International Journal of application or innovation in engineering and management

ijaiem.com/July 2023/ Volume 12/Issue 2/Article No-1/16-23
ISSN: 2319-4847

A Structured Method for Confirming the Order of Execution

Sampada.N.Chavanke ,Manipal, Sampada.N.Chavanke @gmail.com

Article Info

Received: 17-04-2023 ***Revised: 21 -05-2023*** ***Accepted: 06-06-2023*** ***Published:15/07/2023***

ABSTRACT: The finest features of the Web and component-based programming are combined in web services. Web service change management has drawn a lot of interest because to the growing need to include dynamic changes to services in long-term composition. Few studies have focused on the methods utilized, despite the fact that several have attempted to provide an optimal solution for dynamic changes. In this work, we concentrate on offering a structured method for assessing the modifications. Finite State Machines have been used to verify the sequence of execution in Long Term Composed Services (LCS), and the Passport system has been used as a case study to clarify the process.

KEYWORDS: Service Oriented Architecture, Web Services, Change Management, Business Policy, Long term Composed Services

INTRODUCTION

Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications. The rapid adoption of Web services is motivating a paradigm shift in enterprise structure from the traditional single entity to a collaboration of Web services. Such enterprises open the door of entrepreneurship to all Web users by facilitating functionality outsourcing on the web. The dynamically changing business environment, however, acts as a hurdle to the success of a business when incorporation of the changes without any issues is considered.

Change management involves a set of processes that are employed to ensure that significant changes are implemented to a business process during its maintenance phase. The purpose of the change management process is to ensure that: business risk is managed and minimized; standardized methods and procedures are used for efficient and prompt handling of all changes; all changes to service assets and configuration items are recorded in the configuration management system; and all authorized changes support business needs and goals.

Today organizations in all industries particularly financial services, retail and communications are increasingly dependent upon IT and a highly available network to meet their business objectives.

The necessity for change increases with the market demand and technology. Though there are many existing change management approaches to satisfy the normal changes, they fail to support the evaluation of dynamic changes within the business constraint.

Therefore, this paper focuses on providing a formal approach for the evaluation of the changes made. Any change made to the logic should not drastically affect the actual order in which the entities are executed, i.e. the change should not affect the order of execution to such an extent that the actual nature of the composed service is altered. So, the verification of order of execution in Long term Composed Services (LCS) has been performed using Finite State Machine. The use of standard and formal approach assures efficiency of the change evaluation.

RELATED WORK

Cuadrado.F et al. [1] proposed a method for automating management operations which provides self-configuration capabilities over the services infrastructure which first defines a model covering all the information required for automating the management of the system, including the means to describe the system and diagnose its correctness (through the stability and desirability formulas) and then describes a satisfiability-based engine that can diagnose the health of any given configuration, and in case it is incorrect, explore the potential solutions and propose the required changes for reaching a new, correct state. It further presents a mechanism for reconfiguring the runtime system through

International Journal of application or innovation in engineering and management

ijaiem.com/July 2023/ Volume 12/Issue 2/Article No-1/16-23
ISSN: 2319-4847

the application of the identified changes. [2] proposed an Ev-LCS, an end-to-end framework that specifies, reacts to, and verifies top-down changes in a LCS. This framework first proposes a formal model which provides the grounding semantics to support the automation of change management and a set of change operators that allow specifying a change in a precise and formal manner by proposing a set of algorithms to automatically implement them.

It then proposes a change enactment strategy that actually implements the changes. Dimitris Apostolou et al. [3] proposed an ontology-based approach for developing and maintaining e-Government services that can effectively deal with changes which enables the systematic response of e-Government systems to changes by applying formal methods for achieving consistency when a change is discovered and also enables the knowledgeable response of service designers and implementers to changes by utilizing design rationale knowledge. Sabri MTIBAA and Moncef TAGINA [4] present a change management framework for a citizen-centric healthcare service platform. A combination between Petri nets model to handle changes and reconfigurable Petri nets model to react to these changes are introduced to fulfill healthcare goals. S. Mtibaa and M. Tagina [5] present a distributed telemedicine environment reaping from both the benefits of Service Oriented Approach (SOA) and the strong telecoms capabilities.

PROPOSED METHODOLOGY

ORDER OF EXECUTION

In a business logic L encompassing set of rules R, functions F, parameters P and dependency D, change evaluation can be determined based on the order of execution of rules and functions. In a normal structural programming language, the order of execution depends on the control, branch and functions.

In an object oriented programming, the message sequence determines the order of execution. This concept is extended further and used in the business logic for analyzing the dependency. Example- In a Passport system domain, consider a change request for verifying the age of the applicant (e.g minor, major) while checking the age, minor means need to check parents citizenship and other proofs.

FSM REPRESENTATION AFTER CHANGE

Here after implementing the change request, order of the execution of the program changes which is shown below. This includes an additional transition from $q_{01\ to}$ qo, qo to q_{11} and q_{00} to q_{01}.

Here the state represents rule and transition is represented by using the symbol δi which includes the current state and the input which may also be an internal transition.

Within the internal transition, the state is the function and similarly its transition includes current state and input which includes parameter set, policy set and dependency set. Each state has an exceptional state which decides whether that state can be rolled back or not.

Logic is said to be executed successfully, if each and every rule and function under it executes in order. Any change in the order of execution of rules is mapped in the dependency set of that rules. The order of execution is the evaluation methodology in change management that assists in the change measure. In response to the changes from the analyst, the source manager sorts out the required logic.

The corresponding logic is decomposed into rules, functions and parameters. Then the requested change is fetched from the corresponding rule set or function set. The fetched rule or function is analyzed with the dependency set for consistency. The transition function for that change is analyzed and an equivalent FSM is generated. The next state of the particular rule or function is predefined using the FSM state transition table. This STT can be utilized to provide the appropriate control flow in the logic.

Algorithm Change Measure (Order of Execution)

Input: Change Specification cs_o (Execution order)

begin

Analyze the change specification cs_0 for completeness and finiteness

for all cs_0 !null

if (rule | function | parameter) in cs_o is !complete **then**

International Journal of application or innovation in engineering and management

ijaiem.com/July 2023/ Volume 12/Issue 2/Article No-1/16-23
ISSN: 2319-4847

Discard request
Current_state:= previous_ state
else
Map cs_o with the existing logic set L
if $cs_o \in L$ **then**
Retrieve the corresponding rule, function and parameter from L
Modify the **STT** (State Transition Table) as per the cs_o
else
Add cs_o to the L with modification in the **STT**
$L' := L \cup cs_0$
end if
Update the **STT** based on cs_o
Current_STT := (previous_STT,cs_0)
Current _state:=(previous_state,Current_STT)
if Computability (previous_state,Current_state) **then**
compute (ΔCM_0)
else
Discard changes
Restore previous state
End

Fig 1: Algorithm for Order of Execution

The above algorithm provides an effective approach for analyzing and evaluating the changes based on order of execution using the STT (State Transition Table).

BEFORE CHANGE

In the evaluation of change request using order of execution, input in the transition includes parameter set and function set. With the help of transition, order of execution can be easily identified.

STATE TRANSITION TABLE BEFORE IMPLEMENTING THE CHANGE

Here the δ_0 (R2, $\boldsymbol{\delta}_{01}$) represents the transition in which R2 represents the current state and δ_{01} represents the input for δ_0 and it is also the internal transition for δ_0.

STATE TRANSITION TABLE AFTER IMPLEMENTING THE CHANGE

After implementing the requested change, the internal transition δ_{11} of state q1 includes another internal transition δ_{01} which goes to the internal state q_{12} through state q0 which finally again goes to the internal state q_{11} of the state q1.

International Journal of application or innovation in engineering and management

ijaiem.com/July 2023/ Volume 12/Issue 2/Article No-1/16-23
ISSN: 2319-4847

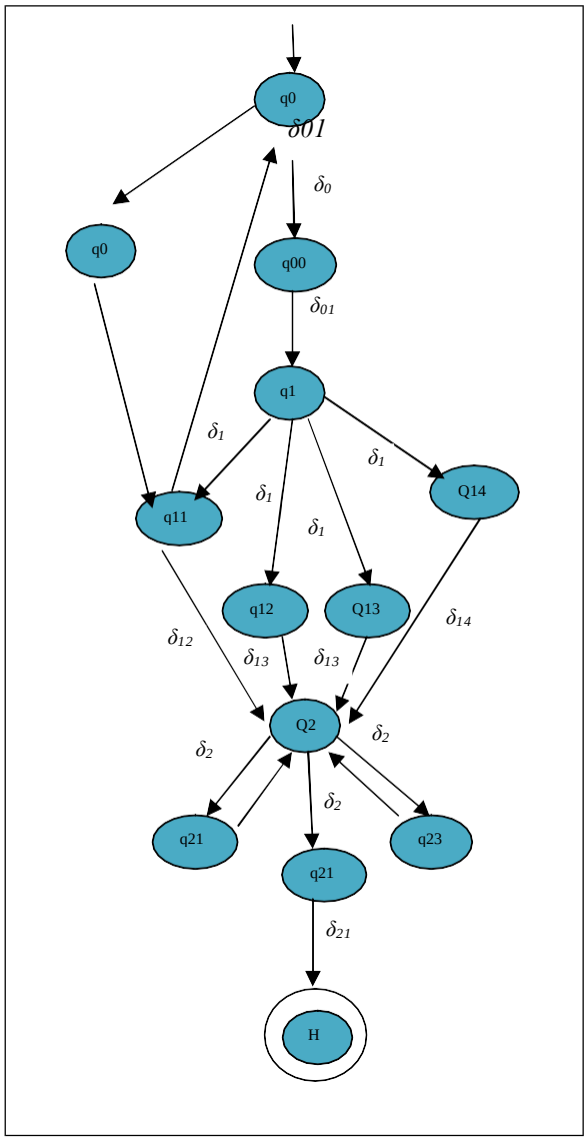

Fig 2: Constructed Finite State Machine

Table 1: State Transition Table before change

CURRENT STATE	TRANSITION		NEXT STATE
q0	δ_0(R2, δ_{01})	δ_{01}(q_{00}, {P1, P_8} {q_{01}, q_{00}})	q1
q1	δ_1(R3,{δ_{10}, δ_{11}, δ_{12}, δ_{13}, δ_{14}})	δ_{10}(q_{11}, { P_8, P_{14}, P_{15} } { q_{00}})	q2
		δ_{11}(q_{12},	q2

International Journal of application or innovation in engineering and management

ijaiem.com/July 2023/ Volume 12/Issue 2/Article No-1/16-23
ISSN: 2319-4847

		{ $P_{16,}$ P_{17} } { q_{00}})	
		δ_{12}(q_{13}, { P_{18}, P_{19} } { q_{00}})	q2
		δ_{13}(q_{14}, {P_{20} } { q_{00}})	q2
q2	δ_2(R4,{δ_{01},δ_{10}})	δ_{20}(q_{21}, {P_{21},P_{22}, $P_{23,}$$P_{24}$} {$q_{00}$, q_{21}})	q2
		δ_{01}(q_{22}, { $P_{26,}$ P_{27}, P_{28}} q_{00}, q_{21}})	q2
		δ_{21}(q_{22}, { P_{29}, P_{30}} { q_{00}, q_{21}})	H

Table 2: State Transition Table after change

CURRENT STATE	TRANSITION		NEXT STATE
q0	δ_0(R2, δ_{01})	δ_{01}(q_{00}, {P1, P_8} { q_{01}, q_{00}})	q1
q1	δ_1(R3,{δ_{10}, δ_{11}, $\delta_{12,}$ $\delta_{13,}$ δ_{14}})	δ_{10}(F4, { $P_{8,}$ $P_{14,}$ P_{15} } { q_{00}})	q2
		δ_{11}(q_{12}, { $P_{16,}$ P_{17} } { q_{00}})	q2
		δ_{01}(q_{01}, { $P_{14,}$ P_{15} } { q_{00}})	F2
		δ_{12}(q_{13}, { P_{18}, P_{19} } { q_{00}})	q2
		δ_{13}(q_{14}, {P_{20} } { q_{00}})	q2
q2	δ_2(R4,{δ_{01},δ_{10}})	δ_{20}(q_{21}, {P_{21},P_{22}, $P_{23,}$$P_{24}$} { q_{00}, q_{21}})	q2

International Journal of application or innovation in engineering and management

ijaiem.com/July 2023/ Volume 12/Issue 2/Article No-1/16-23
ISSN: 2319-4847

		$\delta_{01}(q_{22}$, { P_{26}, P_{27}, P_{28}} { q_{00}, q_{21}})	q2
		$\delta_{21}(q_{22}$, { P_{29}, P_{30}} { q_{00}, q_{21}})	H

Thus the Order of Execution has been verified using formal and standard approach. The use of Finite State Machine assures the efficiency in the estimation of order of execution.

Sample OOE Evaluation: The FSM generated for order of execution involves the rules, functions and parameters states through which the transition takes place. In the below transition table, transition is formulated as start state q0 to the business logic which involves rules functions and parameters. The * denotes that transited state involves composite elements. Thus the start state q0 initially enters the rule 1 which is again composite R1*.

Table 3: State Transition Table

Start State	Transition	Next State
q0	∂(qo,BL*)	q2
q0	∂(q0,R1*)	q1
q1	∂(q1,{F1,F2..})	q2
q1	∂(q1,Exception)	E
q2	∂(q2,Exception)	E

Table 4: Change Measure Table

Transition id Tid/Tid'	Rule id Rid/Rid'	Function id Fid/Fid'
1	R1	-
2	-	F1
3	-	F2

Table 5: Sample OOE Calculation

Tid	Rid	Rid'
1	R1	R1
5	R2	R3
8	R3	R2
12	R4	R4

Therefore OOE= 2/4 = 50% [50% deviation in order of execution]

The following graph in figure 3 shows the percentage of deviation in order of execution observed considering the similar requests for a sample LCS set. The adoption of formal methodology has increased the detection of deviation which implies that the evaluation process has been fine-tuned.

International Journal of application or innovation in engineering and management

ijaiem.com/July 2023/ Volume 12/Issue 2/Article No-1/16-23
ISSN: 2319-4847

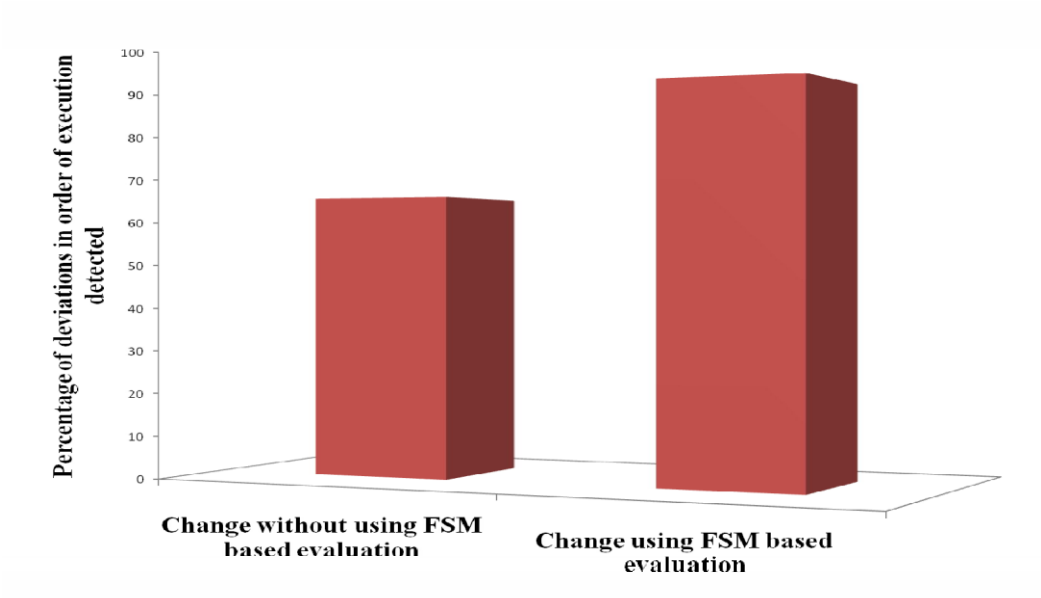

Fig 3: Percentage of deviations in Order of Execution observed

CONCLUSION AND FUTURE WORK

The proposed methodology, Finite State Machine thus efficiently performs the evaluation of the changes made. Order of execution in LCS has been evaluated with the aid of the state transition table. The elucidation of the proposed approach using Passport system as the case study gives a clear idea of the change scenario and the evaluation of order of execution. The future enhancement is to include more factors for change evaluation which will aid in the assessment of the deviations in the functionality of an LCS after a change.

REFERENCES

An Autonomous Engine for Services Configuration and Deployment," by Felix Cuadrado, Juan C. Duenas, and Rodrigo Garcı´a-Carmona, IEEE Transactions on Software Engineering, Vol. 38, No. 3, May/June 2012.

[2] "Ev-Lcs: A System For The Evolution Of Long-Term Composed Services," IEEE Transactions on Services Computing, Issue 99, 2011, by Xumin Liu, Athman Bouguettaya, Jemma Wu, and Li Zhou.

[3] Tomás Pariente Lobo, Barbara Thoenssen, Gregoris Mentzas, Lucitris Apostolou, and Ljiljana Stojanovic, "A collaborative decision framework for managing changes in e-Government services," Government Information Quarterly, Elsevier, 2011.

[4] Sabri MTIBAA and Moncef TAGINA, "Using High Level Petri Net to Manage Changes in Citizen-Centric Healthcare Service Platform," International Journal of Advanced Computer Science and Applications (IJACSA), 2011.

[5] S. Mtibaa and M. Tagina, "Change Management in a Distributed Telemedicine Environment: An Automated Petri-Net Based Approach," Journal of Telecommunications, Vol. 15, Issue 1, July 2012.

In March 2006, Minhong and Huaiqing Wang published "From process logic to business logic—A cognitive approach to business process management" in the Elsevier journal of information & management, Vol. 43, Issue 2, pp. 179–193.

[7] Florian Rosenberg, Xiaobing Wu, Armin Haller, Xumin Liu, Athman Bouguettaya, and Salman Akram. The article "A Change Management Framework for Service Oriented Enterprises" was published in the International Journal of Next-Generation Computing (IJNGC) in 2010.

[8].Shuying Wang and Capretz, Miriam A. M. IEEE International Conference on Web Services, "A Dependency Impact Analysis Model for Web Services Evolution," IEEE Computer Society, 2009.

[9].Ying Zou, Jin Guo, and Hua Xiao. IEEE International Workshop on Systems Development in Service-Oriented Business Environments (SDSOA'07), "Supporting Change Impact Analysis for Service Oriented Business Applications."

[10]. The article "From process logic to business logic—A cognitive approach to business process management" by Minhong and Huaiqing Wang appeared in the Elsevier Journal of Information & Management in March 2006, volume 43, issue 2, pages 179–193.

[11]. "Towards context-adaptable Web service policies" by H. Yahyaoui, L. Wang, A. Mourad, M. Almullah, and Q.Z. Sheng was published in Procedia Computer Science, Vol. 5, pp. 610–617, 2011.

[12]. "Dynamic monitoring framework for the SOA execution environment" by Daniel Żmuda, Marek Psiuk, and

Krzysztof Zieliński was published in the Proceedings of ICC 2010, Procedia Computer Science, Vol. 1, pp. 125–133, Issue 1, May 2010.