An algorithm for transforming design text ROM diagram into FBS model

Min Wang, Yong Zeng, Lei Chen, Armin Eberlein

Concordia Institute for Information Systems Engineering, Faculty of Engineering and Computer Science, Concordia University, 1455 de Maisonneuve Blvd. West, EV 07.633, Montreal, Quebec, Canada H3G 1M8

Department of Computer Science & Engineering, American University of Sharjah, Sharjah, United Arab Emirates

A R T I C L E   I N F O

Article history:
Received 23 August 2011
Received in revised form 15 January 2013
Accepted 12 February 2013
Available online xxx

Keywords:
Natural language
Design text
Function–Behavior–State (FBS) model
Product–Environment System (PES)
Recursive Object Model (ROM)

A B S T R A C T

In this paper, a novel algorithm is proposed to transform a ROM diagram obtained from a design text into a FBS model. Each state of the transformation process is defined by four features: ROM (Recursive Object Model), POS (Part of Speech), PES (Product–Environment System), and FBS (Function–Behavior–State). The transformation algorithm is thus constituted by transition rules which change one transformation state to another, and procedures which apply the transition rules to a given ROM diagram. A software prototype R2FBS is presented as a proof of concept to assist the transformation. Two examples are used to demonstrate how the proposed approach works.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Design information is usually recorded in various design documents, such as customer requirements, informal notes of phone calls, meetings, e-mails and faxes. However, more formal information can be contained in design patents using a legal technical format, or formal design specifications that provide details of the accomplished design. This means that design information can be contained in various representations, such as text, verbal statements, graphic models and mathematical expressions. Research has been conducted in the processing of different kinds of design information, such as geometric modeling of products [1], sketch representations [2], management of design knowledge [3], and product requirements modeling [4–7]. Among all the representations that express design information, natural language is the most flexible yet ambiguous means; graphic models are the most effective and the most efficient; mathematical language is, however, the most precise.

Along with the advancement of computing technologies, more and more design tasks are directly or indirectly supported by computers. Directly, some design tasks are automated such as geometric modeling, structural analysis and optimization. Indirectly, some design tasks are being conducted through collaboration between human and computers such as drafting, innovation, and requirements elicitation. In order to support the entire design process, emerging CAD/E systems must be able to support the smooth integration of systems with other systems and with their human users [8]. The basis for this integration is a semantic model that can accommodate the communication of systems with each other and with their human users [9].

Technologies supporting this communication range from early efforts on geometric reasoning [10,11] to recent progresses on data/knowledge mining [12,13]. Another application in CAD/E that is closely related to semantics is the acquisition of design knowledge from existing design documents. For example, a design patent may contain novel solutions to a design problem. It is useful to extract from the design document critical design information relevant to a new design or redesign. Critical information can only be identified if the semantics of the document is understood. It must also be pointed out that the design document by itself and in its textual form is only a piece of passive knowledge. Its application depends on how the designer understands and digests the active design knowledge implied in the document so that they can be logically associated with a design situation. By definition, the understanding of a document is the transformation of the document into a formal representation constituted by a set of semantic components and their relationships [14].

A lot of research and development have been dedicated to formal and structured modeling as this would help represent semantic information in the design process. For example, in...
software engineering, Universal Modeling Language (UML) [15–17] is a widely adopted software modeling notation to specify, construct and document the artifacts [18]. A few semantic approaches have been developed to process class diagrams [19], state machines [20], interactions [21], use cases [19], OCL [22], activity diagrams [23], and so on. However, UML is a software-specific language, and does not support the general needs of design in other domains. Therefore, the system modeling language OMG SysML [24,25] was created and has been steadily gaining popularity in different domains [26]. SysML is used to specify, analyze, and design systems that may include hardware, software, and personnel. It allows engineers to describe how a system interacts with its environment, and how its parts must interact to achieve the desired system behavior and performance. The SysML model provides a shared view of the system, enabling a design team to detect issues early and prevent problems that would otherwise delay development and degrade design quality. Since SysML is based on UML, it also facilitates integration between systems and software development.

In the design of engineering systems, especially mechanical and architectural systems, function is recognized as the bridge between human desires and physical behavior of artifacts. Among various function-based models [27–31], the Function–Behavior–State (FBS) model was proposed by Umeda and Tomiyama as a framework to represent a design object hierarchically and to define a function as an association of human intention and behavior [27]. The FBS model has drawn a lot of attention in design research as it provides a knowledge representational scheme for conceptual design [27], and for the knowledge intensive engineering framework [32]. Erdem et al. based on a review of various Function Modeling (FM) approaches in the fields of artificial intelligence, design theory, and maintenance, propose a general framework of FM [30]. This general Function Modeling framework extends the FBS model to function–behavior–physical phenomenon–state model (FBPhPS) by combining the FBS model with a qualitative reasoning system (QRS) [30].

FBS modeling requires that its users understand the product requirements thoroughly and distinguish different functional stages and relationships between the functions. This could be a challenging task for a complex engineering project. The design document for a complex engineering product or process may include a large amount of information, which is tedious for human processing. A computer-aided FBS modeling system is indispensable in such a case. To support the application of the FBS modeling theory, a software tool – the FBS modeler – was developed to support the conceptual design. The FBS modeler provides a function decomposition method, which includes causal and task decompositions [27]. The decomposition process largely depends on the designer’s knowledge and experience with the FBS theory. The purpose of this paper is to generate the FBS model from an important kind of design document – design text in natural language by developing a generic experience-independent method.

In order to generate the FBS model from a design text, the following research issues have to be investigated:

- How to capture the meaning of a text?
- How to generate the function representation scheme from the captured meaning of the text during the FBS modeling?
- How to construct the right FBS model by simulating the function recognition processes?

To the authors’ best knowledge, no research results have been reported in the literature for transforming design text into FBS model. However, efforts have been made to develop automatic or semi-automatic processes to bridge unrestricted natural language text and conceptual models. Tjoa and Berger proposed an approach to transforming natural language based requirements specifications into an EER model [33]. Mala and Uma present an approach to extracting the object-oriented elements of the required system [34]. Gnies et al. developed an automatic tool for the analysis of natural language requirements [35]. Liu et al. proposed a methodology with Use-case language schemas to automate natural language requirements analysis and class model generation based on the Rational Unified Process (RUP) [36]. Due to the difficulties in natural language processing [37] and the huge gap between natural language and structured models [35,38,39], those efforts have achieved very limited success.

This paper attempts to deal with the gap between natural language and structured conceptual model through an intermediate model – Recursive Object Model (ROM) [11], which captures the semantic information of the concerned natural language. Since the application area is engineering design, a FBS model is used as our target conceptual model. Through study and practice in engineering, the ROM based transformation can help to extract system dynamics during the earlier design stage [40] and facilitate the general modeling process [41] and specific design methods such as TRIZ [42]. The proposed approach first generates the ROM diagram of a design text describing, for example, a part of product requirements or design pattern descriptions. Then the key elements included in the text, such as product components, product environment, and relations between them are extracted based on predefined rules. Finally, the key elements are transformed into a FBS model. In this paper, we will focus on the transformation from a design text ROM diagram to a FBS model.

The rest of this paper is organized as follows: the next section will introduce the algorithm in terms of input, output and the transformation of the input to the output. Then the algorithm for the generation of FBS model from a design text ROM diagram is described. Two examples are presented to demonstrate how the algorithm proposed in this paper works. The last section gives conclusions and points out future directions.

### 2. Algorithm structure

According to the traditional understanding, an algorithm is a finite, unambiguous description of an effective procedure for the solution of a class of problems. The procedure in an algorithm is often called a transformation. A transformation is defined by a set of transitions which deal with all the possible cases included in the class of problems for which the algorithm was designed [43].

Fig. 1 shows the transformation process from a design text to a FBS model. This process can be divided into two sub-processes: first, the design text described in natural language will go through a linguistic analysis process using the computer tool ROMA, which generates a ROM diagram for the design text; then, another transformation process is needed to transform the ROM diagram into a FBS model. Since the first process has been dealt with elsewhere [11], this paper focuses on the second process. Therefore, the input of this transformation is the ROM diagram corresponding to a design text whereas the output is a FBS model.

In the following sections, the input and output of transformation from ROM to FBS are addressed. The relations between the input and output are analyzed, and accordingly the transition rules are derived. At last, the algorithms are described. The foundation of this discussion is Axiomatic Theory of Design Modeling (ATDM) [44].

#### 2.1. Axiomatic Theory of Design Modeling (ATDM): an introduction

The Axiomatic Theory of Design Modeling (ATDM) is developed to represent structures of design, especially the conceptual design.
A key concept in ATDM is the structure operation, denoted by \( \mathcal{O} \), which is defined as the union (\( \cup \)) of an object \( O \) and the interaction (\( \mathcal{O} \)) of the object with itself [44].

\[
\mathcal{O} = O \cup (O \mathcal{O}),
\]
where \( \mathcal{O} \) is the structure of object \( O \).

In addition, an object is primitive if and only if

\[
\mathcal{O} = O = 0.
\]

A primitive object includes only one object. The designation of a primitive object depends on the context of design and the designer's expertise.

During the design process, design documents keep evolving from informal and unstructured to more formal and structured representations. However, as was indicated in [4,7], each design state embodies both design problem and design solutions. At any stage of design, all the design information is included in the structure of the Product–Environment System, which is defined as the structure of an object (\( \Omega \)) including both a product (\( S \)) and its environment (\( E \)).

\[
\forall E \subseteq S \subseteq E \cup S, \exists \Omega = \mathcal{O}(E \cup S) = \mathcal{O}(E) \cup \mathcal{O}(S) \cup (E \mathcal{O} S) \cup (S \mathcal{O} E), \quad \exists \Omega \quad (3)
\]

where \( \Phi \) is the null object, which is included in any object; \( \mathcal{O} E \) and \( \mathcal{O} S \) are structures of the environment and the product, respectively; \( E \mathcal{O} S \) and \( S \mathcal{O} E \) are the interactions between the product and its environment [44].

Corresponding to the subjective and objective realms adopted by Erdem et al. [30], we can divide the environment \( E \) into subjective and objective environments. The subjective environment, denoted by \( E_s \), includes the users of the product whereas the objective environment, denoted by \( E_o \), includes all of the other environment components that have an impact on the behavior of the product. Therefore,

\[
\forall \Omega = \mathcal{O}(E_s \cup E_o \cup S) = \cup (E_s \cup (E_o \cup S)), \quad (4)
\]

Eq. (4) is illustrated in Fig. 2.

Based on the structure operation, the transformation system (\( \Sigma \)) from a ROM diagram (ROM) to a FBS model (FBS) can be formally represented in Eq. (5).

\[
\Sigma = \mathcal{O}(\text{ROM} \cup \text{FBS}) = \bigcup \mathcal{O}(\text{ROM}) \cup \mathcal{O}(\text{FBS}) \cup (\text{FBS} \mathcal{O} \text{FBS}), \quad (5)
\]

which is illustrated in Fig. 3.

The transformation algorithm to be developed is part of ROM\( \rightarrow \)FBS. In order to develop this algorithm, the structures of the ROM diagram (\( \equiv \text{ROM} \)) and of the FBS (\( \equiv \text{FBS} \)) must first be formalized.

2.2. ROM

A design text includes paragraphs, phrases, and words. Its structure can be modeled by a ROM (Recursive Object Model) diagram [11], which uses five symbols to represent primitive object, compound object, constraint relation, predicate relation and connection relation, as shown in Table 1.

The ROM has been applied to software engineering [45], language translation [46,47], requirements elicitation [48], and cognitive design research [49].

2.3. FBS

The FBS model is a hierarchical knowledge representation scheme that defines a function as an association between human intention and behavior [27]. The FBS model includes functions,
behaviors, states, and physical phenomena. In this research, we study the FBS model based on Umeda and Tomiyama's work. FBS modeling includes three parts: representation of function, FBS diagram, causal decomposition and task decomposition [27]. In order to represent function, the concepts of F-B relationship, state, behavior, physical phenomena and aspect are introduced. A FBS diagram is used to distinguish between the subjective part and the objective part of a design object, to represent a function as an association of subjective concepts and objective concepts rather than just either of them, and to represent a design object hierarchically in order to support a modeling process that details functional and behavioral descriptions concurrently. Based on the FBS diagram, two approaches were proposed for functional decomposition: causal and task decompositions [27].

In this subsection, we will reformulate FBS using the ATDM theory (Axiomatic Theory of Design Modeling). As will be shown later in this paper, this reformulation will be the foundation for the development of an algorithm to transform a design text into a FBS model.

In FBS modeling, Umeda and Tomiyama define a function as “a description of behavior recognized by a human through abstraction in order to utilize it” [27]. The ROM diagram of this definition is shown in Fig. 4, which reveals the relation between function, behavior and human. This relation is formally represented in Eq. (6):

\[ F \subseteq E_b \bowtie B, \]

where \( F \) denotes function, \( E_b \) is human environment, and \( B \) is behavior. Eq. (6) implies that the function \( (F) \) can be represented as a human perception or abstraction of behavior.

In order to define behavior, the concept of state is introduced. “A state is represented as \( S(E, A, R) \), where \( E \) denotes identifiers of entities included in this state; \( A \) denotes attributes of entities; \( R \) denotes relations in the state that includes relations among entities, between entities and attributes, and among attributes” [27]. This statement can be represented using Axiomatic Theory of Design Modeling [44]. The state \( S(E, A, R) \) is represented by Eq. (7) and the relation \( R \) is denoted in Eq. (8).

\[ S = E \sqcup A \sqcup R, \]  

\[ R = (E \bowtie E) \cup (E \bowtie A) \cup (A \bowtie E) \cup (A \bowtie A). \]  

Substituting \( R \) in Eq. (8) into Eq. (7), we get

\[ S = E \sqcup A \sqcup (E \bowtie E) \cup (E \bowtie A) \cup (A \bowtie E) \cup (A \bowtie A) = \bowtie (E \sqcup A). \]

Since \( A \) denotes attributes of entities \( (E) \), \( E \) can be seen as a part of \( A, \) i.e. \( E \subseteq A, \)

\[ S = \bowtie A. \]  

“Behavior is defined by sequential one and more changes of states over time. Behavior \( b \) is represented as \( (s_0, t_0), (s_1, t_1), \ldots (s_n, t_n) \) \((n \geq 0, s_i \in S, t_i \in T)\), where \( S \) and \( T \) denote a set of states and an ordered set of time respectively” [27]. Therefore, behavior is a kind of relation from one state to another.

\[ S_t = S \bowtie T, \]

\[ b \subset B. \]

“A physical phenomenon \( PP \) causes a state transition from \( (s_i, t_i) \) to \( (s_j, t_j) \) \((i < j)\), where \( s \) represents the required condition for activating this phenomenon” [27]. We use \( E_t \) to denote the environment and \( S_p \) to denote the product in a product system. Then the physical phenomenon \( PP \) is a kind of relation from environment to product as shown in Eq. (13).

\[ PP \subseteq E_t \bowtie S_p. \]

Thus behavior \( b \) can be described by its initial state \( (s_0, t_0) \) and a set of physical phenomena \( PP \).

“An aspect \( ASP \) is defined as \( ASP = (E, A, R, PP, T) \), where \( E, A, R, PP, \) and \( T \) denotes sets of all entities, attributes, relations, physical phenomena and time of the current interest respectively” [27]. Aspect ASP can be represented by Eq. (14), which is the structure of the product system. Therefore, aspect is a kind of description of product system which consists of product, environment and relations.

\[ ASP = (S_p \sqcup E_t \sqcup R) \subseteq \bowtie (S_p \sqcup E_t). \]  

By decomposing the product structure, behavior \( B \) can be divided into a series of primitive behaviors, which can be represented as Eq. (15).

\[ B \supset B_1 \sqcup B_2 \sqcup \ldots \sqcup B_n. \]  

Therefore, Eq. (6) can be expanded as Eq. (16)

\[ F \subseteq E_b \bowtie B \subseteq E_b \bowtie (B \bowtie B_1) \cup (B \bowtie B_2) \cup \ldots \cup (B \bowtie B_n) \]  

\[ \cup \ldots \cup (B \bowtie B_n) = F_1 \cup F_2 \cup \ldots \cup F_n \cup (F_1 \bowtie F_2) \]  

\[ \ldots \cup (F_m \bowtie F_n). \]  

From the representations of behavior and state in Eqs. (11) and (12), a state is the relation of the structure of attributes within the state to time, and behavior is the change of states over time. Hence, a function could also be decomposed by time. If the decomposition is by time, it is the causal decomposition; otherwise, it is task decomposition.

Table 2 summarizes the representations of FBS corresponding to the Product–Environment System. This correspondence provides the mathematical foundation for the transformation from...
ROM to FBS, since the ROM diagram for a design text implies a Product–Environment System.

2.4. Transformation algorithm

In the present research, we attempt to derive the rules for all the transitions from an input ROM diagram to an output FBS model. According to Section 2.3, the transformation from ROM to FBS can be decomposed into two parts: from a ROM diagram (ROM) to a Product–Environment System (PES) and from a Product–Environment System to a FBS model (FBS). This is shown in Fig. 5.

To derive the algorithm for the transformation from ROM to FBS, firstly, the state of transformation should be defined clearly; and secondly, the transition rules should be developed in such a way that all of the possible transformation states can be processed.

2.4.1. Representation of states in transformation from ROM to FBS

In order to define any state in the transformation from ROM to FBS, it is critical to list all of the necessary features for each state. In this research four types features are identified, which are: POS feature, ROM feature, PES feature, and FBS feature.

Firstly, since each object in a ROM diagram is a word in the design text that needs to be processed, every object in a ROM diagram must have a Part of Speech (POS). In addition, some of the objects can be further classified according to their linguistic functions. For example, some noun objects describe humans, some other noun objects have their verb counterparts, and some verbs are linking verbs. The POS feature for a transformation state thus includes noun (n), verb (v), adjective (a), adverb (ad), determiner (d), preposition (p), and conjunction (c), together with predefined attributes associated with some semantic functions that the object may carry and are related to the transformation. Secondly, the ROM feature for a transformation state are objects, predicate relations, constraint relations and connection relations. Thirdly, the PES features for a transformation state are primitive components included in a Product–Environment System, which are product, product components, product attributes, product attribute values, environments, environment components, environment attributes, environment attribute values, and relations among them. Finally, FBS features for a transformation state are function, behavior, state, physical phenomena and aspect.

In transforming a ROM diagram to a FBS model, any state may include a combination of the four above mentioned features: POS, ROM, PES and FBS. Each object in the starting state has both POS and ROM features defined and the other two unknown whereas the ending state is constituted by the objects with all four features defined. Therefore, during the process of transformation from a ROM diagram into a FBS model, an object can be represented as quadruple of features, which is denoted by \( f(O) \) as:

\[
f(O) = (\text{ROM}(O), \text{POS}(O), \text{PES}(O), \text{FBS}(O)). \tag{17}
\]

The aim of the research presented in this paper is to identify FBS features from ROM and POS features through PES features. Table 3 summarizes these four types of object features.

In fact, the category of object and the number of relations associated with each object reflects the role and importance of this object in the ROM diagram. ROM features of an object \( \text{ROM}(O) \) are a list of relations \( R \) that relate a set of objects \( O \). Each type of objects has relations with other objects in the ROM diagram. For example, a noun object may be constrained by other objects, may constrain other objects, and may have a predicate relation to or from other objects. Any object may connect with other objects of the same POS by conjunctions.

All of the possible relations to a noun object are illustrated in Fig. 6. The constraint relation to a noun object \( N \) from an adjective or noun object \( B \) is denoted by \( C_b(N, B) \); the constraint relation to a noun object \( N_b \) is denoted by \( C_b(N, N_b) \); the constraint relation to noun object \( N_t \) through a preposition object \( P \) is denoted by \( C_b(N_t, P, N) \); the predicate relation directing from noun object \( N_1 \) to \( N_2 \) through verb \( V_1 \) is denoted by \( V_1(N_1, N_2) \); the predicate relation directing from object \( N_1 \) to a noun or adjective object \( A \) through verb \( V_2 \) is denoted by \( V_2(N_1, A) \); the connection relation between \( N \) and noun object \( N_2 \) is denoted by \( C_r(N_2, N) \). Thereby, the ROM feature of a noun object \( N \) can be denoted by:

\[
\text{ROM}(N) = C_b(N, B) \cup C_b(N, N_b) \cup C_b(N_t, P, N) \cup V_1(N_1, N) \cup V_2(N_1, A) \cup C_r(N_2, N). \tag{18}
\]

![Fig. 5. Transformation from ROM to FBS.](image1)

![Fig. 6. ROM feature of a noun object N.](image2)
As is shown in Fig. 7, a verb object \( V \) may be constrained by an adverb object \( A \), connected with another verb \( V_1 \) by conjunction, or has predicate relation directing from a noun object \( N \) to a noun or adjective object \( B \). The ROM feature of a verb object \( V \) can be denoted by

\[
ROM(V) = C_{i}(A, V) \cup V(N, B) \cup C_{m}(V, V_1).
\]  

(19)

Meanwhile, the type of verbs in a predicate relation may determine the role of the verbs in the design. The verbs can be categorized into meta verbs \( V_m \), function verbs \( V_f \), linking verbs \( V_l \), and structure verbs \( V_s \) as shown in Table 4. A meta verb relates designers to a product or its working environment; a function verb relates a product to its working environment; a linking verb introduces an attribute of the product; and a structure verb defines the components of a product.

An adjective \( A_i \) in Fig. 8, may constrain a noun object \( N_2 \), may be constrained by an adverb \( A_o \), may have a predicate relation directed by a linking verb \( V_l \), or connect with another adjective \( A_j \). The basic ROM diagram for an adjective object is shown in Fig. 8. The ROM feature of an adjective object \( A_i \) can be denoted by

\[
ROM(A_i) = C_{i}(A_j, N_2) \cup C_{i}(A_o, A_j) \cup V_l(N_1, A_j) \cup C_{m}(A_j, A_{j1}).
\]  

(20)

Similarly, an adverb \( A_o \) may constrain a verb object \( V \) or an adjective object \( A_i \), and may be connected with another adverb \( A_{o1} \), as is shown in Fig. 9. The ROM feature of an adverb object \( A_o \) can be denoted by

\[
ROM(A_o) = C_{i}(A_o, V) \cup C_{i}(A_o, A_j) \cup C_{m}(A_o, A_{o1}).
\]  

(21)

A determiner \( D \) can only constrain a noun object \( N \) as is shown in Fig. 10. The ROM feature of a determiner object \( D \) can be denoted by

\[
ROM(D) = C_{i}(D, N).
\]  

(22)

A preposition \( P \) may constrain a verb object \( V \) or a noun object \( N_2 \), from a noun object \( N_2 \), as is shown in Fig. 11. The ROM feature of a preposition object \( P \) can be denoted by

\[
ROM(P) = C_{i}(N_2, P, V) \cup C_{i}(N_2, P, N_3).
\]  

(23)

A conjunction \( C_j \) can only connect two same types of objects \( B_1 \) and \( B_2 \) as is shown in Fig. 12. The ROM feature of a conjunction object \( C_j \) can be denoted by

\[
ROM(J) = C_{i}(B_1, B_2).
\]  

(24)

According to the analysis of word features in a ROM diagram, different types of words may play different roles in a Product–Environment System. For example, a noun object can be a product, a product component, an environment, or an attribute. An adjective or adverb object can be an attribute. A verb object can be an interaction between two other objects. Preposition and conjunction objects connect other PES features into a system. The mappings between POS features, ROM features, PES features, and FBS features are described in Table 5.

### Table 4

<table>
<thead>
<tr>
<th>Verb category</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta verbs ( V_m )</td>
<td>Relates designers to a product</td>
<td>Design, develop</td>
</tr>
<tr>
<td>Function verbs ( V_f )</td>
<td>Relates a product to its environment</td>
<td>Support, maintain, raise</td>
</tr>
<tr>
<td>Linking verbs ( V_l )</td>
<td>Introduces a product’s properties</td>
<td>Be, is, are</td>
</tr>
<tr>
<td>Structure verbs ( V_s )</td>
<td>Defines a product’s components</td>
<td>Have, include, consist of</td>
</tr>
</tbody>
</table>

---

Please cite this article in press as: M. Wang, et al., An algorithm for transforming design text ROM diagram into FBS model, Comput. Industry (2013), http://dx.doi.org/10.1016/j.compind.2013.02.007
Table 5
Object mappings between POS, ROM, PES and FBS features.

<table>
<thead>
<tr>
<th>POS features</th>
<th>ROM features</th>
<th>PES features</th>
<th>FBS features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun (N)</td>
<td>ROM(N) = C(I(B, N)) ∪ C(N, N2) ∪ C(N, N3) ∪ C(N, P, N) ∪ C(V(N, P, N) ∪ C(V(N, A), N) ∪ C(N, N2)).</td>
<td>Product (P)</td>
<td>State (S)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Product component (C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attribute (A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attribute value (V)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Environment (E)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Environment component (C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Environment attribute (A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Environment attribute value (V)</td>
<td></td>
</tr>
<tr>
<td>Verb (V)</td>
<td>ROM(V) = C(A, V) ∪ V(N, B) ∪ C(V, V).</td>
<td>Relation (R)</td>
<td>Function (F)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relation (R)</td>
<td>Behavior (B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relation (R)</td>
<td>State (S)</td>
</tr>
<tr>
<td>Determiner (D)</td>
<td>ROM(D) = C(D, N).</td>
<td>Relation (R)</td>
<td>Null</td>
</tr>
<tr>
<td>Preposition (P)</td>
<td>ROM(P) = C(N, P, P) ∪ C(N, N2, P, N1).</td>
<td>Relation (R)</td>
<td>Null</td>
</tr>
<tr>
<td>Conjunction (C)</td>
<td>ROM(J) = C(B1, B2).</td>
<td>Relation (R)</td>
<td>Null</td>
</tr>
</tbody>
</table>

center objects. In most cases, a center object is a noun object. A center object is important in ROM and is often the starting point for analyzing the ROM diagram. For example, in Fig. 6, which shows the ROM feature of a noun object N, object N has two predicate relations and two constrained relations, which could affect the semantics of the noun object; therefore, N is the center object. Noun objects play roles in PES such as products, components, attributes, and environments. The center of a PES is product; therefore, determining the product object is a precondition for identifying PES features from a ROM diagram.

Rule 1 given in Table 6 is used to identify the product object from a ROM diagram. There are two possibilities: (1) the noun object has a predicate relation directed by a meta verb such as “design” and “develop” with human object being its subject; and (2) the noun object is a center object that is related to at least one function verb directed toward other objects, but no function verb directed toward it. After the product object is identified, the PES feature of the product object is updated. Then the components, component values, attributes, and environments can be identified recursively according to related rules listed in Table 6. Relations exist among product, product components, component value, and environments through predicate relations and constrain relations.

Some examples given in Table 7 demonstrate how the rules in Table 6 are applied according to the ROM diagram shown in Fig. 13.

2.4.2.2. Transformation from a PES to a FBS model. Transformation from Product–Environment System to FBS model is the process of identifying FBS features based on PES, ROM and POS features. The components of FBS features include states, functions, behaviors, physical phenomena and aspect.

Transition rules from PES to a FBS model are shown in Table 8. It must be noted that since function and behavior are generally distinguished only relatively according to the stage of a design [4], they are treated by the same rule.

2.4.3. The algorithm

Fig. 14 shows the framework for the transformation from a design text to a FBS model. First, the design document described in natural language will go through a linguistic analysis process using
Table 6
Transition rules from ROM to PES.

**Identify product**

**Rule 1:** If the POS feature of an object O in a ROM diagram is noun (N), and its ROM feature satisfies one of the following conditions, then this object is a product (P):

(a) The object has a predicate relation directed from a human (E_p) object through a meta verb (V_m).

(b) The object is a center object which has at least one predicate relation directing toward another object (O_x) through a function verb (V_f), but no predicate relation directed toward it from object (O_y) through a function verb.

\[(POS(O)=N) \land (ROM(O)=V_m(E_p, O) \lor V_f(O, O_x) \land \neg V_f(O, O_y)) \rightarrow (PES(O)=P), \exists O.\]

**Identify product components**

**Rule 2:** If the POS feature of an object O in a ROM diagram is noun (N) and its ROM feature satisfies one of the following conditions, then this object is a product component (C_p):

(a) The object is constrained by a product (P) or any other product component (C_p) object and is neither an attribute nor environment.

(b) The object has a predicate relation directed from a product or product component through a structure verb (V_s).

(c) The object is constrained by a preposition object (P) which connects a product or product component.

\[(POS(O)=N) \land (ROM(O)=C_p(P, O) \lor V_s(O, C_p), O) \lor C_p((P_1:O, C_p), P, O) \rightarrow (PES(O)=C_p), \exists O.\]

**Identify product attributes**

**Rule 3:** If the POS feature of an object O in a ROM diagram is noun (N) and its ROM features satisfy one of the following conditions, then this object is an attribute (A_p) of a product (P) or component (C_p):

(a) It constrains a product or a product component and is neither a product nor product component.

(b) It has a predicate relation directed from a product or product component through a linking verb (V_l).

\[(POS(O)=N \land (A_p(A_p, O) \land (ROM(O)=C_p(O, P) \lor V_l(O, A_p, O)) \rightarrow (PES(O)=A_p), \exists O.\]

**Identify product attribute value**

**Rule 4:** If the POS feature of an object O in a ROM diagram is noun (N) and its ROM features satisfy one of the following conditions, then this object is an attribute value (V_a):

(a) It constrains an attribute or a product attribute.

(b) It has a predicate relation directed from an attribute object through a linking verb (V_l).

\[(POS(O)=N \land (A_p(A_p, O) \land (ROM(O)=C_p(O, P) \lor V_l(O, A_p, O)) \rightarrow (PES(O)=A_p), \exists O.\]

**Identify environments**

**Rule 5:** If the POS feature of an object O in a ROM diagram is noun (N) and its ROM feature satisfies one of the following conditions, then this object is an environment (E_p) of a product (P) or product component (C_p):

(a) It has a predicate relation directed from a product or product component through a function verb (V_f), but it is not a product, product component, attribute, or attribute value.

(b) It has a predicate relation directed from an environment object through a function verb (V_f).

(c) It constrains a function object (V_f) of a product or product component through a preposition object (P).

\[(POS(O)=N \land (ROM(O)=V_f(E_p, C_p), O) \lor V_f(E_p, O) \lor C_p((P_1:O, C_p), P, O)) \rightarrow (PES(O)=E_p), \exists O.\]

**Identify environment components**

**Rule 6:** If the POS feature of an object O in a ROM diagram is noun (N) and its ROM feature satisfies one of the following conditions, then this object is an environment component (C_p):

(a) The object is constrained by an environment (E_p) or environment component (C_p), but it is not a product or a product component.

(b) The object has a predicate relation directed toward it from an environment object through a structure verb (V_s).

(c) The object is constrained by a preposition object (P) which connects an environment object.

\[(POS(O)=N \land (ROM(O)=V_s(E_p, C_p), O) \lor C_p((E_p, C_p), O) \lor C_p((E_p, C_p), P, O)) \rightarrow (PES(O)=C_p), \exists O.\]

**Identify environment attributes**

**Rule 7:** If the POS feature of an object O in a ROM diagram is noun (N) and its ROM feature satisfies one of the following conditions, then this object is an attribute (A_p) of an environment (E_p):

(a) It constrains an environment (E_p) or environment component (C_p).

(b) It has a predicate relation directed from an environment or an environment component through a linking verb (V_l).

\[(POS(O)=N \land (A_p(A_p, O) \land (ROM(O)=C_p(O, E_p) \lor V_l(O, A_p, O)) \rightarrow (PES(O)=A_p), \exists O.\]

**Identify environment attribute value**

**Rule 8:** If the POS feature of an object O in a ROM diagram is noun (N), adjective (A_p), or adverb (A_v) and its ROM feature satisfies one of the following conditions, then this object is an environment attribute value (V_m):

(a) It constrains an environment attribute object (A_p).

(b) It has a predicate relation directed from an environment attribute object and the predicate verb is a linking verb (V_l).

\[(POS(O)=N \land (A_p(A_p, O) \land (ROM(O)=C_p(O, A_p) \lor V_l(O, A_v, O)) \rightarrow (PES(O)=V_m), \exists O.\]

**Identify relations**

**Rule 9:** Relations exist among product, product components, and environments. Those relations reflect constraint relation or predicate relations in the ROM diagram.

The computer tool ROMA, which generates the ROM diagram of the design text. Then, the ROM diagram is transformed into the FBS model through another computer tool called R2FBS based on the transition rules introduced in the previous sections. This section will introduce the algorithms transforming a ROM diagram to a FBS model through the PES.

Since all of FBS features come directly from PES features, once PES features are determined, a FBS feature is defined, hence,
Rule

condition

the

physical

object,

First

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,

and

is

a

product,
can be identified according to Rule 2.2. If S has a predicate relation directed to a noun or adjective by a linking verb, then an attribute of the product can be identified according to Rule 3.2. If S has a constraint relation by a preposition, then a product component identified according to Rule 2.3. If S has a constraint relation with a noun object, then a product attribute can be identified according to Rule 3.1. If S constrains another noun object, then a product component can be identified according to Rule 2.1. For each identified component, its sub functions, environments and attributes then can be determined by related algorithms.

In the algorithm of identifying environments from an environment shown in Table 13, if environment E has a predicate relation directing to others through a function verb, then a new environment can be identified according to Rule 5.2. If E has a predicate relation directing to others through a structure verb, then an environment component can be identified according to Rule 6.2. If E has a predicate relation directing to others through a linking verb, then an environment attribute can be identified according to Rule 7.2. If E has a constraint relation by a noun or preposition, then a new environment can be identified according to Rules 6.1 and 6.3. Whenever an environment is identified, new environments related to which can be then identified recursively by the algorithm.

### 3. Examples

A software prototype called R2FBS has been developed based on the transition rules presented in the previous section. The prototype is implemented in the Microsoft Windows environment using C#. The input of this software is a XRD file which stores a ROM diagram corresponding to a design text and the output is a FBS model. R2FBS has two critical functional parts. One is the XML parsing combined with graph traversal algorithms and calculation of relations for each object. The other is an algorithm that identifies the Product–Environment System and transforms the Product–Environment System into a FBS model. Two examples are used to show how the algorithms work.

It must be pointed out that the examples used here are short paragraphs. A more complex text may increase the size of ROM diagram. Though theoretically, the present algorithms will work for ROM diagrams of any complexity, separate research is indeed needed for how to efficiently transform large text into a set of shorter paragraphs. We will report this research in a separate paper.

#### 3.1. Design patent

A United States Patent on “a low temperature clothes dryer” is chosen as an example to show how the rules are applied. The following gives the description of the design patent:

A low temperature clothes dryer having a drying chamber provides removable horizontal screens supporting clothing items and a hanging bar for hanging clothes to be dried. A timing control allows setting the time of operation of the drying cabinet. An electric heater with thermostat is provided to initially raise and maintain the air temperature within the drying chamber to at least about 90 degrees F. The dehumidifier is then operated, providing for circulation through the ducts and drying cabinet by an internal fan. The dehumidifier has an evaporator, through which warm, humid air is passed, thereby cooling the air and condensing water therefrom, the water being collected in a removable container or drained through a drain hose. The fan forces the cooled, dried air through a condenser which heats the dried air for recirculation through the drying chamber by means of ducts, thereby drying the clothing therein.


The text in the design patent of this low temperature clothes dryer is transformed into a ROM diagram as in Fig. 15, which is the input for proposed algorithm to automatically generate a FBS model from a ROM diagram.

In this example, all the objects are identified and the numbers of relations on each object in the ROM diagram are calculated. The major noun objects and relation numbers are listed in Table 14.
By applying the algorithm R2FBS introduced in the previous section, the FBS modeling process is shown in the following three steps:

**Step 1:** Determining product by applying algorithm Product(ROM)

The input is ROM, for each noun object in which, the constraint relations and predicate relations are calculated. The noun object “dryer” has five predicate relations and three constraint relations, which is the greatest number of predicate and constraint relations in all noun objects. Furthermore, it has no predicate relation directing toward it. Therefore, “dryer” can be identified as product object based on Rule 1.

**Step 2:** Identifying functions and environments by applying algorithm Function_Environment(“dryer”)

The input is product “dryer”, which is a noun form of the verb “dry” and its closest constraint is “clothes”. Therefore, the main function is “dry clothes”, and environment is “clothes” according to Rule 12.

<table>
<thead>
<tr>
<th>Object</th>
<th>Number of predicate</th>
<th>Preposition object</th>
<th>Predicate + object</th>
<th>Role in FBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dryer -- dry clothes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dryer</td>
<td>5</td>
<td></td>
<td>Having chamber; provides bar; provides screens; provides heater; having dehumidifier</td>
<td>Product</td>
</tr>
<tr>
<td>Dryer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dryer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hanging bar</td>
<td>1</td>
<td>Through, within</td>
<td>Hanging clothes</td>
<td>Component</td>
</tr>
<tr>
<td>Horizontal screens</td>
<td>1</td>
<td></td>
<td>Supporting items</td>
<td>Component</td>
</tr>
<tr>
<td>Electric heater</td>
<td>2</td>
<td></td>
<td>Maintain temperature; raise temperature</td>
<td>Component</td>
</tr>
<tr>
<td>Dehumidifier</td>
<td>4</td>
<td></td>
<td>Provide for circulation; has evaporator; condensing water; cooling air</td>
<td>Component</td>
</tr>
<tr>
<td>Timing control</td>
<td>1</td>
<td>Through, from</td>
<td>Setting time</td>
<td>Component</td>
</tr>
<tr>
<td>Evaporator</td>
<td>0</td>
<td>By</td>
<td></td>
<td>Component</td>
</tr>
<tr>
<td>Internal fan</td>
<td>0</td>
<td></td>
<td></td>
<td>Component</td>
</tr>
<tr>
<td>Thermostat</td>
<td>0</td>
<td></td>
<td></td>
<td>Component</td>
</tr>
<tr>
<td>Ducts</td>
<td>0</td>
<td>Through</td>
<td></td>
<td>Component</td>
</tr>
<tr>
<td>Removable container</td>
<td>0</td>
<td>In</td>
<td></td>
<td>Component</td>
</tr>
<tr>
<td>Drain hose</td>
<td>0</td>
<td>Through</td>
<td></td>
<td>Component</td>
</tr>
<tr>
<td>Condenser</td>
<td>0</td>
<td>Through</td>
<td></td>
<td>Component</td>
</tr>
<tr>
<td>Clothes</td>
<td>0</td>
<td></td>
<td></td>
<td>Environment</td>
</tr>
</tbody>
</table>

Fig. 15. ROM diagram for the low temperature clothes dryer.

Please cite this article in press as: M. Wang, et al., An algorithm for transforming design text ROM diagram into FBS model, Comput. Industry (2013), http://dx.doi.org/10.1016/j.compinf.2013.02.007
Step 3: Identifying product components and attributes by applying algorithm Component_Attribute(“dryer”)

The algorithm searches for noun and adjective objects that constrain the “dryer”. Those objects are identified as attributes. For example “low temperature” is an attribute of “dryer” according to Rule 3.1.

Then the algorithm searches for noun objects which are directed from “dryer” by structure verbs of “provides” and “having”; therefore, the components of “screens”, “bar”, “chamber”, “heater”, and “dehumidifier” are identified according to Rule 2.2. Based on identified product components, the algorithm calls Function_Environment(component) and Component_Attribute(component) recursively, then sub functions, environments, and components of these components can be identified, such as a function of “supporting items” for “screens”, function of “hanging clothes” for “bar”, attribute of “drying” for “chamber”, attribute of “removable” for “screens”, and components of “ducts”, “evaporator”, “thermostat”, “container”, “hose”, and “condenser” are identified through related rules. For each newly identified environment, Environment_Environment(environment) is called to identify environments related to it.

At last, the output of the design patent example is shown in Table 15, which lists the identified components of PES and FBS by prototype of R2FBS. A FBS diagram based on the elicited output is shown in Fig. 16, which illustrates the product, components, functions, environments, attributes and the relations among them for the dryer patent example.

3.2. Requirement text

This second example is extracted from an industrial project. This project aims to identify and develop system requirements starting from a brief description of the energy trading business as shown below.

Energy trading is the activity involving trading energy related commodities, such as power, natural gas, crude oil, and refined products like fuel oil, heat oil, gasoline etc. Energy is not only a consumer product, but also an investment product. As a consumer product, energy producers need to know existing demand, potential demand, and existing supply and potential supply; as an investment product, investment institutions need to know the return and risk of the investment. Given the huge demand of energy and big energy price volatility, an automation system is the only choice to manage the energy trading.

Table 15

<table>
<thead>
<tr>
<th>PES and FBS</th>
<th>Product</th>
<th>Dryer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main function</td>
<td>Dry clothes</td>
<td></td>
</tr>
<tr>
<td>Product component</td>
<td>Horizontal screens, hanging bar, drying chamber, thermostat, timing control, electric heater, dehumidifier, ducts, internal fan, condenser, removable container, drain hose, evaporator</td>
<td></td>
</tr>
<tr>
<td>Product attribute (state)</td>
<td>Low temperature for dryer, removable for screens, horizontal for screens, hanging for bar, drying for chamber, electric for heater, timing for control, internal for fan, removable for container, drain for hose</td>
<td></td>
</tr>
<tr>
<td>Sub-function</td>
<td>Screens – supporting items, bar – hanging clothes, control – setting time, heater – maintain temperature, heater – raise temperature, dehumidifier – providing for circulation, fan – forces air, condenser – heats air, dehumidifier – condensing water, dehumidifier – cooling air</td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td>Items, clothes, clothing, temperature, time, circulation, water, air</td>
<td></td>
</tr>
<tr>
<td>Environment attribute</td>
<td>Clothing for items, dried for clothes, air for temperature operation for time, cooled for air, dried for air, there for water, warm for air, humid for air, there for air</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 16. FBS diagram of design patent example.

Please cite this article in press as: M. Wang, et al., An algorithm for transforming design text ROM diagram into FBS model, Comput. Industry (2013), http://dx.doi.org/10.1016/j.compiind.2013.02.007
In the same way, we generate a ROM diagram for the text, which is illustrated in Fig. 17.

The output of R2FBS is shown in Table 16, and the FBS diagram is illustrated in Fig. 18.

3.3. Summary

As presented above, the first example about clothes dryer is a patent text, which is associated with the final stage of design; therefore the generated PES–FBS diagram is focused more on product aspects with functions. In contrast, the second example about automation system is a requirement text, which is associated with the early design stage; the PES–FBS diagram is mainly composed by environments of the product. Though the given examples used only two short paragraphs respectively, the principles and concepts can be applied to long and large documents. The challenge with large document lies mainly in the complexity of ROM diagrams. The results of examples show that the proposed approach for transformation of design text into FBS model is feasible.

Table 16
The output of requirement text example.

<table>
<thead>
<tr>
<th>PES and FBS</th>
<th>Product</th>
<th>Automation system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Product attribute</td>
<td>Automation, only choice</td>
</tr>
<tr>
<td></td>
<td>Main function</td>
<td>Manage trading</td>
</tr>
<tr>
<td></td>
<td>Product component</td>
<td>Null</td>
</tr>
<tr>
<td></td>
<td>Sub-function</td>
<td>Null</td>
</tr>
<tr>
<td>Environment</td>
<td>Environment attribute</td>
<td>Trading, energy, activity, commodities, product, consumer, demand, producers, supply, investment, return, institutions, risk, power, natural gas, crude oil, refined products, fuel oil, heat oil, gasoline</td>
</tr>
<tr>
<td></td>
<td>Energy for trading, consumer for product, existing for demand, potential for demand, huge for demand, energy for demand, potential for supply, existing for supply, investment for product, investment for risk, energy for commodities, such as for commodities, like for refined products</td>
<td></td>
</tr>
</tbody>
</table>

Please cite this article in press as: M. Wang, et al., An algorithm for transforming design text ROM diagram into FBS model, Comput. Industry (2013), http://dx.doi.org/10.1016/j.compind.2013.02.007
4. Conclusions and future work

Functional modeling is important at early stages of product design, for which most design information is described by unrestricted natural language. A precise and complete function model is very helpful for engineers. In this paper a novel algorithm is proposed to generate a structured conceptual model–FBS model from the ROM diagram corresponding to a design text. Both the ROM diagram and the FBS model are related to a Product–Environment System through the Axiomatic Theory of Design Modeling (ATDM). Rules are developed to map the objects and relations in a ROM diagram to the concepts in a FBS model. An algorithm is introduced to support the transformation from a ROM diagram to a FBS model, based on which a software prototype is developed. A design patent text and a requirement text are used as examples to show how the proposed approach works.

It must be indicated that the proposed approach does not intend to exclude human users from the loop. On the contrary, this approach may help engineers better understand requirements, especially in a large project, by reducing the ambiguities of human understanding in analyzing requirements and by increasing the consistency of the final function models when multiple engineers are involved.

As can be seen from this paper, our current approach largely depends on the capability and capacity of the ROMA system, which captures the semantics of natural language text. Therefore, the accuracy of ROMA is of a critical importance. Although ROMA is already very robust, it is still under further development. Another problem that needs to be dealt with is the study of the structure of requirements documents so that they can be pre-processed by the ROMA system. The rules for transforming a ROM diagram to a FBS model should be further validated through a more comprehensive system test based on statistical analysis.

Acknowledgments

The research reported in this paper is partially supported by NSERC (grant number RGPIN 298255). We are very grateful to the insightful comments made by the anonymous reviewers, which have helped us significantly improve our research and this paper.

References


[46] K. Wen, J. Wang, R. Li, Y. Li, Cross-language transformation based on recursive object model in understanding product requirements, in: SDPS, Jeju Island, South Korea, 2011.


Min Wang is a Ph.D. candidate in Electrical and Computer Engineering at Concordia University. Her research topic is requirements modeling: from natural language to conceptual models using ROM analysis. She holds a Master’s degree in Electrical and Computer Engineering from Concordia University, Montreal, Canada, and a Bachelor of Computer Science from Shenyang Jianzhu University, Liaoning, China. Her research interests include requirements engineering, question asking, and design science.

Dr. Yong Zeng is a Professor in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Canada. He is Canada Research Chair in Design Science (2004–2014). He received his B.Eng. degree in structural engineering from the Institute of Engineering at the Chinese University of Hong Kong, 1998. He received his Ph.D. degree in engineering science from the University of Calgary in 2001. His research is focused on the modeling and computer support of creative design activities. He and his research group have been approaching the research from philosophical, mathematical, linguistic, and computational perspectives. His research results, which range from the science of design, requirements engineering, human factors engineering, computer-aided product development, product lifecycle management, IF protection, and finite element modeling, have been applied in manufacturing industry, pharmaceutical industry, services industry, and municipality.

Lei Chen (1976–2012) was a software consultant at TKSystems Inc., Montreal. He received his B.S. degree in Information System Application from Capital Uni-

Please cite this article in press as: W. Wang, et al., An algorithm for transforming design text ROM diagram into FBS model, Comput. Industry (2013), http://dx.doi.org/10.1016/j.compind.2013.02.007