# Modeling Timed Elaborate Requirements in Service-Oriented

Seyed Morteza Babamir (babamir@kashanu.ac.ir)[1] and Seyyed Hosein Seyyedi Arani (h.seyyedi@yahoo.com)[2]
[1] Department of Computer, University of Kashan
[2]Islamic Azad University, North Tehran Branch

**Abstract -** *In recent years, various organizations by following the concept of service oriented architecture, offer their services with independent and reusable programs on Internet. Since these services can be called by application programs and other services, the concept of implementing inter-organizational workflow by dynamically composing the services is being developed. The necessary requirement of this development is the existence of formally defined standard methods for specification of these compositions in an abstract way.*

*In a service oriented system, users' requirements fall into simple and elaborate categories. To satisfy the former, it is sufficient to call one service; however, to satisfy the latter, it is necessary to call a composition of services. Additionally, to satisfy many of elaborate requirements, time constraint is a determinant of requirement satisfaction. In this paper, we first specify common types of elaborate and time-based users' requirements in service oriented systems, and then using an three-step approach, suggest a specific composition of services for each type of requirement. In the first step of the approach, an operator is introduced for informal stating a composite service. In the second step, the composition is formally specified using a model based on Transition Timed Petri-Nets, and in the third step, the model is defined formally.*

**Keywords:** Service oriented architecture (SOA), Web service, Services composition, Requirement specification, Petri-Nets.

## 1 Introduction

Service oriented architecture (SOA) [1,2,3], is a distributed architecture in which an integrated set of services (web services) interactively with each other, offers various services to clients. (services are reusable, autonomous and self descriptive application programs that can be accessible via messaging). Services can be presented on a network of computers with different operating systems platforms as well as various programming languages.

In SOA, every person or organization can be a service provider or a service requester and the communication between them is achieved with XML messages and SOAP [4] protocols. Services specifications are stored in a repository called UDDI[1] [5]. The language that is used for this specification is WSDL[2] [4].

Before SOA, distributed object oriented architectures were popular and amongst the CORBA [6] is distinguishable from the rest. Hence a comparison of SOA with CORBA is useful. In CORBA, the idea is that every application program can remotely access objects of other programs and therefore must specify and call the method and its parameters. In this way, when a client requests a service from a service provider, it would be necessary for the client to know the names and the parameters needed for the call. Thus there is not fully abstraction for the client. However, in SOA messages and operations for accessing the services are defined independent of physical implementations and technical details. In fact, client only specify what has to be done and does not involve in how it is being carried out.

In SOA, data is available to all systems and every service provider receives a SOAP message, carries out the needed operations on its local data with its local methods and procedures and returns the result to the client. Therefore, in SOA dependency between programs is omitted. Also, in SOA it is possible to provide services dynamically, at execution time and according to users needs, whereas CORBA has not this feature.

In CORBA, a large set of rules exists which are complex and difficult to learn by users and also setting up, maintenance and transfer of programs are not simple. However in SOA, only Internet standards such as HTTP are needed which are available everywhere and to learn and use them is easy.

After object oriented architectures, the message oriented middleware (MOM[3]) [7], was developed which did not have the problems of object oriented architecture. In this architecture, all clients and service providers must install the middleware on their operating systems in order to be able to have transparent communication with each other; however in SOA, communication between a client and a service provider is completely based on SOAP and extra software is not needed.

In SOA, as a distributed system, different programming languages and platforms exist, and any service can interact with other services independent of its programming language and operating system. Hence extensive costs of integrating platforms are omitted and concentration on the purpose of a service or services composition is possible.

It is necessary that services, as reusable elements, be able to participate in various compositions in order to each composition can carry out a process. In the past, for executing a process in a form of chained tasks by different organizations (B2B[4]), it was necessary to use a human agent to carry the result of one task and hand it over to an operator for performing the following task, and this human interference caused more cost, time and risk of human errors. Therefore automation of B2B process has been introduced as a big goal. SOA has been achieved to the goal in this way that a chain of tasks in a process from beginning to the end, can be performed by a composition of base services. However to make a big improvement in electronic services on the web environment, the ability of composing reusable ready services, quickly and dynamically, with respect to variable needs and conditions, is essential. The result would be organizations agility in response to variable conditions.

To reach these capabilities, use of visual and descriptive tools that can quickly and easily model services and their compositions as well as analyzing them before implementing, was necessary and in this paper we discuss these issues, the tool, that we use for specification of services is Petri-Nets in

[1] Universal Description, Discovery and Integration
[2] Web Services Description Language
[3] Message Oriented Middleware
[4] Business To Business

which the operation of services and their compositions are shown visually and abstractly that makes it easier for the reader to understand them. Also for the presented models, services can be analyzed and features such as deadlocks and accessibility in service compositions are discussed.

Previously, Petri-Net was used for specification of services and their compositions [8] but the important parameter of time was not considered. Timing limitations in receiving a response from services is an important issue that we will discussed in this paper. In SOA, accessibility of service when invoked and also, receiving a response from it in specified time after issuing a request is not reliable and guarantying service execution in a specified time is an important challenge. Most of the services must have a timing parameter to insure quality of service. For instance, consider a shopping service that must be activated for shopping online based on SOA, and by convention, selling agent must assure delivery of chosen goods to the client address within a specified time. In most of services like this example, disregarding time constraint causes the outcome of a service delivery useless. Therefore, discussing the timing issue in SOA is extremely important.

In a service oriented system, users' requirements fall into simple and elaborate categories. To satisfy the former, it is sufficient to call one service; however, to satisfy the latter, it is necessary to call a composition of services. Additionally, to satisfy many of elaborate requirements, time constraint is a determinative factor for requirement satisfaction.

In this paper, we first specify common types of elaborate and time-based users' requirements in service oriented systems, and then using a three-step approach, suggest a specific composition of services for each type of requirement. In the first step of the approach, an operator is introduced for informal stating a composite service. In the second step, the composition is formally specified using a model based on Transition Timed Petri-Net, and in the third step, the model is defined formally. The use of the approach makes: (1) easy analyzing elaborate users' requirements of service-oriented systems and (2) acquiring a appropriate composition of the services. By the approach, it is possible to verify some cases such as deadlock and state reachability.

In section 2 of this paper, related works are pointed out and SOA and its components are introduced in section 3. The need for visual modeling as well as introducing Petri-Nets as one of the tools for modeling are discussed in section 4. The base services we used are introduced in section 5. Common types of elaborate requirements and our approach for specifying composite services which satisfy them are presented in section 6. The technologies that support SOA are described in section 7. Finally, we conclude our discussion in section 8.

## 2 Related works

Since a decade ago, workflow management systems have become popular. The idea of these systems was that for an organization to be successful, it is necessary to manage its internal business processes. However, today what is known as B2B operation is inter-organizational workflow which its purpose is to automate the existing commercial processes between different organizations so that, for reaching a common target, they interact with each other. With presence of SOA, automation of this process with the noticeable idea of dynamic composition of services at execution time is being followed. All of these ideas need tools for specification.

In case of workflow management, [9,10,11] specified workflow using Petri-Nets. In [12] authors has specified inter-organizational workflows using Petri-Nets. [13], by following this work and considering every organization functionality as a service, has modeled workflow between them as a composition of services.

In [8], a number of operators for proposing different compositions of web services were given and formal meaning of these operators is shown using Petri-Nets. Any relation among services shown as an expression of these operators can be converted to a model in Petri-Nets. Also, by using several features for these operators, it is possible to transform and improve relationships between them in such a way that their initial properties be unchanged.

This work is valuable in our view, because the existing relations between web services in most of the compositions can be considered as an expression consisting of one or several introduced operators and hence would cause simplicity and organization in expressing the compositions. Also, presenting formal specification of the compositions using Petri-Nets is appropriate. However, in this work, time as an effective parameter is not considered. In service oriented environment, it is possible that a service do not be accessible sometimes or unable to respond in a certain time period. Hence, ability to guarantee a service operation in a time period is a major difficulty. In business applications that use service orientated paradigm, to guaranty quality of service, time limitation must be considered. New feature of our work, is presenting an approach for expressing and specifying compositions that are faced with timing limitations. Therefore, several common kinds of requirements with this limitation are stated and for response to every kind, a composition of web services is suggested. Then, by introducing an operator, the meaning of that composition is expressed. Finally every composition is formally specified using timed Petri-Nets and a formal definition of that is presented.

## 3 Service Oriented Architecture

SOA is a way of designing distributed software systems using services as building blocks. Services are independent of platform, autonomous and reusable applications which are identified and accessible by related interfaces. Communication with services can be done by message passing without need to any knowledge of detailed internal information of them. This means that to use these services, it is not necessary to know how they are implemented. Services which are implemented with different programming languages on different operating systems can interact and composed together to provide bigger services for implementing processes.

The three fundamental standards of web services technology are: WSDL, SOAP and UDDI. Structure of data in different documents that these standards deal with is XML. In XML as compared to HTTP, labels are not only used for formatting data to be presented, but provide a tree structure for data. Additionally in comparison with HTTP frame, that use a constant set of predetermined labels, XML allows users to specify their own labels.

The first layer of SOA illustrates how different standards can be used for transferring information in service oriented environment. SOAP (located in the second layer of SOA) is an important standard in SOA and used to describe message which interchange with a service at execution time to call it. The SOAP message comprises an XML document in which operations that must be carried out and parameters sent to a service are described. Often, a SOAP message might include other information for stating how a message must be processed by a receiver.

WSDL (located in the third layer of SOA), is the language used for description of services, these descriptions are placed in UDDI. The information content of this description is: (1) description of different parts of a service, (2) network address in which the service is placed and (3) how to call the service or in other word, formatting of the messages that must be exchanged for service execution.

UDDI (located in the third layer of SOA), is a service registry, in which service providers place their and their services information. When requester, needs a service, UDDI is searched and using existing information about the service provider, requester bind to the provider and the service would be offered. Searching for services in UDDI is based on name and identifier of services and name of groups that they are belong to.

The fourth layer of SOA shows some standards that are stated for quality of service in SOA. The fifth layer of SOA shows kinds of services according to their structure. A service can be atomic (base) or composite. Every composite service is made of several atomic ones, which interact with each other to implement a process. Management of this communication is by one of orchestration or choreography methods. According to the first method, a central coordinator controls the base services execution. In the second one, base services inhabit according to a plan that they all compromise on it in advance, and therefore they are coordinated. Compositions that we propose in this paper are according to the latter method.

The sixth layer of SOA shows possible ways that via them users can use services. Users can call a service directly via browser. Consequently, the result would be shown to them via it. Also they can propose their request via user interface of an application. Consequently, the application would call the service and get the result and deliver it to the user.

# 4 Petri-Nets

Petri-Net [14,15,16] is a graphical and mathematical tool, used for specification and study of concurrent ,asynchronous and/or distributed information processing systems. Since service oriented systems can have these features, Using Petri-Nets for specification of these systems is appropriate.

A Petri-Net shown in Figure 1 is a directed and connected graph and has three components: (1) nodes indicate either a place or a transition. In Figure 1, nodes p1 and p2 are places and nodes t1 and t2 are transitions. (2) Arcs indicate connection between a place and a transition and vice versa such as that one connects place p1 to the transition t1. An arc can not directly connect two places or two transitions. (3) Tokens which are placed in places. If there is at least one token in every input place of a transition, the transition is called "enabled" such as the transition t1 in Figure 1. When a transition is fired a token will be removed from each its input place and a token will be placed in each its output.

"The use of visual modeling techniques such as Petri-Nets in the design of complex Web services is justified by many reasons. For example, visual representations providing a high-level yet precise language allows to express and reason about concepts at their natural level of abstraction. A Web service behavior is basically a partially ordered set of operations. Therefore, it is straight-forward to map it into a Petri-Net. Operations are modeled by transitions and the state of the service is modeled by places. The arrows between places and transitions are used to specify causal relations." [8] Thus, firing of a transition that causes moving tokens from some places to others, models an operation that changes the state of a system.

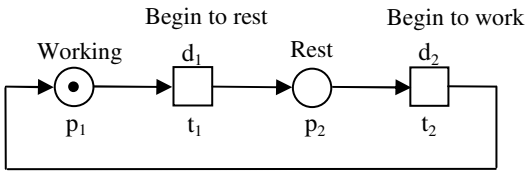


Figure 1. A TTPN

Timed Petri-Net (TPN), is a Petri-Net in which timed places or timed transitions exist. In this paper, Timed Transition Petri-Nets (TTPN) is used. A timed transition with a timed label d is a transition which fires after the time delay d of enabling time.

The Petri-Net shown in Figure 1 is a TTPN in which t1 and t2 are timed transitions. This TTPN models behavior of a person that sometimes does a job and sometimes rests. When the token is placed in p1, the person is doing its job. After the elapsed time d1, rest time arrives (transition t1) and the person state changes to rest state (residing of the token in p2). After the elapsed time d2, from beginning of the rest, rest time terminates (transition t2) and the person begins doing its job again (residing of the token in p1).

## 4.1 Formal definitions

**Definition 1.** "A Petri-Net is a 5-tuple,

$PN = (P,T,F,W,M_0)$ where:

- $P = \{p_1,p_2, \ldots,p_m\}$ is a finite set of places,
- $T = \{t_1,t_2,\ldots,t_m\}$ is a finite set of transitions,
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),
- $W: F \rightarrow \{1,2,3,\ldots\}$ is a weight function,
- $M_0: P \rightarrow \{0,1,2,3,\ldots\}$ is the initial marking,
- $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$." [16]

Any number $n \geq 1$, as a number of weight function, written on an arc, if the arc connects a place to a transition will mean the necessary requirement for enabling the transition is existence of n tokens on the place and if the arc connects a transition to a place will mean that by firing of the transition, n tokens will be placed on the place. In our presented models we assume that all number of weight function are 1, and hence we do not mention weight function for our definition of models.

Input place is a place including no input arc and when a token is placed on that place, it will mean the service has received the necessary information from environment for its operation and is in the ready state. Output place is a place that dose not have output arc and when a token is placed on that place, it will mean that the service has returned the results of its operation to environment and is in the complete state. In this paper, we assume that Petri-Net which describes a service or a composition of services has only one input place and one output place. Also, it is necessary to mention that in the above definition, initial marking, $M_0$, is number of tokens in places of

Petri-Net at beginning. We assume that at beginning, the number of tokens in input place of the models are 1 and in other places are 0, and therefore in our definitions we do not mention $M_0$.

According to definition 1, a formal definition for a TTPN can be stated as:

**Definition 2.** A Petri-Net is a 6-tuple, TTPN = PN ∪ LT where:

- PN is a Petri-Net, and
- LT: T → D is latency time function of transitions. D, set of transitions latency numbers, is a set of numbers that each of them is the latency time number of a transition. This number shows that how many units of times after enabling, the transition will fire.

In this paper, for each transition which latency time number is larger than 0, the related number is showed by a constant d, that is put beside transition. For transitions that this constant does not appear beside it, this number is zero.

## 5 Base services

Each service can implement a specific operation that it is developed for it. However, to reply to a majority of requirements, a process must be done that for implementing it, several base services must be composed. For implementing different processes, base services communicate and coordinate with each other in different shapes, each shape appropriate for the process being implemented. Hence for implementing a process, a composite service will be developed by composing base services.

Elaborate requirements are those that a couple of services should be executed and therefore a composition of base services should be developed. In the section 6, we will suggest a couple of service compositions for madding reply to common kinds of elaborate requirements of users in service oriented systems. We assume that necessary base services for the compositions are ready; so, we should only concentrate on how to compose them. The base services, S1,…,S5 are shown in Figure 2. As shown in Figure 5, a base service is represented by a rectangle and only those nodes that play a role in communication between services. Different shapes considered for these services are to distinguish between them in a composition and is in some cases because of their different roles which must be played in compositions.

Sets of transitions considered for services S1,…,S5 are T1,…,T5, respectively. Sets of places considered for these services are P1,…,P5, respectively. Sets of arcs of these are F1,…,F5, respectively and sets of latency time numbers of their transitions are D1,…,D5, respectively. Also, statement (x,y) means an arc which connects node x to node y.

## 6 Requirements and Suggested Approach

In this section, we propose 4 common kinds of users' requirements of service oriented systems, and offer appropriate compositions for replying to them and also our approach to expressing and specifying them. The property of these requirements is existence of time constraint in them which effects in control flow of related compositions. We chose these 4 common kinds to show implementing of our approach upon them, but these are not all possible kinds. To specify appropriate compositions for other similar kinds of requirements, the approach offered here, can be used. For informal specifying compositions, new operators must be identified. For formal specifying them using TTPN, every operation must be modeled as a rectangle representing related service and every time constraint must be modeled as a timed transition in appropriate place between these services. Also, for identifying these compositions formally, the definition offered in this paper for TTPN, can be used.

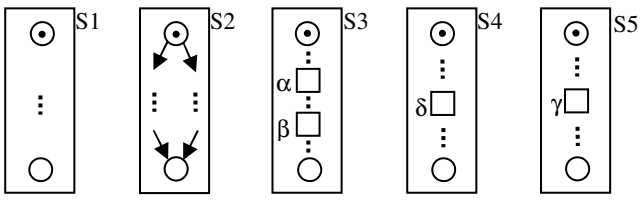


Figure 2. Base services

**Requirement 1.** A service should be executed but after it is called, the reply should not take more than a specified time period. If a reply is received within this time period, the service is finished, otherwise another service, which have the same functionality must be requested from a different provider. After this request, any reply received sooner is accepted and other reply is ignored. This means execution of one of the two services is enough but if a reply within the specified time period is not received, this is possible that the service provider can not respond or/and the response will arrive late. Therefore, the second service is called to reduce the risk of relying only on one service provider.

**Example.** All the cases where the requested service is provided by different providers can be as an example for this requirement. Because in these cases, it is possible that the requester be unsure about the accessibility of the service and hence if a first provider's reply does not arrive within a specified time period, requests the same service from another provider. Of course, it is also possible that two services be called simultaneously, so that they execute in parallel. For carrying out the job in shortest time, without considering limitations, parallelism is the best method but in cases where reduction in costs of a service delivery is important, the mentioned way is the best way because if both service providers execute the service in parallel, more resources on network is used and also, the requester must pay both providers.

As another example, the protocol "At least once semantics" [7], in distributed systems, can be mentioned where a request such as reading from database must be executed at least once, and if be done more than this, there is no problem and have no value.

**S1 $\oplus_T$ S2.** This expression shows a composition of two services S1 and S2, (section 5), appropriate for response to requirement 1. For expressing control flow between two services, we identify operator $\oplus_T$. Function of the operator shown in Figure 3, using a model based on TTPN. According to the figure, firing of transition t1, causes the service S1 to begin and at the same time a token is placed in p2. If S1 finish its execution before expiring the time period d, execution of composite service will finish. Otherwise, with respect to the presence of a token in place o1 and an "Inhibitor Arc" [17] between place o1 and transition t2, this transition fires and causes service S2 to start. Inhibitor arc is distinguishable from other arcs by a small circle at the end of it, at transaction side. Its difference in functionality is that necessary requirement for enabling of its connected transition is nonexistence of token in its connected place. With the start of the service S2, it is probable that any of the two services finishes sooner. With termination of any of the two, the

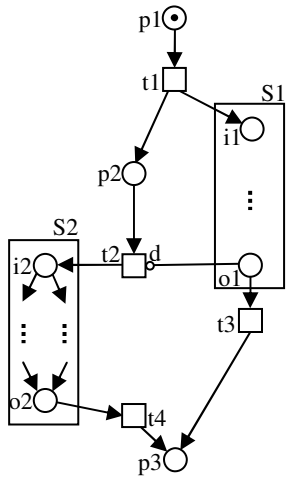


Figure 3. A presentation of S1 $\oplus_T$ S2

composite service will finish and continuation of the other service will be unimportant.

**Definition 3.** TTPN related to composite service S1 $\oplus_T$ S2 (Figure 3), is a 6-tuple, TTPN = (P,T,F,W,$M_0$,LT), where in it:

- P = P1 ∪ P2 ∪ {p1,p2,p3},
- T = T1 ∪ T2 ∪ {t1,t2,t3,t4},
- F = F1 ∪ F2 ∪ {(p1,t1),(t1,i1),(t1,p2),(p2,t2), (t2,i2),(o1,t3),(t3,p3),(o2,t4), (t4,p3)},
- ($o_1$,$t_2$) is an Inhibitor arc in it, and
- LT: T → $D_1$ ∪ $D_2$ ∪ {0, d, 0, 0}.

Figure 4 shows a TTPN, for the same composite service, that its definition is like definition 3, with this difference that inhibitor arc (o, t2), is not exist in it.

**Requirement 2.** Two services must be executed and they can be executed in parallel, if execution of one of them that is called second one, respect to its execution order, be started when a specified time period elapses from starting the first one execution.

**Example.** In a two part exam, each part is carried out by a service. According to a plan, the second part must begin, a specified period of time, after the first one was begun. It is not

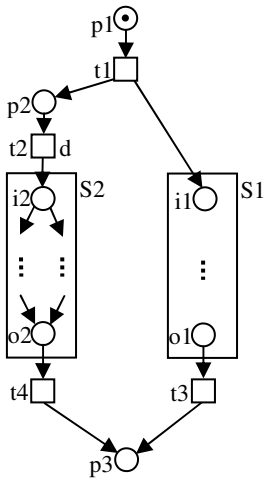


Figure 4. Another presentation of S1 $\oplus_T$ S2

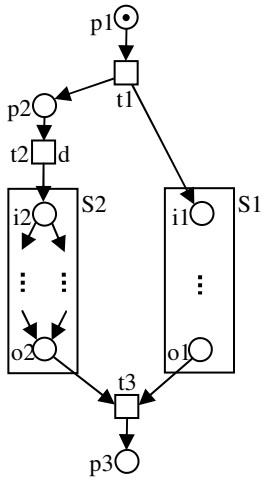


Figure 5. S1 $\|_T$ S2

necessary for a candidate to finish replying to the first part of the exam before the start of the second part, but he can continues replying to the first one concurrently. When the candidate finalizes his reply to two exam parts, the whole of exam is finished. Therefore in this example, the first service starts first and after elapsing specified period of time the second one begins its execution. It is not necessary that the first service can be finished at this moment and from this moment two services can be executed parallel. When the two executions of services are finished, the composite service will be terminated.

**S1 $\|_T$ S2.** This expression shows a composition of two services S1 and S2 (section 5), which is appropriate for responding to requirement 2. For expressing control flow between two services, we identify operator $\|_T$. Operation of this operator is shown in Figure 5, using a model based on TTPN. According to Figure 5, firing transition t1 causes beginning the service S1 and at the same time a token is placed in p2, which this enables transition t2. After elapsing time period d, regardless of whether or not S1 is finished, transition t2 fires and service S2 starts its execution. Termination of this composite service execution will be due to the termination of both base services executions.

**Definition 4.** TTPN related to composite service S1 $\|_T$ S2 (Figure 5) is a 6-tuple, TTPN = (P,T,F,W,$M_0$,LT), where in it:

- P = P1 ∪ P2 ∪ {p1,p2,p3},
- T = T1 ∪ T2 ∪ {t1,t2,t3},
- F = F1 ∪ F2 ∪ {(p1,t1),(t1,i1),(t1,p2),(p2,t2), (t2,i2),(o1,t3),(o2,t3),(t3,p3)},
- LT: T → $D_1$ ∪ $D_2$ ∪ {0,d,0}.

**Requirement 3.** The execution of a service must repeats in a specified time period and when the time period finished, the service execution must be stopped.

**Example.** Selling a product that is offered via a service, within a specified period of time is with extra facilities, and the service can not be called by buyers, after this time period.

**$\mu_T$S1.** This expression shows a composition consist of the service S1, (section 5), appropriate for response to requirement 3. For expressing control flow in this composition, we identify operator $\mu_T$. Operation of this operator is shown in Figure 6, using a model based on TTPN. According to the Figure 6, firing of transition t1 causes beginning the service S1 and at the same time a token is placed in p2, which this enables transition t2. Termination of S1 execution, which causes a token to be placed

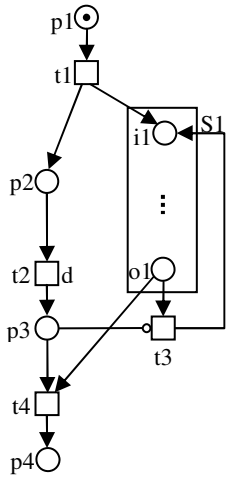


Figure 6. $\mu_T$S1

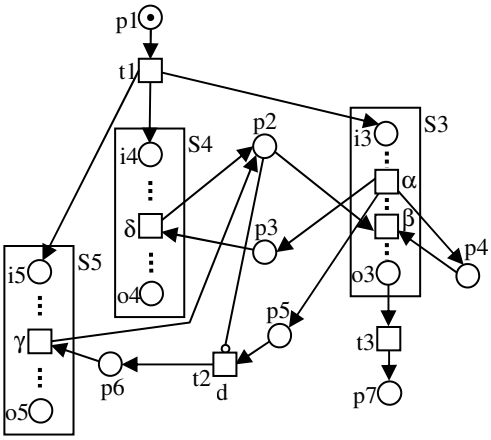


Figure 7. : S3 $_T\|_\delta$ S4 $\|_\gamma$ S5

in place o1, it is possible that one of two following states occurs: (1) if the time period d is not elapsed and as a consequence, transition t2 is not fired and a token is not placed in place p3, then with respect to the arc (p3,t3), an inhibitor arc, the transition t3 fires and the service starts its execution again, (2) if the time period d, is elapsed and as a consequence, a token is placed in place p3, then the transition t4 fires and the composite service will terminate.

**Definition 5.** TTPN related to composite service $\mu_T$S1 (Figure 6), is a 6-tuple, TTPN = (P,T,F,W,$M_0$,LT), where in it:

- P = P1 ∪ {p1,p2,p3,p4},
- T = T1 ∪ {t1, t2,t3,t4},
- F = F1 ∪ {(p1,t1),(t1,i1),(t1,p2),(p2,t2),(t2,p3), (o1,t3),(t3,i1),(o1,t4),(p3,t4), (t4,p4)}
- And (p3,t3) is an inhibitor arc in it, and
- LT: T → $D_1$ ∪ {0,d,0,0}.

**Requirement 4.** A service in its execution time needs some operations of another service which is being executed in parallel and hence that service must be called and the returning response must not take more than a specified period of time. If the caller service does not receive a response within that period, another service that does similar operations must be called. As soon as one of called services responds, the caller service operation will continue and the response of the other service will be ignored. In fact, when an operation of a caller service is terminated the composite service will be terminated.

**Example.** Some products are made from several components. A service which does the providing and selling operations for that product may request a component for the product from the service of distributor of that component and the sale service of the final product in one point of the course of its execution, must receive the requested component to continue its operation which might include coordination of the received component with other ones to produce final product. Therefore, this service must not wait more than a planned timing period and if the requested component is not delivered, must request this component from sale service of another distributor of it. From this time on, the component of whichever distributor, which be delivered sooner will be used and the response from the other distributor will be ignored.

**S3 $_T\|_\delta$ S4 $\|_\gamma$ S5.** This expression shows a composition of three services S3, S4 and S5, (section 5), appropriate for response to requirement 4. For expressing control flow between three services, we identify operator $_T\|_\delta$ $\|_\gamma$, in this expression. Operation of this operator is showed in Figure 7, using a model based on TTPN. According to the figure, firing of transition t1, causes the service S3 to begin. By firing transition α of this service, transition δ of S4 is enabled and also a token is placed in both p5 and p4. If δ, be fired before elapsing time period d, which starting of it is the moment of placing token in p5, a token is placed in p2 and transition β of S3, is enabled and this service, which needs result of transition δ to follow its operation from transition β, can continue its operation. With termination of this service, the composite service will be terminated, also. Otherwise, with elapsing the time period d, with respect to the existence of token in p5 and nonexistence of token in p2, the transition t2 fires and enables the transition γ, from S5, which has similar functionality as δ. After which, the quicker response from either of services S4 or S5, will cause S3, to continue its operation and the composite service to be terminated and with considering that firing of β, will cause removing token from p4, after its firing, whether or not another service respond, will not have any effect.

**Definition 6.** TTPN related to composite service S3 $_T\|_\delta$ S4 $\|_\gamma$ S5 (Figure 7), is a 6-tuple,

TTPN = (P,T,F,W,$M_0$,LT), where in it:

- P = P3 ∪ P4 ∪ P5 ∪ {p1,p2,p3,p4,p5,p6,p7},
- T = T3 ∪ T4 ∪ T5 ∪{t1, t2,t3},
- F = F3 ∪F4 ∪F5 ∪{(p1,t1),(t1,i3),(t1,i4),(t1,i5), (α,p4),(p4,β),(α,p3),(p3,δ),(δ,p2),(p2,β),(α,p5), (p5,t2),(t2,p6),(p6,γ),(γ,p2),(o3,t3),(t3,p7)}
- And (p2,t2) is an inhibitor arc in it, and
- LT: T → $D_1$ ∪ $D_2$ ∪ $D_3$ ∪ {0,d,0}.

# 7 Conclusion

In this paper, several common kinds of elaborate requirements in service oriented environments in which time parameter can have effects were proposed. Then, a composition of services that can respond to each requirement was suggested and an approach for expressing and specifying these compositions was offered. In this approach, firstly, every composition is expressed informally by introducing an operator and then is specified formally by using a model based on TTPN

and finally is defied formally. Offered models of compositions show them visually and clearly for different readers.

In case of a need for repeating resulted composition to several levels, the yielding compositions in higher levels can be expressed easily by an expression of proposed operators and modeled by composing the proposed models.

To precede the time management issue in service oriented environments, we can extend the represented approach in this paper. This issue is particularly useful in E-commerce. Also, discussion of other aspects of quality of service in these environments can be very useful to continue in future.

## References


[1] Newcomer E., Lomow G., 2004, *Understanding SOA with Web Services*, Addison Wesley.

[2] Erl T., 2005, *SOA; Concepts, Technology & Design*, Prentice Hall.

[3] Josuttis N., 2007, *SOA in Practice*, O'Reilly.

[4] Weerawarana S., Curbera F., Leymann F., Storey T. and Ferguson D., 2005, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*, Prentice Hall.

[5] Apte N., Mehta T., 2002, *UDDI: Building Registry-Based Web Services Solution*, Prentice Hall.

[6] Sommerville I., 2007, *Software Engineering*, 8th Edition, Addison Wesley.

[7] Tanenbaum A., Van Steen M., 2007, *Distributed Systems; Principles and Paradigms*, Prentice Hall.

[8] Hamadi R., Benatallah B., 2003, *A Petri Net-based Model for Web Service Composition*, Proceedings of the 14th Australasian database Conference (ADC'03).

[9] Aalst W., 1997, *Verification of Workflow Nets*, Azema P., Balbo G. Editors, Proceedings of 18th International Conference on Application and Theory of Petri Nets (ICATPN'97), Toulouse, France, pp. 407-426.

[10] Aalst W., 1998, *The Application of Petri Nets to Workflow Management*, Journal of Circuits, Systems and Computers 8(1), pp. 21-66.

[11] Aalst W., Hee K., 2002, *Workflow Management: Models, methods and systems*, MIT Press.

[12] Aalst W., Weske M., 2001, *The P2P Approach to Interorganizational Workflows*, Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAISE'01).

[13] Aalst W., Pesic M., 2007, Baresi L., Nitto E. Editors, *Test and Analysis of Web Services,* Springer, Chapter 2, pp. 11-55.

[14] Peterson J., 1981, *Petri Net Theory and the Modeling of Systems*, Prentice Hall.

[15] Reisig W., 1985, *Petri Nets: An Introduction*, Springer.

[16] Murata T., 1989, *Petri Nets: Properties, Analysis and Applications*, Proceedings of the IEEE, April 1989, Vol. 77(4).

[17] René D., Hassane A., 2005, *Discrete, Continuous, and Hybrid Petri Nets*, Springer.