WS-mcv: An Efficient Model Driven Methodology for Web Services Composition

Fayçal Bachtarzi
Department of Computer Science, University Mentouri, Constantine, Algeria
Email: bachtarzi@misc-umc.org
Allaoua Chaoui
Department of Computer Science, University Mentouri, Constantine, Algeria
Email: a_chaoui2001@yahoo.com
Elhillali Kerkouche
Department of Computer Science, University of Jijel, Jijel, Algeria
Email: elhillalik@yahoo.fr

***Abstract*— Web services are available applications on the Web which can be invoked by users to accomplish a potentially business task. However, to meet user's requirements, it becomes necessary to dynamically organize existent services and combine them, responding thus to a new purpose. In this paper, we propose a methodology called WS-mcv (Web Service Modeling, Composing and Verifying) that addresses the main problems arising in Web service composition area. WS-mcv represents an efficient and modular multistep approach achieved by breaking service composition into three processes: service modeling, automatic composition and formal verification. The proposed methodology makes use of the G-Net framework to allow an easiest modeling of basic and existent services. We propose a collection of expressive G-Net based operators that successfully solves complex Web service composition. WS-mcv also defines means to ensure composition correctness. All the processes of WS-mcv have been successfully automated in a model transformation based visual environment.**

***Index Terms*— Web services composition, G-Nets, MDE, Graph transformation, $ATOM^3$, G-Net Algebra**

I. Introduction

Web services are software components available on the Web that implement business collaborations between corporations. They can be invoked via Internet to accomplish a potentially business task making possible interactions between applications and e-customers. Programs or external users can access Web services using 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]. Web services have the particularity to provide specific and general functionalities and, in most cases, cannot respond to user's requirements. To provide users customized services, it becomes then necessary to combine existent basic services. The process achieving this task is called Web service composition. Current solutions based on UDDI, WSDL and SOAP offer solutions for description, publication, discovery and interoperability of Web services but do not accomplish their complex composition. Research in the area of service composition has focused on trying to provide models expressed in different formalisms. Some of the propositions used different kinds of Petri Nets, basic Petri nets [4], colored Petri Nets [5] [6] and Object oriented Petri Nets [7]. Other proposals exploit semantic features offered by Ontologies [8] [9] [10]. In this paper, we address the Web service composition problem by defining an efficient multistep methodology called WS-mcv (Web Service Modeling Composing and Verifying). WS-mcv has the advantage to resolve the main problems arising in Web service composition area. It breaks service composition process into several phases to offer solutions for both 1) specifying services, 2) automatically composing them and 3) ensuring their correctness. For Web services specification, we have proposed a set of modeling rules which allows modeling Web services in a high level Petri Nets framework called G-Nets [11]. For services composition, we have defined a G-Net based algebra that successfully solves complex composition. The proposed algebra supports basic constructs as well as more elaborate ones. All the operators within the algebra are syntactically and semantically defined by means of G-Nets. To ensure Web services correctness, we exploit translation rules [12] which enables to transform G-Net specifications into their equivalent Predicate/Transition Nets (PrT-Nets) [13]. Unlike others approaches which develop their own verification tools [14], we perform this transformation in order to exploit existing tools with a variety of analysis techniques for PrT-Nets. Each of the underlying well-defined phases of our methodology is performed by a different process which has been successfully automated. As the main requirement of our approach is to offer a high level of genericity and to make abstraction of a particular implementation, we propose to use Model Driven Engineering (MDE) techniques that support model evolution and manipulate models as instances of meta-models. The modeling process is implemented as a visual environment that allows designing the services according to a G-Net meta-model. The composition and verification processes, including the proposed operators, are implemented by graph transformation techniques. The remainder of this paper is organized as follows. In the next section, we present some related work. Section 3 outlines

doi:10.4304/jsw.7.12.2874-2885

the overall approach and presents the different phases of WS-mcv methodology. We introduce modeling, composition and verification processes in Section 4, 5 and 6 respectively. For each of them, we describe the operating mode and the solutions adopted for its implementation. We finally conclude the paper by summarizing the main contributions and identifying future research directions.

II. Related Work

Various techniques for web service composition have been suggested in the literature. Most of them try to provide languages, semantic models and platforms in order to propose efficient solutions to this problem. Syntactic (XML-based) service composition [15] has a limited ability to support automatic composition. This is essentially due to the absence of semantic representations of the available services. Indeed, composition languages such as BPEL4WS [16] provide a set of primitives that allows interaction between services being composed. In these approaches, flow of processes and bindings between ervices are specified in advance. On the contrary, semantic approaches [8] [10] [5] [6] allow describing various aspects of Web services using machine-understandable semantics or solid mathematical basis. Semantic approaches are mainly classified into two categories: ontology-driven approaches and Petri Nets based approaches. In the following, we present some works related to each of the identified classes.

A. Ontology-driven approaches

Ontology-driven approaches for Web services composition [8] [9] [17] [10] use terms from pre-agreed ontologies to declare preconditions and effects of the concerned services. Works of [10] lead to OWL-S, which is a particular ontology for declaring and describing services. OWL-S provides a standard vocabulary that can be used together with the other aspects of the OWL description language to create service descriptions. Similarly, SAWSDL [8] defines a set of extension attributes useful to annotate WSDL interfaces and operations. These latter are used to publish a web service in a registry. The proposal of [17] is different as it makes use of semantic graph transformations to model web services. In the proposed model, each web service operation is associated with a semantic annotation that describes the input and output messages specifications using RDF graph patterns. The main difference here is that in [10] and [8] the inputs and outputs are expressed by concepts, while [17] describe them in terms of instance-based graph patterns. If these approaches present the advantage of clearly understanding the meaning of the messages, their main drawback remains the difficulty to discover the explicit goal of the services. This latter constitutes a key element when composing by AI planners [18]. WSML [9] also provides a formal syntax for web service modeling based on Description Logics, First-Order Logic and Logic Programming. It allows specifying axioms with variables in the pre- and post-conditions of a service capability. However, it does not have an explicit model to define the components of a message and their semantics. In all these works, the composition problem is modeled as a planning problem based on a reasoning process which uses semantic descriptions of services. Composing by reasoning is a challenging task as it is time consuming and it relies on a set of goals, plans, and rules to design complex processes.

B. Petri-nets based approaches

Existing web service composition works also uses Petri nets framework, simple Petri nets [4] [19] as well as High level Petri nets [7] [5] [6]. In [4] the authors propose a Petri net-based algebra for modeling Web services control flows. Their model is suitably expressive to make possible the creation of dynamic and temporary relationships among services. However, the main drawback is that the data types cannot be distinguishable because an elementary Petri net model is used. The work of [6] also deals with this problem by modeling and composing Web services using Colored Petri nets (CPN) [20]. Their proposal offers semantic support improving the reliability and maintainability of composite services. It also allows analyzing availability, confidentiality and integrity of the composite services. CPN framework is also exploited by [5] where an efficient algebra is suggested to model Web service composition. Algorithms to construct and execute a composite service are also delivered. These two works seems especially connected, even if in [5] the service composition sequence cannot be generated automatically because pre-defined conditions are required. In the Object-Oriented Petri Nets (OOPN) based approach [7], the Web service composition relies on mapping a service as the collaborative objects. Therefore, describing their behavior and communications is easily performed using the OOPN model. Their approach is much interesting since they offer a Web process design tool (WPDT) allowing to graphically doing the composition. The behavior and performances of a system can be checked when studying the process in action. This survey highlights the challenges and the proposed solutions for integrating existing services to create new value-added ones. Due to solid theoretical basis of semantic methods, they are well suited for not only modeling and composing Web services, but also verifying their behavioral correctness. Ontologies are not expressive enough to accomplish this task, because they are better to describe the features of a system rather than its behavior. In our work, we use a kind of object-oriented Petri-Nets for the specification of complex Web services, namely the G-Net framework which is powerful enough to capture the semantics of Web services combinations. The following section outlines the proposed methodology along with the involved phases and technologies.

III. WS-mcv methodology overview

Our approach aims to achieve Web service composition through a simple yet powerful methodology. WS-mcv

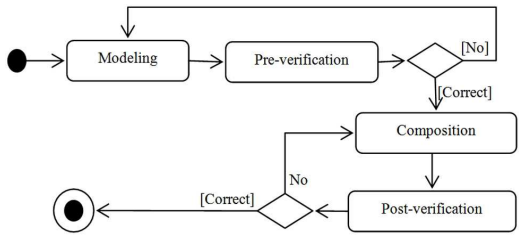

Figure 1. WS-mcv methodology

breaks the composition process into four phases:

1) Modeling of Web services using the basic concepts of the G-Net framework,
2) Pre-verification of each modeled Web service,
3) Composition of the verified Web services using the G-Net algebra, and
4) Post-verification of the resulting service.

The separation of the Web service composition process into a number of well-defined phases has several advantages. First, it simplifies the composition process, since each phase has a specific goal. Second, it facilitates the verification process, since it becomes easier to target the potential errors. Third, if offers more flexibility for automating the whole composition process, since each phase can be independently implemented. The complete composition process based on the proposed methodology is illustrated by UML activity diagram of Fig. 1. We can see the execution order of the different phases as well as the interactions between them. We specify that each phase uses the results of the previous one and achieved by an independent process. In the following, we give more details about each phase.

A. Modeling phase

The modeling phase is the first step of WS-mcv. Its role is to translate the services specifications into G-Nets. The basic concepts of G-Nets are: the interface and the internal structure. These two concepts are used for designing the component services while taking into account the modeling constraints imposed by the G-Net framework. The intention is to gain benefits of the modularity and the flexibility offered by this formalism on the one hand and to exploit its ease of conceptual modeling on the other hand. This phase is achieved by the modeling process which will be wholly described in Section 4.

B. Composition phase

The composition phase takes as input G-Nets representing the services to compose, together with a composition formula and generates a new value added service as a G-Net. To perform this operation, we propose a G-Net based algebra. This algebra offers a representative set of operators that can be applied to the G-Net services. The composition formula provided as input is in fact an algebraic expression where operands are the handled services and operators are graph manipulating operations performed on them. The complete composition process accomplishing this phase is presented in detail in Section 5.

C. Pre/Post-verification phases

These two phases represent in WS-mcv the second and the fourth phases respectively and are both achieved by the verification process. Accomplishing verification before and after the composition has the main advantage to facilitate this operation. Traditional approaches don't accomplish verification at all or only verify the resulting service. In doing so, it becomes difficult to localize the potential errors. Unlike other approaches, ours detects if the anomalies occur in the component services or in the composite one. Pre-verification phase is carried out before composition. It intends to verify whether the obtained G-Net models will be executing as expected and don't contain behavioral inconsistencies such as deadlock or livelock. It is convenient to detect and correct possible errors as early as possible. If necessary, steps (1) and (2) are repeated until the specification of the modeled services passes the verification. Post-verification phase is applied after composition in order to check the correctness of the resulting composition; i.e. the integration of the partner services correctly runs. Hence, steps (3) and (4) may also be repeated until the composition of the concerned services passes the verification.

D. WS-mcv realization

As we have defined above, WS-mcv methodology intends to accomplish Web service composition into several steps, each one achieved by a specific process. Our proposal, as we will see in the next sections, doesn't remain at the descriptive level. We propose to describe not only the operating mode of each process, but also the techniques adopted for its implementation. To implement the WS-mcv processes, we have identified three requirements that must be met by our system:

1) It shall support the evolution of the used modeling language, i.e. possible extensions of the G-Net framework.
2) It shall be convivial, to allow users designing and manipulating models (G-Net specifications) in a direct and intuitive way.
3) It shall offer a high level of genericity that allows users to make abstraction of a particular implementation.

To meet these requirements, we propose to:

1) Use the syntax of the visual modeling language (G-Net) by means of meta-modeling.
2) Exploit a visual environment that allows designing the services according to the G-Net meta-model.
3) Express model manipulation (i.e. composition) by means of graph-transformation.
4) Use MDE (Model Driven Engineering) techniques to provide a generic approach that manipulates models as instances of meta-models.

IV. Modeling using G-Net Framework

In this section, we first present the G-Net framework, and then we give formal definitions of G-Net services and web service together with some modeling rules. We finally describe the operating mode and the implementation of the modeling process.

A. The G-net framework

G-Net is a Petri Net based framework introduced by [11]. It is used for the modular design and specification of complex and distributed information systems. This framework provides a formalism that extensively adopts object oriented structuring into Petri Nets. The intention is to take advantages from the formal treatment and the expressive comfort of Petri Nets and at the same time to gain benefits from object-oriented approach (reusable software, extensible components, encapsulation, etc...). A system designed by the G-Net framework consists of a set of autonomous and loosely coupled modules called G-Nets. Similarly to an object in the object oriented programming concept, a G-Net satisfies the property of encapsulation i.e. a module can only access another one throw a well defined mechanism called G-Net abstraction. A G-Net is composed of two parts: the Generic Switch Place (GSP) and the Internal Structure of the G-Net (IS). The GSP is a special place and represents the visible part of the G-Net i.e. the interface between a G-Net and other ones. The Internal structure is the hidden part of the G-Net; it represents the internal realization of the designed system. The notation used for IS specification is very close to the Petri Net notation [21]. For more elaborate introduction to G-Nets, the reader is referred to [11] [22]. Like a G-Net system, Web services are assimilated to a distributed system that consists of a set of loosely coupled modules which communicate throw messages exchange. Thus, modeling Web services using G-Net is straightforward.

B. Web services as G-nets

In order to reduce the specification ambiguity and to help designers to understand description and possible behaviors of Web services, we give some formal definitions about G-Net service and Web service.

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

- $GSP(MS, AS)$ is a special place that represents the abstraction of the service where:
  - MS is a set of executable methods in the form of $< MtdName >< description >= \{[P1 : description, ..., Pn : description](< InitPL >)\}$
    where $< MtdName > and < description >$ are the name and the description of the method respectively.
    $< P1 : description, ..., Pn : description >$ is a set of arguments for the method and $< InitPl >$ is the name of the initial place for the method.
  - AS is a set of attributes in the form of $< attribute - name >= \{< type >\}$ where $< attribute - name >$ is the name of the attribute and $< type >$ is the type of the attribute.
- $IS(P, T, W, l)$ is the internal structure of the service, a modified predicate/transition net [13], where:
  - $P = NP \cup ISP \cup GP$ is a finite and non-empty set of places where 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)
  - $l : P \rightarrow O \cup \{\tau\}$ is a labeling function where O is a set of operation names and τ is a silent operation.

Definition 2. (Web Service) A Web service is a tuple $S = (NameS, Desc, Loc, URL, CS, SGN)$ where:

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

Since Web service designer may be unfamiliar with G-Nets, we present modeling rules of a Web service into G-Net concepts.

1) Each Web service is represented by a different G-Net.
2) A service operation is modeled by a method in the G-Net. Then each method is associated a piece of Petri Net in the IS of the G-Net.
3) Messages exchanged by the service and its customers are modeled by tokens.
4) The state of the service is modeled by the position of the tokens in the G-Net.
5) Synchronization and coordination of information exchange between places is modeled by a transition associated with input and output arcs.
6) Interconnection between different G-Nets is carried by the ISP notation that represents the primary communication mechanism (for example, integrating the ISP of a server in the IS of a customer service specifies a client/server relation).

C. Operating mode

MDE approach is founded on the massive use of models during all the steps of an application life cycle. It en-

sures model stability by using meta-models as structuring elements. For designer's applications, using these techniques prevents them to hardly encode their applications when creating custom modeling environments (domain-specific tools), as MDE raises the level abstraction from source code to models. Once the meta-model is defined, it is easy to make small modifications to obtain customized variations of the modeling formalism for specific use.

To implement the modeling process, we exploit the powerful features of a tool called $ATOM^3$, A Tool for Multi-formalism and Meta-Modeling [23]. $ATOM^3$ allows describing (or meta-modeling) different kinds of formalisms used to model systems. In $ATOM^3$, Entity Relation-ship (ER) formalism extended with constraints is available at the meta-meta-level. Therefore, the designer must use the ER formalism when modeling new meta-formalisms. Given the meta-model of the G-Net formalism, $ATOM^3$ can automatically generate a visual modeling tool to create and edit models in this formalism. In the context of our work, we make use of the G-Net meta-model defined by [12]. As shown in Fig. 2, the meta-model contains four classes $(G-NetsGSP, G-NetsIS, G-NetsPlace, G-NetsTransition)$ and five relations $(GNetsRealisation, G-Nets-hasPlaceInsid, G-Nets-hasTransitionInsid, G-NetsPl2Tr, G-NetsTr2Pl)$. To ensure a correct appearance of G-Nets models, the G-Nets meta-model associates graphical constraints to each G-Net entity. For example, a place is associated to a circle and a transition is associated to a rectangle. These constraints are specified when creating the meta-model in $ATOM^3$. Once the tool is generated (according to the meta-model), the user interface buttons allow the designer to create entities of his model defined in the G-Nets meta-model. He then applies the modeling rules defined above to conceptualize any service in the G-Net formalism. The created G-Net services can be stored, edited and modified. Fig. 3 illustrates the complete modeling process. After the compilation of the G-Net meta-model, $ATOM^3$ only accepts syntactically correct models in this formalism. The right window in the figure shows an example of a modeled service edited by the generated tool. The G-Net service reproduces the behavior of a checkout system service. The GSP of the service contains one method $(mtd.Collect[Bill : data](PMC)(GP))$ which receives the attribute 'Bill' and have PMC and GP as initial and goal places respectively. The Checkout service checks the payment mode that the client invokes (PMC). According to the payment mode, the service performs the necessary operations.

V. Composing using the G-Net Algebra

This section first presents the G-Net based algebra that allows combining G-Net services and then shows how the proposed set of operators is implemented using Graph transformation techniques.

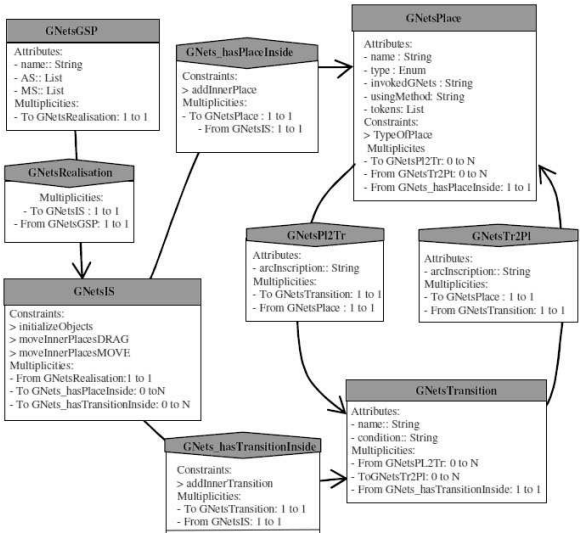

Figure 2. The G-Nets meta-model

A. The G-net based algebra

WS-mcv allows combining existing G-Net services to obtain a new value added one that best meets end users' requirements. For example, a service of hotel booking can collaborate with a Web mapping service like Google Maps API Web Service [24] to inform customers about the location of hotels. The collaboration of these services generates a composed Web service which performs the original individual tasks as well as a new one. Various constructs for Web service composition were discussed in later works [4] [25] [26]. Based in these works, we present an algebra that combines existing Web services for building more complex ones. We will take Sequence, Parallel, Alternative, Iteration and Arbitrary Sequence as basic constructs. We also define three more developed constructs which are Discriminator, Delegation and Selection. The BNF-like notation below describes the grammar defining the set of services that can be generated using our algebra's operators.

$$S ::= \epsilon \mid X \mid S \blacktriangleright S \mid S \blacktriangleleft\blacktriangleright S \mid \circlearrowleft S \mid S \Leftrightarrow S \mid S \square S \mid (S \boxdot S) \gg S \mid Deleg(S1, o, S2) \mid Select[S1 : Sn]$$

In what follows, we first give an informal definition of each operator and then we define its syntax and formal semantics in terms of G-Nets.

The Empty service (ϵ) is the Zero Service; i.e. it performs no operation. It is used for technical and theoretical reasons.

The Sequence operator ($S1 \blacktriangleright S2$) 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.

The Alternative operator or Mutual Exclusion operator ($S1 \blacktriangleleft\blacktriangleright S2$) is a composite service. When applied to a pair of services S1 and S2, it reproduces either the

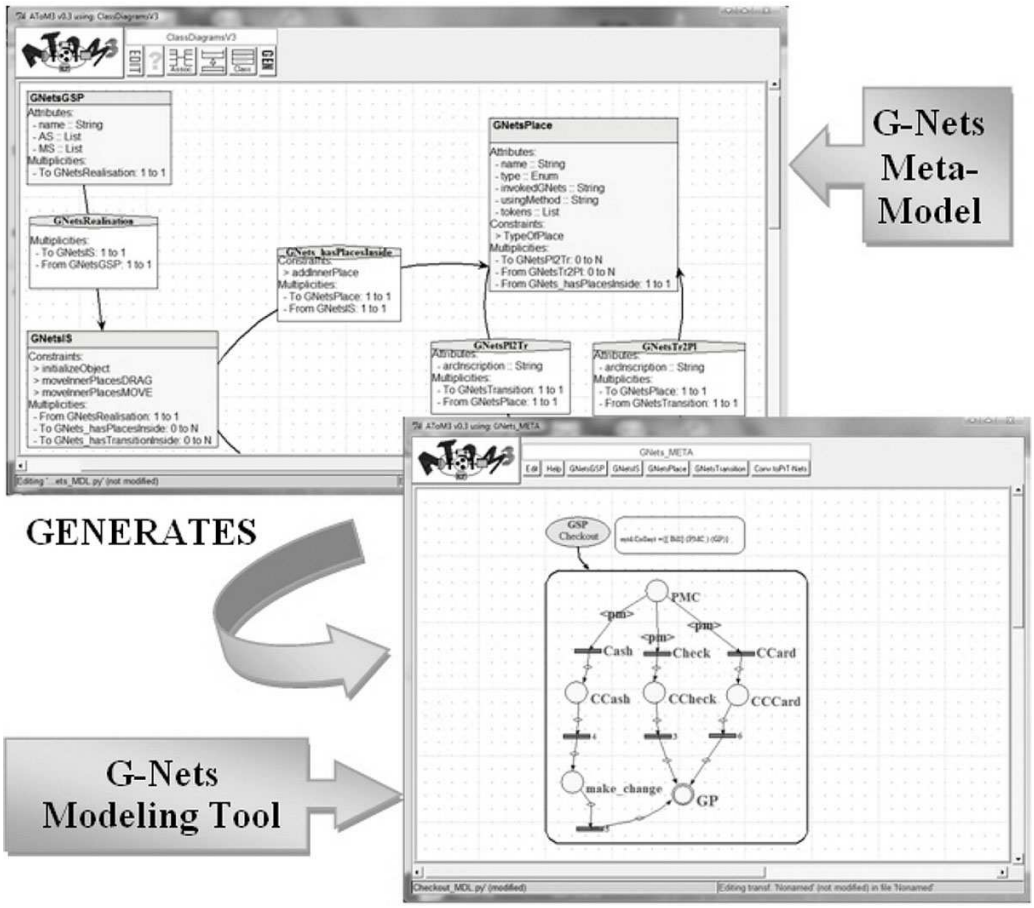

Figure 3. Customized modeling tool with $ATOM^3$

behavior of S1 or S2, but not both. For example the service Identification is followed either by the service Allow-access or the service Deny-access.

The Iteration operator ($\circlearrowleft S$) represents a composite service where one service is successively executed multiple times in a row. An example of use of this construct is when a customer orders from a service, a good a certain number of times.

The Arbitrary Sequence operator ($S1 \Leftrightarrow S2$) is an unordered operator that performs the execution of two services that must not be executed concurrently. This construct is useful when there is no benefit to execute services in parallel. For example when there is no deadline to accomplish the global task and the parallelism generates additional costs.

The Parallel Operator ($S1 \square S2$) builds a composite service. Given two services S1 and S2, it performs S1 and S2 at the same time and independently (without communication 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.

The Discriminator operator ($(S1 \boxdot S2) \gg S3$) is a composite service built on three services S1, S2 and S3. It submits redundant orders to different services performing the same task (S1 and S2 for example) and waits the outputs from S1 and S2. The first service (among S1 and S2) which responds to the request activates the service S3. All other late responses will be ignored. Note that S1 and S2 are performed in parallel and without communication. The main goal of this 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 Delegation operator ($Deleg(S1, o, S2)$), where o is an operation ($o \in O1$, O1 being the set of operations of S1) which is replaced by the ISP of another more specialized service (S2). In a given service, this operator is used to delegate a task to another service that has more abilities to execute it. This operator contributes to increase quality of service, enhances cooperation between enterprises and decreases the development efforts.

The Selection operator ($Select[S1 : Sn]$) is a complex operator that is applied to n services (S1,,Sn); it sends requests to different services through messages passed by their ISPs. According to the responses and rank criteria, the Selection operator chooses the best service between its competitors for performing a particular task that a company would to subcontract. This operator provides ways to maintain relationships with different suppliers which can offer different prices and provide different level of quality of service. It contributes then to increase the

independency of a company against its suppliers.

The proposed algebra verifies the closure property. This property 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 build more complex services by aggregating and reusing existing services through declarative expressions of service algebra. Semantics of the composition operators is characterized by description of the GSP and IS parts of the component services. We also focus on the dynamic behavior of the resulting service and this to address the Web service composition problem. Table 1 summarizes the G-Net algebra operators; in particular, it gives their syntax and formal semantics. The notations which are common to all the operators are:

- NameS is the name of the new service,
- Desc is the description of the new service,
- Loc is the location of the new service. It can be in the same server as one of the component service (s) or in a new server,
- URL is the invocation of the new service.

B. Operator's implementation

For the implementation of the composition process, we make use of meta-modeling and model transformation techniques based on the G-Net modeling language. The syntax of the class of models (G-Net) is graphically meta- modeled in an appropriate formalism, the Entity-Relationship Diagrams. Since the abstract syntax of the used models is graph-like, graph rewriting can be used to perform model transformation. Regarding to existing classification criteria, the kind of transformation that is applied in our approach is:

- Endogenous (in contrast with exogenous model transformation), since the meta- model used to express both the source and target models is the same and,
- Horizontal (in contrast with vertical model transformation), since the source and target models reside on the same level of abstraction.

To implement the G-Net algebra's operators, we have defined a graph grammar which consists in a set of transformation rules. The complete grammar includes twelve rules which can be applied to perform any operator present in a submitted composition formula. When the graph grammar execution finishes, we obtain a new G-Net service that models the composite service. Due to page limitations we show in Fig. 4 only three rules. In all of these rules, the nodes and different connections are labeled by numbers that identify them. These identifiers are used during the application of the rules.

If an identifier is present in both the left hand side (LHS) and right hand side (RHS) of a rule, the corresponding element (node or connection) will be preserved in the result. If this identifier appears only in LHS, the corresponding element will be deleted. If this identifier appears only in RHS, the corresponding element will be created. As we will see, the identifiers are also used

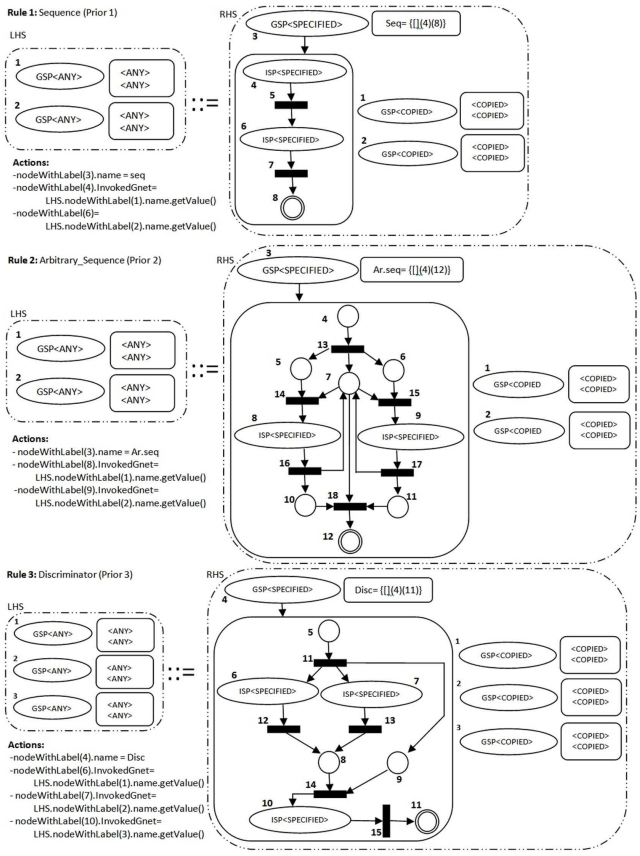

Figure 4. Some rules of the graph grammar for G-nets services composition

in the python code to compute the attributes values $' < SPECIFIED >'$. The elements attributes values in LHSs of the rules are compared with the elements attributes values of the host graph during the matching process. The first rule aims to implement the working of the ***Sequence*** operator. The LHS of the rule corresponds to the GSPs of the two G-Net operands. When representing only the G-Net interface (GSP), we make abstraction of the internal structure. In LHS, we have set all the attributes values to $< ANY >$. The RHS represents the resulting G-Net service. In this latter, the attributes of the nodes 3, 4 and 6 have the additional label $' < SPECIFIED >'$. This label specifies that the attribute value is computed by python code defined in the $'Actions'$. The code is executed only if the rule is applied and the computation of the value is based on the attributes' nodes of the LHS. For example, in the first rule, the action: $nodeWithLabel(4).InvokedGnet = LHS.nodeWithLabel(1).name.getValue()$ assigns the value of the attribute ***'name'*** of the node (1) to the value of the attribute ***'InvokedGnet'*** of the node (4). The two other rules are based on the same reasoning to perform ***Arbitrary Sequence*** and ***Discriminator*** operators.

Like the modeling process, the composition process is also performed with $AToM^3$, since this tool offers capabilities for model manipulation by graph transformation. The graph grammar presented above is stored in the

TABLE I.
THE G-NETS BASED ALGEBRA FOR WEB SERVICE COMPOSITION

Operator	*Syntax*	*Semantic*
Sequance	$S1 \blacktriangleright S2 = (NameS, Desc, Loc, URL, CS, SGN)$	$CS = CS1 \cup CS2$. $SGN = (GSP, IS)$ where $GSP = (MS, AS)\|MS = Mtd.seq\{[...](p1)\}$, $AS = \emptyset$; $IS = (P, T, W, L)\|P = \{p1, p2, p3\}$, $T = \{t1, t2\}$, $W = \{(p1, t1), (t1, p2), (p2, t2), (t2, p3)\}$, $L = \{(P1, Isp(S1)), (P2, Isp(S2)), (P3, goal)\}$.
Alternative	$S1 \blacktriangleleft\blacktriangleright S2 = (NameS, Desc, Loc, URL, CS, SGN)$	$CS = CS1 \cup CS2$. $SGN = (GSP, IS)$ where $GSP = (MS, AS)\|MS = Mtd.Alt\{[...](p1)\}$, $AS = \emptyset$; $IS = (P, T, W, L)$ where $P = \{p1, p2, p3, p4\}$, $T = \{t1, t2, t3, t4\}$, $W = \{(p1, t1), (t1, p2), (p2, t3), (t3, p4), (p1, t2), (t2, p3), (p3, t4), (t4, p4)\}$, $L = \{(P1, \tau), (P2, Isp(S1)), (P3, Isp(S2)), (P4, goal)\}$
Iteration	$\circlearrowleft S1 = (NameS, Desc, Loc, URL, CS, SGN)$	$CS = CS1$. $SGN = (GSP, IS)$ where $GSP = (MS, AS)\|MS = Mtd.iter\{[...](p1)\}$, $AS = \emptyset$;$IS = (P, T, W, l)$ where $P = \{p1, p2\}, T = \{t1, t2\}$, $W = \{(p1, t1), (t1, p1), (p1, t2), (t2, p2)\}$, $l = \{(P1, Isp(S1)), (p2, goal)\}$
Arbitrary Sequence	$S1 \Leftrightarrow S2 = (NameS, Desc, Loc, URL, CS, SGN)$	$CS = CS1 \cup CS2$. $SGN = (GSP, IS)$ where $GSP = (MS, AS)\|MS = Mtd.ar.seq\{[...](p1)\}$, $AS = \emptyset$; $IS = (P, T, W, L)$ where $P = \{p1, p2, p3, p4, p5, p6, p7, p8, p9\}$, $T = \{t1, t2, t3, t4, t5, t6\}$, $W = \{(p1, t1), (t1, p2), (t1, p3), (t1, p4), (p2, t2), (t2, p5), (p5, t4), (t4, p7), (t4, p3), (p7, t6), (t6, p9), (p3, t3), (p3, t6), (p4, t3), (t3, p6), (p6, t5), (t5, p3), (t5, p8), (p8, t6)\}$ $L = \{(P1, \tau), (P2, \tau), (P3, \tau), (P4, \tau), (P7, \tau), (P8, \tau), (P5, Isp(S1)), (P6, Isp(S2)), (P9, goal)\}$
Parallel	$S1 \square S2 = (NameS, Desc, Loc, URL, CS, SGN)$	$CS = CS1 \cup CS2$. $SGN = (GSP, IS)$ where $GSP = (MS, AS)\|MS = Mtd.par\{[...](p1)\}$, $AS = \emptyset$; $IS = (P, T, W, l)$ where $P = \{p1, p2, p3, p4\}, T = \{t1, t2\}$, $W = \{(p1, t1), (t1, p2), (t1, p3), (p2, t2), (p3, t2), (t2, p4)\}$ $l = \{(P1, \tau), (P2, Isp(S2)), (P3, Isp(S2)), (p4, goal)\}$
Discriminator	$(S1 \boxdot S2) \gg S3 = (NameS, Desc, Loc, URL, CS, SGN)$	$CS = CS1 \cup CS2 \cup CS3$. $SGN = (GSP, IS)$ where $GSP = (MS, AS)\|MS = Mtd.disc\{[...](p1)\}$, $AS = \emptyset$; $IS = (P, T, W, l)$ where $P = \{p1, p2, p3, p4, P5, P6, p7\}$, $T = \{t1, t2, t3, t4, t5\}$, $W = \{(p1, t1), (t1, p2), (t1, p3), (t1, p5), (p2, t2), (p3, t3), (t2, p4), (t3, p4), (p4, t4), (p5, t4), (t4, p6), (p6, t5), (t5, p7)\}$ $l = \{(P1, \tau), (P4, \tau), (P5, \tau), (P2, Isp(S1)), (P3, Isp(S2)), (P6, Isp(S3)), (p7, goal)\}$
Delegation	$Deleg(S1, o, S2) = (NameS, Desc, Loc, URL, CS, SGN)$	$CS = CS1 \cup CS2$, $SGN = (GSP, IS)$ where $GSP = (MS, AS)\|MS = MS1$, $AS = AS1$; $IS = (P, T, W, L)\|$ $P = P1 \backslash L^{-1}(o) \cup Isp(S2)$, $T = T1$, $W = W1 \cup \{(t, Isp(S2))\|t \in \bullet L^{-1}(o)\} \cup \{(Isp(S2), t)\|t \in L^{-1}(o)\bullet\} \backslash \{(p, t)\|p \in L^{-1}(o)\} \backslash \{(t, p)\|p \in L^{-1}(o)\}$ $L = L1 \cup \{(Px, Isp(S2))\}$
Selection	$Select[S1 : Sn] = (NameS, Desc, Loc, URL, CS, SGN)$	$CS = \bigcup_{i=1}^{n} CS_i, SGN = (GSP, IS)$ where $GSP = (MS, AS)\|MS = Mtd.Select[](p1), AS = AS_{n+1}$; $IS = (P, T, W, L)\|$ $P = p_1, ..., p_{2n+3}$, $T = t_1, ..., t_{2n+2}$, $W = (p_1, t_1) \cup \bigcup_{i=2}^{n+1}(t_1, p_i) \cup \bigcup_{i=2}^{n+1}(p_i, t_2) \cup (t_2, p_{n+2}) \cup \bigcup_{i=1}^{n}(p_{n+2}, t_i) \cup \bigcup_{i=3}^{2n+2}(t_i, p_{n+i}) \cup \bigcup_{i=n+3}^{2n+2}(p_i, t_i) \cup \bigcup_{i=n+3}^{2n+2}(p_i, t_{2n+3})$, $L = \{(P1, \tau)\} \cup \{(p_2, Isp(S1.req)), ..., (p_{n+1}, Isp(Sn.req))\} \cup \{(p_{n+2}, SelectService)\} \cup \{(p_{n+3}, Isp(S1.mtd)), ..., (p_{2n+2}, Isp(Sn.mtd))\} \cup \{(p_{2n+2}, goal)\}$

user area of $ATOM^3$. The actions associated to each rule are specified by Python code. Fig. 5 illustrates the implementation of the first rule in $AToM^3$. The reader can see the LHS and RHS of the rule as well as the python code (in the top of the figure) which specifies the action discussed above. The composition process starts by the importation of G-Net services previously modeled. The user enters the composition formula according to the defined algebra. Since the formula may involve several operators, the corresponding rule(s) of each operator is (are) applied to the imported operand services. Consider the example composition scenario that occurs when a customer wants to get a product as soon as possible. The customer submits redundant orders to two Provider services. Once he obtains a response from the fastest service, he starts the payment procedure. This scenario can be performed using the ***Discriminator*** operator. The payment is achieved by the ***Checkout*** G-Net service presented above and the providers are modeled by the G-Net services ***Provider1*** and ***Provider2***. The ***Provider1*** and ***Provider2*** services start by checking availability of the required product (CA). If the product is available, they make a bill and send it to the customer. In the case where the product is not available, they trigger their respective restock procedure and recheck availability.

In Fig. 6, we present the services composition using our graph transformation based tool. The three services being modeled and imported in the tool, the user enters the formula (***Provider1*** $\boxdot$ ***Provider2***) $\gg$ ***Checkout*** . $ATOM^3$ applies our graph grammar. At the end of grammar execution, we obtain the composite service Disc shown in the right side of Fig. 4. We can see that Disc invokes Provider1 and Provider2 through their ISPs. The first service which responds to the request activates the Checkout service.

VI. VERIFICATION PROCESS

WS-mcv methodology deals with the formal verification in order to test and repair design errors even before actual running of the (composed) service. The intention is to raise reliability of Web service composition by ensuring that a composite G-Net service will behave as required by its specification and that the system and its components contain no errors or behavioral anomalies (such as deadlock and livelock). To perform formal verification on systems modeled by Petri-Nets like languages, current

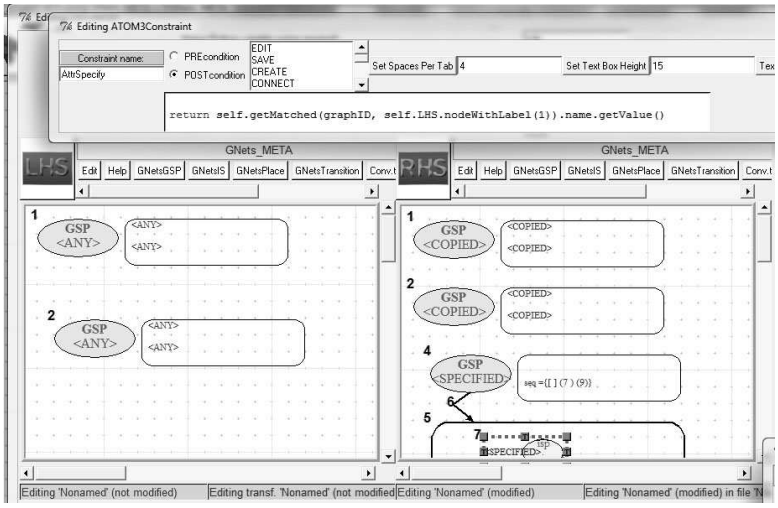

Figure 5. The first rule in $ATOM^3$

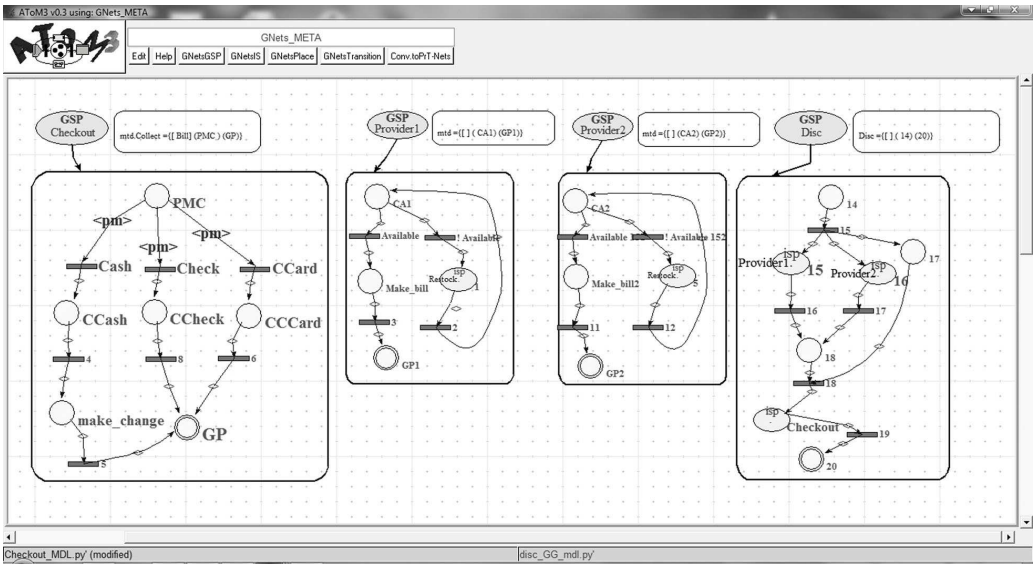

Figure 6. G-nets services composition using our tool

researches exploit the powerful features of reachability graph analysis techniques. Since there is no reachability analyzer tool for the G-Net framework, we make use of translation rules which enables to transform G-Net specifications into their equivalent Predicate/Transition Nets (PrT-Nets). This transformation is performed in order to exploit an existing tool with a variety of analysis techniques for PrT-Nets namely PROD reachability analyzer [27]. PROD creates a reachability graph of a system modeled as PrT-Nets. From that graph, users can search for terminal nodes, path leading to that terminal nodes and path which satisfy some given properties. The complete manual of PROD can be found in [28]. Unlike other approaches, WS-mcv methodology allows performing verification before and after the composition in order to detect if the anomalies occur in the component services or in the composite one. The two phases of Pre and Post verification occur in the same way except that the former concerns the operand services and the latter concerns the resulting service. In this way we guarantee a correct Web service modeling and composition. The verification process is then divided in two tasks: 1) the transformation of a G-Net service into an equivalent PrT-Net and 2) the translation of the obtained PrT-Net into PROD description.

A. G-net/PrT-net transformation

To perform this task, we still use MDE techniques in order to make model transformation. The works of [12] present a graph transformation based framework that allows transforming a G-Net specification into its equivalent PrT-Net using a defined graph grammar. This latter has a slight inconvenient as, when applied to a source model, it progressively deletes it. As a consequence, we have improved these grammar rules in order to preserve the source model (G-Net model). We propose to exploit the modified grammar in $ATOM^3$ environment. Once we provide our tool the meta-model of the PrT-Net formalism

[12] and the modified G-Net/PrT-Net grammar, it can then support the two formalisms of G-Net and PrT-Net. It can also automatically generate PrT-Net specifications from G-Net ones. The verification process starts by specifying the G-Net service the user wants to analyze. The transformation rules are then applied to the G-Net model. Once it finishes, we obtain an equivalent PrT-Net specification. This transformation is illustrated by the user interface of $ATOM^3$ in Fig. 7. In the left side of the figure, we can see the ***Provider1*** service in the G-Net formalism. The right side of the same figure represents its equivalent resulting PrT-Net.

B. Prt-net/PROD net description transformation

To perform the analysis using PROD, we need to convert the PrT-Net specification into PROD's Net description language. This language is C preprocessor language extended with net description directives. PROD compiles this net description and generates the full reachability graph. However, at present time, this task is still manually performed. Later, when the user interface of our tool will be complete, the user doesn't have to be familiar with PROD. Using the reachability graph, we can verify many important properties such as boundedness, liveness and reachability which can be used as general criteria of correctness of composition. The result of this transformation is illustrated by Fig. 8. This figure represents the resulting PROD net description file of the service ***Provider1*** previously described in PrT-Net formalism.

VII. CONCLUSION

In this paper, an efficient model-driven methodology for Web service composition has been presented. The proposed methodology offers solutions for both modeling existent services, successfully composing them and verifying their correctness. The main contributions of this paper are:

- The definition of a set of modeling rules for Web service specification into G-Net concepts.
- The proposition of a G-Net based algebra that allows combining G-Net services by means of basic and complex operators.
- The formal definitions of G-Net services as well as the introduced operators.
- The implementation of the proposed operators by an efficient graph grammar.
- The specification of a verification method to ensure composition correctness.

All the phases of our methodology have been realized under different processes. The modeling and composition processes have been implemented with a customized visual tool which allows editing and manipulating models in the G-Net formalism. The verification process, which is partially automated, is based on model transformations performed by $ATOM^3$ to produce models that can be verified by PROD. To the best of our knowledge, WS-mcv is the only approach that makes use of the G-Net framework to provide a complete solution for Web service composition. Compared to other Petri Nets based approaches, ours presents several advantages. It requires less effort when modeling complex services and produces more reduced models. Furthermore it offers a visual tool and deals with the formal verification. In future work, we will propose to meta-model the WSDL description language and to define a graph grammar which allows translating Web services described in WSDL language into equivalent G-Net services. Then we will extend our tool with a new module that can import existing services in WSDL and automatically model them into G-Net concepts. We also plan to improve the verification process by automating the transformation task from PrT-Nets to PROD's Net description language. This will avoid users to be familiar with PROD Net descriptions.

REFERENCES

[1] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the web services web an introduction to soap, wsdl, and uddi," *IEEE INTERNET COMPUTING*.

[2] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web services description language (wsdl) 1.1," Mar 2001, [Online]. Available: http://www.w3.org/TR/wsdl.

[3] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, , H. F. Nielsen, S. Thatte, and D. Winer, "Simple object access protocol (soap) 1.1," May 2000, [Online]. Available: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[4] R. Hamadi and B. Benatallah, "A petri net based-model for web service composition," in *proc. the 14th australasian database conference*, adelaide. Darlinghurst: Australian Computer Society, 2003, pp. 191–200.

[5] G. Yubin, D. Yuyue, and X. Jianqing, "A cp-net model and operation properties for web service composition," *Chinese Journal of Computers (Chinese edition)*, vol. 29, Number 7, p. 10671075, 2006.

[6] Z. Zhang, F. hong, and H. xiao, "A colored petri net-based model for web service composition," *Journal of Shanghai University (English Edition)*, vol. 12, Number 4, pp. 323–329, 2008.

[7] X. Feng, Q. Liu, and Z. Wang, "A web service composition modeling and evaluation method used petri net," in *Proc. APWeb Workshops*, 2006, pp. 905–911.

[8] R. Akkiraju and B. Sapkota, "Semantic annotations for wsdl and xml schema usage guide," (2007), [Online]. Available: http://www.w3.org/TR/sawsdl-guide/.

[9] J. Bruijn, D. Fensel, U. Keller, H. M Lausen, R. Krummenacher, A. Polleres, and L. Predoiu, "The web service modeling language wsml," 2005, [Online]. Available: http://www.wsmo.org/wsml/.

[10] D. Martin, M. Burstein, J. Hobbs, and al, "Owl-s: Semantic markup for web services," [Online]. Available: http://www.w3.org/Submission/OWL-S/.

[11] Y. Deng, S. K. Chang, J. C. A. De Figueiredo, and A. Psrkusich, "Integrating software engineering methods and petri nets for the specification and prototyping of complex information systems," in *Proc. The 14th International Conference on Application and Theory of Petri Nets*, Chicago, June21–25, 1993, pp. 206–223.

[12] E. H. Kerkouche and A. Chaoui, "A formal framework and a tool for the specification and analysis of g-nets models based on graph transformation," in *proc. of International Conference on Distributed Computing and Networking CDCN09*, India, January 2009, p. 206211.

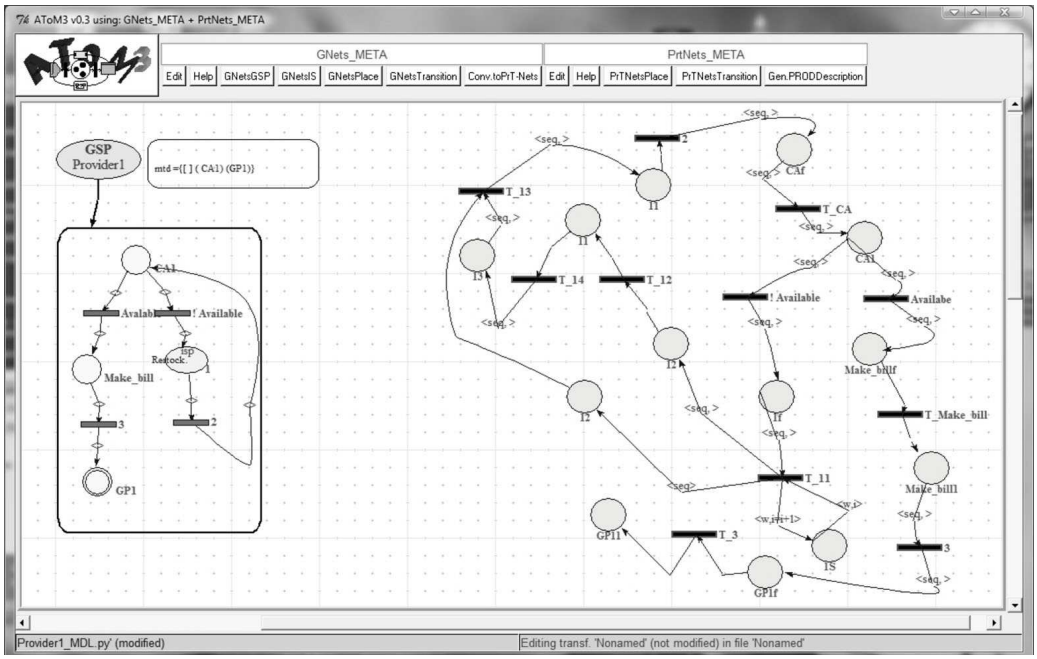

Figure 7. Provider1 G-net service and its equivalent PrT-Net

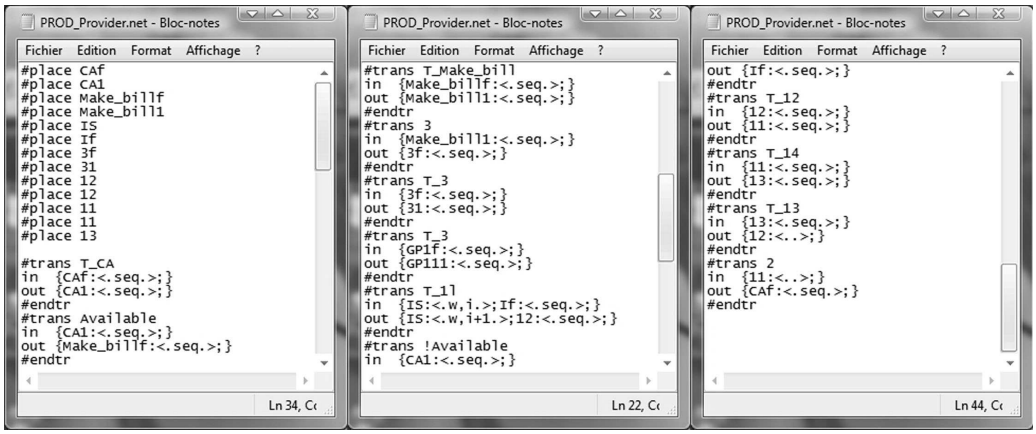

Figure 8. PROD net description for the Provider1 PrT-Net

[13] H. J. Genrich and K. Lautenbach, "System modeling with high level petri nets," *Theorical Computer Science*, vol. 13, pp. 109–136, 1981.

[14] T. He and L. Li, "Research on verification tool for software requirements," *JOURNAL OF SOFTWARE*, vol. 07, Issue 7, pp. 1069–1616, JULY 2012.

[15] M. Ter Beek, A. Bucchiarone, and S. Gnesi, "Formal methods for service composition," *Annals of Mathematics, Computing and Teleinformatics*, vol. 1, Issue 5, pp. 1–10, 2007.

[16] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, . *Business Process Execution Language for Web Service (BPEL4WS) 1.0*. Published on the World Wide Web by BEA Corp, IBM Corp and Microsoft Corp, Aug 2002.

[17] Z. Liu, A. Ranganathan, and A. Riabov, "Modeling web services using semantic graph transformations to aid automatic composition," in *IEEE International Conference on Web Services (ICWS 2007)*, Salt Lake City, Utah, July13–19, 2007.

[18] B. Srivastava and J. Koehler, "Web service composition - current solutions and open problems," in *ICAPS 2003 workshop on Planning for Web Services*, July22 2003.

[19] B. Li, Y. Xu, J. Wu, and J. Zhu, "A petri-net and qos based model for automatic web service composition," *JOURNAL OF SOFTWARE*, vol. 07, Issue 1, pp. 149–155, JANUARY 2012.

[20] K. Jensen, "Coloured petri nets- a high level language for system design and analysis," in *Lecture Notes in Computer Science 483. Advances in Petri Nets 1990 Springer-verlag*, 1990.

[21] C. A. Petri, "Kommunikation mit automaten (in german)," Ph.D. dissertation, University of Bonn, Germany, 1962.

[22] A. Perkusich and J. C. A. De Figueiredo, "G-nets: A petri net based approach for logical and timing analysis of complex software systems," *Journal of Systems and Software*, vol. 39, Issue 1, pp. 39–59, Oct 1997.

[23] J. De Lara and H. Vangheluwe, "Atom3: A tool for multi-formalism modelling and meta-modelling," in *proc. of European Conferences on Theory And Practice of Software Engineering ETAPS02*, 2002, p. 174 188.

[24] Google, "Google maps api web service," (2005), [Online]. Available: From: http://code.google.com/intl/com/apis/maps/.

[25] S. Narayanan and S. McIlraith, "Analysis and simulation of web services," *Computer Networks*, vol. 42, Number 5, pp. 675–693, 2003.
[26] D. Zhovtobryukh, "Context-aware web service composition," Ph.D. dissertation, University of Jyvaskyla, Finland, 2006.
[27] PROD, "Prod: An advanced tool for efficient reachability analysis, version 3.4.01." 1995, [Online]. Available: http://www.tcs.hut.fi/Software/prod/.
[28] K. Varpaaniemi, J. Halme, K. Hiekkanen, and T. Pyssysalo, Helsinki University of technology, Tech. Rep.

Fayçal Bachtarzi is currently a Ph.D. candidate at Mentouri University of Constantine, Algeria. He received his Master in computer science from the same University in 2010. His research interests include web service compostion, model driving engeneering, formal verification and distributed systems.

Allaoua Chaoui is full Professor with the department of computer science, Faculty of Engineering, University Mentouri Constantine, Algeria. He received his PhD degree in 1998 from the University of Constantine (in cooperation with the CEDRIC Laboratory of CNAM in Paris, France). His research interests include Mobile Computing, formal specification and verification of distributed systems, and graph transformation systems.

Elhillali Kerkouche is Associate Professor in the department of Computer science, University of Jijel, Algeria. His research field is Formal Methods and Distributed Systems.