Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 6 (2012) 163 – 170

Procedia Technology

2nd International Conference on Communication, Computing & Security [ICCCS-2012]

Evaluation of Change Factors for Web Service Change Management

Thirumaran.M[a] , Dhavachelvan.P[b], Aishwarya.D[c], Kiran Kumar Reddy[d]

[a]*Department of Computer science and Engineering, Pondicherry Engg College, India*
. [b]*Department of Computer science and Engineering, Pondicherry University, India.*
[c]*Department of Computer science and Engineering, Pondicherry Engg College, India.*
[d]*Department of Computer science and Engineering, Pondicherry Engg College, India.*

Abstract

Service oriented architecture (SOA) is a smart designing principle which has been evolved for integrating business tasks. Business activities that have to be designed based on SOA are implemented via web services. Using web services (WS) one can exchange data between different applications and different platforms. Service providers register their services in the service registry and consumer obtains the required services from the same. The main concern in this routine which directly sways business growth rate is change management. Change management is an emerging issue in web service computing where clients might want to change the obtained services at some period of time. But in order to do it they should be requesting the provider programmers each and every time and separate payment has to be done for that task. In order to reduce this complexity we propose a new model for implementing change requests by business analysts themselves. Here we propose a new dynamic schema driven business logic model using Finite State Machine (FSM) to accomplish WS change management in a best manner so that business growth rate can be increased. This model is distinctively done for business analysts to perform changes in the services on their own instead of depending on the programmers. Furthermore a predictive model is contrived using cellular automata for supporting business analysts. The predictive model includes the change factors like order of execution; similarity measure, schema validation, and mapping function and time/space complexity which appears when a particular change request is executed.

Keywords: Web Services; Change Management; Change factors; Finite State Machine; Cellular Automata.

1. Introduction

SOA can be viewed in terms of various levels of abstractions, ranging from those that concern within an application to those that concern service applications interacting across enterprises. In this architecture,

doi:10.1016/j.protcy.2012.10.020

change management plays a big issue. It completely affects the business growth due to the vast competition that is present between enterprises based on customer satisfaction. According to the changes in the real world or changes in the need of the customers, the web services provided also should be changed and provided. These changes can be made slowly or immediately. The change that has to be made immediately affects the business growth tremendously because the changes will cause sudden change in the usage of the service. According to the change made, the customers might start using it more or less and accordingly our profit will get increased or lowered. We can consider ticket reservation as example, sometimes the ticket fare can be lowered or increased and according to the trend the end customers might start using our clients' service. But every time when the fare amount has to be changed, the business analyst cannot contact the provider programmer which will introduce a delay in the change management process. He might feel more comfortable when he himself is provided with an option to change the services provided to him. So we introduce a business logic model that allows the business analyst himself to make the required changes according to his own customers' needs. Here we use business logic model instead of business process model because here the process is not implemented from the scratch, only small changes has to be made to the existing source codes. The changes can be made at three levels – rule, function and parameter. The changes can also be extended to dependency between two rules or functions or parameters. Since the business analyst himself is allowed to make the changes the language which is used here to represent the code involved for a web service is XML. The machine process able schema gives a clear hierarchical business workflow. The statements will be available which can be directly understood and the changes can be accomplished. This paper further says about the literature survey made in section 2, describes the new proposed idea with algorithms for change measures in section 3 and case study made on banking domain in section 4 with conclusion and references in section 5 and 6 respectively.

2. Literature survey

Yang et al presented a method for detecting failures in implementation of services that are running currently in concurrent order and also from rules in Service Oriented architecture based atmospheres. It also finds the dependencies that co exist among the services and are extracted from log files when any change occurs [1]. However this approach has various shortcomings such as no graphical views are generated for the dependencies existing among the various service logics and fails for correctness of the solution. We use tools to generate a graph for the existing dependencies when a rule in the service logic has to be modified. Business dealings are obligatory to accomplishing the company of changes within legislation and strategy. However the successful executions of innovative legislation are a lot expensive, might include long escort period which cause huge loss to industry trade. Yiwei Gong and others proposed a novel technique for business processes in favor of building litheness and agility while applying changes through innovative or improvised policies. It also elaborated a new research method for software reuse by joint policy execution [2]. But this technique fails to bring agility and flexibility in business logic as compared to business process. Dimitris Apostolouet all presented an ontology driven method for managing changes in e-Government services based on SOA. It enables methodical reaction of these systems to changes by applying proper techniques intended for accomplishing stability whenever a change is exposed [3]. However this method deals at business process level and not at business logic layer. In [4], the authors presented a novel method to represent changes in ontology which have been organized in layers and provided data independence from versions, etc. Also deposits of change circulation stratagems are allowed to keep distributed copy of the same ontology synchronized. This method does not clearly indicate how the changes in ontology are mapped at business layers. A methodology for handling changes in Long-term Composed Services (LCS) was proposed by Xumin Liu, Athman Bouguettaya and others. In this technique the authors have categorized changes in two ways:- Top-down and Bottom-Up scaffolds. Also the functional and non-functional properties of the composed services have been identified [5], [6],[13] and [14]. When compared with the work described in this paper, business logic is taken as importance as business analysts prefer implementing the business logic directly and not interested in hierarchy of business process. We also have identified the QoS parameters into account as it has to satisfy both

the service providers and analysts as well as clients. As business policies change continuously, there is a need for service-oriented systems to be submissive and adaptive. Hans Weigand et al presented an elucidation to policy changes in business atmospheres by dividing the SOA based architectures on policy enforcements and their specifications [7]. Their method has been highly successful to Adaptive Service Oriented Architectures (ASOA). However mapping of business policies with respect to service dependencies have high impact and this paper lacks at achieving consistency and correctness of the policies implemented across various services. H. Yahyaoui and others formulated a method to manage and handle the changes propagated at run time through Web Service Policy Language (WSPL) which permits perspective pattern by mutually policy and rule stages [8]. However our work concentrates at all the layers of the service to be redeployed. Pedro Antunes and Hernâni Mourãothrow in Business Process Management systems (BPM) through resilience sustain. The system introduced a deposit of services integrating resilience support in BPM. Harry Jiannan Wang and J. Leon Zhao modeled a new approach for changes in work flow management systems called Constraint-centric Workflow Change Analytics (CWCA). The core functionality of this system is it analyzes change anomalies in workflow based architectures. An algorithm (procedure) is used to discover changes in SOA context and open-source rule engine is exposed to afford a verification of notion execution of CWCA [9].Shari S.C. Shang, Eldon Y. Li,Ya-Ling Wu, Oliver C.L. Hou have done a survey on existing life cycle models in context of web services. These representations illustrate the variety of open Web 2.0 relevance and presented a scaffold in support of a superior kind of working patterns and significance scheme in the Web 2.0 model [10]. This paper has identified a new business model which primarily concentrates at web service business logic. Minhong Wang and Huaiqing Wang presented a model for managing business logic of the services within the business processes. Since many of the existing models come up with changes at process level, this paper has identified the changes propagated at business logic layer [11]. This approach has some similarity with our work but lacked proof and confidentiality as the changes implemented at business logic layer are completely measurable and finite. Daniel Żmuda, MarekPsiuk and Krzysztof Zieliński presented some innovative challenges in SOA based environments to observe and watch the implementation strategies occurring dynamically. Dynamic run-time reconfiguration is the prime concept of that approach [12]. But this approach needs technical IT staff to reconfigure the entire business process each and every time when changes are done to the process as our work eliminates this burden.

3 Proposed Change Management Model

The proposed system workflow involves the following process. Initially when the change request is made, it is sent to the change request manager who sees in which part of the coding the change has to be made. The corresponding source code is tracked and a FSM is constructed to find the flow. Here we design only business logic process model instead of business process model. Usually the business process models are designed with Petri net- a mathematical model which helps in depicting the flow of a particular process. A Petri net structure has 4 components – state, transition, token and edge, Petri nets are difficult to be analyzed and moreover the number of states increases to a very large extent when the number of services and sub services itself become large. Some of the major difficulties one may face due to the usage of Petri Nets are- it has slow and inefficient retrieval of services, large memory conception, and large processing time, system which is inflexible in searching and tracking of services, system which might lose its token and end in undesirable state.

Moreover we are going to make only small changes to the already existing code; hence the Petri net structure can be replaced by finite state machine – which is a simple mathematical model with only 2 components – nodes and transitions. This usage of FSM model eliminates the difficulty in using Petri nets and further gives a full description of services with its subservices details, the flow of control between them. By replacing Finite State Machine with Petri nets we can obtain - Fast, efficient retrieval of services, a system with less memory conception, a system with less processing time, a system, flexible in searching and tracking of services.

Once the FSM is constructed, the business logic set is extracted and property evaluation table is designed, from where the rule or the function or the parameter in which the change has to be made is found. Then the corresponding XML schema is extracted and dependency analysis is made. Based on the dependency analysis

the changes are made if they don‘t have any risk affecting the business growth. Then the new logic schema in inserted again the web service is updated and executed according to the new change. Additionally a support is provided to the business analysts by calculating change measures – the factors through which he can judge whether the change is incorporated rightly and it has less risk factor. This predictive model is implemented via cellular automata and graphs are developed based on the changes made. The architecture depicting the above process is shown below.

3.1 Change request

The change request should be given with 3 parameters – Command, Resource, and Condition where command says what operation should be done like whether the source has to be updated, modified, deleted or inserted. The resource says on which file the change has to be modified and with the help of the resource file it extracts the source code. The condition parameter gives the constraints based on which the change has to be made.

3.2 Source code

The resource file in which the change has to be made is found from the resource parameter in the change request and the particular part of source code is extracted but the extracted form will not clearly how it will affect other parameters or rules or functions if a change is made in this particular part of the code.

3.3 FSM construction

Hence we go for a finite state machine construction to view the perfect flow between parameters, functions and rules. It identifies all set of codes. It ensures the completeness of logic for which the change has to be made. Two types of FSM will be created – FSM for rules and FSM for functions. In FSM for rules, the rules will become the nodes and the functions will become the transitions. In FSM for functions, the functions will become the nodes and the parameters will become the transitions.

3.4 Business logic set

Then in the next step business logic schema is constructed in which each rule, function and parameters involved in the code which has to be changed is converted to xml language so that the business analyst will be able to easily understand in which part to make the change and how to make the change. The business logic designed consists of four parts – rule schema, function schema, parameter schema and dependency set.

The overall business logic set consists of four columns with rule set, function set, parameter set and dependency set. The rule set consists of all the rules involved. The rules are represented as R1, R2, and R3 etc. The function set consists of all the functions involved like F1, F2, and F3 etc. The parameter set consists of the overall parameters involved as P1,P2,P3 etc. finally the dependency set consists of the dependency existing between these rules, functions and parameters and they are represented as D1,D2,D3 etc.

3.5 Existing business logic schema

Finally after the dependency is analyzed, the schema where the change can be made is developed in front of the business analyst, where he is allowed to make any changes according to the request in run time and the changes are immediately updated as soon as made. The updated schema is also developed in parallel along with property evaluation table with which the business analyst can confirm or deny the changes made.

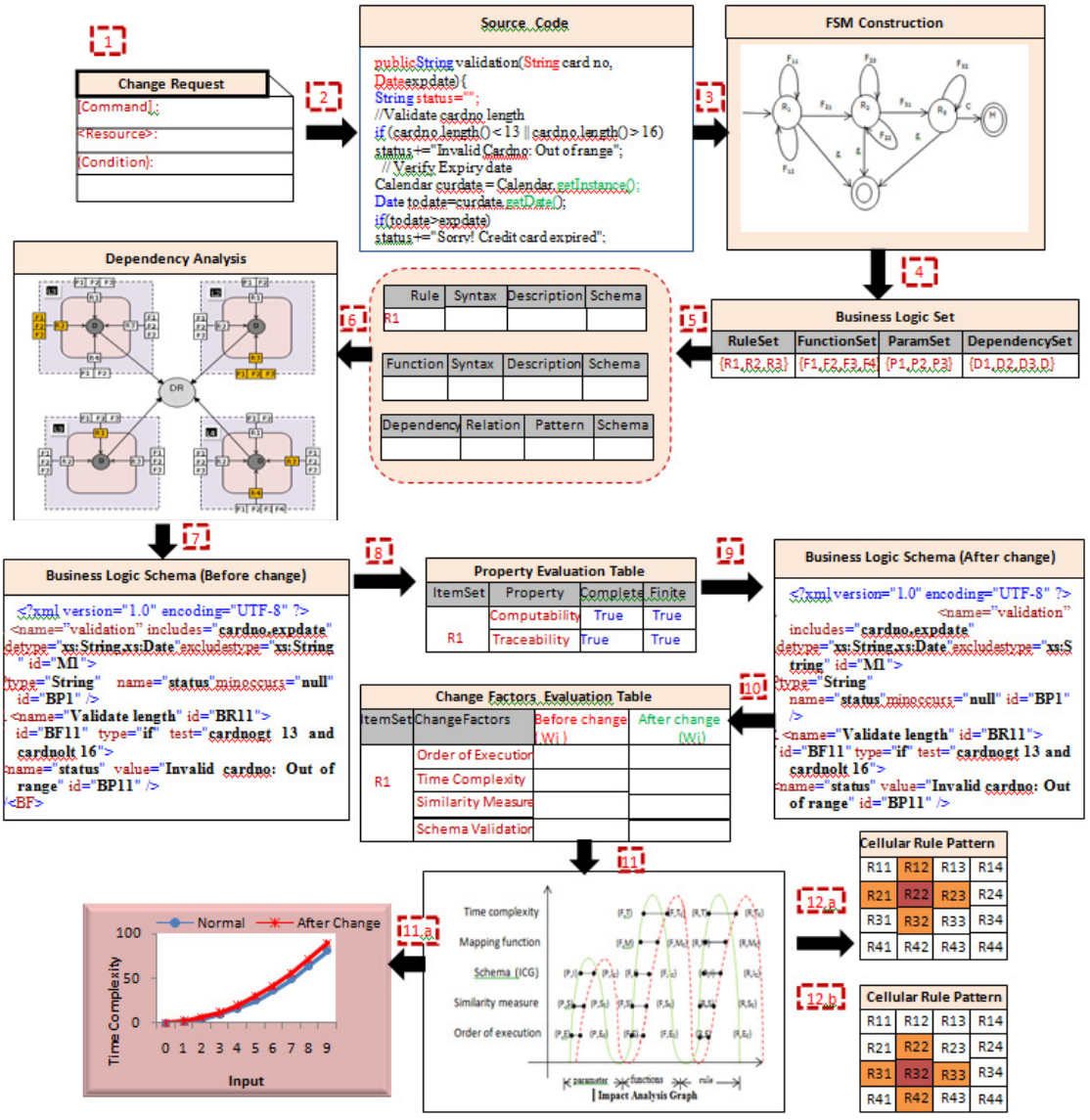

Fig 1 Change Management Workflow Diagram

3.6 Property evaluation table

The property evaluation table finds computability and traceability. Computability gives a whether a particular change can be made without any risks and traceability says where all a change might affect in future code. While trying to change it notifies in rum time by highlighting links.

3.7 Dependency analysis

In dependency analysis, rules similarity and rule dependency are found. In general parameters are involved inside functions and functions are involved within rules. There will be same parameters involved between 2 functions and there will be functions between 2 rules. These similarities and dependencies are analyzed in this

part. For example consider a banking domain service application which involves loan services. It will involve many types of loan services within it like educational loan service, home loan service. These loan services will involve common service like surety, personal information checking etc. So when a small change has to be made in some particular function in some particular type of loan service it will affect the other type of loan services. This dependency is checked here.

3.8 Change factor evaluation

The five change factors which are going to be evaluated are order of execution, schema validation, similarity measure, time complexity and mapping function.

3.8.1 Schema validation

Here the schema before change and after change is checked. The schema segment is checked for completeness and finiteness whether the entire schema involved in change specification is entire. This algorithm takes change specification and BL schema as input. Once if the schema is complete and finite the new changes made to schema will be updated in business logic. If they are not complete then the change specification is rejected.

Algorithm Change Measure (Schema Validation)

Input: Change Specification cs_{sc} with Business Logic Schema
begin
Analyze the change specification cs_{sc} for Completeness and Finiteness
for all cs_{sc} not Null
if (rule | function | parameter) in cs_{sc} is not Complete **then**
Discard request
Current_state:= Previous_ state
else
Map cs_{sc} with the existing Business Logic Set L
end if
if $cs_{sc} L$ **then**
Retrieve the corresponding rules, functions and parameters from L
Generateand compose the schema segment for the associated rules and functions
Modify the schema as per the cs_{sc}corresponding to elements, sub-elements and attributes
else
Add cs_{sc} to the L with modification in the Schema definition
L:=L cs_{sc}
end if
Update the Schemabased on cs_{sc}
Current_schema:= Union(Previous_schema, cs_{sc})
if Parse(Current_schema) = Accept **then**
if Computability(Current_schema,Previous_schema)= True **then**
compute
else
discard changes
restorePrevious_state
end

3.8.2 Order of Execution

Order of execution says whether a particular change made has change the execution order of rules or functions in the code. The algorithm runs by getting the execution order and the change specification(CS) as input. The given CS is checked for completeness and finiteness. Completeness finds whether the logic involved in specification is complete and finiteness finds whether all nodes reach final state in the workflow formed through FSM. If both these dint get satisfied then, the request is discarded else it is mapped with the Business Logic (BL) and corresponding changes are made.

3.8.3 Time complexity

This algorithm finds the change made in time complexity. It takes the change specification as input. If the change specification involves addition of any rules or functions then the new time is added along with the existing time and if the change specification involves deletion of any rule or function then the time is reduced from the existing one.

3.8.4 Similarity Measure

Similarity measure identifies whether any change is made in input output statements or control statements involved in the change specification code. Two factors which are checked here are iteration and condition.

3.8.5 Mapping Function

Mapping function says whether any change made in code affects business‘ policies. If the change that has to confirmed affects any business‘ policy then the change specification is rejected else it is accepted. We have algorithm for mapping function also but it is still under enhancing process for more matching of policies.

3.9 Predictive model

Predictive model is done for increasing the business analyst confidence. It is implemented through cellular automata. A discrete cellular state space L is a set of locally interconnected finite state automata (FSAs) that is typically created by a regular d-dimensional lattice of FSAs. For example, in two-dimensions, a cellular automaton consists of a lattice of L = m × n squares where each square, in the literature called a cell, is represented by one FSA. A local value space defines all possible states for each FSA. The state σ of each FSA can be in one of the finite number of states $\sigma \in \equiv \{0, 1,2, 3, 4, \ldots ,k-1,k\}$ The composition of all possible states of all cells create a state space of the whole cellular automaton. A neighborhood N is created by a set of N topologically neighboring cells, which are influencing a change of the state of each updated cell in the next simulation step. Boundary conditions can be periodic, fixed or reflecting among others. The most important are periodic boundary conditions, which are used to simulate the infinite lattice using a finite one. Periodic boundary conditions are implemented as a torus in two dimensions.

4 Conclusion

The proposed dynamic business logic model and 5 new algorithms for change factors evaluation like order of execution, similarity measure, time complexity, mapping function and schema validation reduces the change management issue in WS change management. The proposed model is implemented in a bank domain case study and the various change factors are evaluated and presented through a graph. We have also developed predictive model using cellular automata based on which the business analyst gain confidence for the changes which he make during emergency times. Next our future work is to predict business growth rate based on the changes made thereby giving hands on support to the business analysts.

5 References

[1]. Yang Xiao, Urban, S., "Using Rules and Data Dependencies for the Recovery of Concurrent Processes in a Service-Oriented Environment", IEEE Transactions on Services Computing, Issue 1, Volume 5, 2012, pp.59-71.
[2]. Yiwei Gong, Marijn Janssen, "From policy implementation to business process management: Principles for creating flexibility and agility", Government Information Quarterly, Volume 29, Supplement 1, Elsevier B.V, January 2012, Pages S61–S71.
[3]. DimitrisApostoloua, GregorisMentzasb, LjiljanaStojanovicc, Barbara Thoenssend, TomásPariente Lobo, ―A collaborative decision framework for managing changes in e-Government services**",** Government Information Quarterly Volume, Elsevier B.V., January 2011, Pages 101–116.
[4]. RaúlPalmaa, Oscar Corchoa, Asunción Gómez-Péreza, Peter Haase, ―A holistic approach to collaborative ontology development based on change management‖, Elsevier journal of Web Semantics: Science, Services and Agents on the World Wide Web, Volume 9, Issue 3, September 2011, Pages 299–314.
[5]. Xumin Liu, AthmanBouguettaya, Jemma Wu, and Li Zhou, "Ev-LCS: A System for the Evolution of Long-term Composed Services", IEEE Transactions on Services Computing, Issue: 99, ISSN : 1939-1374, 23 June 2011. 20
[6]. Xumin Liu, AthmanBouguettaya, Qi Yu, Zaki Malik, "Efficient change management in long-term composed services", ORIGINAL RESEARCH PAPER, Journal of Service Oriented Computing and Applications, Springer-Verlag London Limited 2010, pp.87-103.
[7]. Hans Weigand, Willem-Jan van den Heuvel, Marcel Hiel, "Business policy compliance in service-oriented systems", Elsevier journal of Information Systems, Volume 36, Issue 4, June 2011, Pages 791–807.
[8]. H. Yahyaoui, L. Wang, A. Mourad, M. Almullah, Q.Z. Sheng,"Towards context-adaptable Web service policies", in Procedia Computer Science, Volume 5, 2011, Pages 610–617.
[9] Harry Jiannan Wang, J. Leon Zhao, "Constraint-centric workflow change analytics", Elsevier journal of Decision Support Systems, Volume 51, Issue 3, June 2011, Pages 562–575.
[10] Shari S.C. Shang, Eldon Y. Li,Ya-Ling Wu, Oliver C.L. Hou, "Understanding Web 2.0 service models: A knowledge-creating perspective", Elsevier journal of Information & Management, Volume 48, Issues 4–5, May 2011, Pages 178–184.
[11] Minhong Wang, Huaiqing Wang, "From process logic to business logic—A cognitive approach to business process management", Elsevier journal of Information & Management, Volume 43, Issue 2, March 2006, Pages 179–193.
[12]. Daniel Żmuda, MarekPsiuk, Krzysztof Zieliński, "Dynamic monitoring framework for the SOA execution environment", in proceedings of ICCS 2010,Procedia Computer Science, Volume 1, Issue 1, May 2010, Pages 125–133.
[13]. Salman Akram, Athman Bouguettaya, Xumin Liu, Armin Haller, Florian Rosenberg, Xiaobing Wu. A Change Management Framework for Service Oriented Enterprises, International Journal of Next-Generation Computing (IJNGC), Vol. 1, No. 1, 09 2010, Pages 1-077.
[14]. Liu, Xumin**,** Akram, Salman, Bouguettaya, Athman, Change Management for Semantic Web Services, 1st Edition., ISBN 978-1-4419-9328-1, 2011.