}$ where q_{n+d}, $q_{n+e} - q_{n+d} \in Q_a$ and $q_{n+e} \in Q_b$. q_{n+d} should not reach any of the states in set Q_b and q_{n+e} should not reach any of the states in set Q_a. For any service n iteration is said to be observed if and only if $\exists n\ \delta^*(q_n, L) => q_n$ i.e. if any state q_n in a service reaches the same state q_n using the logic L then iteration is said to be observed. The factors are evaluated and updated as shown in step 6 of Fig. 2.

2.3.3. *QoS factor evaluator*

QoS factors influence the changes made to the composite service in the idea that any service in composition with devastating variations in its non functional behavior after change is not preferable as it might result in the failure to meet the business competitors (Xiong et al., 2009; Wang et al., 2009). So when a change has been made to the logic, the quality metrics pave the way for making the behavioral analysis and committing the change. Response time, execution time, cost and availability are the vital QoS factors evaluated. Web service availability is the ratio of the expected value of the uptime of a service to the aggregate of the expected values of up and down time. Response time is the measure of how long it takes for a service to respond from the time of invocation of the service. Execution time is the measure of the time taken for the complete execution of a rule or service in composition. Cost is measured from the estimation of the execution time of the rule or service in composition. The QoS factors are evaluated and updated as shown in step 7 of Fig. 2. After these evaluations are done, the property evaluation is done again as shown in step 8 of Fig. 2 and the change evaluation values are updated as reference points in the business logic schema before the change is committed. Based on these values, the impact analysis is done. Temporal analysis is performed which involves the behavior analysis and growth rate analysis which focuses on the profit or loss gained from the composition.

3. Change impact analysis

The change, constraint and the QoS factors together are called the decidability factors which determine whether the change made can be committed or not through the estimation of decidability. For every change request once the decidability factors are measured, they are considered for impact analysis. The amount of impact (impact factor) of each decidability factor on the logics is determined first. The generalized formula for determining impact factor is: $I_f = \sum(D_{fv} * I_v)/ND_f$ where, I_f – impact factor, D_{fv} – decidability factor value, I_v – impact value, ND_f – total number of decidability factors. The decidability factor values (D_{fv}) are taken from the measures done in the change, constraint and the QoS evaluators. The impact values (I_v) for each decidability factor is provided in Table 2. Impact value is the priority of impact that a particular decidability factor has on the business logic which is estimated from the behavior of the factor in the previous similar incidents. The decidability factors are rated from 1 to 5. For example consider the factors correctness, sequence and response time. They are of higher importance to logic than the other factors because they have been the reason for the failure in the previous similar incidents. Hence they are rated with highest impact value 5. These impact values can also be given by the analyst according to his business and the prediction based on probability estimation. This is shown in step 9 of Fig. 2.

For every impact value (1–5), only a particular percentage of allowance is allowed to act on the logic while making a change. For example, if a decidability factor's impact value is 5, then it is of higher impact i.e. the factor has been the

Table 2 Impact values of the decidability factors.

Decidability factor	Factor type	Impact value from incidents
Order of execution, I_{OOE}	Change factor	3
Similarity measure, I_{SM}	Change factor	2
Business Policy Enforcement, I_{BPE}	Change factor	4
Correctness, I_{CS}	Change factor	5
Synchronization, I_{SY}	Constraint factor	4
Branch, I_{BR}	Constraint Factor	3
Parallel, I_{PL}	Constraint factor	2
Sequence, I_{SQ}	Constraint factor	5
Iteration, I_{IT}	Constraint factor	1
Availability, I_{AV}	QoS factor	2
Execution time, I_{ET}	QoS factor	4
Response time, I_{RT}	QoS factor	5
Cost, I_{CT}	QoS factor	3

reason for the failure of a change in the maximum number of previous incidents and hence only 40% of its impact is allowed to act on the logic where as if a decidability factor's impact value is 1, it is of lower impact hence 60% of its impact is allowed to act on the logic. This allowance of impact is termed as the threshold in the impact analysis part. The threshold calculation for each kind of impact value is listed in Table 3. This is shown in step 10 of Fig. 2. Threshold estimation, T_i is based on the following: $T_i = D_f(1 + IA)$, where, D_f = maximum value of the decidability factor from successful incidents, IA = percentage of impact allowed for that threshold type i.

As the impact values are rated based on their effect on logics, the impact factor determination is also classified according to the rating. Hence the impact factor for the decidability factors having impact value 5 is calculated as Impact_Very_High and abbreviated as I_{VH}, impact value 4 is calculated as Impact_High and abbreviated as I_H, impact value 3 is calculated as Impact_Medium and abbreviated as I_M, impact value 2 is calculated as Impact_Low and abbreviated as I_L and

Table 3 Threshold values for impact values.

Threshold name	Impact value	Percentage of impact allowed (%)
Threshold_Very_High – T_{VH}	5	40
Threshold_High – T_{H}	4	50
Threshold_Medium – T_{M}	3	50
Threshold_Low – T_{L}	2	55
Threshold_Very_Low – T_{VL}	1	60

impact value 1 is calculated as Impact_Very_Low and abbreviated as I_{VL} based on the formulation. The impact values determined are compared with the threshold values and if all the change, constraint and QoS factor values are within the limit of allowance indicated by the successfully satisfied change, constraint and QoS factor patterns, then that particular change request's decidability changes to '1'.

Else, the decidability is set to '0'. The decidability algorithm is given in Fig. 3. The algorithm adumbrates the decidability calibration by analyzing the impact analysis set I. Every decidability factor value CFV is compared with the maximum successful decidability factor values CFM and if found greater that the maximum value, is checked for its corresponding impact factor. The threshold values are computed with respect to the impact factor values. For instance, threshold high is computed if the impact factor belongs to impact high. Based on this decidability is calibrated.

When the decidability is changed to '1', the various cellular automaton patterns are generated based on this decidability and manageability for rules, functions and parameters as shown in step 11 of Fig. 2 and the approval is sent to the runtime manager.

3.1. Estimation of probability through incident matching

The various cellular automata patterns including the property, change, request, QoS, rule, function, parameter, and incident patterns generated act as the knowledge base to aid in knowledge discovery and decision making. The probabilities of risk, accuracy, automation, and incident matching are determined and from the extracted knowledge, the functional deviations, variations in QoS, violations in SLA and policy and failure are discovered. Probability of accuracy indicates the extent to which the evaluations are accurate. This is estimated based on the following formulation, $P(\text{accuracy}) = \left[\sum_{x=1}^{n}\text{ConEval}(x) + \sum_{x=1}^{n}\text{ChangeEval}(x) + \sum_{x=1}^{n}\text{QoSEval}(x)\right]/(3*n)$.

where ConEval(x) is the result of evaluation of the constraint factors for incident x by the cellular automata, ChangeEval(x) is the result of evaluation of the constraint factors for incident x by the cellular automata, QosEval(x) is the result of evaluation of the constraint factors for incident x by the cellular automata and n is the total number of incidents recorded. Probability of risk indicates the amount of risk involved in the evaluations made. This is estimated based on the following formulation,

$$P(\text{risk}) = 1 - \left\{\sum_{x=1}^{n}\left[\frac{\sum_{i=1}^{5}\text{ConRes}_i(x)*\text{ConPr}_i(x)}{\sum_{i=1}^{5}\text{ConPr}_i(x)} + \frac{\sum_{i=1}^{4}\text{ChangeRes}_i(x)*\text{ChangePr}_i(x)}{\sum_{i=1}^{4}\text{ChangePr}_i(x)} + \sum_{x=1}^{n}\text{QoSEval}(x)\right]\middle/(3*n)\right.$$

where ConRes$_i$(x) is the value of the constraint factor i in incident x in the cellular automata, ChangeRes$_i$(x) is the value of the change factor i in incident x in the cellular automata, ConPr$_i$(x) is the priority for the constraint factor i of incident x, QosEval(x) is the result of evaluation of the constraint factors for incident x by the cellular automata, Change Pr$_i$(x) is

```
Algorithm Decidability_Calibration ( Impact Analysis set I )
Begin
For every $CFV_i$ do
If ($CFV_i$ == $CFM_i$) // In case that the computed value of the factor is already less than the maximum value
Increment count by one
End If
Else
While $IV_i \in I_{VH}$ // For Impact Very High type of factors
Compute $T_{VH}= CFM_i * (1- IA_{VH})$;
If ($CFV_i >= T_{VH}$) // In case that the computed value of the factor is not below the threshold
Increment count by one
End If
End While
While $IV_i \in I_H$ // For Impact High type of factors
Compute $T_H= CFM_i * (1- IA_H)$;
If ($CFV_i >= T_H$) // In case that the computed value of the factor is not below the threshold
Increment count by one
End If
End While
While $IV_i \in I_M$ // For Impact Medium type of factors
Compute $T_M= CFM_i * (1- IA_M)$;
If ($CFV_i >= T_M$) // In case that the computed value of the factor is not below the threshold
Increment count by one
End If
End While
While $IV_i \in I_L$ // For Impact Low type of factors
Compute $T_L= CFM_i * (1- IA_L)$;
If ($CFV_i >= T_L$) // In case that the computed value of the factor is not below the threshold
Increment count by one
End If
End While
End For
If (count = = $N_{CF}$)
Set Decidability value to one;
Else
Set Decidability value to zero;
End If
End
```

Figure 3 Decidability calibration algorithm.

the priority for the constraint factor i of incident x, Probability of degree of automation is the estimation of the extent to which correct and smooth changes are made by automated decisions made.

$$P(\text{degree of automation}) = \frac{\sum_{x=1}^{n}[\text{PropEval}(x) * \text{ConEval}(x) * \text{ChangeEval}(x) * \text{QoSEval}(x)]}{n}$$

where ConEval(x) is the result of evaluation of the constraint factors for incident x by the cellular automata, ChangeEval(x) is the result of evaluation of the constraint factors for incident x by the cellular automata, QosEval(x) is the result of evaluation of the constraint factors for incident x by the cellular automata and n is the total number of incidents recorded.

Probability of incident matching is the estimation of the extent to which incidents are matched. This is estimated based on the following formulation,

$$P(\text{incident matching}) = \frac{\sum[(PR_C \epsilon PR_P) + (PS_C \epsilon PS_P)]}{PR_P + PS_P}$$

where PR_C is the possible rule combinations for received rule, PS_C is the possible service combinations for received composition set, PR_P existing rule combinations for received rule, PS_P existing service combinations for received composition set.

```
Algorithm BEHAV_ANAL($C_F$, $Diff_i$, $CD'$, $D_i$, i)

Begin
For all $CR_N$ do
    Determine P(risk) from $CR_{N-1}$
    If P(risk) > Thr_Value then
        For all $CD_N'$ do
            For every decidability factor i do
                Estimate $Diff_i := Diff_i + (T_i - D_i)$
            End For
            Select i with greater $N*T_i - Diff_i$
            If $I_i < 5$ then
                Increase $I_i$ by 1
                Change cell positions according to transition rule
            Else
                Consider CAB recommendations for revising IA for the threshold types
            End If
        End For
    End If
End For
End
```

Figure 4 Algorithm for behavioral analysis.

This is followed by behavioral analysis using Probabilistic Cellular Automata.

3.2. Behavioral analysis using Probabilistic Cellular Automata

The Probabilistic Cellular Automata is a stochastic cellular automaton where two rules are followed synchronously for the updation of the cells. One of the rules applied here is for the determination of the decidability of the automaton by comparing every cell state with the threshold. The second rule is for the determination of the cell positions dynamically based on the probabilistic measures. So, the states of the new entities are chosen according to some probabilistic distribution and behavioral analysis is thus performed and the predictions are made. The algorithm for behavioral analysis is shown in Fig. 4. The algorithm adumbrates the dynamic change of cell positions based on the probability measures. Considering all change requests, CR, the probability of risk is estimated from the recorded incidents and is compared with the threshold value which in turn is estimated through incident matching. If the probability value exceeds the threshold, then it indicates majority of the recorded incidents have failed to satisfy the decidability criterion. So, the impact ratings, priorities and the cell positions of the factors have to be revised. Therefore, the change requests with decidability failed, CD' are extracted and the total sum of the difference between threshold and decidability factor value of all unsuccessful incidents is estimated for every decidability factor *i*. Then the decidability factor with greater deviation from the threshold is chosen and the impact priority of the factor is increased. The decidability is verified and the cell positions are dynamically and synchronously changed by the Probabilistic Cellular Automata according to the transition rule as shown in step 12 of Fig. 2.

3.2.1. Transition rule

The transition rule followed by the PCA for changing the cell positions dynamically is elucidated with the aid of Fig. 5. The impact values at the top position, X_1, at the right position X_2, at the bottom position X_3, and at the left position X_4 are compared with the impact values configured for the cells, O_1, O_2, O_3, O_4, at the respective positions. The transition rule applied for the PCA is shown below:

```
Transition Rule for PCA:
If ($X_N$ > O3 && $X_N$ < O1) then
    Move cell position of $X_N$ to $X_2$ (right)
End If
If ($X_N$ > O4 && $X_N$ < O2) then
    Move cell position of $X_N$ to $X_3$ (bottom)
End If
If ($X_N$ < O3) then
    Move cell position of $X_N$ to $X_4$ (left)
End If
If ($X_N$ > O2) then
    Move cell position of $X_N$ to $X_1$ (top)
End If
```

The transition rule is elucidated in the following Fig. 5. On backtracking from cell X_4, the corresponding FSM before and after change is extracted and this makes it possible to trace back any point of change.

The change history stores the incident property, change factor, constraint factor and QoS factor patterns and the audit log keeps track of the change incidents, change outcome and the change log. From this rich knowledge base, the agent performs decision making and risk analysis (Tjoa et al., 2011; Marschall et al., 2012). The patterns are detected and the Probabilistic Cellular Automata performs the behavioral analysis by estimating the probability of accuracy, risk, automation and incident matching (Plebani and Pernici, 2009) by employing the JESS rule engine for the purpose of decision making and this ultimately leads to the goal and value prediction.

4. Experimental results

This section elucidates the change management scenario with the help of an application from a business domain. Further,

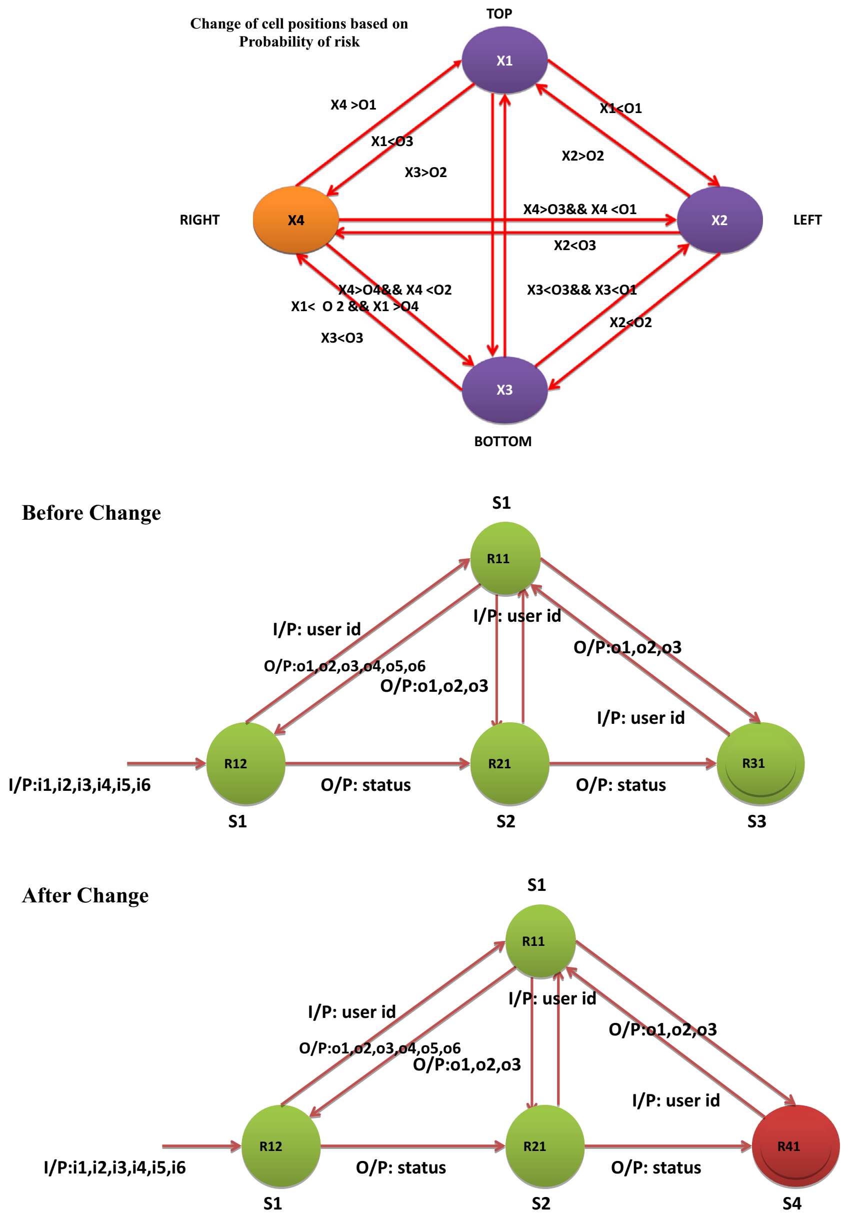


Figure 5 Rule followed for cell positioning and behavioral analysis.

the results of change impact analysis and incident matching approach over the LCS have been depicted.

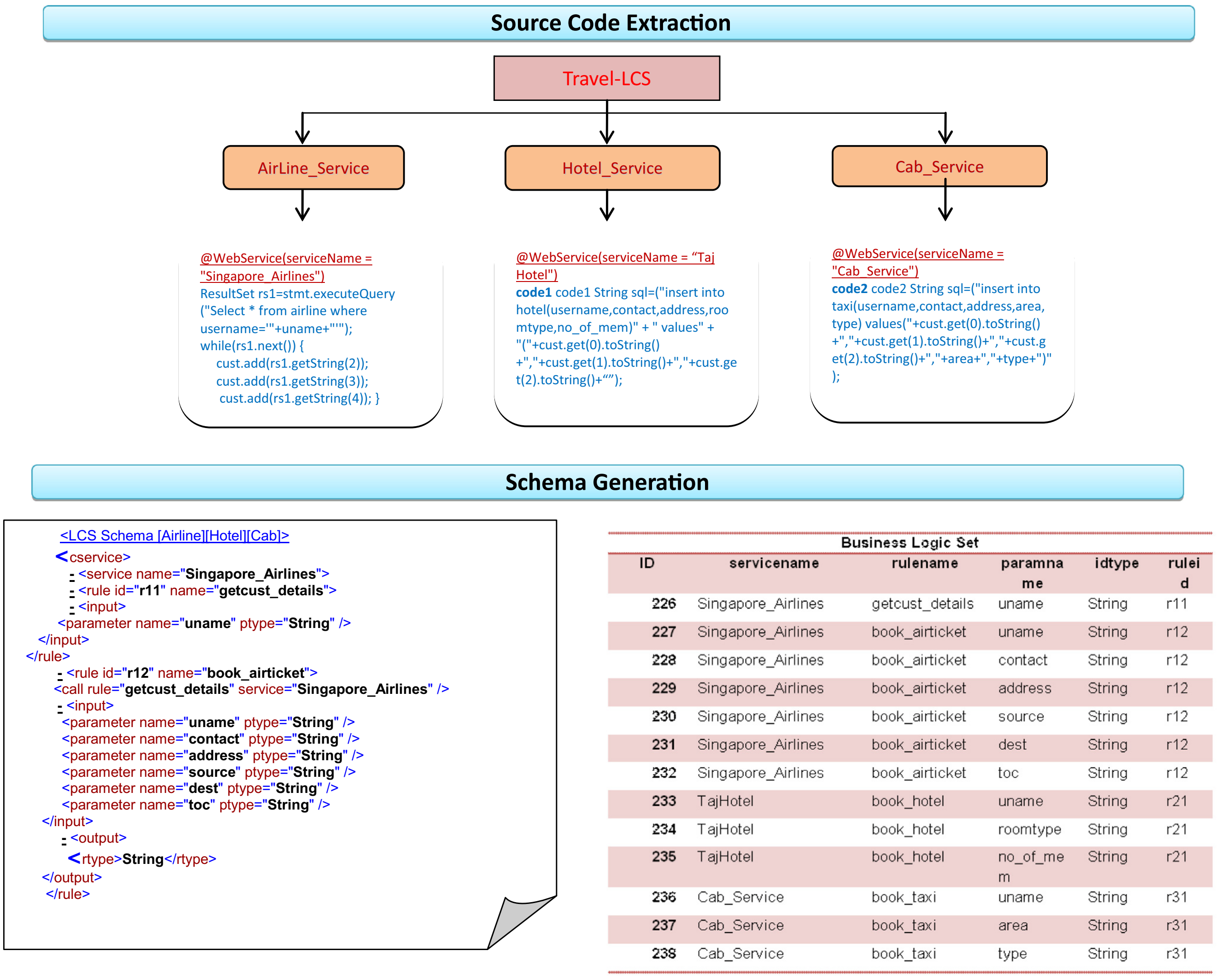
4.1. Change management scenario

In this section, an application from the travel domain is illustrated for elucidating the change management and change impact analysis process. Consider a travel agency LCS that offers many types of functionalities airline, hotel and cab service. The composed service offers submission of the customer details – get_cust_details, booking the air ticket – book_airticket, booking the hotel – book_hotel and booking the cab – book_cab executing in sequence.

Now, consider a change request which demands the removal of 'book_airticket'. This removal has to be done very cautiously since the execution of the other rules and services must not be affected. On receiving a change request, the change impact analysis framework extracts the corresponding source code based on the domain and context analysis as shown in Fig. 6. After this, dependency analysis is done and the business logic set is formed. The corresponding composed partial schema before change is then built and validated with the complete composed schema.

The schema under consideration is termed as a partial schema because it reflects only the part of the business logic under change while the complete composed schema reflects the entire business process. After successful schema validation, the Finite State Machine before change is constructed as shown in Fig. 7. The property pre evaluation is performed over this FSM and the results are shown in Fig. 8 calibrating the manageability.

On successful manageability calibration, the business analyst is allowed to make a change. There are various criteria considered before the deletion process. A rule in the FSM can-



```
@WebService(serviceName =
"Singapore_Airlines")
ResultSet rs1=stmt.executeQuery
("Select * from airline where
username='"+uname+"'");
while(rs1.next()) {
    cust.add(rs1.getString(2));
    cust.add(rs1.getString(3));
     cust.add(rs1.getString(4)); }
```

```
@WebService(serviceName = "Taj
Hotel")
code1 code1 String sql=("insert into
hotel(username,contact,address,roo
mtype,no_of_mem)" + " values" +
"("+cust.get(0).toString()
+","+cust.get(1).toString()+","+cust.ge
t(2).toString()+"");
```

```
@WebService(serviceName =
"Cab_Service")
code2 code2 String sql=("insert into
taxi(username,contact,address,area,
type) values("+cust.get(0).toString()
+","+cust.get(1).toString()+","+cust.g
et(2).toString()+","+area+","+type+")"
);
```

```
<LCS Schema [Airline][Hotel][Cab]>
<cservice>
  - <service name="Singapore_Airlines">
  - <rule id="r11" name="getcust_details">
  - <input>
 <parameter name="uname" ptype="String" />
</input>
</rule>
 - <rule id="r12" name="book_airticket">
 <call rule="getcust_details" service="Singapore_Airlines" />
 - <input>
  <parameter name="uname" ptype="String" />
  <parameter name="contact" ptype="String" />
  <parameter name="address" ptype="String" />
  <parameter name="source" ptype="String" />
  <parameter name="dest" ptype="String" />
  <parameter name="toc" ptype="String" />
</input>
  - <output>
   <rtype>String</rtype>
</output>
 </rule>
```

Business Logic Set

ID	servicename	rulename	paramname	idtype	ruleid
226	Singapore_Airlines	getcust_details	uname	String	r11
227	Singapore_Airlines	book_airticket	uname	String	r12
228	Singapore_Airlines	book_airticket	contact	String	r12
229	Singapore_Airlines	book_airticket	address	String	r12
230	Singapore_Airlines	book_airticket	source	String	r12
231	Singapore_Airlines	book_airticket	dest	String	r12
232	Singapore_Airlines	book_airticket	toc	String	r12
233	TajHotel	book_hotel	uname	String	r21
234	TajHotel	book_hotel	roomtype	String	r21
235	TajHotel	book_hotel	no_of_mem	String	r21
236	Cab_Service	book_taxi	uname	String	r31
237	Cab_Service	book_taxi	area	String	r31
238	Cab_Service	book_taxi	type	String	r31

Figure 6 Source code extraction, schema generation and business logic set extraction for travel LCS.

not be deleted if it is being called by any other rule or if it is the only call to any other rule or if there is no link between the previous and the next rule or service other than the rule to be deleted. Only then, the analyst is allowed to make the initial move toward deletion. This aids in reducing the risk involved in making the change in addition to the change evaluation process. 'book_airticket' is neither called by any other rule nor is the only call to any other rule.

It is not the only link between the previous and the next rule either. So, the analyst is allowed to make the change at the schema level. The constructed FSM after the incorporation of the deletion of 'book_airticket' is shown in Fig. 9. Fig. 10 depicts the change process and the schema and FSM before and after change.

Before the change is committed, the change factor evaluation, constraint factor evaluation and QoS factor evaluation for the travel agency application are performed and the corresponding decidability factor values are estimated. This is followed by change impact analysis where the previously recorded similar change incidents are extracted and the impact value estimations are done as shown in Fig. 11. The threshold estimations and the impact factor estimations are also done and the probabilistic measures are estimated for behavioral analysis. PCA then performs the dynamic updation of the cell states followed by the decidability calibration. The change is then committed on successful decidability calibration and rolled back if the calibration fails. The impact analysis done based on the estimated decidability factor values is shown in Fig. 11. Probability of risk is estimated as P (risk) = 0.233. Probability of accuracy is estimated as P (accuracy) = 0.7667, Probability of degree of automation as P (aut) = 0.723 and probability of incident matching as P (inc) = 0.89. Fig. 12 shows the results of impact analysis for 50 change requests to travel LCS for different impact factors. The blue curve indicates the decidability factor value of the particular factor belonging to the corresponding impact factor category and the red curve indicates the corresponding threshold estimations.

Fig. 13 shows the change in cell position based on the probability function from previous neighboring states at different time stamps and for different change requests. Figs. 14–17 show the improvement observed in the response time, execution time, cost and availability after change is made with change impact analysis. The blue curve indicates the

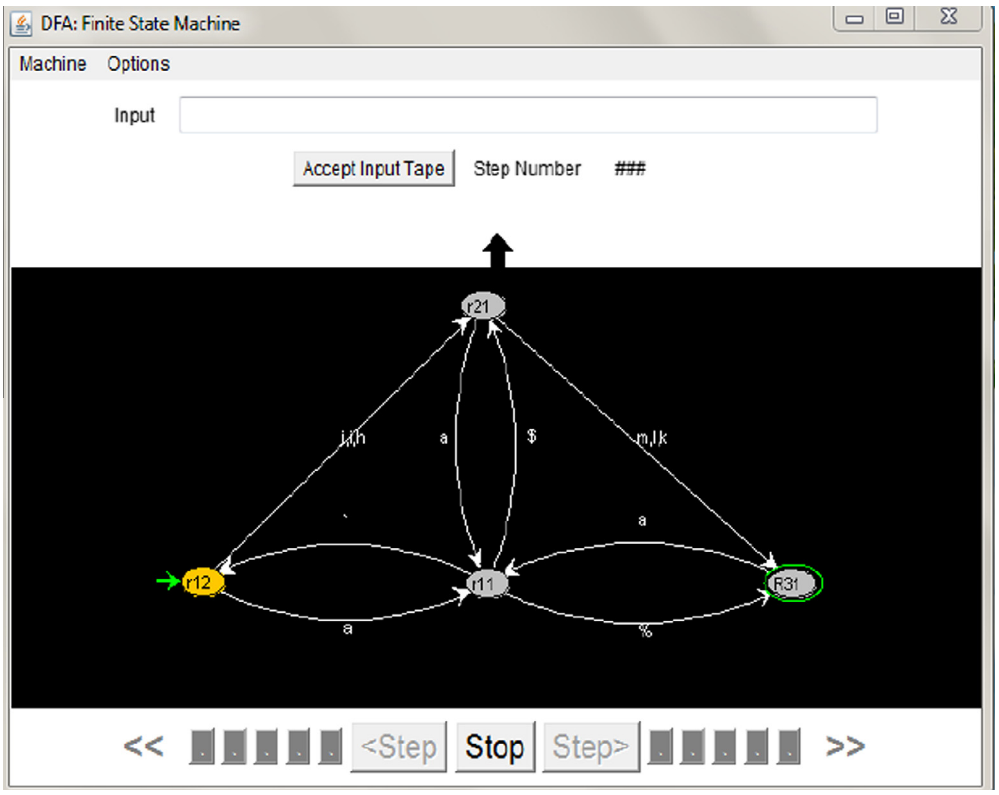
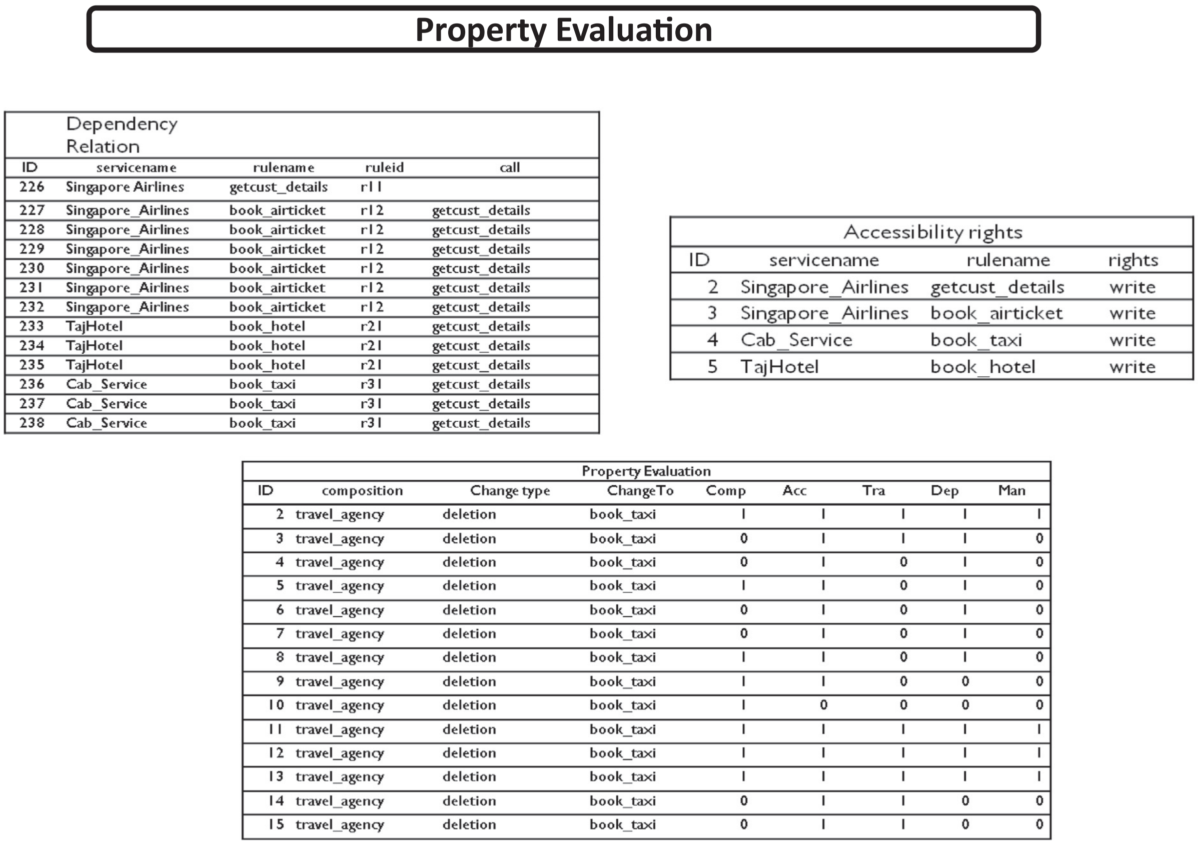


Figure 7 Constructed FSM before change for travel LCS.



Property Evaluation

	Dependency Relation			
ID	servicename	rulename	ruleid	call
226	Singapore Airlines	getcust_details	r11	
227	Singapore_Airlines	book_airticket	r12	getcust_details
228	Singapore_Airlines	book_airticket	r12	getcust_details
229	Singapore_Airlines	book_airticket	r12	getcust_details
230	Singapore_Airlines	book_airticket	r12	getcust_details
231	Singapore_Airlines	book_airticket	r12	getcust_details
232	Singapore_Airlines	book_airticket	r12	getcust_details
233	TajHotel	book_hotel	r21	getcust_details
234	TajHotel	book_hotel	r21	getcust_details
235	TajHotel	book_hotel	r21	getcust_details
236	Cab_Service	book_taxi	r31	getcust_details
237	Cab_Service	book_taxi	r31	getcust_details
238	Cab_Service	book_taxi	r31	getcust_details

Accessibility rights			
ID	servicename	rulename	rights
2	Singapore_Airlines	getcust_details	write
3	Singapore_Airlines	book_airticket	write
4	Cab_Service	book_taxi	write
5	TajHotel	book_hotel	write

Property Evaluation								
ID	composition	Change type	ChangeTo	Comp	Acc	Tra	Dep	Man
2	travel_agency	deletion	book_taxi	1	1	1	1	1
3	travel_agency	deletion	book_taxi	0	1	1	1	0
4	travel_agency	deletion	book_taxi	0	1	0	1	0
5	travel_agency	deletion	book_taxi	1	1	0	1	0
6	travel_agency	deletion	book_taxi	0	1	0	1	0
7	travel_agency	deletion	book_taxi	0	1	0	1	0
8	travel_agency	deletion	book_taxi	1	1	0	1	0
9	travel_agency	deletion	book_taxi	1	1	0	0	0
10	travel_agency	deletion	book_taxi	1	0	0	0	0
11	travel_agency	deletion	book_taxi	1	1	1	1	1
12	travel_agency	deletion	book_taxi	1	1	1	1	1
13	travel_agency	deletion	book_taxi	1	1	1	1	1
14	travel_agency	deletion	book_taxi	0	1	1	0	0
15	travel_agency	deletion	book_taxi	0	1	1	0	0

Figure 8 Property evaluation results for travel LCS.

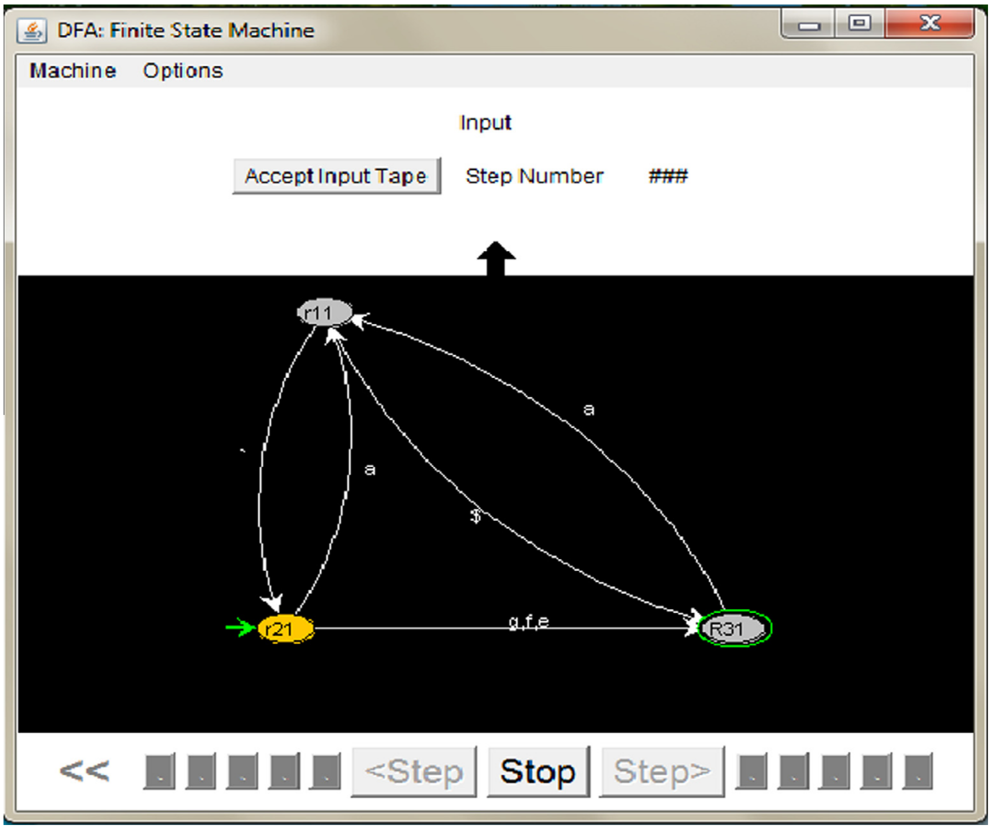
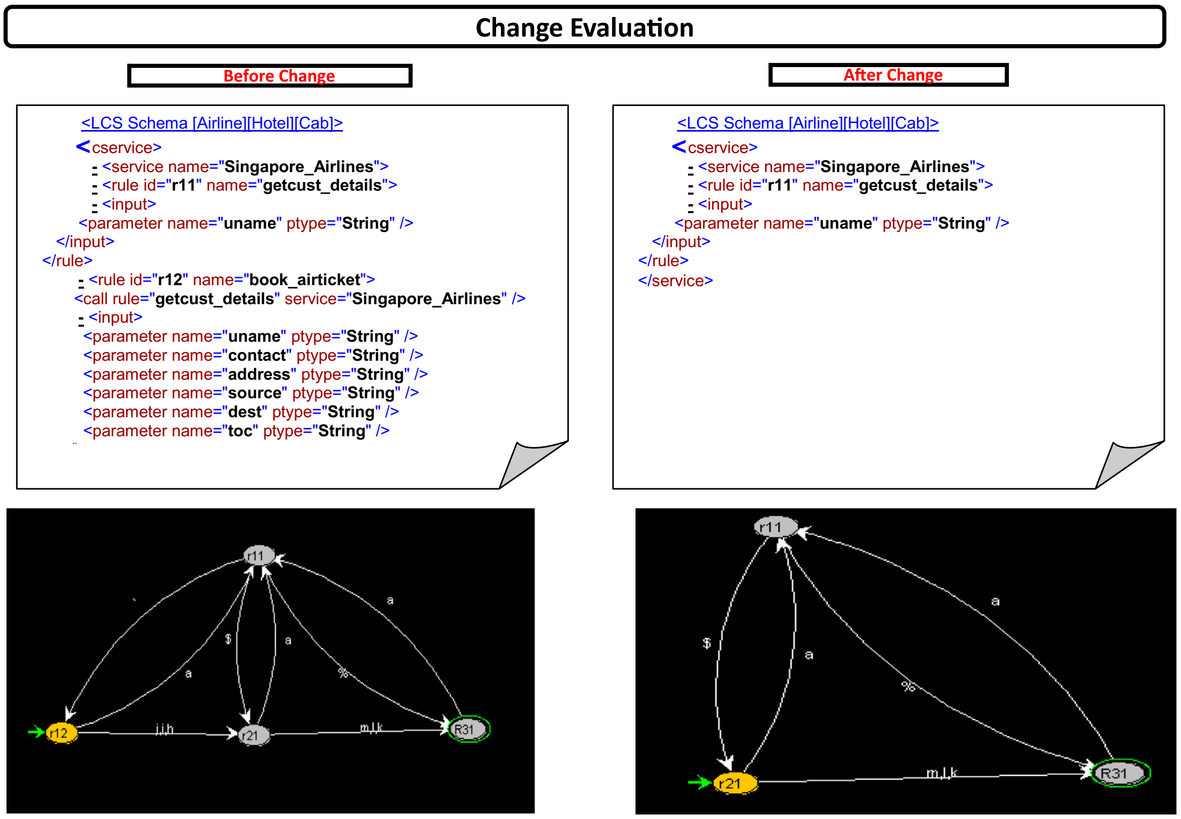


Figure 9 Constructed FSM after change for travel LCS.



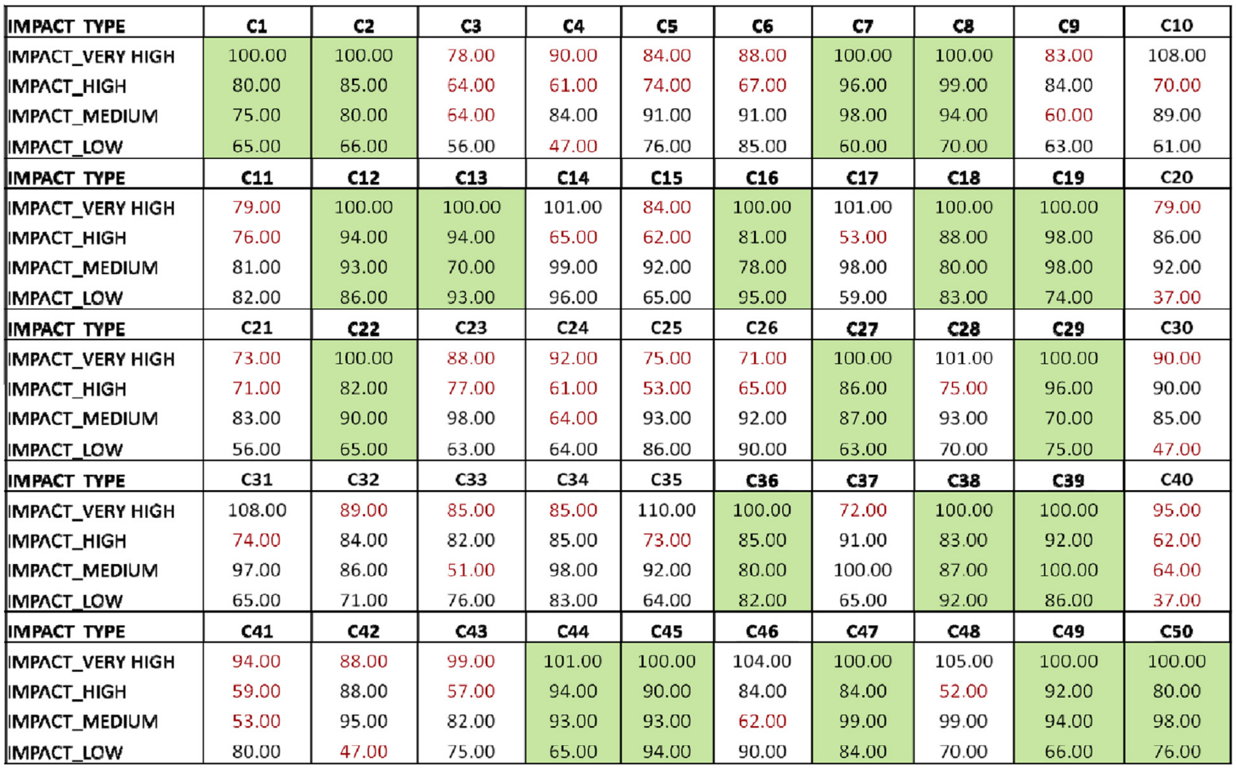
Before Change:

```
<LCS Schema [Airline][Hotel][Cab]>
<cservice>
 - <service name="Singapore_Airlines">
 - <rule id="r11" name="getcust_details">
 - <input>
  <parameter name="uname" ptype="String" />
 </input>
</rule>
 - <rule id="r12" name="book_airticket">
  <call rule="getcust_details" service="Singapore_Airlines" />
 - <input>
  <parameter name="uname" ptype="String" />
  <parameter name="contact" ptype="String" />
  <parameter name="address" ptype="String" />
  <parameter name="source" ptype="String" />
  <parameter name="dest" ptype="String" />
  <parameter name="toc" ptype="String" />
```

After Change:

```
<LCS Schema [Airline][Hotel][Cab]>
<cservice>
 - <service name="Singapore_Airlines">
 - <rule id="r11" name="getcust_details">
 - <input>
  <parameter name="uname" ptype="String" />
 </input>
</rule>
</service>
```

Figure 10 Change evaluation for travel LCS.



IMPACT TYPE	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
IMPACT_VERY HIGH	100.00	100.00	78.00	90.00	84.00	88.00	100.00	100.00	83.00	108.00
IMPACT_HIGH	80.00	85.00	64.00	61.00	74.00	67.00	96.00	99.00	84.00	70.00
IMPACT_MEDIUM	75.00	80.00	64.00	84.00	91.00	91.00	98.00	94.00	60.00	89.00
IMPACT_LOW	65.00	66.00	56.00	47.00	76.00	85.00	60.00	70.00	63.00	61.00
IMPACT TYPE	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
IMPACT_VERY HIGH	79.00	100.00	100.00	101.00	84.00	100.00	101.00	100.00	100.00	79.00
IMPACT_HIGH	76.00	94.00	94.00	65.00	62.00	81.00	53.00	88.00	98.00	86.00
IMPACT_MEDIUM	81.00	93.00	70.00	99.00	92.00	78.00	98.00	80.00	98.00	92.00
IMPACT_LOW	82.00	86.00	93.00	96.00	65.00	95.00	59.00	83.00	74.00	37.00
IMPACT TYPE	C21	C22	C23	C24	C25	C26	C27	C28	C29	C30
IMPACT_VERY HIGH	73.00	100.00	88.00	92.00	75.00	71.00	100.00	101.00	100.00	90.00
IMPACT_HIGH	71.00	82.00	77.00	61.00	53.00	65.00	86.00	75.00	96.00	90.00
IMPACT_MEDIUM	83.00	90.00	98.00	64.00	93.00	92.00	87.00	93.00	70.00	85.00
IMPACT_LOW	56.00	65.00	63.00	64.00	86.00	90.00	63.00	70.00	75.00	47.00
IMPACT TYPE	C31	C32	C33	C34	C35	C36	C37	C38	C39	C40
IMPACT_VERY HIGH	108.00	89.00	85.00	85.00	110.00	100.00	72.00	100.00	100.00	95.00
IMPACT_HIGH	74.00	84.00	82.00	85.00	73.00	85.00	91.00	83.00	92.00	62.00
IMPACT_MEDIUM	97.00	86.00	51.00	98.00	92.00	80.00	100.00	87.00	100.00	64.00
IMPACT_LOW	65.00	71.00	76.00	83.00	64.00	82.00	65.00	92.00	86.00	37.00
IMPACT TYPE	C41	C42	C43	C44	C45	C46	C47	C48	C49	C50
IMPACT_VERY HIGH	94.00	88.00	99.00	101.00	100.00	104.00	100.00	105.00	100.00	100.00
IMPACT_HIGH	59.00	88.00	57.00	94.00	90.00	84.00	84.00	52.00	92.00	80.00
IMPACT_MEDIUM	53.00	95.00	82.00	93.00	93.00	62.00	99.00	99.00	94.00	98.00
IMPACT_LOW	80.00	47.00	75.00	65.00	94.00	90.00	84.00	70.00	66.00	76.00

Figure 11 Impact analysis results for 50 change requests and impact succeeded change requests.

estimation before change and red curve indicates the estimation after change.

4.2. Goal formulation

The various goals of the framework have been identified and depicted in Table 4. The proposed framework aims at providing efficient change management with reduced risk, high accuracy, high degree of automation and incident matching. These goals of the framework are evaluated based on the metrics shown in the table. For the estimation of risk, the number of automatons in which either manageability i.e. property evaluation or decidability i.e. change evaluation comprising of change factor, constraint factor and QoS factor evaluation has failed is to be extracted from the entire set of incidents. This indicates the extent to which runtime support and decision making assured by the framework have failed to meet the business needs.

The degree of automation is the extent to which the decisions are automatically made by the framework. This is indicated by the number automatons which have proceeded to and succeeded in the decidability calibration after manageability calibration from the entire set of incidents i.e. the extent to which the decisions have been automatically made. Incident matching indicates the possible number of combinations of rule and service patterns out of the existing number of patterns from the entire set of incidents i.e. the depth and level of incident matching performed by the framework. Accuracy is the measure which indicates the extent to which the decisions made by the framework have been correctly and cautiously made i.e. how accurate is the behavior of the framework. This is estimated from the number of manageability and decidability automatons which have succeeded in incorporating the change.

5. Related work

In this section the various works pertinent to the proposed framework and approaches are discussed. Change management is a structured approach for making sure that the changes are thoroughly and smoothly made and that the lasting benefits of the change are achieved.

5.1. Related change management work

Liu et al. (2011) have presented a framework where managing changes in LCSs has been modeled as a dual service query optimization process (Liu et al., 2011). The optimization has been performed by considering only the non functional factors like reputation and the QoS factors. However, we have considered the factors for enabling the analysts to manage the functional and non functional requirements of the consumers. Akram et al. (2010) have proposed an automatic change management framework that is based on the petri net models which is used to manage triggering changes and reactive changes (Akram et al., 2010). Also the functional and non-functional properties of the composed services have been identified. However, we have used FSM for the modeling of the changes as they facilitate incident matching with low time and space complexity. We also have identified the QoS

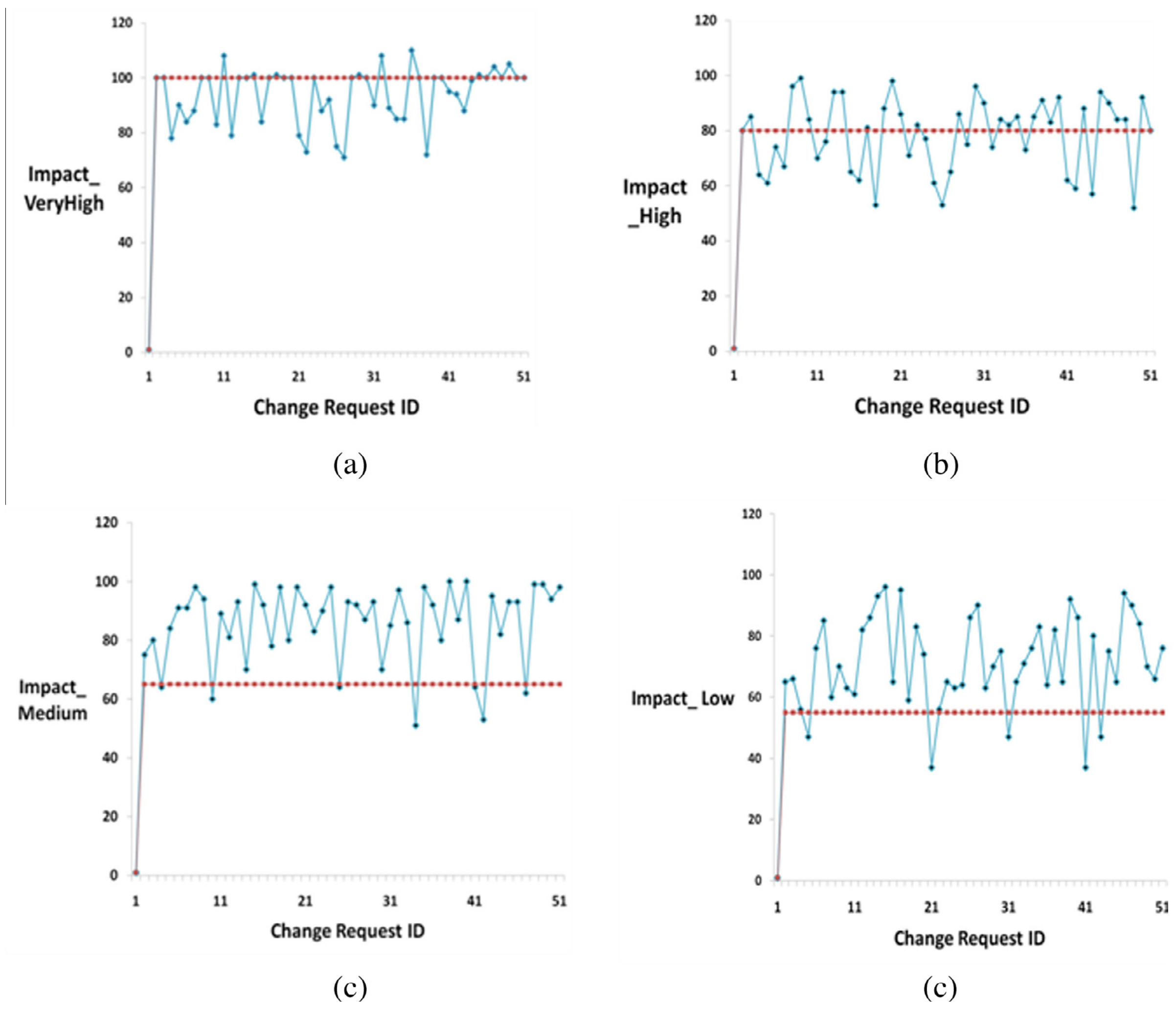


Figure 12 Impact analysis results (a) for Impact_VeryHigh, (b) for Impact_High, (c) for Impact_Medium and (d) for Impact_Low.

parameters into account as it has to satisfy the service providers and analysts since as business policies change continuously, there is a need for service-oriented systems to be submissive and adaptive. Dijkman et al. (2011) have proposed the metrics for the evaluation of the node matching, structural and behavioral similarities of business (Rajeswar et al., 2014). The evaluation of similarity among business process models is only dealt whereas its applicability in web service change management is not considered. Tjoa et al. (2011) have proffered a formal approach which enables risk-aware business process modeling and simulation (Koriem et al., 2003). But the estimation of risk is based on the detection of intrusion and the estimation of policy violations alone. In our work, we have estimated the metric risk from the evaluation of change, constraint and QoS factor evaluation.

5.2. Methodology adopted in the related work

Liu et al. (2013) have proposed a change management framework where managing changes in LCS which deals with both functional and non functional change requests (Liu et al., 2013). But there is no methodology adopted for performing emergency changes and change impact analysis was not done. In our work, by implementing Finite State Machine, a standard methodology is adopted for the evaluation of manageability and decidability factors. Craiu and Lee (2006) have developed likelihood-based methods for estimating rules of cellular automata aimed at the regeneration of observed regular patterns (Craiu and Lee, 2006). But the idea of pattern generation using likelihood inference has not been focused in the view of change management. In our work, we have implemented Probabilistic Cellular Automata (PCA) for the inference from historical patterns of change incidents. The concept of Probabilistic Arithmetic Automata (PAA) described as markov chains over a larger state space and their benefits in their utility as a modeling technique has been contemplated by Marschall et al. (2012) and Dijkman et al. (2011). The claim of PAA as a modeling technique and its justification has paved the way for the idea of implementing the Probabilistic Cellular Automata (PCA) in our work for the decision making and prediction from the historical change incidents. Maamar et al. (2008) have suggested a holistic approach for managing software change impact (Yu et al., 2008). But, this

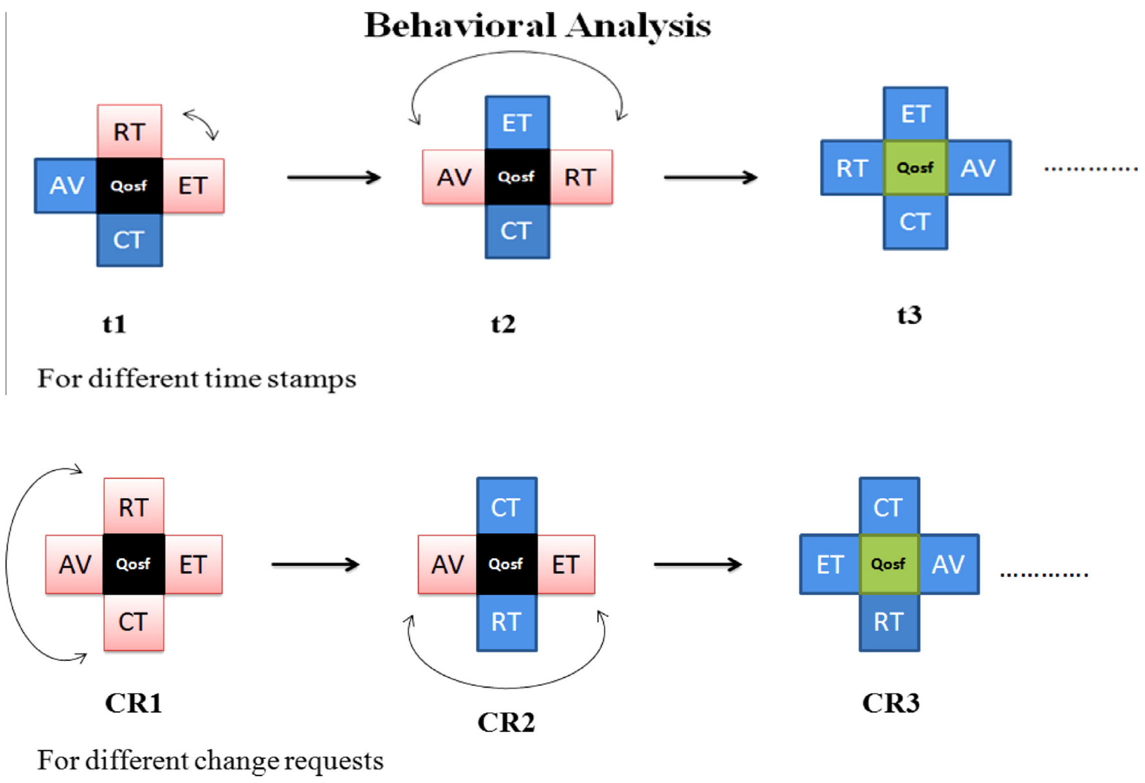


Figure 13 Cell position change based on probability function and behavioral analysis using PCA.

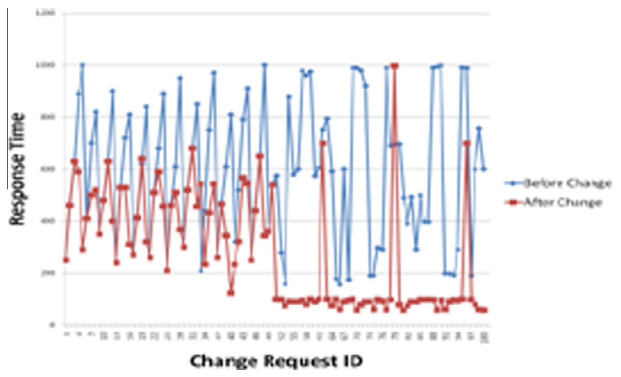


Figure 14 Change in the estimation of response time for 100 change requests of travel LCS.

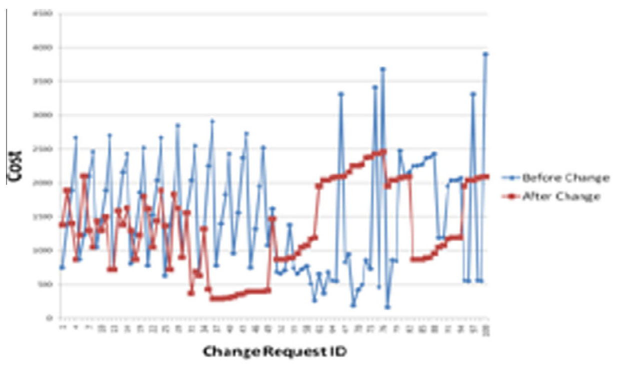


Figure 16 Change in the estimation of cost for 100 change requests of travel LCS.

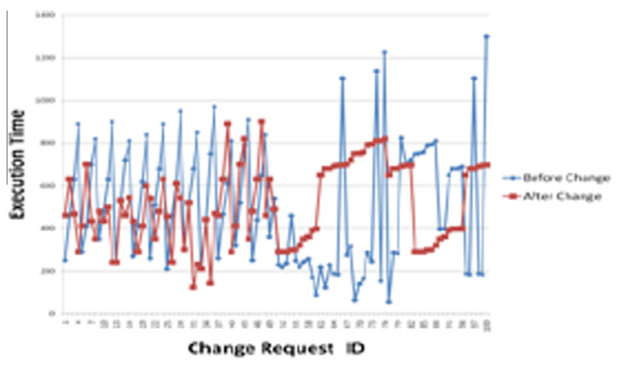


Figure 15 Change in the estimation of execution time for 100 change requests of travel LCS.

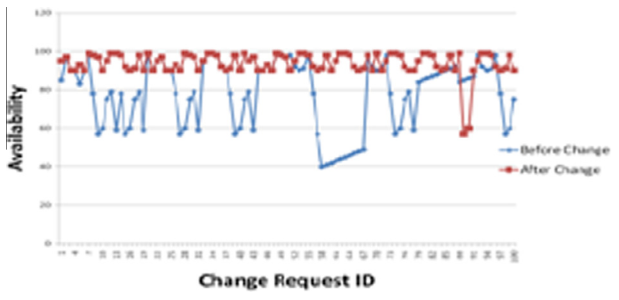


Figure 17 Change in the estimation of availability for 100 change requests of travel LCS.

Table 4 Evaluation metrics for CIA framework.

Goals	Formulation
Risk	$\text{Risk(RI)} = \sum \left[\left[\frac{m'}{m}\right] \cup \left[\frac{d'}{d}\right] \right] \times 100$ m' – Number of automata in which manageability failed m – Total number of automata generated in manageability pattern d' – Number of automata in which decidability failed d – Total number of automata generated in decidability pattern
Degree of automation	$\text{Degree of Automation} = \sum \left[\frac{\left[1-\frac{m'}{m}\right] \cap \left[1-\frac{d'}{d}\right]}{CR_n} \right] \times 100$ m' – Number of automaton in which manageability failed m – Total number of automatons generated in manageability pattern d' – Number of automaton in which decidability failed d – Total number of automaton generated in decidability pattern CR_n – Number of change requests
Incident matching	$\text{Patterns} = \sum [(\text{PR}_\text{P}\text{C}_{\text{PR}_C} + \text{PS}_P\text{C}_{\text{PR}_C})]$ PR_C – Possible rule combinations for received rule PF_C – Possible service combinations for received composition PR_P – Existing rule combinations for received rule PF_P – Existing service combinations for received composition
Accuracy	$\text{Accuracy} = \sum \left[\left[1 - \frac{m'}{m}\right] \cap \left[1 - \frac{d'}{d}\right] \right] \times 100$ m' – Number of automaton in which manageability failed m – Total number of automatons generated in manageability pattern d' – Number of automaton in which decidability failed d – Total number of automaton generated in decidability pattern

work illustrates how context and policy are woven into web services composition scenarios only. In our work, we have incorporated the estimation of Business Policy Enforcement (change factor) for hunting for any policy violations among services and accessibility (property) for checking if the services in composition are accessible. All these issues are the hurdles in the current scenario which hinder the change management process in long term composed services in enabling the analysts to make the changes in a timely and cost effective manner and with reduced risk.

5.3. Related works in web services

The analysis done (Rajeswar et al., 2014) brings out the importance of considering QoS for service composition. This research work has analyzed a number of QoS requirements, various QoS models and also the difficulties that are faced for QoS based service composition. The research work has provided a utility function that can be used to evaluate the QoS parameters of various services, but this work does not consider the various trade-offs that will be faced by QoS based composition during runtime. These QoS factors have not been focused in the change management view of web services. Thus in our work, QoS factors are considered during the change management phase, as they help in verifying the impact caused due to the change over the quality of the business process. The cookie (Venkatesan et al., 2013) based accounting model proposed identifies duplicate service requests which are used by most of the attackers to affect the availability of the servers. This model works by recording each request in the client side and a hash value of the corresponding request is also stored in the server. Thus any client attempting to send a duplicate request or modify the content of the cookie can be easily identified, thereby protecting the server from DOS attacks.

5.4. Related work based on petri nets

Koriem et al. (2003) has proposed a system called SRN Equivalent Model (EM) for evaluating the performance of POS based system. This model is a combination of petri net theory and control theory and has two main nets. The main net consists of the known parts and the other net (second) consists of parts that feed the known parts. Considering petri nets for change management has certain difficulties as their complex structures brings difficulties in the analyzing them. Moreover, representation of petri nets is task based which is not suitable for analysts. Therefore, in our work FSM representation of the services in terms of rules, policy etc. make it easier for the analyst to achieve successful changes. FSM also achieves the decidability of the problems that were not possible when petri nets were used. The verification and validation of the changes will also become a complex task as the number of services increases.

6. Discussion

In this section, we present the contributions of our research and its limitations.

6.1. Research contributions

This research makes the following contributions by presenting a formal approach for change impact analysis of LCS using Probabilistic Cellular Automata. Specifically this research:

- Presents a formal approach for analyzing the LCS for effective runtime management by property evaluation and to evaluate the changes over the LCS by change evaluation using Finite State Machine as the methodology.
- Presents a novel framework with standard procedures and methodologies for managing the changes in a timely and cost effective manner finding the inconsistencies in the changes made.
- Presents and performs change impact analysis and behavior analysis over the knowledge extracted from the patterns using Probabilistic Cellular Automata (PCA) with efficient incident matching enabling to trace the exact reason for failure in case of change being discarded.

In the existing works, petri nets are used as mathematical models which help in depicting the flow of a particular process (Tan et al., 2009). A petri net structure has 4 components – state, transition, token and edge. Petri nets are complex to be analyzed and moreover the number of states increases to a very large extent when the number of services in composition is large. Petri nets are suitable for concurrent and complex processes like business process alignment, workflow management and the focus is more on the top down changes alone (Tan et al., 2009) whereas Finite State Machines are suitable for sequential process with every state followed by a transition to another state or end state which is the focus of our work (Chenthati et al., 2010). The petri nets are represented as tasks [Event, Action. Condition] and are suitable for event based change verification which is suitable only for the IT developers and not for the analysts, whereas FSM which is represented as rule, function, and policy assures a representation supporting the analysts which is much needed in the change scenario. Decidability of the problems is not possible to be judged using petri nets whereas it is possible using FSM with standard algorithmic conversion and automated decision making. The default language handling capability of the FSM notifies every new member added to the language, so the analyst can be notified when any new rule, function or even a parameter is added to the logic which is not possible in petri nets. FSM is the best choice in case of incident mapping because of its backtracking capability with cellular automata compared to petri nets. Rule level changes in the reflected directly in the source code by the FSM.

With respect to the behavioral analysis, petri nets can be used to model the behavior of concurrent process whereas FSM can be used to model the I/O behavior which is much needed in our case. The primary goal of the framework is to respond to the changes correctly and cautiously at low cost and time. Since petri nets are complex structures they demand large memory conception and high processing time leading to high time and space complexity. Petri nets are comparatively slow and inefficient in the retrieval of services and inflexible in searching and tracking of them. There are high changes for a petri net based system to reach an undesirable state due to the loss of its token. Hence, petri nets are suitable only for modeling concurrent processing associated with deterministic task, event and condition and not for modeling non deterministic sequential processing associated with language theory like FSM.

6.2. Limitations

Generalization of the results from this work must be done with caution in light of its limitations. Though petri nets have a lot if issues with respect to our focus as mentioned, they also have a few advantages compared to FSM. The representation as tasks enables high expressiveness in petri nets while the expressiveness is low in the case of FSM. Petri nets are much suited for concurrent processes and in the case of the change scenario demanding the analysis of concurrent processes, petri nets are more advantageous (Akram et al., 2010; Tan et al., 2009). Though this problem can be solved when FSM are used by the use of Tuning Machines, petri nets are readily suitable for concurrent deterministic processing while FSM are not suitable for concurrent processing. Petri nets can easily represent the safe and unsafe states of the Service Oriented Enterprises (Akram et al., 2010). When a new change request arises for a new LCS set which has never been encountered before, the change impact analysis and incident matching are not done. Though the property, change, constraint and QoS factor evaluation are evaluated, the inference cannot be made based on the probabilistic measures from the previous incidents. However the results of the new change are recorded as a new incident and this aids in impact analysis for future similar change requests.

Change management provides large space for intense research. Future enhancements of this work would be to implement tuning machines in order to support analysis of concurrent processes extending the framework to support change optimization and to support automated identification of change decay over time.

7. Conclusion

Based on the behavioral analysis, the degree of automation, accuracy, incident matching and risk are to be predicted based on incident matching. Hence the proposed framework is expected to assess, change and manage the business logic of the web services in long term composition with efficient incident matching, high degree of automation and authorization, high degree of knowledge transfer, manageability, decidability, high success and incident rates at low risk and cost without compromising the quality of the business process. The framework facilitates in choosing the 'best' services from multiple service providers who compete to offer the similar services with a different user-centric quality and paves way for the consumers or end users to be highly benefited from the open competition among businesses. This aids in the enhancement of the change management process in long term composed services so that the changes can be made by the analysts in a timely and cost effective manner and with reduced risk.

References

Akram, Salman, Bouguettaya, Athman, Liu, Xumin, Haller, Armin, Rosenberg, Florian, Xiaobing, Wu, 2010. Change management framework for service oriented enterprises. Int. J. Next-Generation Computing (IJNGC) 1 (1), 1–077.

Apostolou, Dimitris, Mentzas, Gregoris, Stojanovic, Ljiljana, Thoenssen, Barbara, Lobo, Tomás Pariente, 2010. A collaborative decision framework for managing changes in e-Government services. Elsevier J. Gov. Inf. Q. 28 (1), 101–116.

Chenthati, Deepak, Vaddi, Supriya, Mohanty, Hrushikesha, Damodaram, Avula, 2010. Verification of web services modeled as finite state machines. In: Proceedings of IEEE International Conference on Mathematical/Analytical Modeling and Computer Simulation (AMS), pp. 526–531.

Chua, David K.H., Aslam Hossain, Md., 2012. Predicting change propagation and impact on design schedule due to external changes. IEEE Trans. Eng. Manage. 59 (3), 483–493.

Craiu, Radu V., Lee, Thomas C.M., 2006. Pattern generation using likelihood inference for cellular automata. IEEE Trans. Image Process. 15 (7), 1718–1727.

Dijkman, Remco, Dumas, Marlon, van Dongen, Boudewijn, Kaarik, Reina, Mendling, Jan, 2011. Similarity of business process models: metrics and evaluation. Elsevier J. Inf. Syst. 36 (2), 498–516.

Koriem, Samir M., Dabbous, T.E., El-Kilani, W.S., 2003. A new petri net modeling technique for the performance analysis of discrete event dynamic systems. J. King Saud Univ. Comput. Inf. Sci. 15, 95–128.

Liu, Xumin, Bouguettaya, Athman, Qi, Yu, Malik, Zaki, 2011. Efficient change management in long-term composed services. Springer J. Serv. Oriented Comput. Appl. 5 (2), 87–103.

Liu, Xumin, Akram, Salman, Bouguettaya, Athman, 2011. Change Management for Semantic Web Services, first ed. Springer Science and Business Media, ISBN 978-1-4419-9328-1.

Liu, Xumin, Bouguettaya, Athman, Wu, Jemma, Zhou, Li, 2013. Ev-LCS: a system for the evolution of long-term composed services. IEEE Trans. Serv. Comput. 6 (1), 102–115.

Maamar, Zakaria, Benslimane, Djamal, Mostéfaoui, Ghita Kouadri, Subramanian, Sattanathan, Mahmoud, Qusay H., 2008. Toward behavioral web services using policies. IEEE Trans. Syst. Man Cybern. 38 (6), 1312–1324.

Marschall, Tobias, Herms, Inke, Kaltenbach, Hans-Michael, Rahmann, Sven, 2012. Probabilistic arithmetic automata and their applications. IEEE/ACM Trans. Comput. Biol. Bioinf. 9 (6), 1737–1750.

Plebani, Pierluigi, Pernici, Barbara, 2009. URBE: web service retrieval based on similarity evaluation. IEEE Trans. Knowl. Data Eng. 21 (11), 1629–1642.

Rajeswar, M., Sambasivam, G., Balaji, N., Saleem Basha, M.S., Vengattaraman, T., Dhavachelvan, P., 2014. Appraisal and analysis on various web service composition approaches based on QoS factors. J. King Saud Univ. Comput. Inf. Sci. 26 (1), 143–152.

Rovegard, Per, Angelis, Lefteris, Wohlin, Claes, 2008. An empirical study on views of importance change impact analysis issues. IEEE Trans. Software Eng. 34 (4), 516–530.

Setzer, Thomas, Bhattacharya, Kamal, Ludwig, Heiko, 2010. Change scheduling based on business impact analysis of change-related risk. IEEE Trans. Network Serv. Manage. 7 (1), 58–71.

Tan, Wei, Fan, Yushun, Zhou, MengChu, 2009. A petri net-based method for compatibility analysis and composition of web services in business process execution language. IEEE Trans. Autom. Sci. Eng. 6 (1), 94–106.

Tjoa, Simon, Jakoubi, Stefan, Goluch, Gernot, Kitzler, Gerhard, Goluch, Sigrun, Quirchmayr, Gerald, 2011. A formal approach enabling risk-aware business process modeling and simulation. IEEE Trans. Serv. Comput. 4 (2), 153–166.

Venkatesan, S., Saleem Basha, M.S., Chellappan, C., Vaish, Anurika, Dhavachelvan, P., 2013. Analysis of accounting models for the detection of duplicate requests in web services. J. King Saud Univ. Comput. Inf. Sci. 25 (1), 7–24.

Wang, Qianxiang, Shao, Jin, Deng, Fang, Liu, Yonggang, Li, Min, Han, Jun, Mei, Hong, 2009. An online monitoring approach for web service requirements. IEEE Trans. Serv. Comput. 2 (4), 338–351.

Xiong, Peng Cheng, Fan, Yu Shun, Zhou, Meng Chu, 2009. Web service configuration under multiple quality-of-service attributes. IEEE Trans. Autom. Sci. Eng. 6 (2), 311–321.

Yu, Qi, Liu, Xumin, Bouguettaya, Athman, 2008. Deploying and managing web services: issues, solutions and directions. Springer VLDB J. 17 (3), 537–572.