Chapter 6

Conclusions

A model based formal approach is proposed in this thesis, which can contribute to constructing software systems for enterprise back-office applications in CBSD environments.

(The assumptions and the premises of the thesis)

This approach has been studied on the assumption that:

1. there are more than one domain-experts who have several pieces of knowledge on enterprise activities or operations,

2. each domain-expert provides us with correct and consistent information or pieces of knowledge, however they are different from each other between experts,

3. each domain-expert can explicitly classify all the constituents which are relative to the enterprise operations into a finite number of groups,

4. we are given a finite number of software components which are qualified, or in other words, those without any bugs, and

5. we can identify all the inputs, outputs and the relationship between them (that is, functionality) regarding to each software component.

In addition, the thesis assumes that an enterprise operation is a set of business processes.

The proposed approach regards software development lifecycle as being composed of three stages, namely enterprise modeling, software composition and adaptability evaluation.

(The differentiation from the related work)
Unlike this approach, previous methodologies and researches treated those three stages separately, or covered at most two of them. Therefore it have been difficult to construct software systems seamlessly from enterprise models, while this approach makes it possible.

In addition, as mentioned in Section 2, the related work to this thesis is not so formalized as this approach is, therefore the thesis provides us with a more comprehensive methodology for enterprise modeling and software composition than the traditional approaches do.

(The achievements)

The achievements and contributions of this thesis in each stage of software lifecycle can be summarized as follows.

In the first stage, we identify enterprise-wide model units which compose the structure of the enterprise, that is, the functionality and the behavior of the enterprise. RST provide us with several ways to integrate different pieces of knowledge owned by domain-experts. At the end of this stage, we have the accurate enterprise model expressed in CPN.

The second stage is to compose software systems from available software components, which are adaptable to the enterprise model. This stage has two steps. The first step is to identify functionally adaptable software components to the enterprise model. The algebraic aspects of the model and the components are used to find the adaptable components. The adaptability is evaluated based on \( \Sigma \) homomorphism. By using RST in this step, we can identify functional equivalency between the model and the components even though they have the different signature. The second step is to synthesize the software systems which are adaptable to the enterprise model from behavioral viewpoints. Decision tables extracted from the CPN model are used to express the behavioral aspect of the model. In this step, we can optimize the behavioral aspects of the model by using RST. This technique can minimize the size of the resultant software systems.

The third stage is the adaptability evaluation. We can formally evaluate the adaptability of the resultant software systems to the enterprise model, from static and dynamic viewpoints. \( \Sigma \) algebra is used to evaluate the static or functional adaptability of the software systems, while process algebra is used to evaluate the dynamic or behavioral adaptability of them.

The approach mainly focuses on enterprise back-office applications, however it can be easily extend for other domains, as far as they can be modeled by CPN.

(Future work)

In order to make this approach more effective, computer supported tools would be required, and for those tools, the following issues must be cleared.

1. Knowledge representation, registration and inquiry methods on computer
systems.

2. Efficient set operations for RST calculi.

3. CPN representation on computer systems.

4. Efficient extraction methods for decision tables, \( \Sigma \) algebras and CCS expressions from CPN models.

5. Glue code generation methods for decision tables.

6. Software component registration and inquiry systems.

Some part of the above issues are already implemented in several tools, e.g. CPN modeling tools [39], however we need more integrated tools implementing those issues for automated enterprise modeling and software composition according to the proposed approach.
Bibliography


Appendix A

Evaluating Adaptability from Data Viewpoint

In this chapter, a software adaptability from another viewpoint is discussed. The two types of software adaptability discussed in this thesis, that is, functional adaptability and behavioral adaptability, assume that the following conditions are satisfied.

1. All the inputs to each component are available at the beginning of the component execution.

2. Each component can perform its function instantaneously.

However, in actual environments, these conditions are not always realistic. For example, some components might get their inputs from databases which are available after the components start, and the elapsed time of their executions might not be negligibly short.

In such cases, the environment in which the components run might be changing during their execution periods. This environmental change during component executions could affect the adaptability of software systems, since the outputs of the components could vary according to the states of the environment in which they run. Software systems for enterprise back-office applications usually implemented as transaction processing systems. As stated in section 5.1, those transactions are ACID (Atomicity, Consistency, Isolation and Durability) transactions.

It has been being pointed out that traditional ACID transaction model is not sufficient to support complex business processes and operations in which many tasks are interrelated each other. Many kinds of advanced transaction models have been proposed in order to enhance the simplicity of ACID transaction model. These enhancements include nested transactions [14], SAGAS [15], [51], Contract [27], [78], and so forth [36], [42]. On exceptional conditions, such as system
failure, exceptional status, or invalid input data, database recovery mechanisms in these advanced transaction models are much more complicated than ACID transaction model because of their complex inter transaction dependencies. For example, the nested transaction model and SAGAS require those mechanisms to commit or abort an original transaction along with all the sub-transactions which were spawned by it. On the other hand, ConTract model provide us with the capability of defining recovery scenarios for failures.

Another relaxation for ACID property of transactions is workflow management systems (WFMS) which coordinate not only executions of transactions but also organizations and resources. WFMSs provide a little bit different recovery mechanisms from advanced transaction models [23], [45], [74], since they do not have database management capability, and they regard a database recovery scenario itself as a kind of workflow. Among these enhancements, compensation is recognized as one of the most important techniques for assuring data consistency [22], [42], [45]. Compensation, or also known as semantic undo, is defined as some functionality to remove the effect of committed transactions. However, the concept of compensation is defined in very vague and intuitive way. In most cases, it is up to application people to define and design compensation, although there are few guidelines or definitions of compensation from database management point of view.

A.1 Data Consistency in Software Systems

From database management system (DBMS) point of view, COMMIT or ABORT of transactions always assures data consistency if application systems are designed correctly. This is true when a single transaction performs all database updates required for a business operations that the transaction reflects. This is not a practical approach for concurrency of transaction executions. For example, let us consider a transaction processing shown in Figure A.1.

In this example, if we build single transaction for doing this business operation, we might suffer from degradation of concurrency. In many cases, we divide such a logical operation into smaller parts so called atomic transactions. These logical operations are composed of several atomic transactions which constitute units of database recovery. Such a unit is called sphere [45], [46], [67] or compensation scope [22]. We use the term sphere in this thesis.
A.2 Modeling Transaction Processing and Data Consistency

In order to discuss data consistency in transaction processing, we first define a concept of states in database systems, based on ACID transactions. Although there are several commercial transaction-processing (TP) monitors which support the above enhanced transaction models, most of the TP monitors used in enterprise back-office applications are based on ACID model. Therefore, this thesis only focuses on TP monitors for the ACID model hereafter. In addition, "locks" that TP monitors or DBMSs hold are regarded as *pessimistic locks*, that is, database update operations are always isolated from other transactions by locks as ACID property implies, since TP monitors and DBMSs used in mission critical applications usually use those locks for stable transaction processing. A state in a database system can be defined as a set of all records or tuples reside in the databases which are concerned by the application systems [41]. Transactions and spheres transform from one state to the other state according to their inputs and the states under which they are run. We denote transaction processing and state transformation as the followings.

1. State are denoted as $S = \{ r_i \}$, $r_i \in db_j$
   where $r_i$ is a database record in database $db_j$.

2. $\langle I, S_0 \rangle \xrightarrow{T} \langle S_1 \rangle$
   means a transaction $T$ transforms state $S_0$ to $S_1$ when the input of $T$ is $I$.

3. $\langle I, S_0 \rangle \xrightarrow{\Sigma} \langle S_1 \rangle$
means a sphere $\Sigma$ transforms state $S_0$ to $S_1$ when the input of $\Sigma$ is $I$.

4. $T_i = \langle f_1, \ldots, f_n \rangle$

means an instance $T_i$ of transaction type $T$ performs a series of database update functions $f_1, \ldots, f_n$ in this order. \(^1\)

## A.3 Defining Consistent States

Data consistency or consistent states in database systems are very application oriented matters and difficult to be defined formally. In addition, during concurrent transaction processing, there are some indefinite records at completion of COMMIT process of a transaction. These indefinite records are LOCKed by other transactions which have not been COMMITted or ABORTed. Therefore we need to consider the treatment of these indefinite records to define consistent state. In order to avoid these indefinite records, we define an overlaid state. This state is built by overlaying indefinite records by the beginning states of any unfinished transactions. The overlaying is done chronologically reverse order. For example, in Figure A.2, at the COMMIT time of transaction $T_1$, the state $S_1$ contains some indefinite records held by the transaction $T_2$ and $T_3$. These records are overlaid by the records in the state $S_2$ and $S_3$ in this order. With these overlaid states, a consistent state can be defined recursively in the following form.

1. The initial state of a database is a consistent state. \(^2\)

\(^1\)these functions and their sequences can be varied according to the inputs of the transaction and each state under which the function $f_k$ is executed.

\(^2\)In other words, a state is consistent just after initial database load

---

![Figure A.2: Concurrent transaction execution and overlaid state](image-url)
2. An overlaid consistent state is transformed to another overlaid consistent state by successful execution of a sphere.

There are several levels of consistent states. They are:

1. System level consistent state, which means all of the database records compose consistent state.

2. Application level consistent state, which means all records or projections of records that a specific application has interest compose consistent state for the application.

3. Sphere level consistent state, which means only a sphere interested records or projections of records compose consistent state for the sphere.

4. Sphere instance level consistent state, which means a sphere instance interested records or projections of records compose consistent state for the sphere instance.

A.4 Evaluating Consistency and Designing Compensation

Before defining compensation, we first define the concept of semantically equivalent state. Since a state of a database system is a set of records, there could be semantically equivalent states. For example, a state which includes the canceled-order record "x" is semantically equivalent to a state which does not include that record. We denote $S_1 \sim S_2$ if a state $S_1$ is semantically equivalent to a state $S_2$.

This $\sim$ is apparently equivalence relation and consistent states are divided into equivalence class $S_e$.

Compensation or a compensating transaction is a series of database updates which removes the effect of a previously executed transaction in order to make the state consistent.

Let us consider the state in Figure A.1 that Rent a Car Reservation fails. This state is inconsistent if application requires that Travel Request record, Hotel Reservation record, Rent a Car Reservation record, Flight Reservation record and Coupon Issued record must be exist simultaneously. In this case, we must remove Travel Request record, Hotel Reservation record, and Flight Reservation record.

Generic definition of compensation can be defined as follows.

1. Let $(I, S_0) \xrightarrow{\Sigma} (S_1)$ be a state transition from $S_0$ to $S_1$ by an incomplete execution of sphere $\Sigma$ which is denoted by $\Sigma'$.
2. $\Sigma'$ can be compensated by compensating sphere $\Sigma_{CT}$ if

$$\forall S_i, \Sigma_i((I, S_i) \xrightarrow{\Sigma_i} (S_i)) \Rightarrow \exists S'_0, \exists \Sigma_{CT}[(I, S'_0) \xrightarrow{\Sigma'} (S_i), (I, S_i) \xrightarrow{\Sigma_{CT}} (S'_i)]$$

and $S'_0 \sim S_i$.

$S'_0$ and $S'_i$ are semantically equivalent at sphere instance level, and $S'_0$ is consistent state for the sphere instance $\Sigma'$ (Figure A.3).

In most cases, we are not able to expect the beginning state of compensation sphere $\Sigma_{CT}$. Therefore, $\Sigma_{CT}$ should be executable from any reachable states from $S_1$.

Although we do not know any internal algorithms of $\Sigma_{CT}$, we can define the external properties as the above. An internal view of compensation is discussed below.

![Figure A.3: Compensation - external view](image)

As discussed in Section A.2, each instance of transaction execution is a series of database update functions $\langle f_1, \ldots, f_n \rangle$. Those updates are reflected in databases at COMMIT time of the transaction. Therefore, each $f_i$ does not actually transform any states. However, we can imagine a virtual state transition performed by each $f_i$. This virtual state is the state on the assumption that COMMIT is issued just after $f_i$.

Suppose $\langle I, S_0 \rangle \xrightarrow{T_i} \langle S_i \rangle$ and $T_i = \langle f_1, \ldots, f_n \rangle$.

Compensation of $T_i$ which is denoted by $\langle T_i \rangle_{CT}$, can be constructed if

1. Each $f_j$ has inverse function $f_j^{-1}$

2. $\forall S_k$ (reachable from $S_1$) [$S_k \in dom(f_j^{-1})$]

If above conditions are satisfied, compensation $\langle T_i \rangle_{CT}$ is constructed as $\langle T_i \rangle_{CT} = \langle f_n^{-1}, \ldots, f_1^{-1} \rangle$
These inverse function $f^{-1}_j$ can be replaced by a function $g_j$ that satisfies $g_j(S_k) \sim f^{-1}_j(S_k)$ for all $S_k$ which are reachable from $S_1$. The typical case of this equivalent function is to put the cancelled record instead of deleting the original record.

In order to make business process automation systems safe\(^3\), the following items should be taken into account.

- **Defining consistent states** of databases and logical units of operations (spheres).
- **Defining semantically equivalent consistent state classes**.
- **Recognizing possible state transitions** in both normal and exceptional cases.

By identifying these items, the external views of required compensations are revealed.

When designing each transaction, possible function sequences $\langle f_1, \ldots, f_n \rangle$ should be defined for every reachable (consistent) state. These sequences of functions are needed for determining the structures of compensating transactions.

Another consideration at system design time is to determine how many sphere instances could be influenced by the incomplete sphere. Since each transaction composing a sphere can externalize its database updates before the original sphere completes. These updates can be used by another sphere instances although the original sphere fails. This situation could make the state inconsistent because several spheres use erroneous records.

In order to avoid this situation, all influenced sphere instances should be compensated. We can deal with such compensation in the same way.

\(^3\)We define that a system is safe if all of the inconsistent state can be transformed to consistent state by compensation.
Appendix B

A Sample Application

This chapter shows how the proposed approach will be applied to business applications, using a simple example.

B.1 A Brief Description of the Sample Application

The sample application we use in this thesis is a business process for customer order entry and product delivery, including manufacturing and accounting activities.

By applying the proposed approach to this application, we can evaluate its feasibility in the real world from the following viewpoints.

1. Correctness
   Is the approach provide us with appropriate methods to construct the software systems which are reflecting requirements correctly?

2. Complexity
   Is the complexity of the proposed methods in this approach acceptable?

3. Scalability
   Is the approach scalable to more large-scale applications?

4. Usability
   Can we use the approach easily in comparison with the other software development methodologies?

We assume there are two domain-experts from a sales department and a manufacturing department, who are denoted by $e_1$ and $e_2$ respectively. The following is the brief description of each expert’s view of the sample application.

- The view of $e_1$
  Each customer order form is first examined at a reception office, then an
order number is assigned. The name of the ordered product is transformed into a product number. The order information is passed to an inventory management section and a credit check section in order to validate the order. The check results are passed to an order evaluation section. The result of the evaluation is passed to a manufacturing department in order to obtain a bill or a rejection letter.

- The view of e2
A sales department passes a check result of the order to an order evaluation section. This section creates one of the following information.

1. shipping and billing information
2. a product manufacturing order
3. order rejection information

Shipping information is processed by a distribution section in order to make an invoice and a shipment log. Billing information is processed by an accounting section to make a bill. Rejection information is used by a customer service section to make a rejection letter.
B.2 Identifying the Static Aspect of Requirements

We assume each domain-expert ($e_1$ and $e_2$) provides us with the following pieces of knowledge of resource, organization and task, as shown in Table B.1 and Table B.2.

<table>
<thead>
<tr>
<th>$X^{(1)}$</th>
<th>$X^{(2)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1/IND(R_1^{(1)})$</td>
<td>$U_1/IND(R_1^{(2)})$</td>
</tr>
<tr>
<td>$X_1^{(1)}$: product name</td>
<td>$X_1^{(2)}$: product number</td>
</tr>
<tr>
<td>$X_2^{(1)}$: product number</td>
<td>$X_2^{(2)}$: quantity</td>
</tr>
<tr>
<td>$X_3^{(1)}$: customer name</td>
<td>$X_3^{(2)}$: order number</td>
</tr>
<tr>
<td>$X_4^{(1)}$: customer address</td>
<td>$X_4^{(2)}$: customer name</td>
</tr>
<tr>
<td>$X_5^{(1)}$: credit card number</td>
<td>$X_5^{(2)}$: customer address</td>
</tr>
<tr>
<td>$X_6^{(1)}$: quantity</td>
<td>$X_6^{(2)}$: price</td>
</tr>
<tr>
<td>$X_7^{(1)}$: order number</td>
<td>$X_7^{(2)}$: credit card number</td>
</tr>
<tr>
<td>$X_8^{(1)}$: check result value</td>
<td>$X_8^{(2)}$: product number</td>
</tr>
<tr>
<td>$X_9^{(1)}$: warehouse number</td>
<td>$X_9^{(2)}$: amount</td>
</tr>
<tr>
<td>$X_{10}^{(1)}$: amount</td>
<td>$X_{10}^{(2)}$: warehouse number</td>
</tr>
<tr>
<td>$X_{11}^{(1)}$: order form</td>
<td>$X_{11}^{(2)}$: check result</td>
</tr>
<tr>
<td>$X_{12}^{(1)}$: availability check form</td>
<td>$X_{12}^{(2)}$: shipment information</td>
</tr>
<tr>
<td>$X_{13}^{(1)}$: credit check form</td>
<td>$X_{13}^{(2)}$: billing information</td>
</tr>
<tr>
<td>$X_{14}^{(1)}$: availability check form</td>
<td>$X_{14}^{(2)}$: manufacturing order</td>
</tr>
<tr>
<td>$X_{15}^{(1)}$: credit check result</td>
<td>$X_{15}^{(2)}$: rejection information</td>
</tr>
<tr>
<td>$X_{16}^{(1)}$: evaluation result</td>
<td>$X_{16}^{(2)}$: bill</td>
</tr>
<tr>
<td>$X_{17}^{(1)}$: rejection letter</td>
<td>$X_{17}^{(2)}$: invoice</td>
</tr>
<tr>
<td>$X_{18}^{(1)}$: bill</td>
<td>$X_{18}^{(2)}$: shipping log</td>
</tr>
<tr>
<td>$X_{19}^{(1)}$: unknown</td>
<td>$X_{19}^{(2)}$: rejection letter</td>
</tr>
<tr>
<td>$X_{20}^{(1)}$: unknown</td>
<td>$X_{20}^{(2)}$: unknown</td>
</tr>
</tbody>
</table>
Table B.2: Integrating knowledge of the experts

<table>
<thead>
<tr>
<th>Organization Model Units</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y^{(1)} = U_2/IND(R^{(1)}_2)$</td>
<td>$Y^{(2)} = U_2/IND(R^{(2)}_2)$</td>
<td></td>
</tr>
<tr>
<td>$Y_1^{(1)}$: reception office</td>
<td>$Y_1^{(2)}$: sales department</td>
<td></td>
</tr>
<tr>
<td>$Y_2^{(1)}$: inventory management</td>
<td>$Y_2^{(2)}$: production</td>
<td></td>
</tr>
<tr>
<td>$Y_3^{(1)}$: credit management</td>
<td>$Y_3^{(2)}$: accounting section</td>
<td></td>
</tr>
<tr>
<td>$Y_4^{(1)}$: manufacturing</td>
<td>$Y_4^{(2)}$: distribution section</td>
<td></td>
</tr>
<tr>
<td>$Y_5^{(1)}$: order evaluation</td>
<td>$Y_5^{(2)}$: customer service</td>
<td></td>
</tr>
<tr>
<td>$Y_6^{(1)}$: unknown</td>
<td>$Y_6^{(2)}$: order evaluation</td>
<td></td>
</tr>
<tr>
<td>$Y_7^{(1)}$: unknown</td>
<td>$Y_7^{(2)}$: unknown</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task Model Units</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z^{(1)} = U_3/IND(R^{(1)}_3)$</td>
<td>$Z^{(2)} = U_3/IND(R^{(2)}_3)$</td>
<td></td>
</tr>
<tr>
<td>$Z_1^{(1)}$: order entry</td>
<td>$Z_1^{(2)}$: sales activity</td>
<td></td>
</tr>
<tr>
<td>$Z_2^{(1)}$: inventory check</td>
<td>$Z_2^{(2)}$: production</td>
<td></td>
</tr>
<tr>
<td>$Z_3^{(1)}$: credit check</td>
<td>$Z_3^{(2)}$: billing</td>
<td></td>
</tr>
<tr>
<td>$Z_4^{(1)}$: evaluation</td>
<td>$Z_4^{(2)}$: shipping</td>
<td></td>
</tr>
<tr>
<td>$Z_5^{(1)}$: manufacturing</td>
<td>$Z_5^{(2)}$: rejection</td>
<td></td>
</tr>
<tr>
<td>$Z_6^{(1)}$: unknown</td>
<td>$Z_6^{(2)}$: evaluation</td>
<td></td>
</tr>
<tr>
<td>$Z_7^{(1)}$: unknown</td>
<td>$Z_7^{(2)}$: unknown</td>
<td></td>
</tr>
</tbody>
</table>

Integration of those different concepts between the experts $e_1$ and $e_2$ is performed in the following way.

1. Concepts on resources

\[
\begin{align*}
X_1^{(1)} &= X_8^{(2)}, \\
X_2^{(1)} &= X_2^{(2)}, \\
X_3^{(1)} &= X_3^{(2)}, \\
X_4^{(1)} &= X_4^{(2)}, \\
X_5^{(1)} &= X_5^{(2)}, \\
X_6^{(1)} &= X_6^{(2)}, \\
X_7^{(1)} &= X_7^{(2)}, \\
X_8^{(1)} &= X_8^{(2)}, \\
X_{10}^{(1)} &= X_{10}^{(2)}, \\
X_{12}^{(1)} &= X_{12}^{(2)}, \\
X_{14}^{(1)} &= X_{14}^{(2)}, \\
X_{16}^{(1)} &= X_{16}^{(2)}, \\
X_{18}^{(1)} &= X_{18}^{(2)}, \\
X_{20}^{(1)} &= X_{20}^{(2)}. \\
\end{align*}
\]

\[
\begin{align*}
X_1^{(1)} \cap X_2^{(2)} &= X_1^{(1)}, \\
X_2^{(1)} \cap X_3^{(2)} &= X_2^{(1)}, \\
X_3^{(1)} \cap X_4^{(2)} &= X_3^{(1)}, \\
X_4^{(1)} \cap X_5^{(2)} &= X_4^{(1)}, \\
X_5^{(1)} \cap X_6^{(2)} &= X_5^{(1)}, \\
X_7^{(1)} \cap X_8^{(2)} &= X_7^{(1)}, \\
X_8^{(1)} \cap X_{10}^{(2)} &= X_8^{(1)}, \\
X_{12}^{(1)} \cap X_{14}^{(2)} &= X_{12}^{(1)}, \\
X_{14}^{(1)} \cap X_{16}^{(2)} &= X_{14}^{(1)}, \\
X_{16}^{(1)} \cap X_{20}^{(2)} &= X_{16}^{(1)}. \\
\end{align*}
\]

2. Concepts on Organization

\[
\begin{align*}
Y_1^{(1)} \cap Y_1^{(2)} &= Y_1^{(1)}, \\
Y_2^{(1)} \cap Y_1^{(2)} &= Y_2^{(1)}, \\
Y_3^{(1)} \cap Y_1^{(2)} &= Y_3^{(1)}. \\
\end{align*}
\]

92
\[Y_4^{(1)} \cap Y_2^{(2)} = Y_2^{(2)}, Y_4^{(1)} \cap Y_3^{(2)} = Y_3^{(1)}, Y_4^{(1)} \cap Y_4^{(2)} = Y_4^{(1)}, Y_4^{(1)} \cap Y_5^{(2)} = Y_1^{(1)}, Y_5^{(1)} = Y_6^{(2)}\]

3. Concepts on Task
\[Z_1^{(1)} \cap Z_2^{(2)} = Z_1^{(1)}, Z_3^{(1)} \cap Z_2^{(2)} = Z_3^{(1)}, Z_3^{(1)} \cap Z_3^{(2)} = Z_3^{(2)}, Z_3^{(1)} \cap Z_3^{(3)} = Z_3^{(3)}, Z_5^{(1)} \cap Z_4^{(2)} = Z_4^{(2)}, Z_5^{(1)} \cap Z_3^{(2)} = Z_3^{(2)}\]

According to the above set operations, we obtain the following enterprise-wide model units, as shown in Table B.3.

<table>
<thead>
<tr>
<th>Table B.3: Enterprise-wide model units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enterprise-wide Resource Model Units</strong></td>
</tr>
<tr>
<td>[X = U_1/IND(R_1^{(1)} \cup R_1^{(2)})]</td>
</tr>
<tr>
<td>X&lt;sub&gt;1&lt;/sub&gt; : product name, X&lt;sub&gt;2&lt;/sub&gt; : product number, X&lt;sub&gt;3&lt;/sub&gt; : customer name</td>
</tr>
<tr>
<td>X&lt;sub&gt;4&lt;/sub&gt; : quantity, X&lt;sub&gt;5&lt;/sub&gt; : credit card number, X&lt;sub&gt;6&lt;/sub&gt; : customer address,</td>
</tr>
<tr>
<td>X&lt;sub&gt;7&lt;/sub&gt; : check result, X&lt;sub&gt;8&lt;/sub&gt; : order number, X&lt;sub&gt;9&lt;/sub&gt; : warehouse number,</td>
</tr>
<tr>
<td>X&lt;sub&gt;10&lt;/sub&gt; : price, X&lt;sub&gt;11&lt;/sub&gt; : amount, X&lt;sub&gt;12&lt;/sub&gt; : order form,</td>
</tr>
<tr>
<td>X&lt;sub&gt;13&lt;/sub&gt; : availability check form, X&lt;sub&gt;14&lt;/sub&gt; : credit check form, X&lt;sub&gt;15&lt;/sub&gt; : shipment information,</td>
</tr>
<tr>
<td>X&lt;sub&gt;16&lt;/sub&gt; : rejection information, X&lt;sub&gt;17&lt;/sub&gt; : availability check result, X&lt;sub&gt;18&lt;/sub&gt; : credit check result,</td>
</tr>
<tr>
<td>X&lt;sub&gt;19&lt;/sub&gt; : bill, X&lt;sub&gt;20&lt;/sub&gt; : rejection letter, X&lt;sub&gt;21&lt;/sub&gt; : manufacturing order,</td>
</tr>
<tr>
<td>X&lt;sub&gt;22&lt;/sub&gt; : shipment log, X&lt;sub&gt;23&lt;/sub&gt; : rejection information, X&lt;sub&gt;24&lt;/sub&gt; : billing information,</td>
</tr>
<tr>
<td>X&lt;sub&gt;25&lt;/sub&gt; : invoice</td>
</tr>
<tr>
<td><strong>Enterprise-wide Organization Model Units</strong></td>
</tr>
<tr>
<td>[Y = U_2/IND(R_2^{(1)} \cup R_2^{(2)})]</td>
</tr>
<tr>
<td>Y&lt;sub&gt;1&lt;/sub&gt; : reception office, Y&lt;sub&gt;2&lt;/sub&gt; : inventory management, Y&lt;sub&gt;3&lt;/sub&gt; : credit management,</td>
</tr>
<tr>
<td>Y&lt;sub&gt;4&lt;/sub&gt; : order evaluation section, Y&lt;sub&gt;5&lt;/sub&gt; : production, Y&lt;sub&gt;6&lt;/sub&gt; : accounting section,</td>
</tr>
<tr>
<td>Y&lt;sub&gt;7&lt;/sub&gt; : distribution section, Y&lt;sub&gt;8&lt;/sub&gt; : customer service</td>
</tr>
<tr>
<td><strong>Enterprise-wide Task Model Units</strong></td>
</tr>
<tr>
<td>[Z = U_3/IND(R_3^{(1)} \cup R_3^{(2)})]</td>
</tr>
<tr>
<td>Z&lt;sub&gt;1&lt;/sub&gt; : order entry, Z&lt;sub&gt;2&lt;/sub&gt; : inventory check, Z&lt;sub&gt;3&lt;/sub&gt; : credit check,</td>
</tr>
<tr>
<td>Z&lt;sub&gt;4&lt;/sub&gt; : evaluation, Z&lt;sub&gt;5&lt;/sub&gt; : manufacturing, Z&lt;sub&gt;6&lt;/sub&gt; : billing,</td>
</tr>
<tr>
<td>Z&lt;sub&gt;7&lt;/sub&gt; : shipping, Z&lt;sub&gt;8&lt;/sub&gt; : rejection</td>
</tr>
</tbody>
</table>

In addition, we assume some of those units are combinations of other units. Those combinations are as follows.

\[X_{12} = X_1 \times X_3 \times X_4 \times X_5 \times X_6\]
\[X_{13} = X_8 \times X_2 \times X_3\]
\[X_{14} = X_8 \times X_4 \times X_5 \times X_6\]
\[X_{15} = X_8 \times X_2 \times X_3 \times X_7 \times X_9\]
\[X_{16} = X_8 \times X_2 \times X_7\]
\[ X_{17} = X_8 \times X_2 \times X_7 \times X_4 \]
\[ X_{18} = X_8 \times X_5 \times X_7 \times X_3 \times X_6 \]
\[ X_{19} = X_1 \times X_3 \times X_4 \times X_5 \times X_6 \times X_{11} \]
\[ X_{20} = X_3 \times X_6 \times X_1 \]
\[ X_{21} = X_8 \times X_2 \times X_4 \]
\[ X_{22} = X_1 \times X_8 \times X_1 \times X_9 \]
\[ X_{23} = X_8 \times X_2 \times X_4 \times X_3 \times X_0 \]
\[ X_{24} = X_8 \times X_3 \times X_6 \times X_5 \times X_{11} \]
\[ X_{25} = X_1 \times X_4 \times X_3 \times X_6 \]
B.3 Identifying the Dynamic Aspect of Requirements

Since functional boundary between the expert $e_1$ (sales department) and $e_2$ (manufacturing department) resides around the tasks of evaluation and manufacturing, we have to integrate the knowledge of the two experts on those tasks regarding functional model units and behavioral model units.

B.3.1 Functional Model Units

Pieces of knowledge of functions around evaluation and manufacturing are assumed to be as follows:

1. The expert $e_1$ has the following piece of knowledge of evaluation.
   
   \[
   (X_{14}^{(1)}, X_{15}^{(1)}) \xrightarrow{F_1^{(1)}} X_{17}^{(1)}
   \]
   
   The function $F_1^{(1)}$ transforms availability check form ($X_{14}^{(1)}$) and credit check form ($X_{15}^{(1)}$) into evaluation result ($X_{17}^{(1)}$).

2. The expert $e_2$ has the following pieces of knowledge of evaluation
   
   \[
   X_{11}^{(2)} \xrightarrow{F_1^{(2)}} X_{12}^{(2)}
   
   X_{11}^{(2)} \xrightarrow{F_2^{(2)}} X_{13}^{(2)}
   
   X_{11}^{(2)} \xrightarrow{F_3^{(2)}} X_{14}^{(2)}
   
   X_{11}^{(2)} \xrightarrow{F_4^{(2)}} X_{15}^{(2)}
   \]
   
   The functions $F_1^{(2)}$, $F_2^{(2)}$, $F_3^{(2)}$, and $F_4^{(2)}$ are transforms evaluation result ($X_{11}^{(2)}$) into shipping information ($X_{12}^{(2)}$), billing information ($X_{13}^{(2)}$), manufacturing information ($X_{14}^{(2)}$), and rejection information ($X_{15}^{(2)}$) respectively.

Since $X_{14}^{(1)} \cap X_{12}^{(2)} = X_{14}^{(1)}$ and $X_{15}^{(1)} \cap X_{12}^{(2)} = X_{15}^{(1)}$ hold as we assumed in Appendix B.2, the inputs to $F_1^{(1)}$ and $F_2^{(2)}$ ($i = 1, \ldots, 4$) have the common sub-domain. Similarly, the outputs of $F_1^{(1)}$ and $F_2^{(2)}$ ($i = 1, \ldots, 4$) have the common sub-domain, since

\[
X_{17}^{(1)} \cap X_{12}^{(2)} = X_{12}^{(2)}, \quad X_{17}^{(1)} \cap X_{13}^{(2)} = X_{13}^{(2)}, \quad X_{17}^{(1)} \cap X_{14}^{(2)} = X_{14}^{(2)} \quad \text{and} \quad X_{17}^{(1)} \cap X_{15}^{(2)} = X_{15}^{(2)}
\]

hold. Therefore, the following new functions are the candidates of the integrated enterprise-wide functions.

\[
(X_{14}^{(1)}, X_{15}^{(1)}) \xrightarrow{F_1} X_{12}^{(2)}

(X_{14}^{(1)}, X_{15}^{(1)}) \xrightarrow{F_2} X_{13}^{(2)}

(X_{14}^{(1)}, X_{15}^{(1)}) \xrightarrow{F_3} X_{14}^{(2)}

(X_{14}^{(1)}, X_{15}^{(1)}) \xrightarrow{F_4} X_{15}^{(2)}
\]
\[(X_{14}^{(1)}, X_{15}^{(1)}) \xrightarrow{F_1} X_{15}^{(2)}\]

where \(F_i (i = 1, \ldots, 4)\) is the restriction of \(F_i^{(2)} (i = 1, \ldots, 4)\) to the common input sub-domain.

The above expression can be rewritten by using the enterprise-wide model units as follows.

\[(X_{17}, X_{18}) \xrightarrow{F_1} X_{15}\]
\[(X_{17}, X_{18}) \xrightarrow{F_2} X_{21}\]
\[(X_{17}, X_{18}) \xrightarrow{F_3} X_{24}\]
\[(X_{17}, X_{18}) \xrightarrow{F_4} X_{23}\]

Those model units are decomposed into more basic model units according to appendix B.2 as

\[X_8 \times X_2 \times X_7 \times X_8 \times X_5 \times X_7 \xrightarrow{F_1} X_8 \times X_2 \times X_3 \times X_7 \times X_8\]
\[X_8 \times X_2 \times X_7 \times X_8 \times X_5 \times X_7 \xrightarrow{F_2} X_8 \times X_2 \times X_3\]
\[X_8 \times X_2 \times X_7 \times X_8 \times X_6 \times X_7 \xrightarrow{F_3} X_3 \times X_6 \times X_5 \times X_{11}\]
\[X_8 \times X_2 \times X_7 \times X_8 \times X_5 \times X_7 \xrightarrow{F_4} X_8 \times X_2 \times X_3 \times X_6\]

All other functions related to the tasks in Appendix B.2 are owned independently by either the expert \(e_1\) or the expert \(e_2\), therefore, there is no need to integrate them.

### B.3.2 Behavioral Model Units

Similarly to the case of functional model unit integration, we focus on the task around the boundary of the sales department and the manufacturing department, that is, we focus on the task evaluation. The task-task regarding to evaluation is viewed by the expert \(e_1\) and \(e_2\) as follows.

1. The expert \(e_1\)'s view
   
   The pre-conditioned tasks of evaluation \((Z_{4}^{(1)})\) are inventory check \((Z_{3}^{(1)})\) and credit check \((Z_{3}^{(1)})\). The successor task of it is manufacturing \((Z_{6}^{(1)})\). Therefore, the task-task relationship is denoted by \((\langle Z_{2}^{(1)}, Z_{3}^{(1)}, Z_{4}^{(1)}, (Z_{6}^{(1)}) \rangle)\). The task evaluation is associated with the function \(F_{1}^{(1)}\) and the organization order evaluation \((Y_{5}^{(1)})\), and the behavioral model unit regarding to the task evaluation is denoted by \((\langle Z_{2}^{(1)}, Z_{3}^{(1)}, Z_{4}^{(1)}, Y_{5}^{(1)}, (F_{1}^{(1)}), (Z_{6}^{(1)}) \rangle)\).

2. The expert \(e_2\)'s View
   
   The pre-conditional task of evaluation is sales activity \((Z_{6}^{(2)})\). The successors of evaluation are production \((Z_{2}^{(2)})\), billing \((Z_{3}^{(2)})\), shipping \((Z_{4}^{(2)})\) and
rejection \( Z_{2}^{(2)} \)). The task-task relationship is denoted by
\[
\langle Z_{1}^{(2)}, Z_{2}^{(2)}, Z_{3}^{(2)}, Z_{4}^{(2)}, Z_{5}^{(2)}, \rangle
\]

The task evaluation is associated with the functions \( F_{1}^{(2)}, F_{2}^{(2)}, F_{3}^{(2)} \) and \( F_{4}^{(2)} \), and it also is associated with the organization order evaluation \( Y_{0}^{(2)} \).

Therefore, the behavioral model unit can be denoted by
\[
\langle Z_{2}^{(2)}, Z_{6}^{(2)}, Y_{6}^{(2)}, \langle F_{1}^{(2)}, F_{2}^{(2)}, F_{3}^{(2)}, F_{4}^{(2)} \rangle \langle Z_{2}^{(2)}, Z_{3}^{(2)}, Z_{4}^{(2)}, Z_{6}^{(2)}, \rangle \rangle
\]

In order to integrate those two behavioral model units, we first examine the basic model units composing the behavioral model units. The relationships are:

- \( Z_{2}^{(1)} \cap Z_{1}^{(2)} = Z_{2}^{(1)} = Z_{2} \) (inventory check)
- \( Z_{3}^{(1)} \cap Z_{1}^{(2)} = Z_{3}^{(1)} = Z_{3} \) (credit check)
- \( Z_{4}^{(1)} = Z_{3}^{(2)} = Z_{4} \) (evaluation)
- \( Y_{6}^{(1)} \cap Y_{6}^{(2)} = Y_{6}^{(1)} = Y_{4} \) (order evaluation section)
- \( Z_{5}^{(1)} \cap Z_{2}^{(2)} = Z_{5}^{(2)} = Z_{5} \) (manufacturing)
- \( Z_{6}^{(1)} \cap Z_{3}^{(2)} = Z_{6}^{(2)} = Z_{6} \) (billing)
- \( Z_{7}^{(1)} \cap Z_{4}^{(2)} = Z_{7}^{(2)} = Z_{7} \) (shipping)
- \( Z_{8}^{(1)} \cap Z_{6}^{(2)} = Z_{8}^{(2)} = Z_{8} \) (rejection)

In addition, \( F_{1}^{(1)} \) and \( \langle F_{1}^{(2)}, F_{2}^{(2)}, F_{3}^{(2)}, F_{4}^{(2)} \rangle \) can be integrated into \( \langle F_{1}, F_{2}, F_{3}, F_{4} \rangle \) as stated above.

Therefore, we can integrate the individual behavioral model unit into an enterprise-wide behavioral model unit as:
\[
\langle Z_{2}, Z_{3}, Y_{4}, \langle F_{1}, F_{2}, F_{3}, F_{4} \rangle, \langle Z_{2}^{(2)}, Z_{3}^{(2)}, Z_{4}^{(2)}, Z_{6}^{(2)}, \rangle \rangle
\]

The other behavioral model units can be constructed independently to the domain-experts, since there is no overlap between the tasks which each expert recognizes.
B.4 Constructing the CPN Model

The CPN model constructed from the model units could be represented by the diagram (Figure B.1) and the description table (Table B.4 and Table B.5).

![Diagram of CPN model]

Figure B.1: Sample CPN model
Table B.4: Sample CPN model description - part 1

<table>
<thead>
<tr>
<th>Color sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S = {C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}} )</td>
</tr>
<tr>
<td>( C_1 ) : product name, ( C_2 ) : product number, ( C_3 ) : quantity</td>
</tr>
<tr>
<td>( C_4 ) : customer name, ( C_5 ) : credit number, ( C_6 ) : customer address</td>
</tr>
<tr>
<td>( C_7 ) : check result, ( C_8 ) : order number, ( C_9 ) : warehouse number</td>
</tr>
<tr>
<td>( C_{10} ) : price, ( C_{11} ) : amount</td>
</tr>
<tr>
<td>( D_1 = C_1 \times C_3 \times C_4 \times C_5 \times C_6 ) (order form)</td>
</tr>
<tr>
<td>( D_2 = C_3 \times C_2 \times C_3 ) (availability check form)</td>
</tr>
<tr>
<td>( D_3 = C_2 \times C_5 \times C_4 \times C_6 ) (credit check form)</td>
</tr>
<tr>
<td>( D_4 = C_5 \times C_2 \times C_3 \times C_7 ) (availability check result)</td>
</tr>
<tr>
<td>( D_5 = C_3 \times C_5 \times C_4 \times C_7 \times C_6 ) (credit check result)</td>
</tr>
<tr>
<td>( D_6 = C_2 \times C_2 \times C_7 ) (manufacturing order)</td>
</tr>
<tr>
<td>( D_7 = C_6 \times C_2 \times C_7 ) (rejection information)</td>
</tr>
<tr>
<td>( D_8 = C_6 \times C_4 \times C_6 \times C_5 \times C_{11} ) (order form)</td>
</tr>
<tr>
<td>( D_9 = C_6 \times C_2 \times C_7 \times C_6 \times C_9 ) (shipment information)</td>
</tr>
<tr>
<td>( D_{11} = C_6 \times C_1 \times C_9 ) (shipment log)</td>
</tr>
<tr>
<td>( D_{12} = C_4 \times C_6 \times C_1 ) (rejection letter)</td>
</tr>
<tr>
<td>( D_{13} = C_1 \times C_3 \times C_4 \times C_6 ) (invoice)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Places</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P = {p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}} )</td>
</tr>
<tr>
<td>( p_1 ) : reception office, ( p_2 ) : inventory management, ( p_3 ) : credit management</td>
</tr>
<tr>
<td>( p_4 ) and ( p_5 ) : order evaluation section, ( p_6 ) : production, ( p_7 ) : customer service</td>
</tr>
<tr>
<td>( p_8 ) : accounting section, ( p_9 ) : distribution section</td>
</tr>
<tr>
<td>( p_{10}, p_{11} ) and ( p_{12} ) : outside of the enterprise</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T = {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8} )</td>
</tr>
<tr>
<td>( t_1 ) : order entry, ( t_2 ) : inventory check, ( t_3 ) : credit check</td>
</tr>
<tr>
<td>( t_4 ) : evaluation, ( t_5 ) : manufacturing, ( t_6 ) : rejection</td>
</tr>
<tr>
<td>( t_7 ) : billing, ( t_8 ) : shipping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Color Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C(p_1) = D_1 ), ( C(p_2) = D_2 ), ( C(p_3) = D_3 ),</td>
</tr>
<tr>
<td>( C(p_4) = D_4 ), ( C(p_5) = D_5 ), ( C(p_6) = D_6 ),</td>
</tr>
<tr>
<td>( C(p_7) = D_7 ), ( C(p_8) = D_8 ), ( C(p_9) = D_9 ),</td>
</tr>
<tr>
<td>( C(p_{10}) = D_{10} ), ( C(p_{11}) = D_{11} ), ( C(p_{12}) = D_{12} )</td>
</tr>
</tbody>
</table>
Table B.5: Sample CPN model description - part 2

<table>
<thead>
<tr>
<th>Arc Expression Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E(p_1 \rightarrow t_1) = Id, )</td>
</tr>
<tr>
<td>( E(t_1 \rightarrow p_1) = (h_1(x_1), h_2(x_2), \text{Proj2}) )</td>
</tr>
<tr>
<td>( E(t_1 \rightarrow p_3) = (h_1(x_1), \text{Proj4}, \text{Proj3}, \text{Proj5}) )</td>
</tr>
<tr>
<td>( E(p_2 \rightarrow t_2) = Id, \ E(p_3 \rightarrow t_3) = Id, )</td>
</tr>
<tr>
<td>( E(t_2 \rightarrow p_4) = (\text{Proj1}, \text{Proj2}, h_5(x_1, x_2)) )</td>
</tr>
<tr>
<td>( E(t_3 \rightarrow p_5) = (\text{Proj1}, \text{Proj2}, h_4(x_2)) )</td>
</tr>
<tr>
<td>( E(p_4 \rightarrow t_4) = Id, \ E(p_5 \rightarrow t_4) = Id, )</td>
</tr>
<tr>
<td>( E(t_4 \rightarrow p_6) = (\text{Proj1}, \text{Proj2}, \text{Proj3}) )</td>
</tr>
<tr>
<td>( \text{If } u(\in C_7 \text{ in } D_4) = \bot \text{ and } v(\in C_7 \text{ in } D_5) = T )</td>
</tr>
<tr>
<td>( E(t_4 \rightarrow p_7) = (\text{Proj1}, \text{Proj2}, \text{Proj4}) )</td>
</tr>
<tr>
<td>( \text{If } v(\in C_7 \text{ in } D_5) = \bot )</td>
</tr>
<tr>
<td>( E(t_4 \rightarrow p_8) = (\text{Proj1}, \text{Proj8}, \text{Proj9}, \text{Proj6}, h_5(x_2, x_3)) )</td>
</tr>
<tr>
<td>( \text{If } u(\in C_7 \text{ in } D_4) = T \text{ and } v(\in C_7 \text{ in } D_5) = T )</td>
</tr>
<tr>
<td>( E(t_4 \rightarrow p_9) = (\text{Proj1}, \text{Proj2}, \text{Proj3}, \text{Proj4}, h_6(x_2)) )</td>
</tr>
<tr>
<td>( \text{If } u(\in C_7 \text{ in } D_4) = T \text{ and } v(\in C_7 \text{ in } D_5) = T )</td>
</tr>
<tr>
<td>( E(p_6 \rightarrow t_9) = Id, \ E(p_7 \rightarrow t_9) = Id, )</td>
</tr>
<tr>
<td>( E(p_8 \rightarrow t_9) = Id, \ E(p_9 \rightarrow t_9) = Id, )</td>
</tr>
<tr>
<td>( E(t_6 \rightarrow p_9) = (h_7(x_1), h_8(x_1), h_9(x_1), h_{10}(x_1)) )</td>
</tr>
<tr>
<td>( E(t_9 \rightarrow p_9) = (\text{Proj1}, \text{Proj2}, \text{Proj3}, \text{T}, h_5(x_2)) )</td>
</tr>
<tr>
<td>( E(t_6 \rightarrow p_{10}) = (h_{11}(x_1), h_{12}(x_1), h_{13}(x_1)) )</td>
</tr>
<tr>
<td>( E(t_9 \rightarrow p_{10}) = (h_{14}(x_1), h_{15}(x_1), h_{16}(x_1), \text{Proj4}, \text{Proj3}, \text{Proj5}) )</td>
</tr>
<tr>
<td>( E(t_8 \rightarrow p_{11}) = (\text{Proj1}, h_{14}(x_1), \text{Proj5}) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initialization Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I(p_1) = { x_1, x_2, x_3, x_4, x_5 } \in D_1 )</td>
</tr>
<tr>
<td>( I(p_i) = \emptyset \ (i \neq 1) )</td>
</tr>
</tbody>
</table>
Component Selection

In order to select adaptable components to the requirements, we first have to transform the CPN model into Σ algebra. The transformation is performed for each transition one after another. We choose transition \( t_4 \) (evaluation) which performs most complicated function. The other transitions can be transformed in the similar ways, but more easily.

First, we examine the carriers (or sorts) associated with this transition \( t_4 \). The input arcs to \( t_4 \) are \((p_4 \rightarrow t_4)\) and \((p_5 \rightarrow t_4)\), which have the arc functions \( E(p_4 \rightarrow t_4) = I_d \) and \( E(p_5 \rightarrow t_4) = I_d \) respectively. Those \( I_d \) functions remove one token from the places \( p_4 \) and \( p_5 \) which are associated with the token colors \( D_4 \) and \( D_5 \) respectively, then provide them to the transition \( t_4 \) on as-is base. Therefore, the carriers composing the input to the \( S \)-sorted functions associated with the transition \( t_4 \) should be:

\[
D_4 \times D_5 = C_8 \times C_3 \times C_3 \times C_7 \times C_8 \times C_5 \times C_7 \times C_4 \times C_6
\]

Each argument for those functions would be in the form of:

\[
(x_1, \ldots, x_9) \in D_4 \times D_5
\]

However, there are duplicated color sets in the input, that is, \( C_8 \) appears twice in it. In addition, those color sets initially come from the same order form \( D_1 \), therefore they must be identical each other. \( C_7 \) (check result) which appears twice in \( D_4 \times D_5 \) is not identical each other, since they are the results of different checks (inventory check and credit check).

This leads us to make such new set of carriers as:

\[
D = C_8 \times C_3 \times C_3 \times C_7 \times C_7 \times C_5 \times C_4 \times C_6
\]

and such the new argument as:

\[
(x_1, \ldots, x_8) \in D
\]

As for data transformation, there are four output functions:

\[
E(t_4 \rightarrow p_6)
\]
\[
E(t_4 \rightarrow p_7)
\]
\[
E(t_4 \rightarrow p_8)
\]
\[
E(t_4 \rightarrow p_9)
\]

and we must consider all of those functions independently.

1. \( E(t_4 \rightarrow p_8) \)

As stated in Appendix C.3, this function can be denoted by:

\[
E(t_4 \rightarrow p_8) = (Proj_1, Proj_2, Proj_3)
\]

for the input token color

\[
D_4 \times D_5.
\]

This function can be denoted by:

\[
f_1(x_1, \ldots, x_8) = (x_1, x_2, x_3) = y_1
\]

where

101
\((x_1, \ldots, x_8) \in D\) and 
\(y_1 = (x_1, x_2, x_3) \in D_6 \quad (= C_8 \times C_2 \times C_3)\)

2. \(E(t_4 \rightarrow p_7)\)
\(E(t_4 \rightarrow p_7) = (\text{Proj} 1, \text{Proj} 2, \text{Proj} 4)\)
as stated in Appendix B.3. Similarly to \(E(t_4 \rightarrow p_6)\), we can conclude 
\(f_2(x_1, \ldots, x_8) = (x_1, x_2, x_4) = y_2\)
where 
\((x_1, \ldots, x_8) \in D\) and 
\(y_2 = (x_1, x_2, x_4) \in D_7\)

3. \(E(t_4 \rightarrow p_8)\)
\(E(t_4 \rightarrow p_8) = (\text{Proj} 1, \text{Proj} 7, \text{Proj} 8, \text{Proj} 6, h_5(x_2, x_3))\)
where 
\(x_2 \in C_2, x_3 \in C_3\) and 
\(h_5(x_2, x_3) \in C_{11}\).
The function \(h_5\) calculates the amount from the ordered product number and 
the ordered quantity. The S-sorted function derived from this arc function 
can be denoted by:
\(f_3(x_1, \ldots, x_8) = (x_1, x_7, x_8, x_6, h_5(x_2, x_3)) = y_3\)
where 
\((x_1, \ldots, x_8) \in D\) and 
\(y_3 = (x_1, x_7, x_8, x_6, h_5(x_2, x_3)) \in D_8\)

4. \(E(t_4 \rightarrow p_9)\)
\(E(t_4 \rightarrow p_9) = (\text{Proj} 1, \text{Proj} 2, \text{Proj} 3, \text{Proj} 4, h_6(x_2))\)
where 
\(x_2 \in C_2\) and 
\(h_6(x_2) \in C_9\).
The function \(h_6\) transforms the ordered product number into a warehouse 
number, or in other words, finds a warehouse which stocks the ordered products. The S-sorted function derived from this arc function can be denoted by:
\(f_4(x_1, \ldots, x_8) = (x_1, x_2, x_3, x_4, h_6(x_2)) = y_4\)
where 
\((x_1, \ldots, x_8) \in D\) and 
\(y_4 = (x_1, x_2, x_3, x_4, h_6(x_2)) \in D_9\)

From the above discussion, the carriers associated with the transition \(t_4\) are:
\(A_{\sigma_1} = C_8, A_{\sigma_2} = C_2, A_{\sigma_3} = C_3,\)
\(A_{\sigma_4} = C_7, A_{\sigma_5} = C_5, A_{\sigma_6} = C_4,\)
\(A_{\sigma_7} = C_6, A_{\sigma_8} = D_6, A_{\sigma_9} = D_7,\)
\(A_{\sigma_{10}} = D_8, A_{\sigma_{11}} = D_9,\)
and the functions associated with this transition are:

102
\( (f_1)_{\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6\sigma_7\sigma_8} \\
(f_2)_{\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6\sigma_7\sigma_9} \\
(f_3)_{\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6\sigma_7\sigma_{10}} \\
(f_4)_{\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6\sigma_7\sigma_{11}} \).

The next step is to compose \( \Sigma \) algebra from available components. We assume we found the following components in some component sources (e.g. legacy systems).

- \( \gamma(x_1) = (x_1) = Id \) (for any \( x_1 \))
- \( \gamma_2(x_1, x_2) = (p_1(x_1, x_2)) \) where \( x_1 \in C_2' \) and \( x_2 \in C_3 \).
  A function \( p_1 \) calculates the amount for some kinds of products.
- \( \gamma_3(x_1, x_2) = (p_2(x_1, x_2)) \) where \( x_1 \in C_2'' \) and \( x_2 \in C_3 \).
  A function \( p_2 \) calculates the amount for the other kinds of products. Let \( C_2' \) and \( C_2'' \) satisfy \( C_2 \subseteq (C_2' \cup C_2'') \).
- \( \gamma_4(x_1) = (w(x_1)) \) where \( x_1 \in C_2 \) and \( w(x_1) \in C_9 \).

In such case, we can define the \( \Sigma \) algebra for those components as:

1. \( g_1(x_1, \ldots, x_8) = (\gamma_1(x_1), \gamma_1(x_2), \gamma_1(x_3)) = y_1 \) where \( (x_1, \ldots, x_8) \in D \) and \( y_1 = (\gamma_1(x_1), \gamma_1(x_2), \gamma_1(x_3)) \in D_6 \).
2. \( g_2(x_1, \ldots, x_8) = (\gamma_1(x_1), \gamma_1(x_2), \gamma_1(x_4)) = y_2 \) where \( (x_1, \ldots, x_8) \in D \) and \( y_2 = (\gamma_1(x_1), \gamma_1(x_2), \gamma_1(x_4)) \in D_7 \).
3. \( g_3(x_1, \ldots, x_8) = (\gamma_1(x_1), \gamma_1(x_7), \gamma_1(x_8), \gamma_1(x_9), \gamma_2(x_2, x_3)) = y_3 \) where \( (x_1, \ldots, x_8) \in D, x_2 \in C_2' \cap C_2 \) and \( y_3 = (\gamma_1(x_1), \gamma_1(x_7), \gamma_1(x_8), \gamma_1(x_9), \gamma_2(x_2, x_3)) \in D_8 \).
4. \( g_4(x_1, \ldots, x_8) = (\gamma_1(x_1), \gamma_1(x_6), \gamma_1(x_7), \gamma_1(x_8), \gamma_3(x_2, x_3)) = y_4 \) where \( (x_1, \ldots, x_8) \in D, x_2 \in C_2'' \cap C_2 \) and \( y_4 = (\gamma_1(x_1), \gamma_1(x_6), \gamma_1(x_7), \gamma_1(x_8), \gamma_3(x_2, x_3)) \in D_9 \).
5. \( g_5(x_1, \ldots, x_8) = (\gamma_1(x_1), \gamma_1(x_2), \gamma_1(x_3), \gamma_1(x_4), \gamma_4(x_2)) = y_5 \) where \( (x_1, \ldots, x_8) \in D \) and \( y_5 = (\gamma_1(x_1), \gamma_1(x_2), \gamma_1(x_3), \gamma_1(x_4), \gamma_4(x_2)) \in D_9 \)
The carriers derived from the above components are:

\[ B_{\sigma_1} = C_8, \quad B_{\sigma_2} = C_2, \quad B_{\sigma_3} = C'_2 \cap C_2, \]
\[ B_{\sigma_4} = C''_2 \cap C_2, \quad B_{\sigma_6} = C_3, \quad B_{\sigma_8} = C_7, \]
\[ B_{\sigma_9} = C_5, \quad B_{\sigma_9} = C_4, \quad B_{\sigma_9} = C_6, \]
\[ B_{\sigma_{10}} = D_6, \quad B_{\sigma_{11}} = D_7, \quad B_{\sigma_{12}} = D_8, \]
\[ B_{\sigma_{13}} = D_9 \]

The functions associated with the above components are:

\[ (g_1)_{\sigma_1 \sigma_2 \sigma_3 \sigma_4 \sigma_5 \sigma_6 \sigma_7 \sigma_8 \sigma_9 \sigma_{10}} \]
\[ (g_2)_{\sigma_2 \sigma_3 \sigma_4 \sigma_5 \sigma_6 \sigma_7 \sigma_8 \sigma_9 \sigma_{11}} \]
\[ (g_3)_{\sigma_3 \sigma_4 \sigma_5 \sigma_6 \sigma_7 \sigma_8 \sigma_9 \sigma_{12}} \]
\[ (g_4)_{\sigma_4 \sigma_5 \sigma_6 \sigma_7 \sigma_8 \sigma_9 \sigma_{12}} \]
\[ (g_5)_{\sigma_4 \sigma_5 \sigma_6 \sigma_7 \sigma_8 \sigma_9 \sigma_{13}} \]

In order to adjust the sorts \( f_3 \) and \( (g_3, g_4) \), we define the new carriers for \( f_3 \) according to the discussion in section 4.2.2. Those new carriers are:

\[ A_{\sigma''_2} = C''_2 \cap C_2 \quad \text{and} \quad A_{\sigma''_6} = C''_2 \cap C_2 \]

and the function \( f_3 \) is divided into two restrictions (to \( A_{\sigma''_2} \) and \( A_{\sigma''_6} \)) as:

\[ (f_3)_{\sigma_1 \sigma_2 \sigma_3 \ldots \sigma_7 \sigma_{10}} \quad \text{and} \quad (f_3)_{\sigma_1 \sigma_2 \sigma_3 \ldots \sigma_7 \sigma_{10}} \]

Evidently, the \( \Sigma \) algebra \((A, F)\) is \( \Sigma \) homomorphic to \( \Sigma \) algebra \((B, G)\) where

\[ A = \{ A_{\sigma_1}, A_{\sigma_2}, A_{\sigma_3}, A_{\sigma_4}, A_{\sigma_5}, A_{\sigma_6}, A_{\sigma_7}, A_{\sigma_8}, A_{\sigma_9}, A_{\sigma_{10}}, A_{\sigma_{11}} \} \]
\[ F = \{ f_1, f_2, f_3, f_4, f_5 \} \]
\[ B = \{ B_{\sigma_1}, B_{\sigma_2}, B_{\sigma_3}, B_{\sigma_4}, B_{\sigma_5}, B_{\sigma_6}, B_{\sigma_7}, B_{\sigma_8}, B_{\sigma_9}, B_{\sigma_{10}}, B_{\sigma_{11}}, B_{\sigma_{12}}, B_{\sigma_{13}} \} \]
\[ G = \{ g_1, g_2, g_3, g_4, g_5 \} \]

The other transitions in the CPN model are more simple, and we can identify the adaptable components in the similar way.
B.6 Extracting Control Structure of the CPN Model

In order to show how the proposed approach extracts the control structure of the CPN model, we choose the transition $t_4$ (evaluation), since it has the most complex control structure in the sample application.

Let the inputs to the transition $t_4$ be

$$(u_1, u_2, u_3, u_4) \in D_4 \ (= C_8 \times C_2 \times C_3 \times C_7)$$

and

$$(v_1, v_2, v_3, v_4, v_5) \in D_5 \ (C_9 \times C_5 \times C_7 \times C_4 \times C_6).$$

According to the description table of the CPN model in Appendix B.4, we can derive the decision table in the form of Table B.6. The columns $C_8, C_3, C_8, C_5,$

<table>
<thead>
<tr>
<th>$C_8$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_7$</th>
<th>$C_8$</th>
<th>$C_5$</th>
<th>$C_7$</th>
<th>$C_4$</th>
<th>$C_6$</th>
<th>$\Psi$</th>
<th>$n$</th>
<th>$\Pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>⊥</td>
<td>-</td>
<td>T</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$g_1$</td>
<td>$D_6$</td>
<td>$p_6$</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>⊥</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$g_2$</td>
<td>$D_7$</td>
<td>$p_7$</td>
</tr>
<tr>
<td>-</td>
<td>$C_2'$</td>
<td>-</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$g_3$</td>
<td>$D_8$</td>
<td>$p_8$</td>
</tr>
<tr>
<td>-</td>
<td>$C_2''$</td>
<td>-</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$g_4$</td>
<td>$D_8$</td>
<td>$p_8$</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>T</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$g_5$</td>
<td>$D_9$</td>
<td>$p_9$</td>
</tr>
</tbody>
</table>

$\leftarrow C(p4) \rightarrow \leftarrow C(p5) \rightarrow$

$C_4$, and $C_0$ in the conditions are dispensable, and we can remove one of them. The above decision table could be a source of some middlewares.

The other transitions have much more simple control structure, and we can construct the similar decision tables more easily.
B.7 Adaptability Evaluation

We assume the software system that is implemented from the requirements as shown in Figure B.2. The differences between this model and the requirements model are as follows.

1. The transition inventory check in the requirements model, which is denoted by $t_4$, is performed by two different transitions in the software system model, that is, $t_2$ (availability check) and $t_3$ (inventory check) according to the type of the ordered product.

2. The transition rejection in the requirements model, which is denoted by $t_9$ in the requirements model, is performed serially by the different two transitions $t_7$ (rejection record) and $t_{11}$ (rejection letter).
3. The transition shipping, which is denoted by $t_8$ in the requirements model, is performed parallel by the two different transitions $t_9$ (invoicing) and $t_{10}$ (shipping).

B.7.1 Static Adaptability Evaluation

Static adaptability, or functional adaptability, can be accomplished in the same way as shown in Appendix B.5. We can conclude whether a subnet in the requirements CPN model is $\Sigma$ homomorphic to a subnet in the software CPN model, by transforming them into $\Sigma$ algebras.

B.7.2 Dynamic Adaptability Evaluation

Since the control structures of the CPN models of the requirements and the software system, which is currently considered, are identical except the transitions $t_2$, $t_6$ and $t_8$ in the requirements model, we only have to focus on those transitions.

We assume S-sorted functions associated with those transitions are shown in Table B.7. In addition, the subnets $SN_R(\{t_2\})$, $SN_R(\{t_6\})$ and $SN_R(\{t_8\})$ are assumed to be functionally equivalent (or $\Sigma$ homomorphic) to the subnets $SN_L(\{t_1, t_2\})$, $SN_L(\{t_7, t_11\})$ and $SN_L(\{t_9, t_{10}\})$ respectively, where $SN_R(\{t_i\})$ represents a subnet including a set of the transitions $\{t_i\}$ in the requirements model, while $SN_L(\{t_j\})$ represents a subnet including a set of transitions $\{t_j\}$ in the software system model.

Table B.7: Transitions and S-sorted functions

<table>
<thead>
<tr>
<th>Requirements model</th>
<th>Software system model</th>
</tr>
</thead>
<tbody>
<tr>
<td>transition</td>
<td>S-sorted function</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$f_{21}$</td>
</tr>
<tr>
<td></td>
<td>$f_{22}$</td>
</tr>
<tr>
<td></td>
<td>$f_{23}$</td>
</tr>
<tr>
<td>$t_6$</td>
<td>$f_{01}$</td>
</tr>
<tr>
<td></td>
<td>$f_{02}$</td>
</tr>
<tr>
<td></td>
<td>$f_{03}$</td>
</tr>
<tr>
<td>$t_8$</td>
<td>$f_{81}$</td>
</tr>
<tr>
<td></td>
<td>$f_{82}$</td>
</tr>
</tbody>
</table>

Supposing that the correspondences between the S-sorted functions are as follows:

$f_{21} \leftrightarrow g_{21}, f_{22} \leftrightarrow g_{22}, f_{23} \leftrightarrow g_{31}$,
\[ f_{61} \leftrightarrow g_{71}, \quad f_{62} \leftrightarrow g_{111}, \quad f_{63} \leftrightarrow g_{112} \]

\[ f_{81} \leftrightarrow g_{91}, \quad f_{82} \leftrightarrow g_{101} \]

we can define the common action names in CCS as follows.

Action name \( a_1 \) \( \leftrightarrow (f_{21} \text{ and } g_{21}) \)

Action name \( a_2 \) \( \leftrightarrow (f_{22} \text{ and } g_{22}) \)

Action name \( a_3 \) \( \leftrightarrow (f_{23} \text{ and } g_{31}) \)

Action name \( b_1 \) \( \leftrightarrow (f_{31} \text{ and } g_{71}) \)

Action name \( b_2 \) \( \leftrightarrow (f_{32} \text{ and } g_{111}) \)

Action name \( b_3 \) \( \leftrightarrow (f_{33} \text{ and } g_{112}) \)

Action name \( c_1 \) \( \leftrightarrow (f_{31} \text{ and } g_{91}) \)

Action name \( c_2 \) \( \leftrightarrow (f_{32} \text{ and } g_{101}) \)

The CCS expressions of the above subnets by those actions are as follows:

\[ SN_B(\{t_5\}) = a_{1,0} + a_{2,0} + a_{3,0} \]

\[ SN_B(\{t_6\}) = b_{1,0} + b_{2,0} + b_{3,0} \]

\[ SN_B(\{t_7\}) = c_{1,0} + c_{2,0} \]

\[ SN_L(\{t_2, t_3\}) = (a_{1,0} + a_{2,0})|a_{3,0} \]

\[ SN_L(\{t_7, t_{11}\}) = b_{1,0}(b_{2,0} + b_{3,0}) \]

\[ SN_L(\{t_9, t_{10}\}) = c_{1,0}|c_{2,0} \]

By examining the action transition chart in Figure B.3, we can conclude that:

- \( SN_B(\{t_2\}) \) and \( SN_L(\{t_2, t_3\}) \) are dynamically equivalent (strong bisimulation).

- \( SN_B(\{t_4\}) \) and \( SN_L(\{t_7, t_{11}\}) \) are NOT dynamically equivalent (neither strong bisimulation nor observation congruence).

- \( SN_B(\{t_8\}) \) and \( SN_L(\{t_9, t_{10}\}) \) are dynamically equivalent (strong bisimulation).
Figure B.3: Action transition chart
B.7.3 Data Adaptability Evaluation

The concept of data adaptability is only concerned with software systems, since it assumes transaction processing and database systems. We suppose the following conditions in order to evaluate data adaptability.

- The transitions $t_8$ (billing), $t_9$ (invoicing) and $t_{10}$ (shipping) reside in the same sphere, that is, they must be COMMITed or ABORTed together.

- The transition $t_8$ (billing) updates the following data.
  "product name", "customer name", "quantity", "credit number", "customer address", "amount"

- The transition $t_9$ (invoicing) updates the following data
  "product name", "quantity", "customer name", "customer address"

- The transition $t_{10}$ (shipping) updates the following data
  "order number", "product name", "warehouse number"

All of the above data are assumed to be created. In such case, we can regard those transitions which are implemented as transactions as sequences of functions. Those sequences can be denoted by:

- billing
  $t_8 = f_1 f_2 f_3 f_4 f_5$ where
  $f_1 : \emptyset \rightarrow a_1 \text{ (product name)}$
  $f_2 : \emptyset \rightarrow a_2 \text{ (customer name)}$
  $f_3 : \emptyset \rightarrow a_3 \text{ (quantity)}$
  $f_4 : \emptyset \rightarrow a_4 \text{ (credit number)}$
  $f_5 : \emptyset \rightarrow a_5 \text{ (amount)}$

- invoicing
  $t_9 = g_1 g_2 g_3 g_4$ where
  $g_1 : \emptyset \rightarrow b_1 \text{ (product name)}$
  $g_2 : \emptyset \rightarrow b_2 \text{ (quantity)}$
  $g_3 : \emptyset \rightarrow b_3 \text{ (customer name)}$
  $g_4 : \emptyset \rightarrow b_4 \text{ (customer address)}$

- shipping
  $t_{10} = h_1 h_2 h_3$ where
  $h_1 : \emptyset \rightarrow c_1 \text{ (order number)}$
  $h_2 : \emptyset \rightarrow c_2 \text{ (product name)}$

110
\( h_3 : \emptyset \longrightarrow c_3 \) (warehouse number)

The above functions \( f_i, g_i \) and \( h_i \) have their inverse functions \( f_i^{-1}, g_i^{-1} \) and \( h_i^{-1} \), which are often referred to as deletion.

In addition, one of the above three transitions fails, e.g. invoicing fails, we can remove the update of system state performed by billing and shipping, by making such compensation transitions as:

- \((t_6)_{CT} = f_5^{-1}f_4^{-1}f_3^{-1}f_2^{-1}f_1^{-1}\)
- \((t_{10})_{CT} = h_3^{-1}h_2^{-1}h_1^{-1}\)

Those compensations would make the system consistent, since the update of the system state performed by this sphere only affect the part of the state which is related to the order that this sphere (or sphere instance, more strictly speaking) is currently processing.

Therefore, the sample software system can be regarded to have data adaptability.