Research on Verification Tool for Software Requirements

Tao He
Software Engineering Department, Shenzhen Institute of Information Technology, Shenzhen, China
Email: he_tao@foxmail.com

Liping Li
Computer and Information Institute, Shanghai Second Polytechnic University, Shanghai, China
Email: llping2000@yahoo.com.cn

***Abstract*—To verify the software requirements of network software, a verification tool OWLSVerifyTool is proposed, designed and developed to deal with model checking of Web service composition model in this paper. It can convert OWL-S documents into Petri nets document and then analysis and verify it in Petri nets with engine in dynamic context. While compositing the DL reasoning engine Pellet and F-logic-based reasoning engine Flora-2, it can play their respective advantages to reason and verify static model in static context of software requirement. The automated validation tool can effectively verify software requirement meta-model based on Web service described with OWL-S.**

***Index Terms*—Web Service, Software Requirement, Verification Tool**

I. Introduction

The network software disposed in network environment is a kind of special ultra-large service-oriented computation of complex software system. Requirement engineering of Network software are facing many problems at present [1-4], because of its dynamic topology, uncertainty of users, and its continuous increasing requirements. However, present software requirement modeling and verification technique lack enough support to service-oriented computation, and are unable to gather the Web service resources in the network effectively, provide high dependable Web service resources to enhance development of information system efficiency [5-7].

Requirement verification is an important process in software requirement engineering. If without it, the project may lead to be unsuccessful. The design and development of verification tool is very important, for its enhancing efficiency of verification, improving software development process, and guarantee software quality. Semantic Web service language has carried on the clear description to Metamodel various levels of software requirement, and carries on the formalized modeling by Petri net and F-logic, and verifies the uniformity of Web service semantic restraint.

Verification of Web service combination has two kinds of research methods at present: verification based on work flow BPEL4WS and based on semantic OWL-S. Now regarding the latter there are few research achievements. In literature [8] for controls flow and the data flow on the modeling, it transforms directly the OWL-S process model into the simpler Promela modeling, and verifies it with SPIN. However, its data flow modeling is too simple to verify whether the input /output type match.

Model checking methods successfully applied to a large extent with support of the automated tools. Currently the well-known OWL and OWL-S model checking tools are: KAON2,FaCT++,Racer and so on. But existing parsing/validation tools have the flaw that verified nature are incomplete, the verifying and inference mechanism is simple, emphasis on OWL-S static model validation and neglect dynamic model validation. To this end, it is necessary to develop a new tool for model checking.

This paper has designed OWL-S model examination prototype tool OWLSVerifyTool. It takes OWL-S storage documents (*.owl or *.xml) as the input, simultaneously carries on dynamic model and static model verification. The dynamic model, transforms with the document format switch to the PNML document, then directs or transforms the PNML document to corresponding form, then input to Petri net verification DiNAMiCS and Tina engine to verify it. The static model, combining description logic DL inference and Flora-2 rule, takes OWL-S storage documents(*.owl or *.xml) as input, simultaneously unifies DL inference (Pellet) and the Flora-2 rule to carry on inference alternately, carries on the analysis verification of the static model, and output the results.

II. General Organization of Verification of Verification Tool

OWL-S verification tool OWLSVerifyTool is mainly composed of dynamic model verification and static model verification modules. The designing structure of verification tool is shown in Figure 1.

doi:10.4304/jsw.7.7.1609-1616

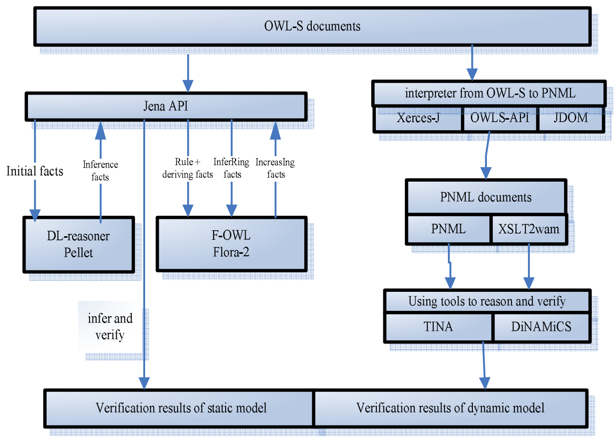

Figure 1. General architecture diagram of validation tool

OWL-S is the Web service frame described in OWL language, but OWL represents ontology by class, attribute, value, and concept relations and so on. When describing RGPS software requirement, it often uses essential factor and relational of standard ontology, a series of axioms as well as the formal restraint semantic to build its model. This paper calls it “static model”. A static model, often not evolving, can only express invariant and special condition. formalism methods such as Z language, DL, VDM, F-logic, which are based on set theory and the first-order predicate calculus, may use for modeling it. This paper uses DL and F-logic. To verification of static model, its realization of definition standard, accuracy, uniformity and completeness of inference must be considered.

In the static model verification module it transforms the OWL-S documents through Jena API. First, inputs it with rules to the Pellet engine to infer for obtaining the new fact, then combine the new fact and the original fact, transforms it to the form that the engine Flora-2 can accepts. Eventually, inputs it with rules to engine Flora-2 to infer and verify. The output is the verification result of static model.

Uses the Petri net in the dynamic model aspect to take the verification model, and carries on the verification with DiNAMiCS and Tina engine; mainly verify its accuracy (activeness, boundedness), Reachability, final state, security, and so on.

As shown in Figure 1, first inputs OWL-S documents in unified user interface. Transforms the OWL-S documents into the PNML documents in the dynamic model verification module with interpreter respectively, simultaneously carries on the PNML documents verification. Because the input form of DiNAMiCS engine is different from PNML slightly, which is the wam form, first transforms it into wam form by XSLT, and then inputs it to DiNAMiCS to carry on inference and model checking. Tina may directly input PNML form documents to carry on it. After verification by two engines, merge the results to obtain dynamic model verification result of the OWL-S documents.

III. General Organization of Verification of Verification Tool

A. Interpreter of the transforming from OWL-S to PNML

This interpreter can carry on analysis of OWL-S service and transform it into PNML form. For the transformation from OWL-S service to Petri net can reuse many methods, algorithms, and reuse tools to inspect the equivalence of Petri net (for example literature [9-11]). Narayanant and McIlraith have first defined the Petri net semantics of DAML-S in [12] (the OWL-S preceding edition). However, their semantics is not the combinatorial property for it is unable to process any-order control structure. DaGen tool [13] transforms Petri net semantics of DAML-S description in [12] to referring Petri net. DaGen inserts to a Reference Net Workshop (Renew), and causes the Petri net simulator execution of Reference Network as well as the graph draw. However, DaGen has not had the intermediate Petri net file of transform. The interpreter described in this paper can transform OWL-S process model expressing service behavior to Petri network described by PNML format.

When execute the reasons, how to share the input/output data in different process? In fact, when input and output data are shared by many processes, the OWL-S process model may be defined by the input/output binding mechanism. The interpreter deals with this problem by carrying out a suitable analysis sentence of OWL-S process model.

This paper comes through the XML resolver to transform OWL-S documents to the PNML form. The interpreter from OWL-S to the PNML is a Java Servlet. Input a URL pointing to OWL-S service description (or file system path) from Web client of the interpreter, and sends it to the serve, analyzing by the Servlet in the background conversion, and returning the Petri net describing in PNML of OWL-S service.

B. PNML verification and XSLT transformation

1) PNML Correctness Verification module: To change the default, adjust the template as follows.

Through to the OWL-S documents' transformation, the Web service which the OWL-S documents describe definitely may use the Petri net simulation and indicate by the PNML document that like this may verify the Web service operation flow which using the Petri net's correlation analysis method and the tool the OWL-S documents describe whether to have in the flowage structure design question. Therefore the next stage is transforms, but results in the PNML document hands over by PNML Correctness the Verification module processes. PNML Correctness Verification module construction is like chart 2.

PNML Correctness the Verification module contains the PNML resolver, the accurate verification, the security, the Reachability, the deadbolt lock, finally the shape, and the durable verification and so on. PNML Parser is responsible for the PNML document which analyzes transmits. The accurate verification, the security, the Reachability, the deadbolt lock, the shape, the durability through the execution coverage diagram, may reach analysis methods separately finally and so on chart, incidence matrixes, condition agenda, migration matrix to carry on the verification. But the PNML document's proving program is first starts by the accuracy and the secure nature, next is the Reachability nature, and finally is the deadbolt lock, the final state and the durable nature. So long as this verification step has an item will be unable through to transmit makes a mistake harms the information and stops the entire proving program.

PNML Correctness Verification module contains:

*PNML decoder

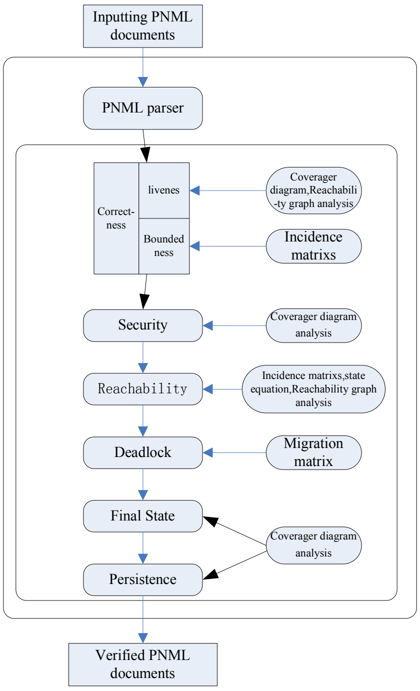

Figure 2. PNML Correctness Verification module structure

PNML Parser is responsible to receive the PNML document after OWL-S file conversion. First analyzes the Web service operation flow which describes in the PNML document, again storehouse which describes the PNML document, migration and arc by array way storage. PNML Parser through analyzes the PNML document and separately the storehouse, the migration and the arc by the array form storage, will then transmit these three objects by the parameter way for the nature proving program. The nature proving program after receiving the Petri net model the image parameter the basis itself uses again the analysis method, constructs by the parameter in information may reach the tree, the incidence matrix and so on mathematics type.

* The secure verification

For Web service composition, the tool will use cover tree analysis to verify the Petri-Net model of the security property. Therefore realizes the cover tree and the coverage diagram method application procedure analysis in the PNML accuracy verification proxy service combine the Petri-Net model after PNML the resolver analysis Web.

* Reachability verification

The Petri-Net model regarding the Web services the Reachability nature to use the incidence matrix, the equation of state or may reach the chart the analysis method verification. Therefore is by realizes the incidence matrix and the equation of state method application procedure analysis in the PNML accuracy verification proxy service combines the Petri-Net model after PNML the resolver analysis Web.

C. Calculation method verification tool DaNAMiCS and Tina

DaNAMiCS is a modeling verification tool, can use for to analyze the Petri net and to color the Petri net, and can reduce the modeling grid complexity. DaNAMiCS includes suppressing the arc the support to help a system model the foundation. The DaNAMiCS important merit is that it supports some analysis tool and the method, for instance matrix invariant and migration matrix, structure analysis, as well as some simple and advanced performance analysis. Because DaNAMiCS has compared to other OWL-S verification tool more analysis tools, we use XSLT to transform the PNML documents are the wam forms, like this can induct them to DaNAMiCS.

Tina (time Petri net analyzer) is a tool that analyzes the Petri net and a time Petri net. It may construct and reach the chart and can carry on the analysis to the Petri net's structure:

The accurate analysis confirmed that a system's integrity is maintained, it including analyzes net's activeness and the boundedness. We will use the accuracy to represent the net to be live and have, and with the accuracy explained that this model net expressed the correct system.

Besides these essential attributes, we must confirm some other attributes, including net whether to contain the final state, net whether safe (security), whether it is lasting (durability) and so on.

1) Analysis tools and methods:

- Coverage diagram

We must inspect the construction algorithm of the coverage diagram. It will need to renew, so that processes increases newly suppresses the arc and has the capacity storehouse institute order of complexity. Carry out the algorithm that must be correct and the most superior movement.

* Analyze the correctness with invariant

DaNAMiCS will be able to calculate the incidence matrix. It will be able to determine P- and the T- invariant from the incidence matrix. This may use for surveying the boundedness and the deadbolt lock that does not exist.

* Analyze the correctness with coverage analysis

The user will be able to choose the option in DaNAMiCS, which domain through assigns the functional analysis to need to investigate. The coverage diagram will be produced for the accuracy and tests.

- Coverage analysis

In many situations, the invariant analysis cannot produce about the Petri net model accurate conclusion. It needs to carry on the spreadability analysis. It is a two stepped process. The first stage is the coverage diagram construction. The coverage diagram is all may reach marking the set. The second section is this stage analysis. The first stage is called the coverage diagram production, has the greatest time consumption and the complexity; the second section analyzes the section, it will process afterward.

The spreadability analysis goal produces a coverage diagram. The ordinary Petri net's cover analysis is obtains through the following direct forward algorithm. However, because of DaNAMiCS a Petri net's expansion class, namely contains suppresses the arc the net, the time and the immediate migration, and the storehouse capacity, the coverage diagram structure including. We have to construct our own algorithms, revise and greatly expand the initial algorithm to find the overlay network.

2) The nature analysis of Coverage diagram:

After discussion chart product, now it will analyze the coverage diagram to appraise the following nature, such as activeness, boundedness, final state, security and durable process and so on.

a) Activity

A Petri net most important nature is an activeness, this relates judges some system whether to collapse or whether systematic some part infinite loop. A Petri net's live performance through the following two rules, but determined from the coverage diagram:

* If a net is live and has, then it has the strong connection coverage diagram.

* A net has, the net is lives, and when only all migration in the coverage diagram connects in the module the demonstration is a label finally at least. A strong connection chart is the random point may arrive in the chart from the chart through a series of ways other all point charts. A chart's final module is a series of points. A strong connection chart has a correct final module. Loads the above two principles to carry on the spreadability analysis to a useful form, we may say, if each migration can cause a coverage diagram all final module's marking to enable, then the net is active. Thus, the definite active question became finds connects a module's question finally, this was the algorithm question which easy to understand. Abbreviate this algorithm specific code here.

b) Boundary analysis

The net has boundary. If in a Petri nets (N, M0), institute's token quantity of each storehouse to may reach marking willfully from M0 not to surpass limited number k, then we should say that the Petri nets is K has or has simply.

c) Terminal analysis

The final state is that in Petri net chart a marking may arrive from each other. This is very important to the software, namely, regardless of the current condition is anything, can always arrive at the ultimate objective. The final state existence is easy to calculate. If final, strong connects module's quantity to be equal to 1, then this Petri net contains the final state.

d) Security analysis

If a Petri net does not have the storehouse to be able in the net the packet of energy including an above request to be willing, then calls it safely. This is the Petri net very important attribute; net's storehouse in the system is the condition mark. If the storehouse contains a request to be willing, then the condition is effective, otherwise the condition is untenable. A storehouse contains is more than a request to be willing not to have the logical significance in these net's type, usually the expression somewhere has a mistake in the design. As mentioned above, if in a Petri net's storehouse institute the request is willing quantity are most, then it is safe. The security may be determined by all mark of linear search chart simply. If has not met has the storehouse contains an above request to be willing a marking, then this net is safe.

e) Durability

If a Petri net enables the migration to random two, an initiation's migration ever does not forbid other migration to enable, then calls it lastingly. If a lasting net's migration enables, it will maintain enables to initiate until it.

IV. DL and F-Logic Verification Module

The OWL language is based on description logic (DL), uses in the knowledge which the concept code and the concept inherit. Description logic is the first-order logical subset. This paper is FOL subset design inference Flora-2 designs the OWL-S inference engine with one, can carry on the inference effectively in the FOL expression subset, moreover understand easily and use.

A. *Specific organization illustration*

Like Figure 3, the static model inference verification module combined based on DL inference engine Pellet and based on F-logic inference engine Flora-2, displays its respective superiority to carry on the inference and the verification.

In This paper develops the method uses DL inference (Pellet) and F-logic system (Flora-2) pair of OWL the DL ontology carries on a rotation inference process, this ontology frame supports by Jena the semantic Web. In order to unify both's merit, this paper and a series of F-logic rule apply alternately the DL inference in the ontology. During this process, the ontology exchanges in two systems between. Therefore an OWL DL subset may transform is F-logic, may also carry on the reverse transformation.

Pellet+Flora-2 combination mainly based on ontology language OWL DL. This “the DL territory” is constitutes “the F-logic territory” by the F-logic rule the expansion. The special place of combination is two domain hosts - from the relations: the OWL ontology takes a host (ontology); it uses F-logic to take from (ontology) supports the tool. Generally the process may be described as follows: First, the DL inference with the existing knowledge reasoning new things information, outputs the ontology together with the F-logic rule in an output subset. If this inference obtains the recent information, then this recent information is added to the original ontology, becomes the new expansion the ontology, at the same time will contain has the new expansion ontology process to restart.

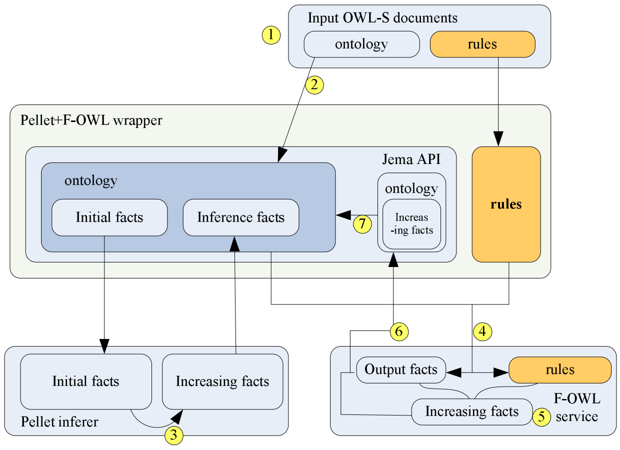

Figure 3. Architecture and evolution strategy.

The ontology through and the F-logic rule inferential reasoning increases the recent information in the entire process from the DL inference, and evolves gradually. The F-logic system uses a unidirectional knowledge library, for example the system uses the rule receive output fact which in each circulation defines. Therefore needs to define as far as possible much knowledge in the ontology scope and a clearing house needs the fact. The F-logic rule only uses for to infer these not to be able by OWL the DL direct processing inference duty.

a) Input components:

According to states the frame, the Pellet+Flora-2 input by following constitutes:

Ontology: OWL the DL ontology, it uses the form which Jena supports to carry on the code. It should contain possesses infers kind and the attribute definition by the F-logic rule.

Rule: In order to derive the new fact, but assigns a series of F-logic rule. These rules code with the XML mark. They are transformed through the Pellet+Flora-2 system F-logic. These input module uses in inferring the process alternately.

b) Calculation strategy:

(1) The input is composed of two parts: A series of F-logic rule and an OWL-S ontology. Two modules both load to the Pellet+Flora-2 inducing equipment.

(2) The Jena frame use assigns the ontology to construct the model; this model binds to OWL DL on inference (Pellet). At the same time, the rule transforms from the XML form to the regular F-logic grammar.

(3) DL inference (Pellet) uses for from the model which establishes to infer the recent information. The new fact becomes the ontology a part and is verified with heavy responsibility for the primitive fact. If the inference cannot further infer the new things information, the process continues the next step.

(4) Output (initial and inference) an application fact subset, and transforms is F-logic. Transforms the fact and the F-logic rule submits together to the F-logic system.

(5) In order to infer the recent information, the F-logic system applies the F-logic rule which in the knowledge library defines.

(6) The F-logic knowledge library submits to the Pellet+Flora-2 inducing equipment and transforms an OWL compatible grammar. The transformation knowledge library uses for to construct a new temporary ontology.

(7) Jena the frame inspects whether all new ontology's information has defined in the old ontology or certain information whether is new. If is the latter, then two main bodies both must merge, for example, increases the recent information to expand the ontology to the old ontology at the same time process 3 to restart in the step. If possesses by the Flora-2 inferential reasoning information already is this ontology part, and does not have to discover the information again, then the process terminates.

Describes the process produces an evolved the ontology. Each circulation increases some recent information, enables the next circulation to push causes more information. Finally, DL inference (Pellet) and the F-logic system can discover the recent information. Both construct the model, the ontology as well as the output subset, they are complete and stable. But if has the new rule definition or new fact increase, has the possibility to need to increase the circulation, until achieves a new steady state.

The OWL test defines an OWL parallel shot with the example documents to be as follows: A OWL uniform checker takes documents the input, the returns are consistent, inconsistent or the unknown result.

- Uniformity check

This inspection causes ontology to remove willfully the contradictory fact. The OWL abstract syntax and the semantic documents provide an ontology uniform formalization definition, but Pellet uses this formalized definition to carry on the inspection. In the DL terminology, this is inspects one (this is equal about Tbox to an OWL uniform checker) the Abox uniform operation.

- Concept satisfaction

This inspects one kind whether to have the possibility to have the random example. If the class does not satisfy, it will then define a kind of example to cause the entire ontology not to be inconsistent.

- Classification

Calculate each naming class between the subclass relations to found the class to inherit completely. The class inherits may use for to reply inquires, thus obtains kind of all direct subclasses or the only direct subclass.

- Realization

Found one individual respective most special kind; or in other words, for each individual account direct type. Because the direct type is the definition which inherits about one kind, therefore realizes can only after the classification carries out. The use classification inherits, may also obtain all types for this individual.

c) Combination of Jena and Pellet inference engine:

Jena is uses for to construct the semantic net specially the application software, it was RDF, RDFS and OWL has provided the environment which realized programmable. The inference function is in a Jena sub-system. Jena provides the inducing equipment with Racer, FaCT, Pellet and so on is also same, is aims at the ontology the inducing equipment, but Jena is in itself not “the inducing equipment designs the expert”, its oneself contains the inducing equipment basically is one kind of CLISP coordination ontology domain production pattern rule forward reasoning system. Therefore, its operating efficiency is not very high. Front end but it allows hanging through the DIG connection receives on the backstage different inference engine. Thus, Racer, FaCT, Pellet like this may also be used in Jean “specialized” a inducing equipment.

In order to unify self-definition rule and complete OWL DL function of Pellet, presently the inducing equipment lamination method, similar to the self-definition rule, unifying OWL/RDFS, are used to carry on the code test to the front ontology example. First uses Pellet for the source data establishment inducing equipment model, establishes one again from the definition rule inducing equipment, this inducing equipment takes the Pellet model the first floor data use. In this kind of situation the first floor Pellet inducing equipment may understand from the define name attribute hasSibling transitivity. The first floor inference function can carry on independent inference computation and the results are submitted to the upper formation inducing equipment, however the upper formation inducing equipment may take the inference result of Pellet as source data to carry on self-defining relation calculation once more.

The inducing equipment stack-up used exterior inducing equipment Pellet to have the inference completeness at the same time and the decidability OWL DL support, on the other hand fully has also displayed from definition rule nimble widespread superiority. This kind of stack-up is uses one kind of function which start the source software can realize to be strong at present, result complete semantics inducing equipment solution..

d) Frorid-2 and F-OWL tools:

F-OWL is one infers the engine based on Flora-2 to the OWL ontology. It uses object-oriented knowledge library language Flora-2 to transform F-logic, HiLog and the migration logic unified language is XSB, and infers engine's application procedure using XSB to carry out in the development platform. The F-OWL essential characteristic including carries on the inference with the OWL ontology model ability, defines the axiom rule support knowledge parallel shot ability with Flora-2, as well as is the Java application integrates opening application program interface (API).

F-OWL is described and expanded with Flora-2 in XSB. F-OWL provides command line connection; a simple graphical user interface and Java API meet the different need. Comes with F-OWL on the ontology to infer is usually composed of the following four steps:

(1) Load attachment application procedure related rule to engine in;

(2) Increases new FDF and the OWL statement (for example ontology or assertion) to the engine. In the OWL sentence's triples (ontology, predicate, and object) by the transformation are 2 frame styles: Ontology (predicate, object) @ model.

(3) Inquires the engine. RDF and the OWL rule is the recursion uses for to have all legitimate triples. If an inquiry does not have the variable, when a question's explanation was discovered that returns to true the reply. If the question includes the variable, then the variable with substitutes from the explanation and the returns value;

(4) If needs, the ontology and triples may delete. Otherwise, XSB retrieves triples which the table the form preservation calculates causes afterward inquiry to be more rapid.

F-OWL uses one to infer the OWL ontology based on the frame system. F-OWL supports knowledge library's parallel shot, extracts the hideaway knowledge through the resolution, and supports through the introduction rule carries on further complex infers. F-OWL is a full function inference engine, it easy to use and can with many kinds of query languages and the regular language integration.

Under the open Web environment, usually the tentative data is incomplete, and all facts are by no means known. How will this paper study this fact to affect one to infer engine's movement? In semantic Web, an inference engine possibly does not need to produce the evidence, but should be able to inspect the evidence. We will use F-OWL to analyze in semantic Web the information and the evidence.

In an independent system the nonuniformity is the danger, but should control it in some kind of degree. But must control in semantic Web the nonuniformity is difficult. Therefore it needs to have in semantic Web processes nonuniform and the contradictory information specific mechanism. This mechanism has two steps: Surveys inconsistent and analyzes not consistently.

The inconsistent survey is based on the inference engine's nonuniform statement. The possible value and the ontology element verifies with forcefully can with the relational restraint conflict, causes the nonuniformity. For example, owl:equivalentClass, imposing a restraint on resources whose ontology is equivalence class, is just similar to owl:disjointWith imposing a restraint to the ontology on resources whose ontology is not a equivalence class. Triples (a owl: equivalentClass b) and (a owl:disjointWith b) has not caused direct inconsistency, until using survey rule that (A owl:equivalentClass B) & (A owl:disjointWith B) is inconsistent.

When surveys the nonuniformity, Namespaces can be helpful to the track nonuniform origin, marks each Web page and does not have two righteousness in semantic Web to process it. Then infer the engine to connect the trust system to appraise the name space the credibility. [14] and [15] in maintain the semantic Web credible system aspect have obtained many remarkable achievements and the thought. Once has the credible appraisal result, the agent may take three different measures: (a) through infers that which the engine accepts recommends; when (b) both has been incredible rejects; (c) lets the user choose.

B. From OWL DL to F-logic

The description relation between description logic and F-logic may be appled to the precise structure of OWL DL. As mentioned above, RDF and OWL are based on composition of a triples: ontology, a predicate and an object sentence. Therefore, an OWL ontology is constructed by a list sentences. The assigned transformation has used F-logic basic atom and molecule. With the F-logic rules, OWL the DL structure can carry on more transformations.

From XML to the F-logic rule transformation is and the fact transformation close related. Especially a restraint proposition's transformation to OWL the DL proposition to the F-logic transformation is comparable. Moreover, the switching process relies on the ontology, applies on this ontology the restraint and the rule. First, a restraint (rule) the proposition ontology, the attribute, the object may through use URIs to quote in the ontology the resources, either expression variable, either founds the new resources. Next, has some attributes to define as the function attribute, and this definition is a ontology part. This need to use a function attribute to transform a regular proposition is F-logic. Through constructs the predicate to use the F-logic equality internally, or assigns the predicate, the predicate to transform by the direct way F-logic.

C. Combination inference of Pellet and Flora-2

The architecture shows as Figure 3. The point core is the Jena frame. It contains one to the ontology, the fact and the rule knowledge library (for example concept and attribute). To the OWL-S inference, a Pellet inference's example connects the point.

Carries on the inference main idea with DL+Flora-2 is divides the service is (i) OWL the concept (TBox) inference, the (ii) rule application, as well as (iii) inherits. The inference fact may in the OWL part (ABox, for example transitivity, reflexivity, symmetry), or permits through the rule to the more complex knowledge inferential reasoning. Knowledge library KB=(L, P, D), thus by the expression is

- OWL ontology L,
- Regular finite set P (by F-logic or RuleML style XML mark), as well as
- inherits the atom (default) finite set D.

The computational process is as follows, it starts on the Jena point by the OWL-S ontology:

The first step: Calculates one to assign the fact storehouse the OWL-S model. Because it's open world characteristic, the OWL/DL inference only contains the limit denial. Random will derive by the consideration will be afterward "possible". The OWL-S inference first needs to guarantee completely safely.

The second step: Derivation rule application. All related fact and the rule output Flora-2 together, bottom-up appraises by its application; the fact which produces as the result returns to related to based on the Jena core. In the rule's situation, it in later the step the fact which obtains through the OWL-S inference inferential reasoning is also completely safe.

Circulation: The above step is the iteration, could not infer again until the new fact.

Inherits the step: When the OWL-S inference and the rule application cannot further infer the fact, then the default inherits occurs only.

Iteration: So long as the new fact is tacitly approved inherits infers, the above internal iteration will restart. This corresponding default inherits the F-logic semantics - using the default inherits only when the rule applies is unable to infer again takes the post of the He Xin fact the fixed point, restarts iterative - this explanation is "may infer", and the default logic semantics is compatible.

In summary, this paper verification tool supports the inference type includes:

- Assigns a kind of type the subclass or the ultra class;
- Assigns the attribute type the sub-attribute or the ultra attribute;
- Assigns the example is the class (all or direct ultra kinds) He Zhong type;
- Two assign the example or two types whether same or different;
- Example's assigns the attribute value is anything;
- Examples of a kind of type;

Assign the example (current only to be able to gain direct attribute) all attributes, as well as to obtain the more inquiry abilities to combine in together inquiry.

D. Example

The following takes requirements meta model of an airline seat reservation system as an example, to validate the model instance.

```
- <rdf:RDF xml:base="http://www.di.unipi.it/~corfini/owls/processmodels/(geography)_GeographicLocations.owl.xml">
  - <owl:Ontology rdf:about="">
      <owl:versionInfo>$Id: GeographicLocations.owl,v 1.00 2007/06/07$</owl:versionInfo>
    - <rdfs:comment>
        OWL-S UNIPI and UMA repository: GeographicLocations OWL-S Process Model - it returns, if possible, state, city and street given a geographic
        coordinate (i.e., latitude and longitude) - it returns latitude, longitude and height given a city and a state - it computed the distance between two given
        coordinates
      </rdfs:comment>
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/Service.owl"/>
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/Process.owl"/>
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/Profile.owl"/>
      <owl:imports rdf:resource="http://www.di.unipi.it/~corfini/owls/ontologies/geography.owl.xml"/>
      <owl:imports rdf:resource="http://www.di.unipi.it/~corfini/owls/ontologies/country.owl.xml"/>
      <owl:imports rdf:resource="http://www.di.unipi.it/~corfini/owls/ontologies/address.owl.xml"/>
      <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl"/>
    </owl:Ontology>
  - <!--
      ##########################################################################
    -->
  - <!--
      Definition of top level Process (GeographicLocation) as a composite process
    -->
  - <process:CompositeProcess rdf:ID="GeographicLocation">
    - <rdfs:label>
        This is the choice between the three main operations provided by the service
      </rdfs:label>
      <process:invocable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</process:invocable>
    - <process:hasInput>
      - <process:Input rdf:ID="GeographicLocation_latitudeIn">
```

Fig 4. The converted OWL-S document using the above mapping rules from UML to OWL-S

After transforming UML of software requirements to OWL-S documents, we use OWLSVerifyTool to convert OWL-S document further to PNML document describing with Petri nets.

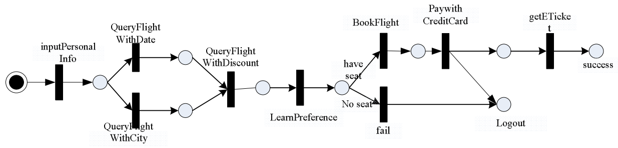

Fig 7. The airline booking service in Petri nets

Finally we use OWLSVerifyTool to carry on dynamic model verification and static model verification, and obtain the corresponding results.

V. CONCLUSION

This paper has realized the integration static model and the dynamic model inspection is a body's automated verification tool prototype; In the static model aspect, gives the method which DL description logic reasoning (Pellet) and the F-logic rule (Flora-2) unifies, has used two kind of system forward reasoning fully and latter to the inference merit, combined based on DL inference engine Pellet and based on F-logic inference engine Flora-2, displays its respective superiority to carry on the inference and the verification. An OWL DL subset may transform is F-logic, and also provides the reverse support.

Should automate the verification tool prototype to be able service to provide the effective verification support to OWL-S the description Web, have the important theory and the practical significance, and has the widespread application prospect.

ACKNOWLEDGMENT

This work was supported in part by a grant from Shenzhen Science and Technology Program (JC201006020820A and JC201006020791A), Team of Scientific and Technological Innovation of Shenzhen Institute of Information Technology (CXTD2-002), Natural Science Foundation of Guangdong Province, China (S2011040000672), Guangdong Provincial Education and Science Project of the 11th "five-year plan"(2010tjk411), Research Fund of Shenzhen Institute of Information Technology (YB201008, JY2010003).

REFERENCES

[1] He Keqing, Liang Peng, Li Bing , et al. Meta-modeling of Requirement for Networked Software, An Open Hierarchical & Cooperative Unified Requirement Framework URF [J] .DCDIS2B , 2007 , 14 (S6) : 293-298..

[2] Charles Haley, Robin Laney, Jonathan Moffett, Bashar Nuseibeh, "Security Requirements Engineering: A Framework for Representation and Analysis," IEEE Transactions on Software Engineering, vol. 34, no. 1, pp. 133-153, Jan. 2008, doi:10.1109/TSE.2007.70754.

[3] Ivan Jureta, John Mylopoulos, Stephane Faulkner, "Revisiting the Core Ontology and Problem in Requirements Engineering," re, pp.71-80, 2008 16th IEEE International Requirements Engineering Conference, 2008

[4] Wilco Engelsman, Henk Jonkers, Henry M. Franken and Maria-Eugenia Iacob, Architecture-Driven Requirements Engineering, Advances in Enterprise Engineering II, Lecture Notes in Business Information Processing, 2009, Volume 28, 134-15

[5] [3B] Matthias Klusch, Benedikt Fries, and Katia Sycara; OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 7, Issue 2, April 2009, pp. 121-133.

[6] Cesare Pautasso, Olaf Zimmermann Frank Leymann, Restful web services vs. "big"' web services: making the right architectural decision, WWW '08 Proceeding of the 17th international conference on World Wide Web, pp. 805-814

[7] John Erickson, Keng Siau, Web Services, Service-Oriented Computing, and Service-Oriented Architecture: Separating Hype from Reality, Journal of Database Management (JDM), Volume 19, Issue 3. 2008. pp. 13-17.

[8] Shoichi Morimoto, A Survey of Formal Verification for Business Process Modeling, Computational Science – ICCS 2008, Lecture Notes in Computer Science, 2008, Volume 5102/2008,Springer, pp.514-522.

[9] Qi Guo, Tianshi Chen, Haihua Shen, Yunji Chen, Weiwu Hu, On-the-Fly Reduction of Stimuli for Functional Verification, ats, 2010 19th IEEE Asian Test Symposium, 2010, pp.448-454.

[10] Feng He, Jiajin Le. Hierarchical Petri-nets model for the design of e-learning system. Second International Conference, Edutainment 2007, Hong Kong, China, June 2007. Lecture Notes in Computer Science, Vol. 4469, pp.283-292.

[11] Juan C. Vidal, Manuel Lama, Alberto Bugarín, OPENET: Ontology-based engine for high-level Petri nets, Expert Systems with Applications, Volume 37, Issue 9, September 2010, pp. 6493-6509

[12] Sylvain Hallé, Graham Hughes, Tevfik Bultan and Muath Alkhalaf, Generating Interface Grammars from WSDL for Automated Verification of Web Services, Service-Oriented Computing, Lecture Notes in Computer Science, 2009, Volume 5900/2009, pp.516-530.

[13] Moldt, D., Ortmann, J.: DaGen: A Tool for Automatic Translation from DAML-S to High-Level Petri Nets. In Wermelinger, M., Margaria, T., eds.: FASE 2004, LNCS 2984, Springer (2004) 209–213D. Kornack and P. Rakic, “Cell Proliferation without Neurogenesis in Adult Primate Neocortex,” Science, vol. 294, Dec. 2001, pp. 2127-2130

[14] Matthew Staats and Mats P. E. Heimdahl: Partial Translation Verification for Untrusted Code-Generators, Formal Methods and Software Engineering, Lecture Notes in Computer Science, 2008, Volume 5256/2008, pp.226-237.

[15] Emilia Oikarinen, Tomi Janhunen: A Translation-based Approach to the Verification of Modular Equivalence, Journal of Logic and Computation, Volume19, Issue4, May 6, 2008, pp. 591-613.

Tao He received the PhD degrees in computer science from the Shanghai University of Shanghai, in 2009. From 2009 to 2011, he was an associate professor in the Institute of Shenzhen Institute of Information Technology, Shenzhen, China. His research interests are in software engineering, software testing, software formalization as well as Web engineering.

Liping Li received the PhD degrees in computer science from the Shanghai University of Shanghai, in 2011. From 2007 to 2011, she was an associate professor in the Shanghai Second Polytechnic University, Shanghai, China. Her research interests are in software engineering, software testing, software formalization as well as Web engineering.