# Semantic Object Framework – A language independent approach to integrate semantic web and object oriented programming

Abstract

Object-oriented programming is a mainstream of present software development, whereas a number of problems still existing in the integration of the semantic web and the object-oriented programming must be solved, and these problems being solved include: manipulating RDF API abstraction by using object methods, automatically converting data into RDF format, supporting an architecture for various programming languages, using statements to describe the semantics of classes and attributes, supporting query of inheritance and heterogeneous data between classes and attributes, and verifying the consistency of data and semantics. The main reason to solve these problems mentioned above is that the description capability of the relationship between object-oriented classes has more limitation than that used by RDF, and thus the failure of direct mapping exists between the object-oriented class and the semantic web class. Although object-oriented RDF APIs that are recently implemented for specific programming languages try to alleviate the workload of development, these APIs generally only solve certain part of problems mentioned above, with the result of lack of the robustness of solutions. This paper proposes Semantic Object Framework (SOF), and it incorporates the benefits of both object-oriented design and semantic web, and uses embedding comments to describe the semantic relationship between classes and attributes in order to solve all problems mentioned above.

一祥翻譯社 樣本 請勿模製
Elegant Translation Service Sample
Do not copy

## 1. Introduction

Semantic web extends the description ability of WWW, and its basic concept is to define the semantic relationship of data and allow that the data resulted from additional process can be shared or processed by automotive tools (machines). In various applications, this can be advantageously utilized for the effective search of needed data, process automation, integration and reuse. RDF is a data model representation of semantic web and, meanwhile, a basic RDF document is a statement that is consisted of three elements, wherein these three elements are Subject, Predicate, and Object. However, APIs that are presently used to manipulate RDF are on the basis of Triple-oriented APIs to manipulate statements that are consisted of three elements.

When developing semantic web applications, RD data models must be handled in programming languages. However, the methods of present mainstream software development utilize object-oriented programming, but it is not suitable for handling semantic web data. Model View Controller (MVC) that is most widely used in modern object-oriented programming is used as a function-dividing architecture for designing classes, and the most common implementation is that data objects pertaining to Model classes are converted into record format for the storage of relational database. However, RDF utilizes Triple-oriented statements as a basis of data format, but there is a significant diversity between this and the format used by MVC Model classes, and a lack of semantic relationship description between classes exists in object-oriented classes, therefore it is not easy to automatically convert data objects to RDF for the provision of semantic query. If a large amount of existing data is needed to be converted to Triple-oriented format for storage, this could be a real challenge from the view of performance and cost.

一祥翻譯社 樣本 請勿複製 Elegant Translation Service Sample Do not copy

This paper proposes that SOF closely associates object-oriented design with semantic web so that program developers don’t need to learn different concepts, and you may easily publish data object as RDF format by simply applying object-oriented design methods, and heterogeneous data query can be also made in accordance with the semantic relationship between classes and attributes.

An overview of this paper is briefly described as follows:
In chapter 2, we will survey what semantic web development technologies are currently in use and their merits of these existing solutions, and thus readers may fully understand currently used semantic web development technologies.
In chapter 3, we will discus what problems are needed to be solved by associating object-oriented design and semantic web, and give you information about the concept of a total solution you need.
In chapter 4, it describes the architecture of SOF and Class diagrams in order that readers may understand the internal design principle of this system.

In chapter 5, it gives an example of union query of both Gmail and ThunderBird address books that are actually applied to the reality.
In chapter 6, it describes the trend for future study.
In chapter 7, it gives a conclusion of this paper.

## 2. Survey

一粹翻譯社 樣本 Elegant Translation Service Sample 請勿複製 Do not copy

An introduction to the background of the development of four semantic web solutions, Jena [cited from Jena Document], ActiveRDF [cited from ActiveRDF Document], D2R[cited from D2R Document], and EClass [cited from EClass Document] will be given in the following paragraphs, and these four solutions provide developers with various optional layers. Due to the significant diversity among them, readers may fully understand the merits and demerits of present semantic web development solutions after completing the introduction of these four solutions.

### Using Jena Solution

Jena is a present most popular Java RDF APIs that can utilize Triple Oriented APIs to read/write and query RDF data. The main advantage of this solution is that it entirely supports RDF low level operation and is also widely used. Accordingly, the stabilization of implementing products can be obtained. Owing to low level APIs supported and the lack of the integration of object-oriented programming design, each operation step must be described in detail at the phase of use, and however, even the consideration of the abstract idea of object is also not available.

### Using ActiveRDF Solution

ActiveRDF is a RDF object-oriented API that is based on Ruby programs, and it further abstracts the Triple Oriented APIs, and uses object-oriented methods to manipulate RDF document, and the lower level implementation still stores the results derived from the conversion of data in Triple Oriented Storage, and the purpose of which is to simplify the difficulty of calling APIs. The purpose of such a solution of object-oriented APIs is not for the provision of automatically publishing existing Model object data as semantic web for further query. Additionally, due to the limitation of API implementation, this solution only provides for the use of single programming langue, but not for cross-programming language use.

Using D2R Solution

D2R is a special solution, and its idea is to directly convert the records in relational database to RDF format for facilitating the read/write or query of RDF. Because the target being manipulated is database, D2R is applicable to any programming language, so that D2R will fully automatically take the action of format conversion, and the intervention of programmers to convert data is not necessary as long as the mapping relationship between database tables and RDF is clearly specified. Again, due to the database used as a target being manipulated by D2R, the object-oriented encapsulation is not supported, and thus there is no way to manipulate data by using objects.

Using EClass Solution

EClass solution proposes an idea for changing Java or C++ syntax, and it renames the Class declared in Java language as EClass, and thus the definition of language has been directly changed. Capital E means extension, because it allows developers to define semantic relationship between attributes. However, if you want to change programming syntax that has been widely used, you will meet an obstacle, the syntax definition used must not only affect existing programming languages, but also describe the semantic relationship between classes and attributes. Moreover, the query function of cross-heterogeneous data objects is not yet mentioned in EClass solution, and thus this should be enhanced in the future.

## 3. Problem Statement

This chapter will discuss some problems resulted from the combination of semantic web and object-oriented design, and show a table concluded from what problems have been solved by existing solutions so that readers may refer to the main diversity between solutions. The following will discuss what functionalities and features must be supported for each issue while combining semantic web and object-oriented design.

一样翻譯社 樣本 Elegant Translation Service Sample 請勿複製 Do not copy

## 3.1 Manipulating RDF API Abstraction by Using Object Methods

Although low level RDF APIs provide entire RDF read/write and query operations, developers have no way to utilize objects to manipulate RDF data, therefore the duration of development is longer and program codes are relatively larger and thus the maintenance is not a ‘piece of cake’. Therefore, object-oriented design is utilized to abstract RDF API in order that developers may more effectively write productive program codes.

For solving this problem, SOF allows developers to utilize object-oriented APIs to make a query, and its corresponding query result may be also returned by means of data objects.

一祥翻譯社 樣本 Elegant Translation Service Sample 請勿複製 Do not copy

## 3.2 Automatically Converting Data into RDF Format

Although RDF APIs may store data in Triple Store for the purpose of semantic query, developers themselves must accomplish the action of converting data objects into Triple Store format, and as you know, this is a trivial matter and time consuming. In other words, if there is a development architecture which has an ability to fully automatically convert data objects into RDF format for publishing, the time to develop system can be significantly saved.

For solving this problem, SOF allows that developers don’t need to carry out the responsibility of converting data to RDF, and quite the contrary, the course of publishing data objects as RDF format can be fully automatically done by SOF, and thus significantly reduce developers’ burden. Moreover, SOF also provides an embedded Web Server which allows programs provided by the 3rd party to use HTTP Protocol to read latest RDF format data.

## 3.3 Supporting an architecture for Various Programming Languages

At present, for aiming at various programming languages, some object-oriented APIs may manipulate RDF documents [cited from Python, Ruby ActiveRDF, and related APIs], whereas these implementations target to specific languages, and thus an inconsistency exists between them.

For solving this problem, SOF syntax is not specially bound to a certain object-oriented programming, since SOF utilizes comments to describe the semantics of classes and attributes so that SOF Parser can be repeatedly used through a minimum modification needed in various programming languages. And meanwhile, many programmers just only learn one usage of SOF and thus they can be applied in various programming languages.

## 3.4 Using Statements to Describe the Semantics of Classes and Attributes

The natural way to combine both semantic web and object-oriented design is to describe the semantics of classes or attributes. At the time of defining classes, you should also clearly express the semantic relationship, and lately, the query of data objects can be taken. However, in case of that classes and attributes need to be modified, you must usually modify the syntax of programming language so that modifying the semantic relationship of both classes and attributes under the case without adversely affecting the original programming syntax is an important issue.

一樣翻譯社 樣本 請勿複製 Elegant Translation Service Sample Do not copy

For solving this problem, SOF utilizes embedding comments to allow using part of RDF and OWL syntaxes for modifying the semantic relationship between object-orient classes and attributes.

## 3.5 Momentarily Maintaining the Synchronization of Semantic Description File and Class Definition

In present semantic web implementation, some solutions provide independent RDF semantic description files to additionally modify the relationship between existing data, whereas this will result in the need of momentarily maintaining synchronous update between files otherwise the inconsistency will occur.

A well known example is that program API description document and program codes themselves are mutually independent files, so that the description document is not usually updated in time with the result of obsolete and erroneous description.

For fully combining program description documents and program codes themselves in order to prevent the inconsistency between them, JavaDoc utilizes a way of comments embedded in program codes, and thus programmers may easily maintain the consistency between JavaDoc and program codes. [cited from JavaDoc Technology][cited from PyDoc Technology]

For solving this problem, SOF applies similar principles to maintain the synchronization of program codes and semantic description, including a way of embedding in program codes.

## 3.6 Supporting the Inheritance between Classes and Attributes and the Query of Heterogeneous Data

In different databases, the situation of entirely different column names with the same meaning is usually occurred, e.g. the column name in Email in database A may be called “email”, but in database B, it may be called “mail”. In case of progress of union query of all Emails stored in two different databases, the semantics of both email and mail must be clearly defined so that computer may essentially know both terms mean the same thing. We now have no a good architecture to define the semantic relationship between classes and attributes in object-oriented programming codes in order that the system may automatically handle the query function of different attribute names with the same meaning.

一祥翻譯社 樣本
Elegant Translation Service Sample
請勿複製
Do not copy

In addition, the union query of heterogeneous data source may also create such a case, e.g. in data objects resulted from query, they may pertain to different classes, and thus a mechanism should be provided in order to allow object-oriented programming codes to distinguish classes pertaining to different data objects, and respectively manipulate attributes based on diverse classes that attributes belong to.

For solving this problem, SOF allows you to utilize comments to maintain an inheritance relationship between attributes, and allows developers to make union query of heterogeneous data objects.

## 3.7 Consistency Check of Data and Semantics

After modifying the semantics of classes or attributes, the situation of some conflicts between data objects and semantics may occur, e.g. in account management system, if we assign an Email value pertaining to one unique account, then a conflict may be occurred when two accounts have the same

Email value. A good solution must be easily applied to figure out this conflict in order to ensure the consistency of data and semantics.

For solving this problem, SOF provides APIs for querying objects in order that developers can make consistency check of semantics.

**Comparison table of SOF and existing solutions (Jena/ActiveRDF/D2R/EClass)**

This paper proposes Semantic Object Framework (SOF) that is designed to fully solve seven problems mentioned above. With reference to the content in Table 1, it describes the detailed comparison among five semantic web development schemes that are used to effectively solve seven problems mentioned above, wherein X denotes "impossible to solve this problem" while O denotes "this problem can be solved".

一祥翻譯社 樣本
Elegant Translation Service Sample
請勿複製
Do not copy

Table 1: A function comparison table of five semantic web development schemes

| Problem | Jena | ActiveRDF | D2R | EClass | SOF |
|---|---|---|---|---|---|
| 3.1 | X | O | X | O | O |
| 3.2 | X | X | O | O | O |
| 3.3 | X | X | O | X | O |
| 3.4 | X | X | X | O | O |
| 3.5 | X | X | X | O | O |
| 3.6 | X | X | X | X | O |
| 3.7 | X | X | X | X | O |

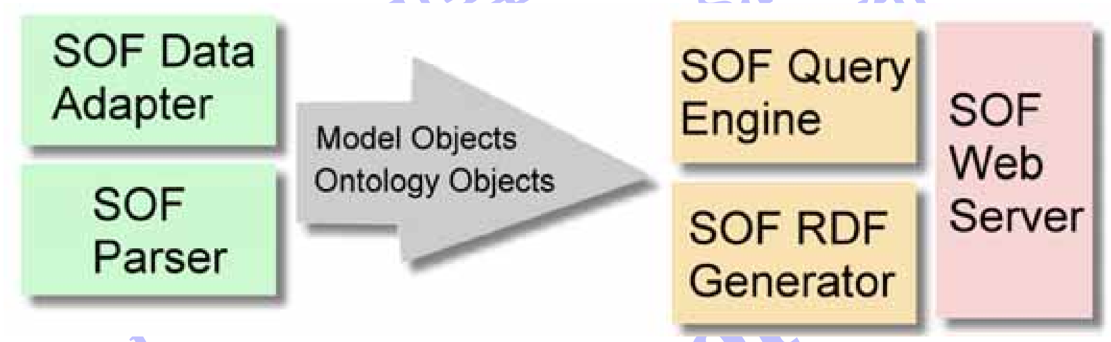
## 4. Solution

This section proposes the SOF architecture to solve seven problems mentioned in Chapter 3, and also applies an architecture diagram to describe how the SOF architecture accomplishes its design target. Finally, in this section, we use an example of the heterogeneous address book query to allow readers to experience how SOF successfully integrates object-oriented design and semantic web technologies.

SOF Architecture

An introduction to the SOF architecture will be given below, including an introduction to the use of five main modules as well as the workflow of input/output relationship among them.

[Diagram 1] Five SOF primary modules



SOF is consisted of five mail modules, with reference to diagram 1, an arrow shape model objects include data object of data content while ontology objects include the semantic relationship between classes and attributes. The main use of five modules will be described respectively in the following paragraphs.

SOF data adapter

The function of this module is to read various data sources to convert these data sources to model objects. Data sources can be CSV file format or records in database or APIs used for reading various proprietary data. Programmers may also write data adapters for other data sources so that the consistency of data process for SOF can be accomplished once the data sources are converted to model objects. Meanwhile, SOF provides a flexible architecture with various data source format for the purpose of future use.

The output of SOF data adapter is model objects wherein data are presented by way of objects, and model objects include all actual data contents, i.e. attribute value of each data and, moreover, they will become input parameters of both SOF Query Engine and SOF RDF Generator.

## SOF parser

The function of this module is to parse SOF statements from the comment lines in source program in order to generate ontology objects. For providing independent features, SOF parser must support the use for various popular objected-oriented languages, whereas the description of comments in each language is not quite the same so that SOF utilizes a writing syntax cross different languages, and thus SOF statements can be merged with all comments in different programming languages, meanwhile all programming languages share the same SOF parser codes.

The output of SOF parser is ontology objects wherein the semantic relationship of both classes and attributes are presented by the representation of objects, and ontology objects will become input parameters of SOF RDF Generator and SOF Query Engine.

一祥翻譯社 樣本 Elegant Translation Service Sample 請勿複製 Do not copy

## SOF RDF generator

The purpose of this module is to output model objects in RDF format in order that the codes provided by third parties can read RDF format data due to that the semantic relationship among model objects has been recorded in ontology objects so that RDF format files including semantic relationship can be ultimately generated.

## SOF query engine

The purpose of this module is to provide object-oriented APIs with union query cross-heterogeneous data sources. In addition to the utilization of the object-oriented style by query APIs, the query results will return to developers in a way of the union of object arrays. Due to that the array of model objects returned may be respectively dedicated to multiple different classes, suitable APIs for type conversion must be provided to handle the issues of format conversion.

SOF web server

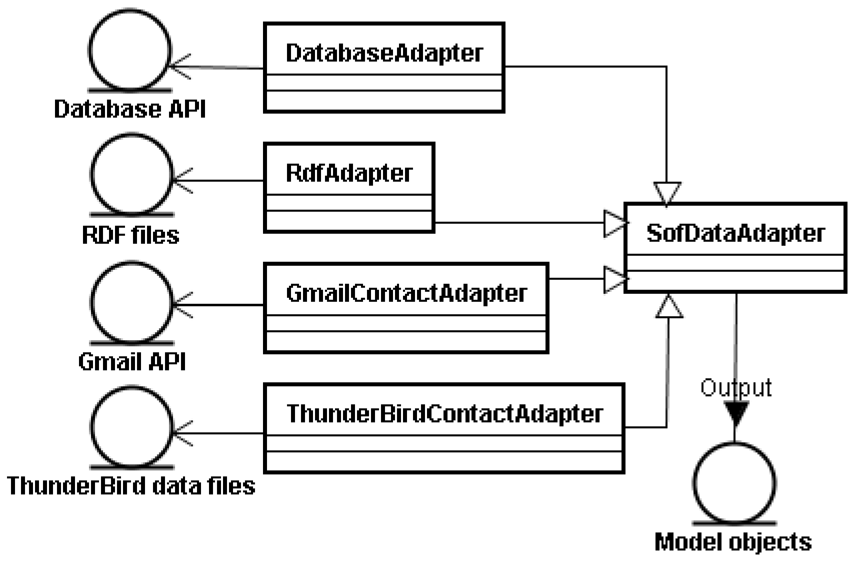
The purpose of this module is to provide an entry for HTTP Protocol in order that programs provided by third parties can read latest RDF documents due to that SOF utilizes a way of dynamic conversion to convert data to RDF, so that any RDF documents are read from SOF web server, any changes of the latest content of model objects can be guaranteed, an thus we are no longer concerned about the consistency of data.

一祥翻譯社 樣本 請勿複製
Elegant Translation Service Sample
Do not copy

SOF Class Diagram

An introduction to a class diagram of four mail modules will be given in the follow paragraphs, wherein SOF web server only provides interfaces for request from external and thus the description of this class diagram will be ignored here.

[Diagram 2] Class diagram of SOF data adapter



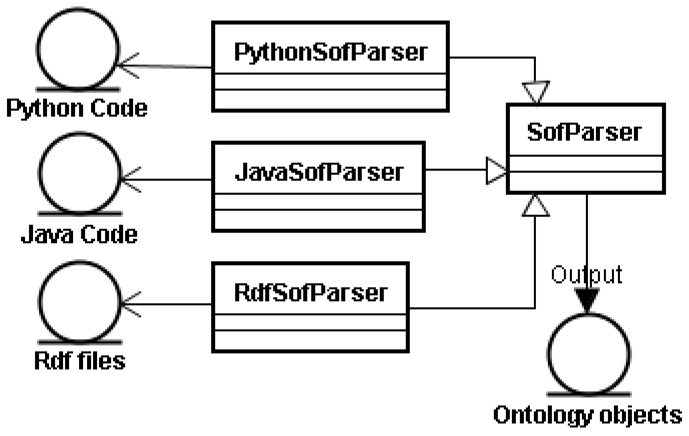
The input terminal of SOF data adapter may be various data sources, and after the model object format outputted is converted from these data, object-oriented APIs can be utilized to read and write these model objects.

With reference to diagram 2, we here may observe four different SOF data adapters, wherein: DatabaseAdapter may read records through database APIs, RdfAdapter reads data files in RDF format, GmailContactAdapter reads address book data through Gmail APIs, and finally ThunderBirdContactAdapter reads address book data file format from ThunderBird, whereas all these four different SOF data adapters are inherited from SofDataAdapter class so that they have common operation methods.

In object-oriented programming, Model-View-Controller (MVC) is a usually used method (it is cited from related MVC papers) wherein Model represents data themselves. The ultimate output format of SOF data adapter is aimed at Models in MVC. In general, model objects provide operation methods of reading and writing object attributes.

一祥翻譯社 樣本 Translation Service Sample 請勿複製 Not copy

[Diagram 3] Class diagram of SOF parser

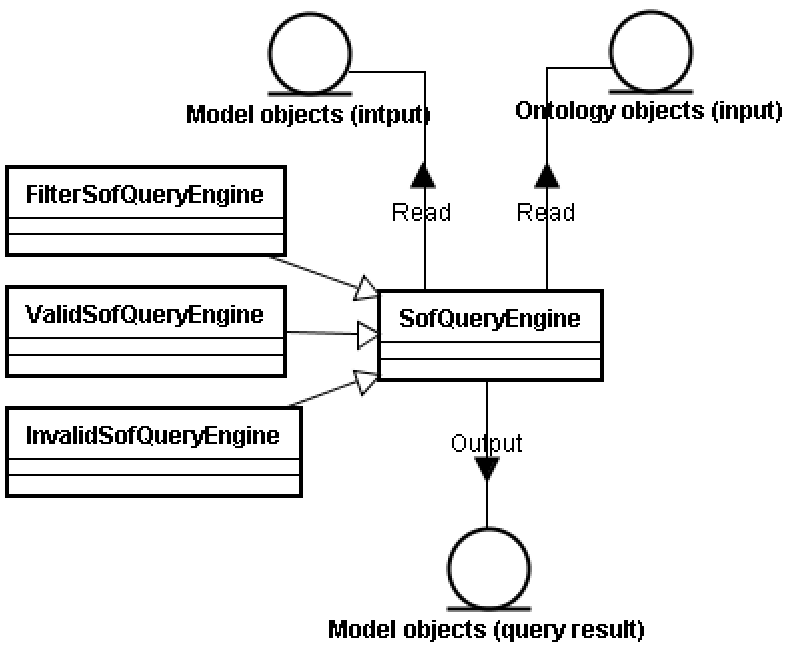


The input of SOF parser is source codes of various programs, and these source codes include SOF statements, wherein these SOF statements describe the semantic relationship between classes and attributes, and SOF parser will convert SOF statements to ontology objects and output them.

With reference to diagram 3, we may observe three different SOF parsers, wherein: PythonSofParser is responsible to read Python codes, JavaSofParser is responsible to read Java codes, and RdfSofParser has the responsibility of reading the semantic relationship of classes in RDF file format. These three classes are inherited from SofParser class, and shared program codes for these three classes may be implemented in SofParser class.

Ontology objects outputted by SOF parser record the semantic relationship between classes and attributes, and those can be also presented by the representation of objects. If ontology objects and model objects are used at the same time, then the semantic query to cross-heterogeneous data sources can be taken on model objects.

[Diagram 4] Class diagram of SOF query engine



The inputs of SOF query engine are model objects and ontology objects, wherein model objects can be object array combined from various classes, and ontology objects may explain inheritance relationship and semantic relationship between these model objects. SOF query engine may accept query statements, and output the query results in the form of model objects.

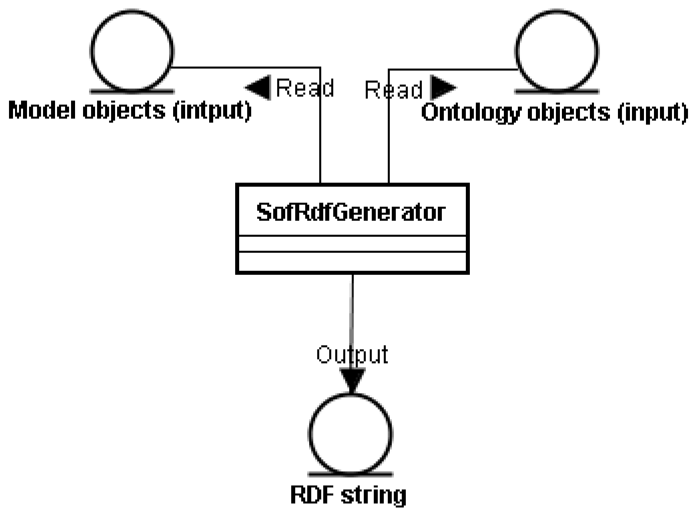
With reference to diagram 4, three different SOF query engines are shown, wherein: FilterSofQueryEngine has a responsibility of conditionally filtering semantic query, ValidSofQueryEngine is responsible for querying model objects that are coincided with semantic rules, and InvalidSofQueryEngine aims at querying model objects pertaining to illegal semantics. These three are inherited from SofQueryEngine, and output results are all model objects.

If ultimately generated query results, model objects, are outputted from

FilterSofQueryEngine, only model objects accorded with query conditions will be listed, and at the time of query, you may enter object arrays for various classes, so the query results also include objects used by different classes. Due to that program developers may use APIs to obtain original class types of model objects, special processes can be taken for different classes, if necessary. If the query results belong to the output of InvalidSofQueryEngine, then model objects will also include the reason descriptions why this object is classified as an illegal object in order that developers know how to modify this error.

祥翻譯社 樣本 Service Sample 勿複製 ot copy

[Diagram 5] Class diagram of SOF RDF generator



The inputs of SOF RDF generator are model objects and ontology objects, respectively, and this SOF RDF generator may output RDF strings including semantic relationship between classes and attributes by combining these two inputs.

With reference to diagram 5, the last RDF string output may be directly stored in file or accessed by other applications with HTTP service through SOF web server. Due to that RDF string is W3C standard format and includes the data contents of model objects as well as the semantic relationship of ontology objects, any application which can handle RDF format can easily query and merge RDF strings.

## 5. Examples

This Chapter will utilize realistic examples to describe how to use SOF. SOF provides two main functions, wherein one is used to automatically convert data objects and publish RDF format derived therefrom, and the other one is used to make semantic query over cross-heterogeneous data sources. In our examples, address book data supported by two different softwares, Gmail and ThunderBird, will be used to demonstrate the main functions of SOF. Due to that different attribute names are utilized by these two address books, data format is not quite the same, so that if developers want to write programs for making a query over different address books, they will suffer many bothersome format conversion flows. Here, we take Python language as an example, SOF is utilized to add semantic relationship to the attributes applied by Gmail and ThunderBird address books at the time of the declaration of classes. After the completion of establishing semantic relationship, two functions of SOF will be demonstrated, wherein one is to utilize SOF to automatically and simultaneously publish these two address books as RDF format, and the second one allows you to make semantic query over different heterogeneous address books.

一行翻譯社 樣本
Elegant Translation Service Sample
請勿複製
Do not copy

### Using OWL Syntax to Define Two different Address Book Classes in Comments

Before making a union query over two different address books, we first define a class named “Contact” in order that this class may have common attributes used by these two address books, and from the view of semantics, this class will be lately inherited by GmailContact and ThunderBirdContact.

```
class Contact(Model):
    partOfName=''
    partOfAddress=''
    #owl:InverseFunctionalProperty Contact_email
    email=''
    phoneNumber=''
    #Contact_officePhoneNumber rdfs:subClassOf Contact_phoneNumber
    officePhoneNumber=''
```

```
#Contact_homePhoneNumber rdfs:subClassOf Contact_phoneNumber
homePhoneNumber=''
#Contact_mobilePhoneNumber rdfs:subClassOf Contact_phoneNumber
mobilePhoneNumber=''
#Contact_faxPhoneNumber rdfs:subClassOf Contact_phoneNumber
faxPhoneNumber=''
```

Due to that Contact class in classical Model-View-Controller (MVC) design model belongs to Model data class, we declare class Contact(Model) to represent Contact inherited to Model class. Data classes inherited to Model can be serially stored in database, and conditional data query is allowed.

一祥翻譯社 樣本 Elegant Translation Service Sample 請勿複製 Do not copy

The meaning of the presentation of the attribute name partOfName is a contact person's name, and the possibility of the presentation of contact person's name contains surname/name/middle name/full name/nickname etc. We here allow partOfName to represent any possible segment of name or full name and later, if semantics of any other attributes is inherited to partOfName, and the attribute of which is used to identify one of contact person name strings.

The pound sign '#' used in Python language represents a comment, and due to the SOF syntax is embedded in comments, any 'owl:' or 'rdfs:' included in comments means that this statement is a specific one for SOF. '#owl:InverseFunctionalProperty Contact_email' utilizes OWL syntax to modify its semantics, which means that in case of the coincidence of email strings, the representation of Contact object must be unique person, and this should not be occurred in the case of the coincidence of email strings, but Contact object has two situations mentioned above. In case of that two Contact objects have the same email string in data, SOF thus has ability to find out two Contact objects conflicted and pass it to programmers and thus, they can solve illegal semantics used in data by applying various strategies. These OWL statements are helpful for programmers to apply rich syntaxes to limit the relationship between data objects.

The representation of email attribute is E-mail. However, in different address book software, email attribute name may have the following types: Email/email/mail/Mail/emailAddress/EmailAddress, and the semantics used by these different attribute names are fully identical in meaning. If we want to display all attribute values of all emails cross heterogeneous address books, the semantics of the attribute ‘rdfs:subClassOf inherited to Contact_email’ must be used in attributes in various emails.

Other attributes are easily to be understood so that they are ignored.

Next, we will now take a look on how the GmailContact is inherited to well-defined attributes.

```
#GmailContact rdfs:subClassOf Contact
class GmailContact(Model):
    #GmailContact_name rdfs:subClassOf Contact_partOfName
    name=''
    #GmailContact_email rdfs:subClassOf Contact_email
    email=''
    #GmailContact_phone rdfs:subClassOf Contact_officePhoneNumber
    #GmailContact_phone rdfs:subClassOf Contact_homePhoneNumber
    phone=''
    #GmailContact_mobile rdfs:subClassOf Contact_mobilePhoneNumber
    mobile=''
    #GmailContact_fax rdfs:subClassOf Contact_faxPhoneNumber
    fax=''
    company=''
    title=''
    #GmailContact_address rdfs:subClassOf Contact_partOfAddress
    address=''
```

The representative meaning of #GmailContact rdfs:subClassOf Contact is that the class GmailContact is semantically inherited to Contact class so that if any object query commands are being used to query all Contact data object, GmailContact object inherited to Contact class is also within the scope of targets being queried and later, we will show that ThunderBirdContact is also semantically inherited to Contact, so that when developers want to query data objects from two different address books, Gmail/ThunderBird, SOF may automatically determine that both GmailContact and ThunderBirdContact objects must be involved within the scope of query if the target being queried is Contact class, and thus the purpose of querying heterogeneous address books may easily accomplished.

一祥翻譯社 樣本
Elegant Translation Service Sample
請勿複製
Do not copy

This comment line ‘#GmailContact_name rdfs:subClassOf Contact_partOfName’ describes that the name attribute in GmailContact class is semantically inherited to the partOfName attribute of Contact class. This means that if developers specify the string value of Contact_partOfName attribute that is being queried at the later time, SOF will also automatically query the string value of GmailContact_name attribute.

Next, the statement being introduced is with reference to a multiple inheritance relationship with GmailContact_phone: ‘#GmailContact_phone rdfs:subClassOf Contact_homePhoneNumber’ represents the attribute ‘GmailContact_phone’ is probably be a office telephone, or may be a home telephone. Due to that RDF syntax allows multiple inheritance relationship, SOF may still allow you to semantically describe the multiple inheritance relationship for classes or attributes even if multiple inheritance is not supported in programming languages, such as Java. We take GmailContact_phone as an example, whichever developers choose Contact_officePhoneNumber or Contact_homePhoneNumber as a target being queried at the later time, SOF will always automatically query GmailContact_phone attributes.

一祥翻譯社 樣本 Elegant Translation Service Sample 請勿模製 Do not copy

Due to that the semantic inheritance relationship of other attributes pertaining to GmailContact is very straight, readers may understand their meaning from program code segment, and thus the explanation of which is ignored. The following example is given for the purpose of seeing how ThunderBirdContact is semantically inherited to Contact.

```
#ThunderBirdContact rdfs:subClassOf Contact
class ThunderBirdContact(Model):
     #ThunderBirdContact_mail rdfs:subClassOf Contact_email
     mail=''
     #ThunderBirdContact_givenName rdfs:subClassOf Contact_partOfName
     givenName=''
     #ThunderBirdContact_sn rdfs:subClassOf Contact_partOfName
     sn='' #first name
     #ThunderBirdContact_cn rdfs:subClassOf Contact_partOfName
     cn='' #full name
     #ThunderBirdContact_telephoneNumber rdfs:subClassOf
Contact_officePhoneNumber
```

```
    telephoneNumber=''
    #ThunderBirdContact_homePhone rdfs:subClassOf
Contact_homePhoneNumber
    homePhone=''
    #ThunderBirdContact_fax rdfs:subClassOf Contact_faxPhoneNumber
    fax=''
    #ThunderBirdContact_mobile rdfs:subClassOf
Contact_mobilePhoneNumber
    mobile=''
    #ThunderBirdContact_homeStreet rdfs:subClassOf
Contact_partOfAddress
    homeStreet=''
    #ThunderBirdContact_mozillaHomeLocalityName rdfs:subClassOf
Contact_partOfAddress
    mozillaHomeLocalityName=''
    #ThunderBirdContact_mozillaHomeState rdfs:subClassOf
Contact_partOfAddress
    mozillaHomeState=''
    #ThunderBirdContact_mozillaHomePostalCode rdfs:subClassOf
Contact_partOfAddress
    mozillaHomePostalCode=''
    #ThunderBirdContact_mozillaHomeCountryName rdfs:subClassOf
Contact_partOfAddress
    mozillaHomeCountryName=''
    #ThunderBirdContact_street rdfs:subClassOf Contact_partOfAddress
    street='' #street of company
    #ThunderBirdContact_l rdfs:subClassOf Contact_partOfAddress
    l='' #locality name of company
    #ThunderBirdContact_postalCode rdfs:subClassOf
Contact_partOfAddress
    postalCode='' #postal code of company
    #ThunderBirdContact_c rdfs:subClassOf Contact_partOfAddress
    c='' #country name of company
    title=''
    department=''
    company=''
    mozillaHomeUrl=''
```

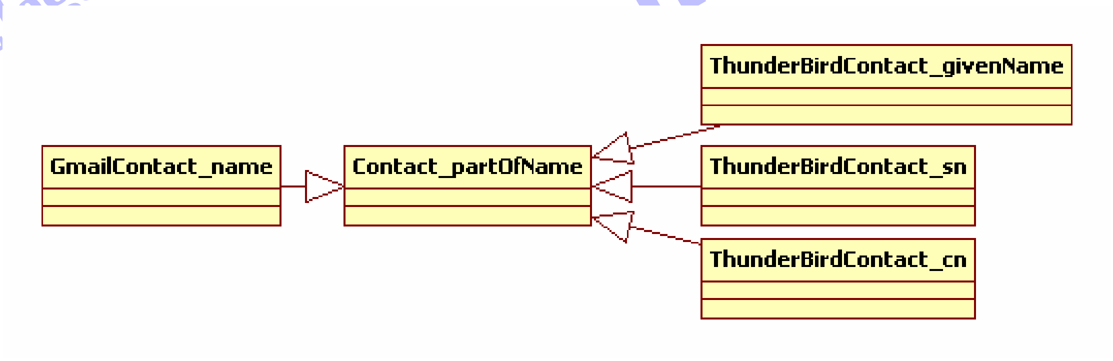
一祥翻譯社 樣本
Elegant Translation Service Sample
請勿複製
Do not copy

ThunderBirdContact class and GmailContact class are all semantically inherited to Contact class, and readers may know this class is more complex than GmailContact from various attributes pertaining to ThuderBirdContact, and particularly, those are with reference to address. No distinction exists between Home address and company address, even though country/county/street attributes are also not distinguished. GmailContact only adopts an address attribute to represent all possible address string. In ThunderBirdContact, the number of attributes with reference to address is up to nine, and these attributes are semantically inherited to Contact_partOfAddress.

With further reference to diagram 6, in ThunderBirdContact, number of attributes pertaining to contact person name is three, while GmailContact has only a name attribute to represent this contact person name. Attributes pertaining to ThunderBirdContact, GmailContact, and name are all semantically inherited to attributes of Contact_partOfName.

一祥翻譯社 樣本 Elegant Translation Service Sample 請勿複製 not copy

Diagram 6 - a diagram of attributes inherited to Contact_partOfName



Up till now, we have established the semantic relationship of Contact, GmailContact, and ThunderBirdContact, because SOF utilizes comments to embed semantic description into programming codes, and thus readers who are reading these codes may easily find out the mapping relationship of semantics between attributes. Next, no matter whether SOF publishes data objects as RDF format or a query is being taken on data objects by SOF, they can be easily accomplished.

## Automatically Publishing Address Books as RDF Format

SOF allows address books are automatically published as RDF format, due to that HTTP access must be adopted in RDF format data, so that an SOF Web Server existed in SOF system is responsible for providing entry point of HTTP, and thus the corresponding data object RDF format can be accessed as long as URL is appropriately used, e.g. http://127.0.0.1:8080/sof/Contact/, which may be used to obtain RDF data format of data objects pertaining to Contact. At this time, data objects pertaining to GmailContact and ThunderBirdContact will be used together in a format of RDF for further reading being taken by developers. If a developer only wants to access RDF pertaining to its sub-class in person, the following URL may be used to accomplished this goal, e.g. http://127.0.0.1:8080/sof/GmailContact/ which may be used to access RDF format data of all data objects pertaining to GmailContact, but not including data objects pertaining to ThunderBirdContact.

Due to that SOF adopts an implementation technology to dynamically convert data objects to RDF format, the latest change data can be obtained from URL each time, and the performance can be promoted by applying Cache technology. If data is not changed, the same can be obtained by adapting last RDF result from Cache; if data is changed, it is needed to automatically regenerate a RDF format document.

一祥翻譯社 樣本
Elegant Translation Service Sample
請勿複製
Do not copy

## Making a Querying on cross-heterogeneous address books

After address books pertaining to Gmail and hunderBird are published as semantic web, the union query on cross-heterogeneous database sources is one of most useful functions in semantic web. The following codes has ability to find out data objects pertaining to email attribute ending with ‘nctu.edu.tw’ string from sub classes inherited to Contact.

It is noted that both GmailContact and ThunderBirdContact represents Email attribute names are not the same. In Gmail, the Email attribute is called email, but in ThunderBird, Email attribute is called mail. Although both attribute names are not the same, the function of union query is not adversely affected, because they are inherited to attributes pertaining to Contact_email, so that all matched data objects will be found.

```
lstContact=Contact.objects.get("email like '%nctu.edu.tw'")
intCounter=0
for contact in lstContact:
    intCounter+=1
    print '=== Contact %s ==='%intCounter
    print 'partOfName:\n      %s'%contact.partOfName
    print 'email:\n      %s'%contact.email
```

Code segments described above will find out data objects with Email name ended with ‘nctu.edu.tw’ from all classes inherited to Contact, wherein Contact.objects.get is a key statement to query objects, and its syntax is similar to SELECT command used in database SQL. The following statement is show while that is compared to SQL command :

```
select * from Contact where email like '%nctu.edu.tw'
```

After matched data objects are found, they may have two different types: GmailContact or ThunderBirdContact. A for loop is then followed, and partOfName and email attributes pertaining to data objects found are displayed on the screen. The result is shown below:

```
=== Contact 1 ===
partOfName:
    "GmailContact_name":"Bowen Chiu",
email:
    "GmailContact_email":"bowen@nctu.edu.tw",
=== Contact 2 ===
partOfName:
    "ThunderBirdContact_givenName":"Kao",
    "ThunderBirdContact_sn":"Gloria",
    "ThunderBirdContact_cn":"Gloria Kao",
email:
    "ThunderBirdContact_mail":"gloria@nctu.edu.tw",
```

In this case, two records about data objects are displayed, wherein the first record is that Contact 1 pertains to data objects of GmailContact class, and the string value of contact.partOfName is "GmailContact_name":"Bowen Chiu".

一祥翻譯社 樣本 Elegant Translation Service Sample 請勿複製 Do not copy

From it, we found this is a key:value pair, wherein the leading word, ‘key’, is GmailContact_name which allows developers know a name “Bowen Chiu” followed belongs to GmailContact_name.

The second record, data objects pertaining to Contact 2, belongs to ThunderBirdContact class, so that the representative meaning of contact.partOfName is more complex, the value of contact.partOfName here corresponds to an array that is delimited by the comma, ‘,’, character, and key:value pair for givenName, sn, cn three attributes in ThunderBirdContact is listed due to that these attributes in ThunderBirdContact class are inherited to contact.partOfName, so that key:value pairs of these three attributes will be put in the query result of contact.partOfName after the result of union query comes out.

一样翻譯社 樣本 Elegant Translation Service Sample 請勿複製 Do not copy

If it is needed to provide different data display method for each class, developers may determine a class that this object belongs to in accordance with ‘key:value’ pair pertaining to returned data objects, so that at the time of union query, display format for different classes can be properly adjusted or you don’t need to adjust display format in case of simple use, and all ‘key:value’ pairs found will be directly displayed on the screen.

From this, we may see the strength of query made over cross-heterogeneous address books by SOF. You only simply describe the inheritance relationship between attributes described above in comments, ad thus different attribute names but with the same meaning can be distinguished so that it is easy and convenient for you to find out all matched data at the same time.

### Querying Data Sets with Legal or Illegal Semantics

Due to that RDF may aim to restrict the semantic relationship between objects, it is possible that limitation conditions assigned to certain data objects may be illegally assigned by RDF, so that in some situation, we may need to distinguish between legal data or illegal data.

For example, if we wish to find out non-duplicated mail name lists, wherein GmailContact and ThunderBirdContact may probably contains duplicated contact person data, an SOF statement ‘#owl:InverseFunctionalProperty

Contact_email' previously defined can be utilized at this moment. In the statement, the value of Contact_email attribute which is limited must belong to unique Contact, i.e. any two Contact data objects may not have the same Email value. If a same Email attribute value exists in more than two Contact data objects, they will be viewed as the same Contact from the view of semantics. Through such a limitation, we may utilize getInvalid() API to find which data objects violate this principles, and display them on the screen. We here take the following program segment as an example:

```
lstContact=Contact.objects.getInvalid()
for contact in lstContact:
    print 'partOfName:%s'%contact.partOfName
    print 'phoneNumber:%s'%contact.phoneNumber
    print 'invalid reason:%s'%contact.getInvalidReason()
```

一抒翻譯社 樣本 請勿複製 Elegant Translation Service Sample Do not copy

The first illegal data is shown below:

```
partOfName:
   "GmailContact_name":"Bowen Chiu",
phoneNumber:
   "GmailContact_phone":"+88635727001",
   "GmailContact_mobile":"+886922387002",
invalid reason:validation fail ->
owl:InverseFunctionalProperty(Contact_email,Contact)
```

The second illegal data is shown below:

```
partOfName:
   "ThunderBirdContact_givenName":"Chiu",
   "ThunderBirdContact_sn":"Bowen",
   "ThunderBirdContact_cn":"Bowen Chiu",
phoneNumber:
   "ThunderBirdContact_telephoneNumber":"+88635727001",
   "ThunderBirdContact_homePhone":"+88638885003",
   "ThunderBirdContact_mobile":"+886993288002",
invalid reason:validation fail -> owl:InverseFunctionalProperty Contact_email
```

We found that the first record of illegal data objects belongs to GmailContact, and the second one pertains to ThunderBirdContact. Although they are different data objects, they are considered as illegal data because of identical Email attribute values, i.e. the same Contact is used from the view of semantics. SOF successfully cross two different address books, thus it is important that the function to find semantically duplicated data object is generally used in printing non-duplicated mail name list and thus the same thing mailed to the same person can be avoided. Meanwhile, contact.getInvalidReason() command may even display the cause of the violation of RDF semantic limitation resulted from data objects for further actions being taken by developers, such as deleting a redundant data object or merging these two data objects into one.

If you intend to use a command to read all legal data objects, ‘lstContact=Contact.objects.getValid()’ can be used to add all legal data objects into lstContact array, wherein these objects are Contact data objects which have no duplicated Email attributes.

一祥翻譯社 樣本 Elegant Translation Service Sample 請勿複製 Do not copy

## 6. Future Work

SOF architecture proposed in this paper may automatically publish data objects as RDF format, and the union query can be also taken on heterogeneous data. Although this is a powerful class/object publication flow, the tool modules extended from the SOF idea are not quite enough, especially in the part of automation support for IDE development tools. If an IDE development environment for various languages to support auto complete, dynamic syntax checking, and even the semantic relationship between class which are graphical representation can be established in the future, the mutual synchronization between semantic diagrams and program codes may be accomplished, and thus an abstract development environment for specifying the semantic relationship of classes can be utilized. These mentioned above are the case of applying SOF idea to gradually accomplish the desired goal.

In addition, in case of that an illegal syntax of SOF statement occurs or a conflict of semantics exists between SOF statements, we need more powerful tool to automatically check such a problem and report the checking result to developers for further actions in the future. We do believe that related development tools will be well supported after SOF idea is getting more considerable and introduced.

## 7. Conclusion

This paper concentrates on the publication of data objects to RDF document that provides SOF solutions for automatic conversion. In the past, in case of publishing data objects as RDF, the publication can generally be done through manually converting data objects to Triple Store. In this paper, SOF idea is used to modify semantics between classes and attributes that are embedded in program codes, and enhance the lack of the description of relationship between classes and attributes in object-oriented languages. In addition to remaining the synchronization of relation description between classes and program codes, SOF may also provide tools set independent of programming languages that can support for various languages and are not limited by the implementation in a specific language. SOF provides a quite direct publication flow for semantic web, allows you to make a query over cross-heterogeneous data sources, and successfully incorporate the merits of both object-oriented programming and semantic web.

一祥翻譯社 樣本
Elegant Translation Service Sample
請勿複製
Do not copy