Chapter 3

Enterprise Modeling in CBSD

Enterprise modeling is the process of building models of whole or part of an enterprise (e.g. process models, data models, resource models, new ontologies etc.) from knowledge about the enterprise, previous models, and/or reference models as well as domain ontologies and model representing languages [77]. This process is one of the key activities in business and software related projects such as business process engineering/reengineering, information system development and so forth. An enterprise model which is the result of enterprise modeling, can be defined as one representation of a perception of an enterprise, and might be made of several submodels including process models, data models, resource models and organization models. Those submodels correspond to *viewpoints* in knowledge of an enterprise.

Each modeling methodology has its own viewpoints on enterprise. For example, CIMOSA has those of "organization", "resource", "information" and "function". On the other hand, ARIS has those of "organization", "data", "control" and "function". Other methodologies also have the similar viewpoints.

Those methodologies provide us with systematic ways to construct enterprise models or submodels from their viewpoints. However, we often are face with a difficulty at the early stage in enterprise modeling when applying those methodologies to real applications.

The difficulty is that we have no systematic way to identify the elements or the components of enterprise models, which compose the enterprise models. The above methodologies regard identifying such elements as a part of *requirements engineering* [72], and presume the elements to be given. Although *requirements engineering* provide us with many methods to identify the elements, they are not formal enough to use them in various situations. In addition, since requirements engineering does not provide us with appropriate ways to coordinate or integrate pieces of knowledge which domain-experts have, we have to make considerable efforts to identify consistent model elements.

In this chapter, some formal ways are discussed, which are useful to identify the above model elements and to construct enterprise models by them. Since this thesis often uses set theory as one of the formal techniques, we hereafter use the term "model units" instead of "model elements", in order to avoid confusing elements in models with those in set theory.

The chapter deals with two topics. One is a method to identify model units from many pieces of knowledge on an enterprise that various domain-experts have. Several types of model units in an enterprise are defined to express functionality and behavior of it. Rough Set Theory (RST) is used to integrate many pieces of knowledge for identifying model units of those types.

The other is a procedure to construct enterprise models using the identified model units. Colored Petri Nets (CPN) are used as a single notational method to express an enterprise.

3.1 Identifying Basic Model Units and Their Relationships

It is the first step of most enterprise modeling methodologies to define model units which would be forming the final models to be built up. In other words, this step defines what composes the enterprise at the abstraction level to be considered. For example, *entities*, *attributes*, and *relationships* in Entity-Relationship (ER) models, *activities* and *control flows* in process models, or *objects*, *methods* and *relationships* in Object Oriented (OO) models are those model units. They are usually identified through such informal human activities as sessions, consultations, hearing, observation and so on, involving relevant domain-experts, which are proposed by *requirements engineering*.

Those informal activities produce many pieces of knowledge on the enterprise, usually are provided by domain-experts, which are biased by their missions and roles. In addition, those pieces could mutually conflict from each other. Therefore, we need some methods to coordinate those *biased* pieces of knowledge into *neutral* ones to build up the consistent enterprise models. Several researches focus on multiple viewpoints or aspects of requirements [32], [73], however, no systematic ways have not been proposed yet, which can integrate those viewpoints in order to identify such *neutral* or *non-biased* model units for enterprise modeling.

Each methodology, which deals with enterprise modeling, has a unique way to express complex enterprise structure, and has its own modeling frameworks along with model units. Therefore, there could be different sets of model units according to the methodology we use.

However, those methodologies regard business processes as one of the most

important viewpoints for understanding enterprises [1], [24], [66], [68], [77], and it is possible to define common model units among the methodologies for expressing business processes. Those units will be interpreted by each methodology to convert them into its unique forms and could be the core units of its modeling framework, since business processes are the commonly understandable core among the methodologies.

There are several definition of a business process, such as, "a set of logically related tasks performed to achieve a defined outcome" [18] or "a structured, measured set of activities designed to produce a specified output for a particular customer or market. It implies a strong emphasis on how work is done within an organization" [19].

There are several ways to define the types of model units in order to express a business process. For example, we can regard *organization*, *data*, *control* and *function* as the model units [62], or *function*, *information*, *resource* and *organization* as them.

In this thesis, we use the following five model units, which will provide sufficient information on a business process.

1. Resources : Resources are externally observable substances which are dealt with by some business operations or information systems. They can be transformed into other ones, be created or be deleted. They could have multiple attributes which characterize each resource by their values.

2. Organization : Organization consists of all the people related to enterprise operations, along with relevant equipment and/or facilities.

3. Tasks : Tasks are externally observable actions which affects the states of enterprises.

4. Functions : Functions are transformation rules among the resources, or the model units of resources in the enterprise.

5. Behavior : Behavior is a set of ordered sequences of tasks which are associated with organization [1]. Each sequence can include such control structure as conditional/unconditional forks, joins or iterations. Those sequences are often referred to as *business processes*.

The first three types are regarded as representing the static aspect of an enterprise, since they have no temporal properties, and are observed at any instants, while the remaining two types represent the dynamic aspect, since they have temporal properties and are observed in particular periods.

[1] Strictly speaking, the model units of tasks and organization

In large-scale enterprise modeling, we have to involve many domain-experts in order to obtain pieces of knowledge on their specialty areas in the enterprise. Those pieces of knowledge could be the sources of the model units for the enterprise models. However, they are usually biased by the roles, missions and backgrounds of the domain-experts. In other words, the pieces of knowledge are provided from the experts' own viewpoints. Therefore, they could conflict or be inconsistent from each other when we try to identify the model units of the above five types.

The model units of the types of *resources*, *organization* and *tasks* can be regarded as *concepts* defined in those types, whereas those of *functions* and *behavior* can be regarded as *rules* among the above concepts. The concepts are the results of generalization or abstraction of the substances in those types. This generalization is usually based on the knowledge of the domain-experts, and therefore could be different from each other among the experts. In order to formalize the generalization and the knowledge behind it mathematically, we define three sets of all the externally observable or recognizable substances which reside in the enterprise as follows.

1. U_1 : A set of all the substances which are recognized as *resources* by external observation. A substance in the enterprise is recognized as a *resource* when it is transformed in any way. The *resources* are the *influencees* of enterprise operations, and include information and data.

2. U_2 : A set of all the people, equipment and facilities which could become the actors or performers in enterprise operations. They are the *influencers* of the enterprise operations, and compose organization.

3. U_3 : A set of all the tasks performed in the enterprise. Tasks are the actions performed in the enterprise, which exert influence upon the *resources*.

Each domain-expert can make groups or classes in the above sets U_1, U_2 and U_3, according to his/her knowledge on the enterprise. Those groups or classes could be the candidates of the model units of *resources*, *organization* and *tasks*.

The other two types of the model units mentioned above, that is, *functions* and *behavior*, are defined as relationships among the groups in U_1, U_2 and U_3. The model units of *functions* are the relationships between the groups in U_1 which are expressed as *transformation rules*, while those of *behavior* are the relationships between the groups in U_3 associated with the groups in U_2 and *functions*. Those relationships also are obtained from the knowledge of the domain-experts.

The model units obtained from each domain-expert are *individual* ones, or in other words, those units compose an enterprise model from his/her viewpoint. Therefore, we have to integrate those units into *enterprise-wide* ones, in order to

construct the consistent enterprise models. In the following section, we discuss how those *individual* model units are integrated into *enterprise-wide* ones.

Figure 3.1 and Figure 3.2 show our approach intuitively.

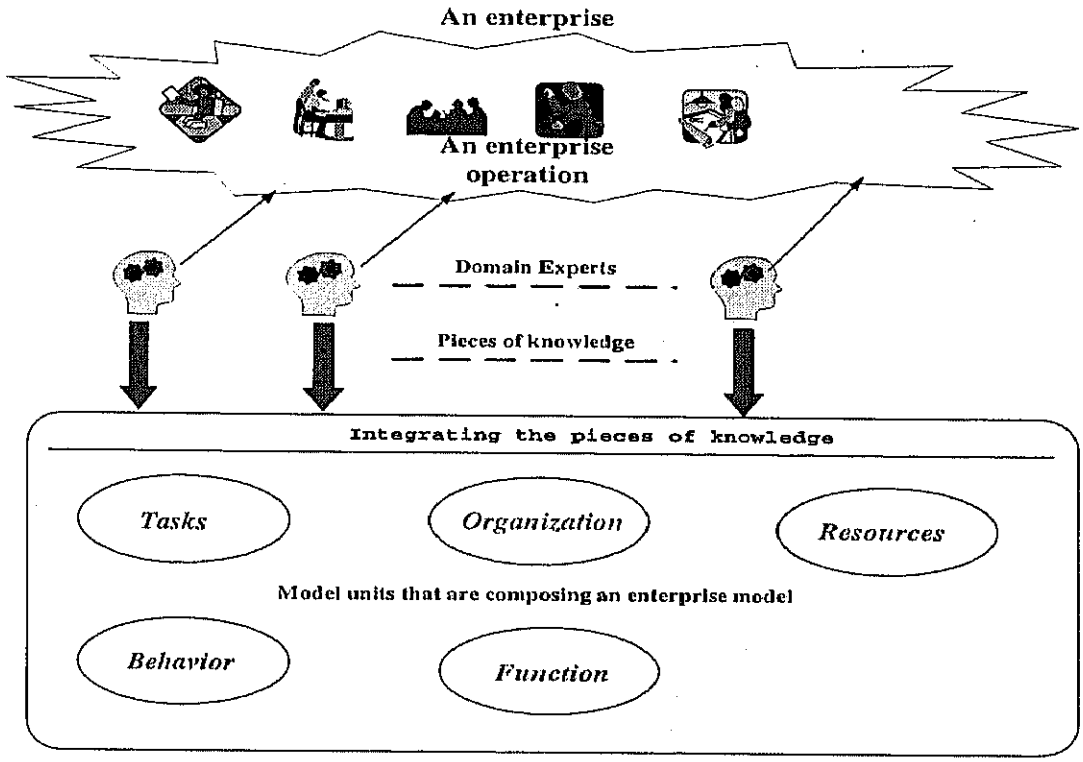

Figure 3.1: Enterprise modeling process - 1

3.1.1 Knowledge Integration with Rough Set Theory

As mentioned above, it is caused by the differences of the knowledge owned by the domain-experts that the model units are different or inconsistent from each other among them. Therefore, we need a way to express and analyze heterogeneity and homogeneity between the knowledge or pieces of the knowledge of the domain-experts. In order to deal with this aspect of knowledge, we use Rough Set Theory (RST) [57], [58].

RST provides us with theoretical aspects of reasoning about data, and deals with various knowledge and concepts based on set theory.

RST regards knowledge as the ability to classify objects, and regards concepts as the classes (or groups) to which the objects belong. A set of objects, called

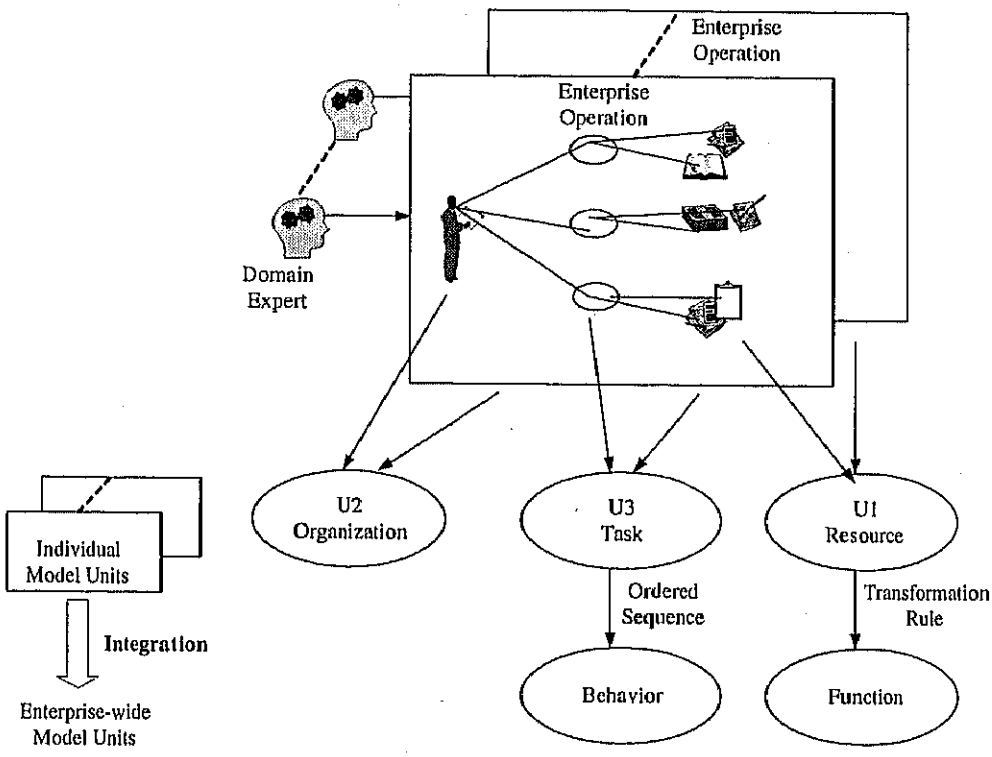

Figure 3.2: Enterprise modeling process - 2

universe U, is classified by knowledge into the classes $X_1, \ldots, X_n$, where $X_i \cap X_j = \emptyset$ and $U = \cup X_i$. Each X_i can be regarded as a concept in U.

This classification corresponds to an equivalence relation[2] over U according to standard set theory. Therefore knowledge in RST can be represented by equivalence relations. A family of equivalence relations and a universe U compose a knowledge base $K = (U, \mathrm{R})$, where $\mathrm{R} = \{R_1, R_2, \ldots, R_n\}$ is a family of equivalence relations over U.

Partial knowledge in K is represented by $\mathrm{P} \subseteq \mathrm{R}$, and the finest classification by this P is obtained by the equivalence relation $IND(\mathrm{P}) = \cap \mathrm{P}$ which is called an indiscernibility relation over P[3].

The class to which $x \in U$ belongs in this finest classification is

$$[x]_{IND(\mathrm{P})} = \bigcap_{R \in \mathrm{P}} [x]_R$$

where $[x]_R$ means the class to which x belongs, when U is classified by the equivalence relation R.

The family of all the equivalence relations definable in K is denoted by

$$IND(K) = \{IND(\mathrm{P}) : \emptyset \neq \mathrm{P} \subseteq \mathrm{R}\}.$$

Classes that are classified by an equivalence relation R are called *R-basic categories*, and we regard the categories as concepts. If $X \subseteq U$ is the union of some *R-basic categories*, X is called *R-definable*, otherwise it is called *R-undefinable*.

By an equivalence relation R derived from a given knowledge base $K = (U, \mathrm{R})$, that is, by $R \in IND(K)$, any subset $X \subseteq U$ is recognized in the following two ways.

$$\underline{R}X = \cup(\{Y \in U/R : Y \subseteq X\}$$
$$\overline{R}X = \cup(\{Y \in U/R : Y \cap X \neq \emptyset\}$$

where U/R means the family of all classes that is classified by R, that is, *R-basic categories*.

The former is called the *R-lower approximation of* X and the latter is called *R-upper approximation of* X. $BN_R(X) = \overline{R}X - \underline{R}X$ is called the *R-boundary of* X.

For example, suppose we are given a set of five cars as the universe U, and we have such three pieces of knowledge on cars as *color*, *size*, and *style*, then those cars are characterized by the tuple of the attributes (*color*, *size*, *style*). By those attributes, we can classify the U, therefore they are pieces of knowledge in RST. If those tuples are $(red, large, sedan)$, $(red, large, sedan)$, $(blue, large, sedan)$, $(blue, small, coupé)$ and $(green, medium, coupé)$ as shown in Table 3.1, then the knowledge base is denoted by $K = (U, \mathbf{R})$, where $U = \{1, 2, 3, 4, 5\}$ and $\mathbf{R} = \{R_1, R_2, R_3\}$. R_1, R_2 and R_3 are the equivalence relation which correspond

[2] A relation $R \subseteq U \times U$ which is reflexive, symmetric and transitive.

[3] $IND(\mathrm{P})$ is an equivalence relation, since intersection of several equivalence relations is also an equivalence relation.

to the attributes *color*, *size* and *style* respectively. By those equivalence relations, U is classified in three ways as follows.

$U/R_1 = \{\{1,2\},\{3,4\},\{5\}\}$
$U/R_2 = \{\{1,2,3\},\{4\},\{5\}\}$
$U/R_3 = \{\{1,2,3\},\{4,5\}\}$

This knowledge base $K = (U, \mathbf{R})$ can also be denoted by the following table, which is called *Knowledge Representation System* (KRS) with the attributes *color*, *size* and *style*.

Table 3.1: Knowledge representation system

U	*color*	*size*	*style*
1	red	large	sedan
2	red	large	sedan
3	blue	large	sedan
4	blue	small	coupé
5	green	medium	coupé

We can regard such KRS as a decision table by dividing the attributes into *conditions* and *decisions*. For example, by regarding the attribute *color* and *size* as *condition*, while *style* as *decision*, we can interpret the above KRS as a set of decision rules, such as "*if a car is red and large, then it is a sedan*".

The generic form of decision tables is shown in Table 3.2.

Table 3.2: Decision table

U	C_1	$\cdots$	C_p	D_1	$\cdots$	D_q
1	c_{11}	$\cdots$	c_{1p}	d_{11}	$\cdots$	d_{1q}
2	c_{21}	$\cdots$	c_{2p}	d_{21}	$\cdots$	d_{2q}
$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
x	c_{x1}	$\cdots$	c_{xp}	d_{x1}	$\cdots$	d_{xq}
$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$

In this table, the set of attributes $\{C_1, \ldots, C_p\}$ represents conditions, and the set of the attributes $\{D_1, \ldots, D_q\}$ represents decisions.

A decision table is said to be *consistent* if :

$\forall x, y \in U \ (x \neq y)$

$[(c_{x1}, \ldots, c_{x_p}) = (c_{y1}, \ldots, c_{yp}) \Longrightarrow (d_{x1}, \ldots, d_{xq}) = (d_{y1}, \ldots, y_{yq})] \cdots (1)$

In other words, we can derive a unique set of decisions by a set of conditions from a *consistent* decision table.

By regarding condition $\mathrm{C} = \{C_1, \ldots, C_p\}$ and decision $\mathrm{D} = \{D_1, \ldots, D_q\}$ as the individual knowledge, we can classify the universe U in two ways, that is, $U/IND(\mathrm{C})$ and $U/IND(\mathrm{D})$, where IND(**C**) and IND(**D**) are the equivalence relations called indiscernibility relations, which are defined as :

$IND(\mathrm{C}) = \{(x, y)|(c_{x1}, \ldots, c_{x_p}) = (c_{y1}, \ldots, c_{y_p})\}$

$IND(\mathrm{D}) = \{(x, y)|(d_{x1}, \ldots, d_{x_q}) = (d_{y1}, \ldots, d_{y_q})\}$

where $(x, y) \in U \times U$.

The above formula (1) is equivalent to the following formula (2).

$IND(\mathrm{C}) \subseteq IND(\mathrm{D}) \cdots (2)$

An attribute C_i is called **D**-*dispensable* in C, if the new condition C', which is created by removing C_i from C, satisfies

$IND(\mathrm{C}') \subseteq IND(\mathrm{D}) \cdots (3)$.

In such case, we obtain the same result from the decision table with the condition C' as that from the original one. By repeating such operation while the formula (3) is satisfied, we can construct the smallest decision table equivalent to the original one, which is called a *reduct* of the original one [4].

Figure 3.3 shows an example of knowledge integration between two domain-experts, who has some pieces of knowledge of products. In this figure, a domain-expert in a sales department has some pieces of knowledge to classify the resources into four groups, that is, "premium", "attachment", "product" and "part". On the other hand, another expert in a manufacturing department has the different pieces of knowledge to classify the resources into such three groups as "component", "material" and "product". By integrating those pieces of knowledge, we obtain more granular classification shown as $U/(R_1 \text{ and } R_2) = U/(R_1 \cap R_2)$.

3.1.2 Identifying the Static Aspect of Requirements

The static model units are the constituents composing the enterprise models from *resource*, *organization*, and *task* viewpoints respectively. Those units can be regarded as *concepts* defined in the sets of all the relevant substances or instances to the above viewpoints, which are denoted by U_1, U_2 and U_3 as mentioned in the previous section.

Those U_i would be the universes in terms of RST, and the concepts would be the classes defined in those universes by the knowledge of each domain-experts.

As stated above, knowledge in RST is the ability to classify objects, and represented by an equivalence relation over the set of the objects. This knowledge can also be regarded as the viewpoint on the objects. For example, if an expert has the knowledge reflecting the viewpoint of a project, then it classify U_2 (organizational

[4]There could be multiple *reducts* in one decision table

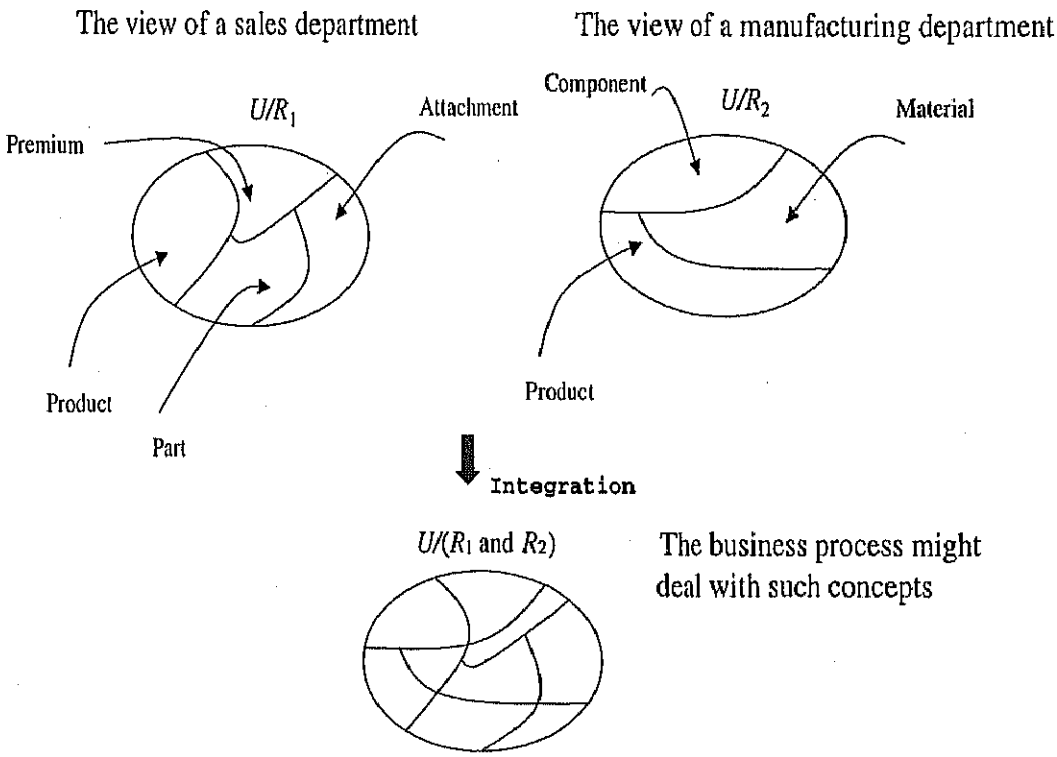

Figure 3.3: An example of knowledge integration

objects) into the teams related to the project[5].

Since each expert has multiple viewpoints on each $U_j (j = 1, 2, 3)$, the knowledge of the domain-expert e_i is denoted by

$$\mathrm{R}_j^{(i)} = \{R_{j1}^{(i)}, R_{j2}^{(i)}, \dots\}$$

where $R_{jk}^{(i)}$ is the kth knowledge of the expert e_i on universe U_j.

The total knowledge we acquire from these experts is

$$\mathrm{R}_1 = \bigcup_{i=1}^{p} \mathrm{R}_1^{(i)}, \mathrm{R}_2 = \bigcup_{i=1}^{p} \mathrm{R}_2^{(i)}, \mathrm{R}_3 = \bigcup_{i=1}^{p} \mathrm{R}_3^{(i)},$$

where p is the number of the experts, and they compose the knowledge bases in the enterprise,

$K_1 = (U_1, \mathrm{R}_1), K_2 = (U_2, \mathrm{R}_2), K_3 = (U_3, \mathrm{R}_3)$.

We can define the basic model units as the classes classified by the knowledge.

The task units, the organization units and the resource units will be denoted by the following respectively.

$$\mathrm{X} = \{X \in U_1/IND(\mathrm{R}_1)\}$$
$$\mathrm{Y} = \{Y \in U_2/IND(\mathrm{R}_2)\}$$
$$\mathrm{Z} = \{Z \in U_1/IND(\mathrm{R}_3)\}$$

where $IND(\mathrm{R}_i) = \cap \mathrm{R}_i$ is the indiscernibility relation over R_i as defined in the previous section.

The above model units correspond to the finest classification of U_i $(i = 1, 2, 3)$, and can be regarded as *building blocks* or *components* in an enterprise model. Any particular application can be composed by those *building blocks*, therefore they are reusable and could contribute to reduce the number of objects to be managed in enterprise modeling.

When a new application arises along with new pieces of knowledge on an enterprise, we can evaluate whether those pieces are really new or are expressed by the current knowledge in the following way.

1. Let a set of equivalence relations corresponding to those new pieces of knowledge of U_1 (resource), U_2 (organization) and U_3 (task) be R'_1, R'_2 and R'_3 respectively, and the model units based on them be:
   $\mathrm{X}' = \{X'_1, X'_2, \dots\}$
   $\mathrm{Y}' = \{Y'_1, Y'_2, \dots\}$
   $\mathrm{Z}' = \{Z'_1, Z'_2, \dots\}$

2. Calculate lower and upper *approximation* of each X'_i, Y'_i and Z'_i based on the current enterprise-wide knowledge R_1, R_2 and R_3 respectively, which are the basis of current model units.

[5]people who are not involved in the project will be classified into one class, say, named *unrelated class*

3. Let $\mathfrak{R}_i$ be $IND(\mathbf{R}_i)$. If the following three conditions
   $\underline{\mathfrak{R}_1}X_i = \overline{\mathfrak{R}_1}X_i$, $\underline{\mathfrak{R}_2}Y_i = \overline{\mathfrak{R}_2}Y_i$, and $\underline{\mathfrak{R}_3}Z_i = \overline{\mathfrak{R}_3}Z_i$
   hold, we can express the new application by the current model units. Otherwise, we need to integrate those new knowledge $\mathbf{R}'_1$, $\mathbf{R}'_2$ and $\mathbf{R}'_3$ into our current knowledge.

We did not consider any hierarchy among the model units in the previous discussion. However, from practical viewpoints, those model units often compose hierarchy. For example, a model unit "order form" might be composed of other model units like "order number", "product name", "quantity" and so on.

In order to deal with such hierarchy, we introduce the concept of *level* in model units. We define the level of a model unit X_i is 0 when it is not composed of other model units, and we denote it by X_{0i}. Level 1 model units are defined as those which are composed of level 0 model units. We can define level n model units inductively as those which are composed of at most level $(n-1)$ model units.

A level n model unit is denoted by X_{ni}, and would be expressed by a Cartesian product of other model units like:

$X_{ni} = X_{i_1 j_1} \times \ldots \times X_{i_m j_m}$

where $i_1, \ldots, i_m \leq n-1$ and $max(i_1, \ldots, i_m) = n-1$.

Any model unit can be expressed by a Cartesian product of level 0 model units by decomposing it iteratively until all the model units composing it become level 0.

Hierarchy of model units can be defined in all the types of static model units, that is, *resource*, *organization* and *task*. However, we only focus on hierarchy in *resource* model units, since hierarchy in *organization* or *task* model units can be expressed as that of *data* representing *organization* and *task*, and *data* are included in *resource* in our definition.

In order to show how this method works, let us think of a reference model process, which represents order processing including "*order acceptance*", "*order validation*", "*production*", "*shipment*" and so on. Assuming that there are two domain-experts named e_1 and e_2, from a sales department and a production department respectively, we would be provided with the pieces of knowledge in the form of $U_j/IND(\mathbf{R}_j^{(i)})$ as the static model units, that is, we would have the pieces of knowledge on U_1, U_2 and U_3 as:

$\mathbf{X}^{(i)} = \{X_1^{(i)}, \ldots, X_{m_i}^{(i)}\}$ (concepts or classes of e_i in U_1)

$\mathbf{Y}^{(i)} = \{Y_1^{(i)}, \ldots, Y_{n_i}^{(i)}\}$ (concepts or classes of e_i in U_2)

$\mathbf{Z}^{(i)} = \{Z_1^{(i)}, \ldots, Z_{o_i}^{(i)}\}$ (concepts or classes of e_i in U_3)

Table 3.3 shows an example of such pieces of knowledge we could obtain.

As mentioned above, enterprise-wide static model units are obtained by the equivalence relations $\mathbf{R}_1^{(1)} \cup \mathbf{R}_1^{(2)}$, $\mathbf{R}_2^{(1)} \cup \mathbf{R}_2^{(2)}$ and $\mathbf{R}_3^{(1)} \cup \mathbf{R}_3^{(2)}$. The pieces of

Table 3.3: Integrating knowledge of the experts

e_1	e_2
$\mathbf{X}^{(1)} = U_1/IND(\mathbf{R}_1^{(1)})$	$\mathbf{X}^{(2)} = U_1/IND(\mathbf{R}_1^{(2)})$
$X_1^{(1)}$: product name	$X_1^{(2)}$: product number
$X_2^{(1)}$: product number	$X_2^{(2)}$: quantity
$X_3^{(1)}$: customer name	
...	...
$\mathbf{Y}^{(1)} = U_2/IND(\mathbf{R}_2^{(1)})$	$\mathbf{Y}^{(2)} = U_2/IND(\mathbf{R}_2^{(2)})$
$Y_1^{(1)}$: reception office	$Y_1^{(2)}$: sales office
$Y_2^{(1)}$: inventory management	$Y_2^{(2)}$: production planner
...	...
$Y_9^{(1)}$: factory	
...	
$\mathbf{Z}^{(1)} = U_3/IND(\mathbf{R}_3^{(1)})$	$\mathbf{Z}^{(2)} = U_3/IND(\mathbf{R}_3^{(2)})$
$Z_1^{(1)}$: order entry	$Z_1^{(2)}$: sales activity
$Z_2^{(1)}$: inventory check	$Z_2^{(2)}$: production
...	...
$Z_4^{(1)}$: evaluation	
...	
$Z_8^{(1)}$: production	
...	

knowledge in Table 3.3 would be integrated by those equivalence relations into the enterprise-wide model units in Table 3.4.

A more complete example is shown in appendix B.

3.1.3 Identifying the Functional Aspect of Requirements

The functions discussed in this thesis are *transformation rules* between the *resources* identified in an enterprise. As mentioned above, there could be hierarchies in those resources, or resource model units, and domain-experts might express those functions (or transformation rules) using arbitrary level of resource model units. Such arbitrary use of resource level prevents rigorous comparison between the functions provided by the domain-experts, and would become obstacle for identifying the *enterprise-wide* functions.

Therefore, we assume all the functions are expressed with level 0 resource model units or are converted into those with level 0 resource model units. Since hierarchies are expressed by Cartesian products of lower level resource model

Table 3.4: Enterprise-wide model units

$\mathbf{X} = U_1/IND(\mathbf{R}_1^{(1)} \cup \mathbf{R}_1^{(2)})$ X_1 : product name, X_2 : product number X_3 : customer name, X_4 : quantity ...
$\mathbf{Y} = U_2/IND(\mathbf{R}_2^{(1)} \cup \mathbf{R}_2^{(2)})$ Y_1 : reception office, Y_2 : inventory management, ... Y_5 : production, Y_6 : accounting section, ...
$\mathbf{Z} = U_3/IND(\mathbf{R}_3^{(1)} \cup \mathbf{R}_3^{(2)})$ Z_1 : order entry, Z_2 : inventory check Z_3 : credit check, Z_4 : evaluation ... Z_7 : shipping, ...

units, it is possible to convert any function into one expressed by level 0 resource model units.

The transformation rule, or functionality of an function can be expressed in several ways, such as primitive recursive functions, Turing machines, specification languages and so forth. The thesis only focuses on *semantic equivalency* between functions, and do not discuss the ways to express functions in detail.

Assuming an expert e_i has the model units $X_1^{(i)}, \ldots, X_{m_i}^{(i)}$ over U_1 [6], a function $F_j^{(i)}$ provided by the expert is denoted by:

$$A_{j1}^{(i)} \times \cdots \times A_{jn_{ij}}^{(i)} \xrightarrow{F_j^{(i)}} B_{j1}^{(i)} \times \cdots \times B_{jn'_{ij}}^{(i)}$$

where $A_{jk}^{(i)}$, $B_{jl}^{(i)} \in U_1/IND(\mathbf{R}_1^{(i)}) = \{X_1^{(i)}, \ldots, X_{m_i}^{(i)}\}$, n_{ij} and $n'_{ij} \in \mathbf{N} = \{0, 1, \ldots\}$ and $max(n_{ij}, n'_{ij}) > 0$. If $n_{ij} = 0$, the function represents creation of resources, whereas if $n'_{ij} = 0$, it represents deletion of resources. In all other cases, that is, $n_{ij} > 0$ and $n'_{ij} > 0$ hold, it represents such transformation rule between resources as:

$$\{\langle x_{j1}, \ldots, x_{jn_{ij}} \rangle\} \xrightarrow{F_j^{(i)}} \{\langle y_{j1}, \ldots, y_{jn'_{ij}} \rangle\},$$

where $x_{jk} \in A_{jk}^{(i)}$ and $y_{jl} \in B_{jl}^{(i)}$. $\{A_{jk}^{(i)}\}$ and $\{B_{jl}^{(i)}\}$ compose the *carriers* of $F_j^{(i)}$ in terms of *many-sorted algebra* [79].

Those $F_j^{(i)}$ are the functional model units of each domain-expert, and could be

[6] Those X_i are the equivalence classes in U_1.

regarded as *individual* model units.

Those functions are provided by domain-experts in conjunction with the tasks associated with them. Since a task usually can perform several functions, or resource transformations in business processes, multiple functions could be associated with a task.

When two task units have a common part, which are provided by two different domain-experts e_i and e_k, the two set of functions associated with those two task units must have common parts. Arbitrary two tasks $T_j^{(i)} \in \mathrm{Z}^{(i)} = U_3/IND(\mathrm{R}_3^{(i)})$ and $T_l^{(k)} \in \mathrm{Z}^{(i)} = U_3/IND(\mathrm{R}_3^{(k)})$ are said to have a common part if $T_j^{(i)} \cap T_l^{(k)} \neq \emptyset$ holds.

Let $\mathrm{F}_j^{(i)} = \{F_{j1}^{(i)}, \ldots, F_{jm_{ij}}^{(i)}\}$ and $\mathrm{F}_l^{(k)} = \{F_{l1}^{(k)}, \ldots, F_{lm'_{kl}}^{(k)}\}$ be the set of functions associated with the task model units $T_j^{(i)}$ and $T_l^{(k)}$ respectively. If $T_i^{(i)} \cap T_l^{(k)} \neq \emptyset$ holds, that is, those task model units include some common tasks within them, there must be the *partially equivalent* functions between $\mathrm{F}_j^{(i)}$ and $\mathrm{F}_l^{(k)}$.

We define two functions $\mathbb{A} = A_1 \times \cdots \times A_m \xrightarrow{F} \mathbb{B} = B_1 \times \cdots \times B_n$ and $\mathbb{C} = C_1 \times \cdots \times C_m \xrightarrow{G} \mathbb{D} = D_1 \times \cdots \times D_n$ are *partially equivalent* if:

1. There are two permutations

$$p_C = \begin{pmatrix} 1 & 2 & \ldots & m \\ r_1 & r_2 & \ldots & r_m \end{pmatrix}$$

and

$$p_D = \begin{pmatrix} 1 & 2 & \ldots & n \\ s_1 & s_2 & \ldots & s_n \end{pmatrix}$$

which satisfy $A_u \cap C_{r_u} \neq \emptyset$ and $B_v \cap D_{s_v} \neq \emptyset$
for all $u \in \{1, \ldots, m\}$ and $v \in \{1, \ldots, n\}$.

2. $\forall \vec{x} \in (A_1 \cap C_{r_1}) \times \cdots \times (A_m \cap C_{r_m})$
$[F_j^{(i)}(\vec{x}) = F_l^{(k)}(\vec{x}) \in (B_1 \cap D_{s_1}) \times \cdots \times (B_n \cap D_{s_n})]$

When the two functions F and G satisfy the above conditions, we can define the integrated function $\mathcal{F}$ as:

$\Delta = (A_1 \cap C_{r_1}) \times \cdots \times (A_m \cap C_{r_m}) \xrightarrow{\mathcal{F}} \Gamma = (B_1 \cap D_{s_1}) \times \cdots \times (B_n \cap D_{s_n})$
where $\mathcal{F}(\vec{x}) = F_j^{(i)}(\vec{x}) = F_l^{(k)}(\vec{x})$ if $\vec{x} \in \Delta$.

In addition, two functions are defined, which are the remainders of the integrated function $\mathcal{F}$, as the restrictions [7] of $F_j^{(i)}$ to $\mathbb{A} - \Delta$ and that of $F_l^{(k)}$ to $\mathbb{C} - \Delta$.

[7] a function $g : B \xrightarrow{g} C$ is called the restriction of a function $f : A \xrightarrow{f} C$, if $B \subseteq A$ and $\forall x \in B[g(x) = f(x)]$ holds.

We denote them by $\bar{F}_j^{(i)}$ and $\bar{F}_l^{(k)}$ respectively.

The functions provided by all the domain-experts are integrated by applying the above method in the following way.

First, let $\mathbf{F}_0$ be the set of all the functions that each expert provides, that is, $\mathbf{F}_0 = \{F_j^{(I)}\} = \{F_1^{(1)}, \ldots, F_1^{(p)}, \ldots\}$.

Select two functions which can be integrated in terms of the above discussion, and replace them with the integrated function and the remainders to make the new set of functions $\mathbf{F}_1$. By repeating this operation until there is no pair of functions to be integrated, we would obtain $\mathbf{F}_2$, $\mathbf{F}_3, \ldots \mathbf{F}$ iteratively. $\mathbf{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots\}$ is the set of functions which can not be integrated any more.

Those functions $\mathcal{F}_i$ are the enterprise-wide function model units, since they are derived using all the pieces of knowledge on the functions reside in the enterprise.

The domain of definition and the image of $\mathcal{F}_i$, denoted by:

$\Delta_i = \mathcal{D}_i 1 \times \cdots \times \mathcal{D}_{im_i}$ and

$\Gamma_i = \mathcal{E}_i 1 \times \cdots \times \mathcal{E}_{in_i}$

might include $\mathcal{D}_j$ and $\mathcal{E}_k$ which are identical to none of the enterprise-wide resource model units, however they can be expressed by a union of some enterprise-wide resource model units.

In order to show how this method works, we use the following two functions F and G, assuming that they are provided by the different domain-experts in customer order processing. The function F and G are:

$A_1 \times A_2 \times A_3 \xrightarrow{F} B_1$

$C_1 \times C_2 \times C_3 \xrightarrow{G} D_1$

where A_1 : product number, A_2 : order number, A_3 : quantity, C_1 : parts number, C_2 : order number, C_3 : quantity, B_1 : warehouse number and D_1 : warehouse number.

The function F transforms the tuple of ⟨product number, order number, quantity⟩ to ⟨warehouse number⟩, while G transforms ⟨parts number, order number, quantity⟩ to ⟨warehouse number⟩.

Assuming that $A_1 \cap C_1 \neq \emptyset$, $A_2 = C_2$, $A_3 = C_3$, $B_1 = D_1$, and F and G are identical in the domain $(A_1 \cap C_1) \times A_2$, we can identify the integrated function $\mathcal{F}$ as:

$(A_1 \cap C_1) \times A_2 \times A_3 \xrightarrow{\mathcal{F}} B_1$

where $\mathcal{F}$ is the restriction of F (and G) to the subdomain $(A_1 \cap C_1) \times A_2 \times A_3$ in the way discussed in this section.

3.1.4 Identifying the Behavioral Aspect of Requirements

As mentioned in Section 3.1, the behavior of an enterprise is defined as "a set of ordered sequences of tasks which are associated with organization". We define

the ordered sequence of tasks in the following way.

First we define the predecessors and successors of a task to determine the sequence. Predecessors are the pre-conditional tasks to perform that task. Successors of a task are the tasks which must be performed after that task ends.

The basic structure of the behavior of an enterprise is depicted by a set of tuples composed of "predecessors", "a task (a central task)" and "successors". In addition, since a task is associated with an organization and functions, we define a basic structure of behavior (or a behavioral model unit) as a tuple of "predecessors", "a task", "an organization", "functions" and "successors".

In order to make the integration of those behavioral model units, we divide the tuple into the following three parts.

1. Task-task relationships, which express the sequence of task execution.
2. Task-organization relationships, which express the performers of tasks.
3. Task-function relationships, which represent the resources to be dealt with by the tasks, and the transformation rules of the relevant resources.

In the same way as the other model units, those of *behavior* are also elicited from each domain-expert, and the above relationships are composed by the *individual* model units of *tasks*, *organization* and *functions*, when they are provided by the experts. Therefore, we have to integrate those *individual* relationships into *enterprise-wide* relationships in the enterprise.

We can integrate the *task-task* relationships and *task-organization* relationships in the similar way. First, we discuss the integration of *task-task* relationships.

Let the knowledge base on U_3 (tasks) of a domain-expert e_i be
$K_3^{(i)} = (U_3, \mathrm{R}_3^{(i)})$.
The set of model units of tasks, which the expert e_i owns, is denoted by
$\mathrm{Z}^{(i)} = U_3/IND(\mathrm{R}_3^{(i)}) = \{Z_1^{(i)}, \ldots, Z_{m_i}^{(i)}\}$
There could be several types of *task-task* relationships, however following four types would be typical ones in many business processes.

1. One or more tasks succeed to a task
2. A task succeeds to the completion of a set of tasks
3. A task can be performed with no conditions
4. A task has no succeeding tasks

The above control structure is denoted by a set of such tuples as

$(\langle P_{j1}^{(i)}, \ldots, P_{jm_{ij}}^{(i)} \rangle, T_j^{(i)}, \langle S_{j1}^{(i)}, \ldots, S_{jn_{ij}}^{(i)} \rangle)$,

where $P_{jk}^{(i)} \in \mathrm{Z}_{(i)}$ is a task to be performed as *precondition* to a task $T_j^{(i)} \in \mathrm{Z}_{(i)}$, whereas $S_{jl}^{(i)} \in \mathrm{Z}_{(i)}$ means the succeeding task to be performed after $T_j^{(i)}$.

When a task $T_j^{(i)}$ has precondition tasks and succeeding tasks, $\langle P_{jk}^{(i)} \rangle$ and $\langle S_{jl}^{(i)} \rangle$ are non-empty, while if it does not have those tasks, the corresponding tuple $\langle P_{jk}^{(i)} \rangle$ or $\langle S_{jl}^{(i)} \rangle$ would be empty.

The two pieces of knowledge owned by two different domain-experts e_i and e_k, which are denoted by:

$\tau_{ij} = (\langle P_{j1}^{(i)}, \ldots, P_{jm_{ij}}^{(i)} \rangle, T_j^{(i)}, \langle S_{j1}^{(i)}, \ldots, S_{jn_{ij}}^{(i)} \rangle)$

and

$\tau_{kl} = (\langle P_{l1}^{(k)}, \ldots, P_{lm_{kl}}^{(k)} \rangle, T_l^{(k)}, \langle S_{l1}^{(k)}, \ldots, S_{ln_{kl}}^{(k)} \rangle)$

respectively, can be integrated if they satisfy the following conditions.

1. $T_j^{(i)} \cap T_l^{(k)} \neq \emptyset$
2. $\forall r \in \{j1, \ldots, jm_{ij}\}$
   $[P_r^{(i)} \cap T_l^{(k)} \neq \emptyset$ or $\exists s \in \{l1, \ldots, lm_{kl}\}[P_r^{(i)} \cap P_s^{(k)} \neq \emptyset]]$
   and
   $\forall s \in \{l1, \ldots, lm_{kl}\}$
   $[T_j^{(i)} \cap P_s^{(k)} \neq \emptyset$ or $\exists r \in \{j1, \ldots, jm_{ij}\}[P_r^{(i)} \cap P_s^{(k)} \neq \emptyset]]$
3. $\forall r \in \{j1, \ldots, jn_{ij}\}$
   $[S_r^{(i)} \cap T_l^{(k)} \neq \emptyset$ or $\exists s \in \{l1, \ldots, ln_{kl}\}[S_r^{(i)} \cap S_s^{(k)} \neq \emptyset]]$
   and
   $\forall s \in \{l1, \ldots, ln_{kl}\}$
   $[T_j^{(i)} \cap S_s^{(k)} \neq \emptyset$ or $\exists r \in \{j1, \ldots, jn_{ij}\}[S_r^{(i)} \cap S_s^{(k)} \neq \emptyset]]$

The first condition requires the center tasks of the behavioral model units must have a common part. The second condition requires all the predecessors in one behavioral model unit must have common parts with either:

1. the center task of the other model unit, or
2. at least one of the predecessors of the other model unit.

The third is the same condition for the successors as the second.

Those conditions represent that any task in a *task-task* relationship of one domain-expert is overlapped with some tasks in a *task-task* relationship of the another expert.

When two pieces of knowledge τ_{ij} and τ_{kl} satisfy the above condition, those two pieces can be regarded as *overlapped*, and we can derive the integrated piece of knowledge from them by the following procedure.

1. define the new task units $\mathcal{Z}$, $\mathcal{Z}^{(i)}$ and $\mathcal{Z}^{(k)}$ as
   $\mathcal{Z} = T_j^{(i)} \cap T_l^{(k)}$,
   $\mathcal{Z}^{(i)} = T_j^{(i)} - \mathcal{Z}$ and
   $\mathcal{Z}^{(i)} = T_l^{(k)} - \mathcal{Z}$.

2. For each $P_r^{(i)}$ $(r = j1, \ldots, jm_{ij})$ in τ_{ij}, define the new task units $\mathcal{P}_{r_a}^{(i)}$ $(a = 1, 2, \ldots)$ and $\overline{\mathcal{P}}_r^{(i)}$ as $\mathcal{P}_{r_a}^{(i)} = P_r^{(i)} \cap P_{s_a}^{(k)}$ and $\overline{\mathcal{P}}_r^{(i)} = P_r^{(i)} - \cup \mathcal{P}_{r_a}^{(i)}$ respectively, where $P_{s_a}^{(k)} \in \{P_{l1}^{(k)}, P_{l2}^{(k)}, \ldots\}$ in τ_{kl} satisfies $P_r^{(i)} \cap P_{s_a}^{(k)} \neq \emptyset$. There are at least one such $P_{s_a}^{(k)}$ according to the assumption above. On the other hand, for each $P_s^{(k)}$ $(s = l1, \ldots, lm_{kl})$ in τ_{kl}, define the new task units $\mathcal{P}_{s_b}^{(k)}$ and $\overline{\mathcal{P}}_s^{(k)}$ as
   $\mathcal{P}_{s_b}^{(k)} = P_s^{(k)} \cap P_{r_b}^{(i)}$ and $\overline{\mathcal{P}}_s^{(k)} = P_s^{(k)} - \cup P_{s_b}^{(k)}$ respectively,
   where $P_{r_b}^{(i)} \in \{P_{j1}^{(i)}, \ldots, P_{jm_{ij}}^{(i)}\}$ in τ_{ij} satisfies $P_s^{(k)} \cap P_{r_b}^{(i)} \neq \emptyset$. Evidently, $\{\mathcal{P}_r^{(i)}\} = \{\mathcal{P}_s^{(k)}\}$ holds.

3. For each $S_t^{(i)}$ in τ_{ij} and $S_u^{(k)}$ in τ_{kl}, define the new task units $\mathcal{S}_{t_c}^{(i)}$, $\overline{\mathcal{S}}_t^{(i)}$, $\mathcal{S}_{u_d}^{(k)}$ and $\overline{\mathcal{S}}_u^{(k)}$ in the same way as the step 2.

By the above procedure, we will derive the integrated piece of knowledge, or in other words, the common part of two pieces of the knowledge between τ_{ij} and τ_{kl}, in the form of

$$\tau = (\langle \mathcal{P}_{r_a}^{(i)} \rangle, \mathcal{T}, \langle \mathcal{S}_{t_c}^{(i)} \rangle),$$

where $r = j1, \ldots, im_{ij}$, $t = j1, \ldots, jn_{ij}$, $a = 1, 2, \ldots$ and $c = 1, 2, \ldots$.

The rest of the pieces of knowledge τ_{ij} and τ_{kl} is denoted by

$$\overline{\tau}_{ij} = (\langle \overline{\mathcal{P}}_{j1}^{(i)}, \ldots, \overline{\mathcal{P}}_{jm_{ij}}^{(i)} \rangle, \mathcal{T}^{(i)}, \langle \overline{\mathcal{S}}_{j1}^{(i)}, \ldots, \overline{\mathcal{S}}_{jn_{ij}}^{(i)} \rangle)$$

and

$$\overline{\tau}_{kl} = (\langle \overline{\mathcal{P}}_{l1}^{(k)}, \ldots, \overline{\mathcal{P}}_{lm_{kl}}^{(k)} \rangle, \mathcal{T}^{(k)}, \langle \overline{\mathcal{S}}_{l1}^{(k)}, \ldots, \overline{\mathcal{S}}_{ln_{kl}}^{(k)} \rangle).$$

The set of all pieces of the knowledge on *task-task* relationships $\mathbf{T}_1 = \{\tau_{ij}\}$can be integrated by the following way.

1. Select two pieces of knowledge τ_{ij} and τ_{kl} in $\mathbf{T}_1$, and replace them by τ, $\overline{\tau}_{ij}$ and $\overline{\tau}_{kl}$ if they can be integrated. Otherwise, select another pair.

2. Repeat 1 until there are no pairs to be integrated in the set $\mathbf{T}_1$.

This procedure produces a set of integrated pieces of knowledge on *task-task* relationships

$\mathrm{T} = \{(\langle \mathcal{P}_{i1}, \ldots, \mathcal{P}_{im} \rangle, \mathcal{T}_i, \langle \mathcal{S}_{i1}, \ldots, \mathcal{S}_{in} \rangle)\}$.

Those $\mathcal{P}_{ix}$, $\mathcal{T}_i$ and $\mathcal{S}_{iy}$ would be the new enterprise-wide task units for the behavioral aspect of the enterprise, if they are different from the enterprise-wide task units identified in Section 3.1.2.

The two other task related relationships, that is, *task-organization* relationship and *task-function* relationship can be integrated in the similar way.

As for *task-organization* relationship, we first assume that each task is performed by a fixed one organization [8]. A *task-organization* relationship owned by an expert e_i is denoted by a set of such tuples as

$(\langle P_{j1}^{(i)}, \ldots, P_{jm_{ij}}^{(i)} \rangle, T_j^{(i)}, O_j^{(i)}, \langle S_{j1}^{(i)}, \ldots, S_{jn_{ij}}^{(i)} \rangle)$

where $O_j^{(i)} \in \mathrm{Y}^{(i)} = U_2/IND(\mathrm{R}_2^{(i)}) = \{y_1^{(i)}, \ldots\}$.

The integration of two pieces of such knowledge can be performed in the similar way to *task-task* relationship, since one intersection operation $O_i^{(i)} \cap O_l^{(k)}$ is added to the integration of *task-task* relationships. Therefore, we would obtain enterprise-wide *task-organization* model unit in the form of:

$\mathrm{T}_o = \{\langle \mathcal{P}_{i1}, \ldots, \mathcal{P}_{im} \rangle, \mathcal{T}_i, \mathcal{O}_i, \langle \mathcal{S}_{i1}, \ldots, \mathcal{S}_{in} \rangle\}$

where $\mathcal{O}_i$ is the enterprise-wide model unit of *organization* for the behavior of the enterprise. The *task-function* relationships are also integrated in the similar way. As mentioned in Section 3.1.3, each task is associated with several functions. When two tasks $T_j^{(i)}$ and $T_l^{(k)}$ satisfy $T_j^{(i)} \cap T_l^{(k)} \neq \emptyset$, we can integrate the associated functions.

Let the integrated functions of $T_j^{(i)} \cap T_l^{(k)}$ be $\mathcal{F}_1, \ldots, \mathcal{F}_p$. In the procedure of *task-task* relationships integration, we can associate those $\mathcal{F}_1, \ldots, \mathcal{F}_p$ to the integrated task unit $\mathcal{Z} = T_j^{(i)} \cap T_l^{(k)}$. By adding this operation to the procedure, we would obtain a set of tuples:

$\mathrm{T}_f = \{(\langle \mathcal{P}_{i1}, \ldots, \mathcal{P}_{im} \rangle, \mathcal{T}_i, \langle \mathcal{F}_1, \ldots, \mathcal{F}_p \rangle, \langle \mathcal{S}_{i1}, \ldots, \mathcal{S}_{in} \rangle)\}$.

The above T, T_o and T_f are combined into the form:

$\mathrm{B} = \{(\langle \mathcal{P}_{ij} \rangle, \mathcal{T}_i, \mathcal{O}_i, \langle \mathcal{F}_{ik} \rangle, \langle \mathcal{S}_{il} \rangle)\}$

where $j = 1, \ldots, m$, $k = 1, \ldots, p$ and $l = 1, \ldots, n$. An example of the procedure discussed in this section is shown in Appendix B.

3.2 Expressing the Requirements in CPN

After all the enterprise-wide model units are identified, we have to compose an enterprise model or models by expressing the structure of the enterprise using

[8] When this assumption is not satisfied, we can decompose the piece of knowledge until it is satisfied

those model units.

The enterprise model can be regarded as the basis for software system construction, and should be expressed in such ways as:

1. describing functionality and behavior of the enterprise
2. being easily transformed to software systems
3. being understandable by the people both in business realm and software realm

Among numerous model notation methods, Petri-nets, especially Colored Petri Nets (CPN) are one of the most suitable methods satisfying the above 1 through 3 [1], [4], [64], since we can imbed any functions or functional notations in CPN models, which will be used as the specification of the implementation of it. Besides, CPN provide us with rich capability of graphical notation, which will be understandable by the experts in the business realm and software realm.

CPN are one of the enhancements of Petri nets, and formally defined as follows [39].

$CPN=(S, P, T, A, N, C, G, E, I)$,

where

S : a finite set of non-empty types, called color sets,
P : a finite set of places,
T : a finite set of transitions,
A : finite set of arcs $P \cap T = P \cap A = T \cap A = \emptyset$,
N : node function $A \rightarrow P \times T \cup T \times P$,
C : a color function $P \rightarrow S$,
G : a guard function $T \rightarrow$ expression,
E : an arc expression function $A \rightarrow$ expression[9] and
I : an initialization function :
$P \rightarrow$ closed expression.

In this thesis, we use Petri net diagrams shown in Figure 3.4 and description tables of CPN like Table 3.5. A Petri net diagram shows the structure of a CPN model, while a description table expresses the behavioral characteristics of a CPN model. Figure 3.4 and Table 3.5 describe a simple order processing. A token in the place p_1 represents an order denoted by (x, i), where x is a product name and i is the number of the ordered product x. A token in p_2 represents the currently available product which is denoted by (y, j), where y is the name of the product and j is the price of the product y. A token in p_3 represents the product in a basket, and a token in p_4 represents the sales amount. The notation "s`" before a token symbol means "there are s tokens in the place".

[9]The function can yields multiple tokens, or more strictly speaking, a multi-set over the color

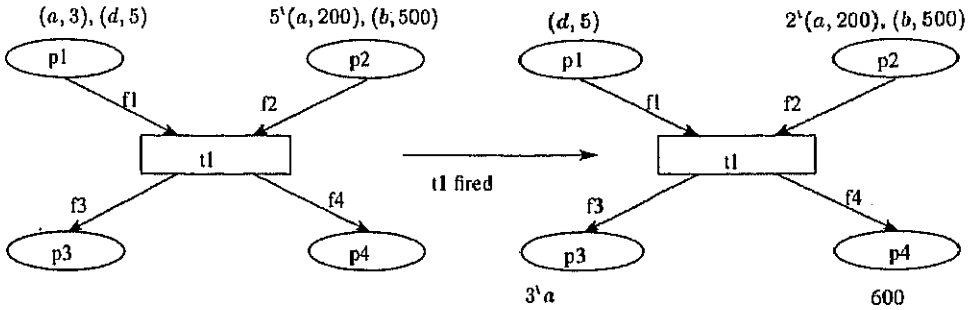

Figure 3.4: CPN diagram

Table 3.5: CPN description table

Color sets
P : {a, b, c, d, e} (product name)
I : {i } i=1,2,... (integer)
O : (x, i) $(x \in P, i \in I)$ (order), S : (y, j) $(y \in P, j \in I)$ (stock)
Initial Marking (Initialization function)
$M(p_1) = \{(a, 3), (d, 5)\}$
$M(p_2) = \{5`(a, 200), (b, 500)\}$
Guard function
None
Color function
$C(p_1) = O, C(p_2) = S, C(p_3) = P, C(p_4) = I$
Arc function E(a)
$f1 = Id(a = p_1 \rightarrow t_1)$
$f2 = 5`Id(a = p_2 \rightarrow t_1)$
$f3 = i`x(a = t_1 \rightarrow p_3)$
$f4 = i * j(a = t_1 \rightarrow p_4)$

As mentioned in Section 3.1, we focus on business processes in order to build enterprise models. Therefore, CPN must be able to deal with business processes for use in enterprise modeling. Petri nets including CPN are proven to have the ability to express business processes from various aspects [1], [24], [64]. Before discussing CPN modeling of business processes, we first define a business process more formally than we did in Section 3.1. We define a business process in the following way.

1. A business process is a set of interrelated "activities" or "tasks" of which execution sequence is controlled by some "business rules".

2. An activity is a set of transformational functions which affect resources

set in the corresponding place.

and/or information, and are performed by an organization

3. A business rule is a logic to perform business process, including constraints and/or conditions for performing the activities.

An interpretation of CPN models in terms of business processes is as follows.

1. Each transition represents an activity in a business process, which transforms resources and/or information. The transformation rules are described by arc expression functions.

2. Each input place represents an organization or a person that performs an activity.

3. Structure of a CPN model, that is, connections between places and transitions, represents the business rules which control the business process.

4. Guard and initialization functions also represent the business rules.

5. Each token represents resource or information processed by the activities. A color set represents a type of those resources and information.

Such CPN models will be composed of the model units discussed in this thesis in the following way.

1. For each behavioral model unit
   $(\langle \mathcal{P}_{ij} \rangle, \mathcal{T}_i, \mathcal{O}_i, \langle \mathcal{F}_{ik} \rangle, \langle \mathcal{S}_{il} \rangle)$,
   put $\mathcal{O}_i$ and $\mathcal{T}_i$ as a place and a transition respecticely, then draw an arc from $\mathcal{O}_i$ to $\mathcal{T}_i$.

2. Draw an arc from $\mathcal{P}_{ij}$ to $\mathcal{O}_{ij}$, if the transition $\mathcal{P}_{ij}$ was put in step 1.

3. For the transition with no succeeding places, put places as *terminators*.

4. For each transition $\mathcal{T}$ in the model, let $\mathcal{O}$ be the place associated with it, $\mathcal{P}_1, \ldots, \mathcal{P}_M$ be the transitions from which input arcs come to $\mathcal{O}$, and $\mathcal{O}_1, \ldots, \mathcal{O}_N$ be the places to which arcs go out from $\mathcal{T}$. In addition, let $\mathcal{F}_1, \ldots, \mathcal{F}_p$ be such functions associated with $\mathcal{T}$ as:
   $\mathcal{D}_{i1} \times \cdots \times \mathcal{D}_{im} \xrightarrow{\mathcal{F}_i} \mathcal{E}_{i1} \times \cdots \times \mathcal{E}_{in}$,
   $\mathcal{F}'_{j1}, \ldots, \mathcal{F}'_{jp'}$ be such functions associated with the transition $\mathcal{P}_j$ as:
   $\mathcal{D}'_{jk_1} \times \cdots \times \mathcal{D}'_{jk_{m'}} \xrightarrow{\mathcal{F}'_{jk}} \mathcal{E}'_{jk_1} \times \cdots \times \mathcal{E}'_{ik_{n'}}$. Select a subset $\Gamma \subseteq \{\mathcal{F}_{jk}\}$ which satisfies $\{\mathcal{D}_{ik}\} \subseteq \{\mathcal{E}'_{jk_l}\}$, where $\{\mathcal{E}'_{jk_l}\}$ means all the transformed resource units by the functions in Γ. Then put those functions in Γ on the associated input arcs. For the transitions which have no transitions as successors, put all associated functions on the arcs to *terminator* places.

5. Put *Proj* (projection) functions on input arcs to the transition if $\{\mathcal{E}'_{jk_l}\}$ include surplus resource units to $\{\mathcal{D}_{ik}\}$, in order to eliminate the surpluses. Otherwise, put *ID* functions on the input arcs.

We finally obtain the CPN model expressed by Figure 3.5 and Table 3.6 from the sample application. The detailed description how the procedure applied is shown in Appendix B.

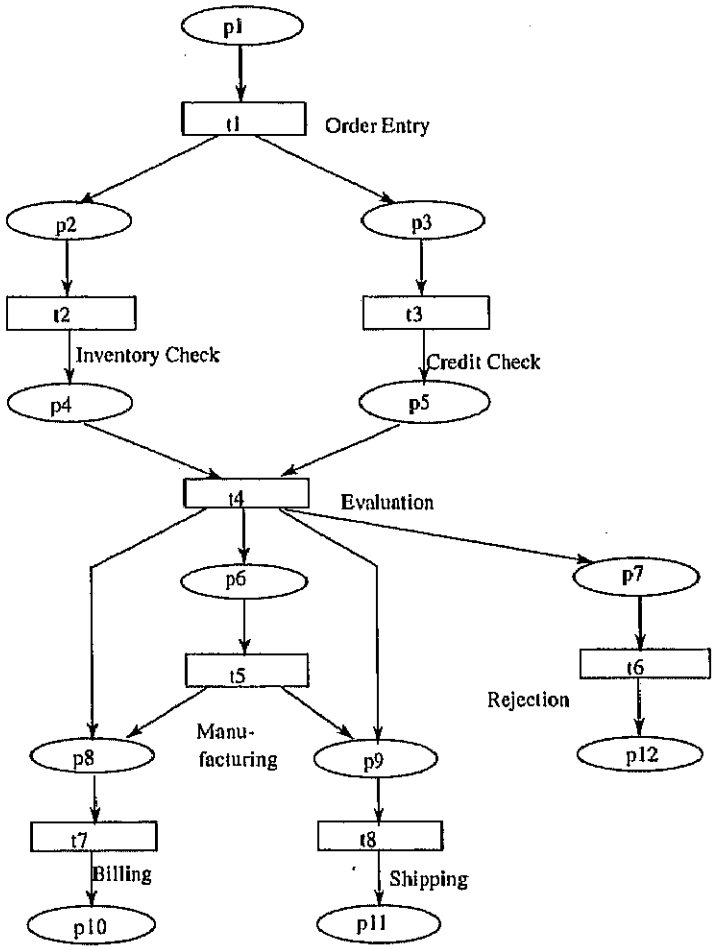

Figure 3.5: Sample CPN model

Table 3.6: Sample CPN model description

Color sets $S = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, D_1, \ldots\}$ C_1 : product name, C_2 : product number, C_3 : quantity, C_4 : customer name, C_5 : credit number, C_6 : customer address, C_7 : check result, C_8 : order number, C_9 : warehouse number, C_{10} : Price $D_1 = C_1 \times C_3 \times C_4 \times C_5 \times C_6$: order form $D_2 = C_8 \times C_2 \times C_3$: availability check form $D_3 = C_8 \times C_4 \times C_5 \times C_6$: credit check form $D_4 = C_8 \times C_2 \times C_3 \times C_7 \times C_9$: shipment information $D_5 = C_8 \times C_2 \times C_7$: rejection information ...
Places $P = \{p_1, p_2, \ldots, p_{12}\}$
Transitions $T = \{t_1, t_2, \ldots, t_8\}$
Color Function $C(p_1) = D_1, C(p_2) = D_2, C(p_3) = D_3, C(p_4) = D_4, \ldots$
Arc Expression Function $E(p_1 \rightarrow t_1) = Id, E(p_2 \rightarrow t_2) = Id, \ldots$ Id : Identity function $E(t_1 \rightarrow p_2) = (h_1(x_1), h_2(x_2), Proj\ 2)$ h_1 : assigns an *order number* h_2 : transforms a *product name* to a *product number* $Proj\ i$: projection function $E(t_1 \rightarrow p_3) = (h_1(x_1), Proj4, Proj3, Proj5)$ $E(t_2 \rightarrow p_4) = (Proj1, Proj2, h_3(x_1, x_2))$ $E(t_3 \rightarrow p_5) = (Proj1, Proj2, h_4(x_2))$ $h_3 : C_2 \times C_3 \rightarrow C_7$ availability check $h_4 : C_5 \rightarrow C_7$ credit check ...
Initialization function $I(p_1) = 1`(x_1, x_2, x_3, x_4, x_5) \in D_1$ $I(p) = \emptyset\ (p \neq p_1)$