# Chapter 6
# Formal Modeling of Enterprise Cloud Bus System: A High Level Petri-Net Based Approach

**Gitosree Khan, Sabnam Sengupta and Anirban Sarkar**

**Abstract** The chapter focuses on an abstraction layer of SaaS architecture for multi-agent-based inter-cloud environment, called Enterprise Cloud Bus System (*ECBS*) to conceptualize the different behavioral facets of such system in service and cloud computing paradigm. The model is formalized using a set of high level Petri-net-based formal constructs called High Level Enterprise Cloud Bus Petri-net (*HECBP*) with varieties of relationship types among participation cloud bus components. It is accompanied with a rich set of Petri-net graphical notations and those are used to specify the effective toward modeling interactions among the heterogeneous agent present within the cloud bus of *ECBS* at conceptual level design of multi cloud system. The approach facilitates to analyze the behavioral features of inter-cloud architecture and modeled its dynamics at the conceptual level. The *HECBP* is also able to ensure correctness and performance of the system at design time by focusing on meeting the increasing demands for distributed software as a service and making the system functionality more scalable, configurable, and shareable. This chapter includes modeling of several behavioral facets like, fairness, boundedness, liveliness, safeness, etc., in a dead lock-free way. Moreover, this chapter provides a discussion on state-space analysis study, which further validates the theoretical analysis of *HECBP* model and future research scope in this area.

**Keywords** Cloud computing · Service-Oriented architecture · Enterprise cloud bus system · Multi-agent system · Behavioral analysis · High-level enterprise cloud bus Petri-net · Colored Petri-net · Boundedness · Liveness · Reachability · Safeness

G. Khan (✉) · S. Sengupta
B.P. Poddar Institute of Management & Technology, Kolkata, India
e-mail: khan.gitosree@gmail.com

S. Sengupta
e-mail: sabnam_sg@yahoo.com

A. Sarkar
National Institute of Technology, Durgapur, India
e-mail: sarkar.anirban@gmail.com



M. Ramachandran and Z. Mahmood (eds.), *Requirements Engineering for Service and Cloud Computing*, DOI 10.1007/978-3-319-51310-2_6

## 6.1 Introduction

Service-Oriented Computing is an emerging computing paradigm for requirement engineering that utilizes service as software as the basic constructs to support the development of rapid and low-cost composition of software applications. Growing complexity of Enterprise Software applications and increasing numbers of clouds throughout the world have increased the challenges for Software as a Service (*SaaS*) in the recent trends of requirement engineering. Therefore, the need for Cloud computing has evolved as an important key areas of software engineering research and practices, which identifies functional requirements from users along with its benefits of cost effectiveness and global access. By providing on demand access of services to a distributed environment of computing resources in a dynamically scaled and virtualized manner, agent-based cloud computing offers compelling advantages in cost, speed, and efficiency.

Along with the emergence of Service-Oriented Architecture (*SOA*), Software as a Service (*SaaS*) architecture has emerged as a new and evolving domain in cloud computing based on the request/reply design paradigm for Enterprise Software applications. Existing ESB-based systems cannot address the complexity of Enterprise Software applications due to the increase in number of clouds and their services. Therefore, service registration, discovery, scheduling, and composition of services are facing complexity and performances issues nowadays. To address such issues, our previous work focused mainly on the abstraction layer of Software as a service architecture, called Enterprise Cloud Bus (*ECB*) [1, 2]. It models the services, agents, and their interconnections for all the locations for satisfying the client needs. This approach is introduced by integrating agent technology in *ECB* framework.

Modeling and design methodologies for multi-agent-based cloud infrastructure have not taken a shape as yet. One of the most challenging domains is to model various agent-based architecture of *SOA* in distributed cloud computing environment to make the *SOA* system modeling more reliable and robust. Petri-net-based approach is a relevant choice for modeling and analyzing the dynamic behavior of such agent-based cloud architecture. However, those approaches are also less expressive for large systems like inter-cloud architecture comprising of multiple agents and components. Therefore, colored Petri-net (*CPN*) tool is an efficient tool that is used for constructing and analyzing such multi-cloud system. Many of the research works reveals about the behavioral analysis of multi-cloud architecture using colored Petri-net. But they are some shortcomings toward analysis of dynamics multi-cloud architecture. Therefore, high-level Petri-net-based approach is the most suitable one toward analysis of dynamics of such system.

With the aforementioned objectives, the chapter has been organized in six sections. In Sect. 6.2, previous researches related to modeling of multi-agent-based inter-cloud architecture have been summarized with major emphasis on the models based on cloud computing paradigm. In Sect. 6.3, the concept and importance of multi-agent-based inter-cloud architecture (*ECBS*) have been summarized with

major emphasis on the conceptual definition followed by several issues that need to be handled related to multi-agent system (*MAS*) paradigm. This section also introduced the comprehensive summary of key benefits and challenges of *ECBS* in *MAS* environment. In Sect. 6.4, a novel approach on formal modeling and analysis of *ECBS* using high-level Petri-net has been summarized. Here, the colored Petri-net (*CPN*) tool is used for simulation of the model so that this chapter will serve as a valuable source of information for academia and industry related to multi-agent inter-cloud model-based testing especially for cloud-based enterprise applications in optimizing the performance, cost, elasticity, flexibility, high reliability, and availability of the computing resources. This section also includes the analysis of the behavioral features like, safeness, boundedness, liveliness etc., of the model in order to ensure the system dynamics such as high accuracy of system functionality, operational design of system flexibility that comprises of autonomous agents and system components, automation of cloud services and quantitative measurement of system processes, performance, and scalability. In Sect. 6.5, the future research directions of *ECBS* framework using high-level Petri-net have been summarized. Finally, the chapter has been concluded in Sect. 6.6 with identification of key areas of research in the domain of requirements engineering for service and multi-agent-based inter-cloud architecture.

## 6.2 Related Research

Nowadays, Enterprise Software applications are growing in complexity; therefore, there is a rapid increase in number of clouds and their web services. Thus, the demand for cloud computing technology toward organizations is growing exponentially. Lots of researchers discuss over the architectural design of cloud computing and its applications. Among them, [1–3] focus on the architectural driven environment for cloud applications that facilitates monitoring cloud services, composing, and adapting cloud applications. Putting a distributed cloud-based integration and deploying in Enterprise Service Bus with service-oriented architecture processes help enterprises to integrate their process over the cloud, and achieve scalable cloud enabled business applications with greater efficiency. In such cloud computing environment, the competition between service providers has led to a large number of cloud-based solutions that offered to consumers. Moreover, in the work proposed in [4] the author focuses on "vision, challenges and architectural elements of inter-cloud environments." The ability to run and manage the multi-cloud system [5, 6] is a challenge for preventing interoperability and increase scalability, elasticity, and autonomy of multi-cloud system.

The work in [7, 8] provides a "classification of the state-of-the-art of cloud solutions and discusses the dynamic provisioning, deployment and adaptation of dynamic multi-cloud behavior systems which aim at addressing the issues."

The author in [9] proposes a new approach for dynamic autonomous resource management in cloud computing. Several researches, in last decade, have devised conceptual model for multi-agent-oriented system [10, 11]. But, all these approaches have got certain limitations to exhibit the dynamism of internal behavior of the system which comprises of heterogeneous set of components.

In this context, analysis of such dynamics is a major challenge. For the purpose, proper mechanism is required to conceptualize and study the behavioral properties of multi-cloud architecture. Agent-based architecture is most acceptable paradigm to handle the dynamicity of such large-scale system like inter-cloud architecture. The works proposed in [12] have discussed about the structural definition of agent-based hybrid multi-cloud applications. In [13], the structural modeling of agent-based multi-cloud architecture called enterprise cloud bus (ECB) has been discussed followed by service discovery mechanism [14, 15] that helps to identify services during run time. The structural components of ECB system is modeled using UML 2.0 [16]. But, the UML modeling is not suitable for rigorous analysis of multi-cloud dynamics due to its semi-formal nature.

Petri-net [17]-based approach is an obvious choice for modeling and analyzing the dynamic behavior of agent-based inter-cloud architecture. In [18], the modeling and analysis of agent-oriented system has been stated by the author. However, those approaches are also less expressive for large systems like Inter-cloud architecture comprising of multiple agents and components. Moreover, in [19], the conceptual model of multi-cloud architecture called Enterprise Cloud Bus Petri-net (ECBP) has been proposed in MAS domain and modeled its dynamics using Petri-net-based tool called PIPE. For detail reference of modeling and analysis of multi-agent system refer [26–29] of additional reading section.

Therefore, colored Petri-net (CPN) Tools is an efficient tool [20] that is used for constructing and analyzing such multi-cloud system. The work in [21] reveals about the behavioral analysis of Multi-cloud architecture using colored Petri-net. But there are some shortcomings toward analysis of dynamics multi-cloud architecture. Therefore, high-level Petri-net-based approach [22, 23] is most suitable toward analysis of dynamics of such system and accepted as standard. In Sect. 6.3 of this chapter, the ECBS has been stated formally and modeled using high-level Petri-net-based approach, called high-level enterprise cloud bus Petri-net (HECBP) which further used to represent the behavioral analysis of ECBS using high-level Petri-net tool, called CPN.

Moreover, the concepts of HECBP have been implemented using a simulation tool called CPN for further validation of the architecture. The proposed approach is effective toward modeling interactions among the heterogeneous agent present within the cloud bus of ECBS. The model is also effective toward analysis of the key behavioral features of multi-cloud architecture. For detail reference on High-level Petri-net according to ISO/IEC [43] standard of additional reading section.

## 6.3 Multi-agent Based Inter-cloud Architecture (ECBS)

Enterprise Cloud Bus System (ECBS) [13, 14], describes a high-level abstraction layer of SaaS architecture in Inter-cloud environment, where different clouds interacts and collaborates through cloud agent from various locations in order to publish and/or subscribe their services. The detailed set of building blocks in the context of ECBS has been described as follows:

### *6.3.1 Building Blocks of ECBS*

This subsection describes briefly the building blocks of the ECBS:

(a) **Client**: Client is the end-users/actor in multi-cloud environment; here, the *CLIENT* placed the service request through Provider Agent (*PA*).
(b) **Provider Agent (PA)**: *PA* invokes the request from the *CLIENT* and schedules the required services.
(c) **Cloud Universal Description Discovery and Integration (CUDDI)**: *CUDDI* is the extended meta-service registry where *PA* published the client request.
(d) **Enterprise Service Bus (ESB)**: *ESB* is the Bus where the services are published.
(e) **Cloud Enterprise Service Bus (CESB)**: *CESB* is extension of *ESB* that enhance the ESB's to register their services for single cloud environment.
(f) **Cloud Agent (CA)**: *CA* is deployed for collecting various services from different cloud service providers based on various locations, context, etc.
(g) **Hierarchical Universal Description Discovery and Integration (HUDDI)**: *HUDDI* is the extended meta-service registry of *CESB's* in *ECB* where *CA* published the services.
(h) **Scheduling Agent (SA)**: *SA* is deployed in ECB to configure, discover, and schedule the cloud services as per Quality of Service (QoS) parameters.
(i) **MAPPER**: *MAPPER* is the one of the Cloud Bus component where service mapping is done as per Client request.
(j) **LOGGER**: *LOGGER* holds the mapped services before dispatching.
(k) **RES**: *RES* are the resources shared by the cloud bus.

### *6.3.2 Formalization of ECBS*

This chapter is the extensions of ESB's with formal approach toward analysis of dynamics of Inter-cloud architecture. The CloudBus (*CB)* is the set of agents and components (as refer in earlier section) of Enterprise Cloud Bus System. The structural representation of the CloudBus (*CB*) is defined as:

$$CB \rightarrow CLIENT \wedge PA \wedge CUDDI \wedge ESB \wedge CESB \wedge CA \wedge HUDDI \wedge SA \wedge MAPPER \wedge LOGGER \wedge RES \tag{6.1}$$

A multi-cloud environment *Multi-CloudEnv* is that where components of *CB* will work using the following four tuples. It can be defined as

$$Multi - CloudEnv = \{Res, Actor, CB, Relation\} \tag{6.2}$$

In the given cloud environment, *Res* is the set of cloud resource, *Actors* are clients of the cloud environment, *CB* are the set of autonomous cloud bus entities with prespecified functions and *Relation* is the set of semantic association and interactions among the cloud bus entities.

In the context of multi-cloud environment, the cloud bus (*CB*) will comprehend the occurrences of events automatically and response toward the environment with a set of cloud services. Moreover, any agent or component of *CB* acts on the cloud resources Res and is able to function over the web services provided by various cloud service providers. Further, enterprise cloud bus system (*ECBS*) can be defined as

*ECBS = {CB, COL, I},* where, *CB* is the set of Cloud Bus.

Thus, $\forall i,\ CB_i \in CB.\ COL_{ij}$ identifies a set of Collaborations among $CB_i$ and $CB_j$.

Thus, $\forall_{i,\ j},\ COL_{ij} \in COL,\ if\ i \neq j$. The set $I_{ij}$ determines the interaction path between any two Cloud Bus $CB_i$ and $CB_j$ in the *ECBS* system.

Thus, $\forall_{i,\ j},\ I_{ij} \in I,\ if\ i \neq j$.

### 6.3.3 Conceptualization of ECBS in MAS Architecture

Conceptual architecture of *ECBS* defines a set of building blocks and their inter relationship to conceptualize the environmental elements, agents, related events, collaborations, and interactions among the *CESBs*. A conceptual architecture of *ECBS* deals with high-level representation of the agent and other component in inter-cloud architecture.

This section describes the conceptual definition of multi-agent-based enterprise cloud Bus system (*ECBS*). The concept of *MAS* definition in the proposed architecture is an extension of research works in [10, 11]. Agent-based system is the de facto paradigm to handle the dynamicity of multi-cloud architecture like *ECBS*.

The dynamicity of $CB_i$ in the environment multi-CloudEnv are handled using three agents {*PA, SA, CA*} and other components relevant to single cloud architecture as described in Fig. 6.1.

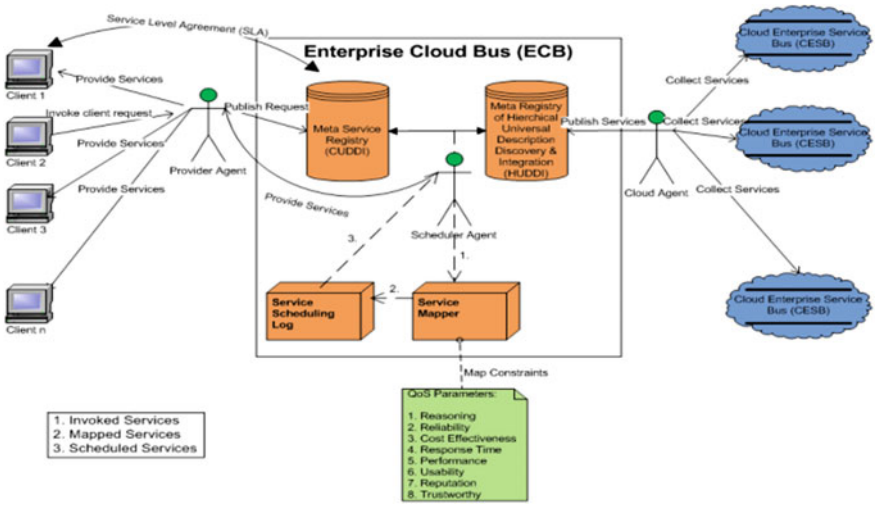



**Fig. 6.1** Enterprise Cloud Bus System Framework (ECBS)

Formally, the dynamic model of any CloudBus *($CB_i$)* can be defined as

$$CB_i = \{CA_i, PA_i, SA_i\} \tag{6.3}$$

Each of the agents within the CloudBus *($CB_i$)* will be invoked if the following conditions hold by different agents;

$ESB \wedge CESB \wedge HUDDI \wedge RES \rightarrow CA_i$

$CLIENT \wedge CUDDI \wedge RES \rightarrow PA_i$

$CUDDI \wedge HUDDI \wedge MAPPER \wedge LOGGER \wedge SCHEDULER \wedge RES \rightarrow SA_i$

Since, agents are the architectural basis of the ($CB_i$). Therefore, the dynamic model of the $CB_i$ is a multi-agent-based system and can be defined as a multi-agent definition as follows:

(a) $CB_i$. $A_i = [Role, E, C, R, PR, K, S, I]$ *where,* $A_i \in \{PA_i, SA_i, CA_i\}$.
(b) Each agent in the $CB_i$ plays a specific set of Roles in the environment Multi-CloudEnv; $E$ is the set of cloud events which occur during various states of the cloud service transitions.
(c) $C$ is a set of environmental conditions in cloud to be checked in order to response on some event in Cloud Bus;
(d) $R$ is a set of environmental cloud resources that are available and necessary for fulfillment of the goal of agents within the $CB_i$. Formally, $(R \subset Res)$;
(e) $PR$ is the properties of agents within the $CB_i$ which will hold the state of the cloud bus and also will maintain the state of the cloud resources $R$ on which the agents is acting;

(f) $K$ is the set of information that forms the main knowledge base. Initially, it comprises of the states of available cloud resources that the agents within the $CB_i$ will use to response on some event. The $K$ can be update dynamically.
(g) $S$ is the set of cloud services that the agents within the $CB_i$ can provide and conceptualize this will determine the capability of the cloud bus components;
(h) $I$ is a set of interactions between the agents reside inside the $CB_i$.

### 6.3.4 Structural Analysis of ECBS

The structural analysis of the *ECBS* system can be studied using Eqs. (6.1), (6.2), and (6.3) as described in the earlier section. The analysis states that, $CA_i$ exists if for all i, *ESB, CESB, HUDDI,* and *RES* components exist. This also implies that for any *CloudBus i*, any change in $CA_i$ will affect the state of *ESB, CESB, HUDDI,* and *RES* only. Similarly, $PA_i$ exists if for all *i*, *CLIENT, CUDDI*, and *RES* exist, which implies for any *CloudBus i*, any change in $PA_i$ will affect the state of *CLIENT, CUDDI,* and *RES* only. Similarly, $SA_i$ exists if for all *i, CUDDI, HUDDI, MAPPER, LOGGER, SCHEDULER,* and *RES* components exist. That implies, for any *CloudBus i*, change in $SA_i$ will affect the state of *CUDDI, HUDDI, MAPPER, LOGGER, SCHEDULER*, and *RES* only.

### 6.3.5 ECBS Elements in MAS Architecture

The roles, events and related services along with the respective resources, properties, and knowledge base of each agents present within the CBi is summarized in Table 6.1.

Provider Agent (*PA*) starts working with the minimal set of knowledge of the environment to render the request, service, and resource token.

The set of Roles $R$ as shown in Table 6.1 to be played by the *PA* will be

$R$ = *{R0: Request Transmitter, R1: Service Provider, R2: Request Provider, R3: Resource Seeker}.*

The set of Events $E$ for the *PA* will be

$E$ = *{E0: Request Transmitted, E1: Service Provided, E2: Request Registered, E3: Resource used & Released}.*

These set of events will be performed after satisfying possible environmental constraints C.

The set of Resources $RS$ = *{RS1: Web Service, RS2: Registries, RS3: Timestamp}.*

Now the *PA* will use several properties to hold the state of the resources and the states of itself.

**Table 6.1** Role collaboration templates of ECBS

| Role | Event | Service | Cloud bus component |
|---|---|---|---|
| *Agent: Provider Agent (PA)* | | | |
| R0: Request Transmitter<br>R1: Service Provider | E0: Request Transmitted<br>E1: Service Provided | S0: SendRequest<br>S1: ProvideService | CLIENT |
| R2: Request Provider | E2: Request Registered | S2: GetRequest | CUDDI |
| R3: Resource Seeker | E3: Resource used & Released | S3: GetResource<br>S4: ReleseResource | RES |
| *Agent: Cloud Agent (CA)* | | | |
| R4: Service Invoker | E4: Service Invoked | S5: InvokeService | ESB |
| R5: Service Collector | E5: Service Collected | S6: CollectService | CESB |
| R6: Service Transmitter | E6: Service Registered | S7: PublishService | HUDDI |
| R7: Resource Seeker | E7: Resource used & Released | S8: GetResource<br>S9: ReleseResource | RES |
| *Agent: Scheduler Agent (SA)* | | | |
| R8: Service Matcher | E8: Service Matched | S10: MatchService | CUDDI |
| R9: Service Seeker | E9: Service Discovered | S11:GetService | HUDDI |
| R10: Service Mapper | E10: Service Mapped | S12: MapService | MAPPER |
| R11: Service Scheduler | E11: Service Scheduled | S13: ScheduleService | SCHEDULER |
| R12: Service Logger | E12: Service Logged | S14: LogService | LOGGER |
| R13: Service Dispatcher | E13: Service Dispatched | S15: DispatchService | CLIENT |
| R14: Resource Seeker | E14: Resource used & Released | S16: GetResource<br>S17: ReleseResource | RES |

Hence, the set of Properties *PR = {PR1: status of Request type which can have various statuses, PR2: status of Service type which can have various statuses, PR3: status of Resource type which can have various statuses}.*

*PA* starts working with the minimal set of knowledge of the environment to render the services. The knowledge base can be updated dynamically once the component of CloudBus starts working.

The set of knowledge

*K = {K0: Details of request, K1: Details of service, K2: Details of resource}.*

With all these and with some defined set of Interactions I *PA* will be performing some services

*S = {S0: SendRequest, S1: ProvideService, S2: GetRequest, S3: GetResource, S4: ReleaseResource}.*

Cloud Agent (*CA*) starts working with the minimal set of knowledge of the environment to render the request, service, and resource token.

The set of Roles *R* as shown in Table 6.2 to be played by the *CA* will be

*R = {R4: Service Invoker, R5: Service Collector, R6: Service Transmitter, R7: Resource Seeker}.*

The set of events *E* for the *CA* will be

*E = {E4: Service Invoked, E5: Service Collected, E6: Service Registered, E7: Resource used & released}.*

These set of events will be performed after satisfying possible environmental constraints *C*.

The set of Resources *RS = {RS1: Web Service, RS2: Registries, RS3: Timestamp}.*

Now the *CA* will use several properties to hold the state of the resources and the states of itself.

Hence the set of Properties *PR = {PR4: status of Request type which can have various statuses, PR5: status of Service type which can have various statuses, PR6: status of Resource type which can have various statuses}.*

*CA* starts working with the minimal set of knowledge of the environment to render the services. The knowledge base can be updated dynamically once the component of CloudBus starts working.

The set of knowledge *K = {K3: Details of request, K4: Details of service, K5: Details of resource}.*

With all these and with some defined set of Interactions I, *CA* will be performing some services *S = {S5: InvokeService, S6: CollectService, S7: PublishService, S8: GetResource, S9: ReleaseResource}.*

Scheduling Agent (*SA*) starts working with the minimal set of knowledge of the environment to render the request, service, and resource token.

The set of Roles *R* as shown in Table 6.2 to be played by the *SA* will be *R = {R8: Service Matcher, R9: Service Seeker, R10: Service Mapper, R11: Service Scheduler, R12: Service Scheduler; R13: Service Logger; R14: Resource Seeker}.*

The set of Events *E* for the *SA* will be *E = {E8: Service Matched, E9: Service Discovered, E10: Service Mapped, E11: Service Scheduled, E12: Service Logged, E13: Service Dispatched E14: Resource used & Released}.*

**Table 6.2** Mapping from ECBS Conceptual Architecture to HECBP

| S. No. | Concepts in ECBS | Concept in HECBP |
|---|---|---|
| 1 | Properties, Knowledge, Cloud Services, Roles of (CLIENT, PA, CUDDI, ESB, CESB, CA, HUDDI, SA, MAPPER, LOGGER, RES) | Place, P |
| 2 | Events of (CLIENT, PA, CUDDI, ESB, CESB, CA, HUDDI, SA, MAPPER, LOGGER, RES) | Transitions, T |
| 3 | Collaborations among (PA, SA, CA) | Set of Arcs, N |
| 4 | Elements of (PA, SA, CA) | Color Function, $C_f$ |
| 5 | Constraints of (PA, SA, CA) | Guard Function, G |
| 6 | Interactions among (PA, SA, CA) | Arc Expression, Exp |
| 7 | Users | Initialization Function, I |

These set of events will be performed after satisfying possible environmental constraints *C*.

The set of Resources *RS = {RS1: Web Service, RS2: Registries, RS3: Timestamp}*.

Now the *SA* will use several properties to hold the state of the resources and the states of itself.

Hence the set of Properties *PR = {PR7: status of Request type which can have various statuses, PR8: status of Service type which can have various statuses, PR9: status of Resource type which can have various statuses}*.

*SA* starts working with the minimal set of knowledge of the environment to render the services. The knowledge base can be updated dynamically once the component of CloudBus starts working.

The set of knowledge *K = {K6: Details of request, K7: Details of service, K8: Details of resource}*. With all these and with some defined set of Interactions I *SA* will be performing some services *S = {S10: MatchService, S11: GetService, S12: MapService, S13: ScheduleService, S14: LogService, S15: DispatchService, S16: GetResource, S17: ReleaseResource}*.

## 6.4 High-Level Enterprise Cloud Bus Petri-Net (HECBP)

High-level enterprise cloud bus petri-nets (*HECBP)* is a graphical representation of *ECBS* system that allows visualization and analysis of the system dynamics and behavioral properties such as safeness, boundedness and liveliness, etc. Proposed *HECBP* is a colored petri-net (*CPN*)-based approach, which is capable to represent the interactions between agents and other cloud components within the cloud bus.

### *6.4.1 Definition: High-Level Enterprise Cloud Bus Petri-Net (HECBP)*

A high-level Petri-net is defined as a directed bipartite graph that has two types of nodes namely places and transitions). The arcs are connector between these nodes that represents state of a transition of the given node. Hence in a formal manner, high-level enterprise cloud bus Petri-net, *HECBP*, is defined by the 8—tuples as follows:

(a) The various elements of the proposed *HECBP* is defined as
*Σ = [Color set for Request token, Color set for Service token, Color set for Resource token]*. $C_f$ *is the color function where,* $C_{re}$ *= {blue for request},* $C_s$ *= {red for service},* $C_r$ *= {black for resource token}*.

(b) $\Sigma$ is a finite set of non-empty types, also called color sets. The set of types determines the data values of CloudBus components, resources, the operations, and functions that can be used in the net expressions (i.e., arc expressions, guards, and initialization expressions), $\Sigma = . \ C_s \cup C_{re} \cup C_r \cup G \cup E \cup I;$

(c) $P$ is a non-empty finite set of places. It comprises of all the CloudBus and their environmental resources. Formally, $P = CB_i \cup Res$

(d) The *CloudBus* place contains all the agents, components, tokens, except events, of a *CloudBus*. $T$ is a non-empty finite set of transitions include all events of any CloudBus, $CB_i$, and resource $R_i$, along with the interactions between the cloud bus present in environment,

(e) $T = CB_i \cup I \cup Ri{*}e_i$. $N$ is the finite set of arcs that map into a pair where the first element is the source node and the second the destination node. The two nodes have to be of different kind. If we say that $T = e\ U\ I$, then it can be said that, $T \times P\ (CB_i)\ U\ (P \times T)$, because arc from $T$ to $P$ is not valid in case of resources. Hence,

(f) $C_f$ is the color function. Formally, $C_f \rightarrow C_b$. The color function $C_f$ maps each place $P$ to a type $C$. $C$ is the color function for CloudBus that contains various tokens of different colors. $C_b = C_s \cup C_{re} \cup C_r$, $C_s$ is the color function for Service token, $C_{re}$, is the color function for Request token, and $Cr$ is the color function for resources. Different tokens have different colors in the Net.

(g) The guard function $G$ maps each transition, $T$ into a Boolean expression where all variables have types that belongs to $\Sigma$. The arc expression function $E$ maps each arc 'a' in the node function into an expression of type $C_f\,(p)$.This means that each arc expression must evaluate to multi-set over the type of the adjacent place, $P$. The initialization function $I$ map each place, $P$, into a closed expression which must be of type $C_f\,(p)$.

An agent within the cloud bus of *ECBS* consists of various elements namely roles, events, constraints, resources, properties, knowledge, interactions, and services which together make *ECBS* successful to achieve the prespecified goal. Mapping of conceptual architectural to *HECBP* has been summarized in Table 6.2. A component will request for a resource. Once a resource is allocated to a component, it will hold the resource until the next transition is fired from that place.

Formally, $CB_i \rightarrow RES$. The graphical notation of place and transition are represented as usual notation of *CPN* and those are Circle and Bar, respectively.

### 6.4.2 HECBP Elements: Places and Transitions

The details of the places $P$, transitions $T$, and tokens t have been illustrated in Tables 6.3 and 6.4, respectively. In this Petri-net model, three types of tokens are considered. They are service, request, and resource token. In Table 6.4, the color set value of token is considered as ($P = 1;\ Q = 2;\ R = 3$) to distinguish among themselves.

## 6.5 Analysis of ECBS Based on HECBP

High-level enterprise cloud bus Petri-net (*HECBP*) is a suitable tool to model the behavior of *ECBS* system. Moreover, several features of dynamic system like, occurrence of finite number of events, deadlock free operations, achievement of goals through firing of events, etc., can be analyzed through the analysis of *HECBP* properties like, safeness, boundedness, liveness, reachability, etc. Further, the *HECBP*-based analysis will give detail insight about the internal behavior of the system.

### *6.5.1 HECBP-Based Analysis of ECBS*

Figure 6.2 shows the *HECBP* net of the *ECBS* system.

The process starts from a place *P0* which is the client and after a transition *T0* will reach a place *P10* from, which the scheduling agent of place *P7* will collect the service for delivering it to the client. The process continues further on and we finally arrive at the place *P7.* Serially as the transitions occur, the process moves on to each of the places as explained in the tables.

The place *P11* is the places for the resources *RS1, RS2,* and *RS3,* respectively. All the cloud bus components will request each of the resources as and when required and once the transition is fired will release it updating the knowledge base. In this system, a place have token such that $Token \rightarrow C \times K \times PR \times S \times R \times I$. From the *HECBP* net, the corresponding reachability graph is obtained and shown in Fig. 6.3.

Some of the crucial behavioral properties have been analyzed using the *HECBP* model. They are as follows:

**Table 6.3** Places and transitions with its descriptions based on Table 6.1

| Places | Component of places | Transitions | Events |
|---|---|---|---|
| P0 | Client | T0 | E0, E3 |
| P1 | PA | T1 | E2, E3 |
| P2 | CUDDI | T2 | E4, E7 |
| P3 | ESB | T3 | E5, E7 |
| P4 | CESB | T4 | E6, E7 |
| P5 | CA | T5 | E8, E14 |
| P6 | HUDDI | T6 | E9, E14 |
| P7 | SA | T7 | E10, E14 |
| P8 | MAPPER | T8 | E11, E14 |
| P9 | SCHEDULER | T9 | E12, E14 |
| P10 | LOGGER | T10 | E13, E14 |
| P11 | RESOURCE (RS1, RS2, RS3)<br>RS1: Web Services; RS2: Registries; RS3: Timestamp | T11 | E1, E14 |

**Table 6.4** Token and its descriptions

| Places | Token | Description of tokens | token parameters | Color set value of token |
|---|---|---|---|---|
| P0 | t0 | Request | Sent | 1 |
| P1 | t0 | Request | Provide | 1 |
| P2 | t0<br>t1 | Request<br>Service | Register<br>Register | 1<br>2 |
| P3 | t1 | Service | Published | 2 |
| P4 | t1 | Service | Provide | 2 |
| P5 | t1 | Service | Collect | 2 |
| P6 | t1 | Service | Register | 2 |
| P7 | t1<br>t1 | Service<br>Service | Register<br>Dispatch | 2<br>2 |
| P8 | t1 | Service | Discover | 2 |
| P9 | t1 | Service | Map | 2 |
| P10 | t1 | Service | Schedule | 2 |
| P11 | t2<br>t2 | Resource<br>Resource | Request<br>Release | 3<br>3 |

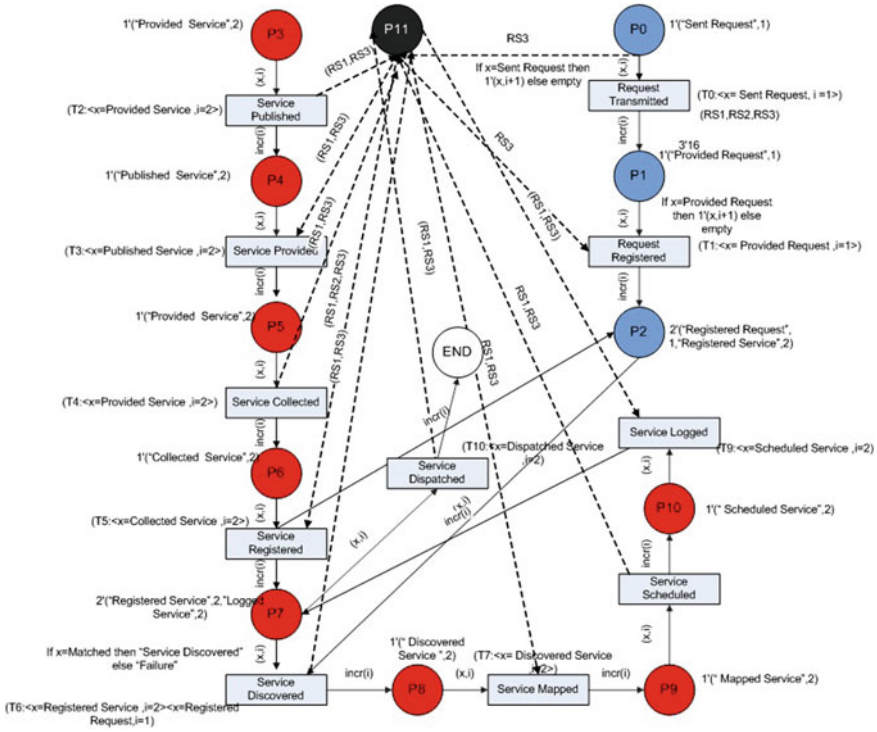
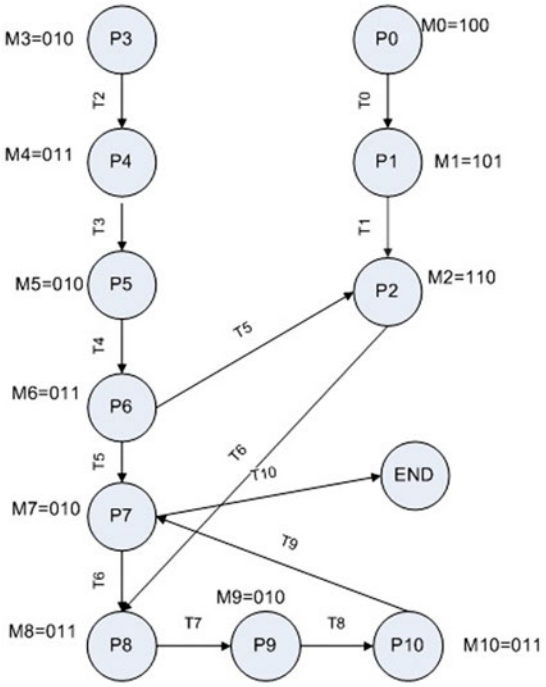



**Fig. 6.2** Enterprise Cloud Bus Petri-net (HECBP)

(a) **Reachability**: Reachability property is a fundamental artifact for analyzing the dynamic properties of any *MAS*-based cloud architecture. However in this section, it has been established from Fig. 6.3 that all the markings in the *HECBP* net are reachable starting from any marking in the net, and hence reachability exists. This guarantee that the *HECBP* net modeled the *ECBS* that will meet the prespecified goal;

(b) **Home Properties**: A marking in the *HECBP* is said to be a home marking if it is a home space. It tells us about the markings to which it is possible to return. In the proposed *HECBP,* the initial marking *M0* is a Home marking and a marking $M0 \in M$ and a set of markings $Z \subseteq M$ be given as: *M0* is a home marking
if: $\forall M' \in [M0] > : M \in [M']$*; and Z is a home space if:* $\forall M' \in [M0] > : Z \cap [M'] > \neq \varnothing]$. In this context, *M* is a home marking if $\{M\}$ is a home space.

(c) **Boundedness**: The boundedness property states that after considering all reachable markings, the number, and type of tokens a place may hold in the net. It can be concluded after analyzing the *HECBP* net that there is no unboundedness at any stage once the process starts and goes from place *P0* to *P7* via *P10,* and thus the boundedness of the *HECBP* net is guaranteed and safe;




| M (Weight) | P (Request) | Q (Service) | R (Resource) |
|---|---|---|---|
| M0 | 1 | 0 | 0 |
| M1 | 1 | 0 | 1 |
| M2 | 1 | 1 | 0 |
| M3 | 0 | 1 | 0 |
| M4 | 0 | 1 | 1 |
| M5 | 0 | 1 | 0 |
| M6 | 0 | 1 | 1 |
| M7 | 0 | 1 | 0 |
| M8 | 0 | 1 | 1 |
| M9 | 0 | 1 | 0 |
| M10 | 0 | 1 | 1 |

**Fig. 6.3** Reachability graph corresponding to Fig. 6.2

(d) **Liveness**: The liveness properties of a *HECBP* model shows a continuous dynamic operation of the proposed net model and ensure that the system is live once transitions are fired. In the proposed *HECBP* net as the component of the cloud bus process starts from *P0* transitions *T0* through *T10* are fired and place *P7* is reached. The net is continuous and hence the liveness property is ensured. Thus the proposed net is live:
(e) **Fairness**: The net *HECBP* is said to be bounded-fairness because at a time single transition are fired. It can also be termed as unconditional fairness because every transition appears infinitely in a firing sequence. Here the net is *B-Fair*.
(f) **Safeness**: Any place in a *HECBP* is declared as safe, as the number of tokens at that place is either 0 or 1 and there is no deadlock present in the net. Also all the place in the concerned net is safe therefore the net as a whole is declared safe.

### *6.5.2 Simulation of HECBP*

There are various tools to analyze Petri-net-based system behavior. In this section, *HECBP* Net is analyzed using *CPN* tools to study the behavioral aspects of multi-cloud architecture defined using proposed conceptual model of *ECBS*. Here, three colors sets namely red, blue, and black have been used to depict the three types of tokens namely request, service and resource token, respectively.

Thus few restrictions have been imposed for simulation of the *HECBP* Net using *CPN* simulation and are summarized as follows:

(a) Before and after Transitions the data types of Tokens have to be same or else transition will not be fired/enabled;
(b) During the simulation process, at any point of time, it cannot be clearly expressed which component is enabled after a Transition is fired;
(c) Multitoken pass can be done but only one single token is being passed at a time.

The advantage of using the concept of high-level Petri-net along with CPN simulator tool for analyzing dynamism of multi-cloud behavior is as follows:

(a) The dynamic component (*PA, CA, SA*) can be handled with ease.
(b) Validation of the agent-based multi-cloud architecture can be done.

The declarations for the generated *HECBP* net corresponding to Fig. 6.2 can be expressed as:

```
colset BOOL=bool;
colset INTINF = intinf;
colset TIME = time;
colset REAL = real;
colset UNIT = unit timed;
```

```
colset STR = with S timed;
var x: STR;
colset IN = with P | Q | R timed;
var i: IN;
colset PRO = product STR * IN timed;
var p: PRO;
```

### 6.5.3 HECBP Simulation Through CPN Tool

In this section, the *HECBP* net is simulated using *CPN* tool and the corresponding simulation results before and after transitions are shown. Figure 6.4 shows the simulation before resource is allocated to the place *P0*. Once the requested resources are allocated to the place *P0,* the relevant transition *T0* is fired and place *P1* is reached. The resources are held up by the place *P1*. They will be returned to the resource pool and only then the next transition will be enabled.

Figure 6.5 shows the simulation after resource is allocated to the required place. Once these resources are released, as shown in Fig. 6.6, next transition *T1* will be enabled and place *P2* will be reached. Hence, it can be seen that the resources are allocated and released dynamically. It is dynamicity of the cloud bus component properties that is very smoothly analyzed with the help of the proposed model.

This process of resource allocation and returning back after use will further continue and finally the place P10 is reached from which the token moves to the P7 from where the service are delivered. Once the simulation is executed, the place P0 will send a request signal for the resources RS3. The request signal is accepted. It can be seen that the net is waiting for transition T0 to be enabled.

Once the resources are allocated it can be seen in Fig. 6.4 that transition T0 is fired. Place P1 is reached and resources are now within place P1. The next transition T1 will be enabled only when the held up resources are freed or released. Further in Fig. 6.7 the service are discovered once the request token in *CUDDI* are matched with the service token by the *SA*. Thereafter, service is dispatched and resources are released.

### 6.5.4 State Space Analysis

Simulation conducts a finite number of executions of the model that is being analyzed. On conducting state-space analysis of the model, the simulation tool usually generates a state-space report that provides the details of the state space and standard behavioral properties of the proposed net. The report also gives a clear idea about the beat upper and lower bounds. After simulation of the HECBP net, the corresponding state-space reports are shown.

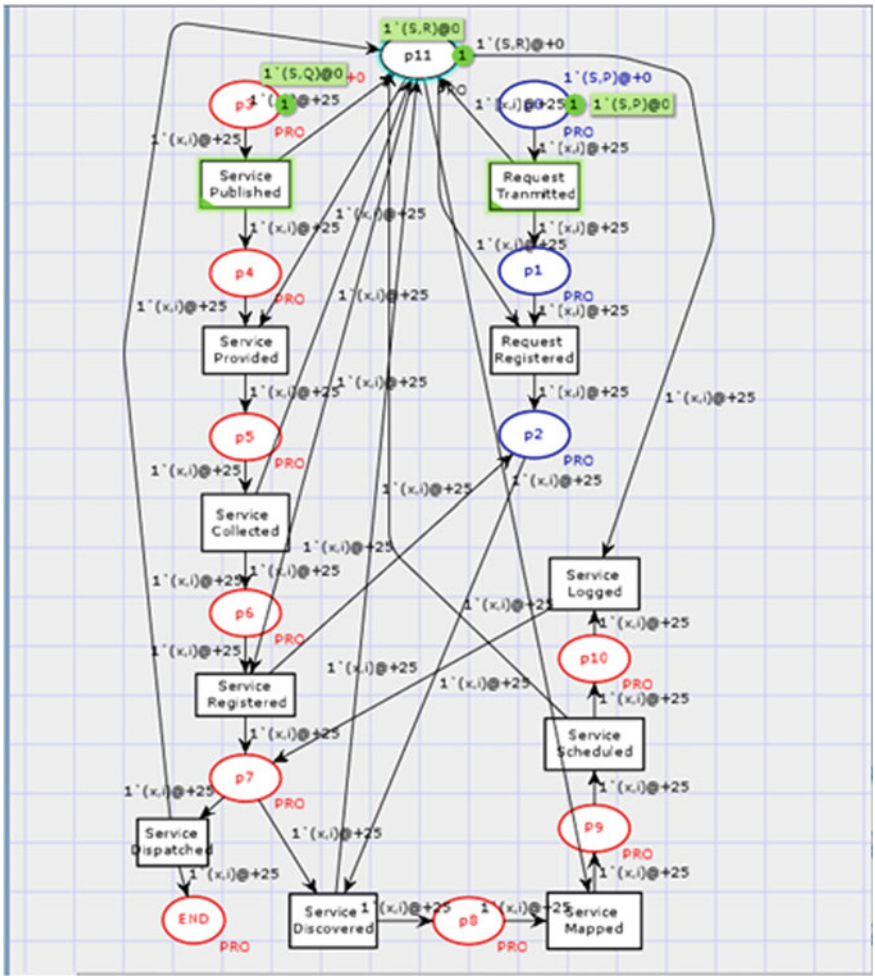



**Fig. 6.4** CPN simulation—before resource allocation

The state-space report (Fig. 6.8) gives the state-space statistics. The boundedness properties are shown in Fig. 6.9 that tells the number of tokens a place may hold after considering all reachable markings. The best upper and lower bounds results are shown in Fig. 6.10. These reports exhibit that the proposed HECBP is bounded and safe.

The next part of the state-space report, Fig. 6.11, specifies the home properties, liveness properties, and fairness properties. In Home properties, the home marking is node *P12*. Even in liveness property node *P12* is regarded as dead because the execution of the process ends at that node namely *P12* from which scheduling agent (*P7*) takes the services and dispatch it to the end-users.

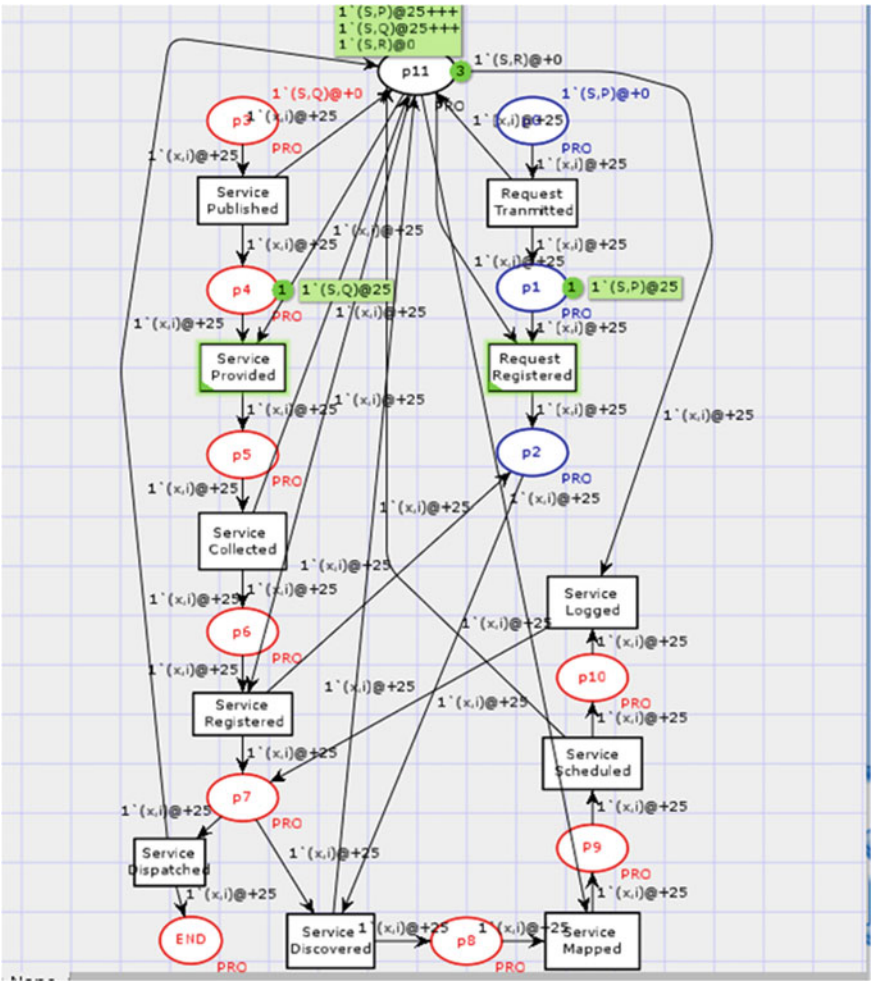



**Fig. 6.5** CPN simulation—after resource allocation

It is observed that the *HECBP* net is live because there are no dead transitions. Similarly, we proved that the fairness property also is true since there is no infinite occurrence sequence in the net.

In the theoretical analysis, a discussion was made on *ECBS* and further using formalized and conceptual definition of the proposed ECBS a corresponding *HECBP* model and its reachability graph was obtained and through which basic dynamic behavioral properties like liveness, safeness, and boundedness were proved.

It was observed and established from the theoretical analysis that all the dynamic properties of *ECBS* were true for the *ECBS* considered as discussion. CPN tool was

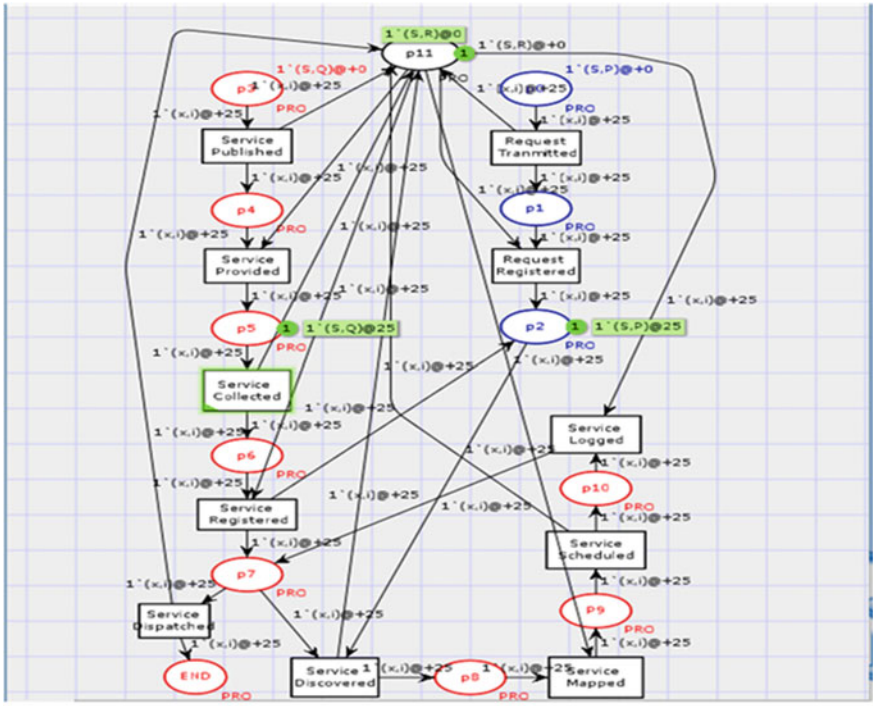



**Fig. 6.6** CPN simulation—resource used and released

used to successfully simulate and design the *HECBP* model. Once the *HECBP* is simulated successfully, the state-space analysis generates the state-space reports, which show that all the dynamic behavioral properties of the *HECBP* are successfully demonstrated and verified. Thus, the simulation results of the *HECBP* model strongly validate the theoretical analysis.

## 6.6 Future Research Directions

With the advancement of requirements engineering for service and cloud computing technology, the enterprise software application has become a prevailing domain focusing on integration and automatic composition of web services over distributed cloud computing environment. Various potential research works still exist for the field of requirements engineering for service and cloud computing system design. In this context, several research proposals are there in literatures for meeting the increasing demands for distributed software as a service and making the software more accessible, scalable, configurable (over a distributed large-scale global network), and shareable. Several researchers have work on conceptual model

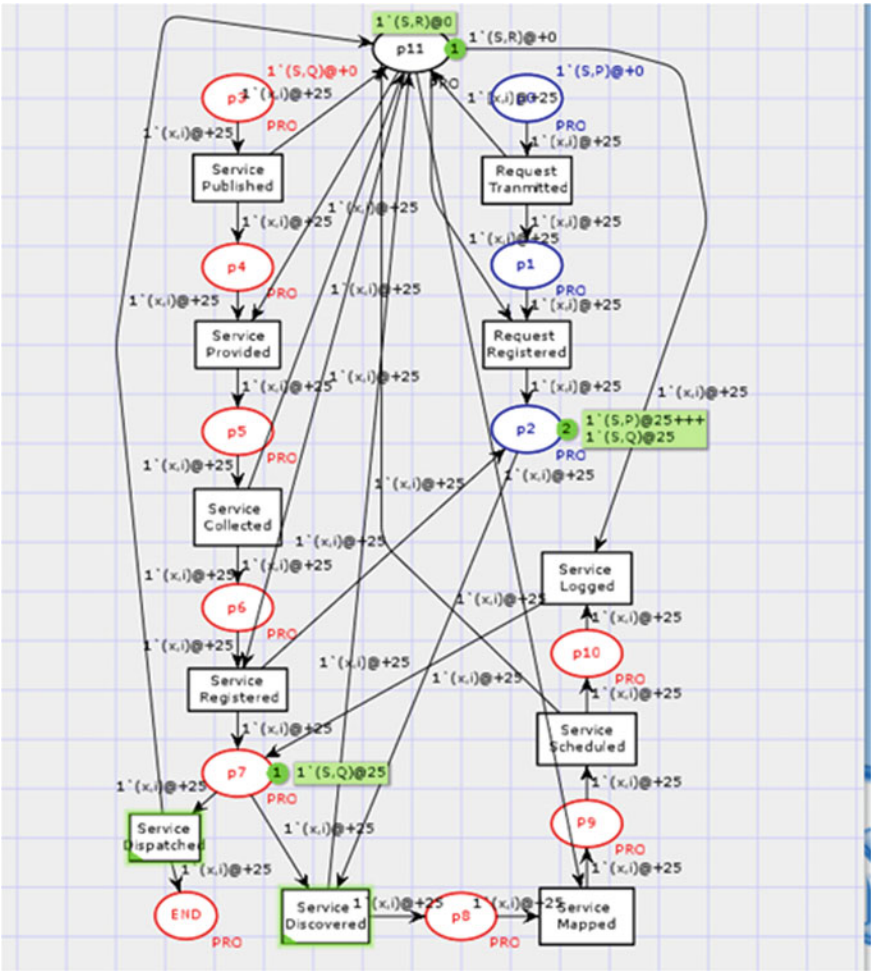



**Fig. 6.7** CPN simulation—service discovered by SA

of multi-cloud architecture in order to address the problem of interoperability, dynamic provisioning, deployment, and adaptation of dynamic multi-cloud behavior systems.

Many of those approaches [24, 25] are also included with MAS based mechanism in inter-cloud architecture that exhibits the dynamism of internal behavior of the system. To handle the dynamicity of such large-scale computing system, Petri-net-based approach [26] are a most acceptable formal modeling tool nowadays. However, none of those proposals are still accepted as standard. Most of those proposals [27–29] are varied in the modeling and analyzing the dynamic behavior of the system. So, more researches are required toward the conceptual level

```
CPN Tools state space report for:
/cygdrive/D/Pendrivecloud
computing/Paper/Paper
Communicated/CPN/cp/CPNM@02-07-
14/ECB@cpn.cpn
Report generated: Wed Sep 30 14:15:38 2015
 Statistics
------------------------------------------
------------------------------

  State Space
     Nodes:  12
     Arcs:   11
     Secs:   0
     Status: Full

  Scc Graph
     Nodes:  12
     Arcs:   11
     Secs:   0
```

**Fig. 6.8** State space report

```
Boundedness Properties
------------------------------------------------------------------------

 Best Integer Bounds
                          Upper      Lower
   New_Page'END 1         1          0
   New_Page'P9 1          1          0
   New_Page'p0 1          1          0
   New_Page'p10 1         1          0
   New_Page'p1 1          1          0
   New_Page'p11 1         3          1
   New_Page'p2 1          2          0
   New_Page'p3 1          1          0
   New_Page'p4 1          1          0
   New_Page'p5 1          1          0
   New_Page'p6 1          1          0
   New_Page'p7 1          1          0
   New_Page'p8 1          1          0
```

**Fig. 6.9** Boundedness properties

modeling using high-level Petri-net-based approach of MAS-based multi-cloud architecture with the aim of realizing the facets of such system more comprehensively.

Besides this, several other research directions are as follows:

(a) **Modeling and Analysis of Service Composition Pattern through HECBP Net**: Multi-agent system [24–27] based Inter-cloud architecture [33–36] requires formal modeling and analysis of service selection and composition

```
Best Upper Multi-set Bounds
      New_Page'END 1          1`(S,Q)
      New_Page'P9 1           1`(S,Q)
      New_Page'p0 1           1`(S,P)
      New_Page'p10 1          1`(S,Q)
      New_Page'p1 1           1`(S,P)
      New_Page'p11 1          1`(S,P)++
1`(S,Q)++
1`(S,R)
      New_Page'p2 1           1`(S,P)++
1`(S,Q)
      New_Page'p3 1           1`(S,Q)
      New_Page'p4 1           1`(S,Q)
      New_Page'p5 1           1`(S,Q)
      New_Page'p6 1           1`(S,Q)
      New_Page'p7 1           1`(S,Q)
      New_Page'p8 1           1`(S,Q)

  Best Lower Multi-set Bounds
      New_Page'END 1          empty
      New_Page'P9 1           empty
      New_Page'p0 1           empty
      New_Page'p10 1          empty
      New_Page'p1 1           empty
      New_Page'p11 1          1`(S,R)
      New_Page'p2 1           empty
      New_Page'p3 1           empty
      New_Page'p4 1           empty
      New_Page'p5 1           empty
      New_Page'p6 1           empty
      New_Page'p7 1           empty
      New_Page'p8 1           empty
```

**Fig. 6.10** Upper and lower bounds

```
Home Properties
------------------------------------------------------------------------

 Home Markings
    [12]
Liveness Properties
------------------------------------------------------------------------

 Dead Markings
    [12]
 Dead Transition Instances
    None

 Live Transition Instances
    None

Fairness Properties
------------------------------------------------------------------------
    No infinite occurrence sequences.
```

**Fig. 6.11** State space report of other behavioral properties

pattern. Verification and validation of service composition in Inter-cloud architecture using high-level Petri-net is shown in [30–32]. The Petri-net-based approach of modeling and analysis of multi-agent system is addressed in [37–42]. But the proposed approach is less expressive when compared to the high-level Petri-net-based approach, which is applicable across majorly on large complex system. Moreover, many researchers have expressed the modeling and analysis using high-level Petri-net approach in [43–45]. Many of the research work are done using color Petri-net (CPN) tool that is further used for constructing and analyzing such multi-cloud system [46]. The work in [47, 48] reveals about the behavioral analysis of multi-cloud architecture using colored Petri-net. However, it does not have any formal background of service selection and composition design pattern. However, both of these proposals lack from the validation with the standard like high-level Petri-net. On the other hand, [49] is a formal methods of integrating high-level Petri-net with Z notation. Both [48, 49] are rigorously formal modeling techniques.

The *HECBP* model can be taken as a standard modeling, to select, compose, identify, and schedule the services during its run time. Service selection and composition modeling of such system is one of the challenging research issues. Compare to the existing methodologies of such modeling, very few methodologies are there based on high-level Petri-net-based approach. Using *HECBP* approach is one of the most prevailing modeling techniques to identify, define, visualize, and specify customer's requirements for formal modeling of service composition pattern. The approach will help to cater the state-of-the-art research and practice relating to requirements engineering for service and cloud computing.

More research directions are required toward the tools and techniques for customer's service requirement engineering and modeling and analysis of service composition pattern in the latest research field. The *HECBP* approach helps to cater the functional requirements of service composition for the needs of the industry. The analysis of the Service Composition Framework can be done based on the degree of heterogeneity of the cloud services. Further, the service composition analysis can be proposed under various categories like service choreography, orchestration, and hybrid.

(b) **Verification and Validation of the Service Composition Pattern using HECBP Net**: Few of the proposed models [50, 51] validate the composition pattern through high-level Petri-net-based approach using tools like CPN. The author in the papers [52, 53] has expressed the verification strategy for web services composition using enhanced stacked automata model. Thus for validation of the service composition pattern of any cloud-based system, the *HECBP* model can be used to validate properties like correctness and completeness of the system pattern.

Further, the verification and validation can be done by comparing the simulation result of *HECBP* model with the theoretical results. Large research

initiative is required toward the validation of the composition pattern in cloud computing environment using high-level Petri-net.

(c) **Service Scheduling using HECBP Net based on QoS parameters**: Very few proposals like [54–56] supported the modeling of service scheduling on quality of service (*QoS*) parameters using high-level Petri-net based approach. Moreover, those proposals also are at cognitive levels. Further, researches are required toward the proposal of scheduling approach using high-level Petri-net model for further validation. The *HECBP* Net can be used for scheduling of cloud services as per user's requirement. This approach helps to identify the functional requirement and enhance the requirement engineering for service and cloud computing paradigm.

(d) **Quality Evaluations of HECBP model**: It is necessary to evaluate the quality metrics of the *HECBP* model of Inter-cloud Architecture. According to research work [57–59], the quality evaluation techniques along with metrics, such as operability, performance, scalability, reliability, etc., are measured based on customer's requirement toward service in cloud computing domain. These concepts are abstracts and cannot be measured directly, but the evaluation of quality metrics at early design phase is important to make the cloud bus system more robust and reliable in service requirements engineering field.

In view of this, there is a necessity of a set of objective quality measurements at the conceptual level design phase of such system to assess the quality metrics in terms of reusability. The scalability factor of the *HECBP* model is another issue that also influences the quality of early design of such system. Thus research effort is required to devise suitable framework for quality evaluation of *MAS*-based Inter-cloud architecture.

## 6.7 Conclusion

Multi-agent-based Inter-cloud Architecture represents dynamic and complex system that consists of heterogeneous autonomous entities (*CA, PA, SA*) and other components present inside the cloud bus. These autonomous entities play some specific roles in the system. Based on these roles, collaborations occur between the participating agents and components in *ECBS*. Participating agents in *ECBS* are proactive and thus interact with the multi-cloud environment or with some other components in the system.

Thus, collaborations and interactions among the participating agents and components are the key factors to design the dynamics of *ECBS* effectively. As a result of this dynamicity of *ECBS*, there are various behavioral properties exist in the *ECBS*. For the purpose, a high-level enterprise cloud bus Petri-net (*HECBP*) has been proposed to analyze and model the behavioral aspects of *ECBS* using a tool called *CPN* for service requirements engineering. The dynamicity of the proposed enterprise cloud bus system benefits the enterprise applications in optimizing the

performance, cost, elasticity, flexibility, high reliability, and availability of the computing resource.

Further, a set of mapping rules have been described for representing the elements of the proposed conceptual framework of the *ECBS* into the *HECBP* net. The proposed requirements engineering methods capture the state-of-the-art research and practice relating to requirements engineering for service and cloud computing. The key benefits of using the proposed mechanism are the capability to represent study and analyze the interactions among the multiple agents present inside the Cloud Bus. Using *HECBP* concepts and corresponding reachability graph, the behavioral properties of an *ECBS* like reachability, safeness, Boundedness, liveness can be analyzed formally. Moreover, simulation of the proposed *HECBP* with *CPN* tool and generated results strongly validate the proposed claims. Interested readers may refer the Additional References for further concepts on cloud-based system and service-oriented system

Future work includes the quality analysis of *ECBS* dynamics from the proposed concepts of *HECBP*. Development of a dedicated simulation tool for the conceptual architecture of *ECBS* and *HECBP* is also a prime future objective. Simulation with *CPN* tool and generated state-space reports will strongly validate the said claims. We hope that this chapter will provide a comprehensive source of information regarding formalization of *MAS*-based inter-cloud architecture and study its behavioral properties.

## References

1. Alexandros, K., Aggelos, G., Vassilis, S., Lampros K., Magdalinos P., Antoniou E., Politopoulou Z., 2014, A cloud-based Farm Management System: Architecture and implementation, In Journal Computers and Electronics in Agriculture, Vol no. 100, pp. 168–179.
2. Ardagna, D., Nitto, E.D., Casale, E., P, P., Mohagheghi, S., Mosser, P., Matthews, A., Gericke, C., Balligny, F. D., Nechifor, C.S, C, Sheridan., 2012. MODACLOUDS, A Model-Driven Approach for the Design and Execution of Applications on Multiple Clouds, International Workshop on Modelling in Software Engineering. pp. 50–56.
3. Brandtzæg, E., M. Parastoo, Mosser, E., 2012. Towards a Domain-Specific Language to Deploy Applications in the Clouds, 3rd International Conference on Cloud Computing, GRIDs, and Virtualization. IARIA, pp. 213–218.
4. Cavalcante, E., 2013. Architecture Driven Engineering of Cloud-Based Applications, IFIP/Springer-Verlag, Germany, Vol no. (22). pp. 175–180.
5. Divyakant, A., Abbadi, A., Das, S., Elmore, A.J., 2011. Database scalability, elasticity, and autonomy in the cloud, In Journal of Database Systems for Advanced Applications, Vol no. (2), pp. 2–15.
6. Buyya, R., Ranjan, R., Rodrigo, N., 2010. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services, In Algorithms and architectures for parallel processing, LNCS Springer Vol no. (6081). pp. 13–31.

7. Elmroth, E., Tordsson, J., Hernández, F., Ali-Eldin, A., Pette R, 2011. Self-management challenges for multi-cloud architectures, Lecture Notes in Computer Science Towards a Service-Based Internet, Vol no. 6994, pp. 38–49.
8. Ferry, N., Alessandro Rossini, Franck Chauvel, Brice Morin, and Arnor Solberg, 2013. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems, In 2013 IEEE Sixth International Conference on cloud computing, pp. 887–894.
9. Saggar, R., Saggar, S., Khurana, N., 2014. Cloud Computing: Designing Different System Architecture Depending On Real-World Examples, International Journal of Computer Science and Information Technologies, Vol. 5 (4), pp. 5025–5029.
10. Sarkar, A., Debnath, N.C., 2012. Measuring Complexity of Multi- Agent System Architecture, 10th IEEE Conference on Industrial Informatics, pp. 998 – 1003.
11. Sarkar, A., 2013. Modeling Multi- agent system dynamics: Graph semantic based approach, 10th International Conference on Service Systems and Service Management. pp. 664–669.
12. Djamel, B., 2013. An agent-based approach for hybrid multi-cloud applications, In Journal of Scalable Computing: Practice and Experience 14, Vol no. 2.pp. 95– 109.
13. Khan, G., Sengupta, S., Sarkar, A., Debnath, N.C., 2014. Modeling of Inter-Cloud Architecture using UML 2.0: Multi- agent Abstraction based Approach, 23rd International Conference on Software Engineering and Data Engineering, pp 149–154.
14. Khan, G., Sengupta, S., Sarkar, A., 2014. WSRM: A Relational Model for Web Service Discovery in Enterprise Cloud Bus (ECB), 3rd International Conference on Eco-friendly Computing and Communication System, India, pp. 117–122.
15. Khan, G., Sengupta, S., Sarkar, A., 2015. Modelling of Services and their Collaboration in Enterprise Cloud Bus (ECB) using UML 2.0, 2015. International Conference on Advances in Computer Engineering and Applications, India, pp. 207–213.
16. Khan, G., Sengupta, S., Sarkar, A., Debnath, N.C., 2015. Web Service Discovery in Enterprise Cloud Bus Framework: T Vector Based Model, 13th IEEE International Conference on Industrial Informatics, pp. 1672–1677.
17. Sofiane, B., Bendoukha, H., Moldt, H., 2015. ICNETS: Towards Designing Inter-Cloud Workflow Management Systems by Petri-nets, In Enterprise and Organizational Modeling and Simulation, 198. Springer International Publishing. pp. 187– 198.
18. Chatterjee, A.K., Sarkar, A., Bhattacharya, S., 2011. Modeling and Analysis of Agent Oriented System: Petri-net Based Approach, 11th Intl. Conf. on Software Engineering Research and Practice (SERP 11), Vol. 1, PP 17 – 23.
19. Khan, G., Sengupta, S., Sarkar, A., 2015. Modeling and Analysis of Enterprise Cloud Bus using a Petri-net Based Approach, 3rd International Doctoral Symposium on Applied Computation and Security Systems (ACSS 2016), Kolkata, India.
20. Jensen, K., Kristensen, L.M., Wells, L.M., 2007. Coloured Petri-nets and CPN tools for modeling and validation of concurrent systems, International Journal on Software Tools for Technology Transfer, Vol no. 05. pp. 213 – 254.
21. Bhuvaneswari, A., Uma, S., Sakthitharan, S., Srinivasan, G., 2014. Assessment of Service Composition Plan using Colored Petri-nets, International Journal of Engineering And Computer Science, Vol no. 3(1), pp. 3736 – 3742.
22. Fitch, D., Xu, H., 2013. A Raid-Based Secure And Fault-Tolerant Model for Cloud Information Storage, International Journal of Software Engineering and Knowledge Engineering 23, Vol no. 05, pp. 627 – 654.
23. Chatterjee, R., Neha, Sarkar, A., 2015. Behavioral Modelling of Multi-agent System: High Level Petri-net Based Approach, International Journal of Agent Technologies and Systems, Vol no. 7(1), pp. 55 – 78.

## Additional References


24. Zambonelli, F., Omicini, A., (2004). Challenges and Research Directions in Agent-Oriented Software Engineering, Journal of Autonomous Agents and Multi- agent Systems, Vol. 9, PP 253–283.
25. Bauer, B., Mulller, J. P., Odell, J., (2001). Agent UML: A Formalism for Specifying Multiagent Software Systems, International Journal of Software Engineering and Knowledge Engineering, Vol 11, No. 3, pp. 1– 24.
26. Far, B. H., Wanyama,T., (2003).Metrics for agent-based software development, Canadian Conference on Electrical and Computer Engineering (IEEE CCECE 2003), Volume 2, PP 1297–1300.
27. Wille, C., Brehmer, N., Dumke, R.R., (2004). Software measurement of agent based systems - an evaluation study of the agent academy, Technical Report Preprint No. 3, Faculty of Informatics, University of Magdeburg.
28. G´omes-Sanz, J. J., Pav´on, J., Garijo, F., (2005). Estimating cost for agent oriented software", In M¨uller, J. and Zambonelli, F., editors, Agent oriented software engineering V. 5th International Workshop, AOSE 2005, Utrecht, The Netherlands, July 2005, Revised Selected Papers, number 3950 in LNCS, pages 218–230, 2006.
29. Klügl, F., (2008). Measuring Complexity of Multi- agent Simulations – an Attempt using Metrics", Booktitle: Languages, Methodologies and Development Tools for Multi- agent Systems, Springer-Verlag Berlin, Heidelberg.
30. Dhavachelvan, P., Saravanan, S., Satheskumar, K., (2008).Validation of Complexity Metrics of Agent-Based Systems Using Weyuker's Axioms, International Conference on Information Technology (ICIT '08), PP 248– 251, 2008.
31. Mala, M., Cil, I., (2011). A taxonomy for measuring complexity in agent based systems, IEEE 2nd International Conference on Software Engineering and Service Science (ICSESS'11), pp. 851–854.
32. Cetnarowicz, K., Cetnarowicz, E., (2000). Multi-agent decentralized system of medical help, Management and control of production and logistics. AGH-University of Mining and Metallurgy, Krakow, Poland.
33. Tsai, W. T. (2005, October). Service-oriented system engineering: a new paradigm. In IEEE International Workshop on Service-Oriented System Engineering (SOSE'05) (pp. 3–6). IEEE.
34. Huhns, M. N., & Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. IEEE Internet computing, 9(1), 75–81.
35. Arsanjani, A. (2004).Service oriented modeling and architecture. IBM developer works, 1–15. Zheng, Z., & Lyu, M. R. (2010, May). Collaborative reliability prediction of service-oriented systems. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1 (pp. 35-44). ACM.
36. Papazoglou, M. P., Van Den Heuvel, W. J. (2003). Service-Oriented Computing: State-of-the-Art and Open Research Issues. IEEE Computer.
37. P. Gruer, V. Hilaire, A. Koukam and K. Cetnarowicz, "A Formal Framework for Multi- agent Systems Analysis and Design", Journal of Expert Systems with Applications, Vol. 23, No. 4, pp. 349–355, 2002.
38. B. Marzougui, K. Hassine, K. Barkaoui, "A New Formalism for Modeling a Multi-agent Systems: Agent Petri-nets", Journal of Software Engineering and Applications, Vol. 3, No. 12, pp 1118–1124, 2010.
39. Tadao Murata, "Petri-nets: Properties, Analysis and Applications", Proceedings of the IEEE, Vol. 77, No. 4, pp 541–580, April 1989.
40. J. R. Celaya, A. A. Desrochers, R. J. Graves, "Modeling and Analysis of Multi- agent Systems using Petri-nets", Jnl. of Comp., Academy Press, Vol. 4 (10), PP 981–996, 2009.
41. W. Chainbi, "Multi- agent Systems: A Petri-net with Objects Based Approach", IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004.

42. S. Pujari, S. Mukhopadhyay, "Petri-net: A Tool for Modeling and Analyze Multi- agent Oriented Systems", Intl. Jnl. Intelligent Sys. and Appls., Vol. 10, 103–112, 2012.
43. ISO/IEC. (2002). High-level Petri-nets–Concepts, definitions, and graphical notation. Final Draft International Standard 15909, version 4.7.1.
44. Zhou, Y., Murata, T., and DeFanti, T. (2000). Modeling and performance analysis using extended fuzzy-timing Petri-nets for networked virtual environments. IEEE Transactions on Systems, Man, and Cybernetics 30(5), 737–756.
45. Zhou, Y., and Murata, T. (2001). Modeling and analysis of distributed multimedia synchronization by extended fuzzy-timing Petri-nets, Journal of Integrated Design and Process Science 4(4), 23–38.
46. Q. Bai, M. Zhang, K. T. Win, "A Colored Petri-net Based Approach for Multi- agent Interactions", 2nd Intl. Conf. on Autonomous Robots and Agents, PP 152–157, 2004.
47. Z. Jun, H. W. Ngan; L. Junfeng, W. Jie, Y Xiaoming, "Colored Petri-nets Modeling of Multi-agent System for Energy Management in Distributed Renewable Energy Generation System," Asia-Pacific Power and Energy Engineering Conference (APPEEC), pp. 1,5, 28–31, 2010.
48. Haas, P. (2002). Stochastic Petri-nets: Modeling, stability, simulation, Springer-Verlag.
49. He, X. (2001). PZ nets—A formal method integrating Petri-nets with Z. Information and Software Technology. 43, 1–18.
50. Dong, W. L., Yu, H., & Zhang, Y. B. (2006, October). Testing bpel-based web service composition using high-level Petri-nets. In 2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06) (pp. 441–444). IEEE.
51. Chemaa, S., Bouarioua, M., & Chaoui, A. (2015). A high-level Petri-net based model for web services composition and verification. International Journal of Computer Applications in Technology, 51(4), 306–323.
52. Nagamouttou, D., Egambaram, I., Krishnan, M., & Narasingam, P. (2015). A verification strategy for web services composition using enhanced stacked automata model. Springer Plus, 4(1), 1.
53. Chen, C. S., Lin, C. H., & Tsai, H. Y. (2002). A rule-based expert system with colored Petri-net models for distribution system service restoration. IEEE Transactions on Power Systems, 17(4), 1073–1080.
54. Azgomi, M. A., & Entezari-Maleki, R. (2010). Task scheduling modelling and reliability evaluation of grid services using coloured Petri-nets. Future Generation Computer Systems, 26(8), 1141–1150.
55. Shen, W. (2002). Distributed manufacturing scheduling using intelligent agents. IEEE intelligent systems, 17(1), 88–94.
56. Yan, H. S., Wang, N. S., Zhang, J. G., & Cui, X. Y. (1998). Modelling, scheduling and simulation of flexible manufacturing systems using extended stochastic high-level evaluation Petri-nets. Robotics and Computer-Integrated Manufacturing, 14(2), 121–140.
57. Kim, H., Lee, H., Kim, W., & Kim, Y. (2010). A trust evaluation model for QoS guarantee in cloud systems. International Journal of Grid and Distributed Computing, 3(1), 1–10.
58. Ardagna, D., Di Nitto, E., Casale, G., Petcu, D., Mohagheghi, P., Mosser, S., … & Nechifor, C. S. (2012, June). Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds. In Proceedings of the 4th International Workshop on Modeling in Software Engineering (pp. 50–56). IEEE Press.
59. Gustafsson, J., Paakki, J., Nenonen, L., & Verkamo, A. I. (2002). Architecture-centric software evolution by software metrics and design patterns. In Software Maintenance and Reengineering, 2002. Proceedings. Sixth European Conference on (pp. 108–115). IEEE.