Advanced Engineering Informatics 30 (2016) 585–603

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Full length article

Automated transformation of design text ROM diagram into SysML models

Wei Wan [a], Hyunmin Cheong [b], Wei Li [b], Yong Zeng [a,*], Francesco Iorio [b]

[a] Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada
[b] Autodesk Research, Toronto, ON, Canada

ARTICLE INFO

Article history:
Received 9 February 2016
Received in revised form 4 July 2016
Accepted 20 July 2016
Available online 6 August 2016

Keywords:
Design text
SysML
Recursive Object Model (ROM)
Knowledge extraction
Model transformation

ABSTRACT

This paper proposes an approach to generating System Modeling Language (SysML) diagrams from a Recursive Object Model (ROM) diagram. The ROM diagram represents entities (or concepts) and three kinds of relations between these entities found in a description text. The generated SysML models include block definition diagram, use case diagram, and activity diagram. Since the SysML is becoming a standard modeling language for specifying, analyzing, designing and verifying complex design in many industry sectors, this transformation process supports knowledge representations of design documents for next generation CAD systems. The proposed approach first analyzes the features of ROM and SysML diagrams and then defines transition rules that transform a ROM diagram into SysML models. A software prototype ROM2SysML is developed based on the proposed approach and two examples are used to demonstrate how the prototype works.

1. Introduction

Over the last four decades computer aided design and engineering (CAD/E) systems have enabled engineers to design and develop products that can serve their customers in a more effective and efficient manner. For next generation CAD/E systems, knowledge is expected to play an increasingly important role [1,2]. General knowledge about design includes specifications, design rules, constraints, and rationale [1]. Exchange and reuse of knowledge can speed up the design process with good quality assurance. One recent trend in CAD/E is managing tasks from different engineering disciplines to support design beyond modeling and analyzing geometries with an extensive knowledge base [3].

Many design theorists agree that a design process can be viewed as a stepwise, iterative, evolutionary transformation process [4–7]. In this recursive design process, designers typically use natural language to describe design systems, particularly at an early stage. However, the ambiguity, incompleteness, and complexity of natural language present challenges in retrieving and reusing the knowledge, analyzing design solutions against design documents, and managing design changes. Hence, formal representation of description text is preferred. As a result, the objective of the current research is to extract structural and behavioral knowledge from system description text and to transform them into conceptual models expressed in SysML [8]. SysML is a graphical modeling language used to support model-based systems engineering. It is developed to meet the evolving standard ISO 10303 AP233 [9] that is a widely accepted specification language for the exchange of product specifications [10]. CAD systems use this standard to exchange data as well. SysML includes nine different diagrams, which are requirement diagram, activity diagram, sequence diagram, state machine diagram, use case diagram, block definition diagram, internal block diagram, parametric diagram, and package diagram. Those diagrams formalize system requirements, structure, behavior, and parametric constraints [8]. Various diagrams in SysML allow engineers to view a design system from different perspectives.

Researchers have investigated approaches to extracting and transforming requirements described in natural language into a formal conceptual model. Luisa et al. reviewed the challenge on existing work in using linguistic instruments to support requirements analysis and to transform natural language based design requirements into more formal and structured specifications [11]. Mens and van Gorp used linguistic tools for endogenous or exogenous transformation in the requirements elicitation phase to clarify the ambiguities and contradictions in design documents [12]. Other approaches for the direct transformation from design documents to a formal conceptual model include Data Model Generator (DMG) proposed by Tjoa and Berger to transform requirement into concepts of the EER model [13] and Natural Language to Object Oriented Design Models (NL-OOML) by Mala and Uma [14]. The

* Corresponding author.
E-mail address: yong.zeng@concordia.ca (Y. Zeng).

http://dx.doi.org/10.1016/j.aei.2016.07.003
1474-0346/

success of these efforts largely depends on the accurate semantic analysis of natural language texts [15].

Two alternatives were identified to overcome the challenges in the direct transformation [16]. The first alternative aims to describe the requirements using a controlled language. Cheong et al. proposed using a controlled natural language (CNL), which is a subset of a natural language that restricts certain syntax and lexicons, as an input tool for CAD system [17]. Through limited syntax and lexicons, ambiguities in natural language can be removed and a CNL can be translated automatically into a formal target language for reasoning. Sarkar et al. used a CNL to input software requirements for their "Design Assistant Tool" [18]. Commercial requirement management tools, such as IBM Rational DOORS [19], leverage the document structure and user added rules to semi-automatically extract design requirements from a design document. However, using a CNL is challenging because it is typically restricted to a specific problem domain and requires expert knowledge to develop and to maintain.

The second alternative is the introduction of an intermediate model that can bridge a natural language and a conceptual model. As a result, the transformation process is divided into two steps: the first is to transform a natural language text into an intermediate conceptual model and the second is to develop formal design models from the intermediate conceptual model. ROM (Recursive Object Model) is such an intermediate model to process a natural language text [20]. Compared with other linguistic models, such as the Stanford parser, which uses 42 dependencies to express universal relations [21], the simplicity of the ROM makes it possible to develop a complete set of rules for model transformation algorithms. ROM has been successfully applied to requirements elicitation [22], model transformation [23,24], product requirements translation [25] and mental stress analysis [26].

The contribution of this present research lies in the transformation of a design text into formal structured models in SysML through the ROM diagram corresponding to the design text. We define the transformation rules for mapping ROM diagrams into SysML diagrams. Currently, the transformation rules include three subsets: rules respectively for block definition diagram (BDD), use case diagram, and activity diagram. Blocks are basic structural elements in SysML that provide unifying concepts to describe the structure of elements or systems. A block definition diagram describes the relationships among blocks. Use case and activity diagrams are behavioral diagrams. A use case diagram presents basic functionality in terms of usages of the system by actors. An activity diagram specifies transformation of inputs to outputs through a controlled sequence of actions. Although many researchers have investigated transformations from natural language into formal models, most of them focus on transforming user requirements into structure diagrams such as a block definition diagram in SysML or a class diagram in UML [18,23,27–30]. In addition to the block diagram, we attempt to capture system behaviors and to generate use case and activity diagrams.

The rest of this paper is organized as follows. Section 2 reviews Recursive Object Model (ROM) and System Modeling Language (SysML). Our main work is presented in Sections 3–5. We define transformation rules in Section 3 and present a software prototype developed to deploy these algorithms in Section 4. Then Section 5 uses two examples to illustrate the implementation of the transformation process. General discussion is provided in Section 6. We conclude the paper and list future directions in Section 7.

2. Preliminary

2.1. Recursive Object Model (ROM)

We use recursive object model (ROM) as our input model to represent a description text. Instead of using syntactic pattern recognition techniques that depend on the structural information for classification and description, ROM captures the semantic information of a natural language based on computational linguistic techniques. Depending on axiomatic theory of design modeling [31], ROM represents and reasons about object structure for description text recursively. Zeng has illustrated that ROM is sufficient to represent a natural language [32].

2.1.1. ROM overview

ROM is a linguistic model for representing natural language. Unlike Entity-Relation model, in which objects are directly supported in database schema and in the query language, objects in a ROM diagram are basic units and connect with other objects to generate new objects recursively. Relations in ROM are also objects and are of only three basic types. Table 1 shows the elements of ROM and their graphical representation. Any object can be represented in ROM with only three basic relations listed below:

1. Constraint relation is a modification relation of one object to another. The constraining object modifies the meaning of the constrained object. In a ROM diagram, an arrow with a dotted head is used to represent a constraint relation.
2. Predicate relation describes an act of an object on another or describes a state of an object. A predicate relation modifies the meaning of both objects in the relation. A solid arrow is used to represent a predicate relation in a ROM diagram.
3. Connection relation connects two objects that do not constrain each other. A connection relation does not modify either of the two objects in the relation. In a ROM diagram, a connection relation is represented by a dashed arrow.

In this paper, we use C_s, C_n, and V to represent "Constraint", "Connection", and "Predicate" relations, respectively. For example, a Connection relation between object 1 (O_1) and object 2 (O_2) can be denoted by $C_n(O_1, O_2)$; a direct constraint relation to object 1 (O_1) from object 2 (O_2) is denoted by $C_s(O_2, O_1)$; the constraint relation to noun object (N_1) from noun object (N_2) through a preposition object (P) is denoted by $C_s(N_1, P, N_2)$; a predicate relation directed from object 1 (O_1) to object 2 (O_2) through a verb v is denoted by $V(O_1, v, O_2)$. Fig. 1 shows the ROM diagram of the sample text below:

> The blades of a blender are constructed of stainless steel for durability and maximum sharpness.

In Fig. 1, the object "*The*" constrains the object "*blades*" using an arrow with a dotted head, which indicates that "*The*" has a constraint relation with "*blades*". The object "*blades*" and the object "*steel*" are related through a predicate relation "*constructed of*", which is represented by a solid arrow. Formally, we use

Table 1
Elements of ROM [32].

Type		Graphic representation	Definition
Object	Object	O	Everything in the universe is an object
	Compound object	O	It is an object that includes at least two other objects in it
Relations	Constraint		It is a descriptive, limiting, or particularizing relation of one object to another
	Connection	ι	It is to connect two objects that do not constrain each other
	Predicate	ρ	It describes an act of an object on another or that describes the states of an object