A high-level Petri net based approach for modeling and composition of web services

Sofiane Chemaa*, Faycal Bachtarzi, Allaoua Chaoui

MISC Laboratory, Department of Computer Science, Faculty of Engineering, University Mentouri Constantine, Algeria

Abstract

Web services are modular, self-describing, self-contained and loosely coupled applications, which intercommunicate via messages exchanging. The evolution of the internet and the emergence of new technologies like e-business have influenced the use of these last ones, which have become popular. The composition of web services is a topic that attracts the interest of researchers. It offers complex problems process ability even with simple existing web services while cooperating with each other. However, modeling tools and formal techniques for the completion of this task are required. In this paper, we show how simple existing web services can be composed, in order to create a composite service, which offers new features. In this context, we propose an expressive object-oriented Petri net based algebra that succeeds in the complex composition of Web services.

Keywords: web services, web services composition, algebra, e-business, object-oriented Petri net.

1. Introduction

The concept of web service basically refers to an application on internet. It is made available by a service provider and utilized by customers via standard internet protocols such as Universal Description, Discovery, and Integration (UDDI) [1], Web Service Description Language (WSDL) [2] and Simple Object Access Protocol (SOAP) [3]. The web services composition is a natural evolution of this technology, it has a remarkable potential ability in enterprise application integration and business to business reorganization. Although solutions for description, publication, discovery, and interoperability of web services are provided by the recent technologies based on WSDL, UDDI, and SOAP, complex composition cannot be achieved. So many researches are devoted to the composition of web services, the majority of these works provide different models expressed in different formalisms. The proposals of [4][5][6] represent Petri net based models, web service models based on semantic annotations are presented in [7] [8], however in [9] graph transformations are utilized by the authors in order to model and compose web services.

In this paper, the problem of web services composition is addressed using a Petri Net based framework called G-NET [10]. Unlike the other approaches, efficient and strong mechanisms for complex systems modeling, that even

*Corresponding author

Email addresses: chemaa@misc-umc.org (Sofiane Chemaa), bachtarzi@misc-umc.org (Faycal Bachtarzi), a_chaoui2001@yahoo.com (Allaoua Chaoui)
high level Petri nets do not support, are provided by this concept. Therefore, our objective is the proposal of a G-net based algebra in order to compose web services in solving complex composition, a web services structure and behavior descriptive formalism; as well as an operators representative set presented by means of syntax, semantics, and the resulting composite service; are offered.

This paper is divided into five major sections. In addition to the introductory section, section 2 presents web service modeling, using G-Nets, section 3 includes algebra for composing web services and its G-Net based formal semantics. Next, a brief preview of some related work is mentioned in section 4. Finally, some prospects and work in progress are discussed.

2. Web services as G-Nets

G-Net is a Petri Net based framework introduced by [10]. It is utilized in the modular design and specification of complex and distributed information systems. A formalism that adopts object oriented structuring into Petri Nets broadly, is provided by this framework. Advantages from the formal treatment and the expressive comfort of Petri-nets are intended to be taken, and benefits from object oriented approach (reusable software, extensible components, encapsulation, etc.), are planned to be gained.

A system designed by the G-net framework is made up of a set of autonomous and loosely coupled modules called G-Nets. The encapsulation property is satisfied by a G-net i.e. a module cannot be accessed by another module, but through a G-net abstraction, which is a well-defined mechanism, like an object in the object oriented programming concept.

A G-net consists of two parts. The Generic Switch Place (GSP), which is a special place that represents the visible part of a G-net i.e. the interface between a G-net and other ones. Besides, the Internal Structure (IS), the G-net invisible part, which constitutes the internal realization of the designed system. Both, specifying notations of the IS and the Petri net are very close [11]. For more details about the G-nets, the reader is referred to [10] [12].

Web services resemble distributed system, which is made up of a set of loosely coupled modules communicating through exchanging messages as a G-net system. Therefore, we model web services using G-Net, easily. We model each operation in the service by a method in the G-net and associate to each one a piece of Petri-Net in the IS of the G-Net. Consequently, we model the state of the service by the position of the tokens in the G-Net.

Next, some formal definitions about G-Net service and Web service are given.

Definition 1. (G-Net Service) A G-net service is a G-Net \( S(\text{GSP}, \text{IS}) \) where:

- \( \text{GSP}(\text{MS}, \text{AS}) \) is a special place that represents the abstraction of the service where:
  - \( \text{MS} \) is a set of executable methods in the form of \(< \text{MtdName} >\langle \text{description} >= \{ \langle P_1 : \text{description}, ..., P_n : \text{description} \rangle \rangle \) where \(< \text{MtdName} \rangle \) and \(< \text{description} \rangle \) are the name and the description of the method respectively. \( < P_1 : \text{description}, ..., P_n : \text{description} > \) is a set of arguments for the method, \(< \text{InitPl} >\) is the name of the initial place for the method and \(< \text{GoalPls} >\) is(are) the name(s) of the goal place(s) for the method.
  - \( \text{AS} \) is a set of attributes in the form of \(< \text{attribute} - \text{name} >= \{ \langle \text{type} \rangle \rangle \) where \(< \text{attribute} - \text{name} >\) is the name of the attribute and \(< \text{type} >\) is the type of the attribute.

- \( \text{IS}(P, T, W, F, Trc, Tra, L) \) is the internal structure of the service, a modified predicate/transition net [13], where:
  - \( P = \text{NP} \cup \text{ISP} \cup \text{GP} \) is a finite and non-empty set of places where \( \text{NP} \) is a set of normal places denoted by circles, ISP is the set of instantiated switch places denoted by ellipses used to interconnect G-Nets, GP is the set of goal places denoted by double circles used to represent final state of method’s execution.
  - \( T \) is a set of transitions.
  - \( W \) is a set of directed arcs \( W \subseteq (P \times T) \cup (T \times P) \) (the flow relation).
  - \( F \) is an application that associates a description to certain elements of \( W \).
  - \( Trc \) is an application that associates a condition to certain transitions (called selectors of transition), this condition is a logical formula constructed from variables which appeared in the inscriptions of adjacent input-arcs.
- \( Tra \) is an application that associates an action to certain transitions, this action is a sequence of affectations of values to variables.

- \( L : P \rightarrow O \cup \{\tau\} \) is a labeling function where \( O \) is a set of operation names and \( \tau \) is a silent operation.

The significance of the using attributes:

- \( V \): indicates either the card is valid or not,
- \( C \): indicates either the confidential code is valid or not,
- \( B \): indicates either the card is blocked or not,
- \( A\text{-exist} \): indicates either the sum to withdraw is available in the account or not,
- \( I \): represents the number of successive errors during the introduction of the confidential code (at each incorrect attempt the machine increment by 1 the number \( I \), if the introduced code is accepted the machine affect the value 0 to \( I \)),
- \( N \): represents the confidential code,
- \( A \): represents the sum to withdraw,
- \( OP \): represents the type of the operation (withdraw if \( OP=1 \) or credit consultation if \( OP=2 \)).

In figure 1, we propose an illustrative example of two services \textit{customer} and \textit{automatic cash dispensers (ACD)} represented by G-Nets. The service \textit{customer} reproduces the behavior of the \textit{client}, who wants to use the \textit{ACD} that insures the credit consultation, and cash withdrawals operations. For this we must have a valid magnetic card and
a confidential code. To accomplish each operation this card must be inserted into the *machine* which automatically verifies its validity. In the affirmation the customer then is invited to introduce his confidential code, which is composed of numbers, by using the *machine* keyboard. If the code is accepted the operations should be displayed on the screen by the *ACD*, therefore the desired operation is selected by the *customer* pressing the suitable icon. Two cases to consider:

- If it concerns an application for credit, the *machine* displays the account balance, whereupon it redispalyes the operations table.
- If it concerns funds withdrawal application, the *machine* displays a window related to this operation, which allows the *client* to introduce the sum he wants to withdraw. After the validation, the validity of the sum is automatically checked by the *machine*. If the sum is not sufficient; the *machine* would make an indication on the screen, it redispalyes the operations table. In the case of the availability of funds, it will distribute the banknotes representing the inserted sum.

However, if the confidential code is erroneous after three successive attempts, the magnetic card will be captured by the *ACD* for security purposes, because it may be that the user of the card is not its owner. Its recuperation depends on a special processing that is not tackled in this example.

It should be noted that the *customer* may change his opinion, and interrupt any operation at any time before the validation.

The ISP notation serves as the primary mechanism for specifying the interconnection between different G-Nets. In the example of Figure 1, we integrate the ISP of *ACD* with its methods *validity*, *code*, *display-op*, *oper* and *withdrawal* in the IS of *Customer* to specify a client-server relation.

**Definition 2. (Web Service)** A Web service is a tuple $S = (\text{Name}_S, \text{Desc}, \text{Loc}, \text{URL}, \text{CS}, \text{SGN})$ where:

- $\text{Name}_S$ is the name of the service used as its unique identifier.
- $\text{Desc}$ is the description of the provided service. It summarizes what functionalities the service offers.
- $\text{Loc}$ is the server in witch the service is located.
- $\text{URL}$ is the invocation of the Web service.
- $\text{CS}$ is a set of the component services of the Web service, if $\text{CS} = \{\text{Name}_S\}$ then $S$ is a basic service, otherwise $S$ is a Composite service.
- $\text{SGN} = (\text{GSP}, \text{IS})$ is the G-Net modeling the dynamic behavior of the service.

The concept of G-Net service and Web service being presented, we show in the next section how Web services can be incrementally composed. We recall that we use G-Nets as a means to offer a flexible and powerful algebra.

### 3. Web Service Composition

The cooperation of two or more different web services leads to a novel task achievement. Consequently, a new value to the services collection is added, giving the example of the collaboration of a Hotel Booking Service and a Web Mapping Service such as Google Maps API for the clients’ guidance. Hence, a complex Web service performing the original tasks in addition to a new one is the result of the cooperation of these services.

[4][14][15] are devoted to discuss different constructs for composing Web services. Based on these works, an algebra combining different Web services is proposed in this paper. Sequence, Parallel, Alternative, Iteration, and Arbitrary Sequence as basic constructs will be taken. Moreover, four more developed constructs which are Discriminator, selection, refinement and replace are defined. After that, each operator formal semantics in terms of G-nets, after its informal definition provision, is given.

#### 3.1. Composition constructs

The BNF-like notation below describes a grammar defining the set of services that can be generated using algebra’s operators.

$$
S ::= \epsilon \mid X \mid S \triangleright S \mid S \triangleleft S \mid \bigcirc S \mid S \leftrightarrow S \mid S \square S \mid (S \cdot S) \Rightarrow S \mid [S \ldots S] \mid \text{Ref}(S, a, A) \mid \text{Rep}(S, S, S)$$

- $\epsilon$ is the Zero Service (or empty service), i.e a service which performs no operation.
• X is a constant service. It consists of a service performing operation that cannot be split into sub-operations. This service is called Atomic.

• \( S_1 \xrightarrow{} S_2 \) represents a composite service that performs one service immediately followed by another, i.e \( \xrightarrow{} \) is a sequence operator.

• \( S_1 \leftrightarrow S_2 \) represents a composite service that can reproduce either the behavior of \( S_1 \) or \( S_2 \), i.e \( \leftrightarrow \) is an alternative (or a Mutual Exclusion) operator.

• \( \odot S \) represents a composite service where one service is successively executed multiple times, i.e \( \odot \) is an iteration operator.

• \( S_1 \models S_2 \) is a composite service that performs any arbitrary sequence of the services \( S_1 \) and \( S_2 \), i.e \( \models \) is an unordered sequence operator.

• \( S_1 \parallel S_2 \) represents a composite service which performs the two services \( S_1 \) and \( S_2 \) at the same time and independently. The resulting service waits until the end of execution of \( S_1 \) and \( S_2 \), i.e \( \parallel \) is a parallel operator.

• \( (S_1 \odot S_2 \odot \ldots \odot S_{n-1}) \models S_n \) is a composite service that waits for the execution of one service (among the n-1 services), before activating the subsequent service \( S_n \), i.e \( (\odot \ldots \odot) \models \) is a discriminator operator. Note that the n-1 first services are performed in parallel and without communication.

• \([S_1|S_2|\ldots|S_n]\) represents a composite service that selects and executes another service among n services available that perform the same task, i.e \([\ldots]\) is an operator of selection.

• \(\text{Ref}(S,a,A)\) represents a service which behaves as \( S \) except for the operation labeled by “a”, which is replaced by “A”. This latter is a modified predicate/transition net, i.e \( \text{Ref}() \) is a refinement operator.

• \(\text{Rep}(S,S_1,S_2)\) represents a composite service similar to \( S \) except for the service component \( S_1 \), which is replaced by \( S_2 \) not empty, i.e \( \text{Rep}() \) is a replace operator.

The proposed algebra verifies the closure property. This latter ensures that the product of any operation on services is itself a service to which we can apply algebra operators. We are thus able to built more complex services by aggregating and reusing existing services through declarative expressions of service algebra.

3.2. Formal Semantics

In this section, we give a formal definition, in term of G-Nets, of the composition operators. Let \( S_i = (Name_{S_i}, Desc_{i}, Loc_{i}, URL_{i}, CS_{i}, SGN_{i}) \) with \( SGN_{i} = (P_i, T_i, W_i, F_i, Trc_i, Tra_i, L_i) \) for \( i = 1, \ldots, n \) be n Web Services such that \( P_i \cap P_j = \emptyset \) and \( T_i \cap T_j = \emptyset \) for \( i \neq j \).

Empty Service. The empty service \( \epsilon \) is a service that performs no operation. It is used for technical and theoretical reasons.

Definition 3. The Empty service is defined as \( \epsilon = (Name_S, Desc, Loc, URL, CS, SGN) \) where:

- \( Name_S = \text{Empty} \).
- \( Desc = \text{“Empty Web Service”} \).
- \( Loc = \text{Null} \), stating that there is no server for the service.
- \( URL = \text{Null} \), stating that there is no invocation for the service.
• CS = \{Empty\}.

• SGN = (GSP, IS) where:
  - GSP = (MS, AS) where MS = ∅, AS = ∅
  - IS = (p, ∅, ∅, ∅, ∅, ∅)

In Figure 2(b), we show the graphic representation of the empty service (ε) in terms of G-Net.

**Sequence.** The sequence operator allows the construction of a service composed of two services executed one after the other. This construction is used when a service should wait the execution result of another one before starting its execution. For example when subscribing to a forum, the service Registration is executed before the service Confirmation.

**Definition 4.** The service $S_1 \triangleright S_2$ is defined as $S_1 \triangleright S_2 = (NameS, Desc, Loc, URL, CS, SGN)$ where:

- NameS is the name of the new service.
- Desc is the description of the new service.
- CS is the concatenation of $CS_1$ and $CS_2$.
- $SGN = (GSP, IS)$ where:
  - GSP = (MS, AS) where MS = Mtd. Seq([...](p_1)(p_3)), AS = [...].
  - IS = (P, T, W, F, Trc, Tra, L) where $P = \{p_1, p_2, p_3\}$, $T = \{t_1, t_2\}$, $W = \{(p_1, t_1), (p_1, p_2), (p_2, t_2), (t_2, p_3), (p_3, t_4), (t_4, p_4)\}$, $F = [...], Trc = [...], Tra = [...], L = \{(p_1, Isp(S_1)), (p_2, Isp(S_2)), (p_3, goal)\}$.

Given the two services $S_1$ and $S_2$ shown in Figure 2(a), the composite service $S_1 \triangleright S_2$ is represented by the G-Net shown in Figure 2(c).

**Alternative.** Given two services $S_1$ and $S_2$, the alternative operator reproduce either the behavior of $S_1$ or $S_2$, but not both. For example the service Identification is followed either by the service Allow-access or the service Deny-access.

**Definition 5.** The service $S_1 \lhd S_2$ is defined as $S_1 \lhd S_2 = (NameS, Desc, Loc, URL, CS, SGN)$ where:

- NameS is the name of the new service.
- Desc is the description of the new service.
- CS is the concatenation of $CS_1$ and $CS_2$.
- $SGN = (GSP, IS)$ where:
  - GSP = (MS, AS) where MS = Mtd. Alt([...](p_1)(p_4)), AS = [...].
  - IS = (P, T, W, F, Trc, Tra, L) where $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t_1, t_2, t_3, t_4\}$, $W = \{(p_1, t_1), (p_1, p_2), (p_2, t_3), (t_3, p_4), (p_1, t_2), (t_2, p_3), (p_3, t_4), (t_4, p_4)\}$, $F = [...], Trc = [...], Tra = [...], L = \{(p_1, T), (p_2, Isp(S_1)), (p_3, Isp(S_2)), (p_4, goal)\}$.

Given the two services $S_1$ and $S_2$ shown in Figure 2(a), the composite service $S_1 \lhd S_2$ is represented by the G-Net shown in Figure 2(d).

**Iteration.** The iteration operator allows the service $S$ to be performed a certain number of times in a row. An example of use of this construct is when a customer orders a good a certain number of times from a service.

**Definition 6.** The service $\odot S_1$ is defined as $\odot S_1 = (NameS, Desc, Loc, URL, CS, SGN)$ where:

- NameS is the name of the new service.
- Desc is the description of the new service.
- CS = $CS_1$.
- $SGN = (GSP, IS)$ where:
  - GSP = (MS, AS) where MS = Mtd. Iter([...](p_1)(p_2)), AS = [...].
  - IS = (P, T, W, F, Trc, Tra, L) where $P = \{p_1, p_2\}$, $T = \{t_1, t_2\}$, $W = \{(p_1, t_1), (t_1, p_1), (p_1, t_2), (t_2, p_2)\}$, $F = [...], Trc = [...], Tra = [...], L = \{(p_1, Isp(S_1)), (p_2, goal)\}$.

If we consider the service $S_1$ shown in Figure 2(a), the composite service $\odot S_1$ is represented by the G-Net shown in Figure 3(a).

**Arbitrary Sequence.** The arbitrary sequence operator specifies the execution of two services that must not be executed concurrently. This construct is useful when there is no benefits to execute services in parallel. For example when there is no deadline to accomplish the global task and the parallelism generates additional costs.

**Definition 7.** The service $S_1 \leftrightarrow S_2$ is defined as $S_1 \leftrightarrow S_2 = (NameS, Desc, Loc, URL, CS, SGN)$ where:
services completely independent. This construct is useful when a service executes multiple atomic services performing the same task (different services). The composite construct obtained by applying the Discriminator operator submits redundant orders to different services through the Web. For customers, best services are those which respond in optimal time and are constantly available.

Given the two services $S_1$ and $S_2$ shown in Figure 2(a), the composite service $S_1 \boxplus S_2$ is represented by the G-Net shown in Figure 3(b).

**Parallel.** Given two services $S_1$ and $S_2$, the parallel operator builds a composite service performing the two services ($S_1$ and $S_2$) in parallel and without interaction between them. The accomplishment of the resulting service is achieved when the two services are completed. This construct is useful when a service executes multiple atomic services completely independent.

**Definition 8.** The service $S_1 \boxplus S_2$ is defined as $S_1 \boxplus S_2 = (NameS, Desc, Loc, URL, CS, SGN)$ where:

- NameS is the name of the new service.
- Desc is the description of the new service.
- CS = $CS_1 \cup CS_2$.
- Loc is the location of the new service.
- URL is the invocation of the new service.
- SGN = $(GPS, IS)$ where:
  
  - $GPS = (MS, AS)$ where $MS = Mtd.ar.seq([\ldots](p_1)(p_9)), AS = \ldots$.
  
  - $IS = (P, T, W, F, T_{rc}, T_{ra}, L)$ where $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}, T = \{t_1, t_2, t_3, t_4, t_5, t_6\}, W = \{(p_1, t_1), (t_1, p_2), (t_1, p_3), (t_1, p_4), (p_2, t_2), (t_2, p_3), (t_2, p_4), (p_3, t_3), (p_3, t_4), (p_3, t_5), (p_3, t_6), (t_6, p_7), (p_5, t_8), (t_8, p_9)\}, F = \ldots, T_{rc} = \ldots, T_{ra} = \ldots, L = \{(p_1, \tau), (p_2, \tau), (p_3, \tau), (p_4, \tau), (p_5, \tau), (p_6, \tau), (p_7, \tau), (p_8, \tau), (p_9, Isp(S_1)), (p_6, Isp(S_2)), (p_9, goal)\}$.

Given the two services $S_1$ and $S_2$ shown in Figure 2(a), the composite service $S_1 \boxplus S_2$ is represented by the G-Net shown in Figure 3(c).

**Discriminator.** The main goal of the discriminator operator is to increase reliability and delays of the services through the Web. For customers, best services are those which respond in optimal time and are constantly available. The composite construct obtained by applying the Discriminator operator submits redundant orders to different services performing the same task ($S_1, S_2, \ldots, S_{n-1}$ for example). The first service which responds to the request activates the service $S_n$. All other late responses will be ignored.

**Definition 9.** The service $(S_1 \boxplus S_2 \boxplus \ldots S_{n-1}) \triangleright S_n$ is defined as $(S_1 \boxplus S_2 \boxplus \ldots S_{n-1}) \triangleright S_n = (NameS, Desc, Loc, URL, CS, SGN)$ where:

- NameS is the name of the new service.
- Desc is the description of the new service.
- CS = $CS_1 \cup CS_2$.
- Loc is the location of the new service.
- URL is the invocation of the new service.
• \( CS = \bigcup_{i=1}^{n} CS_i \), \( SGN = (GSP, IS) \) where:

- \( GSP = (MS, AS) \) where \( MS = Mtd.disl([[\ldots](p_i)(p_{n+3})]]) \), \( AS = [B : Boolean, \ldots] \).
- \( IS = (P, T, W, F, Trc, Tra, L) \) where \( P = [p_i | 1 \leq i \leq n + 3] \), \( T = [t_i | 1 \leq i \leq n + 3] \), \( W = [(p_i, t_i)]1 \leq i \leq n + 2] \cup \{(t_i, p_i), (t_{n+1}, p_{n+2}), (t_{n+2}, p_{n+3}) \}, F = \{(f(p_i, t_i) = [B], f(p_{n+1}, t_{n+1}) = [B], f(p_{n+2}, t_{n+2}) = [B], \ldots, Trc = \{trc(t_{n+1}) = [B \iff True], trc(t_{n+2}) = [B \iff False], \ldots\}, Tra = \{tra(t_{n+1}) = [B \iff True], tra(t_{n+2}) = [B \iff False], \ldots\}, L = \{(p_i, isp(S_{i-1})(\ldots))2 \leq i \leq n \} \cup \{(p_1, t_1), (p_{n+1}, t_{n+1}), (p_{n+2}, isp(S_{n})(\ldots))\} \}

Graphically, given \( S_1, \ldots, S_n \), the composite service \( [S_1 \square S_2 \square \ldots \square S_{n-1}] \supseteq S_n \) is represented by the G-Net shown in Figure 3(d).

**Selection.** let’s have “n” web services \( (S_1, \ldots, S_n) \) provide differently the same service, these services are extended by a specific method (req) which receives and answers to certain requests. (An example one of these services is presented in Figure 3(e)).\( Select \) is an operator which permits, after treating the answers, to choose one service among others which have respond to the same request, this operator offers the possibility to benefit of the best service. The choice is related to several criteria, for example, the best services of sale those which represent seductive prices and suitable delay of delivery.

**Definition 10.** The service \([S_1][S_2][\ldots][S_n] \) is defined as \([S_1][S_2][\ldots][S_n] = (NameS, Desc, Loc, URL, CS, SGN) \) where:

- NameS is the name of the new service.
- Desc is the description of the new service.
- CS = \( \bigcup_{i=1}^{n} CS_i \), \( SGN = (GSP, IS) \) where:

Reference. \( : \) the refinement permits to replace certain operations of the service by more detailed ones. Refinement is the transformation of a design from a high level abstract form to a lower level more concrete form hence allowing hierarchical modeling.

**Definition 11.** The service \( Ref(S_1, a, A) \) is defined as \( Ref(S_1, a, A) = (NameS, Desc, Loc, URL, CS, SGN) \) where:

- NameS is the name of the refined service.
- Desc is the description of the refined service.
- CS = \( \bigcup_{i=1}^{n} CS_i \) if \( a \in L_{(S_1)(P(S_1)))} \), \( CS_i \) otherwise.
- \( SGN = (GSP, IS) \) where:

Graphically, given \( S_1, \ldots, S_n \), the composite service \( [S_1][S_2][\ldots][S_n] \) is represented by the G-Net shown in Figure 4(a).
Another one, which insures the same functions. Here it is a question of behavior equivalence. This operation permits to solve the problem of the availability. i.e if a service is not available; it can be replaced by AI planners [17]. Disadvantages remain the facility to find the explicit services aim. This latter represents an essence when composing patterns in [9]. Unless the advantage of the clear understanding of the messages meaning is presented, their main disadvantages remain the facility to find the explicit services aim. This latter represents an essence when composing by AI planners [17].

4. Related Work

Formerly, various methods related to Web service composition have been proposed in literature, and all of them attempt to offer languages, semantic models, and platforms for the proposal of effective results for that issue. A set of primitives offered by composition languages like BPEL4WS [16] permits the interaction between services being composed. These approaches identify the flow of processes and bindings between services, previously.

Terms from pre-agreed ontologies are used by ontology-driven approaches for composing Web services such as [7], SAWSDL [8] and [9], for declaring the preconditions and the effects of the concerned services. In the two first works, concepts describe the inputs and the outputs. While, they are expressed in terms of instance-based graph patterns in [9]. Unless the advantage of the clear understanding of the messages meaning is presented, their main disadvantages remain the facility to find the explicit services aim. This latter represents an essence when composing by AI planners [17].
Like [4], our work concerns the Petri Net based framework for web services composition. A Petri Net Algebra was proposed in [4]. Despite the full expression of its model, data types cannot be distinguished, because of the use of an elementary Petri net model. Instead of which we utilize high level Petri nets, called G-Nets. In this formalism, we can label the edges by constants or variables denoting the parameters of the tokens. Similarly to edges, we can associate conditions to transitions. The value of the parameters for which the transition is fireable is pre-defined by these conditions. Thus, data types in this approach are distinguishable. This issue is also treated in [5] by modeling Web services and their composition using colored Petri nets [18]. The foremost advantage of this work, Compared to [5], is that less effort is needed when modeling complex services; moreover, more reduced models are produced. The approach of [6] resembles this work; it focuses on modeling the process of Web service composition by a kind of Object Oriented Petri Nets. While this approach has been identified formally, and based on a well-founded framework called G-Nets.

5. Conclusion

In this paper, a simple efficient approach for Web Services composition has been proposed. Benefiting from the formal, modular, and object-oriented aspects of G-Nets is the main advantage of this approach. Besides, the specification and prototyping of complex Web services are allowed by modeling by G-Nets. A G-Nets based algebra is developed for Web service composition. The formal semantics of the composition operators are defined, by means of G-Nets, in this context. Using this underlying framework provides a rigorous approach for verifying properties and detecting inconsistencies between Web services.

In the future, this approach is planned to be extended with advanced operators, which can support more complex combination of Web services. In addition, we are aiming at the development of a Java Prototype tool implementing the introduced operators.

References