Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

Semantic Enterprise Architecture Oriented Test case Generation for Business Process

M. Rahmanian[1], R. Nassiri[2], M. Mohsenzadeh[1], R. Ravanmehr[2]

[1]*Department of computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.*

[2]*Department of computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran.*

Article Info

Article History:

Received 10 August 2021
Reviewed 10 September 2021
Revised 22 November 2021
Accepted 24 November 2021

Keywords:

Enterprise architecture
Business process
Syntactic and semantic modeling
Test case generation

*Corresponding Author's Email Address: *r_nasiri@iauctb.ac.ir*

Abstract

Background and Objectives: The area of enterprise architecture encompasses various domains, the most complicated of which concerns developing enterprise business architecture. Although many state-of-the-art enterprise architecture frameworks describe the architecture by abstract levels, they still fail to provide accurate syntactic and semantic descriptions. Several previous conducted studies were looking for different objectives elaborated on modeling enterprise architectures. However, none of those studies tried to develop a modeling that generates test cases which would later be used for validation and/or verification. Therefore, the main contribution of this study is generating a set of test cases based on the descriptions yielded from enterprise business processes in early steps; then, the amount of later reviews and changes can be significantly lessened.

Methods: Following the objective of accurate validation and/or verification of the enterprise business processes within an enterprise's architecture development, this paper proposes a new method based on the enterprise architecture design. Throughout the iterative cycle of the proposed method, initially, the enterprise goals will be extracted based on the TOGAF framework. Afterwards, it will be subjected to syntactical modeling based on the Archimate language. Then, semantics will be added to the syntactic model of the enterprise business processes based on the WSMO framework and formalize manually to B language by using defined transition rule. Therefore, in order to discover test cases, a set of test coverage will be tested on the formal model.

Results: The proposed method has been implemented in the marketing and sales department of a petrochemical corporation, where the results show the validity and also the effectiveness of the method. Based on the implementation of our method on the selected case study, the details of the business process have been defined based on an enterprise level, the level of abstraction is decreased by syntactic and semantic modeling of enterprise architecture description, the formal descriptions created using the proposed transition rules for sampling.

Conclusion: The proposed method starts from the goals of enterprises; therefore, the output samples are efficiently precise. By adding semantics to the syntactic models of enterprise architecture, the degree of abstraction has been decreased. By creating a formal model, the model can be subjected to sampling. For future work, it is suggested to use the proposed method for the automatic generation of codes.

Introduction

Enterprise architecture is a comprehensive integrated approach that separates and analyzes an enterprise in various aspects and objects from an engineering, but IT-based point of view, to acquire a better understanding of

the entire structures and elements of an enterprise, as well as the forms in which they are connected. Currently, there are multiple enterprise architecture frameworks available worldwide, all of which provide a highly abstract description of an enterprise architecture [1]. In this context, one of the most deployed among those frameworks is the TOGAF, which is based on an iterative development method known as ADM [1]. Based on the TOGAF framework, every enterprise architecture is to be shaped of four domain layers known as the business, data, application, and technology architecture layers [2].

Software testing is a critical and costly process in the Software Development Lifecycle. In fact, a considerable portion of the cost of producing reliable software is associated with this process phase [3], [4]. Nevertheless, if one can generate a set of test cases based on the descriptions yielded from enterprise business processes in early steps; then, the amount of later reviews and changes can be significantly lessened. This is the main idea behind of later steps in this paper.

Several previous studies following different objectives tried to model enterprise architecture [1], [5]-[9], [11]-[24]. However, none of those studies elaborated on modeling to generate test cases for the purpose of verification and/or validation. The main issues addressed by this paper is how to start from the enterprise level, get benefited from the enriched descriptions yielded for enterprise business processes, move to generate proper syntactic and semantic models of the descriptions, and generate prioritized test cases from the well-established models to come up with samples to be used for verification and/or validation.

Based on what aforementioned, we may cope with two main challenges:

A) Syntactic and semantic modeling of the enterprise business processes.

B) Generating proper test cases using the syntactic and semantic models.

Compared to previous studies, innovation in this paper is as follows:

- Starting from the enterprise level (vision, mission and goal) to get descriptions of the enterprise business process.
- Reducing the level of abstraction in architectural descriptions using syntactic and semantic modeling.
- Enabling model sampling using formal transition rules.

Later on, in the next section the related studies would be reviewed, and once the proposed method is explained, it would be implemented in the form of a case study for further evaluation purposes. Finally, this paper ends up with the proposed method conclusions and further suggestions for future works.

Review of Literature

A. Syntactic and Semantic Description of Enterprise Architecture

Several studies have tried to provide syntactic and semantic descriptions of enterprise architecture, while each of them used the descriptions for a specific purpose.

In [1], Zhou et al. to identify, classify, analyze, and evaluate existing methods for EA visualization, reviewed the research papers on EA visualization systematically. They selected and analyzed 112 research papers, and then they categorized them according to their purposes. In none of the studies reviewed in this study, the issue of modeling with the aim of generating test cases has been addressed. In [5], Bouafia and Molnar defined the basics concepts of EA and the purpose and utility of an EA and its place in the IS environment are discussed. The approach presented provides a formal way to use the mathematical analytic methods for exploring misalignment based on different concepts and relation between them. In [6], Hinkelmann et al. believe that modeling for humans is different from modeling for machines. They proposed a combined approach that would be suitable for both humans and machines. In order to create a graphical modeling language, a graphical symbol was designed for each ontology. However, the perspective proposed in this research is to be known as general and not pertaining to any specific domain, particularly while someone is seeking a test case generation solution at this level. In [7], Babkin proposed a method for detecting any logical paradoxes in enterprise architecture models that works under the approach of model checking (verification) in the business process model. Babkin's study uses the ArchiMate language and the MIT Alloy Analyzer tool to describe enterprise architecture and to analyze model limitations, respectively. Furthermore, the study has also developed an editor module that translates enterprise architecture models to the MIT Alloy Analyzer system's language. The main drawback of this model may be its failure to support semantics, whilst the major effort was mainly on model consistency check and syntax matters among models. In [8], Caetano attempted to address three major issues. The first one was about how to use ontology to present enterprise models; the second one was how to use ontology to integrate enterprise models; and the third one was how can use semantic computing techniques to analyze integrated enterprise models. He stated that the main challenge should be on the determination of mapping function among different schemas. Sometimes based on the extent of semantic difference among various schemas, it may become virtually impossible to select a mapping function. This study stated that a conceptual model could be

determined via three components, namely as the subject, the interface, and the object. Through this method, the concepts underlying each of the model components would be described in an ontological fashion. The main drawback of this research is its focus on semantic modeling where sampling is ignored. In [9], Hinkelmann et al. proposed a combined modeling approach for convergence between business and technology. The authors used enterprise ontology proportional to BPaaS concepts. It is noteworthy that OMiLAB LifeCycle backs the development of the BPaaS design environment. The authors believed that the BPaaS ontology is the format of ArchiMEO ontology [10]. The study has expanded various models with different algorithms and mechanisms of semantic transformation to connect graphical models to the BPaaS ontology. The main pitfall here is deployment area if the proposed method for other cases is not considered by authors. In [11], Gokalo believes that the complexity of enterprise architectures is the reason that manual analysis seems to be impossible, and so he proposed using descriptive logic along with ontology. This study categorized inferential activities into five main groups: subscription, sample inspector, relation inspector, compatibility of concepts, and compatibility of the database. Each of the mentioned activities was being described by a different descriptive logic also with varying descriptive capabilities. The study used OWL to provide a high-level description of the meta-model. Using the stated descriptive logic, concepts relating to the elements and the relationships among them had been described in ArchiMate. However, still no sampling capability is available in this method. In [12], Chen et al. suggested that semantic technology should allow different datasets extracted from different data sources in enterprise, to be later integrated into an EA repository, and would prepare the basic information required for decision making corresponding to the outlook of the information systems at hand. In addition, enterprise architecture frameworks such as the TOGAF produce meta-models to be used as guides for generating EA repositories. Considering this content, the authors defined a process for generating SEAM repositories. Furthermore, the data have been subjected to ontology and related to the enterprise architecture. SEAM focuses on modeling the dependencies among the business, information systems, and IT infrastructures. However, still no sampling capability is available in this method. In [13], Hinkelmann et al. developed a framework intended to make a balance between technology and business. Model-based engineering is presented either as a graphical or as a formal model. Enterprise architecture frameworks solely display a general schema of the enterprise architecture and its structures and elements, while in some cases such as the Zackman framework, there is a lack of any specific modeling instrument. As a result, such frameworks cannot be used as a tool for decision making. The study assumed that no language can provide a formal description of an enterprise architecture definition since a perfect modeling language is the one that encapsulates the three components of syntax, semantics, notations, and symbols.

B. Generating Test Cases Based on Enterprise Architecture Description

An important debate pertaining to the domain of software testing includes generating test cases which are normally generated in different forms from various software models. In the following, previous studies related to this domain are to be discussed. Since the present study focuses on enterprise business processes, we will only discuss the studies that fall into this category.

In [3], Sharma et al. identified various factors affecting related aspects of software testing process and therefore the impact of ontology has been observed in the testing and analyzed. They believe that such an elucidation is significant for having knowledge-oriented verification and validation and the wide adoption of ontology helps the domain in manifolds. In [14], Yazdani et al. present a model-based approach to automatically generate test cases from business process models. They first model business processes and convert them to state graphs. Then, the graphs are traversed and transformed to the input format of the "Spec explorer" tool that generates the test cases. The limitations of the proposed algorithm is that it works only for well-structured processes and if the input process is not well-structured, it cannot define states correctly and it so captures invalid paths. In [15], Zhang et al. developed a tool for the automated generation of test cases based on descriptions. Their study used preconditions and post-conditions to produce formal descriptions. Their method was thoroughly based on programming which was unusable for sets containing infinite elements. In [16], Ajay et al. used the activity diagram for the automated generation of test cases. In their method, an activity flow table was established based on the activity diagram, and then based on the former table, an activity flow graph will be yielded. In addition, their study used the Genetic Algorithm to generate a set of optimal test cases. It is noteworthy that their method generates the set of test cases according to the structure of the activity diagram via selecting different paths within it. In [24], Bures et al. created a tool named PCTgen which automatically generated a set of test cases to check a workflow. Unlike previously presented methods, this method assumed that there were no UML documents available. To this end, a directed graph will be used to signify the

workflow. The test cases generated through this method are resulted solely from the sequence of activities existing in the work flow, irrespective of semantics.

C. Motivation of the paper

Investigating the previous studies led us to the conclusion that although many studies have tried to generate semantic models of enterprise architecture, but none of them, have adopted the approach of test case generation with the aim of verification and/or validation using syntactic and semantic models. Therefore, considering enterprises objectives and missions, we intend to generate proper test cases for further use in the system through a new method known as "Semantic Enterprise Architecture Oriented Test Case Generation for Business process ", which is based on the views relating to the TOGAF and model checking of a formal model of syntactic and semantic modeling.

The Proposed Method

The overall framework of the proposed method is based on an iterative cycle, which is derived from the Architecture Development Method of the TOGAF, which itself is a stepped iterative process. The proposed method doesn't elaborate on the transition from the existing status of the enterprise to the desired status; rather than, in this method, the only input is the architecture of the desired status which will be further developed via an iterative cycle. The overall framework of the proposed method is presented in Fig. 1.

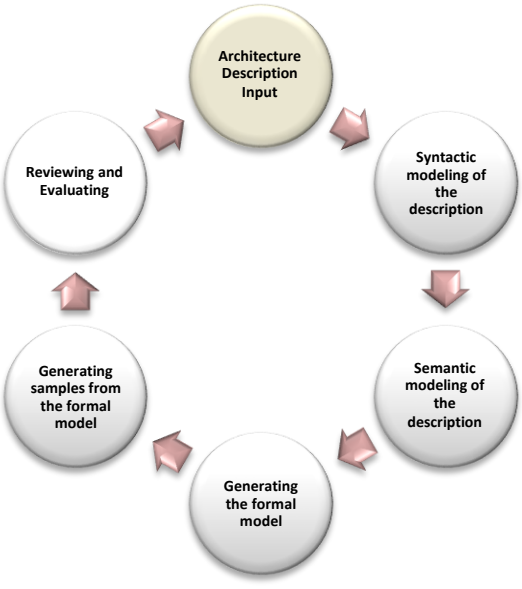

Fig. 1: The overall framework of the proposed method.

A. Receiving the Enterprise Architecture's Description

Based on the TOGAF's view, the entire business processes of an enterprise are rooted and validated in its objectives and missions. At this phase, we will start from the enterprise-level objectives and missions to reach the enterprise business processes. In the meantime, it is assumed that the descriptions have been received from the domain experts. It is worthy of mentioning that the mentioned descriptions have been provided in an oral and/or semi-documented unofficial manner, and hence could not be directly subjected to sampling.

B. Syntactic Modeling

Each model depicts a part of the reality, but a single model alone cannot express all the realities by itself. Since the received (input) descriptions are oral and/or semi-documented and unofficial, at this phase, the products of TOGAF will be generated using the ArchiMate [17]-[20] language and according to the received data. Since the present paper focuses on enterprise business processes, we will only elaborate on the required products corresponding to the TOGAF and the business architecture layer.

The steps involved in syntactic modeling are described in Table 1, as displayed in Fig. 2.

Fig. 2: Steps involved in syntactic modeling.

At the first step, we determine the goals and their respective hierarchies of details based on the extracted descriptions. For each goal, an enterprise face certain requirement. At the second step, we define respected requirements of goals. In order to meet the expressed requirements, one or more proper solutions will be proposed. At the third step, solutions proportional to different requirements will be determined. At the fourth step, to provide an overview that traces essential elements to be built or revised from goals through to components, we create a business footprint model. The Footprint is a complete collection of process, data, application, business unit, and business objective that validates a capability as in TOGAF and finally at the fifth step, to model the entities identified in a business process and the relationship among them, the business entity model is created.

C. Semantic Modeling

Syntactic models are unable to cover the entire knowledge pertaining to an enterprise alone. Additional data required to describe a model and should be expressed in the form of business rules. These rules are either defined by the process, or by the entities existing in a process. To this end, it would be necessary to provide a semantic description of the enterprise business processes. The algorithm used for semantic modeling has been illustrated in Fig. 3. Nevertheless, we use the semantic modeling process by concepts involved

in WSMO [21]-[27]. This is because of the capability of WSMO as a semantic modeling language compliant with the aims of this paper.

For semantic modeling of an enterprise business process, at the first step, the business entities are semantically modeled to determine the ontology of business entities by specifying the name, attributes, types, and constraints on each entity. At the second step, we use axioms to express the rules governing a business process, and finally, at the third step, for modeling the goal of a business process, we use the Goal concept in WSMO to express the goal of an enterprise business process.

Fig. 3: Steps in semantic modeling.

D. Creating the Formal Model

Since semantic models developed with the WSML language [21] cannot be subjected to sampling, in order to provide the required conditions for sampling, at this step, a formal model that can be subjected to sampling will be generated from the description. In order to express the description in a formal sense, the B language [28], [29] will be used. The reason by which the B language is selected for use in the proposed method is that the target description is based on the Abstract State Machine model. Similarly, description B is also based on the abstract state machine. Language B is equipped with suitable supports for validating and checking the model; therefore, the model created using the B language will be adequately suitable for sampling at the next phase. We have defined a set of rules to transform semantic descriptions from WSML language to the B formal language. The description generated in WSML language contains two major parts, the first being the ontology of concepts, and the second being goal description.

I) Rules of Ontology Transformation

Ontology is the main part of WSMO and comprises three parts: the header, concept, and rules. The following rules are abided, while transforming into the B language.

Rule 1: Transforming the Ontology Header.

The header is comprised of several parts of the name, imports-ontology, mediator, and non-functional properties. However, since the sections of mediator and non-functional properties are ineffective, they won't be used in transformation. To this end, according to the relation (1) in Table 1, the transformation of machine B will be executed under the same name given to the defined ontology.

Rule 2: Transforming the Imports-ontology in the ontology.

If a specific ontology is added to the header, it will be added to the machine in the INCLUDES section according to relation (2) in Table 1.

Rule 3: Transforming Concept names in the Ontology.

The concepts in a WSMO ontology are transformed to sets in a B machine in the SETS section. The deferred sets in B usually declare the sets. A deferred set is one that is not initialized at the time of the set declaration. However, the explicit initialization of a set is represented by the initialized set. The set initialization can also be used to map the inheritance of the concepts. A concept with multiple sub-concepts can be transformed as the initialized set with elements representing its sub-concepts.

A concept comprises three sections, namely the name, attribute, and (attribute) type. Concepts are transformed into language B using relation (3) in Table 1. In this sense, the concept's name will be transformed into a set's name (in capital letters) in the SETS section. It is noteworthy that these sets will not be primarily quantified during defining.

Rule 4: Transforming the Attributes of Ontology Concepts.

The attributes of a concept in a WSMO ontology are transformed using the B relations. An attribute of a concept is transformed as a relation over the set representing the concept and the set representing the type of the attribute. Such relations are defined in the INVARIANT section of the B machine. The attribute of a concept is defined as a variable for defining this relationship.

The attributes will be transformed into the B language in the form of relations and based on relation (4) in Table 1. It is worthy of mentioning that the attributes will be defined in the VARIABLES section of the machine.

Rule 5: Transforming the Attribute Types of Ontology Concepts.

Attribute types are defined as a relation from a set representing the concept to a set representing the attribute type. These relations are defined in the INVARIANTS section of the machine based on relation (5) in Table 1. The type of a variable can be one of the main defined types or the types added to the machine in the imports-ontology section.

Rule 6: Transforming the Rules of Concepts.

Rules are constraints expressed in a logical form. In WSML language, rules are added in different sections under the name of axioms in order to show a restriction. For mapping, it is necessary to transform the mappings between the operators from WSML to B language. Using the mappings between different operators that

expressed in the Table 2, rules can be transformed into the B language. It is noteworthy to mention that axioms will be written in the INVARIANT section of the machine.

Table 1: Rules of transforming the ontology from WSML to B language

Relation	Relation Transformation Rule
Header name Transformation	**Ontology**(ontology-name)- > **MACHINE**(ontology-name)Machine (1)
Imports-Ontology Transformation	**Imports-Ontology**(ontology-name)- > **INCLUDES**(ontology-name-Machine) (2)
Concept Name Transformation	**Concept**(name)-> cap(name) in **SETS** (3)
Concept Attributes Transformation	**Concept**(Attribute)- > var in **VARIABLES** (4)
Attribute Type Transformation	Attribute(Type)- > **VARIABLES**(var) : **SETS**(concept) < - > Type (5)

Table 2: Transforming the operators from WSML to B language

Description	WSMO operator	B operator
Conjunction	And	&
Disjunction	Or	or
Negation	Neg, naf	Not
Universal quantifier	For all	!x
Exist quantifier	exists	#=
Equality	=, :=:	=
Inequality	!=	/=
Implication	Implies, ImpliedBy	=>
Reverse implication	ImpliedBy	=>
Membership	memberOf	:
Typing	ofType, impliesType	:
Inheritance	subConceptOf	:>

II) Rules of Goal Transformation

The goal shows the system's behavior and performance in the user's view. A goal's description includes three parts, namely as header, capability, and interface.

A goal specification G is defined as a 3-tuple G = (H, I, C), where H is a goal header, I is a goal interface specification, and C is a goal capability. Below we describe the mapping in the same order.

Rule 1: Transforming Goal Header.

The header of a WSMO goal Specification consists of names, imports-ontology, mediator, and non-functional properties. Since they do not affect the non-functional and the mediator, we do not use them in transformation. As shown in relation (6) in Table 3, the goal declaration is transformed to a B machine declaration by the MACHINE statement. Note that the symbol "->" denotes the "is transformed to" statement. This means that the goal declaration is transformed to the machine declaration in the translated B machine. The naming convention is to use the name of goal Specification, with the suffix "Machine".

Rule 2: Transforming Goal Imports-ontology.

An ontology imported in a goal Specification using the imports-ontology makes all the ontology concepts and instances visible to the goal Specification as if they were included. Therefore, the imports-ontology statement in goal specification is transformed using the INCLUDES statement in the B machine that also makes the included machine visible and accessible in the including machine. The B machine representing the ontology is imported using the INCLUDES statement in the B machine representing the goal Specification. This is shown in relation (7) in Table 3.

Rule 3: Transforming Goal Capability.

Capability is determined with four parameters namely the precondition, assumptions, post-condition, and effects. Each of these parameters is a set of axioms, therefore using the previously mentioned rules for transforming axioms; they will be transformed into the B language via the rules stated in Table 2.

Rule 4: Transforming Goal Interface.

In describing the interface, the parts "signature" and "transition rule" are of importance. States are sets of concepts used in describing the interface. In WSMO, states are based on ASM with the variables in the B machine functioning in a similar way.

Suppose SSIG is the set of states used in the description of the interface, and VAR is the set of variables defined in machine B. in this sense, while mapping the states, according to the relation (8) in Table 3, the set of states will be defined in the variables section. In contrast, their types will be signified in the INVARIANTS section.

Rule 5: Transforming the Rules of Goal Interface Transition.

Suppose that T (G) is the transition rules defined in the target description, and OP(M) is the set of operations defined in machine M. In this case, all transition rules will be mapped into an operation in the machine using the relation (9) in Table 3.

Rule 6: Transforming the Inputs and Outputs of Goal Interface Transition Rule.

An operation is specified with a name; input values, and return values. According to relation (10) in Table 3, the name of the transition rule will be mapped into the operation name while the input parameters and return

values of the transition rule will be mapped into operation input parameters and operation return values, respectively.

Table 3: Rules of transforming the goal from WSML to B

Relation	Transformation Rule
Transforming Goal header name	**Goal**(Goal-name)-> **MACHINE**(Goal-name)Machine (6)
Transforming Goal imports-ontology	**Imports-Ontology**(ontology-name)-> **INCLUDES**(ontology-name-Machine) (7)
Transforming Interface States	**Interface**(SSIG)-> **Machine(**VAR) Type(SSIG)-> VAR : **SETS**(concept) < - > Type (8)
Transforming Transition Rules	T(G)- > OP(M) (9)
Transforming the Inputs and Outputs of Transition Rule	Tri- > Opi Tri(in-concept)= Opi (inArg) ^ Tri(out-concept)= Opi (retype) ^ Tri-name = Opi-Name (10)

E. Generating Samples from the Formal Model

Once the formal model is generated in B language, it will be subjected to sampling to generate test cases, to which end the method of Model Checking will be used. The mentioned method is usually used to study the validity and reliability of state-based formal models. By this method, firstly, a set of traps for formal descriptions are set and subsequently added to the assertion section of formal description; then, the model will be checked. A negative trap is a test predicate obtained through various criteria of test coverage. The model checker searches through different system states for a state in which the assertion is contravened.

We use ProB [30] for model checking and generating test cases. The reason for using ProB is that: first of all, it is fast and automatic; second, it applies a perfect mechanism on the formal description to find contravention instances, and also checks the entire states space; and third, it is suitable for state-based descriptions. The steps involved in this section are shown in Fig. 4.

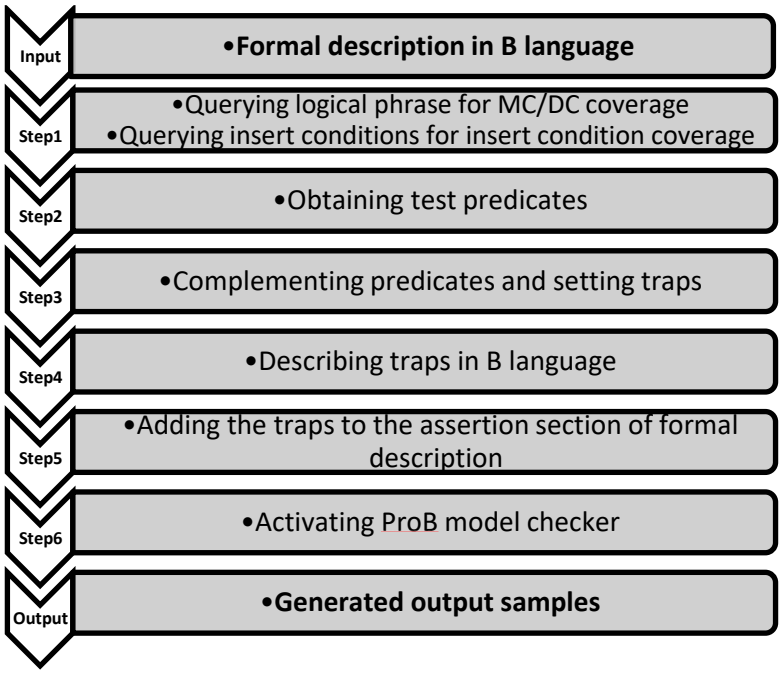

Fig. 4: The flowchart of generating samples from formal model.

I) Obtaining a Trap from Formal B Descriptions

The present paper uses the two criteria of boundary condition coverage and modified condition decision coverage to cover the formal descriptive model. Boundary condition coverage efficiently tests relational phrases, while modified condition decision coverage elaborates on testing predicates and logical phrases.

The traps are obtained based on formal descriptions and criteria of test coverage which are added to the assertion section of the description.

The Boundary Value Testing method tests the system's behavior on the boundary of a variable.

In order to obtain the traps from the MCDC coverage criterion, a logical phrase containing atomic parts will be tested by n+1 test cases. The steps involved in the process of extracting traps from this coverage criterion are described as follows:

1- Determining logical conditions from various parts of the description.

2- Obtaining test predicates for logical phrases through the application of a table-based method.

3- Complementing the entire test predicates and obtaining the traps.

Afterwards, the obtained traps will be added to the assertion section of the description.

Once created, the traps will be consecutively added; then, the model checker will be activated. Since the traps are added to the ASSERTIONS section, we will request the checker to check the model for the defined assertions. If the descriptive model is valid, the checker will run into an error. The steps involved in trap calculation using MC/DC coverage are displayed in Fig. 5.

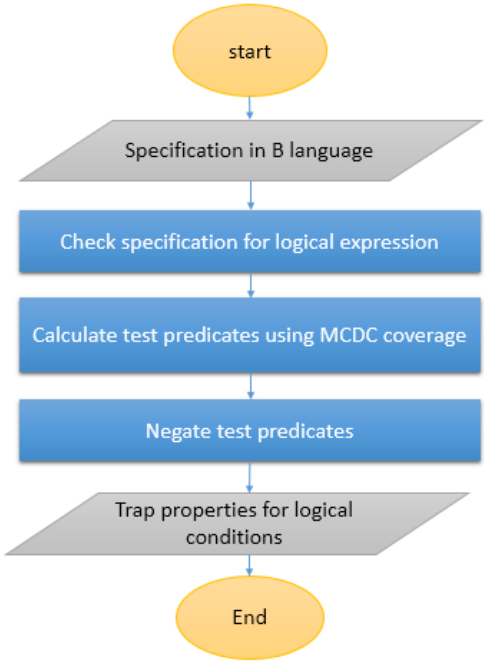

Fig. 5: Steps for trap calculation by using MCDC coverage.

F. Reviewing and Evaluating

Based on the views expressed in the TOGAF, developing an enterprise architecture is a gradually iterative process. At this phase, in case of a need for a change, the evaluation will be made, and the corresponding cycle of the proposed method will be iterated.

Implementing and Evaluating

In order to evaluate the efficiency of the proposed method, we deployed it in the marketing and sales department of a petrochemical corporation as a good case study.

In the following, firstly, the case study will be described, and afterwards, the details and results of the implementation of each section of the proposed method will be studied.

A. Description of the Case Study: Petrochemical Corporation

The vision of this corporation is to become the most well-known producer and distributor of isocyanate in the entire Middle-east.

The studied corporation uses nitric acid and gases such as chlorines, carbon monoxide, hydrogen and toluene to produce high-quality basic petrochemical products such as various isocyanates which are of a higher added value and also to distribute these products in domestic and foreign markets.

The goals of the marketing and sales department of the studied petrochemical corporation have been shown in Table 4, using well-known format of the Balance Score Cards technique.

Table 4: The marketing and sales department goals

Aspect	Goal1	Goal2	Goal3	Goal4	Goal5	Goal6
Financial Aspect	Development of income opportunities	Increased domestic sales	Increased income	Increased exports		
Customer Aspect	Being satisfied by sales staff	Customer loyalty	Being satisfied by the sales mechanism			
Processes Aspect	Development of marketing for new grades	Variability of customers	Development of the communication s process	Improving the process of development of major and regional market studies	Development of mechanisms for improving customers' loyalty	Development of relationships with the global pricing centers
Growth and Learning Aspect	Promoting the personnel's knowledge and skills	Improving the sales mechanism,	Development of knowledge management	Increasing the contacts between the sales and other Departments of the enterprise	Creating a database and promoting integrated Documentation and data bank.	

B. *Syntactic Modeling of the Case Study*

Syntactic modeling of the descriptions has been carried out using the ArchiMate language and the software of Modelio V.7.0. In the following, the steps involved in syntactic modeling, as well as the outputs of each section, will be described. Afterwards, implementing the details of the proposed method will be discussed in terms of a process aspect.

I) *Modeling the Vision Layer*

To provide a full vision that can be used to scope all the work area, the vision phase uses initial schemas of an essentially informal nature. These artifacts are very high level and do not yet involve detailed modeling activities. They will be developed free hand, in the form of images or matrices, in order to prepare later phases. TOGAF defines an enterprise as being a collection of business units with a common set of goals. This shows just how important goals are within an enterprise; they are its reason for existence. Goals are constructed hierarchically. Goals constitute the roots of the goal/objective tree.

According to Table 5, increasing sales and income is a strategic goal for the realization of which there is a need for other objectives, including developing marketing for new grades, leveraging customers, developing the process of efficient communications, developing mechanisms for improving customer loyalty, and developing contacts with global sales centers must be already realized. In order to realize the mentioned requirements, proper and adequate solutions must be found. For instance, in order to realize the determined goals, develop income opportunities, increasing domestic sales, and improving exports, the corporation under study has been suggested to develop its sales process.

Based on the proposed algorithm, in this step, we first draw the Goal model, and then, based on that we draw the requirement model, the solution concept model, and the business footprint model. A solution-concept diagram has been shown in Fig. 6. This model uses preliminary information to share a preliminary vision with all stakeholders by providing general information on the changes that are going to be implemented.

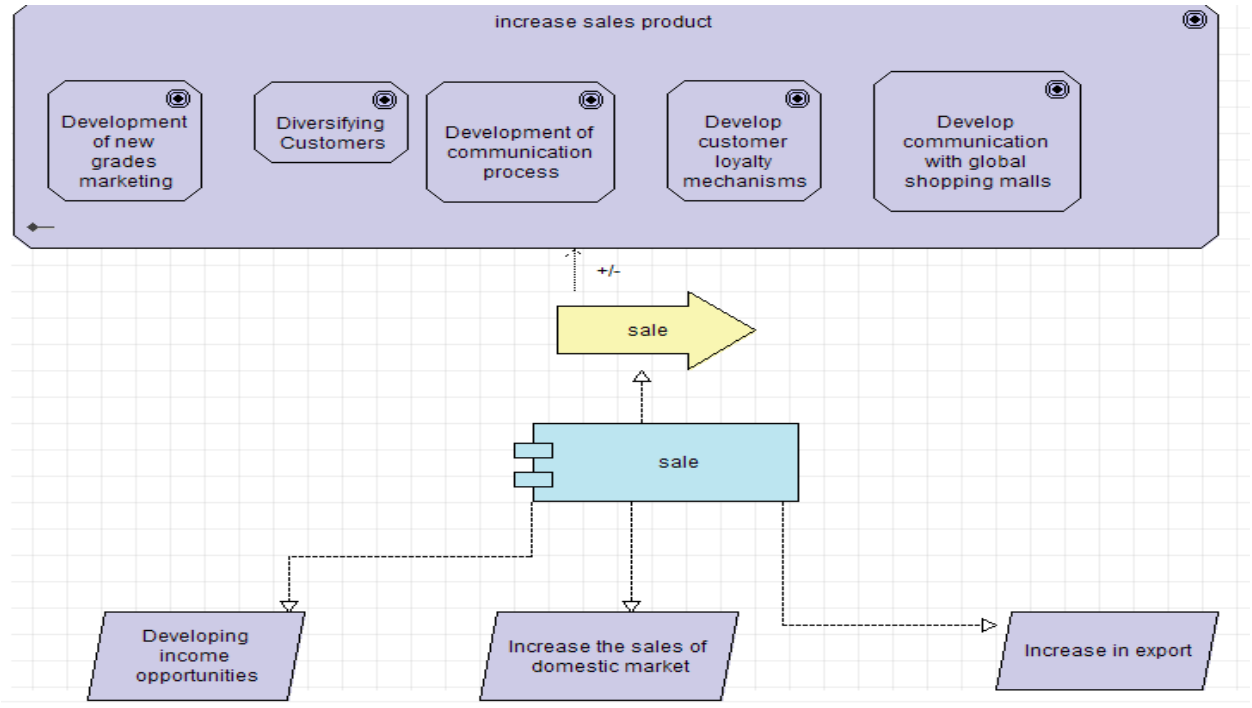

Fig. 6: Solution concept diagram.

II) *Modeling the Business Architecture Layer*

Enterprise architecture puts a very strong emphasis on business architecture. Business architecture endeavors to identify the key business processes to fulfill Business strategies and goals. Based on the obtained information, the sales process group analysis is reported in Table 5. At this step, based on the collected information, models pertaining to the business process are created. A business footprint diagram describes the links between business goals, enterprise departments,

business functions, and business services. These functions and services are also traced with technical components producing the required capabilities. A business footprint diagram is only interested in essential elements that show the connection between organization units and functions in order to produce services. A business footprint diagram has been drawn in Fig. 7 based on the sales department information. It is used to communicate with the management of the enterprise. Business footprint diagrams focus on the current concerns of the business.

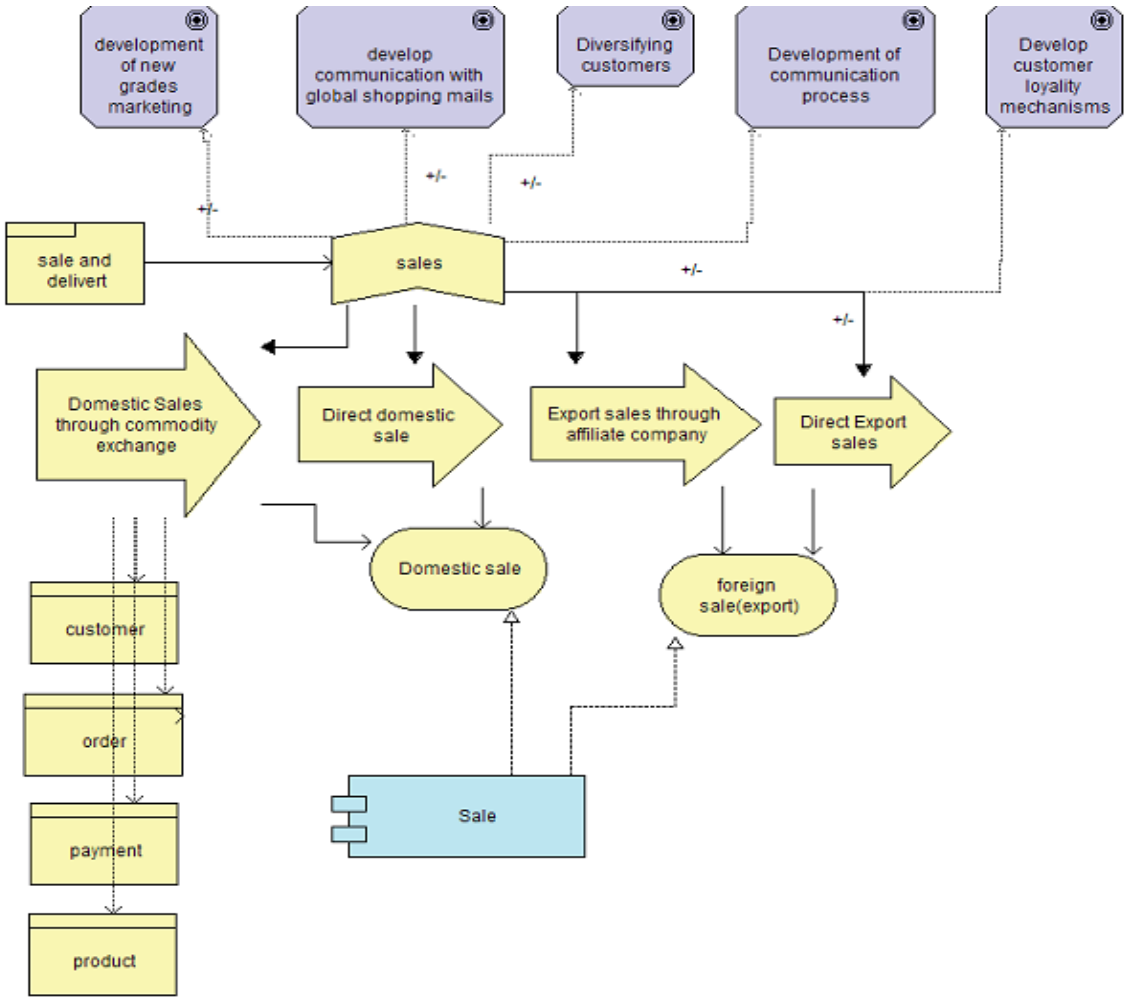

Fig. 7: Business footprint diagram.

The retrieved information shows that in the domain of business knowledge, it is necessary to pay close attention to terminology.

For instance, it must be clear what is meant by customers, purchase request, product, payment, sale bill, freight bill, freighter, and invoice.

Each of the above-mentioned entities is defined by a set of properties and rules governing them. For example, a customer is specified by a name, an ID, an address, and a credit card number. In fact, every customer has their unique IDs, and there could not be any two customers with the same ID. Furthermore, every customer has a unique account number and so on.

C. Semantic Modeling of Business Processes

Among the processes pertaining to the case of the study, we have selected the process of direct domestic sales for the purpose of semantic description. Semantic modeling and the rules governing the client entity are written as follows:

ontology petro_EA_ontology
  nonFunctionalProperties
    wsmostudio#version **hasValue** "0.7.3"
  endNonFunctionalProperties
concept client
  name **impliesType** _string
  surname **impliesType** _string
  Identifier **impliesType** _integer
  address **impliesType** _string
  crediet_card_type **impliesType** _string
  credit_card_balance **impliesType** _integer
  credit_card_number **impliesType** _integer

Table 5: Sales process group analysis

The ID of the Sales and Delivery Process Group			
Existing process group	Sales and delivery	The code of existing process group	EA-company-SAL
The range of existing process group			
Includes the entire products of the company including the final products, middle products, side products, and waste products			
The goals of the existing process group			
Planning and executing the entire activities relating to sales including receiving orders, reviewing orders, sealing contracts, and delivery of products			
Business services			
• Domestic/foreign sales • Product Delivery report • Managing the contracts			
The indices of the existing process group			
• Monetary realization of domestic sales goals • Monetary realization of exported sales goals • Weight realization of domestic sales goals • Weight realization of exported sales			
Main inputs			
Data		**From process**	
Customer needs		Customer services	
Cash received approval		Cash received approval	
Main outputs			
Data		**From process**	
Customer data		CRM	
Sales plan		Production planning and controlling	
Sales invoices		Management of accounts receivable	
Providing product services		Customer services	
Order basket		Supplying the feed	
The owner of the existing process group			
• The chief of sales department			
The beneficiaries of the existing process group			
• petrochemical company as well as the customers			
The existing processes			
1. Domestic Sales through commodity exchange 2. Direct domestic sales 3. Export sales through another company 4. Direct Export sales			

For the client entity, a certain axiom is that account numbers are unique per person (client) and this axiom is described in the following fashion:

axiom uniq_credit_card
definedBy
?x[creditcardnumber **hasValue** ?ccn1] **memberOf** client **and** ?y[creditcardnumber **hasValue** ?ccn2] **memberOf** client:-?ccn1 != ?ccn2.

In order to describe a process, we will try to describe its interface and capability parts. While describing the capability, we use axioms to write the preconditions, post-conditions, assumptions, and effects. For example, one important assumption in this process is that a valid credit card is the one that is either the MCARD or a SCARD. For this purpose, we use an axiom for this assumption named as a valid card. While describing the interface, we will describe the set of states along with the rules of transition among them.

D. Creating the Formal Model

In this section, we use the predefined mapping rules to transform the WSML language semantic description into formal B language. In the first step, the concepts and their attributes will be transformed. The following presents a partial transformation for the concept of client:

MACHINE petroEAontologymachine
SETS
CLIENT
VARIABLES
address,crediet_card_type,crediet_card_balance,creadit_card_number,Identifier,name,surname
INVARIANT
address:CLIENT<->STRING&
credit_card_balance:CLIENT<->INT&
credit_card_number:CLIENT<->INT&
credit_card_type:CLIENT<->STRING&
Identifier:CLIENT<->INT&
name:CLIENT<->STRING&
surname:CLIENT<-> STRING

Describing the goal is comprised of three parts, being the header, capability, and interface, respectively. Since in describing the goal, we have used "input_saleontology". Then, by implementing the expressed rules for transforming axioms, the preconditions, post-conditions, assumption, and effects in the description of capability will be transformed into B language. For example, one assumption maintained in the description is that a credit card is only valid if it is either the MCARD or a SCARD. The following presents the description in both WSML and formal B languages.

Presentation in WSML language:

assumption valid_card
definedBy
?x[credit_card_type **hasValue** Mcard] **memberOf** client **or** ?x[credit_card_type **hasValue** Scard] **memberOf** client.

Presentation in B language:

#(x,credit_card_type).(x:CLIENT & credit_card_type:CLIENT<->STRING => credit_card_type(x)="Mcard" or credit_card_type(x)="Scard")

In the interface part of the goal, we have a set of states and transitions among them. The set of states is written in the VARIABLES section, whereas their types are written in the INVARIANTS section.

E. Obtaining Samples from the Formal Model

In this section, to obtain the traps based on the expressed algorithm in Fig. 6, we used the MC/DC coverage criterion for a logical phrase in the INVARIANT section. Valid_credit_card is a logical phrase in the formal description. The logical phrase is as follows:

A=credit_card_type(x)="Mcard"
B=credit_card_type(x)="Scard")

These two parts are connected by an OR connection operator. In the following, test phrases will be obtained using the table-based method.

Table 6: Sample from logical expression

No.	A	B	A or B	Effect A	Effect B
1	F	F	F	×	×
2	F	T	T		×
3	T	F	T	×	
4	T	T	T		

In Table 6, columns 5 and 6 denote effects A and B, respectively. These two columns signify what parts of a phrase are responsible for the sum of that phrase. In every phrase, the part that results in the occurrence of overall result is referred to as the main part, and the rest are called subsidiary parts.

According to the above table, the test cases of rows 1 and 2 test the effect B, while test cases 1 and 3 test the effect A. as a result, the test cases that test A and B effects are rows 1, 2, and 3. Hence, using these three test cases, one can test the above-mentioned phrase in terms of MCDC coverage.

Afterwards, the obtained test cases will be complemented and then added to the ASSERTION section of the description B.

ASSERTIONS
Not (credit_card_type(x)="Mcard")
Not (credit_card_type(x)="Scard")
Not (credit_card_type(x)/="Mcard" and credit_card_type(x)/=”Scard”)

The third assertion in the last section has been checked using the model checker and displays the output as Fig. 8.

The distance of the fault location from the beginning is the machine mode which is a test case. The result illustrates suitable input and expected values under posed semantic limitations.

A piece of the output graph of the test case has shown in Fig.9.

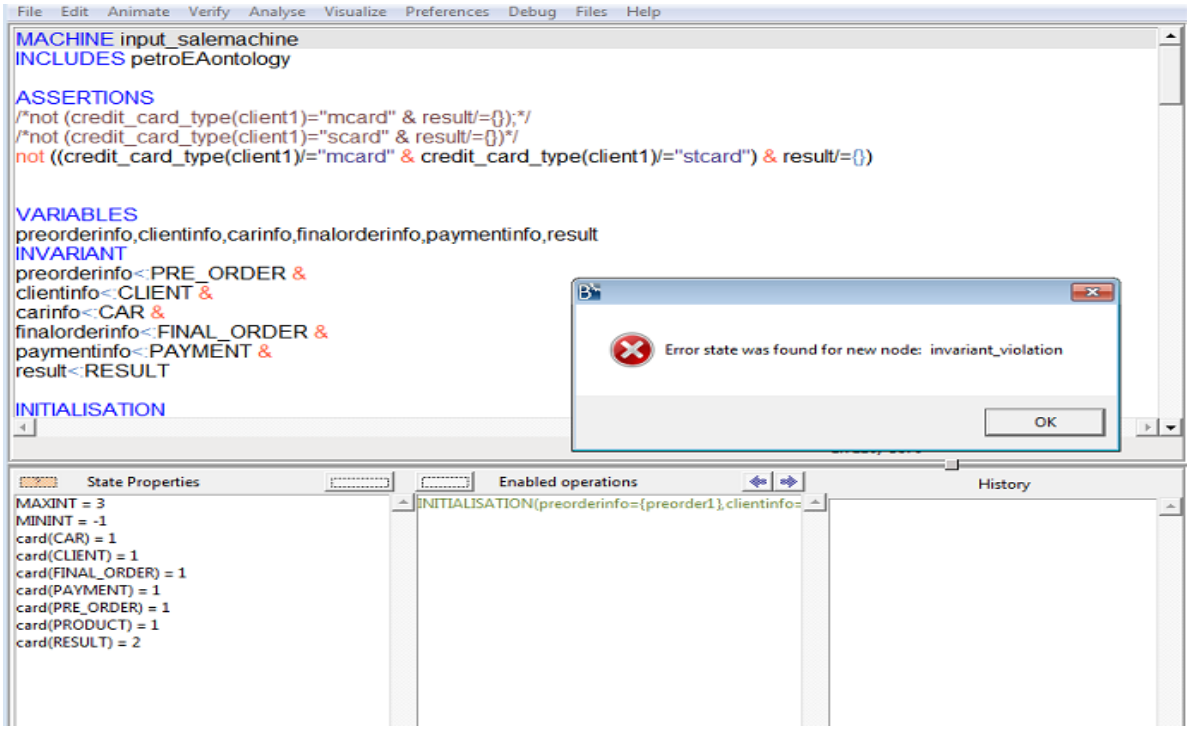

Fig. 8: The output of activating model checker.

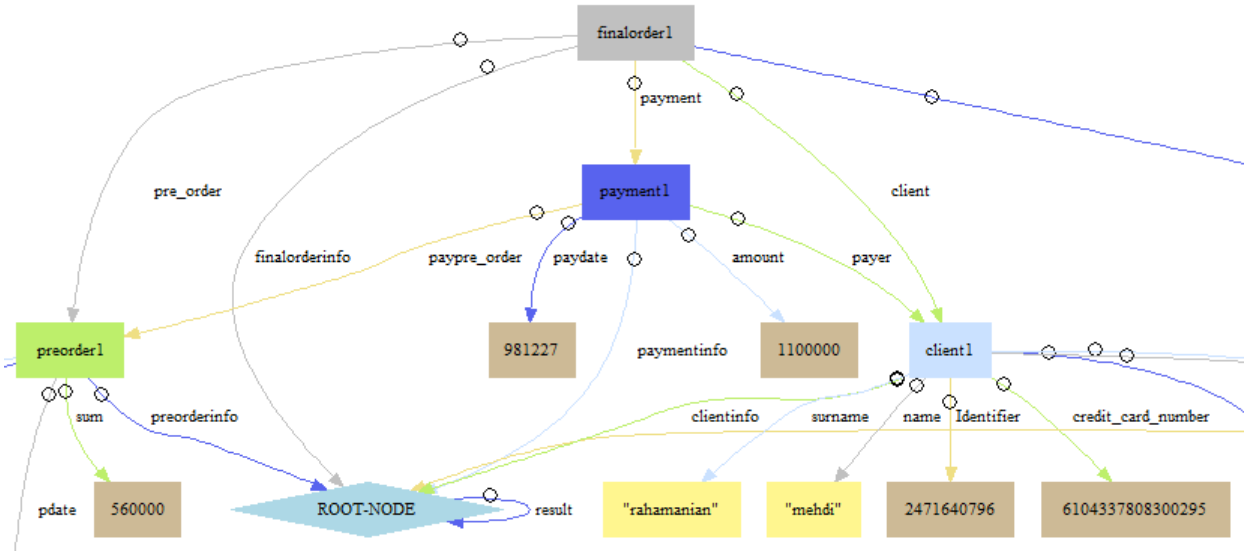

Fig. 9: A piece of output graph of the test case .

F. Evaluation of Implementation

The recent studies are categorized by two indicators:

1- Studies that have studied syntactic and semantic modeling in the field of enterprise architecture. None of which was intended to generate test cases.

2- Studies that have produced test cases without considering the discussion of enterprise architecture.

In comparison to recent studies, as shown in Table 7, the proposed method starts from the enterprise level, get benefited from the enriched descriptions yielded for enterprise business processes, move to generate proper syntactic and semantic models of the descriptions, and generate prioritized test cases from the well-established models to come up with test cases to be used for verification and/or validation. While in [1], [5]-[9], [11]-[13], syntactic and semantic modeling has been used for other objectives and a formal model has rarely been created.

These studies have in no way generated test cases based on enterprise architecture design. In [15], test cases have been generated, regardless of enterprise architecture design.

Table 7: Comparison of propose method by recent studies

Reference	Main contribution	Enterprise architecture design	Syntactic modeling	Semantic modeling	Formal modeling	Test case generation
[1]	Identify, classify, analyze, and evaluate existing methods for EA visualization.	yes	yes	yes	no	no
[5]	Providing a formal way for exploring misalignment of concepts.	yes	yes	no	yes	no
[6]	Proposing a combined approach that enterprise modeling would be suitable for both humans and machines.	yes	yes	no	no	no
[7]	Proposing a method for detection of any logical paradoxes in enterprise architecture models.	yes	yes	no	no	no
[8]	Using ontology to present, integrate and analysis enterprise models.	yes	no	yes	no	no
[9]	Proposing a combined modeling approach for convergence between business and technology.	yes	no	yes	no	no
[11]	Using descriptive logic along with ontology for manual analysis of enterprise design.	yes	no	yes	no	no
[12]	Modeling the dependencies between the business, information systems and IT infrastructures.	yes	no	yes	no	no
[13]	Develop a framework intended to make a balance between technology and business.	yes	yes	no	no	no
[14]	Presenting a model-based approach to automatically generate test cases from business process models	no	yes	no	no	yes
[15]	Developing a tool for the automated generation of test cases based on descriptions	no	no	no	yes	yes
[16]	Using the activity diagram for the automated generation of test cases	no	yes	no	no	yes
[17]	Creating a tool named PCTgen which automatically generated a set of test cases to check a workflow	no	yes	no	no	yes
Proposed method	Generating semantic test case from the enterprise level	yes	yes	yes	yes	yes

Results and Discussion

As mentioned earlier, no studies have been conducted to generate test cases based on descriptions received from enterprise architecture. However, the proposed method has the following advantages and limitations based on implementation of our method on the selected case study.

1- The details of the business process have been defined and started based on the enterprise level.
2- The level of abstraction is decreased by syntactic and semantic modeling of the enterprise architecture description.
3- In order to create a description that would be sampled, the semantic descriptions created using the proposed transition rules.
4- The generated test cases can be used in the validation and/or validation of business software, and because their descriptions are started at the enterprise level, they have high validity.

The limitations of the proposed method:

1- Due to the generality of TOGAF, it has been used for describing an enterprise architecture in this research. It is suggested that other enterprise architectures can also be used.
2- The focus of the proposed method was on enterprise business process. It is suggested that other business elements such as enterprise business services can be examined.
3- We transform semantic description into formal form by predefined rules, manually. It is suggested that, this can be done automatically by writing a parser.

Conclusions

The frameworks of enterprise architecture describe the elements of enterprise architecture at an abstract level and thus fail to elaborate on the details. However, they do provide architecture developers with a general primitive perspective. The main core in every enterprise's architecture consists its business processes. Enterprise processes are resulted by enterprise's goals and missions.

In order to test the verification and/or validation of a software product, it must be evaluated against the expressed descriptions and business rules. Previously, several studies following different objectives have tried to model enterprise architectures.

However, no previously conducted study has elaborated on modeling following the objective of creating test cases for further verification and/or validation purpose.

Therefore, the subject of testing for business software practically starts from the enterprise level (goals, missions, etc.). Therefore, the main contribution of this study is generating a set of test cases based on the descriptions yielded from enterprise business processes in early steps; then, the amount of later reviews and changes can be significantly lessened.

The overall framework of the proposed method is an iterative cycle adopted from the TOGAF Architecture Development Method. Following this cycle, once the primary descriptions of goals, missions, and strategies of the enterprise are retrieved, we will syntactically model the architecture.

Afterwards, the business processes will be modeled semantically, and the description will be transformed into formal B language for the purpose of sampling from the syntactic-semantic modeling.

In order to evaluate the proposed method, it has been implemented on the sales and marketing department of a petrochemical corporation, and the yielded results confirmed the validity of the proposed method. Based on the proposed method, the following values have been created:

1- By adding semantics to the syntactic models of enterprise architecture, more precise and exhaustive descriptions of the processes have been yielded, and in fact, the degree of abstraction has been decreased.
2- Creating a formal model of the syntactic and semantic descriptions can be subjected to sampling. And the resulting samples will be covering both syntax and semantics.
3- The proposed method starts from the missions and strategic goals of enterprises; therefore, the output samples are efficiently precise and complete.

One of the most applied fields in the domain of software engineering is automatic code generation. For future work, it is suggested to use the proposed method for automatic generation of codes according to the descriptions retrieved relating to enterprise architecture, and according to the TOGAF framework, which is a general framework.

Semantic descriptions are suitable tools for providing precise and yet understandable descriptions for machines.

The focus of the present study was centered on business processes in the architecture layer of enterprises. For future work, it is suggested to elaborate on providing semantic descriptions for other layers of architecture as well.

In the present study, the authors have used the TOGAF standard due to its publicity and high applicability; however, it is suggested to use also other standards for analyzing enterprise architecture and providing high-level descriptions.

Author Contributions

M. Rahmanian designed and implemented the proposed method. R. Nassiri collected data. M. Mohsenzade interpreted the results and carried out the data analysis. R. Ravanmehr wrote the manuscript and carried out the data analysis.

Acknowledgment

The authors received no funding from any organization in the course of carrying out the current study. It is certified that the Islamic Azad University is a private research and academic institute.

Conflict of Interest

No potential conflict of interest regarding the publication of this work. Besides, the authors have been completely witnessed the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy.

Abbreviations

TOGAF	The Open Group Architecture Framework
ADM	Architecture Development Method
WSMO	Web Service Modeling Ontology
WSML	Web Service Modeling Language
OWL	Ontology Word Language
EA	Enterprise Architecture
SEAM	Semantic Enterprise Architecture Modeling
SBVR	Semantic of Business Vocabulary and Rule
IS	Information System
BPaaS	Business Process as a Service
OWL	Ontology World Language
SEAM	Semantic Enterprise Architecture Modeling
MC/DC	Modified Condition Decision Coverage
BSC	Balance Score Cards

References

[1] Z. Zhou, Q. Zhi, S. Morisaki, S. Yamamoto, "A systematic literature review on enterprise architecture visualization methodologies," IEEE Access, 8(1): 96404-96427, 2020.

[2] "The TOGAF® Standard, Version 9.2." accessed 1 October 2021.

[3] S. Sharma, L. Raja, D. Pallavi Bhatt, "Role of ontology in software testing," J. Inf. Optim. Sci., 41(2): 641-649, 2020.

[4] A. Mili, F. Tcheir, Software Testing Concepts and Operations, John Wiley & Sons, 2015.

[5] K. Bouafia, B. Molnár, "Analysis approach for enterprise information systems architecture based on hypergraph to aligned business process requirements," Procedia Comput. Sci., 164: 19-24, 2019.

[6] K. Hinkelmann, E. Laurenzi, A. Martin, B. Thönssen, Ontology-Based Metamodeling" Business Information Systems and Technology 4.0. Studies in Systems, Decision and Control, 141: 177-194, 2018.

[7] E. Babkin, A. Ponomarev, "Analysis of the consistency of enterprise architecture models using formal verification methods," Bus. Inf., 3 (41): 30–40, 2017.

[8] A. Caetano, G. Antunes, J. Pombinho, M. Bakhshandeh, J. Granjo, "Representation and analysis of enterprise models with semantic techniques: an application to ArchiMate, e3value and business model canvas," Knowledge Inf. Syst., 50: 315–346, 2016.

[9] K. Hinkelmann, E. Laurenzi, B. Lammel, S. Kurjakovic, "A semantically-enhanced modelling environment for business process as a service," in Proc. 4th International Conference on Enterprise Systems, 4: 143-152, 2016.

[10] K. Hinkelmann, E. Laurenzi, A. Martin, D. Montecchiari, M. Spahic, B. Thönssen, "ArchiMEO: A standardized enterprise ontology based on the ArchiMate conceptual model," in Proc. the 8th International Conference on Model-Driven Engineering and Software Development – MODELSWARD: 417-424, 2020.

[11] G. Antunes, M. Bakhshandeh, R. Mayer, J. Borbinha, A. Caetano, "Using ontologies for enterprise architecture analysis," in Proc. 17th IEEE International Enterprise Distributed Object Computing Conference Workshops, 17: 361-368, 2013.

[12] W. Chen, C. Hess, M. Langermeier, "Semantic enterprise architecture management," in Proc. the 15th International Conference on Enterprise Information Systems (ICEIS-2013): 318-325, 2013.

[13] K. Hinkelmann, D. Karagiannis, B. Thoenssen, R. Woitsch, A. Gerber, "A new paradigm for continuous alignment of business and IT: combining enterprise architecture," Model. Enterpr. Ontol., 79: 77-86, 2015.

[14] A. Yazdani Seqerloo, M.J. Amiri, S. Parsa, "Automatic test cases generation from business process models," Requirements Eng. 24: 119-132, 2019.

[15] W. Zhang, S. Liu, "Supporting tool for automatic specification-based test case generation," in Proc. International Workshop on Structured Object-Oriented Formal Language and Method, 7787: 12-25, 2013.

[16] A.K. Jena, S.K. Swain, D.P. Mohapatra, "A novel approach for test case generation from UML activity diagram," in Proc. International Conference on Issues and Challenges in Intelligent Computing Techniques: 621-629, 2014.

[17] "The ArchiMate® Standards" accessed 1 October 2021.

[18] G. Wierda, Mastering ArchiMate Edition 3.1. : R & A, 2021.

[19] P. Desfray, G. Raymond, Modeling Enterprise Architecture With TOGAF. : Elsevier, 2014.

[20] "Archi-Open Source ArchiMate Modelling" accessed 1 Octobr 2021.

[21] "Web Service Modeling Ontology" accessed 1 October 2021.

[22] D. Fensel, H. Lausen, J. Bruijn, Enabling Semantic Web Services: The Web Service Modeling Ontology. : Springer-Verlag Berlin, 2007.

[23] D. Fensel, F.M. Facca, E. Simperl, I. Toma Web Service Modeling Ontology. In: Semantic Web Services. Springer, Berlin, Heidelberg, 2011.

[24] M. Bures, T. Cerny, M. Klima, "Prioritized process test: More efficiency in testing of business processes and workflows," in Proc. International Conference on Information Science and Applications, 424: 585-593, 2017.

[25] J. Bruijn, H. Lausen, A. Polleres, D. Fensel, "The web service modeling language WSML: An overview," in Proc. European Semantic Web Conference: 590-604, 2006.

[26] J. Bruijn, D. Fensel, U. Keller, "Using the web service modeling ontology to enable semantic e-Business," Commun. ACM, 8(12): 43-47, 2005.

[27] F. Christina, P. Axel, D. Roman, D. John, " Towards intelligent web services: the web service modeling ontology (WSMO)," in Proc. International Conference on Intelligent Computing (ICIC'05): 23-26, 2005.

[28] "The Programming Language B" accessed 1 October 2021.

[29] K. Lano, The B Language and Method: A Guide to practical Formal development. : Springer Verlog, 1996.

[30] M. Leuschel, M. Butler, "ProB: a model checker for B," in Proc. International Symposium of Formal Methods Europe Springer, Berlin-Hei- delberg: 855–874, 2003.

Biographies

Mehdi Rahmanian received his B.S. degree in software Engineering from Shahid Chamran University, Ahwaz, Iran in 2007. He graduated in M.Sc. degree in software Engineering from IAU University, Ahwaz, Iran in 2011. Currently he is a Ph.D. student in IAU University, Tehran, Iran. His interests include software engineering, Enterprise Architecture and Software testing.

- Email: mehdi.rahmanian@srbiau.ac.ir
- ORCID: 0000-0002-3575-5230
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: NA

Ramin Nassiri received his B.S. in Computer software engineering from Tehran University, Tehran, in 1989, the M.S. in Computer software engineering, in 1995 and the Ph.D. degree in Computer software engineering from IAU University, Tehran, in 2003. Currently, he is a faculty in the Department of Computer engineering at the IAU University. He is the author/coauthor of more than 100 publications in professional and/or academic journals and conferences. He is co-founder of three IT companies in Iran and UAE since 2000. Also he has been managing a few national IT projects since past decade to promote public welfare by deploying ICT technologies and enablers. His research focus is basically on Software engineering, Enterprise architecture, Big data, Software testing, IoT and Etc.

- Email: r_nasiri@iauctb.ac.ir
- ORCID: 0000-0002-9488-9044
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: NA

Mehran Mohsenzadeh received his B.E degree (Software Engineering) in 1997 from Shahid Beheshti University and M.E (in 1999) and Ph.D. (Software Engineering) in 2004 from IAU University, Tehran. His major interests are Cloud Computing, Software Engineering and Big Data and has published more than 85 papers (author/co-author) in International Conferences and journals. He is Assistant Professor in the Department of Computer Engineering, Science and Research Branch, IAU University of Iran.

- Email: r.mohsenzadeh@srbiau.ac.ir
- ORCID: 0000-0001-6835-409x
- Web of Science Researcher ID: NA
- Scopus Author ID: 26435355100
- Homepage: NA

Reza Ravanmehr graduated in computer engineering from Shahid Beheshti University, Tehran, in 1996. After that, he gained his M.Sc. and Ph.D. degrees, both in computer engineering, from Islamic Azad University, Science and Research Branch, Tehran, in 1999 and 2004, respectively. His main research interests are distributed/parallel systems, large-scale data management systems, and social network analysis. He has been a faculty member of the Computer Engineering Department at Central Tehran Branch, Islamic Azad University, since 2001.

- Email: r.ravanmehr@iauctb.ac.ir
- ORCID: 0000-0001-9605-5839
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: NA

How to cite this paper:
M. Rahmanian, R. Nassiri, M. Mohsenzadeh, R. Ravanmehr, "Semantic enterprise architecture oriented test case generation for business process," J. Electr. Comput. Eng. Innovations, 10(2): 311-328, 2022.

DOI: 10.22061/JECEI.2021.8218.496

URL: https://jecei.sru.ac.ir/article_1636.html