Expert Systems with Applications 39 (2012) 2893–2906

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A knowledge-based object modeling advisor for developing quality object models

Narasimha Bolloju [a,*], Vijayan Sugumaran [b,c]

[a] *Department of Information Systems, City University of Hong Kong, Hong Kong*
[b] *School of Business Administration, Oakland University, Rochester, MI, USA*
[c] *Department of Service Systems Management and Engineering, Sogang Business School, Sogang University, Seoul 121-742, Republic of Korea*

ARTICLE INFO

Keywords:
Class diagrams
Knowledge-based system
Model quality
Object modeling

ABSTRACT

Object models or class diagrams are widely used for capturing information system requirements in terms of classes with attributes and operations, and relationships among those classes. Although numerous guidelines are available for object modeling as part of requirements modeling, developing quality object models has always been considered a challenging task, especially for novice systems analysts in business environments. This paper presents an approach that can be used to support the development of quality object models. The approach is implemented as a knowledge-based system extension to an open source CASE tool to offer recommendations for improving the quality of object models. The knowledge component of this system incorporates an ontology of quality problems that is based on a conceptual model quality framework commonly found in object models, the findings of related empirical studies, and a set of analysis patterns. The results obtained from an empirical evaluation of the prototype demonstrate the utility of this system, especially with respect to recommendations related to the model completeness aspect of semantic quality.

1. Introduction

Systems analysts use conceptual modeling techniques during the analysis phase of information systems development to capture systems requirements. These requirements are represented using artifacts such as use case diagrams and descriptions, class diagrams, entity-relationship diagrams, and activity diagrams during the initial phases of the systems development life cycle. Many of these artifacts facilitate communication among stakeholders and system developers in defining, documenting and confirming system requirements. The cost and level of effort required to fix any errors in requirement specifications detected in the later phases of systems development can be extremely high (Boehm, 1981). Thus, the overall quality of conceptual models created in the early phases of systems development contributes significantly to the success of the system developed.

When developing conceptual models, novice or inexperienced systems analysts encounter difficulties in the areas of domain-specific knowledge, problem-structuring, and the cognitive process (Schenk, Vitalari, & Davis, 1998). A lack of understanding of domain knowledge among requirement engineers or systems analysts and miscommunication between users and technical personnel are two problems commonly associated with the early stages of systems development (de la Vara & Sánchez, 2008). The effectiveness of a conceptual model is affected by the complex relationships that exist between modeling constructs, task requirements, the analyst's modeling experience and cognitive abilities, and the interpreter's prior exposure to conceptual models (Wand & Weber, 2002). Although numerous guidelines on the development of quality conceptual models are available, novice analysts typically fail to benefit from such assistance due to their limited domain knowledge and modeling experience. Furthermore, the absence of standardized validation procedures makes conceptual modeling a complex task for novice analysts to perform efficiently and effectively (Shanks, Tansley, & Weber, 2003).

In recent years, Unified Modeling Language (UML) techniques (OMG, 2005) such as use case models and domain models (also known as object models) have been widely used for conceptual modeling in object-oriented systems development (Dobing & Parsons, 2006). However, researchers and practitioners criticize UML for its complexity and limited usability. For example, Siau and Cao (2001) evaluate the complexity of UML with respect to other modeling methods by using a set of structural metrics and found that UML is two to eleven times more complex than other modeling techniques, and that object models are the most complex of the UML artifacts to use. Though object models are considered extremely useful in supporting activities such as client verification,

* Corresponding author. Address: Tat Chee Avenue, Kowloon Tong, Hong Kong. Tel.: +852 34427545; fax: +852 34420370.

E-mail addresses: narsi.bolloju@cityu.edu.hk (N. Bolloju), sugumara@oakland.edu (V. Sugumaran).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.08.151

clarifying technical understanding, programmer specifications, and maintenance documentation, even experienced analysts may not be using these models due to lack of understanding of this UML component (Dobing & Parsons, 2006). Novice analysts, in particular, have to overcome not only the steep learning curve associated with mastering object modeling techniques, but also the challenge of applying UML.

Considering the importance of object models (i.e., class diagrams) to systems development and the difficulties associated with the object modeling process, the need to support systems analysts should not be underestimated. Although typical computer-aided software engineering (CASE) tools include functionalities such as those enabling users to draw, document, and navigate among different models (Eriksson, Penker, Lyons, & Fado, 2004), they seldom offer capabilities that support the creation of high-quality object models. Recent years have seen several attempts to enhance the support provided in requirements modeling (e.g., Eshuis, 2006; Popescu, Rugaber, Medvidovic, & Berry, 2008; Shoval, Last, & Yampolsky, 2009). Novel extensions aimed at addressing this aspect of support are expected to not only contribute to the efficiency and effectiveness of the object modeling process, but also to help in shortening the learning process for novice analysts. Thus, the objectives of this study are to: (a) develop a knowledge-based approach that can be used to support the creation of quality UML object models; (b) implement this approach in a prototype as an extension to an existing CASE tool; and (c) demonstrate the usefulness of the prototype.

This paper presents the details of the proposed approach that exploits pattern knowledge for object model development, the implementation and evaluation of a knowledge-based system extension to an open source CASE tool. This extension will assist novice systems analysts in the important requirements capturing process task of developing quality object models. The applicable knowledge for this study is drawn from the conceptual model quality framework (Lindland, Sindre, & Solvberg, 1994) and collaboration patterns (Nicola, Mayfield, & Abney, 2001). The prototype system is evaluated by focusing on its utility of the recommendations provided. This study contributes to the extant literature by proposing a knowledge-based approach that combines a widely used conceptual model framework with a set of analysis patterns related to the business domain and by implementing it in an open source CASE tool to support the development of quality object models.

The rest of this paper is organized as follows. Section 2 reviews a widely used conceptual model quality framework and discusses three broad categories of approaches currently available for supporting the object modeling process. Section 3 presents an overview of the proposed approach and offers an example. Section 4 discusses the architecture and implementation of a knowledge-based system called the object modeling advisor (OMA) that is implemented as an extension to an open source CASE tool. Section 5 reports the results of an empirical evaluation of the extended functionality. After discussing the findings and limitations of this study and assessing its implications for research and practice in Section 6, the paper concludes by highlighting the contributions made to the existing literature.

2. Background

This section briefly reviews the conceptual model quality framework proposed by Lindland et al. (1994) and its suitability for the evaluation of object model quality. It also discusses the typical capabilities of current UML model checking tools and elaborates on three broad, but not distinct, categories of approaches available for supporting the object modeling process.

2.1. Quality of object models

The object modeling process involves the identification of classes, the attributes and operations of such classes, and the relationships among classes in the problem domain of interest. The resulting *object models* are graphically depicted as class diagrams. Object models have a relatively long history of use in both object-oriented programming and systems development (Siau, Erickson, & Lee, 2005) and are considered to be the most useful models in clarifying and understanding the static structure of a system among technical team members (Dobing & Parsons, 2006). However, the difficulties that arise in identifying required model elements (classes, attributes, operations, and relationships) often affect how closely the resulting model reflects the requirements of the system under consideration.

High quality object models not only adhere to commonly advocated notations and guidelines (e.g., the naming of classes, attributes, and associations; presentation layout for better readability), but also include required elements from the problem domain (e.g., no missing classes, attributes, or relationships; avoid using irrelevant classes, attributes, and relationships). Such quality object models can minimize communication problems and result in better understanding of model requirements, thereby reducing the effort required to accommodate such requirements at later stages of systems development.

Prior research related to conceptual model quality is relevant for studying the quality of object models. The quality of conceptual modeling can generally be divided into product and process quality (Poels, Nelson, Genero, & Piattini, 2003). Product quality is related to the resulting artifacts (*i.e.*, conceptual models), whereas process quality relates to the way in which the conceptual model is built. Among the studies addressing conceptual model quality surveyed by Wand and Weber (2002), frameworks for conceptual model quality provide a systematic structure for evaluation. Many quality criteria and frameworks have been proposed in the literature based on the desirable properties of data models (Batini, Ceri, & Navathe, 1992; Reingruber & Gregory, 1994). The framework proposed by Lindland et al. (1994) for evaluating the quality of conceptual models has gained wider acceptance due to its roots in the semiotic theory and applications, as reported in several empirical studies (Bolloju & Leung, 2006; Moody, Sindre, Brasethvik, & Solvberg, 2002; Moody, Sindre, & Brasethvik, 2003; Su & Ilebrekke, 2005). This quality framework addresses both process and product in assessing quality and is built on three linguistic concepts: syntax, semantics, and pragmatics. These concepts are applied to four aspects of modeling: language, domain, model, and audience participation.

According to this framework, the syntactic correctness of a model implies that all statements in the model accord with the syntax of the language. *Syntactic quality* captures how a given model adheres to language rules (i.e., syntax). Therefore, a smaller number of errors and deviations from the rules indicate better syntactic quality. *Semantic quality* is described in terms of validity and completeness goals. The validity goal specifies that all statements in the model are correct and relevant to the problem domain. The completeness goal specifies that the model contains all statements about the problem domain that are correct and relevant. However, it is possible that these two goals, unlike syntactic correctness, may not be achieved completely. *Pragmatic quality* addresses the comprehension aspect of the model from the stakeholders' perspective. It captures how the model has been selected from a large number of possible alternatives to express a single meaning and essentially deals with making the model easy to understand. The comprehension goal specifies that all audience members (or interpreters) completely understand the statements in the model that are relevant to them (i.e., the model projections).