# Ontology-based Optimization Modelling Tool for distributed and ad-hoc business problems

PIAO GUANGYUAN

The Graduate School

Yonsei University

Industrial and Information Engineering

# Ontology-based Optimization Modelling Tool for distributed and ad-hoc business problems

A Master's Thesis

Submitted to the Department of Industrial and Information Engineering

and the Graduate School of Yonsei University

in partial fulfillment of the

requirements for the degree of

Master of Industrial and Information Engineering

PIAO GUANGYUAN

December 2011

This certifies that the master's thesis
of PIAO GUANGYUAN is approved.

_____
Thesis Supervisor: Wooju Kim

_____
Chang Ouk Kim

_____
June Seok Hong

The Graduate School
Yonsei University
December 2011

# Table of Contents

# List of Figures

# ABSTRACT

## Ontology-based Optimization Modelling Tool using SWCL

Semantic Web Constraint Language (SWCL) was proposed in 2006 with the vision of enhancing the quality of humans' decision-making by machines in Semantic Web environments, complementing the missing but important part which acknowledges constraints on concepts represented by ontology. Even though, the XML syntax for SWCL itself is too verbose and has a downside that needs to be generated by hand with deep knowledge of OWL. In this respect, users have to be aware of the syntax and semantics of OWL in deep in order to generate a problem description in SWCL by hand without any error. Moreover, due to the complexity of SWCL, it would be tedious work to generate all constraints with SWCL in many cases. These reasons causes decrease of the efficiency and popularity for SWCL.

In order to cope with these issues, we developed ontology-based optimization modelling tool using SWCL which can help users to manipulate the problem model in SWCL straightforward and make the optimization model intuitive as well. Through our research and development, we supplement the method to represent the class description for each variable in SWCL with Manchester syntax which is developed in response to a demand from a wide range of users, who do not have the background knowledge for Description Logic (DL). We developed and implemented the variable detection algorithm and the related constraints extraction algorithm to construct the optimization model selecting the constraints in SWCL that are distributed and shared in the Semantic Web environments. In addition, the translation algorithm and tool that could be able to

convert the problem model in SWCL to problem description in OPL modelling language.

The developed tool was applied to a production scheduling problem for virtual enterprise to validate the usability of our methodology. This research shows that various problems with different objective and constraints could be manipulated to construct the optimization model by the ontology-based optimization modelling tool.

***Keywords:*** *Semantic Web, Semantic Web Constraint Language, Optimization Modelling*

# 1. Introduction

## 1.1 Background of the Research

Since Semantic Web has been introduced, the Semantic Web society has developed a set of basic foundational frameworks to share knowledge among machines. As Tim Berners-Lee depicted in the Semantic Web Stack [1] and its ontology layers, based on RDF, RDF Schema, and OWL, has been almost stabilized. The technologies from the bottom of the stack up to OWL [2] as well as the new version OWL2.0 [3] are currently standardized and accepted to build Semantic Web applications. It is still not clear, however, how the top of the stack is going to be implemented. All layers of the stack need to be implemented to achieve full visions of the Semantic Web. A logic and rule layer was supposed to be a way to extend it. One of the choices for this logic layer was a Semantic Web Rule Language (SWRL), which was submitted as a recommendation to W3C in 2004. SWRL was designed to extend representational power of OWL by combining Horn-like rules with OWL. In addition, the activities of W3C members' submission like Semantic Web Service Language (SWSL) [4] and Web Rule Language (WRL) [5] show that the expressive power of Semantic Web is not confined only by data, but also richer knowledge such as rules to be expressive, sharable and reusable across the Web environment. In this respect, constraint as a neglected piece, which would be able to improve the representational power significantly, is in need to be extended or involved in Semantic Web environment. The first formal approach proposed to deal with constraints in OWL it was known as Constraint Interchange Format/Semantic Web Rule Language (CIF/SWRL) [6]. Its theoretical foundation, based on First Order Logic (FOL), includes constraints that

conform to logic only, which are not appropriate for handling arithmetic constraints (other important components of constraints) point out that processing arithmetic constraints is essential if we want to deal with decision making in classical optimization problems in the Semantic Web environment because these optimization problems are usually expressed with arithmetic constraints. Furthermore, they described the necessity of exchanging arithmetic constraints over the Web while handling optimization problems with goals. In order to complement existing methods by extending them over their limitations and handling mathematical optimization problems in the Semantic Web environment, the Semantic Web Constraint Language has been proposed by [7] which are compatible with OWL. In this research, they also showed how the SWCL could represent problem of internet shopper in the real world and solve the problem with an intelligent agent that understands information represented by semantic technologies like OWL, SWRL and SWCL. This approach showed the potential of SWCL and the ability to enhance the expressive power of Semantic Web impressively. In addition, sharing interchanging constraints information among each other in the virtual enterprise with SWCL was proposed by [8]. Although in these researches they were using SWCL in a meaningful way, because of the SWCL itself it was needed to be generated by hand, it requires extremely deep understanding about OWL in order to make sure the SWCL correct without any error, in some cases, it would also be a tedious work to generating a lot of constraints exist. Furthermore, generate SWCL manually makes the reference of OWL resources like OWL classes and properties very difficult.

### 1.2 Objective and Scopes of the Research

The primary goal of the work is to propose an effective way to manipulate SWCL with an ontology-based optimization modelling tool as Protégé plug-in to

facilitate the SWCL generation process straightforward and provide CRUD functions additionally with referencing OWL classes and properties in Protégé more intuitively. On top of that, we are going to develop conversion algorithm that translating the optimization model described in SWCL to OPL modelling language in order to get the optimized solution from IBM Ilog Cplex Optimization Studio. With constraints and objective generated by users and the converted optimization model, we can get the optimized solution from IBM Ilog Cplex Optimization Studio with ease which can help decision making or planning for users in specific circumstances like virtual enterprise. Virtual enterprise is a temporal alliance of companies that shares resources to respond to business opportunities better. Obtaining an optimal production schedule for a supply chain of virtual enterprise is usually the most important factor for its success. Assume that there are three vendors within Semantic Web in the virtual enterprise at first. The virtual enterprise has one memory vendor; CPU vendor and assembly vendor in order to produce PC for their users (Suppose that a PC produced by the assembly company is assembled with only the one CPU and two memories). They are sharing the same ontology schema and the objective of this virtual enterprise is to minimize the total cost of the three vendors. With optimization modelling tool we propose in this paper, the prime company (Assembly Company) could be able to define constraints sharing within the virtual enterprise and the objective in SWCL too. The optimization model defined in SWCL will be described in OPL modelling language after conversion algorithm described in section 4. Then, the virtual enterprise could get the optimized objective with optimized schedule separately after solving optimization model defined in OPL modelling language. In addition, when there's additional memory vendor is going to join the virtual enterprise which is sharing the same ontology schema (Figure 1), there's nothing to change but adding the corresponding instances into the ontology. Then with

optimization modelling tool, the virtual enterprise can get the minimized total cost with additional memory vendor 2 with ease. Furthermore, it can also help the virtual enterprise to decide co working with additional memory vendor 2 is a good choice or not. The scenario will be described in detail in section 5.

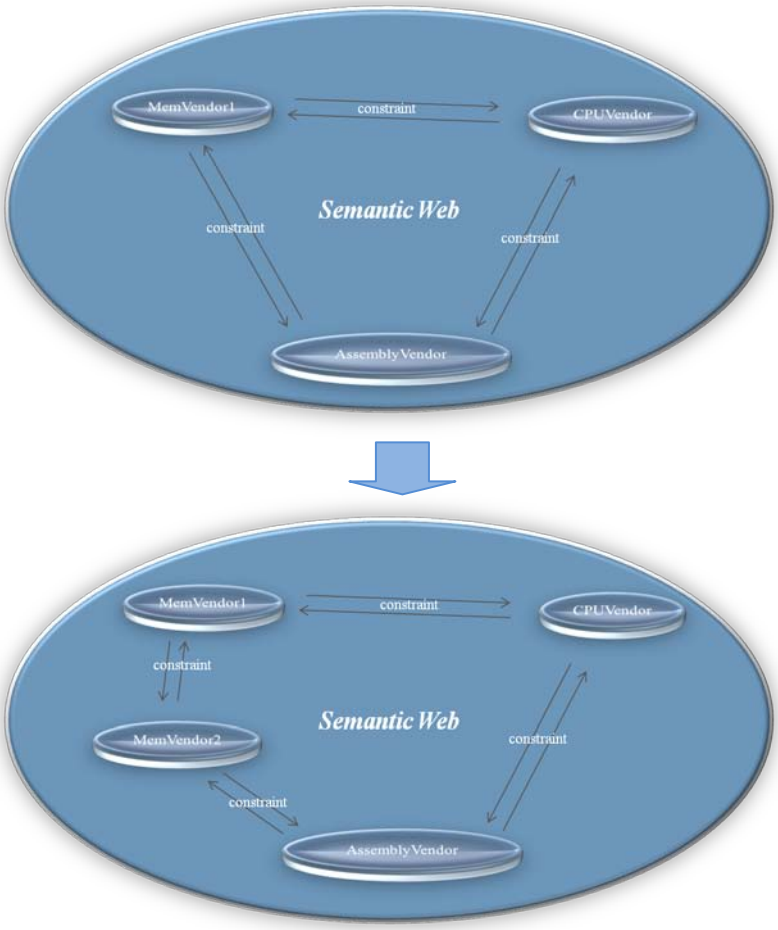


**Fig 1. Virtual Enterprise Scenario**

The account of this work is presented in the following order. Section 2 describes the related works. Section 3 describes the complement of SWCL with Manchester OWL Syntax. In Section 4, we discuss the ontology-based development of optimization modelling tool using SWCL in Protégé 4.x series which can efficiently help the SWCL generation and modification easier and several algorithms adapted in the tool. Section 5 summarizes the presented work and lists of contributions.

# 2. Related works

## 2.1 Semantic Web

The Semantic Web [9], coined by Tim Berners-Lee, is a "man-made woven web of data" that facilitates machines to understand the semantics, or meaning, of information on the World Wide Web. In addition to the classic “Web of documents” W3C is helping to build a technology stack to support a “Web of data,” the sort of data you find in databases. The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term “Semantic Web” refers to W3C’s vision of the Web of linked data. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. In order to archive the goal of the Semantic Web, components for supporting the Semantic Web are proposed by Berners-Lee, 2006. Figure 1 shows the up-to-the-minute Semantic Web reference architecture. There are some technologies such as URI, XML, RDF, OWL, and so forth for sharing meaning on the Semantic Web.

## 2.2 OWL2.0

OWL 2 Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language that knowledge expressed in OWL can be reasoned with by computer programs either to verify the consistency of that knowledge or to make implicit knowledge explicit. OWL documents, known as ontologies, can be published in the World Wide Web and may refer to or be

referred from other OWL ontologies. OWL is part of the W3C's Semantic Web technology stack and OWL 2.0 which adds syntactic sugar to make some common patterns easier to write, new constructs that increase expressivity, extended datatypes and capabilities, simple metamodelling capabilities and other innovations and minor features is the extension of OWL and had been recommended by W3C from 2009.

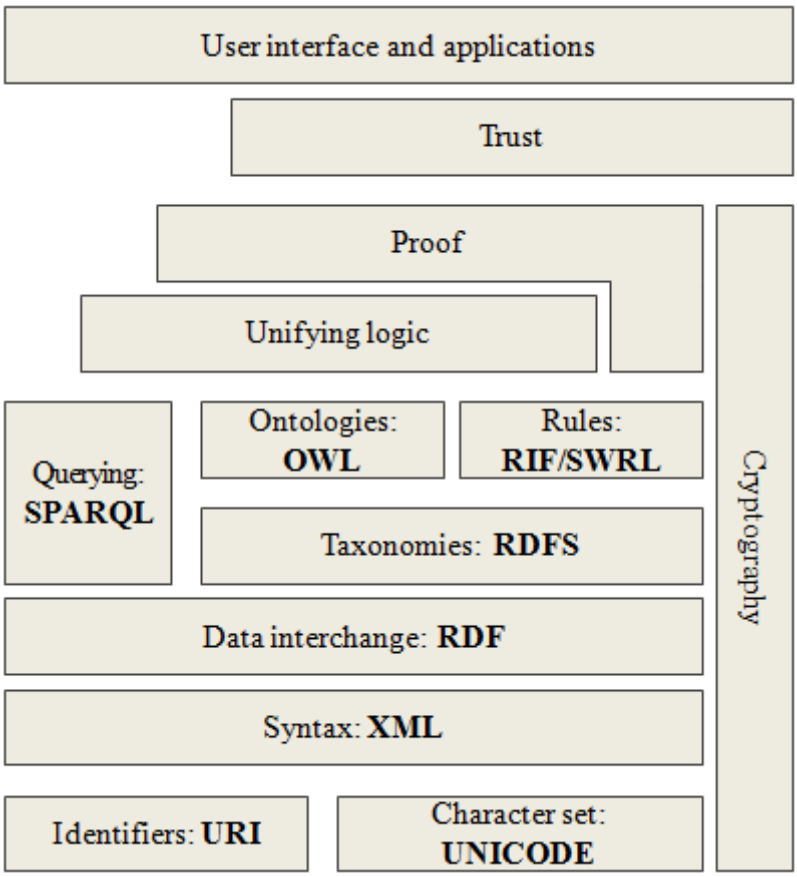


**Fig 2. Architecture of the Semantic Web**

### 2.3 OWL API

The OWLAPI [10] is a Java API and reference implementation for creating, manipulating and serialising OWL ontologies. The latest version of the API is focused towards OWL 2. In addition, when we face the problem which version of Protégé is the

most popular open-source platform that provides a suite of tools to construct domain models and knowledge-based applications with ontologies, in view of the Protégé 3.x series and Protégé 4.x series made by different faculty (the former one is made by Stanford and the latter one made by Manchester), and Protégé 4.x uses the open source, Java-based OWL API that is proving popular with many developers around the world, this makes writing or migrating to and from other systems more straightforward, and a larger developer community which means more assistance and a more robust codebase. In addition, the OWL API supports much of the upcoming OWL 2.0 recommendation. So we decided to use the OWL API and Protégé 4.x series to develop SWCL Editor. However, for developers that require access to the RDF model at the level of a triple store, the Protégé-OWL API from Stanford is an option. Moreover, Protégé 4.0 OWL support is built on top of the OWL API, which provides all of the model manipulation and querying functionality. On top of this, Protégé supports further functionality for developer's convenience - hierarchies, renderer management, search etc.

### 2.4 SWRL and SWRLTab

Semantic Web Rule Language (SWRL) [ 11] is a language based on a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. The proposal extends the set of OWL axioms to include Horn-like rules. It thus enables Horn-like rules to be combined with an OWL knowledge base. A high-level abstract syntax is provided that extends the OWL abstract syntax described in the OWL Semantics and Abstract Syntax document [12]. An extension of the OWL model-theoretic semantics is also given to provide a formal meaning for OWL ontologies including rules written in this abstract syntax. The SWRLTab [13] is a development

environment for working with SWRL rules in Protégé-OWL. It supports the editing and execution of SWRL rules. It provides a set of libraries that can be used in rules, including libraries to interoperate with XML documents, and spreadsheets, and libraries with mathematical, string, RDFS, and temporal operators. A SWRL-based OWL query language called SQWRL is also provided.

### 2.5 SWCL

As mentioned in section 1, SWCL [7] had been proposed, which combines constraints with an OWL knowledge base at an abstract level. They also proposed the abstract syntax of SWCL to facilitate access to constraints and the evaluation of their expressions in the Semantic Web. The syntax for SWCL extends the abstract syntax of OWL described in the OWL Semantics and Abstract Syntax document [14] with additional axioms for constraints used in SWRL [10]. An abstract syntax for SWCL is specified using a version of the EBNF notation used for XML [14] Terminals are quoted, whereas non-terminals are bold but not quoted. The abstract syntax of SWCL is shown in figure 3.

*Axiom* ::= *OptModel*

*OptModel* ::='*OptModel*('*objective subjectTo*')'

*objective* ::='*Objective*('*optimizationInstruction objectiveTerm*')'

*subjectTo* ::='('{*constraint*}')'

*optimizationInstruction* ::='*Minimize*'|'*Maximize*'

*objectiveTerms* ::='*objectiveTerm*('*termBlock*{*termBlock*}')'

*constraint* ::='*Constraint*('[*URIreference*]{*qualifier*} *LHS operator RHS*

*qualifier* ::='*Qualifier*('*variableID* | *variable*')'

*variable* ::='*Variable*('*variableID description*')'

*variableID* ::='*VariableID*('*URIreference*')'

*LHS* ::='*LHS*('*termBlock*{*termBlock*}')'

*operator* ::='*equal*'|'*notEqual*'|'*lessThan*'|'*lessThanOrEqual*'|'*greaterThan*'|'*greaterThanOrEqual*'

*RHS* ::='*RHS*('*termBlock*{*termBlock*}')'

*termBlock* ::='*TermBlock*('*sign* [*aggregateOperator*] {*parameter*} *factor* {*factor*}')'

*sign* ::='+'|'–'

*aggregateOperator* ::='*Sigma*'|'*Production*'

*factor* ::='*Factor*('*individualID datavaluePropertyID*')'|'*Factor*('*variableID datavaluedPropertyID*')'

*parameter* ::='*Parameter*('*variableID* | *variable*')'

**Fig 3. Abstract syntax of SWCL**

Alternatives are either separated by vertical bars ( | ) or provided in different productions. Components that occur at most once are enclosed in square brackets ([…]); those that occur any number of times (including zero) are in braces ({…}). Here, white space is ignored in the productions.

# 3. SWCL Complement

In order to explain our approach more explicitly, we use the simple population example [7]. Suppose that we have sample knowledge about two classes, namely, “Country” and “Province,” and their relationship, as shown in figure 4. Figure 4 also shows that each class has two properties, namely, “partOf” and “populationValue.” The “partOf” property denotes that “Province” is a part of “Country,” whereas “populationValue” indicates the number of inhabitants in the region. The population of

each country should equal the sum of the populations of the provinces belonging to that country. This natural knowledge can be easily represented by a mathematical constraint as Formula (1).

$$\sum_{x \in y.partOf} x.populationValue = y.populationValue, \ for\ all\ y \in (Country)^{c} \quad (1)$$

Where C denotes a class interpretation function; and a.b stands for the value of property b of an instance a. As already mentioned, the SWCL is made to combine constraints with an OWL knowledge base in the Semantic Web. In our optimization modelling tool, there are some requirements that need users to define the description of variable in the process of generating SWCL. In this case, if we use SWCL describe the description of variable directly; the description of variable x will be described like figure 5, which is also verbose at all. In order to make the description of variable more straightforward for users, we apply Manchester Syntax within user interface development which can significantly make the description of variable less verbose and easier to read and write for users.

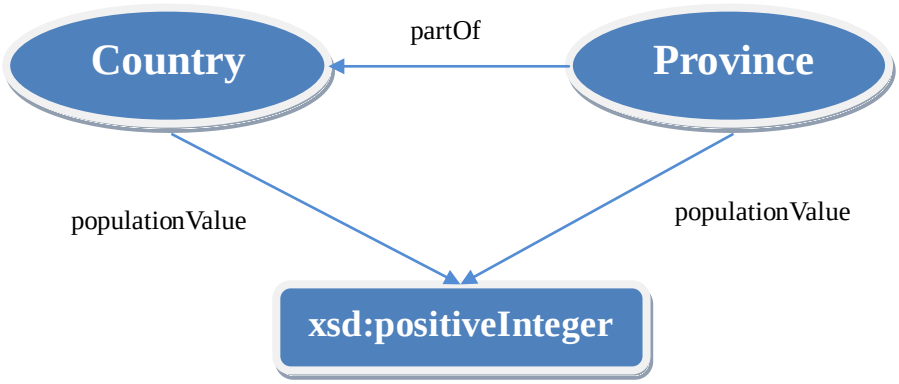


**Fig 4. Country and province classes and the relationship**

```
<swcl:Variable rdf:id="x">
        <rdfs:subClassOf>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#partOf"/>
                    <owl:hasValue rdf:resource="#y"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</swcl:Variable>
```

**Fig 5. Class expression for variable x in SWCL**

### 3.1 Manchester Syntax

The Manchester Syntax was designed with primary considerations to produce a syntax that was concise, did not use DL symbols, and was quick and easy to read and write. These considerations were based on the experience of interaction with users of Protégé-OWL where the syntax would be primarily used to edit class expressions.

Lessons learnt from the GALEN project [15] were also taken into consideration. For example, minimising the number of brackets required to write class expressions, and choosing keywords to promote readability, were taken into account.

| OWL Constructor | DL Syntax | Manchester Syntax |
|---|---|---|
| **intersectionOf** | C ⊓ D | C **AND** D |
| **unionOf** | C ⊔ D | C **OR** D |
| **complementOf** | ¬C | **NOT** C |
| **oneOf** | {a} ⊔ {b} | {a b…} |
| **someValueFrom** | ∃ R C | R **SOME** C |
| **allValuesFrom** | ∀ R C | R **ONLY** C |
| **minCardinality** | ≥ N R | R **MIN** 3 |
| **maxCardinality** | ≤ N R | R **MAX** 3 |
| **Cardinality** | = N R | R **EXACTLY** 3 |
| **hasValue** | ∃ R {a} | R **VALUE** a |

**Fig 6. The Manchester OWL Syntax OWL 1.0 Class Constructors**

Manchester Syntax makes the class expression more natural to read, uses the

natural language keywords, and also makes it easy to paste the plain text representation of the expression into e-mails etc. without incurring the formatting problems that can arise due to the different fonts required to represent the mathematical symbols that are used in the DL syntax. The class descriptions of Manchester Syntax are shown in figure 6. If so, the class expression of variables represented by SWCL Syntax, can be all replaced by Manchester Syntax. The corresponding class expression in Manchester Syntax for variable x could be described as “partOf **VALUE** y”, which is extremely simple and easier to read and write for the users.

# 4. Ontology-based Optimization Modelling Tool using SWCL

SWCL is based on the OWL, which means it references OWL classes and properties within constructing constraints. Take this in account, and complementation with Manchester Syntax, we developed ontology-based modelling tool which can help users manipulate SWCL easier and more intuitively. Furthermore, we developed several algorithms to make the translation process automatically from SWCL to OPL modelling language in order to get the optimized solution from IBM Ilog Cplex Optimization Studio.

## 4.1 Architecture of Ontology-based Optimization Modelling Tool

The architecture of ontology-based optimization modelling tool is displayed in figure 7. There are three main components in it which are constraint editor, optimization model constructor, optimization modelling language translator separately. The first one is developed for helping users manipulate SWCL with user interface which is applying Manchester Syntax in SWCL domain description and referencing resources in protégé intuitively. With the constraints and objective defined from constraint editor,

optimization model constructor can automatically identify objective and constraints related to it. During the process, decision variable identification is done with checking the factor value exists or not. Optimization model identification can help us to find out the model we defined is LP, CSP etc. After the optimization model defined in SWCL, we need to translate it to optimization model described in OPL modelling language in order to get the solution of the problem we defined. Hence, we developed an algorithm to make the translation available and can get the optimized resolution of the problem from IBM Ilog Cplex Optimization Studio.

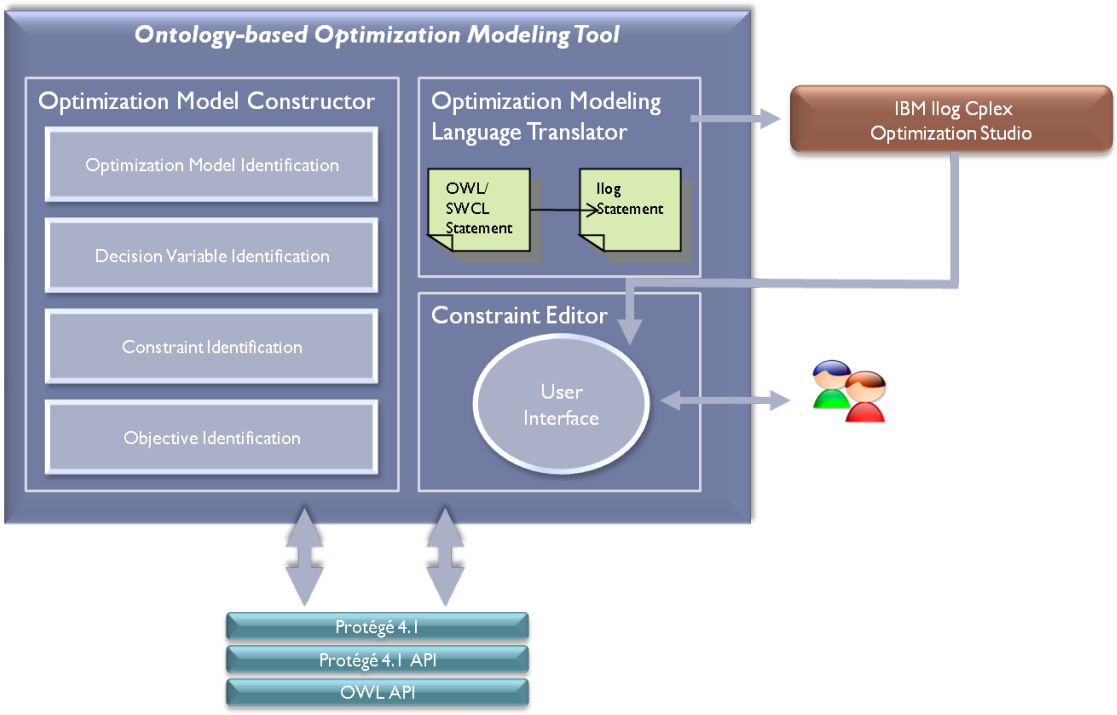


**Fig 7. The architecture of ontology-based optimization modelling tool**

## 4.2 Constraint Editor

Constraint Editor has been developed the similar user interface as SWRLTab does in order to provide familiar user experience for users. Users could simply add, modify and delete the constraints with the Constraint Editor in Protégé. The adding co nstraint frame is displayed in figure 8. The adding constraint frame is developed based

on the SWCL structure. Variable panel is used for declaring the variables which would be used for expressing constraints. As one might expect, some variables could probably be used among different constraints. Thus, the variables declared in one frame would be able to be used in other constraints too. After finishing the constraint, we just need to click on the "OK'' button at the bottom of the frame to check the constraint, it will be displayed at the abstract syntax panel at the bottom of the frame. The added constraint then will be displayed in SWCL View in the Protégé after clicking the "Submit" button in the adding constraint frame. As mentioned in section 3, we apply Manchester Syntax in domain description of variable, which is less verbose, easy to read and write. Furthermore, the Constraint Editor can reference resources like OWL classes and properties in the Protégé intuitively with providing options fro m user interface.

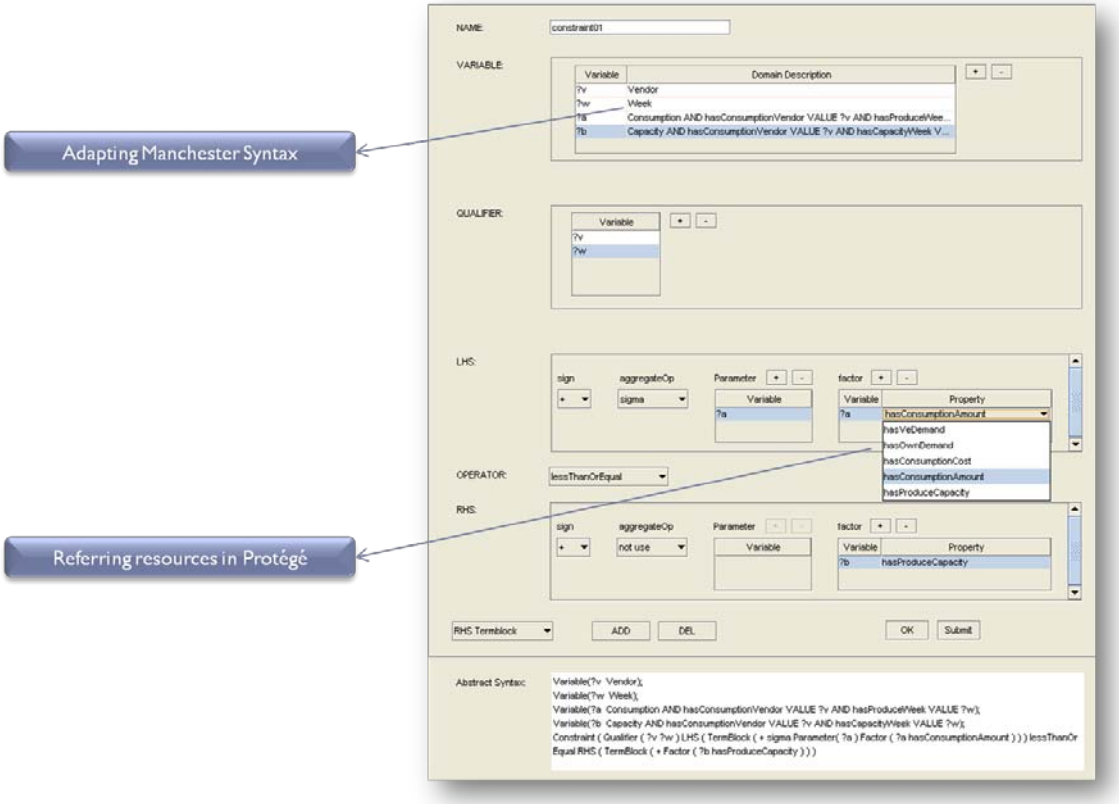


**Fig 8. Frame for adding constraint in constraint editor**

The Objective View displayed in Figure 9 is developed to help users define their objective simply. Users can use the Objective View to select optimization instruction and edit objective term blocks to define the objective.

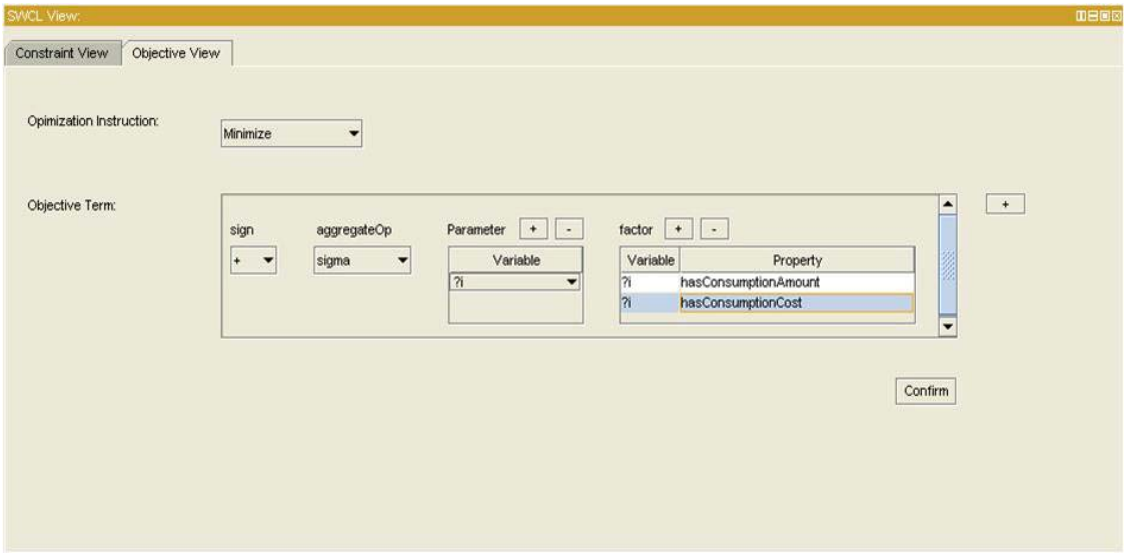


**Fig 9. Frame for define the objective of optimization model**

### 4.3 Optimization Model Constructor

Optimization Model Constructor has four components which are objective identification, constraints identification, decision variable identification and optimization model identification. With these four steps, we can get the objective that user defined, the constraints related to the objective and decision variables as well. In other words, the optimization model described in SWCL is achieved.

1) Objective Identification.

This step identifies the user's objective for specific problem after finishing defines the objective of the problem. As shown in figure 9, the variable used in this step should be declared in SWCL generation process. Suppose variable i is stand for the

"Consumption" class, as one might expect, the user would like to minimize the total consumption cost.

2) Constraint Identification.

Once the objective is given from previous step, we need to extract constraints relevant to the objective automatically. In order to do that, firstly, we need to find out all factors from objective that individuals in the variable of every factor have no value in its property. Then, "decision variable factors set" is initialized with these found factors.

```
initialize decision variable factors set empty
initialize relevant factors set empty
initialize identified    constraint set empty

for all Objective TermBlocks
    for all Factors
        if Factor is "decision variable factor"
            set {Factor's class expression, property} as a pair
            add the pair to decision variable factors set
        endif
    endfor
endfor
repeat
    for all Constraints
        for all Factors in Constraint
            if Factor is exist in decision variables set
            add Constraint to identified constraint set
                if relevant factors in the Constraint is "decision variable"
                    set {Factor's class expression, property} as a pair
                    add the pair to relevant factors set
                endif
            endif
        endfor
    endfor
    set    decision variable factors set as relevant factors set
    set relevant factors set empty
until    decision variable factors set is empty
```

**Fig 10. A general procedure for constraints identification**

Second, for all constraints, we should identify the constraint if there are some

factors in it have the relation to one of decision variable factors". Having relation in two factors means that the property is the same, in addition, the class expression (domain description) of variable should have intersection simultaneously. Aside from this, for all identified constraints, we have to check out relevant factors. In some cases, the relevant factors in identified constraints would have some additional "decision variable factors". Based on this fact, find all constraints including these kind of relevant factors until there are no relevant constraints anymore. The process is described as pseudo code in figure 10.

3) Decision Variable Identification

As a significant step involved in constraint identification, identify the decision variable is relatively straightforward. Take the factor "a.hasConsumptionAmount" as an example, for all individuals in class expression (domain description) of variable a, we can check the property value from the ontology. If we can obtain the value as a form of the literal from the ontology, in that case, we can identify this is constant. On the other hand, if the value is not available from ontology, it will be identified as decision variable.

4) Optimization Identification

In the optimization identification step, an agent classifies what kind of programming model the given problem is (e.g., linear programming problem, integer programming problem, etc.). For the reason of we only deal with the linear programming problem in this research, this part will be our future work and won't be discussed in this paper.

### 4.4 Optimization Modelling Language Translator

In this section, we will explain how to translate the optimization model described in OWL/SWCL statement from previous steps. There is no doubt about that we need a specific solver to solve the problem defined by users. There are many solvers around the world like Prolog, IBM Ilog CPlex Optimization Studio etc. In this research, we simply choose IBM Ilog Cplex Optimization Studio as solver without any specific reason. To obtain a solution with IBM Ilog Cplex Optimization Studio, the problem identified with OWL/ SWCL should be translated into OPL modelling language. This translation task is quite straightforward with the constructs in the relevant constraint set, the decision variable factor set. For each relevant constraint, the optimization modelling tool writes the corresponding OPL modelling language and then adds required decision variables and data declarations. There are two main parts of this translation process, the objective translation and the constraints translation respectively.

1) Objective Translation

Suppose we have the objective defined by user as following, as one might expect, for all individuals in class expression of variable a, there will be the same corresponding numbers of constraints occurred in OPL modelling language. In the mean time, every value of factor for each individual should obtain from ontology. As mentioned above, if there's literal value available, take the value and reflect it in the OPL modelling language. Otherwise, when the factor is decision variable factor, we should declare the variable first in OPL modelling language and reflect as "name of property append name of individual". For example, one of individual of variable a named "v1Consumption11" and the property "hasConsumption" has no value in

ontology. Then, we should declare the variable named "hasConsumptionAmountv1Consumption11" in OPL declaration of variable and reflect it in expression of constraint.

$$\textbf{minimize} \quad \sum_{a} a.hasConsumptionAmount \times a.hasConsumptionCost$$

```
initialize VariableDeclaration String to ""
initialize ObjectiveTo String to ""

for all Objective TermBlocks
initialize Objective TermBlock String to ""
   for all individuals in Parameter
        for all factors in individual
        initialize Factor String to "“
           if factor's variable has the value in factor's property
              append the value to Factor String
           else
             declare the variable with property's name appending individual's name
             append the variable to Factor String
          endif
       append Factor String to Objective TermBlock
     endfor
     endfor
     append Objective TermBlock String to ObjectiveTo String
endfor
```

**Fig 11. Objective translation procedure from SWCL to OPL**

The overall objective translation procedure from SWCL to OPL modelling language is briefly described in figure 11.

2) Constraints Translation

The constraints translation process is much the same as objective translation. In contrast to the objective translation, there are qualifiers in constraints that we have to take attention. Take a constraint described as following, for example, for both qualifiers v and w, we can obtain one corresponding constraint in OPL modelling language.

$for\ all\ \ v \in Vendor, w \in Week$

$a \in Consumption\ AND\ hasConsumptionVendor\ VALUE\ v\ AND\ hasProduceWeek\ VALUE\ w$

$b \in Capacity\ AND\ hasConsumptionVendor\ VALUE\ v\ AND\ hasCapacityWeek\ VALUE\ w$

$\sum_{a} a.hasConsumptionAmount \leq b.hasProduceCapability$

```
for all qualifiers
initialize LHS String to ""
    for all LHS TermBlocks
    initialize LHS TermBlock String to ""
        if there is no Parameter
            for all factors
            initialize Factor String to ""
                if factor's variable has the value in factor's property
                    append the value to Factor String
                else
                    declare the variable with property's name appending individual's
                    name
                    append the variable to Factor String
                endif
            endfor
            append Factor String to LHS TermBlock String
        else
            for all individuals in Parameter
                for all factors in individual
                initialize Factor String to ""
                    if factor's variable has the value in factor's property
                        append the value to Factor String
                    else
                        declare the variable with property's name appending individual's
                        name
                        append the variable to Factor String
                      endif
                endfor
            endfor
        endif
        append LHS TermBlock String to LHS String
    endfor
endfor
```

**Fig 12. Part of constraint translation procedure from SWCL to OPL**

That means, if there are four individuals in class expression of variable v and

four individuals in class expression of variable w respectively, we are going to obtain 4*4 =16 corresponding constraints in OPL modelling language. At each loop, we can easily get the individuals in class expression of variable a. If so, the translation of the LHS part of this constraint is quite the same as objective translation process. The LHS part of constraint translation procedure is described in figure 12.

# 5. Applying Ontology-based Optimization Modelling Tool to Virtual Enterprise

To show the usability of SWCL and our ontology-based optimization modelling tool using SWCL, we apply it to solve the production scheduling problem of a virtual enterprise.

## 5.1 Virtual Enterprise problem

Virtual enterprise is a temporal alliance of companies that shares resources to respond to business opportunities better. Obtaining an optimal production schedule for a supply chain of virtual enterprise is usually the most important factor for its success. In turn, sharing related information about the production of participating companies is a critical success factor needed to achieve an optimal solution in the supply chain domain because it is infeasible to obtain the best production schedule from overall perspectives across the entire virtual enterprise without transparency and timely access to all the production situations of participating companies. A real-world example scenario of a virtual enterprise is shown in figure 13. The virtual enterprise is composed of one prime company and two subcontract companies. Suppose that a computer produced by the prime company is assembled with only the one CPU and two memories. Assume that there is a specific market's need for a computer type, "vcomputer," which is equipped

with a special type of CPU, “vcpu,” and a specific type of memory, “vmem.” In figure 13, a virtual enterprise is formed to meet this requirement; the figure also shows the relationship between the assembly company and the vendor companies that supply the required CPU and memory.

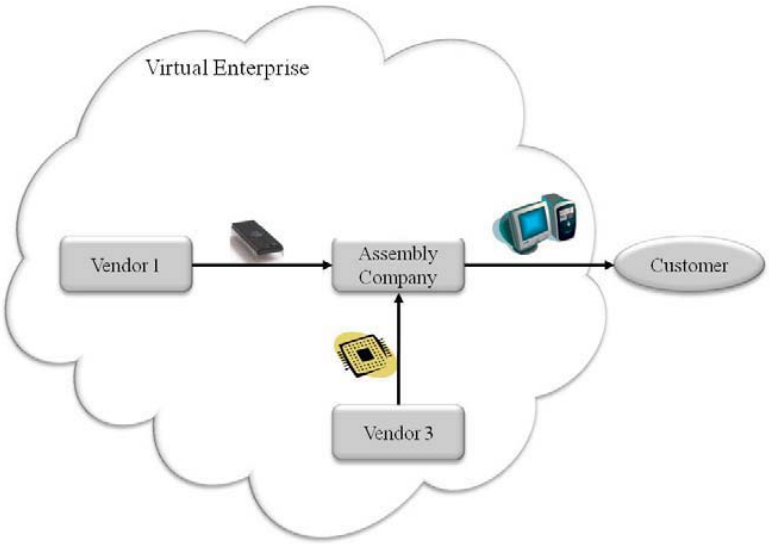


**Fig 13. Supply Chain for Virtual Enterprise**

### 5.2 Constraint sharing among vendors

The virtual enterprise is temporary formed to respond to the new market demand for the product “vcomputer”, in this case, there are several constraints related to “vcomputer” are required to be shared among these participating companies. Before introducing these constraints, we assume that these companies participating in virtual enterprise schedule their production tasks on a weekly basis. The fundamental constraints to be shared are those of intrinsic constraints of each company, which are independent of the virtual enterprise and predetermined by their own. There are two types of such constraints, namely capacity limitation and weekly demand requirement. The two constraints can be formulated by a set of mathematical constraints as follows.

$$\sum_{j=i}^{4} x_{ijk}^{p} \le Capacity_{ik}^{p} \quad for\ i = 1,2,3,4\ k = 1,2,3,4\ and\ \ for\ all\ \ p = cpu, memory, computer \quad 2)$$

$$OwnDemand_{jk}^{p} + VeDemand_{jk}^{p} \le \sum_{i=1}^{j} x_{ijk}^{p} \quad for\ \ j = 1,2,3,4\ k = 1,2,3,4\ and\ \ for\ all\ \ p \quad 3)$$

Where $x_{ijk}^{p}$ stands for the product p produced at i-th week and consumed st j-th week by k-th company (we denote vendors 1, 3 in Fig. 15 as k = 1, 3, respectively, and assembly company as k = 4); $Capacity_{ik}^{p}$ is the maximum weekly production capacity of k-th company for p-th product; $OwnDemand_{jk}^{p}$ is the own demand at i-th week of k-th company for product p; and $VeDemand_{jk}^{p}$ is the demand from virtual enterprise at i-th week of k-th company for product p.

On top of that, there are two additional constraints for the case. 1). the demand of CPU vendors should greater or equal to amount of assembly company. 2). the demand of Memory vendors should greater or equal to two times of producing amount of assembly company. These two constraints can be formulated as following.

$$VeDemand_{i3}^{cpu} \ge \sum_{j=i}^{4} x_{ij4}^{computor} \quad for\ i = 1,2,3,4 \quad 4)$$

$$\sum_{k=1}^{2} VeDemand_{ik}^{memory} \ge 2 \times \sum_{j=i}^{4} x_{ij4}^{computor} \quad for\ i = 1,2,3,4 \quad 5)$$

With these constraints, the virtual enterprise describes its objective to be achieved using our ontology-based optimization modelling tool. The virtual enterprise wants to minimize its total cost incurred in both production and inventory. This can be represented as follows:

$$TC = TC_{cpu} + TC_{mem} + TC_{computer} \quad 6)$$

$$TC_{cpu} = C_3^{cpu} \times \sum_{i=1}^{4} \sum_{j=1}^{i} x_{ij3}^{cpu} + V_3^{cpu} \times \sum_{j=2}^{4} \sum_{i=1}^{j-1} (j-i) \cdot x_{ij3}^{cpu} \quad 7)$$

$$TC_{mem} = \sum_{k=1}^{2}[C_k^{mem} \times \sum_{i=1}^{4}\sum_{j=1}^{i} x_{ijk}^{mem} + V_k^{mem} \times \sum_{j=2}^{4}\sum_{i=1}^{j-1}(j-i)\cdot x_{ijk}^{mem}] \quad 8)$$

$$TC_{computer} = C_4^{computer} \times \sum_{i=1}^{4}\sum_{j=1}^{i} x_{ij4}^{computer} + V_4^{computer} \times \sum_{j=2}^{4}\sum_{i=1}^{j-1}(j-i)\cdot x_{ij4}^{computer} \quad 9)$$

Whereas $C_k^p$ denotes that unit production cost of p-th production in k-th company and $V_k^p$ stands for the unit inventory cost per week of p-th product in k-th company.

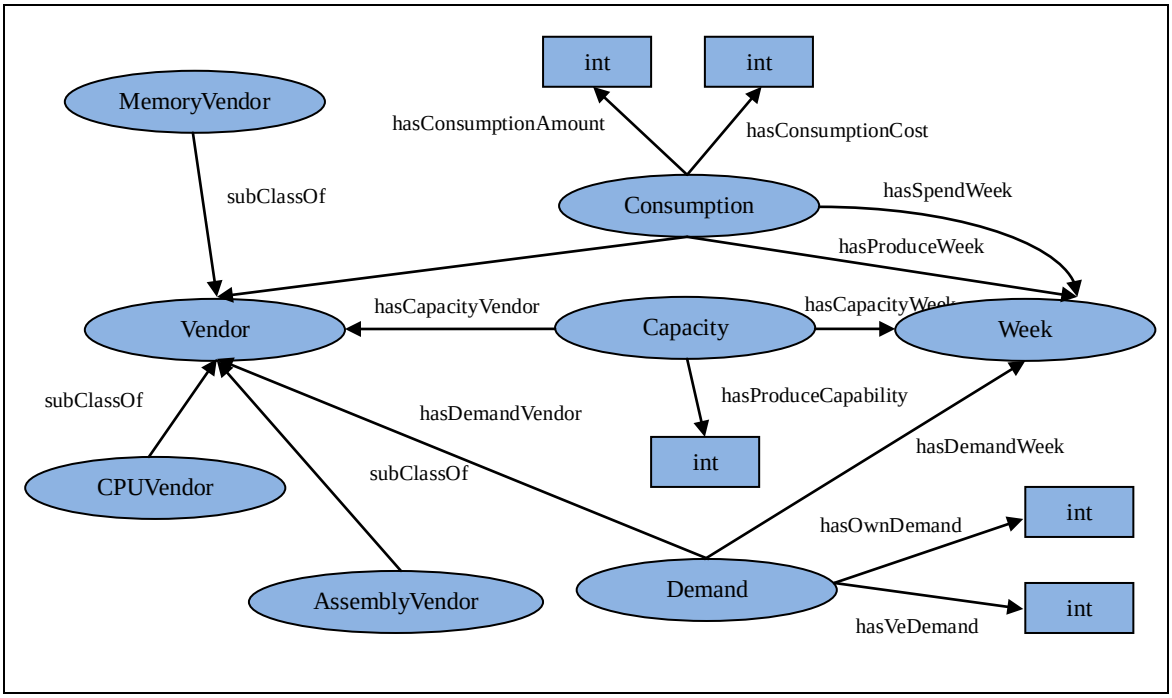


**Fig 14. Schema level of Ontology for Virtual Enterprise**

## 5.3 Virtual Enterprise Ontology Modelling

In order to generate SWCL which is based on the OWL, we need ontology model that represent the information for these companies. The schema level of ontology is modelled like figure 14; Vendor class has three subclasses which are Memory Vendor, CPU Vendor and Assembly Vendor. Consumption class is a class for describing the information about which vendor produce how many substances and spend in which week. Capacity and Demand class is standing for the information about a Vendor's

capacity and demand in a specific week.

### 5.4 Optimization Modelling with Optimization Modelling Tool

Based on the ontology model designed above, we can get the optimization model with optimization modelling tool we mentioned in section 4. Within this process, the constraints should be able to be represented after applying Manchester Syntax in our optimization modelling tool. If so, those constraints in formula 2), 3), 4), 5) will be represented as following based on the virtual enterprise ontology.

I. **Constraint-1:**

Producing amount should less or equal than the produce capability in that week.

$$for\ all\ v \in Vendor, w \in Week$$

$$a \in Consumption\ AND\ hasConsumptionVendor\ VALUE\ v\ AND\ hasProduceWeek\ VALUE\ w$$

$$b \in Capacity\ AND\ hasConsumptionVendor\ VALUE\ v\ AND\ hasCapacityWeek\ VALUE\ w$$

$$\sum_{a} a.hasConsumptionAmount \le b.hasProduceCapability$$

II. **Constraint-2:**

Demand in the week should less or equal than the sum of spending amount.

$$for\ all\ v \in Vendor, w \in Week$$

$$c \in Demand\ AND\ hasDemandVendor\ VALUE\ v\ AND\ hasDemandWeek\ VALUE\ w$$

$$d \in Consumption\ AND\ hasConsumptionVendor\ VALUE\ v\ AND\ hasSpendWeek\ VALUE\ w$$

$$c.hasOwnDemand + c.hasVeDemand \le \sum_{d} d.hasConsumptionAmount$$

III. **Constraint-3:**

The demand of CPU vendors should greater or equal to producing amount of assembly

company.

$for\ all\ \ w \in Week,\ z \in AssemblyVendor$

$e \in CPUVendor$

$k \in Demand\ \ AND\ hasDemandVendor\ VALUE\ e\ AND\ hasDemandWeek\ w$

$f\ \in Consumption\ AND\ hasConsumptionVendor\ VALUE\ \ z\ \ AND\ hasProduceWeek\ VALUE\ \ w$

$$\sum_{k} k.hasVeDemand \geq \sum_{f} f.hasConsumptionAmount$$

IV. **Constraint-4:**

The sum of demands of memory vendors should greater or equal to producing amount of assembly company.

$for\ all\ \ w \in Week,\ z \in AssemblyVendor$

$g \in MemoryVendor$

$q \in Demand\ \ AND\ hasDemandVendor\ VALUE\ \ g\ \ AND\ hasDemandWeek\ \ w$

$$\sum_{q} q.hasVeDemand \geq \sum_{f} 2 \times f.hasConsumptionAmount$$

### 5.5 Translated Optimization Model in OPL modelling language

With conversion algorithm mentioned in section 4.4, as a consequence, we could be able to get the translated optimization model in OPL modelling language as displayed in figure 15.

```
dvar int+ hasConsumptionAmountv1Consumption11;
dvar int+ hasConsumptionAmountv1Consumption12;
```

```
dvar int+ hasConsumptionAmountv1Consumption13;
dvar int+ hasConsumptionAmountv1Consumption14;
; … …
minimize
50*hasConsumptionAmountv1Consumption11+55*hasConsumptionAmountv1Consu
mption12+…+30*hasConsumptionAmountv4Consumption44;

subject to {

hasConsumptionAmountv1Consumption11+hasConsumptionAmountv1Consumption
12+hasConsumptionAmountv1Consumption13+hasConsumptionAmountv1Consump
tion14<=90;
hasConsumptionAmountv1Consumption23+hasConsumptionAmountv1Consumption
24+hasConsumptionAmountv1Consumption22<=80;
hasConsumptionAmountv1Consumption33+hasConsumptionAmountv1Consumption
34<=90;
hasConsumptionAmountv1Consumption44<=90;
hasConsumptionAmountv3Consumption11+hasConsumptionAmountv3Consumption
14+hasConsumptionAmountv3Consumption12+hasConsumptionAmountv3Consump
tion13<=70;
hasConsumptionAmountv3Consumption22+hasConsumptionAmountv3Consumption
23+hasConsumptionAmountv3Consumption24<=80;
hasConsumptionAmountv3Consumption33+hasConsumptionAmountv3Consumption
34<=75;
… …
}
```

**Fig 15. Translated optimization model in OPL modelling language**

Based on the translated OPL modelling language above, now, we could be able to use IBM Ilog Cplex Optimization Studio to solve the problem user generated. As a consequence, we can get the optimized solution and objective with ease from solver. In this case, we get the optimized objective 34000 which is the smallest total consumption cost in this virtual enterprise with specific consumptions for every vendor displayed in figure 16.

```
// solution (optimal) with objective 34000

hasConsumptionAmountv1Consumption11 = 90;
hasConsumptionAmountv1Consumption12 = 0;
hasConsumptionAmountv1Consumption13 = 0;
```

```
hasConsumptionAmountv1Consumption14 = 0;
hasConsumptionAmountv1Consumption22 = 70;
hasConsumptionAmountv1Consumption23 = 10;
hasConsumptionAmountv1Consumption24 = 0;
hasConsumptionAmountv1Consumption33 = 90;
hasConsumptionAmountv1Consumption34 = 0;
hasConsumptionAmountv1Consumption44 = 80;
hasConsumptionAmountv3Consumption11 = 35;
hasConsumptionAmountv3Consumption12 = 0;
hasConsumptionAmountv3Consumption13 = 0;
hasConsumptionAmountv3Consumption14 = 0;
hasConsumptionAmountv3Consumption22 = 45;
hasConsumptionAmountv3Consumption23 = 0;
hasConsumptionAmountv3Consumption24 = 0;
hasConsumptionAmountv3Consumption33 = 45;
hasConsumptionAmountv3Consumption34 = 0;
hasConsumptionAmountv3Consumption44 = 50;
hasConsumptionAmountv4Consumption11 = 30;
…
…
hasVeDemandv2DemandWeek3 = 90;
hasVeDemandv2DemandWeek4 = 80;
hasVeDemandv3DemandWeek1 = 30;
hasVeDemandv3DemandWeek2 = 40;
hasVeDemandv3DemandWeek3 = 45;
hasVeDemandv3DemandWeek4 = 40;
```

**Fig 16. Optimized solution from IBM Ilog Cplex Optimization Studio**

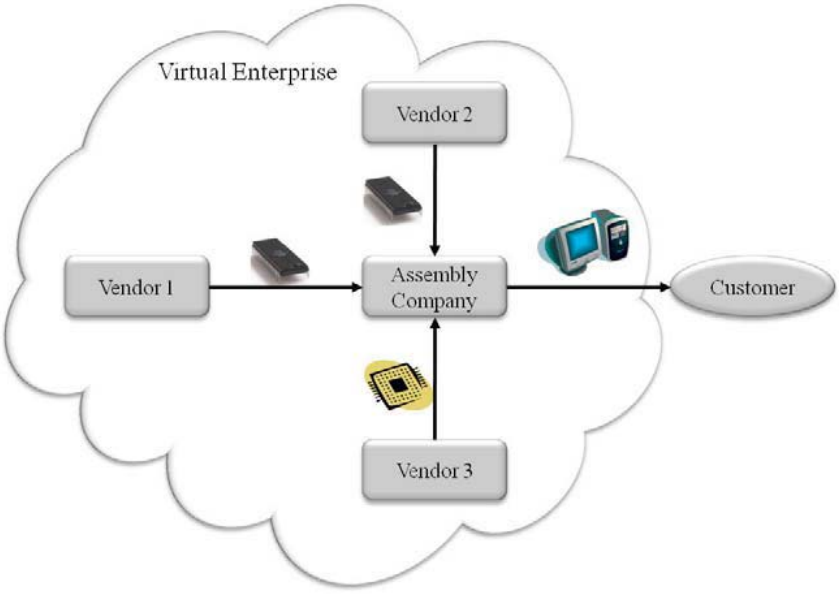


**Fig 17. Supply Chain for Virtual Enterprise with additional memory vendor**

Another scenario is that there's additional memory vendor – vender 2 want to

join the virtual enterprise (figure 17). In this case, the optimization modelling tool could be able to help us to get the optimized solution with additional memory vendor as well as help make decision about it's the better choice to work with additional memory vendor or not. After assuming that vendor 2 is joining the virtual enterprise, the objective is minimize the total cost of 4 vendors including vendor 2 with constraints as the same as previous scenario.

With the same process did in previous case, we could be able to get the translated optimization model described in OPL modelling language with memory vendor 2. Also, after solving the problem defined in OPL modelling language, we could get the optimized solution from IBM Ilog Cplex Optimization Studio with ease. In contrast to the case with one memory vendor, we would get optimized solution 31600, which means, it's better to work with additional memory vendor and we can save 2400 working with additional memory vendor. This also shows that ontology-based optimization modelling tool can deal with distributed and ad-hoc business problems with various objectives.

# 6. Conclusion

In this paper, we developed optimization modelling tool using SWCL which could be able to help users define constraints in a meaningful way by providing user-friendly GUI. In addition, we adapt Manchester Syntax which is designed to describe OWL description and also less verbal, easy to read and write. Furthermore, as the tool developed in Protégé as a plug-in, the resources like OWL classes and properties loading in Protégé could be referenced intuitively. For these reasons, users could be able to define their constraints and objective with our GUI quickly and straightforwardly.

On top of that, we developed the algorithm which can automatically extract the

constraints related to the objective defined by users to construct the optimization model. Additionally, in order to facilitate the conversion of SWCL and OPL modelling language, we developed the conversion algorithm that can transform from SWCL to the OPL modelling language automatically. If so, we can simply get the answer from the IBM Ilog Cplex Optimization Studio with optimized solution of the problem user defined. The virtual enterprise example we took in this paper showed that how efficiently ontology-based optimization modelling tool can help us to provide optimized solution within distributed and ad-hoc business problem. We believe this will firmly help users define and share constraints around the Semantic Web which can help users get optimized objective with supporting decision making for users.

### 6.1 Summary of Contributions

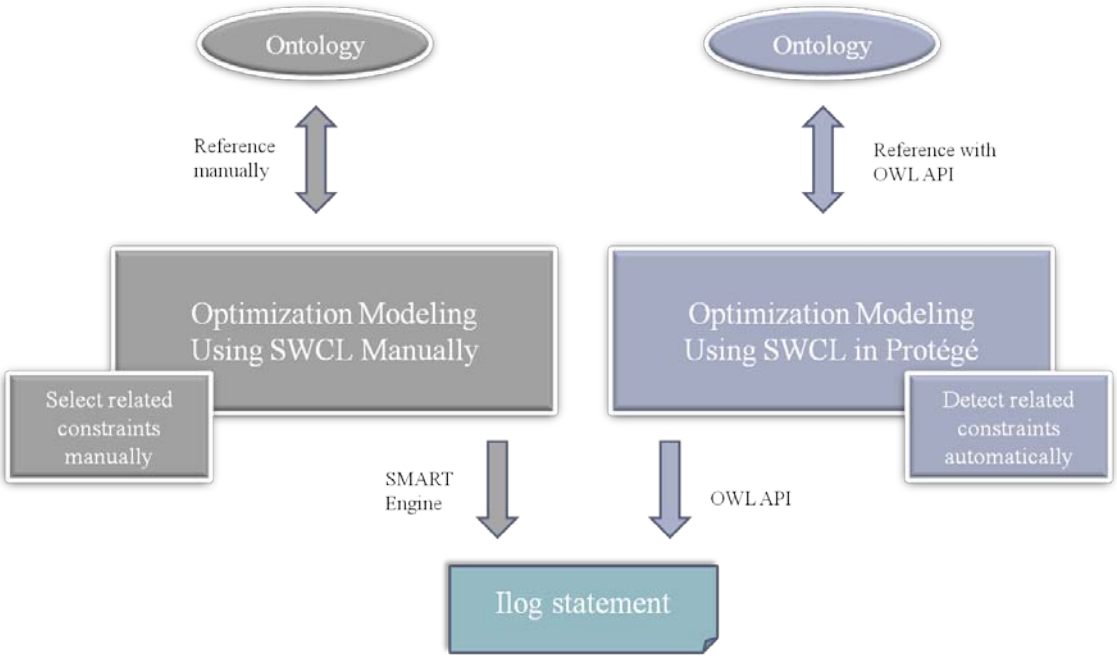


**Fig 18. Comparison with the previous research**

Compare to the previous researches, the significant contributions of the research can be summarized as follows.

1. Optimization modelling process using SWCL can be done with Optimization Modelling Tool in Protégé instead of manual generation.
2. All constraints related to the objective that user defined can be detected automatically, not by user selection.
3. In previous works, user has to check the OWL classes or properties in ontology manually. With Optimization Modelling Tool, user could be able to reference resources in ontology more intuitively.
4. OWL API is used within the interaction of ontology instead of SMART Engine. Because of this, it’s easier to maintenance and update with OWL language.

# References


[1]. http://en.wikipedia.org/wiki/Semantic_Web_Stack

[2]. http://www.w3.org/TR/owl-features/

[3]. http://www.w3.org/TR/owl2-profiles/

[4]. http://www.w3.org/Submission/SWSF-SWSL/

[5]. http://www.w3.org/Submission/WRL/

[6]. McKenzie, C., Gray, P., and Preece, A. Extending SWRL to Express Fully Quantified Constraints. Lecture Notes on Computer Science, 2004.

[7]. Youn, S. H. Supporting System for Shopping Decision Making based on SWCL, Graduate Thesis, 2006.

[8]. Jeong, K. B. Application of Semantic Web Constraint Language SWCL for Virtual Enterprise. Graduate Thesis, 2007.

[9]. http://www.w3.org/standards/semanticweb/

[10]. http://owlapi.sourceforge.net/index.html

[11]. http://www.w3.org/Submission/SWRL/#1

[12]. Pater, S., Patrick, F., Hayes, Lan, H. OWL Web Ontology Language semantics and abstract syntax. 2004.

[13]. http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab

[14]. Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E. (eds). Extensible Markup Language (XML) 1.0. 2000.

[15]. Rector, A., Solomon, W. D., Nowlan, W. A., Rush, T.W. A terminology server for medical language and medical information systems. Methods of Information In Medicine, 1994

# 국문 요약

## Ad-hoc 비즈니스 문제 해결을 위한
## 온톨로지 기반 최적화 모델링에 대한 연구

시맨틱 웹 제약언어 (SWCL) 는 시맨틱 웹 환경에서의 제약적인 표현을 가능하게 함으로써 시맨틱 웹 환경에서의 의사결정에 도움을 주고자 하는 비전아래 2006년에 제안되었다. 하지만 SWCL Syntax 자체가 굉장히 복잡하고 매뉴얼하게 작성되어야 하기 때문에 OWL기반으로 하는 SWCL을 오류가 없이 정확하게 작성하기 위해서는 OWL에 대한 깊은 이해도가 필요하였고 또한 많은 제약식들을 표현하고자 할 때 엄청난 수작업이 필요하게 됨으로써 SWCL의 실제적인 사용과 응용이 어려웠다.

이런 이슈들을 해결하고자 본 논문에서는 온톨로지 기반으로 하는 최적화 모델링 툴을 개발함으로써 SWCL에 대한 작성, 수정, 삭제와 같은 조작을 사용자 친화적으로 할 수 있게 하였을 뿐만 아니라 GUI를 설계하고 개발하는 과정에서 변수의 domain description부분에 사용자 친화적인 Manchester Syntax를 적용함으로써 사용자가 domain description부분에 대한

가독성과 편집성을 높였다. 또한 사용자가 시맨틱 웹 제약언어로 정의한 제약 식들과 목적들을 가지고 최적화 모델링을 함으로써 특정 문제에 대한 최적화된 솔루션을 제공할 수 있도록 시맨틱 웹 제약언어를 IBM Ilog Cplex Optimization Studio의 OPL코드로 전환해주는 알고리즘을 개발하였다.

본 연구를 통하여 SWCL도 SWRL처럼 사용자 친화적인 자체적인 편집기와 외부 솔버를 연계시킴으로써 사용의 편이성과 가능성을 높였고 시맨틱 웹 환경에서의 SWCL을 이용한 제약식 표현을 통하여 분산된 애드호크(ad-hoc) 비즈니스 문제 해결과 더불어 사용자의 의사결정에 더욱 긍정적인 도움이 될 것이다.

# 감사의 글

우선 논문을 성공적으로 끝마칠 수 있도록 지도해주신 김우주 교수님, 홍준석 교수님, 김창욱 교수님에게 감사 드립니다. 바쁘신 와중에도 시간을 내어서 지도해주시고 의견을 내어주시고 마지막 심사까지 최선을 다해주셔서 너무 고맙습니다. 교수님들 모두가 항상 건강하시고 하시고자 하는 일들이 뜻대로 되기를 기원합니다.

그리고 2년이라는 길고도 짧은 유학생활에 항상 알게 모르게 챙겨주었던 연구실 전체 팀원들에게 너무 감사 합니다. 이미 졸업하고 직장생활중인 태영이 형, 려영 누나 항상 행복이 가득한 직장생활을 이어가시기 바라고 연구실의 큰 형들이신 명진형과 성환형, 앞으로의 연구실 연구를 이끌어갈 경민형, 동규형, 광일형, 이연이, 웨이제 누나, 샐리 누나 앞으로 좋은 연구성과가 있기를 바라고 행복이 가득한 하루하루가 되기를 바랍니다. 그리고 함께 졸업하는 지현 누나, 곧 졸업을 하게 될 수연이, 영준이, 종서형 마무리 잘 하시고 좋은 직장에서 자신이 원하는 삶과 미래를 이루어가시기 바랍니다.

마지막으로 중국에 있을 때나 한국에 와서도 항상 든든한 버팀목이 되어주시고 정신적 의지가 되어주시는 ITDT모든 분들께 감사 드립니다. 얼굴은 익숙하지 않아도 마음으로 항상 응원해주시고 기도해주시는 ITDT가 유학생활에서 큰 힘이 되었고 앞으로의 한국 생활에도 큰 힘이 될 것입니다. 앞으로는 누군가에게 힘이 되어줄 ITDT의 한 구성원으로서의 역할을 하는 YUSTAR가 되겠습니다.