A Feature Space-based Business Model Quality Evaluation Method

Zhongjie Wang, Xiaofei Xu and Dechen Zhan
School of Computer Science and Technology, Harbin Institute of Technology, China

It is inevitable that there are more or less diversities between business models created by different modelers, thus it is necessary to evaluate and compare them quantitatively to help decision makers discover whose models are pressing much closer to customer requirements. In this paper, a new approach for business model quality evaluation is presented. In order to deal with business models described by varied modeling languages, a unified and extended feature modeling technique is adopted. Quality of a user-created model is then measured from two views, “completeness” and “soundness”, by assessing the distance between the user model and the standard model with the help of feature space as the tools. An example is briefly shown along with each concept and algorithm for illustration. Benefits and deficiencies of our method are briefly concluded for future works.

Keywords: business model, feature space, model quality evaluation, completeness, soundness

1. Introduction

“Business modeling” refers to the documentation of a business system using a combination of text, graphical or formal notations, to clearly describe reality and understand requirements accordingly, to compute, reason, verify, design and develop software systems based on these business models. A consensus has been reached that business modeling plays an extremely important role in the lifecycle of a software system (especially those complex enterprise software and applications, e.g., ERP, CRM, SCM) to support computer integrated manufacturing management in modern enterprises.

Since “the achievement of software requirement quality is the first step towards software quality” [1] and software requirements are primarily described by business models, if a business model cannot exhibit those realistic requirements correctly and completely, then it is destined that the final software systems are incorrect and incomplete likewise.

However, business modeling has been considered as a process which requires rich knowledge and experiences about the business, i.e., if a modeler had no deep understandings on real-life business, the models he creates would be much more inferior than the models created by domain experts. Therefore, for a decision-maker in a CIM project, it is necessary to determine, when multiple modelers individually create their models for the same business requirements, whose models are better, i.e., to perform “business model quality evaluation”.

In previous literatures, researches on model quality evaluation are mainly classified into the following four aspects:

- Evaluating the capacity of business modeling methodologies and languages, i.e., to judge whether a specific modeling approach and its notations could satisfy those common or specific modeling requirements. For example, [2] proposes a conceptual framework for comparing various reference models based on an elaboration of a linguistics-based classification approach, [3] evaluates the capacity of UML and [4] evaluates UML interaction diagram, with detailed results to show the insufficiencies of UML and interaction diagrams.

- Evaluating syntax soundness, consistency and performance of a model, i.e., to judge whether there are syntax errors in a model. For example, [1] presents an automatic tool for the analysis of natural language
requirements documents to be used for detecting and removing defects that could cause errors in the transition to formal models. For UML-type models, a tool named DesignAdvisor was developed specifically in [5] to analyze and measure the “goodness” of large, complex UML models. [6] adopts the scenarios concept to evaluate the description of requirements. Aiming at data models, [7] presents a set of quality factors (e.g., Simplicity, Completeness, Correctness, Integrity, Flexibility, etc.) and the corresponding proposed metrics.

- **Evaluating semantics soundness and performance of a model**, e.g., for workflow models, identifying exceptional paths [8], verifying authorization reasonability [9], checking whether there are invalid paths [10] or whether the synchronization feature might be preserved at runtime [11], etc. In these approaches, semantics performance is usually verified by simulation, and semantics soundness is usually verified by rule-based logical reasoning.

- **Evaluating runtime feasibility of a model**, e.g., estimating the cost and benefit of an e-business model from an economic value perspective [12], judging whether a supply chain model has sufficiently quick response time and low operation cost [13], etc.

In above four aspects, the first one deviates from our subject (we focus on the quality of models instead of the modeling methodologies), the second one emphasizes syntax forms of the models, while the third and fourth ones focus on semantics quality.

Most of these approaches usually aim at some specific quality features of a specific model and the assessment results just reflect some deficiencies in the model. However, many quality factors could only be obtained after comparison (with a standard model), e.g., completeness of a model; in addition, if there are no flaws on syntax, semantics and performance in two models, how to judge which one is better? Unfortunately, most of the above approaches have neglected such situation.

Considering from another viewpoint, we may see that the above methods have provided different quality indicators of whose metrics is quite related to the concrete forms of the models to be qualified, which makes their application scopes limited and there is a lack of a uniform evaluation method for any forms of models.

Firstly, we consider the issue “model comparison”. Currently, most popular modeling approaches have already discarded the “from scratch” modeling way; on the contrary, after long-term accumulation domain experts have gathered and summarized those general knowledge in each business domain and gathered rich domain models; when a concrete model is required to be established, modelers may directly start from the domain model to carry on some revisions according to specific requirements. This approach is shown in Figure 1.

A model may be considered as a set of model elements and relationships between them, the latter of which may be considered and described as a set of rules or constraints on the former. Evaluating the quality of a user business model \(U\) is comparing it with the domain business model \(D\) and specific requirements \(R\) to see whether \(U\) contains necessary model elements in \(D\) and \(R\) (completeness) and whether the elements in \(U\) satisfy the constraints prescribed in \(D\) and \(R\) (soundness), in a word, it is quantitative calculating the distance between \(U\) and \(D\cap R\) a standard model \(S\).

Secondly, we consider the issue “uniform model evaluation method”. Because there are tens of modeling languages with different notations and forms, and in order to design a general model evaluation method, there must be a uniform model format. In this paper, we try to transform various business model styles into the form of feature space, and then adopt some concepts/methods in traditional feature modeling as the tool to discuss where the distance between \(U\) and \(S\) might exist to propose the concrete metrics for completeness and soundness for quality evaluation.
The rest of this paper is arranged as follows. In Section 2, extended feature modeling method is briefly introduced. In Section 3, feature space-based model quality evaluation method is elaborately proposed with a practical case. The conclusion is given in Section 4.

2. Feature-oriented Business Model

As mentioned above, a business model is composed of a set of business elements and a set of relationships between these elements. Relationships between elements are classified into three types, i.e., composition, generalization and dependency, in which composition organizes business models as hierarchical structure, generalization makes models reusable in different application scenarios and dependency describes semantics associations between the elements.

The reason why we import and extend the feature-oriented modeling techniques [14][15][16] is that it has the ability to describe the three relationships in models.

2.1. Feature and Feature Space

*Feature* is an ontology that is used to describe the knowledge of external world, and is represented as “Terms” or “Concepts” used to describe the services supplied by a specific business domain, e.g., business processes, business activities, business objects, states attributes and operations, etc.

Features are hierarchical, i.e., there is a composition relationship, or “whole-part association”, between parent and child features. According to this property, related features are organized as a multi-layer feature space, denoted as $\Omega = (F, D)$, in which $F$ is feature set and $D$ is the set of feature dependencies between features in $F$.

$\Omega$ is usually represented as the form of a feature tree, where there is one and only one root feature $f_{root}$ and $child(f)$, $parent(f)$, $ancestor(f)$, $descendant(f)$ and $sibling(f)$ are used to denote $f$’s child feature set, parent feature set, ancestor feature set, descendant feature set and sibling feature set, respectively.

A feature item is an instance of a feature, describes the feature’s one possible value under a given business environment, and reflects the variability of the feature. Let $dom(f)$ denote the set of all feature items of feature $f$, which is called the “domain” of $f$. For $\forall \tau \in dom(f)$, $\tau$ is called a value of $f$. A feature is an abstraction of all its feature items, and there exists a generalization-instantiation association (GIA) between a feature and its items.

The instantiation of a feature $f$ is the process of choosing a proper feature item from $dom(f)$ for $f$ to satisfy a specific semantics context, denoted as $\tau(f)$. Similarly, by instantiating a feature vector $Y = (f_1, f_2, \ldots, f_n)$, we can get an instance of $Y$, denoted as $\tau(Y) = (\tau(f_1), \tau(f_2), \ldots, \tau(f_n))$, in which $\tau(f_i) \in dom(f_i)$, $1 \leq i \leq n$. If we instantiate each feature $f_1, f_2, \ldots, f_n$ in $\Omega$, we can get one of $\Omega$’s instance, denoted as $\tau(\Omega)$. All the instances of $\Omega$ constitute $\Omega$’s instance set $T(\Omega)$. It is easy to know that $T(\Omega) \subseteq dom(f_1) \times dom(f_2) \times \ldots \times dom(f_n)$, and $\forall t \in T(\Omega)$, $t = (t_1, t_2, \ldots, t_n)$, in which $t_i \in dom(f_i)$ is the projection of $t$ on $f_i$, also denoted as $t[f_i]$. $t$’s projection on feature set $X$ is denoted as $\tau(X)$.

Figure 2 shows the feature space of a business model for “Account Receivable Management” in Enterprise Resource Planning (ERP) domain, Table 1 lists all the features and feature items in this model and Table 2 shows some of feature dependencies contained in the model. Due to limited space, here we only show those functional features (e.g., business process and business activities) and “execution order” relations between them.

2.2. Feature Dependency

In a feature space $\Omega = (F, D)$, $D$ is the dependency set between features in $F$. As presented in our previous publications [16][17], a feature dependency (FD) is defined as the relationships between two related features in a feature space. According to the structural and semantic relationships between two features, FD can be classified into five types:

- Whole-part Association (WPA)
- Feature Integrity Dependency (FID)
- Feature Value Dependency (FVD)
- Feature Multi-value Dependency (FMVD)
- Feature Semantics Dependency (FSD)
WPA is the simplest FD and represents fixed composition relationship between child and parent features and explicitly behaves as the parent-child structure.

FID is a dependency between a feature \( f \) and its child feature set \( Y \) (i.e., \( Y = \text{child}(f) \)), denoted as \( f \rightarrow Y \). It describes whether each feature in \( Y \) would be selected as an essential part of \( f \)'s instance when \( f \) is instantiated. According to the number of features that are selected for \( f \)'s instantiation, there are four types of FIDs, i.e., mandatory FID, optional FID, single-selection FID and multiple-selection FID, denoted as \( f^M \rightarrow g \), \( f^O \rightarrow g \), \( f^S \rightarrow Y \), \( f^T \rightarrow Y \) respectively.

- Mandatory FID \( (f^M \rightarrow g) \): no matter which instance \( f \) is instantiated, \( g \) is always necessary;
- Optional FID \( (f^O \rightarrow g) \): \( g \) is necessary for some instances of \( f \), however, for other instances of \( f \), it is unnecessary;
- Single instance FID \( (f^S \rightarrow Y) \): for each instance of \( f \), there is one and only one feature for which \( g \) is necessary;
- Multiple instance FID \( (f^T \rightarrow Y) \): for each instance of \( f \), there are possibly multiple features for which \( g \) is necessary;

For example, in Figure 2, from the above definitions we have (1) \( f_1^M \rightarrow \{f_{12}, f_{13}, f_{15}, f_{19}\} \) because \( f_{12}, f_{13}, f_{15}, f_{19} \) are all necessary for \( \tau_{11}, \tau_{12} \) and \( \tau_{13} \); (2) \( f_1^O \rightarrow \{f_{11}, f_{14}, f_{16}\} \) because \( f_{11}, f_{14}, f_{16} \) are only necessary for \( \tau_{11}, \tau_{12} \) and \( \tau_{13} \) respectively; (3) \( f_1^S \rightarrow \{f_{17}, f_{18}\} \) because only one of \( f_{17} \) and \( f_{18} \) may be required by \( \tau_{13} \) at the same time; (4) \( f_3^T \rightarrow \{f_{33}, f_{34}, f_{35}\} \).

FID can be regarded as the dependencies between the “Values” of parent feature and the “Type” of its child features, therefore it is called “Value-type” dependency, i.e., one feature item of parent feature determines which of child features are the essential parts of the parent feature. FID depicts the structural integrity relationships between parent and child features.

FVD and FMVD both depict the restrictions that must be satisfied when different features are instantiated, therefore they are called “Value-value” dependency, i.e., the instances of one feature set uniquely or multiply determine the instances of another feature set. They generally appear between sibling features.

We use \( X \rightarrow Y \) to denote the FVD between \( X \) and \( Y \) and call “\( Y \) feature value dependent on \( X \)”. Similarly, \( X \rightarrow \rightarrow Y \) is adopted to denote the FMVD between \( X \) and \( Y \).

For example, in Figure 2, when \( f_1 \) is instantiated as \( \tau_{11} \) or \( \tau_{12} \), \( f_2 \) must have the value \( \tau_{21} \), and when \( f_1 \) is instantiated as \( \tau_{13} \), \( f_2 \) must have the value \( \tau_{22} \), therefore we have \( f_1 \rightarrow f_2 \).

Similar to functional dependency in relational model, FVD and FMVD also have the characteristics of Reflexivity, Augmentation, Transitivity, Pseudotransitivity, Union and Decomposition, etc. According to Armstrong Axiom [18], we can get a feature set \( X \)'s closure on FD set \( D \), denoted as \( X^+ \), which contains all the features that directly or indirectly depend on features in \( X \).

FSD refers to the semantics association between features. It is irrespective with the values of features, and it represents constraints between the type of related features, i.e., a “Type-type” dependency. FSD is essentially a set of constraints with the following possible types:

- RBAC rule [19] in business process model describes the responsibility constraints between roles and activities, i.e., only when
<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Items</th>
<th>Feature</th>
<th>Feature Items</th>
<th>Feature</th>
<th>Feature Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td></td>
<td>47</td>
<td></td>
<td>47</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Features and feature items in the example model.
condition C is true, can a role R have the right to execute activity A;

- Numerical association rule [20] in business object model describes the association between different business objects, e.g., the “Generated from” association between Purchasing Requirement object and Purchasing Order object, or the “Allocated to” association between Sale Order object and Customer Payment object.

- ECA rule [21] in business process model describes the execution order between different business activities, i.e., only when events E occurs and condition C is true, can activity A be allowed to execute; after A’s execution, it generates new events E’.

For example, in Figure 2, the execution order of \( f_1, f_2, f_3, f_4 \) must be \( f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4 \), therefore we have a FSD \( \text{ExecOrder}(f_1, f_2, f_3, f_4) \).

### 2.3. Feature Space Partition

A business model is reusable, but when it is applied for a specific requirement, its feature space should be instantiated as a semi-abstract or fully concrete model (i.e., those features and feature items that are used for other requirements are not necessarily contained). Therefore, the feature space of a business model may be partitioned into two parts: **mandatory** and **optional** parts. The partition basis is the constraints expressed by feature dependencies.

### 3. Model Quality Evaluation Based on Feature Space Matching

In this section, with the aid of feature space as the uniform form of based business models, we design a model quality evaluation method. The basic evaluation process is to semi-automatically compare and analyze the feature spaces between user model and standard model to measure the distance (“gap”) between them. Larger distance indicates that the user model is far more inconsistent with the standard model, therefore it has lower quality.

Such distance will be assessed from two points of view: **completeness** and **soundness**. **Completeness** indicates where user model contains necessary elements of standard model, while **soundness** indicates whether user model holds those necessary constraints in standard model.

In the following discussion, we will use \( \Omega^S = \langle F^S, D^S \rangle \) to denote standard model (S) and
\( \Omega^U = \langle F^U, D^U \rangle \) as the user model \((U)\). To illuminate our method with an example, we will use the model in Figure 2 as \(S\) and use Figure 3 as \(U\) to evaluate the quality of \(U\) compared with \(S\). Table 3 shows the meanings of those new features that appear in \(U\).

### Table 3. New features and feature items in user model.

<table>
<thead>
<tr>
<th>Features or Feature Items</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{10} )</td>
<td>Accounts receivable for export</td>
</tr>
<tr>
<td>( f_{1021} )</td>
<td>Export tax refund filing</td>
</tr>
<tr>
<td>( f_5 )</td>
<td>Transfer Letter of Credit (L/C)</td>
</tr>
<tr>
<td>( f_{31} )</td>
<td>Invoice management</td>
</tr>
<tr>
<td>( f_{52} )</td>
<td>Cancel invoice</td>
</tr>
<tr>
<td>( \tau_{101} )</td>
<td>Accounts receivable for export</td>
</tr>
<tr>
<td>( \tau_{193} )</td>
<td>Write off account receivable</td>
</tr>
<tr>
<td>( \tau_{51} )</td>
<td>Invoice management</td>
</tr>
</tbody>
</table>

3.1. Completeness

Generally speaking, completeness indicates whether a user model contains the mandatory parts of the standard model. No containing or partial containing means it is possible that some functions are lost in \(U\), or \(U\)’s application scope is narrowed. The more lost functions there are, the less completeness \(U\) has.

Aiming at feature space-based business models, completeness is measured from two aspects:

1. Which mandatory features (in \(S\)) are not contained in \(U\)? If a mandatory feature \(f\) in \(S\) does not exist in \(U\), functions of \(f\) will not be supported in \(U\);

2. Which mandatory feature items (in \(S\)) are not contained in \(U\)? If a mandatory feature item \(\tau\) of \(f\) in \(S\) does not exist in \(U\), then \(U\) cannot be reused in the specific domain provisioned by \(\tau\) of \(f\), therefore \(U\)’s application scope is reduced.

**Definition 1.** (Completeness Matching Degree \(\omega\)) The completeness matching degree of \(U\) compared with \(S\) is defined as the proportion of the number of mandatory features (in \(S\)) contained in \(U\), compared with the total number of mandatory features (in \(S\)).

We use algorithms 1 and 2 to calculate \(\omega\).

Algorithm 1 (Generating the mandatory feature set of a feature)

**GenerateMandatoryPart** \((f, T, S)\)

Input: \( \Omega^S = \langle F^S, D^S \rangle \), \(f \in F^S\), \(T \subseteq \text{dom}^S(f)\), where \(\text{dom}^S(f)\) denotes the domain of \(f\) when it is in \(S\).

Output: \(MP\)

Step 1: Set the domain of \(f\) as \(T\), add \(f\) into \(MP\) and set \(f\) and all its feature items with flag 1;

Step 2: Select one feature \(g\) with no flag from \(MP\) and suppose \(\text{dom}^S(g) = Q\);

Step 2.1: Select one feature item \(\tau\) with no flag from \(Q\) with the essential sub-feature set \(\text{es}_S(\tau)\);

Step 2.1.1: If \(\text{es}_S(\tau) = \emptyset\), then go to Step 2.1.4;

Step 2.1.2: \(\forall k \in \text{es}_S(\tau)\), find those FVD/VMFVDs (from \(D^S\)) between \(f\) and \(k\); then according to these FVD/VMFVDs, choose those mandatory feature items of \(k\), denoted as \(\tau(k)\) (i.e., when \(f\) is instantiated as \(\tau\), \(k\) should be instantiated as any items in \(\tau(k)\));

Step 2.1.3: If \(k \in MP\), let \(\text{dom}(k) = \text{dom}(k) \cup \tau(k)\); otherwise, let \(\text{dom}(k) = \tau(k)\) and add \(k\) to \(MP\);

Step 2.1.4: Set \(\tau\) with flag 1 and continue to execute Step 2.1 until there are no such \(\tau\);

![Figure 3. Feature space for the user model of “Account Receivable Management”.](image_url)
Step 2.2: If there are no such $g$ in Step 2, the algorithm ends. Now $\text{MP}$ contains $f$’s all mandatory descendant features and the corresponding feature items.

Algorithm 2 (Calculating the completeness matching degree between $U$ and $S$)

\[
\text{CalculateMatchDeg} \left( U, S \right)
\]

Input: $\Omega^\ast = \langle f^S, D^S \rangle$, $\Omega^U = \langle f^U, D^U \rangle$

Output: $\omega$

Step 1: Suppose the root feature of $S$ is $f$, with the domain $T = \text{dom}^f (f)$. Call Algorithm 1 to generate $f$’s mandatory descendant features and feature items in $S$, i.e., $\text{MP} = \text{GenerateMandatoryPart} \left( f, T, S \right)$;

Step 2: Calculate the number of features/feature items contained in $\text{MP}$, i.e., $N = |\text{MP}|$;

\[
M = \sum_{g \in \text{MP}} |\text{dom} (g)|;
\]

Step 3: Let $n = 0$, $m = 0$, and $\forall g \in \text{MP}$,

Step 3.1: If $g \notin F^U$, let $n = n + 1$;

Step 3.2: If $g \in F^U$, then $\forall \tau \in \text{dom}^U (g)$, if $\tau \notin \text{dom}^S (g)$, let $m = m + 1$. Repeat Step 3.2 until all the feature items of $g$ have been checked;

Step 3.3: Repeat Step 3 until all $g$ has been checked;

Step 4: $\omega = \left( 1 - \frac{n}{N} \right) \times \left( 1 - \frac{m}{M} \right)$.

Here we use the algorithm for the example, and the results are shown in Table 4. $\omega = 0.5546$ indicates that only 55.46% mandatory elements in standard model are contained in the user model to be evaluated.

<table>
<thead>
<tr>
<th>Features that are contained in the mandatory part of $S$ but not contained in $U$</th>
<th>Features items that are contained in the mandatory part of $S$ but not contained in $U$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_2$</td>
<td>$\tau_{21}, \tau_{22}$</td>
</tr>
<tr>
<td>$f_4$</td>
<td>$\tau_{41}$</td>
</tr>
<tr>
<td>$f_{22}$</td>
<td>$\tau_{152}, \tau_{153}$</td>
</tr>
<tr>
<td>$f_{23}$</td>
<td>$\tau_{221}, \tau_{222}$</td>
</tr>
<tr>
<td>$f_{34}$</td>
<td>$\tau_{231}$</td>
</tr>
<tr>
<td>$f_{35}$</td>
<td>$\tau_{341}$</td>
</tr>
<tr>
<td>$n = 6$, $m = 12$, $N = 26$, $M = 43$, $\omega = \left( 1 - \frac{6}{26} \right) \times \left( 1 - \frac{12}{43} \right) = 0.5546$</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Calculating the completeness matching degree between $U$ and $S$ in the example.

3.2. Soundness

Generally speaking, soundness indicates whether $U$ preserves the semantics constraints in $S$. The more destroyed constraints there are in $U$, the less soundness $U$ has.

Aiming at feature space-based business models (in which semantics constraints are expressed as FDs), soundness may be mainly measured from the following four aspects:

1. Soundness of composition relationships between features (WPA). If some WPA are destroyed, then even if $U$ contains all the mandatory features in $S$, it will still lead to chaos of feature organizations, which will deteriorate the maintainability and understandability of models;

2. Soundness of instantiation relationships between features (FVD/FMVD). If some of FVD/FMVD are destroyed, then the reuse scope of related features will be widened and lead to unallowed interpretations of the models;

3. Soundness of integration relationships between features (FID). If some of FID are destroyed, then the reuse scope of related features will also be widened;

4. Soundness of semantics dependency between features (FSD). If some of FSD are destroyed, model semantics will be lost or intensified.

Definition 2. (Soundness Matching Degree $\theta$)

The soundness matching degree of $U$ compared with $S$ is defined as the degree that (1) $U$ preserves the semantics constraints in $S$ and (2) the semantics constraints in $U$ destroy the constraints in $S$.

In the following subsections, we will present the metrics of $\theta$ aiming at four types of FD respectively.

3.2.1. WPA Soundness

Although in Section 3.1 we have considered the degree that $U$ contains mandatory features in $S$, obviously we have ignored to measure whether the composition relationships between these features in $S$ are preserved in $U$, too. In this section, we try to compare structures of $U$
and $S$ to assess the WPA soundness by considering five types of matching between $U$ and $S$, i.e., the degree that WPA in $U$ satisfies the WPA in $S$.

The five types of matching are imported from [22].

- **Embedded Matching** (EM). This is the soundest matching, in which all the WPA have been preserved and the number of children of a feature is equal in $U$ and in $S$.

- **Area Matching** (AM). Similar to EM, all the WPA are also preserved, whereas the number of children of a feature in $U$ is larger than the number of children of the same feature in $S$.

- **Containment Matching** (CM). The WPA in $S$ possibly no longer maintains in $U$, but all the Ancestor-Descendant relationships in $S$ are certain to maintain in $U$.

- **Strong Constrained Containment Matching** (SCCM). Based on CM, SCCM should also follow the rule that if there are no Ancestor-Descendant relationships between $f_1, f_2, f_3$, then

  \[
  |\text{ancestor}(f_1) \cap \text{ancestor}(f_2)| \\
  = |\text{ancestor}(f_1) \cap \text{ancestor}(f_3)| \in D^S \\
  \iff \\
  |\text{ancestor}(f_1) \cap \text{ancestor}(f_2)| \\
  = |\text{ancestor}(f_1) \cap \text{ancestor}(f_3)| \in D^U
  \]

- **Weak Constrained Containment Matching** (WCCM). Based on CM, WCCM should also follow the rule that if there are no Ancestor-Descendant relationships between $f_1, f_2, f_3$, then

  \[
  |\text{ancestor}(f_1) \cap \text{ancestor}(f_2)| \\
  < |\text{ancestor}(f_1) \cap \text{ancestor}(f_3)| \in D^S \\
  \iff \\
  |\text{ancestor}(f_1) \cap \text{ancestor}(f_2)| \\
  \leq |\text{ancestor}(f_1) \cap \text{ancestor}(f_3)| \in D^U
  \]

It is easy to see that the matching degree of the five types is: $\text{EM} \rightarrow \text{AM} \rightarrow \text{SCCM} \rightarrow \text{WCCM} \rightarrow \text{CM}$, where an arrow aims at a more loose matching from a more tight one. For more discussions about these matchings, please refer to [22].

If $U$ and $S$ satisfy one of the above matchings, the WPA soundness matching degree between them may be measured by calculating the number of editing operations (e.g., insert, delete or modify features/feature items/WPA) which make $U$ accord with $S$. The larger the number is, the smaller WPA soundness should be, and vice versa.

**Algorithm 3 (WPA soundness matching degree)**

**CalculateWPAMatchDegree**($U, S$)

**Input:** $U, S$

**Output:** $\theta_{WPA}$

**Step 1:** Judge which types of matching are possible between $U$ and $S$;

**Step 2:** Calculate the editing cost for each matching between $U$ and $S$ and find the matching with the smallest editing cost $\gamma$;

**Step 3:** Calculate the total size of $S$, i.e.,

\[
K = |F^S| + \sum_{f \in F^S} |\text{dom}(f)| + \sum_{f \in F^S} |\text{child}(f)|;
\]

**Step 4:** $\theta_{WPA} = \exp\left(-\frac{\gamma}{K}\right)$.

In step 1, we may directly import the GenericMatching algorithm from [22] and use some specific optimization strategies (e.g., [23][24]) to reduce the time complexity. Due to limited space, we will not introduce the concrete process of these algorithms.

Using Figure 3 as an example, the WPA soundness matching degree between

- $S$ and $U_1$ is $\exp\left(-\frac{2}{5}\right) = 1$ ($U$ does not need any revision);
- $S$ and $U_2$ is $\exp\left(-\frac{1}{5}\right) = 0.819$ ($g$ should be deleted);
- $S$ and $U_3$ is $\exp\left(-\frac{4}{5}\right) = 0.449$ (delete $g$ and $h$, set $f_4$ and $f_5$ as the children of $f_2$);
- $S$ and $U_4$ is $\exp\left(-\frac{9}{10}\right) = 0.301$ (insert $f_2$ as a child of $f_1$, delete the WPA between $f_4, f_5$ and $f_1$, set $f_4$ and $f_5$ as the children of $f_2$);
- $S$ and $U_5$ is $\exp\left(-\frac{7}{8}\right) = 0.247$ (delete $g$, insert $f_2$ as a child of $f_1$, set $f_4$ as the children of $f_2$, delete the WPA between $f_5$ and $f_1$, set $f_5$ as the children of $f_2$, set $f_3$ as the children of $f_1$);

For the examples in Figure 2 (as $S$) and Figure 3 (as $U$), we could observe that the matching between them is a CM-type matching, and the editing operations to modify $U$ to $S$ are listed in Table 5.
3.2.2. FVD/FMVD Soundness

An FVD/FMVD describes the dependencies between values of two sets of features. If an FVD/FMVD exists in $S$, but does not in $U$, this means that the instantiation of related features in $U$ is no longer constrained by this FVD/FMVD, which enlarges the reuse scope of $U$. Otherwise, if an FVD/FMVD exists in $U$, but does not in $S$, it will reduce $U$’s reuse scope. Both situations are inadvisable.

Algorithm 4 (FVD/FMVD soundness matching degree)

Input: $S$, $U$;

Output: $\theta_{\text{FVD}}$:

Step 1: Suppose $\Theta = F^S \cap F^D$, then try to obtain the $\Theta$’s feature closure in $S$ and $U$ respectively, according to Armstrong axiom, denoted as $\Theta^*$ and $\Theta^U*$, and let $\Sigma = \Theta^* \cap \Theta^U*$;

Step 2: According to the instantiation process, get $\Sigma$’s all possible instances in $S$ and $U$ respectively, denoted as $T(\Sigma^S)$ and $T(\Sigma^U)$;

Step 3: $\theta_{\text{FVD}} = 1 - \frac{|T(\Sigma^U) - T(\Sigma^S)|}{|T(\Sigma^S)|}, \theta_{\text{FVD}} < 1$ indicates that there $\exists t \in T(\Sigma^U) \text{ but } t \notin T(\Sigma^S)$, therefore the reuse scope of $U$ is larger than $S$.

In the example, we have

$$\Theta = F^S \cap F^D = \{f_1, f_3, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19}, f_{21}, f_{24}, f_{31}, f_{32}, f_{33}\},$$

$$|T(\Sigma^S)| = 648, |T(\Sigma^U) - T(\Sigma^S)| = 136,$$

therefore

$$\theta_{\text{FVD}} = 1 - \frac{|T(\Sigma^U) - T(\Sigma^S)|}{|T(\Sigma^S)|} = 1 - \frac{136}{648} = 0.7901.$$

3.2.3. FID Soundness

Algorithm 5 (FID soundness matching degree)

Input: $U, S$;

Output: $\theta_{\text{FID}}$

Step 1: Let $\Theta = F^S \cap F^D$;

Step 2: $\forall f \in \Theta$, according to the FIDs between $f$ and its child features, calculate the following four sets of $f$ both in $U$ and $S$, respectively,

- Mandatory feature set $\text{Mandatory}^S(f), \text{Mandatory}^U(f)$;
- Optional feature set $\text{Optional}^S(f), \text{Optional}^U(f)$;
- Single selection feature set $\text{SingleSelection}^S(f), \text{SingleSelection}^U(f)$;

<table>
<thead>
<tr>
<th>Editing operation</th>
<th>Editing cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete $f_{10}$ and its item $\tau_{101}$</td>
<td>2</td>
</tr>
<tr>
<td>Delete WPA between $f_{10}$ and $f_{19}$</td>
<td>1</td>
</tr>
<tr>
<td>Delete WPA between $f_{10}$ and $f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}$</td>
<td>6</td>
</tr>
<tr>
<td>Delete $f_{101}$ and its item $\tau_{1011}$</td>
<td>2</td>
</tr>
<tr>
<td>Delete $f_{102}$ and its item $\tau_{1021}$</td>
<td>2</td>
</tr>
<tr>
<td>Add WPA between $f_{11}, f_{12}, f_{13}, f_{14}, f_{15}$ and $f_1$</td>
<td>4</td>
</tr>
<tr>
<td>Add two items $\tau_{102}, \tau_{1032}$ for $f_{15}$</td>
<td>2</td>
</tr>
<tr>
<td>Add one item $\tau_{1011}$ for $f_{19}$</td>
<td>1</td>
</tr>
<tr>
<td>Create $f_2$ and its two items $\tau_{21}, \tau_{22}$</td>
<td>3</td>
</tr>
<tr>
<td>Delete WPA between $f$ and $f_{19}$</td>
<td>1</td>
</tr>
<tr>
<td>Add WPA between $f_2$ and $f$</td>
<td>1</td>
</tr>
<tr>
<td>Create $f_2$ and two items $\tau_{221}, \tau_{222}$</td>
<td>3</td>
</tr>
<tr>
<td>Create $f_{23}$ and its item $\tau_{32}$</td>
<td>2</td>
</tr>
<tr>
<td>Delete WPA between $f_{24}$ and $f_5$</td>
<td>1</td>
</tr>
<tr>
<td>Add one item $\tau_{24}$ for $f_{24}$</td>
<td>1</td>
</tr>
<tr>
<td>Add WPA between $f_{21}, f_{22}, f_{23}, f_{24}$ and $f_2$</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Editing operation</th>
<th>Editing cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add one item $\tau_{111}$ for $f_{31}$</td>
<td>1</td>
</tr>
<tr>
<td>Create $f_{34}$ and its item $f_{341}$</td>
<td>2</td>
</tr>
<tr>
<td>Create $f_{35}$ and its item $f_{351}$</td>
<td>2</td>
</tr>
<tr>
<td>Add WPA between $f_{34}, f_{35}$ and $f_1$</td>
<td>2</td>
</tr>
<tr>
<td>Add $f_5$ and item $\tau_{351}$</td>
<td>2</td>
</tr>
<tr>
<td>Add $f_{10}$ and item $\tau_{311}$</td>
<td>2</td>
</tr>
<tr>
<td>Add $f_{21}$ and item $\tau_{21}$</td>
<td>2</td>
</tr>
<tr>
<td>Add $f_{31}$ and item $\tau_{31}$</td>
<td>2</td>
</tr>
<tr>
<td>Add WPA between $f_{41}, f_{42}, f_{43}$ and $f_6$</td>
<td>3</td>
</tr>
<tr>
<td>Add WPA between $f_6$ and $f$</td>
<td>1</td>
</tr>
<tr>
<td>Delete $f_4$ and item $\tau_{311}$</td>
<td>2</td>
</tr>
<tr>
<td>Delete $f_{24}$ and item $\tau_{221}$</td>
<td>2</td>
</tr>
<tr>
<td>Delete WPA between $f_{31}, f_{32}$ and $f_5$</td>
<td>2</td>
</tr>
<tr>
<td>Delete WPA between $f_5$ and $f$</td>
<td>1</td>
</tr>
<tr>
<td>Delete $f_5$ and item $\tau_{51}$</td>
<td>2</td>
</tr>
</tbody>
</table>

$\gamma = 64, K = 94, \theta_{\text{WPA}} = \exp \left( -\frac{64}{94} \right) = 0.5062$

Table 5. Editing operation list and the WPA soundness matching degree for the example.
• Multiple selection feature set
  $\text{MultipleSelection}^S(f), \text{MultipleSelection}^U(f)$.

Step 3: $\forall f \in \Theta$, calculate the FID soundness matching degree between $U$ and $S$, i.e.,
\[
FIDMatchDeg(f) =
\frac{1}{4} \times \left( \frac{|S(f) - S^U(f)|}{|S(f)|} + \frac{|S(f) - S^U(f)|}{|S^U(f)|} + \frac{|U(f) - U^S(f)|}{|U(f)|} + \frac{|U(f) - U^S(f)|}{|U^S(f)|} \right)
\]

Step 4: $\theta_{FID} = \frac{1}{|\Theta|} \sum_{f \in \Theta} FIDMatchDeg(f)$.

In the example, we have
\[
\theta_{FID} = \frac{1}{|\Theta|} \sum_{f \in \Theta} FIDMatchDeg(f) = 0.7920
\]
(Detailed measurement process is ignored here).

3.2.4. FSD Soundness

Because different types of FSD are quite varied, it is difficult to provide a uniform strategy for FSD soundness evaluation, but the basic process may be summarized as follows:

1. Check whether the model elements in $U$ fully satisfy each constraint of FSD in $S$ or not;
2. Check whether the FSD in $U$ destroys the constraints of FSD in $S$ or not.

In practice, specific evaluation methods must be carefully invented for each type of FSDs (e.g., ECA rules, numeric association rules, ECA rules, etc). Due to limited space we will not show the details of these methods.

For our example, we have $\theta_{FSD} = 0.8735$.

3.2.5. Integrated Soundness

Four types of soundness are integrated together to get the final soundness between $U$ and $S$. However, the four types of soundness should not be considered as having the same importance. According to our experience, we consider that the order of their weightiness may be $\theta_{WPA} > \theta_{FSD} > \theta_{FVD} > \theta_{FID}$ and their weights are specified as 0.45, 0.3, 0.15 and 0.1. Therefore, the integrated soundness is calculated by
\[
\theta = 0.45 \times \theta_{WPA} + 0.3 \times \theta_{FSD} + 0.15 \times \theta_{FVD} + 0.1 \times \theta_{FID}
\]
So in the example, we have
\[
\theta = 0.45 \times 0.5062 + 0.3 \times 0.8735 + 0.15 \times 0.7901 + 0.1 \times 0.7920 = 0.6876.
\]

3.3. Total Evaluation

In subsections 3.1 and 3.2 we have introduced two metrics for business model quality evaluation, i.e., completeness and soundness, and the final quality of $U$ compared with $S$ may be calculated by $\text{MatchDeg}(U, S) = 0.5 \times (\omega + \theta)$.

In our practice, the following standards are adopted by decision-makers to determine whether a user model is acceptable or not:

- $\text{MatchDeg}(U, S) \geq 0.8$: $U$ is “good”;
- $0.5 \leq \text{MatchDeg}(U, S) < 0.8$: $U$ is “acceptable”;
- $0.3 \leq \text{MatchDeg}(U, S) < 0.5$: $U$ is “bad”; 
- $\text{MatchDeg}(U, S) < 0.3$: $U$ is “unacceptable”.

In the example, we have
\[
\text{MatchDeg}(U, S) = 0.5 \times (\omega + \theta) = 0.5 \times (0.5546 + 0.6876) = 0.6211.
\]

It means that the consistency between the user model $U$ in Figure 3 and the standard model $S$ in Figure 2 is 62.11% and we may draw the conclusion that $U$ is an “acceptable” model and needs slight revisions.

4. Practical Validation

We applied the evaluation method in a course named “Enterprise Resource Planning: Design and Practice” in the semester of spring 2006.

Aiming at a specific business domain “Purchase Requirement and Order Management”, we asked each student to build their user model (using feature space as modeling language) based on their own understanding on this domain. After their models had been submitted,
we applied our evaluation method on them and got the following results:

<table>
<thead>
<tr>
<th>MatchDeg</th>
<th>Number of student models</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0.8, 1]</td>
<td>5 (17.9%)</td>
</tr>
<tr>
<td>[0.5, 0.8)</td>
<td>11 (39.3%)</td>
</tr>
<tr>
<td>[0.3, 0.5)</td>
<td>9 (32.1%)</td>
</tr>
<tr>
<td>(0.3, 0.3)</td>
<td>3 (10.7%)</td>
</tr>
</tbody>
</table>

Table 6. Results of experiments during the course.

Then, two assistant instructors who both had wide experience in purchase domain, manually evaluated these models and got similar results. This showed that our method was consistent with reality and applicable in practice.

5. Conclusions

In order to solve the problem of “how to quantitatively evaluate and compare the quality of business models”, we propose a new approach for business model quality evaluation, based on feature space with the following contributions:

- Based on the traditional feature modeling techniques, we extend them and import the concept “feature dependency” to uniformly express various forms of business models.
- We measure the distance between user model and standard model from two viewpoints, i.e., completeness and soundness.
- Completeness is used to judge whether a user model contains necessary model elements in the standard model. It can be measured by examining the integrity of features and feature items.
- Soundness is used to judge whether various constraints in a user model can satisfy and not destroy all the constraints in $S$. It can be measured by examining FD’s diversity between $U$ and $S$.

However, our method still has some shortcomings, e.g.,

- Although feature modeling is useful, it is complex to transform other forms of models into feature space-based models. In addition, such form of models seems to lack the ability to describe process-oriented business model;
- We have not yet found a uniform form for various types of FSD, therefore we do not have such a uniform algorithm to evaluate FSD soundness.
- Since business models and information system are both socio-technical systems and not purely technical ones, formal and fully automatic evaluation might be cumbersome, problematic, and leads to irrelevant results.

Further research should address such issues.

References

A Feature Space-based Business Model Quality Evaluation Method


Received: October, 2006
Accepted: June, 2007

Contact addresses:
Zhongjie Wang
Harbin Institute of Technology
School of Computer Science and Technology
West Dazhi Street 92, Harbin, China
e-mail: rainy@hit.edu.cn

Xiaofei Xu
Harbin Institute of Technology
School of Computer Science and Technology
West Dazhi Street 92, Harbin, China
e-mail: xiaofei@hit.edu.cn

Dechen Zhan
Harbin Institute of Technology
School of Computer Science and Technology
West Dazhi Street 92, Harbin, China
e-mail: dechen@hit.edu.cn

Zhongjie Wang is an associate professor at the Harbin Institute of Technology, Harbin, China. His research fields include enterprise and software modeling, software engineering, service-oriented computing and service engineering. He teaches Software Architecture, Introduction to Service Sciences, and Software Engineering.

Xiaofei Xu received his PhD degree in Computer Science from Harbin Institute of Technology in 1998. He is currently a professor, head of School of Computer Science and Technology and School of Software at the Harbin Institute of Technology, Harbin, China. He has more than 300 research papers presented in journals and at conferences. His research interests include database, computer integrated manufacturing (CIM), service sciences, decision support system (DSS), etc.

Dechen Zhan received his PhD degree in Computer Science from Harbin Institute of Technology in 1993. He is currently a professor at the Harbin Institute of Technology, Harbin, China. He has more than 150 research papers presented in journals and at conferences. His research interests include enterprise resource planning (ERP), model-driven architecture (MDA), enterprise modeling, etc.