Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A fuzzy formal concept analysis based approach for business component identification*

Zhen-gong CAI[1], Xiao-hu YANG[†‡1], Xin-yu WANG[1], Aleksander J. KAVS[2]
([1]*School of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*)
([2]*StateStreet Corporation, Boston, MA 02111, USA*)
[†]E-mail: yangxh@zju.edu.cn

Received Sept. 27, 2010; Revision accepted Mar. 23, 2011; Crosschecked July 6, 2011; Published online Aug. 11, 2011

Abstract: Identifying business components is the basis of component-based software engineering. Many approaches, including cluster analysis and concept analysis, have been proposed to identify components from business models. These approaches classify business elements into a set of components by analyzing their properties. However, most of them do not consider the difference in their properties for the business elements, which may decrease the accuracy of the identification results. Furthermore, component identification by partitioning business elements cannot reflect which features are responsible for the generation of certain results. This paper deals with a new approach for component identification from business models using fuzzy formal concept analysis. First, the membership between business elements and their properties is quantified and transformed into a fuzzy formal context, from which the concept lattice is built using a refined incremental algorithm. Then the components are selected from the concepts according to the concept dispersion and distance. Finally, the effectiveness and efficiency are validated by applying our approach in the real-life cases and experiments.

Key words: Business component identification, Formal concept analysis, Business model, Concept clustering, Fuzzy concept
doi:10.1631/jzus.C1000337 **Document code:** A **CLC number:** TP311

1 Introduction

Business components are proposed on the assumption that the complex business logics can be decomposed into a set of self-contained functional units, which can be reused in the context without the initial designers. Decomposing enterprise functions into business components could facilitate the logical understanding and analysis of business design. Constructing business components and assembling them into applications are two phases for component-based software engineering (Herzum and Sims, 2000). The business component construction phase is further divided into four steps—modeling business domain, identifying, designing, and implementing business components. Various definitions have been proposed for components, focusing on business or techniques view as their identification goal (Szyperski, 1998; Herzum and Sims, 2000; Baster *et al.*, 2001). In our work, we focus on identifying abstract, self-contained packages of logical units that implement specific functions from business models (Baster *et al.*, 2001). In general, business models include the business organization model, business function model, business information model, and business process model. The business organization model describes the enterprise organization. Thus, the organization model is usually not discussed in component identification (Scheer,

‡ Corresponding author

* Project supported by the Fundamental Research Funds for the Central Universities, China

2000). The business function model gives the operations on the objects to reach business goals. The business information model defines the relationships between business objects, and the business process model reflects the behavioral relationships between business activities and objects.

The existing methodologies for identifying business components from business models can be classified into two categories: structure-based approaches and feature matching approaches. Former approaches, including object-oriented component development methodology (Lee *et al.*, 1999) and O2BC (objects to business components) (Ganesan and Sengupta, 2001), convert the business models to a tree or graph. The business elements are used as nodes, and the relationships are represented using a weighted edge, where the weight for each type of relationship is set manually (Xu *et al.*, 2003). Then the tree or graph is partitioned into a set of sub-trees or subgraphs, which are used as component candidates. However, the clustering strategy based on the distance matrix (Deursen and Kuipers, 1999) cannot represent the functional features of identified component candidates. Feature matching approaches measure the similarity of business elements by analyzing their properties, which can reflect the difference of the relationships between elements (Meng *et al.*, 2006). However, the influence of the properties on business elements is treated as the same, which is probably not true in practice. Moreover, the identification results should be well organized for manual comprehension and optimization because it is nearly impossible for an algorithmic procedure to achieve optimized results (Birkmeier and Overhage, 2009).

This paper deals with a new approach for component identification from business models using fuzzy formal concept analysis (FCA). Fuzzy FCA was proposed for data mining and has been applied in software engineering and other areas (Tilley *et al.*, 2005). In our approach, business elements, the properties of the elements, and their memberships compose the formal context, which is transformed into a concept lattice for further component selection. First, business models are transformed into a fuzzy formal context, which measures the similarity of the business elements using fuzzy object-property membership. Then the concept lattice is built from the fuzzy formal context. The dispersion and distance of the concepts are analyzed and the concepts closely connected are clustered into components. Finally, the efficiency and effectiveness are analyzed by applying our approach to real cases and experiments. The results are analyzed and discussed by comparison with the clustering-based and the normal FCA-based approaches.

2 Background

2.1 Business component and business model

Business models are the resources for component identification in forward software engineering. A business model mainly consists of two types of basic elements: business objects and business activities. The business activities are executed to implement specific business functions by changing the object status. We discuss the definitions of the business object, business activity, and business component first.

Definition 1 (Business object) A business object could be represented as a 2-tuple $\mathrm{BO} = (A, \mathrm{BOP})$, where $A = \{a_1, a_2, \cdots, a_m\}$ is the attribute set of the object, and $\mathrm{BOP} = \{\mathrm{bop}_1, \mathrm{bop}_2, \cdots, \mathrm{bop}_n\}$ is the set of operations. $\mathrm{bop} = (\mathrm{name}, \mathrm{type}, \mathrm{para}, \mathrm{retType}, \mathrm{pre}, \mathrm{post})$, where name and type are the operation name and operation type respectively, para is the parameter set of the operation, retType is the return type of the operation, and pre and post represent the pre- and post-conditions of the operation, respectively. The attributes and operations represent both the static and behavioral dependency between objects.

Definition 2 (Business activity) A business activity is defined as $\mathrm{BA} = (\mathrm{DS}, \mathrm{ES}, \mathrm{BOP})$, where DS is the business data set in this activity, ES contains the events triggering this activity and those generated by this activity, and BOP is the set of operations, which are provided by the business objects. The similarity between business activities can be represented by the three types of properties—data, events, and business operations.

Definition 3 (Business component) A business component can be an object component or a process component. The business object component is an encapsulation of the business objects with similar functionality. The business process component is an encapsulation of a set of business activities in partial order that provide similar functionality.

2.2 Fuzzy formal concept analysis

FCA focuses on the discovery, sorting, and representation of concepts (Ganter and Wille, 1999). The concept lattice building is a process for concept analysis, which has been applied to information retrieval, software engineering, and knowledge discovery (Deursen and Kuipers, 1999; Quan *et al.*, 2004; Hamza, 2009). In normal FCA, the business element is treated as a concept object, its features being used as concept properties and their relationships as the object-property membership. The membership value is either 1 or 0, where 1 means the element has the property, and 0 not.

However, the memberships may be different and affect the effectiveness of the identification results. One business operation may deal with two objects to reach a function goal. The operation frequency and type on the objects may be different. In other words, the membership strength between an object and its operation is not exact because of the difference of their operation types and frequencies. For example, in the equity trading domain, the trading operation operates both Order and Trade (Order is a request for buying or selling equity, and Trade is a trading record of the orders). The trading operation reads only Order, but creates and frequently updates the generated Trade. Thus, this operation should be closer to Trade than to Order. That is, the membership strength between trading operation and Trade should be larger. Additionally, two objects may share an attribute but with a different effect; e.g., order id is a primary key for Order, but only a foreign reference in Trade. However, the difference cannot be represented in the normal FCA. By classifying the operations on their types and frequencies directly and assigning a weight for each group, the difference in each group would be omitted. Therefore, fuzzy logic is introduced into FCA to represent the strength by which a property is connected to the element. Some critical definitions of the fuzzy FCA are described before our approach is introduced.

Definition 4 (Fuzzy formal context) A formal context is a triple $F = (O, P, R)$, where O is the object set, P is the property set, and $\boldsymbol{R} = \mu(O \times P)$ is a membership matrix for the objects and properties. $\forall o \in O, p \in P$, $\mu(o, p) \in[0,1]$ is a membership function, which measures property p's membership value for object o.

Definition 5 (Extent-intent mapping) Given a fuzzy formal context $F = (O, P, R)$, the common object set of P is defined as $g(P) = \{o \in O | \forall p \in P, \mu(o, p) \geq T_{\min}\}$. Similarly, the common property set of O is defined as $f(O) = \{p \in P | \forall o \in O, \mu(o, p) \geq T_{\min}\}$. $T_{\min}$ is the minimum threshold to filter object-property values.

Definition 6 (Fuzzy formal concept) Given a fuzzy formal context $F = (O, P, R)$, $O_1 \subseteq O$, $P_1 \subseteq P$, $C_1(O_1, P_1)$ is a fuzzy concept if $O_1 = g(P_1) \wedge P_1 = f(O_1)$. O_1 is the extent of C_1 and P_1 is the intent of C_1.

Definition 7 (Fuzzy concept lattice) All the concepts from the fuzzy formal context F are denoted as $L(F)$. A partial order is defined as: if $O_1 \subseteq O_2$ (or $P_2 \supseteq P_1$), then $C_1(O_1, P_1) \leq C_2(O_2, P_2)$. Thus, $(L(F), \leq)$ is a fuzzy concept lattice. In the concept lattice, each concept is represented as a node, and the partial order is represented using an edge from sub-concept to super-concept.

3 Related work

Component identification from business models is an important step for designing component-based software systems. A large amount of research has been done on related topics (Kang *et al.*, 1998; Levi and Arsanjani, 2002; Andristsos and Tzerpos, 2003; Vitharana *et al.*, 2004). The existing approaches (Birkmeier and Overhage, 2009) differ in foundation, procedure, model, supporting measures, etc. For comparison, we discuss the related approaches from component definition, similarity measurement, and the identification strategy.

The feature-oriented reuse method (Kang *et al.*, 1998) is an identification approach to constructing and analyzing a feature model, which is a hierarchical organization of features. It can be used to develop candidate reusable components. The stability-oriented approach (Wang *et al.*, 2006) was proposed to refine the granularity of the feature-based approaches. The identification results are comprehensible using the features contained, and easy to reuse because the feature model considers the domain commonality and variability. However, there are no formal metrics for cohesion and coupling.

Business system planning (IBM, 1984) classifies activities of the business information model by analyzing their shared data objects. An object-oriented component development methodology, COMO (Lee *et al.*, 2001), was proposed by extending unified modeling language (UML) with semantics related to component development. The cases are first clustered by considering 'extends' relationships. Then a CRUD (create, read, update, and delete) matrix is applied to cluster basic processing steps. O2BC (objects to business components) (Ganesan and Sengupta, 2001) is similar to COMO. The business events and domain objects are used as the input of the identification of the logical component candidates. Components are differentiated as entity components and process components (Cheesman and Daniels, 2001). The methods based on the CRUD matrix consider the performance limitations caused by data access conflicts (Arch-int and Batanov, 2003). In addition to the CRUD-based behavioral relationships, researchers (Xu *et al.*, 2003) consider both static and behavioral relationships to obtain cohesive components. Albani *et al.* (2008) mapped the domain models into vertices and edges of a graph and partitioned the graph into components with graph theory. However, the formal metrics are usually implemented by simply summarizing the weights of the relationships, which are set manually according to the experts' experience. Meng *et al.* (2006) measured the business element relationships by analyzing their features; e.g., the features of business objects are their attributes and operations. This approach avoids setting the relationship weights manually, but the influence difference of the feature on business elements is ignored.

Lee *et al.* (2001) proposed an approach for clustering classes into components according to the high cohesion and low coupling principle. In this approach, the key classes are first selected for the component candidates, and then the other classes are assigned to the nearest component candidates by calculating the connectivity between this class and the component candidates. Selecting key classes is the critical problem that prevents the optimization of the results. A formal approach (Jain *et al.*, 2001) using the hierarchical agglomerative clustering algorithm was proposed to iteratively cluster two elements with the most strength. The strength between elements is measured using the weighted static and dynamic relationships. Another hierarchical clustering procedure (Wang *et al.*, 2005) was introduced to group business objects by analyzing the business activities they share. All of these clustering approaches aim to achieve cohesive components. However, the clustering strategy based on the distance matrix cannot represent the features of the identified results.

Concept analysis provides the results with specific features in a lattice structure. Deursen and Kuipers (1999) compared concept analysis and cluster analysis for object identification. Concept analysis has several advantages over cluster analysis, including associating features to the analysis results and assigning elements to multiple groups. Hamza (2009) proposed a framework for identifying stable domain components using FCA and software stability models. The concept lattice shows all the compositions of business elements with shared features in a layered structure. Some heuristics are given to select components from concept lattice. However, the membership value in normal FCA-based approaches is either 0 or 1. The difference in the properties for objects is ignored.

4 Fuzzy formal concept analysis based identification approach

The proposed identification approach using fuzzy formal concept analysis can be divided into three main phases: constructing the fuzzy formal context from business models, building the concept lattice, and selecting components from the concept lattice. The fuzzy FCA can be applied to identify both object components and process components.

4.1 Constructing the fuzzy formal context

For identifying object components, the business objects are treated as concept objects, while both attributes and business operations are treated as properties. The relationships are converted into the memberships. Similarly, for identifying process components, the business activities are treated as concept objects, all the events, data, and operations are treated as properties, and the relationships are also converted into the memberships of FCA. The mapping from business models to components is shown in Table 1. In the following, we will focus on the conversion from the business relationships to the formal membership.

For object components, the operation types Create (C), Update (U), and Read (R) are considered (Arch-int and Batanov, 2003) as the behavioral

Table 1 Mapping from the business model to formal context

Context	Object component	Process component
O	Business objects	Business activities
P	Attributes, operations	DS; ES; BOP
$\boldsymbol{R}$	Eqs. (2) & (3)	Eqs. (4) & (5)

O: object set; P: property set; $\boldsymbol{R}$: membership matrix for the objects and properties. DS: business data set in a business activity; ES: the events triggering this activity and those generated by this activity; BOP: the set of operations

dependency between business operations and business objects. The weights of these three types of operations are $w_{\mathrm{C}} > w_{\mathrm{U}} > w_{\mathrm{R}}$. The dependency strength between the business operation and the business object is the sum of the weights for Create, Update, and Read, i.e.,

$$w_i(p_b) = \sum w_T(p_b) \cdot f_{id}, \quad T = \mathrm{C}, \mathrm{U}, \mathrm{R}, \tag{1}$$

where w_T is the weight of each operation T for property p_b on object o_i, and f_{id} is the frequency of operation. Table 2 gives an example of the object-property matrix.

Table 2 The object-property relationship matrix

Object	a_1	bop$_1$	bop$_2$	bop$_3$	bop$_4$	bop$_5$
o_1	*	C	UR	R		
o_2		U	C		C	
o_3	*			C	U	U
o_4	*			R	R	R
o_5		R	R		R	

Each item is the business operation on the relevant business object. '*' means the object has this attribute. C: create; U: update; R: read

For the static dependency between business objects, the shared business attributes or operations can be used. For an attribute, the more operations the attribute involves, the more closely the attribute belongs to the relevant business object. If an attribute is used in all the operations of the business object, the weight is 1.0. For the operation, the C-U-R weights between the business operation and business object are used as the weight.

$$w_{ia} = |F_i(a)|/|F_i|, \tag{2}$$

$$w_{ip} = w_i(p)/\max w(p), \tag{3}$$

where F_i represents the operation set of object o_i, $F_i(a)$ is a subset of F_i that accesses attribute a, and $|\cdot|$ means the number of elements, so $w_{ia} \in$[0,1]. $w_i(p)$ is the operation weight of property p on object o_i and $\max w(p)$ is the maximum operation weight of property p on one business object; thus, $w_{ip} \in$[0,1]. The fuzzy object-property matrix in Table 3 is constructed for the running example. Here, the weights for C, U, and R are 0.5, 0.3, and 0.2, respectively. The frequency of operation is 1. The attributes or operations can be ignored if they can be owned by only one business object, since they make little contribution to the classification of business objects. The property a_1 is an attribute of the objects. The properties p_1–p_5 are for bop$_1$–bop$_5$, respectively. The minimum threshold for each property can be set according to the business difference of each property; it would be chosen by domain experts or according to historical data if possible. In this example, the thresholds are all set to 0 to keep as many values as possible. This will not affect the description of the other two steps.

Table 3 The fuzzy formal context from Table 2

Object	a_1	p_1	p_2	p_3	p_4	p_5
o_1	0.67	1	1	0.4		
o_2		0.6	1		1	
o_3	0.33			1	0.6	1
o_4	0.33			0.4	0.4	0.67
o_5		0.4	0.4		0.4	

For process component identification, all the data, events, and operations are used as the properties of business activities. The relationships between activities and their properties are mapped to fuzzy memberships. The membership between activity and data is similar to that between the business object and business operation, as in Eq. (4), where w_{id} is fuzzy membership between the ith activity and property d. The relationship between activity and event can also be represented by Read and Create, where Read means the event triggers the activity and Create means the event is generated by the activity. Finally, the relationship between activity and operation is measured considering the proportion of the data in the activity that is used by the operation. As in Eq. (5), $|D_i|$ represents the total number of data in activity i, $|D_i(p)|$ represents the number of data in operation p, and w_{ip} is the fuzzy membership for activity i and operation p.

$$\begin{gathered} w_i(d) = \sum w_T(d) \cdot f_{id}, \quad T = \mathrm{C}, \mathrm{U}, \mathrm{R}, \\ w_{id} = w_i(d)/\max w(d), \end{gathered} \tag{4}$$

$$w_{ip} = |D_i(p)|/|D_i|. \tag{5}$$

Two properties p_1 and p_2 are equivalent if they represent the same business semantic, or have the same contribution for classifying business objects, that is, $g(p_1) = g(p_2)$ in the formal context. The equivalence can be represented as $p_1 \sim p_2$. Obviously, the equivalence is reflective, symmetric, and transitive. For a property set P, the equivalence set of property $p \in P$ is $E(p) = \{p' | (p' \in P) \wedge (p' \sim p)\}$. All the equivalence sets of P are the quotient set of P at the equivalence '$\sim$', defined as $P/\sim = \{E(p) | p \in P\}$. The algorithm for property equivalence analysis (PEA) is described as follows:

Algorithm 1: Property equivalence analysis

```
Input: $P = \{p_1, p_2, \cdots, p_T\}$—a property set.
Output: $P/\sim$—an equivalence set of $P$.
 1: $P/\sim = \varnothing$;
 2: for all $p_i \in P$ do
 3:     $E(p_i) = E(p_i) \cup \{p_i\}$ ;
 4:     $P = P - \{p_i\}$;
 5:     for all $p_j \in P$ do
 6:         if $p_i \sim p_j$ then
 7:             $E(p_i) = E(p_i) \cup \{p_j\}$;
 8:             $P = P - \{p_j\}$;
 9:         end if
10:     end for
11: end for
12: $P/\sim = P/\sim \cup \{E(p_i)\}$;
```

For example, for the properties p_1 and p_2 in Table 3, $g(p_1) = g(p_2)$. Thus, the two properties can belong to the same equivalence set. Similarly, $g(a_1) = g(p_3)$ and they are assigned to the same property set. The equivalence analysis could decrease the number of the properties to be analyzed and also decrease the comparison complexity during concept lattice construction. However, all the properties in the equivalence set need to be analyzed when computing the concept dispersion and distance.

4.2 Building the concept lattice

The concept is a term with extent (business elements) and intent (their properties). The business elements in a concept share common properties, whereas the business elements in different concepts have more distance. A concept lattice provides all the concepts and their partial order from a specific formal context. There are two kinds of approaches for building the concept lattice, the batch algorithm (Nourine and Raynaud, 1999) and the incremental algorithm (Godin *et al.*, 1995; Liu *et al.*, 2007). The latter is preferable for its efficiency. However, when adding a new object, all the nodes in the lattice are analyzed to locate related nodes by calculating property intersection. The procedure can be refined if we can narrow the search scope using the layer characteristic of the lattice which is usually ignored. Some fast incremental building algorithms (Qu *et al.*, 2007) introduce the indexing tree for node location. They indeed improve the building efficiency, but more storage is needed for the indexing tree, especially for large-scale context. In this work, we propose a new procedure for building the concept lattice through the following steps:

1. The equivalence relationship of the properties is analyzed and an equivalence set $E(p)$ is used as an individual property during the building process.

2. The lattice L is initialized with a bottom node N_B, which has all the properties but no objects.

3. Get one object X from the formal context, generating a new node $C_X(X, f(\{X\}))$, and use Algorithm 2, i.e., AddNodeToLattice(L, N_B, C_X), to add C_X into the lattice L.

4. Repeat step 3 until all the objects have been processed.

According to the proposed incremental algorithm, all the nodes in the lattice are indexed by the layer and the search for the existing nodes to be modified is facilitated. Compared with the traditional building approach, our approach has two characteristics:

1. It takes advantages of the layer information of the concept lattice. The number of properties is treated as the layer number of the concept. For example, the node $(o_1, a_1p_1p_2p_3)$ is at layer 4 and the bottom node is at layer N, which is the total property number. Thus, all the nodes are indexed.

2. The search for the nodes to be modified in the lattice could be facilitated by bottom-up searching. If a node has no intersection with the new node, none of its ancestors need to be considered since the super-node has no more properties than its sub-node.

The edge-cutting actions should be executed for each edge operation to remove the potentially redundant edges: for two nodes C_A and C_B, if C_A is the sub-node of C_B, then the edges from the sub-nodes of C_A to C_B can be removed. The concept lattice from Table 3 is presented in Fig. 1.

Algorithm 2: AddNodeToLattice

```
Input: $L$—concept lattice before adding $C_X$,
$N_B$—sub-node to add $C_X$,
$C_X$—the concept to add.
Output: $L'$—concept lattice with $C_X$.
1: $L' \leftarrow L$;
2: find $C_1(O_1, P_1)$ satisfying $P_1 = P_X$ from
   $k = |P_X|$ layers in $L'$;
3: if $\nexists C_1$ then
4:    add $C_X$ to the $k$th layer;
5: else
6:    if an edge is between $C_X$ and $C_1$ then
7:       remove the edge;
8:    end if
9:    for all $C_S \geq C_1$ at layers $k, k-1, \cdots, 1$ do
10:      update $C_S(O_S \cup O_X, P_S)$;
11:      return $L'$;
12:   end for
13: end if
14: $s$=getParentNodes($N_B$);
15: add edge $N_B \to C_X$;
16: while $s \neq \varnothing$ do
17:    get $C'(O', P')$ from $s$ with max$|P'|$;
18:    if $P_\wedge = P' \cap P_X \neq \varnothing$ then
19:       $s = s\cup$ getParentNodes($C'$);
20:       generate new node $C_{\text{new}}(O' \cup O_X, P_\wedge)$;
21:       add edge $C', C_X \to C_{\text{new}}$;
22:       AddNodeToLattice($L', C', C_{\text{new}}$);
23:    end if
24: end while
25: for all $C_T$ without super-node do
26:    add edge $C_T \to$ root of $L'$;
27: end for
28: return $L'$;
```

4.3 Selecting components by concept clustering

Component identification is a process to classify business elements according to their cohesion and coupling. The business elements in a component should be closely connected and the elements in different components should have little coupling. Thus, the identification process should follow two criteria. First, a component should be self-contained and have little coupling with other components. In other words, the identified components satisfy the high cohesion and low coupling criteria. Second, a business element should be assigned to at least one component. If one element is strongly connected to more than one component, the object can be assigned to the component with the strongest connectivity for maintenance or assigned one copy to each of these components for performance.

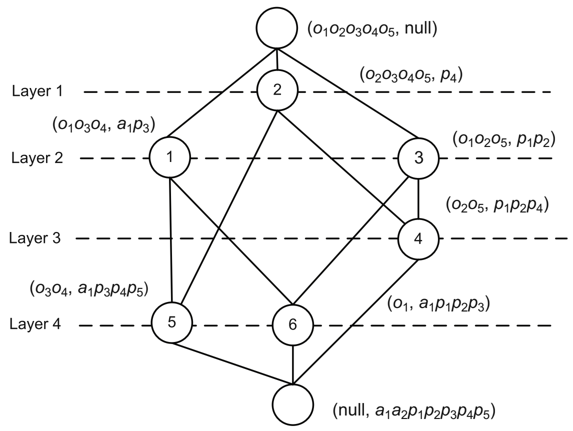

Fig. 1 The fuzzy concept lattice for Table 3

Each concept is a potential component candidate. From the cohesion view, a concept at a lower layer (with a larger property number) is likely to be selected as a component, since its objects share more properties. However, from the usability and reusability views, a concept at the higher layer is preferable since it has a larger granularity. To select the appropriate components, two metrics—concept dispersion and concept distance—are introduced. Concept dispersion means to measure the cohesion of the business objects in one concept, and concept distance is to measure the difference of two concepts. Zhou *et al.* (2007) provided some instructional metrics for measuring the membership of a property to relevant concept. If $C(O_C, P_C)$ is a concept of the fuzzy formal context $F(O, P, R)$, the membership between C and a property p is

$$E_C(p) = \begin{cases} \dfrac{1}{|O_C|}\sum\limits_{i=1}^{|O_C|}\mu(o_i, p), & p \in P_C, \\ \dfrac{1}{|O|}\sum\limits_{i=1}^{|O|}\mu(o_i, p), & \text{otherwise.} \end{cases} \tag{6}$$

The concept distance can then be measured using the membership value between concept and property. The distance between concepts $C_1(O_1, P_1)$ and $C_2(O_2, P_2)$ is defined as the Euclidean distance between nodes C_1 and C_2, represented as $||\mathbf{Dis}(C_1, C_2)||$, where $\mathbf{Dis}(C_1, C_2) = (d_{p_1}, d_{p_2}, \cdots, d_{p_T})$, and $d_{p_i} = |E_{C_1}(p_i) - E_{C_2}(p_i)|$ is the distance between C_1 and C_2 on property p_i.

The dispersion λ of a concept $C(O, P)$ is defined using the membership between the business element and its properties in that concept. The concept node with dispersion is $C(O, P, \lambda)$. The larger is the number of the properties $|P|$ in a concept, the more similar are its objects. Accordingly, the dispersion in such a concept should be lower. Thus, we introduce a logarithm function to enhance the metrics to measure the dispersion of the business objects. $|P|+1$ is introduced to avoid having denominator 0 when $|P|=1$.

$$\lambda = \frac{1}{|P|\ln(|P|+1)} \cdot \sum_{p\in P}\sqrt{\frac{\sum_{o\in O}(\mu(o,p)-E_C(p))^2}{|O|}}. \quad (7)$$

Using the dispersion and distance, an algorithm is proposed to select component candidates from the concepts. Instead of selecting some concepts as components directly using the criteria proposed by Hamza (2009), concept clustering (Quan *et al.*, 2004) is introduced using the concept distance and dispersion. Then the components are selected from the concept clusters. The inputs are concept lattice L, root C_R, coupling threshold T_S, and dispersion threshold T_D, and the output is CS, a set of concept clusters. CS is initially set to null. The component selection method using the concept clustering algorithm is given as follows:

1. The concepts are clustered using Algorithm 3, i.e., CCAlgorithm. A concept is marked if its dispersion exceeds the threshold and it contains at least one object that is not in its sub-concepts.

2. Select the root concept of each cluster as a component candidate except the marked ones and the ones with a single object.

3. Calculate the distance between the remaining concepts (i.e., marked concepts and those with a single object) and the component candidates, and merge them into the nearest component.

In the CCAlgorithm, concept dispersion is used to measure the cohesion. Once the concept dispersion exceeds the given threshold, it cannot be selected as a component candidate for its low cohesion. Thus, lines 3–5 remove the objects that appear in its sub-concepts, and treat the remaining objects, if any, as a marked 'concept'. The marked 'concept' is not a normal concept since it does not satisfy the concept definition. The marked concepts are added into the concept cluster set for further processing.

Algorithm 3: CCAlgorithm

```
Input: $C_R$—root concept for analysis,
$L$—concept lattice,
$T_S/T_D$—distantce/dispersion threshold.
Output: CS—a set of concept clusters.
 1: $F_S(R) \leftarrow \varnothing$;
 2: if $\lambda_{C_R} > T_D$ then
 3:   $O'_R \leftarrow O_R - O_{\text{Sub}}$;
 4:   if $O'_R \neq \varnothing$ then
 5:     CS $\leftarrow$ CS $\cup \{C'_R(O'_R, P_R)\}$;
 6:   end if
 7: end if
 8: for all non-analyzed sub-concept $C_S$ of $C_R$ do
 9:   $F_S(C_S) \leftarrow$ CCAlgorithm$(C_S, L, T_S, T_D)$;
10:   if $C_S \in F_S(C_S)$ then
11:     if $\|\mathbf{Dis}(C_R, C_S)\| > T_S$ or
        $\exists C > C_S, \|\mathbf{Dis}(C, C_S)\| \leq T_S$ then
12:       CS $\leftarrow$ CS $\cup \{F_S(C_S)\}$;
13:     else
14:       insert $F_S(C_S)$ into $F_S(R)$;
15:       CS $\leftarrow$ CS $- \{F_S(C_S)\}$;
16:     end if
17:   end if
18: end for
19: if all sub-nodes of $C_R$ are in $F_S(R)$ then
20:   add $C_R$ to $F_S(R)$;
21:   CS $\leftarrow$ CS $\cup \{F_S(R\}$;
22: end if
```

Via concept clustering, sub-concepts are clustered with their super-concepts if the super-concept/sub-concept distance is lower than the given distance threshold. This is used to avoid the situation in which too many fine-grained components are selected by dispersion only. In step 2, the root concept of each cluster is selected as a component candidate since it consists of all the objects in that cluster. A concept with a single object is not expected to be a component. In step 3, these single objects should be re-computed to merge into the nearest component candidate. The advantages of concept clustering for component selection are:

1. Both cohesion and coupling are considered using concept dispersion and concept distance analysis. Only concepts with dispersion lower than a given threshold can be treated as component candidates. The concepts with small distance should not be assigned to different components.

2. The number of identified fine-grained components can be controlled by concept clustering. Many component selection approaches from concept

lattices prefer concepts with low dispersion (or high cohesion). However, it usually leads to the generation of many fine-grained components.

Returning to the example, after the clustering with T_S=0.6 and T_D=0.3, two clusters are obtained: one contains concept 5 and the other contains concepts 3, 4, and 6. Concept 1 is removed since its subconcept 6 cannot be clustered with it, while concept 2 is removed because of its large dispersion (Fig. 2). The concept distance is represented on edge. Two components $\{o_1, o_2, o_5\}$ and $\{o_3, o_4\}$ are identified. If T_S=0.5 and T_D=0.3, three clusters are obtained and each has one concept: concept 5, 6, or 4. Concepts 1, 2, and 3 are removed. Concepts 4 and 5 are used as component candidates. Concept 6 should be merged into the nearest component candidate, since it has only one single object. The distance to $\{o_1, o_2, o_5\}$ is smaller than to $\{o_1, o_3, o_4\}$, so it is merged with concept 4 and the final results are also $\{o_1, o_2, o_5\}$ and $\{o_3, o_4\}$.

There are no general thresholds that can be applied to a variety of applications. The values can be a proportion of the largest concept dispersion or distance value, which can be set by analysts manually. The thresholds should be adapted repeatedly to achieve the optimized results in practice. For example, if analysts want a larger granularity, they can enlarge the dispersion value and turn down the distance value.

5 Evaluation and discussion

The evaluation of our approach is discussed from the effectiveness and efficiency views. Finding the global optimized results of component identification is an NP complete problem. The algorithmic identification procedures try to find a local optimized solution with finite effect. Thus, effectiveness should be discussed for our procedure. The efficiency of our proposed approach is evaluated by analyzing the execution time for some complex cases.

Our proposed approach was implemented in Java for evaluation. It was applied for effectiveness and efficiency evaluation. The tests were run on a Pentium IV 2.8 GHz computer, with 2 GB RAM.

5.1 Effectiveness validation

A financial business process model was analyzed to discuss the effectiveness of our approach. This

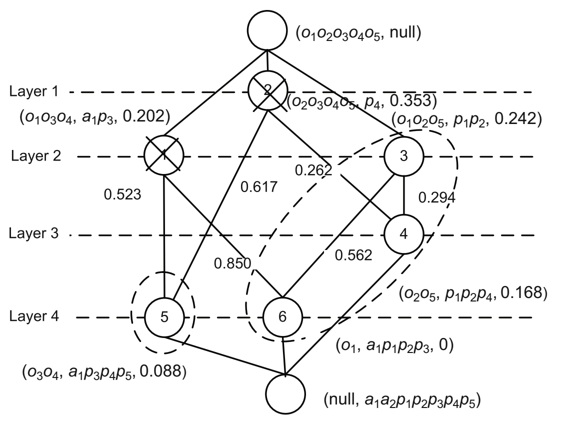
($o_1o_2o_3o_4o_5$, null)
Layer 1
2
($o_2o_3o_4o_5$, p_4, 0.353)
($o_1o_3o_4$, a_1p_3, 0.202)
($o_1o_2o_5$, p_1p_2, 0.242)
Layer 2
1
3
0.262
0.617
0.523
0.294
Layer 3
4
0.562
0.850
(o_2o_5, $p_1p_2p_4$, 0.168)
Layer 4
5
6
(o_1, $a_1p_1p_2p_3$, 0)
(o_3o_4, $a_1p_3p_4p_5$, 0.088)
(null, $a_1a_2p_1p_2p_3p_4p_5$)

Fig. 2 Concept clustering results with the coupling threshold T_S=0.6 and the dispersion threshold T_D=0.3

model is from an industrial equity trading system. The foundational logics are: The system receives orders from the traders. The order contains information such as which equity, at what price, how many shares, and under what constraints to trade, and which account to use for payment. Then the system prices the order according to the pricing strategy in the order. The buy and sell orders are traded if they satisfy the trading constraints. The trading records are reported to external agents for audit and the trading results are sent back to the traders for notification.

To clearly show the process, we used coarse-grained business elements for presentation. Six business objects were selected, and two attributes and eight coarse-grained functions/operations were used for classifying these objects. The relationship matrix between business objects and their properties was obtained from the models. ES and S are short for two attributes 'equity symbol' and 'shares', respectively. The eight filtered functions and their abbreviations are given in Table 4.

Table 4 The eight filtered functions and their abbreviations

Abbr.	Filtered function	Abbr.	Filtered function
EO	Enter order	TS	Generate trade shares
VO	Validate order	PT	Price trade
PO	Price order	RT	Report trade
CT	Create trade	AT	Allocate trade

The relationship matrix was converted into a fuzzy context using our proposed approach considering the operation type and frequency (Table 5).

Table 5 The object-property matrix for the equity trading case

Object	ES	S	EO	VO	PO	CT	TS	PT	RT	AT
Account			0.2	0.9						0.8
Order	0.9	0.8	1	1	1	0.48	0.4			0.2
Equity	1		1	0.3	0.9	0.48		0.3	0.3	0.2
Price	0.9		0.3	0.4	0.9		0.4	1	0.4	0.3
Trade	0.3	0.9				1	1	0.9	1	1
Constraints		0.7	0.2	0.46			0.9	0.8		
$T_{\min}$	0.5	0.7	0.6	0.6	0.8	0.5	0.5	0.8	0.6	0.5

ES: equity symbol; S: shares; EO: enter order; VO: validate order; PO: price order; CT: create trade; TS: generate trade shares; PT: price trade; RT: report trade; AT: allocate trade. $T_{\min}$ is the minimum threshold to filter object-property values

In our experiment, the frequency was set according to the history data of the related industrial systems. The minimum thresholds were set after discussion with experts. The fuzzy concept lattice was built from this context and all the dispersion and distance values were calculated.

The fuzzy concept lattice with dispersion and distance values is given in Fig. 3. The concept lattice can be clustered using $T_D = \max(\|\mathbf{Dis}\|/2)$ and $T_S = \max(\lambda/2)$ to obtain three components {Account}, {Trade, Constraints}, and {Order, Equity, Price}. The single object Account can be further merged into {Trade, Constraints}. It is meaningful because the account is updated only when the trade is executed. If the distance threshold is turned down to $0.4\max\|\mathbf{Dis}\|$, another component {Price} would be obtained, which supports order management and trading functions.

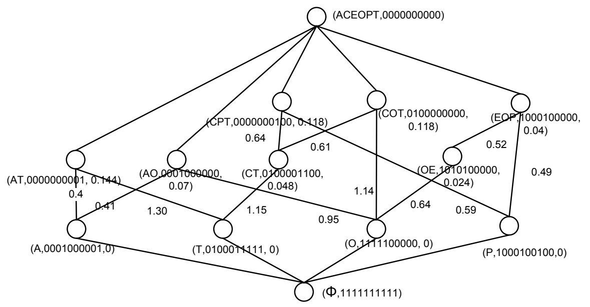

Fig. 3 The fuzzy concept lattice for Table 5. A, O, E, P, T, and C are short for Account, Order, Equity, Price, Trade, and Constraints, respectively. The 0-1 string is a bit mapping for the properties in order; e.g., 0000000100 means the node has only the property PT

Additionally, almost all the trading functions access the equity, including CT, PT, RT, and AT. However, the trading functions deal with orders instead of the equity, whereas the order management checks the equity more frequently for order entering, validation, and pricing. Thus, equity should be closer to order management than to the order trading functions.

For comparison, the clustering-based approach (Meng *et al.*, 2006) and the normal FCA-based approach were used. The clustering-based approach is frequently used for designing reusable artifacts. In the clustering process the business object is selected as the clustering entity. The object-property value is 1 or 0, and the distance is measured using the Sorenson coefficient (Lung *et al.*, 2004). Each object is treated as an initial component first. Then the component distance is computed. The components with the minimum distance are merged into a new one. The distance between components is recalculated and the merging step is repeated until none of the remaining distances is larger than the given threshold. After the first clustering step, the dependency graph is as shown in Fig. 4. The similarity between {Order} and {Equity, Price} is the same as that between {Trade} and {Equity, Price}, because the number of their shared properties is the same. Furthermore, the clustering results do not represent the properties of each identified cluster. It is not flexible for mandatory manual adjustments.

Another approach for comparison is a normal FCA-based approach. The concept lattice can be built similar to our approach. Twenty-seven concepts were generated using the formal context, much greater than the size of the fuzzy concept lattice. The concepts covered all the potential object compositions, including the identification results of our approach. The assessment quality metrics in Hamza (2009) were introduced, where the case was replaced by the business functions as properties. Order and Trade were selected as the enduring business themes, and the other objects as common business objects. The identified components are {Equity, Price}, {Trade, Account}, and {Order, Constraints}. Some manual adjustments are essential for selecting components from normal concept lattice.

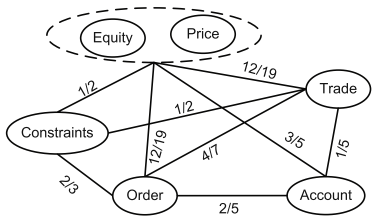

Fig. 4 The graph for the clustering-based approach

According to previous analysis, an algorithmic procedure cannot achieve the global optimized solution. The approaches considering the difference of object-property relationships may improve the accuracy of the identification results. Therefore, the effectiveness of our approach was evaluated by studying fine-grained business elements of large-scale cases.

A financial system and two e-commerce systems (http://www.prestashop.com/en/downloads/ and http://fishcart.org/download.html) were selected to evaluate the effectiveness of our approach. The business logics of e-commerce systems are well-known and the reader is referred to Lau (2006) for details. The other one is for equity trading and its foundational logics have been described above. The business objects and activities of the equity trading system were acquired from system documents. The parameters and identification results of the cases are shown in Table 6, including the numbers of business objects, activities, and events. |Identified| and |Not appropriate| are the numbers of identified components and the components that need to be amended, respectively. The identification precision, R (%), is defined as the ratio of the number of appropriate components to that of the total identified components:

R=|appropriate components|/|total identified components by this approach|×100%.

Whether a component is appropriate is determined by engineers according to the modularity quality (Vitharana, 2000). Nearly all the identified results using our approach were appropriate in experiment.

5.2 Efficiency analysis

The randomly generated contexts were used for efficiency analysis using four parameters: number of objects $|O|$, number of properties $|P|$, the random fuzzy value v, and the average object-property ratio. The first two parameters control the object number and the total property number. The parameter v is used to generate the object-property membership value. The last parameter controls the average number of properties that one object has. Two experiments were designed to evaluate the efficiency.

The first experiment is for the concept lattice building algorithm, which is the foundation for our approach. The performance was compared with that of Godin's algorithm (Godin *et al.*, 1995), which is one of the traditional incremental building algorithms. For the formal context, the object-property ratio was set to 50% and v=1.0. We designed two sets of formal contexts by varying the property or object number:

1. Set $|O|$=100 and vary $|P|$ from 10 to 24 with a step size of 2. For each context, we compared the execution time of these two approaches (Fig. 5a).

2. Set $|P|$=20 and vary $|O|$ from 20 to 140 with a step of 20. For each context, the execution time was recorded (Fig. 5b).

The second experiment is to consider the dispersion and distance thresholds. The dispersion threshold and distance threshold are usually determined from practical experience. The number of the final clusters and the time to obtain

Table 6 Identification results of the cases using the proposed approach

Case	|Object|	|Activity|	|Event|	Component type	|Identified|	|Not appropriate|	R (%)
Fishcart	21	32	26	Object	4	0	100
				Process	8	1	87.5
Pretashop	32	45	35	Object	6	0	100
				Process	9	0	100
Equity trading	56	177	99	Object	12	1	91.7
				Process	23	0	100

'$|\cdot|$' means 'the number of'. $R = (1 - |\text{Not appropriate}|/|\text{Identified}|) \times 100\%$

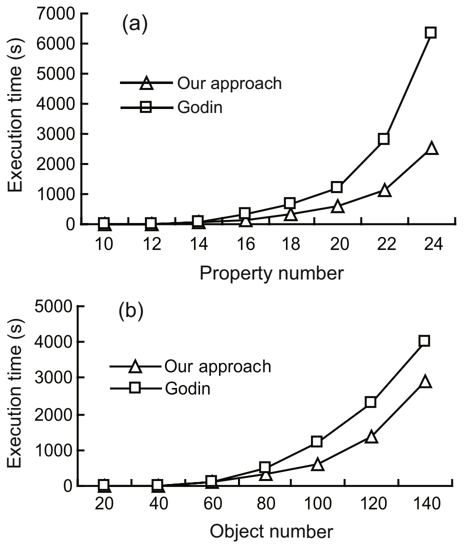

Fig. 5 Performance comparison with Godin's algorithm. (a) The number of objects is $|O|$=100; (b) The number of properties is $|P|$=20

these clusters were investigated by incrementally increasing the dispersion and distance thresholds, according to the same formal context. The context was generated with $|O|$=100, $|P|$=20, $v\in$[0.5, 1.0] and the average object-property ratio was 50% (about 4000 concepts generated). First, the dispersion threshold was set to 50% of the maximum concept dispersion in the lattice, and the distance threshold was increased incrementally from 10% to 100% of the maximum concept distance with a step of 10%. Then the impact of the dispersion threshold was investigated by setting the distance threshold to 50% of the maximum distance in the lattice. Similarly, the dispersion threshold increased from 10% to 100% of the maximum concept dispersion with a step of 10%. The analysis results are given in Fig. 6 for the cluster number and Fig. 7 for the execution time.

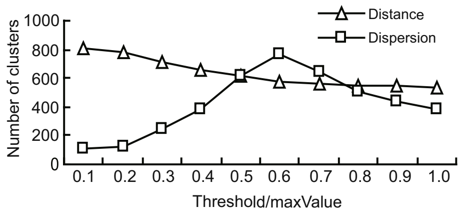

Fig. 6 Cluster number as the threshold changes

From Figs. 6 and 7, the cluster number decreased and the execution time increased as the distance threshold increased. This is obvious since more concepts are clustered for a larger distance threshold. However, for dispersion, the cluster number and execution time increased first and then decreased. This is because when the dispersion threshold is small, many concepts would be directly filtered by it. The concept cluster number (except those with only a single object) would be low and the process would be fast. As the dispersion threshold increases, the number of concepts that need to be processed increases, so the obtained concept clusters and execution time would increase. As the dispersion threshold increases, more and more high layer concepts can be selected and the probability of clustering with their sub-nodes increases. Thus, the concept cluster number would decrease. Additionally, as the sub-nodes are clustered with their super-concepts in higher layers, the duplicate analysis of the sub-nodes would be avoided such that the execution time would be shortened.

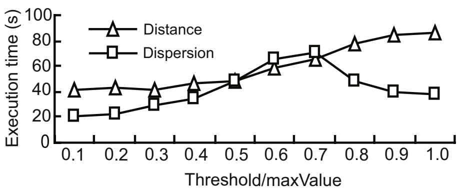

Fig. 7 Execution time as the threshold changes

5.3 Discussion

In addition to the quantitative efficiency and effectiveness evaluation, we provide a qualitative comparison of our approach to other approaches (Table 7). The feature-based approach (Kang *et al.*, 1998) is discussed, as well as the clustering-based and normal FCA-based approaches.

Both the clustering-based approach and our approach use the quantitative similarity metrics throughout the identification process. The normal FCA approach uses the quantitative quality metrics only for component selection. No quantitative metrics are used in the feature-based domain analysis, but its identification results are the most comprehensible among these four approaches. The normal FCA and our approach classify the objects according to their properties. The results in hierarchy represent the functional features of each component, which is easy to understand. However, the clustering results do not reflect the features, and are not comprehensible for analysts. Cohesion and coupling are common

Table 7 The qualitative comparison of the identification approaches

Parameter	Preference			
	Clustering	Feature-based	Normal FCA-based	Our approach
Quantitative	High	N/A	Low	High
Comprehensible	Low	High	High	High
Cohesion/Coupling	High	N/A	Low	High
Automatic	High	Low	High	High
|Intermediate results|	Low	Low	High	Medium
Precision	Medium	High	Low	High

For each characteristic, if it is preferable by one approach to others, the value is High for that approach. A weaker preference means Medium and the weakest means Low. N/A: not available

goals for component identification, especially those with clustering approaches. Our approach focuses on concept cohesion and coupling all through the identification process via dispersion and distance analysis. The normal FCA-based approach uses some criteria of high cohesion to select components from the concepts, but the feature-based approach does not consider the cohesion or coupling metric at all. Currently, many tools or algorithms are proposed to support algorithmic identification procedures using the clustering-based or FCA-based approach, but the feature-based approach mostly relies on manual analysis. The intermediate result set of the clustering and feature-based approaches is small during the identification process. However, FCA creates a concept lattice that contains all the potential object compositions. Our approach refines it by ignoring small object-property values when building the concept lattice. It tries to keep all the probable object compositions, and at the same time control the scale of the concept lattice. Finally, the precision is the ratio of the number of appropriate components identified to the number of the total identified components. The identification precision of our approach, clustering-based approach, and normal FCA-based approach has been discussed in Section 5.1. The feature-based approach is implemented manually according to experts' experience and usually has high precision.

6 Conclusions

We introduce a fuzzy FCA-based approach for identifying business components from business models. In this approach, a fuzzy formal context is constructed from the business models and transformed into a concept lattice with a refined incremental algorithm. The components are selected from the concepts according to the concept dispersion and concept distance to obtain accurate results. The effectiveness and efficiency of our approach are evaluated by several experiments. This approach makes three contributions:

1. The object-property is quantified using a fuzzy value. Different from the traditional 1-0 value, the fuzzy value in this approach could better reflect the relationship.

2. The components identified using fuzzy FCA can represent their functional features and they are organized in a layered structure. Thus, the comprehensibility and optimization of the results can be facilitated.

3. Component selection by concept clustering considers both concept cohesion and coupling, which are essential for component design.

This is ongoing research and the achievements may refine the component-based software design. We are working to extend the fuzzy FCA-based approach to consider the dependency among properties. Also, many manual interactions are needed in our approach to set the thresholds according to engineers' experience. In the future, more cases from other domains should be studied to provide criteria for the selection of the threshold values used in our approach. Additionally, the parallel algorithms for building concept lattice should be considered for large-scale business models.

References

Albani, A., Overhage, S., Birkmeier, D., 2008. Towards a systematic method for identifying business components. *LNCS*, **5282**:262-277. [doi:10.1007/978-3-540-87891-9_17]

Andristsos, P., Tzerpos, V., 2003. Software Clustering Based on Information Loss Minimization. 10th IEEE Working Conf. on Reverse Engineering, p.334-344.

Arch-int, S., Batanov, D.N., 2003. Development of industrial information systems on the Web using business components. *Comput. Ind.*, **50**(2):231-250. [doi:10.1016/S0166-3615(02)00122-7]

Baster, G., Konana, P., Scott, J.E., 2001. Business components—a case study of bankers trust Australia limited. *Commun. ACM*, **44**(5):92-98. [doi:10.1145/374308.374364]

Birkmeier, D., Overhage, S., 2009. On component identification approaches—classification, state of the art, and comparison. *LNCS*, **5582**:1-18. [doi:10.1007/978-3-642-02414-6_1]

Cheesman, J., Daniels, J., 2001. UML Components: a Simple Process for Specifying Component-Based Software. Addison-Wesley, Upper Saddle River, Boston, USA.

Deursen, A., Kuipers, T., 1999. Identifying Objects Using Cluster and Concept Analysis. 21st IEEE Int. Conf. on Software Engineering, p.246-255.

Ganesan, R., Sengupta, S., 2001. O2BC: a Technique for the Design of Component-Based Applications. 39th Int. Conf. and Exhibition on Technology of Object-Oriented Languages and Systems, p.46-55. [doi:10.1109/TOOLS.2001.941658]

Ganter, B., Wille, R., 1999. Formal Concept Analysis: Mathematical Foundations. Springer-Verlag, Berlin, Germany.

Godin, R., Missaoui, R., Alaoui, H., 1995. Incremental concept formation algorithms based on Galois (concept) lattices. *Comput. Intell.*, **11**(2):246-267. [doi:10.1111/j.1467-8640.1995.tb00031.x]

Hamza, H., 2009. A Framework for Identifying Reusable Software Components Using Formal Concept Analysis. 6th Int. Conf. on Information Technology: New Generations, p.813-818. [doi:10.1109/ITNG.2009.276]

Herzum, P., Sims, O., 2000. Business Component Factory: a Comprehensive Overview of Component-Based Development for the Enterprise. John Wiley & Sons, New York, USA.

IBM, 1984. Business System Planning: Information Systems Planning Guide. Technical Report ge20-0527-4, Int. Business Machines Corporation, USA.

Jain, H., Chalimeda, N., Ivaturi, N., Reddy, B., 2001. Business Component Identification—a Formal Approach. Proc. 5th IEEE Int. Enterprise Distributed Object Computing Conf., p.183-187. [doi:10.1109/EDOC.2001.950437]

Kang, K., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M., 1998. FORM: a feature-oriented reuse method with domain-specific reference architectures. *Ann. Software Eng.*, **5**(1):143-168. [doi:10.1023/A:1018980625587]

Lau, S., 2006. Domain Analysis of E-commerce Systems Using Feature-Based Model Templates. MS Thesis, University of Waterloo, Ontario, Canada.

Lee, J., Seung, S., Kim, S., Hyun, W., Han, D., 2001. Component Identification Method with Coupling and Cohesion. 8th IEEE Asia-Pacific Software Engineering Conf., p.79-86.

Lee, S.D., Yang, Y.J., Cho, F.S., Kim, S.D., Rhew, S.Y., 1999. COMO: a UML-Based Component Development Methodology. 6th Asia Pacific Software Engineering Conf., p.54-61. [doi:10.1109/APSEC.1999.809584]

Levi, K., Arsanjani, A., 2002. A goal-driven approach to enterprise component identification and specification. *Commun. ACM*, **45**(10):45-52. [doi:10.1145/570907.570930]

Liu, Z., Qiang, Y., Zhou, W., Li, X., Huang, M., 2007. A fuzzy concept lattice model and its incremental construction algorithm. *Chin. J. Comput.*, **30**(2):184-188 (in Chinese).

Lung, C., Zaman, M., Nandi, A., 2004. Applications of clustering techniques to software partitioning, recovery and restructuring. *J. Syst. Software*, **73**(2):227-244. [doi:10.1016/S0164-1212(03)00234-6]

Meng, F., Zhan, D., Xu, X., 2006. Reusable component design method based on domain business model. *Comput. Integr. Manuf. Syst.*, **12**(9):1402-1410 (in Chinese).

Nourine, L., Raynaud, O., 1999. A fast algorithm for building lattices. *Inform. Process. Lett.*, **71**(5-6):199-204. [doi:10.1016/S0020-0190(99)00108-8]

Qu, L., Liu, D., Yang, J., Zhang, W., 2007. Attribute-based fast incremental algorithm for building concept lattice. *J. Comput. Res. Dev.*, **44**(z3):251-256 (in Chinese).

Quan, T.T., Hui, S.C., Cao, T.H., 2004. A Fuzzy FCA-Based Approach to Conceptual Clustering for Automatic Generation of Concept Hierarchy on Uncertainty Data. 2nd Int. Conf. on Concept Lattices and Their Applications, p.1-12.

Scheer, A., 2000. ARIS-Business Process Modeling. Springer-Verlag, Berlin, Germany.

Szyperski, C., 1998. Component Software: beyond Object-Oriented Programming. Addison-Wesley, Boston, USA.

Tilley, T., Cole, R., Becker, P., Eklund, P., 2005. A survey of formal concept analysis support for software engineering activities. *LNCS*, **3626**:250-271. [doi:10.1007/11528784_13]

Vitharana, P., 2000. Designing and Managing Reusable Business Components. PhD Thesis, University of Wisconsin-Milwaukee, USA.

Vitharana, P., Jain, H., Zahedi, F., 2004. Strategy-based design of reusable business components. *IEEE Trans. Syst. Man Cybern. C*, **34**(4):460-474. [doi:10.1109/TSMCC.2004.829258]

Wang, Z., Xu, X., Zhan, D., 2005. A survey of business component identification methods and related techniques. *Int. J. Inform. Technol.*, **2**(4):229-238.

Wang, Z., Zhan, D., Xu, X., 2006. STCIM: a dynamic granularity oriented and stability based component identification method. *ACM SIGSOFT Software Eng. Notes*, **31**(3):1-14. [doi:10.1145/1127878.1127888]

Xu, W., Yin, B., Li, Z., 2003. Research on the business component design of enterprise information system. *J. Software*, **14**(7):1213-1220 (in Chinese).

Zhou, W., Liu, Z., Zhao, Y., 2007. Ontology Learning by Clustering Based on Fuzzy Formal Concept Analysis. 31st Annual Int. Computer Software and Applications Conf., p.204-210. [doi:10.1109/COMPSAC.2007.161]