Decision Science Letters 5 (2016) 109–118

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

Service oriented architecture assessment based on software components

Mahnaz Amirpour[a*], Ali Harounabadi[b] and Seyyed Javad Mirabedini[b]

[a]*Department of Computer Engineering, Boushehr Branch, Islamic Azad University, Boushehr, Iran*
[b]*Faculty Member of Computer and Science, Central Branch, Islamic Azad University, Tehran, Iran*

C H R O N I C L E

Article history:
Received June 25, 2015
Received in revised format:
June 28, 2015
Accepted July 27, 2015
Available online
July 29 2015

Keywords:
Enterprise Architecture
Architecture Process
Colored Petri Nets
Top Down Approach
Bottom Up Approach
Service Taxonomy

A B S T R A C T

Enterprise architecture, with detailed descriptions of the functions of information technology in the organization, tries to reduce the complexity of technology applications resulting in tools with greater efficiency in achieving the objectives of the organization. Enterprise architecture consists of a set of models describing this technology in different components performance as well as various aspects of the applications in any organization. Therefore, information technology development and maintenance management can perform well within organizations. This study aims to suggest a method to identify different types of services in service-oriented architecture analysis step that applies some previous approaches in an integrated form and, based on the principles of software engineering, to provide a simpler and more transparent approach through the expression of analysis details. Advantages and disadvantages of proposals should be evaluated before the implementation and costs allocation. Evaluation methods can better identify strengths and weaknesses of the current situation apart from selecting appropriate model out of several suggestions, and clarify this technology development solution for organizations in the future. We will be able to simulate data and processes flow within the organization by converting the output of the model to colored Petri nets and evaluate and test it by examining various inputs to enterprise architecture before implemented in terms of reliability and response time. A model of application has been studied for the proposed model and the results can describe and design architecture for data.

© 2016 Growing Science Ltd. All rights reserved.

1. Introduction

Recently, many have argued that the traditional approaches are not appropriate for handling current needs and we need to develop and apply the effective approaches to overcome the complexities in shorter periods of times. A full Service-oriented architecture (SOA) implementation is not only in connection with the development and deployment services but also it reflects the feasibility of using services to integrate distinct applications and to create composite applications. Advantages gained in order to reduce development time for creating product, utilizing flexible applications with fast response and dynamic connectivity with partners depend on successful SOA implementation. Today's

* Corresponding author. Tel:+989177710527
E-mail address: danyalmir@yahoo.com (M. Amirpour)

© 2016 Growing Science Ltd. All rights reserved.
doi: 10.5267/j.dsl.2015.7.005

organizations are complex and integrated systems consist of processes, organizational units, people, information and technology support, as well as their dependencies and relationships among various elements. Understanding, engineering and management of these social, technical and infrastructures are crucial to achieve and maintain the performance of organizations. A service-oriented architecture is an environment in which a process using services to run is standardized and obtained based on service-orientation principles. Any software model has to be designed, which can be used by other systems, i.e. other services being able to use a particular service and benefit from it if necessary. Architecture executable models are extracted from the operating system components, tasks, messages, events, timing and mechanism of transfer of message. These models give the architecture the capabilities to examine the interaction of components, how to allocate tasks to the components, tasks control and system temporal and spatial specification. Reliability is one of the most important non-functional requirements which influence on software development. Several works have been accomplished on non-functional requirements, particularly reliability in the initial level of software development and software architecture. But, the use of any of these methods is limited in real and complex application examples and none of the methods completely evaluates the requirements.

2. SOA concept

SOA is a collection of rules, policies and frameworks enabling a software to provide good performance through a series of separate and independent software services and related to each other for other applications to discover and call the services only by standard and default interfaces without knowing how to implement (Mozaffari et al., 2011). Now, the web service technologies and the best and successful practice implementations show that SOA can be proposed as a practical and possible solution in design and development of the new systems and accretion of current great systems. All functions and transactions in SOA are defined as the services (Javanbakht et al., 2008).

3. SOA Delivery Lifecycle Phases

The lifecycle of SOA delivery project is simply comprised of a series of steps, which needs to be completed to construct the services for a given service-oriented solution (Shin et al., 2010). First operation domain and business process of organization ought to be identified and the services must be modeled. In Service-Oriented Design step, the working plan and service layers and required interfaces must be prepared and designed. Then in Service Development phase, the identified services are taken to develop with programming languages and working places. Next, we have the Service Testing so that the quality of services are examined practically before entering into real operational place. Finally, in Service Deployment step, the produced parts, the defined interfaces are loaded and plugged as distributed. Finally, in the phase of Service Administration, items like monitoring the services, message management and the ways of facing the crises etc. must be considered.

3.1 Service Oriented Analysis Strategies

It is in this initial stage that we determine the potential scope of the proposed SOA of this paper. Service layers are mapped out, and individual services are modeled as service candidates that comprise a preliminary SOA. Different strategies exist on how to organize lifecycle stages to enable delivery of specialized service layers and we explain some of them next.

3.1.1 Top-Down

In Top Down strategy first we should understand and capture the broad functional domains and identify the detail business processes and sub-processes and their handshake points. Then we need to identify the high-level service candidates based on process activities. Top Down strategy allows architects to understand the business context and provide defining and scoping the SOA project boundaries and domains (Ma et al., 2011). It bridges the gap between information technology (IT) and business teams

and treat preliminary list of activities as potential list of services for further analysis and activity/service dependency identifications at a high level.

3.1.2 Bottom-Up

The Bottom-Up approach involves taking the enterprise level application portfolio analysis, assessment for reuse, redundancy, and rationalization effort. In general, the targets for Bottom-Up approach typically will be redundant business logic, multiple copies of business data entities, or implementing the same business logic with multiple products, resulting in huge licensing, operational and maintenance costs, etc. Services identified using this approach may or may not be composed to build the business processes. The contribution of these services to ROI under the reuse line item could be higher depending on the scenario. These services will become part of the basic services, such as the infrastructure services or technical services in SOA service layers.

4. Suggested Approach

The suggested approach in this article offers an easy method for identifying different services in the SOA analysis step, which has the benefits of some of the previous methods without any fiscal and operational overloads of the previous techniques. To make the work easier, we can categorize different services into logical groups showing the operational state of service. The logic of organizational establishments can be divided into two primary regions: Application Logic and Business Logic. Different services in service-oriented architecture can be used to represent either of two logics (or both) (Mozaffari et al., 2011). Therefore, we may consider the collection of service taxonomy for them based on their operational states (Shin et al., 2010). The most primary and important service taxonomy used for different services can be as follow:

- Application services: A generic category used to represent services that contain logic derived from a solution or technology platform. Application services are mostly general and re-useable.
- Business services: A generic category used to represent services that contain business logic. However, individually these services are classified as follow:
- Task Centric Services: A business process-specific variation of the business service that represents an atomic unit of process logic. They are usually low in re-usability.
  - Entity Centric Services: including the services showing one or more business dependent entities and operation of related candidate to these entities.
- Process Services: A service that represents a business process as implemented by an orchestration platform and described by a process definition. Process services reside in the orchestration service layer.

After categorizing the services into the reasonable types to reach the principle of Loosely Coupling (of the main principles of service-orientation), it is necessary for services to be separated physically so that no organizational domains depends on each other. It allows the automatic show of business logic to be developed independent of application logic. The most important service layers are included (Lindsay et al., 2003):

- ✓ Orchestration Service Layer;
- ✓ Business Service Layer;
- ✓ Application Service Layer.

Application Service Layer (lowest) uses mostly the application services to prepare the functions of re-usability related to the process data of old or new application centric programs domains and platforms. Business service layer, using business service, can provide the services showing the business logic of organization. Orchestration service layer includes the collection of one or more

process service combines the application centric and business services for map business process to related services based on the regulations and logic of business and process details, and ensures that the relations are made among different services of special discipline (Rezaei & Shams, 2009). The suggested approach uses Top-Down and Bottom-Up methods in combination to identify and model the mentioned services to benefit the previous methods without having any additional complexities. To identify the application centric services and entity centric services, we can use the of Bottom-Up approach. Moreover, to identify the business services and task centric services, we can use the Top-Down approach. Suggested method, represented summarily in Fig. 1 as follows,

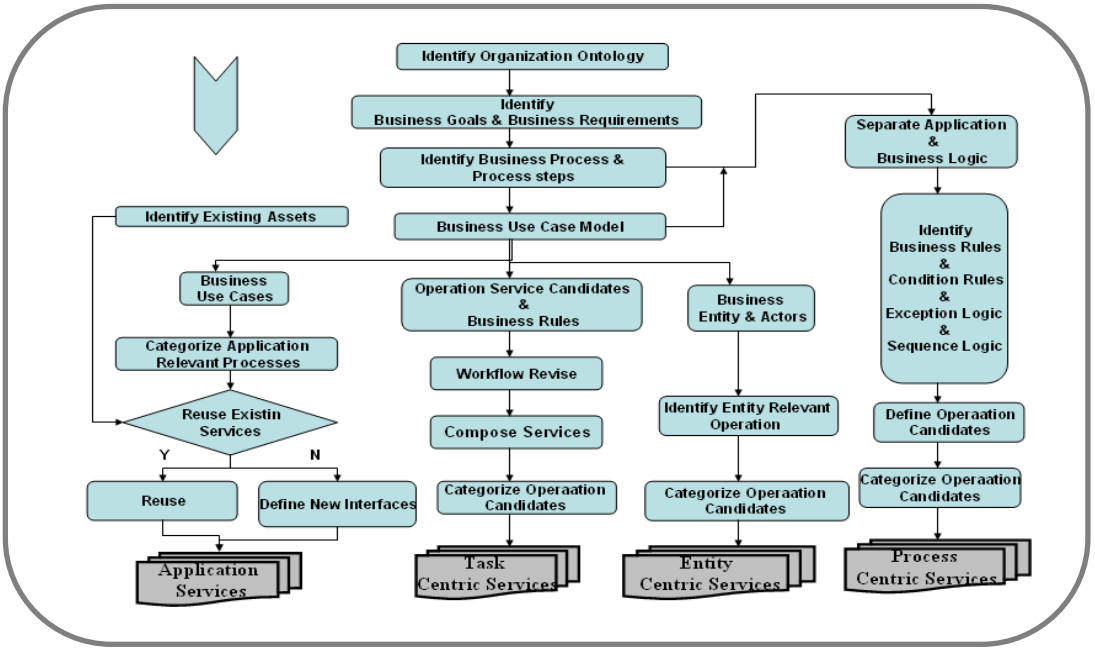

Fig. 1. Suggested method summary

Suggested method, includes the following steps:

- Identify the Business process

In the first step, it is necessary to determine the functional domain of business process and business goals, understand the business requirements, reach the total vision, goals and task of future organization. Furthermore, in order to be developed coincidentally by two Top-Down and Bottom-Up, it is necessary to recognize coincidentally the services and existing assets which can be re-useable (van der Merwe et al., 2013). The main outputs of this step:

1. Business requirements;
2. Business vision;
3. Business process.

- Make Business Use Case model

In this step, it is necessary to recognize and assign detail process steps, business actors, business use cases, business entities, business use case, and operation service candidates & business rules. The main outputs in this step:

1. Business Actors & Entity- Business Use Case;
2. Business Use Case Model;
3. Operation Service Candidate & Business Rules.

- Identify Entity Centric Services

In this step, using last steps outputs, the candidate operations related to one or more dependent entities are identified and the operation candidates are scaled and categorized as the logical groups that call Entity Centric Services.

Output: Entity Centric Services

- Identify Application Services

The step separates first the process requirements, which depends on the technology and application programs. Then it checks, if available, which one of operation candidates can be performed through the existing services of current organization so that it could reuse the existing assets based on the principles of service-oriented, or could change them into required services with little details. Then, it categorizes the remaining operation candidates as the logical groups each of which shows the logic of special service.

Output: Application Services

- Identify Task Centric Services

Inputs:

1. Business Requirements;
2. Business Use Case Model;
3. Operation Service Candidate & Business Rules.

First, the logic of workflow and imaginable relations and reactions are reviewed and, if required, are amended, then for supporting the long-term goals, if necessary, the new business goals are defined and added to system, or if it needs to combine the operation candidates of business services and application services, the related activities are performed and then the operation candidates related to business duties are categorized as the logical groups.

Output: Task Centric Services.

- Identify Process Centric Services

Inputs:

1. Business Process Steps;
2. Business Use Case Model;
3. Operation Service Candidate & Business Rules.

Since the process services consider as a controller services, it is necessary that first separate the business logic and application logic. Also it is necessary to recognize and separate all business rules, condition rules, exception logic and sequence logic, then assign the operation candidates of all controlling, conditions and exceptions operations. Finally, it is necessary to categorize the related operation candidates as the logical groups, each of which shows the logic of special service (Akkasi & Shams et al., 2008).

Output: Process Centric Services.

4.1. Sequence diagram role in reliability assessment

Enterprise architecture describes system at high levels of abstraction by identifying the structural and behavioral aspects. The interaction between components in order to perform tasks is described by a sequence diagram or a collaboration diagram during the software development process. Collaboration

diagram, similar to sequence diagram, shows the messages between components based on the time they occur.The diagram shows functionality of a user while the emphasis is on the interaction between the cases and is plotted according to messages sending time. Special elements and structures are used to model the system load in the diagram in terms of reliability.

4.2. UML sequence diagrams conversion to colored Petri net algorithm

This diagram shows the messages between components sequence based on the time of occurrence. The conversion focuses on the messages and how they transferred, as well as the sender object and the recipient that is why the implementation of this method is simpler and easier to understand. At this stage the aim is to convert sender and recipient components into Petri net. Collaboration diagram in reliability assessment shows the way that a client of a particular type move among service centers. A specific color from colored beads on Petri net can be considered for each set of scenarios that is attributed to a client. Next, each component of the sender and recipient of the message and the messages between them is converted into Petri net. In this case, each message recipient and send component is converted into location - transition- location (Mirshekari et al., 2015).

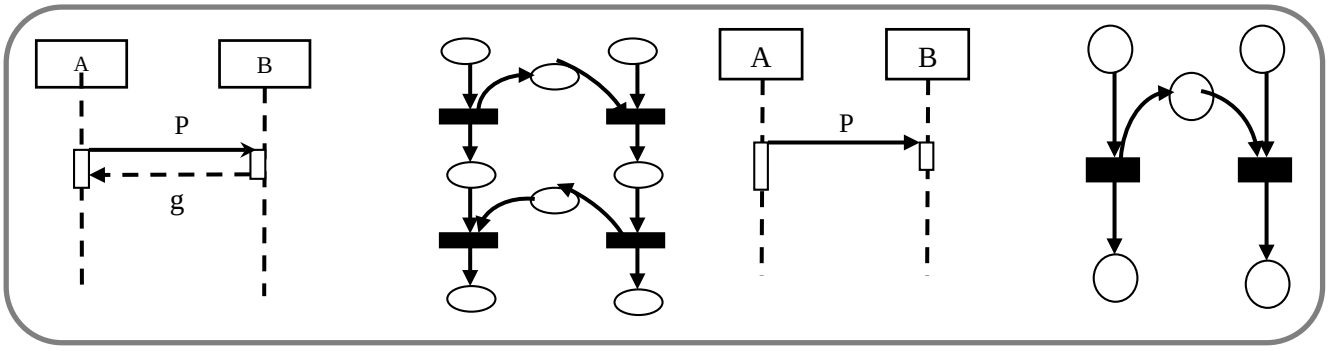

Fig. 2. Asynchronous and synchronous s and real-time message and Petri net equivalent

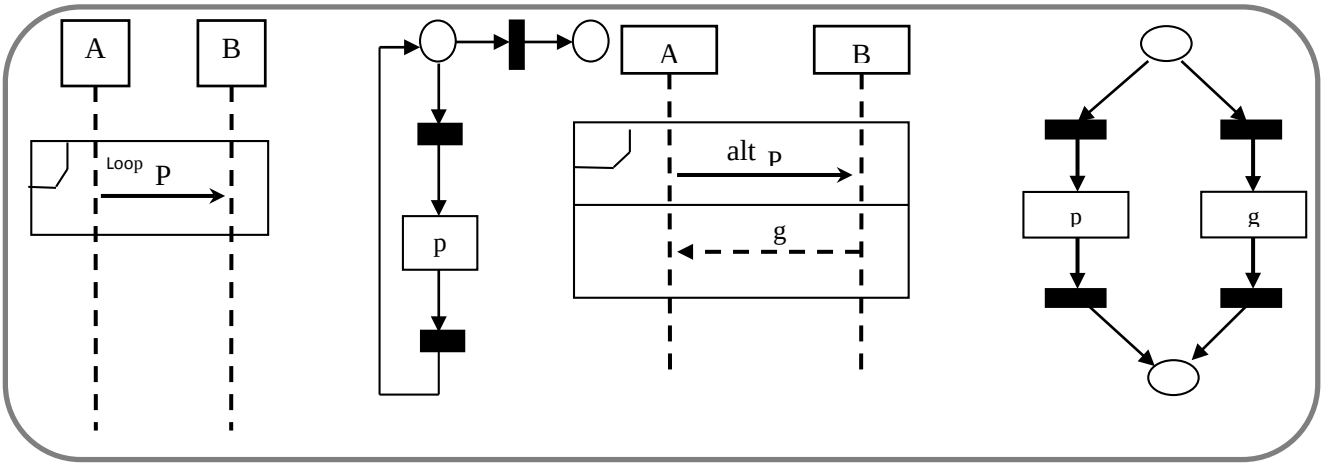

Fig. 4. Selection and iteration structure and its equivalent Colored Petri net

4.3 Annotations associated with reliability

This paper describes a set of indexes to evaluate the reliability. The indexes are added to Unified Modeling Language (UML) diagrams describing the architecture in the form of formats and labels. Indexes provide quantitative information on UML model and quantitative analysis will be possible through this information. Sequence diagram uses special elements and structures to model the system load from the standpoint of reliability. In the first case it is assumed that the time taken to send messages is not very important. A condition has been attributed to each message in diagram, which expresses the possibility of message sent. Also, a number of messages with attributed condition can be sent from a point and run in parallel. Messages located in an iteration structure can be sent several times; this

diagram is used to assess performance, workload and delay of each message and each component periods in the busy condition and failure possibility of any scenario are used to evaluate the reliability.

4.4 Metric calculation of reliability

This paper presents a method for assessing the reliability of the architecture where the reliability data is added to the UML diagrams as formats and labels; this is based on the use of three types of data including the user profile, probability of failure unique to each one of the components and connectors, and their productivity. REcomponent format and Recomfailprob and REpb labels are annotated, also the interaction between the components and REconnector format and REconnfailprob labels are annotated.The reliability prediction algorithm produces beta distribution of component failure rates and requires entering a confidence interval for the failure rate of each component by the user. This article assumes that component failure is independent of other components and will not be transferred to other components. Average probability of failure is calculated as follows according to Eq. (1), after the failure of each component and between two components was calculated per each transition:

$$\Theta ij = \text{prob}(\text{failure of Cij}) = 1 - (1 - \Theta i)^{\text{bpij}}. \quad (1)$$

To calculate the probability of failure between the two components *m* and *1*, if *nms* (*1*, *m*, j) of the interactions of these two components is in Figure j it is expressed with linteract (*1*, *m*, *j*), so reliability ψ_{mj} is calculated using the probability of failure between the two components (ψ_i),

$$\psi_{1mj} = (1 - \psi i)^{\text{linteract}(1,m,j)}. \quad (2)$$

After the failure of any component and between two components in any transition is calculated, to calculate the average probability of failure in each transition the following equation is used that calculates the reliability of the system:

$$\Theta_s = 1 - \sum_{j=1}^{k} pj \left[\prod_{i=1}^{n} [\![(1]\!] - \Theta_i)^{\text{bpij}} \coprod_{(j,j)} (1 - \psi l.1,j)^{\text{linteract}(l,i,j)} \right]. \quad (3)$$

Finally, all transitions are calculated with a failure probability and service time, to calculate the reliability of the entire system, a total failure and service time are calculated in all transitions.

4.5. Calculate availability metrics

Availability is calculated as part of the system that is running and shows the proportion of time that the system is running. Availability is a fraction of time that the system is alive and runs and it means that the system can solve the problems and return to its natural state. System proper running possibility limits at time *t* when *t* is infinity it specifies the size of the attribute but the attribute in the system steady-state is an amount of time that the system is functioning properly and is defined in Eq. (4). MTTF is mean time error and MTTR is the mean time correction. Availability is associated with the reliability; the higher the reliability of the system is, the more chance the system is available and vice versa.

$$\alpha = \frac{Mean\ Time\ to\ Failure}{Mean\ Time\ to\ Failure + Mean\ Time\ to\ Repair} = \frac{MTTF}{MTTF + MTTR} \quad (4)$$

4.6 Applicable models creation

Colored Petri nets and CPN Tools have been used to create applicable models in this paper. In fact, an applicable model of architecture is the formal description of the architecture through which final behavior can be analyzed prior to architecture implemented to be informed of the problems and

inefficiencies and to better implement architecture and prevent additional costs, and even failure. Moreover, the assessment of each metrics of efficacy, such as service time, and the failure rate of each component and between the components are calculated and checked.

5. Case Study

This practical example has defined the goal of identifying services according to all laws and adding hypothetical situations to illustrate various aspects of the method. We assume that the three factors in the model are reliable, and we look forward to figuring out the correct consistency process. Bank, the sender and the carrier are reliable. Bank sends a letter of credit recipient that reflects the competence and credibility of the recipient. In general, LoC is shown as service provided by the bank for the recipient and also sender to ensure payment to the recipient for sent goods. The service is for trust recipient, not the sender, because the recipient is willing to trust the sender to delivery. Deliverer creates a Bill of Lading (BoL) and delivers it to sender in return to the good that should be delivered. Anyone who carries BoL can claim to have the ownership of the goods. Sender gives BoL to the bank and receives product transfer fee. Bank receives fee from the Bol recipient, so the recipient can claim ownership of the goods. The deliverer delivers the goods receiving BoL. Scaling and categorizing the operation candidate related to the working entities and actives from Business process, the services candidates of entity centric services are identified as follows: Since the entities of consignee and carrier are related and dependent to each other, we can describe the operation candidates related to two entities in a form of one service, in order to prepare the more flexibility and efficiency.

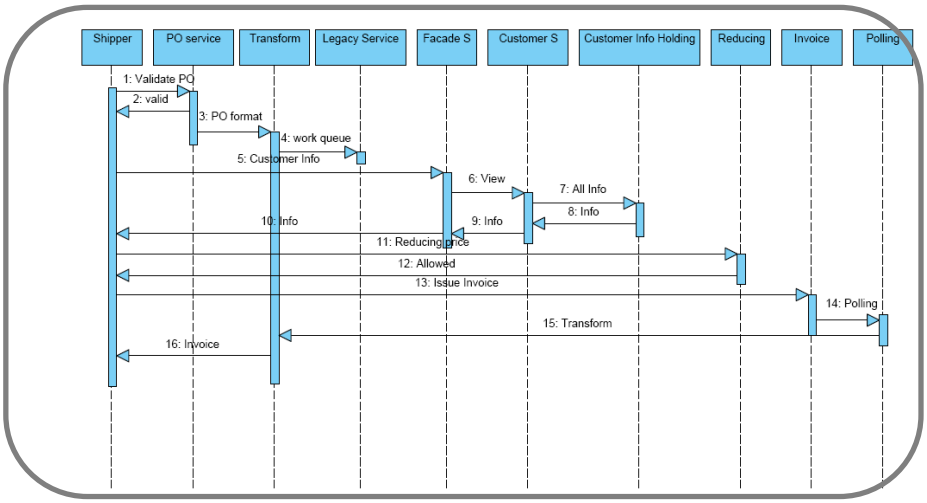

Fig. 6. Operation candidate for services

Separating the operation candidate related to the application working area and application requests, we understand that we can categorize the operation related to the notification to the user and related management, if disconfirming the form in the group of services of Application Services: In the next step, it is necessary to recognize the services dependent to the duty related to performing business tasks, using process steps and combining the business service candidates (See Fig. 7): 1- Validate consignee 2- Confirm authorization

Receiving two entities of working unit invoice, forming of timesheet and consignee and combining the related services and comparing the recorded rates, the operation candidate related to the business task of form confirmation can arrange the operations of form confirmation in terms of service dependent to the task of verify timesheet (See Fig. 8).

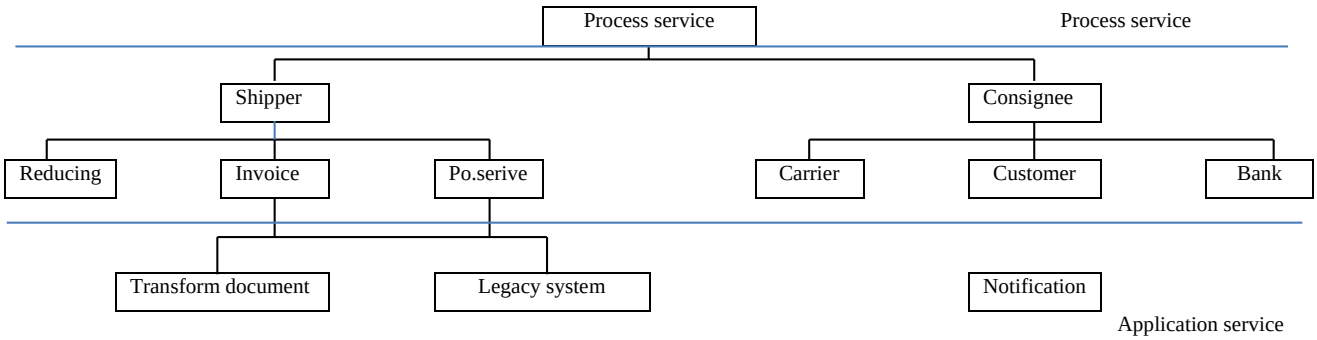

Fig. 7. Service Layers and Identified Services

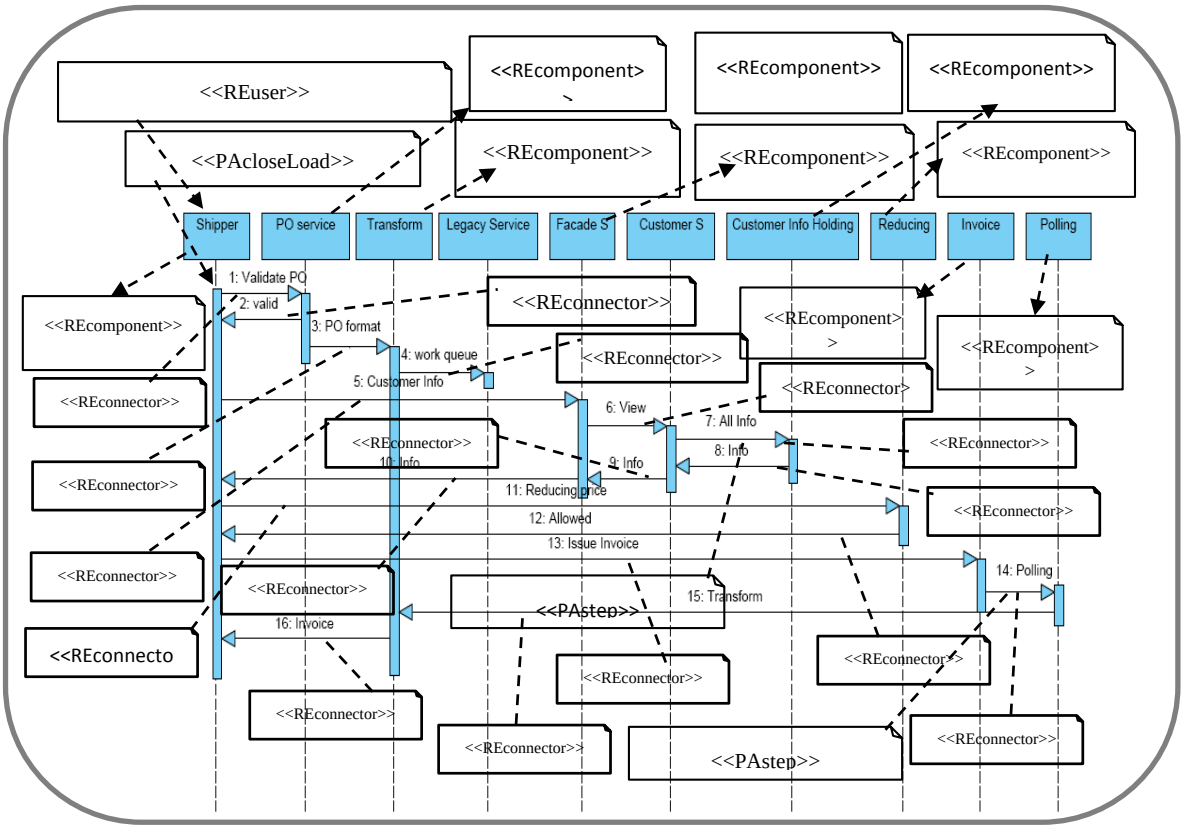

Fig. 8. Sequence diagram of shipper

Finally, we can categorize the related controlling operations in the form of process services through comparing the timesheet and presumed rates and accessing data, which are the components of condition and control operations of working process (See Fig. 9).

6. Conclusion

SOA and service-orientation are implementation-agnostic paradigms that can be realized with any suitable technology platform. Therefore, it seems SOA, as one of the latest approaches in software development, can be the best architecture of IT and communication industry. In spite of the brilliant points, it is not completely elevated and it is necessary that the procedure of future researches and studies be in the field of improving the efficiency and enlightening the techniques and steps of implementing SOA and decreasing in the current complexities and additives. Table 1 summarizes the comparison of the proposed method with other existing ones.

Table 1

Comparison of the suggested models with studied models

Evaluation criteria	Archimate model	Levis model	OSAN model	Suggested model
Reliability	Yes	No	No	Yes
Behavior accuracy	No	Yes	No	Yes
Evaluation with uncertainty	No	No	Yes	Yes
Response time	No	Yes	Yes	Yes
Object orientation	No	Yes	Yes	Yes

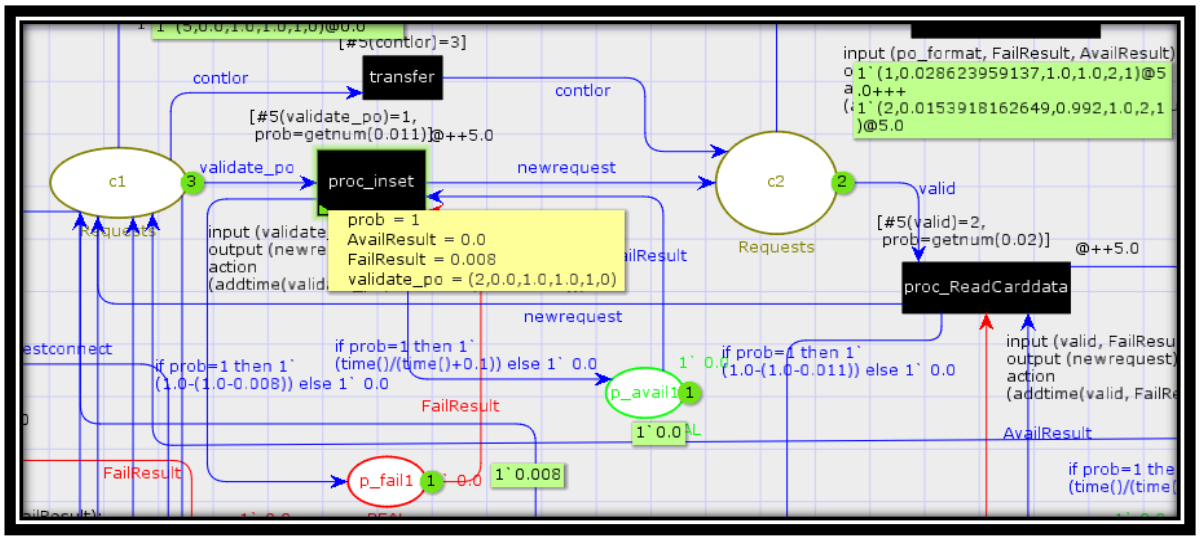

Fig. 9. Modeling in CPN Tools

References

Akkasi, A., Shams, F., & Seyyedi, M. A. (2008, April). Presenting A Method for Benchmarking Application in the Enterprise Architecture Planning Process Based on Federal Enterprise Architecture Framework. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on* (pp. 1-6). IEEE.

Javadpour, R., & Shams, F. (2009, May). Performance evaluation of electronic city architecture using colored Petri nets. In *The 2nd Conference on Electronic City, Tehran*.

Javanbakht, M., Rezaie, R., Shams, F., & Seyyedi, M. (2008, April). A new method for decision making and planning in enterprises. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on* (pp. 1-5). IEEE.

Lindsay, A., Downs, D., & Lunn, K. (2003). Business processes—attempts to find a definition. *Information and software technology*, *45*(15), 1015-1019.

Mozaffari, M., Harounabadi, A., & Mirabedini, S.J. (2011). A method for validating a behavior of enterprise architecture. *World Applied Sciences*, *14*(6), 831-841.

Ma, Z. M., Zhang, F., & Yan, L. (2011). Fuzzy information modeling in UML class diagram and relational database models. *Applied Soft Computing*, *11*(6), 4236-4245.

Mirshekari, A., Abadi, A. H., & Abedini, S. J. M. (2015). Providing a model by open framework and evaluation by colored Petri Nets. *Walia Journal*, *31*(s3), 253-259.

van der Merwe, A., Gerber, A., & van der Linde, J. (2013, November). The Impact of managerial Enterprise Architecture decisions on software development employees. In *Enterprise Systems Conference (ES), 2013* (pp. 1-7). IEEE.

Rezaei, R., & Shams, F. (2009). Providing a comprehensive method for developing and evaluating enterprise architecture plan. In *The first Conference on Enterprise Architecture in Practice*.

Shin, M.E., Levis, A.H., & Wagenhals, L.W. (2010). Transformation of UML-based system model to design/CPN model for validating system behavior. *System Engineering*, 5(4), 288-312.