

## Chapter 3 Research Method

### 3.1 A Ontology-Based Method

As we mention in section 2.3.6, we need a common approach to build up our ontologies for different B2B standards. In this chapter, we present a ontology-based method to model the B2B domain. We develop a system analysis approach for the B2B initiative. Then, we model the business processes and the business documents in the ontology language, OWL. There are seven main steps in this method (shown in Figure 3-1). We analyze the current business process and eCommerce Standard process (step A and B). After analyze the process , we develop a heuristics based method to model business process (step C and D). We get the ontologies from step C and step D. We merge the ontologies (step E) and represent ontologies (step F). Finally, we test these ontologies (step G).

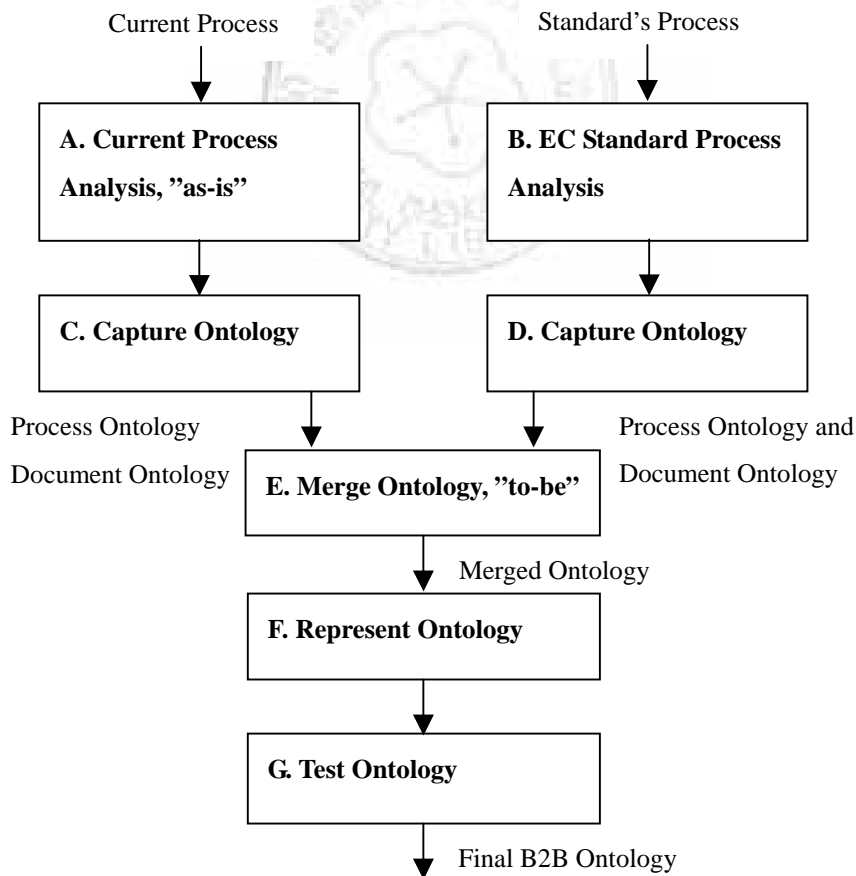


Figure 3-1: The Structure of Ontology-Based B2B Integration Method (This Research)

## 3.2 Business Process Modeling – UML-based

### 3.2.1 UML

The UML is a standard language from the Object Management Group (OMG) with an associated graphical notation for object-oriented analysis and design (OMG, 2003). The UML is a very important part of developing object oriented software and software development process. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software. UML defines nine types of diagrams: class, object, use case, sequence, collaboration, statechart, activity, component, and deployment. We introduce the nine diagrams briefly as follows:

**Use Case Diagrams:** Use case diagrams model the functionality of system using actors and use case.

**Sequence Diagrams:** Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

**Activity Diagrams:** Activity diagrams illustrate the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.

**Class Diagram:** Class diagrams are the backbone of almost every object-oriented method, including UML. They describe the static structure of a system.

**Object Diagram:** Object diagrams describe the static structure of a system at a particular time. They can be used to test class diagrams for accuracy.

**Collaboration Diagrams:** Collaboration diagrams represent interactions between objects as a series of sequenced messages. Collaboration diagrams describe both the static structure and the dynamic behavior of a system.

**Statechart Diagrams:** Statechart diagrams describe the dynamic behavior of a system

in response to external stimuli. Statechart diagrams are especially useful in modeling reactive objects whose states are triggered by specific events.

**Component Diagrams:** Component diagrams describe the organization of physical software components, including source code, run-time (binary) code, and executables.

**Deployment Diagrams:** Deployment diagrams depict the physical resources in a system, including nodes, components, and connections.

Business process is the unique way in which the organization coordinates and organizes the different working activities and tasks to produce a product or a service. We model the organization's business process to show a set of activities and their relationships. Although, UML has nine diagrams, we only adopt four diagrams here. They are the use case diagram, sequence diagram, activity diagram and class diagram to analyze and model the business processes. This is because use case diagram, sequence diagram and activity diagram are more suitable to describe business process than other diagrams. Furthermore, class diagram is easier to convert to ontology, which we will discuss later.

Use case diagram is a convenient way to present business processes in a visual view. It uses simple notation that is easy to build and easy to understand. In addition, sequence diagram and activity diagram are the other ways to capture the detailed business process information and provide more information in order to supplement use cases. Sequence diagram emphasizes the time ordering of messages. Activity diagram shows the flow from activity to activity (Booch, 1999). Finally, we convert the use case diagram, the activity diagram, and the sequence diagram to the class diagram. The class diagram is the most natural diagram mapping to the ontology language.

#### **A. To analyze the current business process, “as-is”**

If we want to analyze the current process, we should initiate a meeting to discuss it. The meeting participants should include the process owners and users, because they understand the processes the most. Through interviewing users, we can discover detailed information about the current processes. The detail information includes the process goal, the process flow, the process user role, the process input, the process output and others. This information should be minuted. According to the meeting minutes, we start to draw the UML diagram. If we understand the current processes more, we will present the process in UML more correctly without losing its semantics. The final drawn diagrams should be presented to the process owner. The process

owner will verify the diagrams match the actual conditions.

**A.1 The use of Use Case Diagram**

Before we draw a use case diagram, we have to gather data. We analyze the process actors, the process preconditions, and the process flow to fill in a form. We take the purchase order as an example. There should be two actors in the purchase order (PO) process: buyer and seller. Before the buyer orders something, he needs make a request for a quote (RFQ) document from the seller first. Then, if the buyer accepts the quote, he sends a purchase order to the seller. When the seller receives the purchase order, he will confirm the order. This scenario is the simplest situation. We fill in the information in the following table.

Table 3-1: A Use Case Example (This Research)

Name: Request Purchase Order	
Actors	Buyer, Seller
Preconditions	The request for quote document exists
Main flow	1. Send a PO 2. Receive a PO confirmation
Alternative flow	None

We transfer the textual description of use case to the use case diagram. The translation method has described by many books such as “*UML Distilled: A Brief Guide to the Standard Object Modeling Language*” (Fowler, 2002).

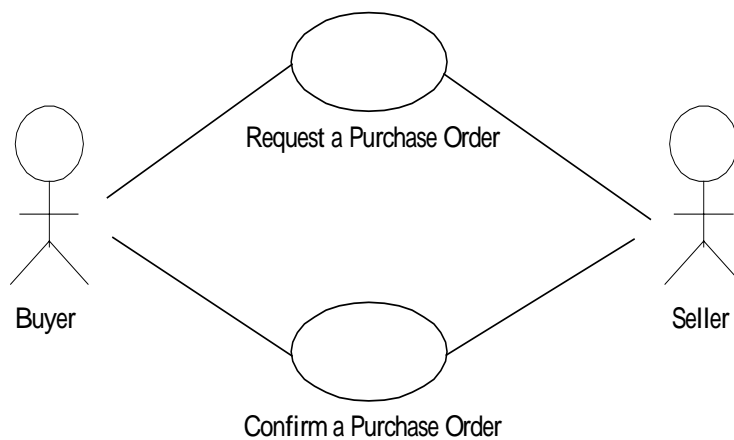


Figure 3-2: A Use Case Diagram Example (This Research)

## A.2 The use of Sequence Diagram

In a sequence diagram, we discover all messages that are exchanged in a business process and their order. It can be extracted from the use case diagram and the meeting minutes. In the PO example, the PO Request is the first message to be sent from the buyer to the seller. When the seller receives the order request, he should check his inventory to determine whether he can fulfill that order or not. Then the PO Confirmation is the next message to be sent from the seller to the buyer. We draw this scenario in sequence diagram as follows.

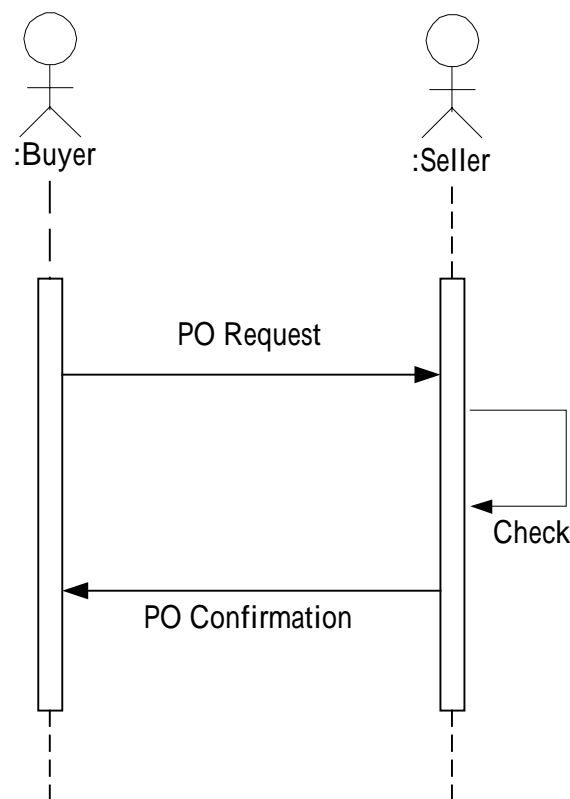


Figure 3-3: A Sequence Diagram Example of Purchase Order (This Research)

## A.3 The use of Activity Diagram

An activity diagram shows the flow from activity to activity. It can present the detailed process flow. We should find the information from discussion at the meeting in order to develop the activity diagram. We need to discover the detailed actions in the flow, initial state and final state. We continue the PO example and finish the activity diagram. In this example, we have three actions: request a purchase order, check inventory for this order and confirm this purchase order.

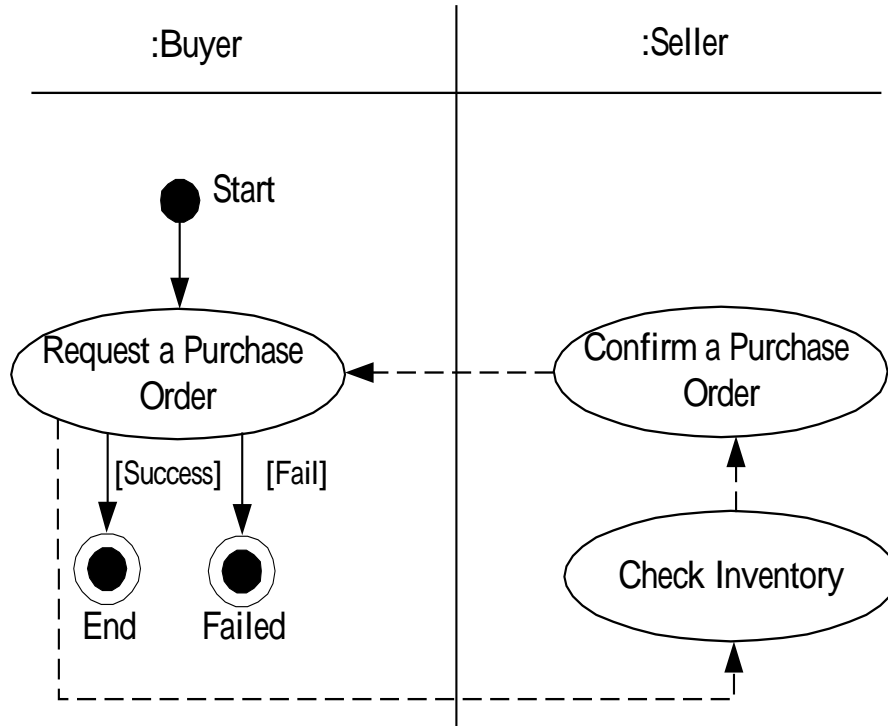


Figure 3-4: An Activity Diagram Example of Purchase Order (This Research)

#### A.4 The use of Class Diagram

We try to extract a generic class concept from the use case diagram, sequence diagram and activity diagram. (As shown in Figure. A Class Diagram Example of Purchase Order) Again, we continue with the PO example. First, we work on the use case diagram, the Figure 3-2: An Use Case Diagram Example of Purchase Order. We discover four components: the two actors (buyer and seller) and the two use cases (Request a Purchase Order and Confirm a Purchase Order). We take the two major elements in the use case diagram, Actor and Use Case, to form the two classes: Actor and Activity. Next, we extract the class Message from the sequence diagram, because sequence diagram describes the message flow and the order between the objects. Then, we work on the activity diagram and we find it consists of several actions. We extract the class Action from the activity diagram. Then, we consider the multiplicity of these classes. However, this generic class diagram does not present the semantic of this PO example. We use generalization to link the Buyer, the Seller and the Actor. The class Buyer and Seller is the subclass of Actor.

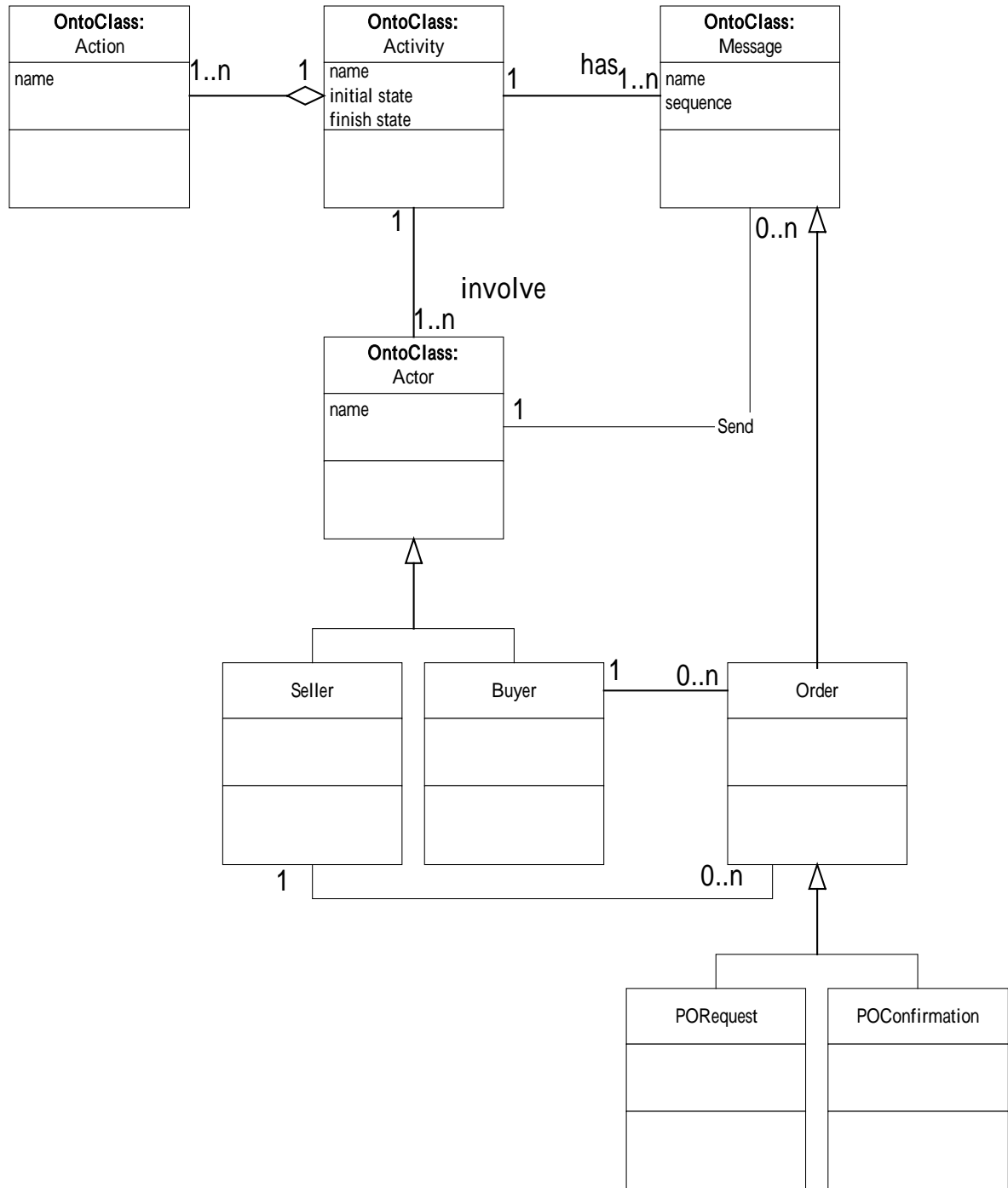


Figure 3-5: A Class Diagram Example of Purchase Order (This Research)

**B. To develop the EC-standard-compliant business process**

We use the four diagrams (use case diagram, sequence diagram, activity diagram and class diagram) to model an EC-standard-compliant business process. The mapping methods between the four diagrams are the same as in step A. The difference between step A and step B is the analytic source. Step A focuses on the existent and

current processes. We have to analyze them through interview and observation. However, we model processes from B2B standard specifications at step B. Some B2B standards have the concept of process, but some do not. If not, we should discuss this with the trading partners to develop a new process based on the B2B standard. In addition, some B2B standards have adopted UML to present their processes in the specification. We can directly refer to those.

**B.1 To design Use Case Diagram**

We develop the use case diagram based on B2B standard specification. A B2B standard specification often describes the process purpose or the process definition by the writing. We discover and extract the basic components of a use case from a process description.

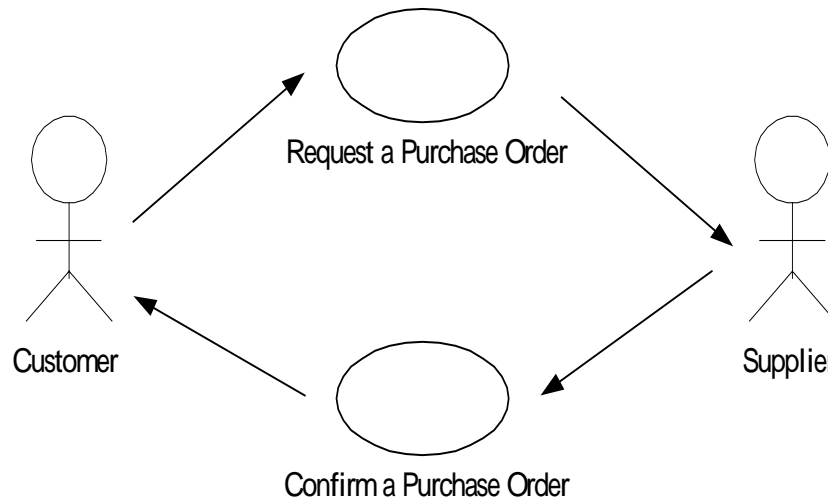


Figure 3-6: A Use Case Diagram Example

**B.2 To design Sequence Diagram**

The B2B standards should specify the sequence of exchanged messages. The last standards often adopt UML to present the sequence. Therefore, we directly use the diagram (shown in Figure 3-7) provided by standards. If the standards do not use UML to present but other methods, we still can analyze the sequence of messages.



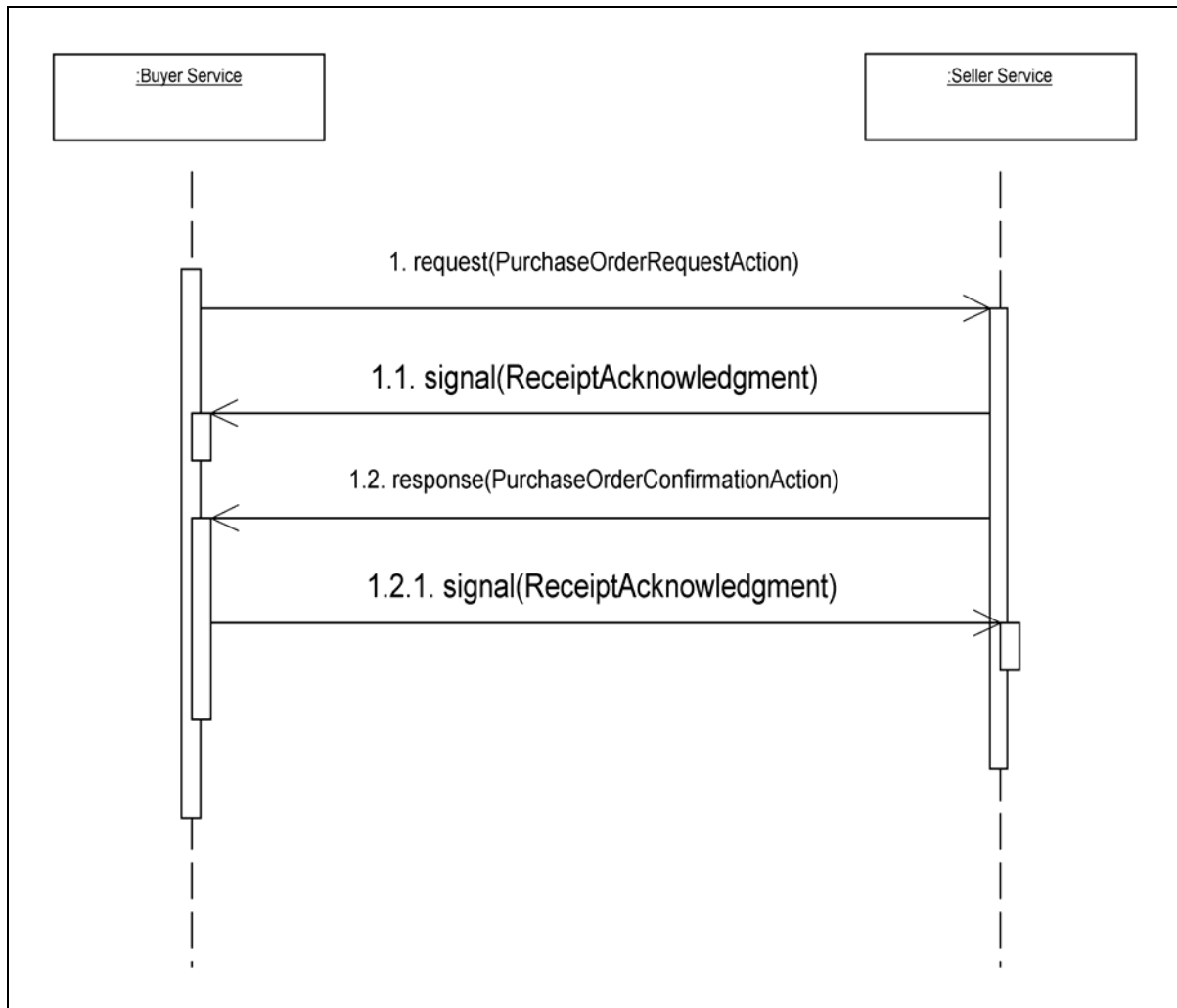


Figure 3-7: A Sequence Diagram to Describe The Purchase Process From Standard

### B.3 To design Activity Diagram

A B2B standard should formalize the public process flow. Such formalization allows partners to follow. We do not expect to manage many different process flows with our trading partners in the real world. A B2B standard provides well-defined process flows (shown in Figure 3-8). We can discover the defined process flow from B2B standard specification or else we can discuss it with the trading partners to develop the new process flow.

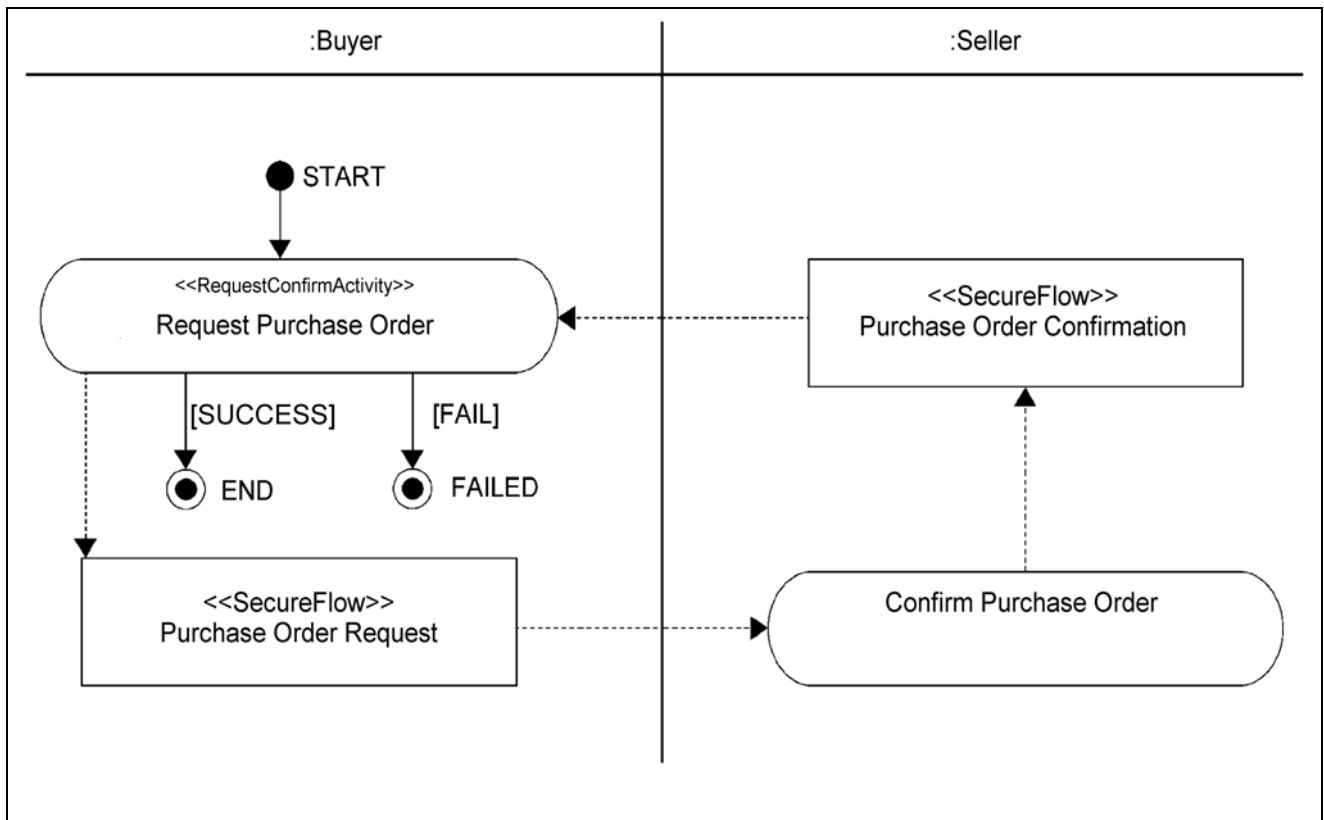


Figure 3-8: An Activity Diagram to Describe The Purchase Process from Standard

### 3.3 Ontology Modeling Heuristics

In this section, we present the heuristics to model ontologies for B2B process and message. The method to build process ontology will be described in C.1 and D.1. The method to build message ontology will be described in C.2 and D.2.

#### C. To capture current B2B ontologies

We build the ontology to describe the B2B domain knowledge. This ontology contains the basic classes and properties. Every business process should fit in with the ontology definition. We have one structure to describe all kinds of process. In order to fulfill this requirement, we discover the basic B2B components and properties.

#### C.1 To design current business process ontology

We depict the B2B interaction as Figure 3-9 from the UML analysis in 3.B and the literature review in section 2.1. To design the process ontology, we should understand the components of B2B process. We can easily to discover from Figure

3-9.

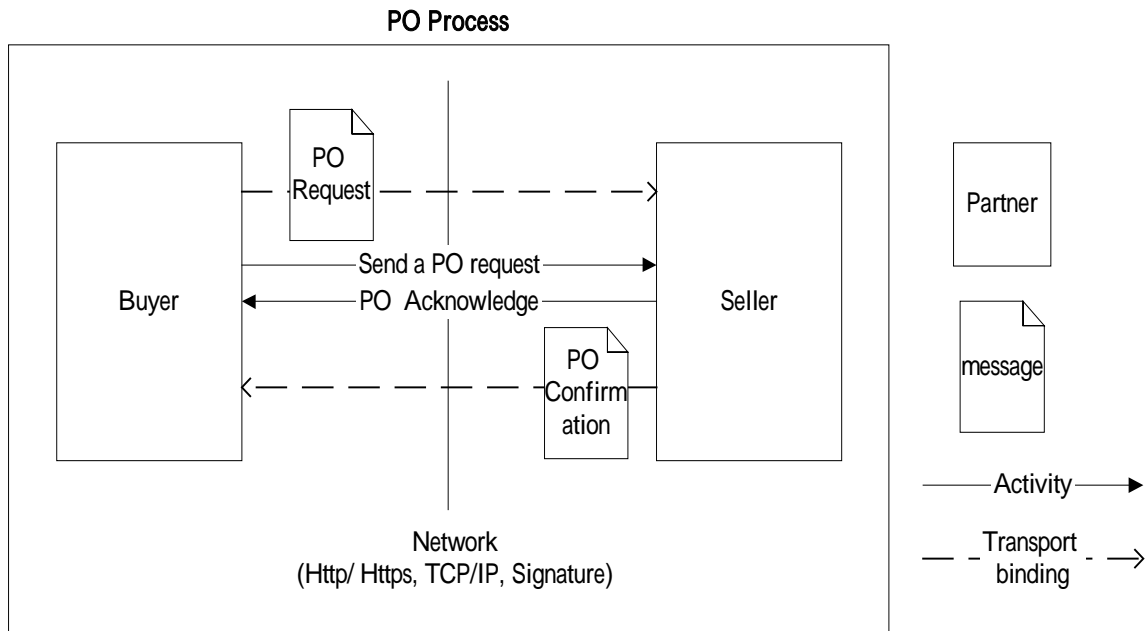


Figure 3-9: The B2B Interaction (This Research)

**C.1.1 To design basic B2B components as basic OWL classes**

The idea is to identify the basic components in B2B domain and how these components can be modeled into the OWL classes.

A B2B interaction describes how a business process is performed between companies. Therefore a business process is the important component when we mention about the B2B. We need a basic class to record a process’s name and definition. The information exchanged in a business process we call business document. A business document contains the message that has pre-defined format. Without the message, the business process has no meaning. We need a class to indicate a business document. The document can be sent or received by several participators. They involve in business process to make process running. These participators may are the enterprises or the users. Therefore we need a class to indicate who are the participators. Besides, a business process may be very complex or simple. The complexity of process flow depends on the business rules that negotiated by participators. The detail activities in business process will describe how business documents can be exchanged. We need a class to describe these activities. We also need a class to describe the technical definitions. It will indicate the technical transport mechanism. Based on the research described in Christoph, (2001), the EC standard should have the following constructs: *process definition, trading partner, message definition and syntax, exchange sequence definition, semantics, security,*

*transport binding*. We also base on the Figure 3-9 and other researches in Section 2.1 to design the following classes:

Process definition: The class `<owl:Class rdf:ID="B2B_Process"/>` is designed to describe business process. The statement, "`<owl:Class rdf:ID="B2B_Process"/>`", in italic type is the OWL language.

Trading partner: The class `<owl:Class rdf:ID="B2B_Partner"/>` is designed to describe partner.

Message definition and syntax: The basic class `<owl:Class rdf:ID="B2B_MessageDocument"/>` is designed. We do not discuss the message's syntax part. Because the syntax has been defined by XML Schema, DTD or other ways. We need not redefine the syntax part. However we define the semantic of business message in detail at next section.

Exchange sequence definition: It describes how business documents can be exchanged, we design the class `<owl:Class rdf:ID="B2B_Activity"/>`.

Semantic: We do not design any special class for this component, because the purpose is designing a semantic B2B ontology.

Security: We can design the class `<owl:Class rdf:ID="B2B_Security"/>`. It describes the EC standard's security mechanism.

Transport Binding: We can design the class `<owl:Class rdf:ID="B2B_TransportBinding"/>`. It describes the EC standard's network transport mechanism.

Security and transport binding are technical definitions. We propose to design another new and more integrated class `<owl:Class rdf:ID="B2B_ActionControl"/>` for action control to replace `<owl:Class rdf:ID="B2B_Security"/>` and `<owl:Class rdf:ID="B2B_TransportBinding"/>`. Properties of security and transport binding will become the properties of action control.

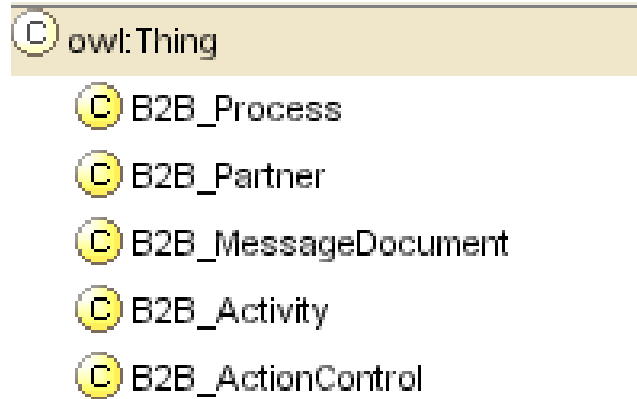


Figure 3-10: The B2B Basic Classes (This Research)

### C.1.2. To design basic B2B properties as OWL object property

After the classes are defined, we analyze the properties. We create these properties into OWL object properties.

Process: Basically, a process has these properties: process name, process definition or purpose, process description, process start state, process end state. In addition, a process also has a document to exchange, a partner role to play, an activity to conduct, and an action control to install. We design the properties as follows:

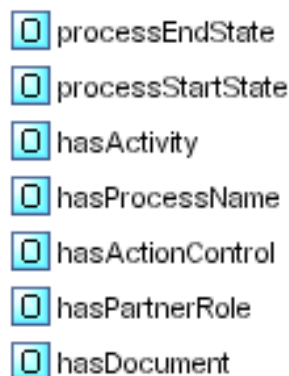


Figure 3-11: The Properties Of B2B Process (This Research)

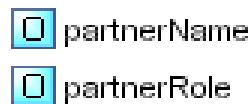
In OWL language:

```

<owl:ObjectProperty rdf:ID="hasDocument">
<owl:ObjectProperty rdf:ID="hasPartnerRole">
<owl:ObjectProperty rdf:ID="hasActivity">
  
```

```
<owl:ObjectProperty rdf:ID="hasActionControl">  
<owl:ObjectProperty rdf:ID="hasProcessName">  
<owl:ObjectProperty rdf:ID="processStartState">  
<owl:ObjectProperty rdf:ID="processEndState">
```

Partner: Partner contains these properties: partner name, partner role in this process, and the role description.



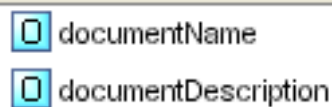
```
Q partnerName  
Q partnerRole
```

Figure 3-12: The Properties Of B2B Partner (This Research)

In OWL language:

```
<owl:ObjectProperty rdf:ID="partnerName">  
<owl:ObjectProperty rdf:ID="partnerRole">
```

MessageDocument: Message contains two properties, document name and document description.



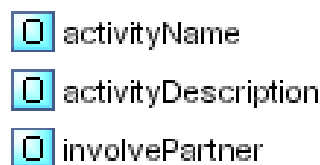
```
Q documentName  
Q documentDescription
```

Figure 3-13: The Properties Of B2B Document (This Research)

In OWL language:

```
<owl:ObjectProperty rdf:ID="documentName">  
<owl:ObjectProperty rdf:ID="documentDescription">
```

Activity: Activity contains properties: name, description, and which partner acts in this activity.



```
Q activityName  
Q activityDescription  
Q involvePartner
```

Figure 3-14: The Properties Of B2B Activity (This Research)

In OWL language:

```
<owl:ObjectProperty rdf:ID="activityName">
<owl:ObjectProperty rdf:ID="activityDescription">
<owl:ObjectProperty rdf:ID="involvePartner">
```

ActionControl: When exchanging a business document, we consider its transport protocol, the performing time, the retry counts if error occurs, and the security requirements.

- actionName
- actionDescription
- byProtocol
- actionTimeToPerform
- actionRetryCounts
- isSSLRequired
- isAuthorizationRequired
- isNonRepudiationRequired

Figure 3-15: The Properties Of B2B Action (This Research)

In OWL language:

```
<owl:ObjectProperty rdf:ID="actionName">
<owl:ObjectProperty rdf:ID="actionDescription">
<owl:ObjectProperty rdf:ID="byProtocol">
<owl:ObjectProperty rdf:ID="actionTimeToPerform">
<owl:ObjectProperty rdf:ID="actionRetryCounts">
<owl:ObjectProperty rdf:ID="isSSLRequired">
<owl:ObjectProperty rdf:ID="isAuthorizationRequired">
<owl:ObjectProperty rdf:ID="isNonRepudiationRequired">
```

### C.1.3 If the value of a property has been specified; use its value as the restriction.

We have analyzed the current business process in 3.A. We use the information from the analysis to restrict the property. For example, we can find that “PO Request” is one of current business documents in PO process from Figure 3-5. We give the value to build the following class.

```

<owl:Class rdf:ID=" PORequest">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#B2B_ MessageDocument" />
  </rdfs:subClassOf>
  <rdf:comment>A request to accept a purchase order for fulfillment.</rdf:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasDocumentName" />
      </owl:onProperty>
      <owl:hasValue> PORequest </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID=" POConfirmation">

```

**C.1.4 If you find other needed and specialized relationships or properties, you can add and create its corresponding OWL properties.**

You can create additional properties to describe current business process.

**C.1.5 If you find individuals for a class, you can create a corresponding instance in OWL.**

**C.2 To model current business document ontology**

We analyze the processes among partners. We focus on the core of B2B process analysis, that is, message centric analysis. We develop the process ontology based on the semantics of messages. The semantics refer to the context, the meaning, the terminology, and the relationship of messages.

**C.2.1 To discover the basic data entities**

The basic data entity is the field that contains the real data in the message. Usually, this kind of entity is defined as #PCDATA in DTD. Such a basic data entity can be defined as a generic OWL class. If the specification has given each data entity's metadata, such as entity's data type, entity's description, entity data's representation. The metadata can be defined as a property of OWL class.



```

- <PO>
- <Header>
  <PONumber>po100001</PONumber>
  <Version>0</Version>
</Header>
+ <Item>
+ <Item>
</PO>

```

← Basic Data Entities

Field Name	Description	Data Type
PONumber	The unique number to identity a purchase number	String
Version	The version of purchase order	Integer

Figure 3-16: An Example of Basic Data Entities (This Research)

For example, in this Figure 3-16, the tag <PONumber> and <Version> are the basic data entities. We create the class <owl:Class rdf:ID="PONumber"/>. We may find there is metadata given to describe this field from database schema. We can design the corresponding properties according the metadata. From this case, they are the data type and the field description. We add two properties, “hasDescription” and “hasDataType”, to complete the class “PONumber”, as follows:

```

<owl:Class rdf:ID="PONumber">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasDescription"/>
    <owl:hasValue>
      <xsd:String rdf:value=" The unique number to identity a purchase number"/>
    </owl:hasValue>
  </owl:Restriction>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasDataType"/>
    <owl:hasValue><xsd:String rdf:value="Integer" /></owl:hasValue>
  </owl:Restriction>
</owl:Class>

```

### C.2.2 To discover the composite data entities

A composite data entity is composed of two or more basic data entities. It groups a set of related entities based on the message’s XML schema or DTD. For the composite data entities, you can create an OWL class for the composite data entity, and group its basic data entities through the <owl:onProperty> link.

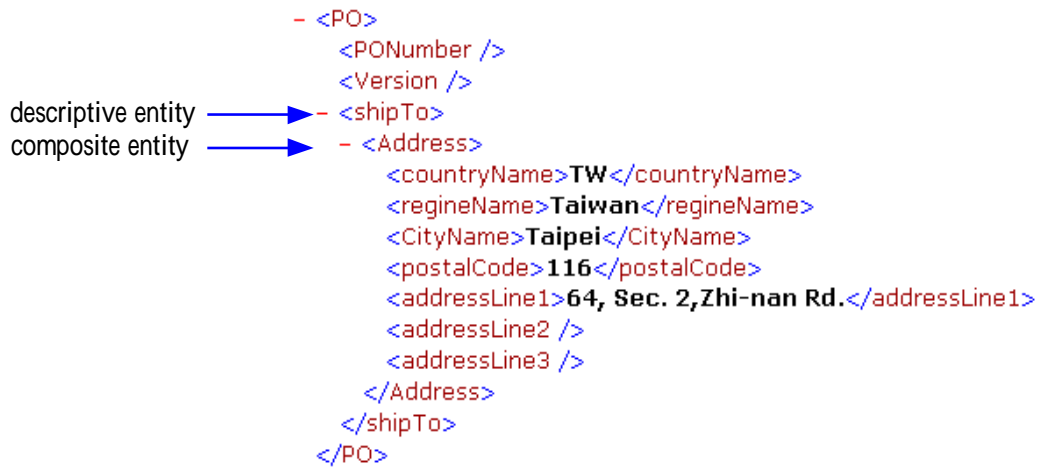


Figure 3-17: An Example of Descriptive and Composite Entities (This Research)

For example, we have a composite entity, the tag <Address>, in this Figure 3-17. We create a class named “Address” and build its all properties. They are “hasCountryName”, “hasRegineName”, “hasCityName”, “hasPostalCode”, and “hasAddressLine1”. We group the class and properties as follows:

```

<owl:class rdf:ID="Address">
  <rdf:comment>The collection of business properties that provide address information for
contacting a person, organization or business.</rdf:comment>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasCountryName"/>
  </owl:Restriction>
  <owl:Restriction>
  </owl:Restriction>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasRegineName"/>
  </owl:Restriction>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasCityName"/>
  </owl:Restriction>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasPostalCode"/>
  </owl:Restriction>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasAddressLine1"/>

```

```
</owl:Restriction>
</owl:class>
```

### C.2.3 To discover the descriptive entities in the message

Sometimes, there is a descriptive entity that does not contain data. It just describes a simple business meaning of the data entity. We do not treat it as a class. We take it as a property that named business property. Therefore we use an OWL object property to model it.

There is a descriptive entity, the tag <shipTo>, from the Figure 3-13. We design an object property named “shipTo” to represent the descriptive entity and give its domain and range value.

```
<owl:ObjectProperty rdf:ID="shipTo">
  <rdfs:domain rdf:resource="#PO" />
  <rdfs:range  rdf:resource="#Address" />
</owl:ObjectProperty>
```

### C.2.4 To build the business document ontology

Using the business document name that is defined in the 3.A, we can create an OWL class. Then, we find all elements under the root element based on DTD or XML Schema. These elements will be the properties of this class. Next, we need to make sure all the entities of business document are created and their relations have been connected.

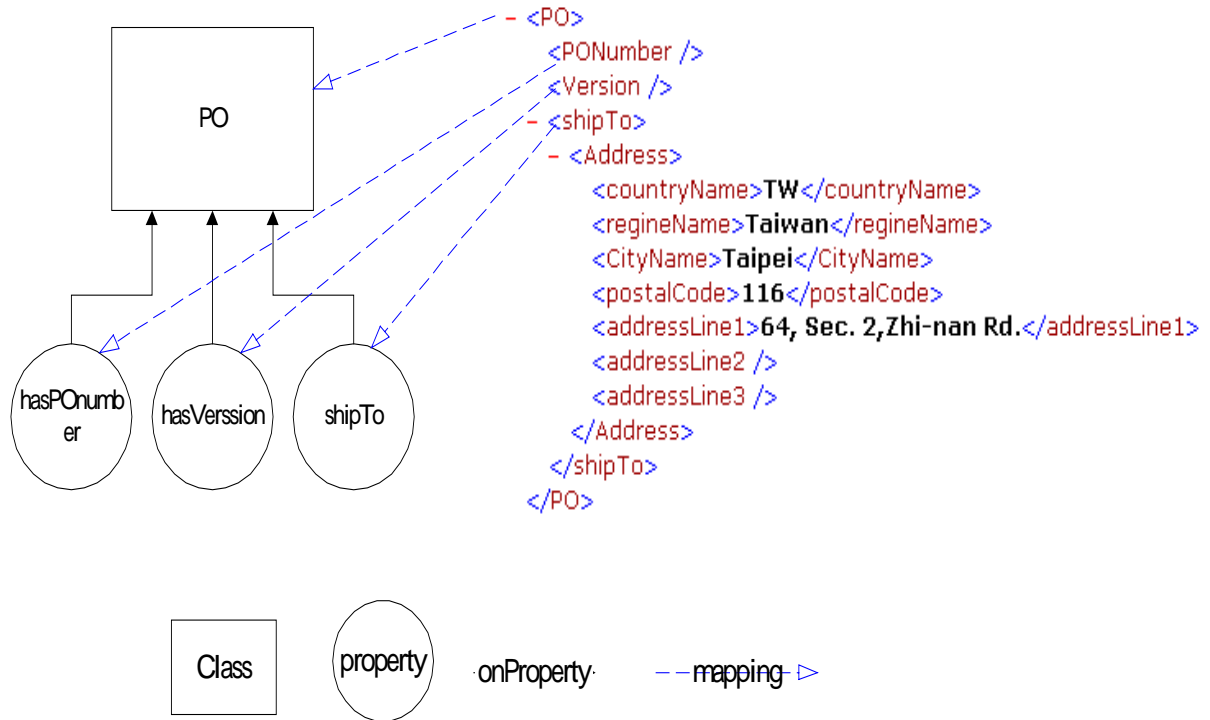


Figure 3-18: To Build The Business Document Ontology (This Research)

### C.3 To reconcile current business constraints

We may have constraints on each entity, each message, and each process. We design these constraints into OWL.

After designing current business process and document ontology, we create the EC standard ontology. The most steps in 3.C and 3.D are alike. We only specify the different in 3.D.

### D. To capture EC standard's ontologies

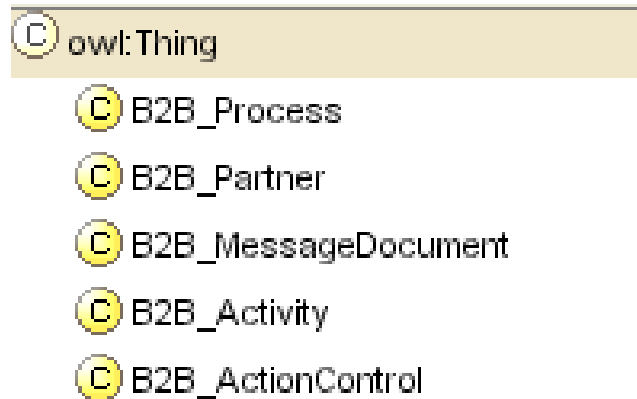
We have built the UML model of B2B standard at 3.B. The next step is building the B2B standard ontology.

For building the EC standard's ontologies, we need the B2B standard specifications and the definition of business document. The definition of business document is often encoded as DTD or XML Schema. We use the definition of document to build message ontology.

#### D.1 To design EC standard's process ontology

##### D.1.1 To design basic EC components as basic OWL classes

We also design five basic classes that are the same in C.1.1.



Notice that not all EC standards implement these components. Only the description and discussion above are designed as the basic owl classes. The first set of heuristics is mainly one-to-one class creation because this is the first step and starts from the basics. If the EC standard specifies other special components that we do not mention above, you can design your own class.

#### **D.1.2. To design basic properties in EC standard into OWL object property**

We have designed the basic properties in C.1.2, but if we find more properties in the EC standard specification. We create these properties into OWL object properties.

#### **D.1.3 If the value of a property has been specified in the standard, use its value as the restriction.**

The following is an example showing a table that lists the business document information in the purchase order process. There are two business documents in this table. They are “Purchase Order Request” and “Purchase Order Confirmation”. There is a description to correspond with each document.

Table 3-2: An Example of Business Document Information Provided by Standard

Business Documents	
Business Document	Description
Purchase Order Request	A request to accept a purchase order for fulfillment.
Purchase Order Confirmation	Formally confirms the status of line item(s) in a Purchase Order. A Purchase Order line item may have one of the following states: accepted, rejected, or pending.

We extract this information to form the following class in OWL.

```

<owl:Class rdf:ID="BusinessDocument">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#PurchaseOrderRequest">
      <rdf:comment>A request to accept a purchase order for fulfillment.</rdf:comment>
    </owl:Thing>
    <owl:Thing rdf:about="#PurchaseOrderConfirmation">
      <rdf:comment>Formally confirms the status of line item(s) in a Purchase Order.
    </owl:Thing>
  </owl:oneOf>
</owl:Class>

```

**D.1.4 If you find other needed and specialized relationships or properties, you can add and create its corresponding OWL properties.**

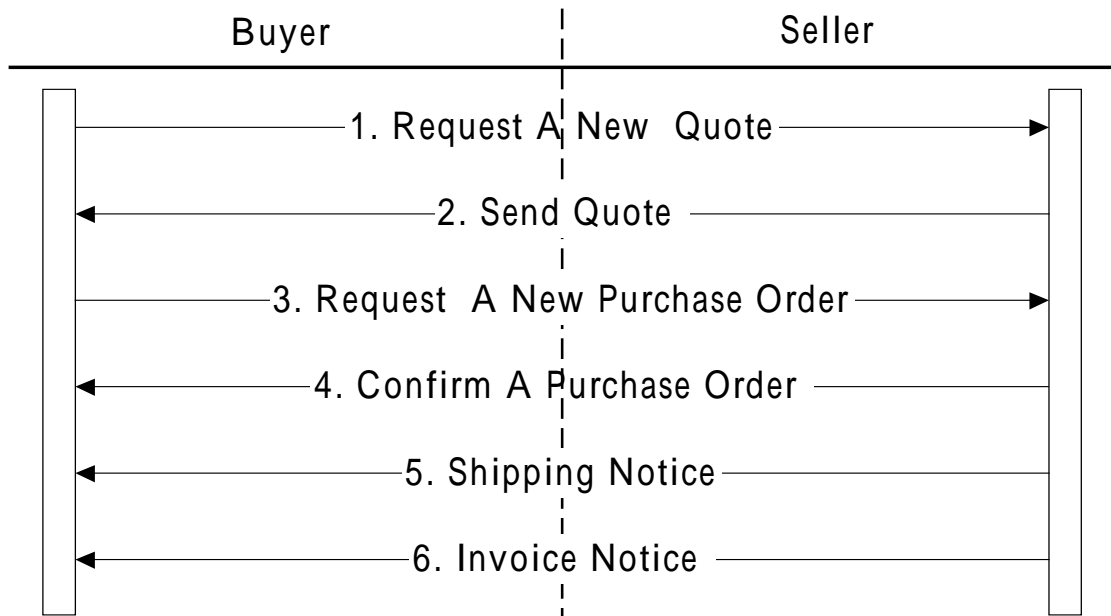


Figure 3-19: An Example of The Sequence of Processes (This Research)

If EC standard defines the sequence of processes, we can know the order of processes (shown in Figure 3-19). For example, the “Request A New Purchase Order” process must be executed after “Request A New Quota” and before “Shipping Notice”. Thus we have a special relationship between processes. We design these relationships into the properties "preProcessName" and "followedProcessName" for the class “process”.

```

<owl:Class rdf:ID="Process">
<owl:ObjectProperty rdf:ID="preProcessName">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="#Process"/>
  <rdfs:range rdf:resource="#Process"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="followedProcessName">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="#Process"/>
  <rdfs:range rdf:resource="#Process"/>
  <owl:inverseOf rdf:resource="#preProcessName"/>
</owl:ObjectProperty>

```

</owl:Class>

**D.1.5 If you find individuals for a class in the specification, you can create a corresponding instance in OWL.**

### **D.2 To model EC standard's document ontology per partner**

The method to model EC standard's document ontology is the same with C.2. We can find the required data definition in EC specification.

#### **D.2.1 To discover the basic data entities**

#### **D.2.2 To discover the composite data entities**

#### **D.2.3 To discover the descriptive entities in the message**

#### **D.2.4 To build the business document ontology**

### **D.3 To reconcile EC standard's constraints**

A B2B standard may have constraints on each entity, each message, and each process. In addition, the trading partners may define their own special constraints on the process ontology. We design these constraints into OWL.

## **3.5 The Merge, Representation and Testing of Ontologies**

When we initiate and implement a new B2B initiative each time, we are dealing with different B2B standards, different business partners and different situations. However, we have the existing ontology in our ontology repository. These differences cause the ontology mismatch. The mismatch may not keep ontology consistency. In this section, we discuss the mergence of ontology in order to get a better ontology that is more suitable the real environment.

The ontologies can be represented through several tools and ways. However, in order to merge the ontologies, the facility to display the differentia between ontologies is a key function that we need to consider. The most important functions of such tools are considered as follows (Klein et al., 2002):

- Read in ontologies, ontology updates, adaptations or mappings
- View a specific version or variant of an ontology
- Differentiate ontologies:
  - Show changed formal definitions



- Show changed comments
- Show type of change: conceptualization or explication
- Automatic inconsistency checks of ontology combinations

**E. To merge ontologies**

We develop two process ontologies in 3.C and 3.D. The two ontologies are with the same domain. The mergence of ontology is a critical issue. The ontology should remain in consistent state after merging. The keys of merging are discovering the differences and developing the corresponding rules between the ontologies. The differences between ontologies include: the changes of class name, the addition or deletion of classes, the addition or deletion of properties, and the mergence or split of classes. The ontology tool should provide the function to display the differences between two ontologies. It highlights the differences such that we can adjust these differences to merge our ontology.

We want to merge two ontologies into a final ontology. It is normal to find conflicts and differences between the two. At the same time, there are the similar parts between the two, too. The more differences between two ontologies, the more and larger extent of change between the old and new processes. This analysis helps us to know which parts of the process will be changed and different at the project implementation phase. Then we can tune the changed parts of the process.

How do we reconcile the conflicting parts? First, we adopt the ontology of the process proposed by B2B standard as our base ontology. Because the process proposed by B2B standard is the process we what to be. We tune the B2B standard ontology directly according the correspondence rules (shown in Table 3-6) that we develop as follows:

Table 3-3: The Correspondence Rules Of Merging Ontology (This Research)

Level	Conflict Type	Current (Old)	Standard (New)	Condition Description	Rules
Class Level	Schematic conflicts	None	New	Standard has a new class, which doest not exist in current process.	We keep the new class in the ontology. All the properties of the new class should be retained, too.

		Existed	None	The current process exist an old class, which does not appear in standard process.	If the old class will no longer exist in the future, we discard them. Else we should add the old class from the old ontology to the new ontology.
	Semantic conflicts	Existed Class A	New Class B	They are with the different class names but the same meaning	We reserve the old class A and add it to new ontology. Then, we use the <i>owl:sameAs</i> to state the two classes are equivalent. However, we use the class B usually.
		Existed Class A	New Class A	They are with the same class name but different meanings.	We keep the name of the new class. However we change the name of old class to another new name.
Property Level	Schematic conflicts	None	New	There are additional properties in a class.	We use and adopt these properties in the new ontology.
		Existed	None	There are deletion properties in a class.	We have to determine whether the properties are no longer useful. If we do not use these properties any more, we discard them. If we still need these properties, we should reserve them and add them to new class.  We adjust the minimum cardinality of these old properties to 0. Because, they are not necessary properties in the new class.
	Semantic conflicts	Existed Property A	New Property B	They are with the different property names but the same meaning	We reserve the old property A and add it to new ontology. Then, we use the <i>owl:equivalentProperty</i> to state the two properties are equivalent. We use the property B usually.
		Existed Property A	New Property A	They are with the same property name but different meanings.	We keep the name of the new property. However we change the name of old property to another new name.

### **F. To represent ontologies**

We have discussed the functions of ontology representation at the beginning in 3.5. The representation of ontologies should clear and easy to understand.

### **G. To test ontologies**

To verify the ontologies merged in 3.E, we consider two aspects, the syntactic and semantic. We test the syntactic of ontology through the tool. The tool can present the ontology and validate the inconsistency of syntax.

Next, we consider another respect, semantic. The newly designed ontology may not be consistent with the real environment. The inconsistency exists between the database schema, the real business processes and the old version of ontology. We can ask the process owners to verify the new ontology. We extract the database schema or the E-R diagram to compare the consistency between the business message ontology. We can also compare the consistency through the trading partner agreements that specify the business rules between companies. We record the differences between the new ontology and the real environment. This information helps us to adjust our business process and refine the next version of ontology.

We can use many internal or external documents in the company to verify the developed ontology. If we find a new concept from the ontology and the new concept does not exist in the original environment, we should consider adopting the new concept into our process. We can consider a possibility of business process reengineering through the validation of ontology.

## **3.6 Discussion**

We have described the steps from 3.A to 3.G above. These steps help us to develop the ontology of B2B domain. We review these steps and correspond to Figure 3-1 to explain the whole framework.

In 3.A, we analyze the current business process by the UML approach. In 3.B, we analyze and develop the future business process by the UML approach, according to the process flow recommended by EC standards or the process flow expected from trading partner.

In 3.C, we provide a heuristics-based method to model the ontology of a business process. First, we create the current business process ontology in 3.C.1. Then, we model the current business document ontology in 3.C.2. Next, we reconcile the B2B standard's constraint.

The Step 3.D is most the same with 3.C. The different is Step 3.D model the B2B standard's process and document.

Now, we have two process ontologies that are outputted from 3.C and 3.D. We merge these ontologies in 3.E. After merging the ontologies, we select a tool to represent the ontologies in 3.F. Next, we test and verify the ontologies in 3.G.

Finally, we list these steps as follows:

- A To analyze current business process, "as-is"
- B To develop EC-standard-compliant business process"
- C Ontology modeling - heuristics based
  - C.1 To create current business process ontology
  - C.2 To model current business document ontology
  - C.3 To reconcile with current business constraints
- D Ontology modeling - heuristics based
  - D.1 To create B2B standard's process ontology
  - D.2 To model B2B standard's message ontology per partner
  - D.3 To reconcile with B2B standard's constraints
- E To merge ontologies
- F To represent ontologies
- G To test ontologies

The concept of ontology evolution process is proposed by Stojanovic *et al.* (2002). It has six phases of evolution (Figure 3-20). The process captures the changes and presents the changes to user. Then, it deals with the changes to form a new ontology and verify the new ontology at last.

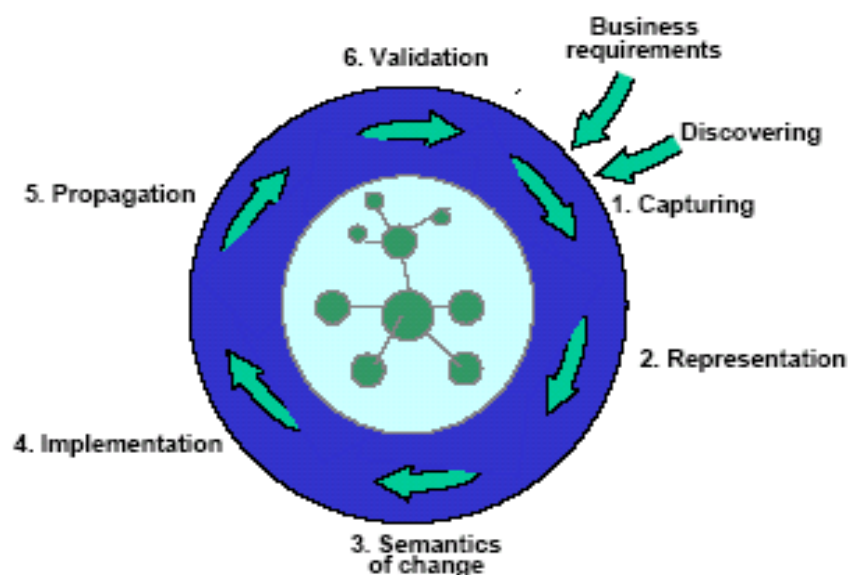


Figure 3-20: Ontology Evolution Process (Stojanovic et al., 2002)

For our research method, we store the ontologies in a repository. However, when each new B2B project is initiated, we build a new ontology. The existing ontologies in a repository should evolve into the new ontologies. We present another ontological evolution process in Figure 3-21 according to our methods. We discover the ontology requirements from method 3.A and 3.B. Then, we capture the ontology from 3.C and 3.D. We provide the method to merge ontologies in 3.E. We represent the ontologies in 3.F and verify it in 3.G.

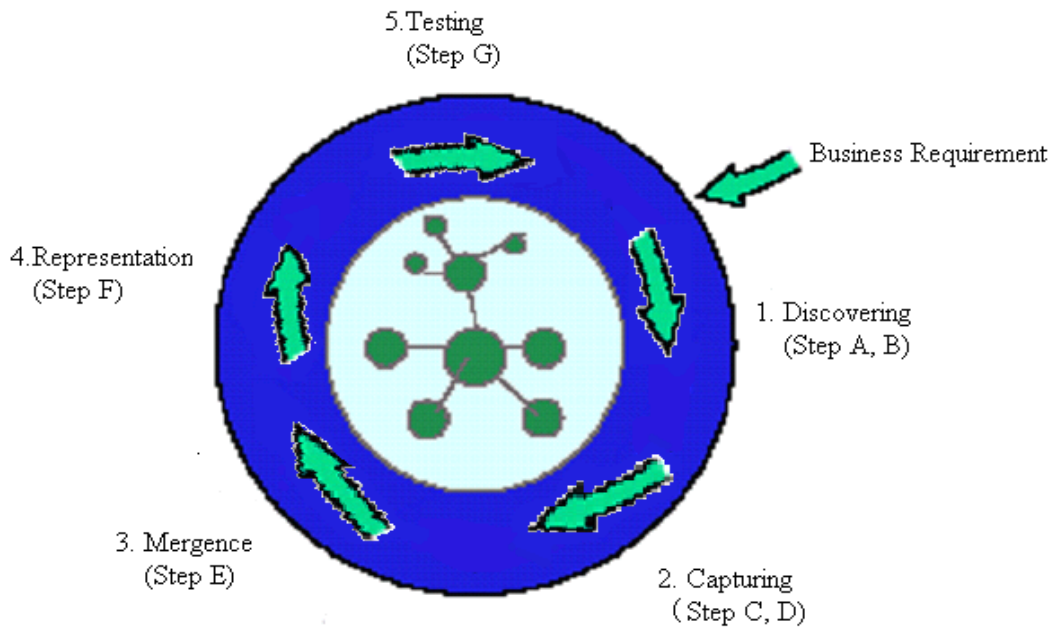


Figure 3-21: A B2B Integration Ontology Evolvment Cycle (This Research)