Expert Systems with Applications 39 (2012) 9398–9410

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A knowledge-based system for improving the consistency between object models and use case narratives

Narasimha Bolloju [a,*], Christoph Schneider [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:
Object modeling
Use case narratives
Class diagrams
Model consistency
Knowledge-based systems

ABSTRACT

In today's systems development environments, object models are playing an increasingly important role in contributing to the agility and flexibility expected of the information systems being built. While current computer-aided software engineering tools can aid in creating object models, they do not provide much support in ensuring that the object models created are consistent with the specifications in use case narratives. This paper presents a methodology and a knowledge-based system to facilitate the verification of consistency of a given object model against a set of use case narratives. The methodology is implemented as a prototype knowledge-based extension to an open source CASE tool. The prototype's ability to reliably extract relevant information from use case narratives and its role in verifying the consistency of object models have been evaluated using a laboratory experiment. By analyzing use case narratives utilizing natural language processing techniques and applying collaboration patterns and heuristics, this methodology can determine missing and invalid model elements to guide the analyst in creating object models that are consistent with the requirements specified in a set of use case narratives. The results from this design science research indicate that the prototype system can be a useful tool to assist in this process.

1. Introduction

Today's tightly connected world has seen rapid advances in the information systems used by individuals, groups, and business organizations. This progress has fueled rapid changes in information systems development (ISD) methodologies, leading to new and continuing challenges for information systems professionals. ISD makes wide use of use case models to capture information system requirements in terms of use case diagrams and descriptions or narratives, which are then used to develop object models by identifying classes with particular attributes and operations, and by the relationships among those classes. With the increasing popularity of the Unified Modeling Language (UML) and object oriented programming languages, object modeling has become an important step in the systems development process. Object models are extremely useful in supporting activities such as client verification, clarifying technical understanding, programmer specifications and maintenance documentation. Consequently, object modeling skills are a crucial component of the skill set of today's information systems professionals.

However, acquiring object modeling skills is often challenging, especially for beginners, as the tasks and solutions are ill-defined (Borstler & Hadar, 2008; Moritz & Blank, 2005; Siau & Cao, 2001; Siau, Erickson, & Lee, 2005). Furthermore, a key challenge in developing object models is the complexity of UML, which has been demonstrated to be between two and 11 times more complex than other modeling methods (Siau & Cao, 2001), so that even experienced analysts prefer not to use the UML method due to lack of understanding (Dobing & Parsons, 2006). Computer-aided software engineering (CASE) tools have been developed to support the "mechanics" of creating object models, by aiding in drawing, documenting, and navigating through object models (Eriksson, Penker, Lyons, & Fado, 2004); however, these tools typically lack features that support the development of object models that are consistent with the requirements captured during the early stages of the development process.

This study addresses that shortcoming by focusing on an important research question: *how can CASE tools be enhanced to support validating that the object models created are consistent with the requirements specified in the use case narratives?* The answer to this question lies in the fact that CASE tools need to be augmented with features that are capable of enhancing the development of object

* 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), christoph.schneider@cityu.edu.hk (C. Schneider), sugumara@oakland.edu (V. Sugumaran).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.02.126

models that faithfully represent a set of given system requirements. Thus, the specific objectives of this research are to: (a) develop a methodology for validating object models by using the relevant information from use case narratives specific to the system requirements, (b) implement the methodology in a prototype system, and (c) evaluate the prototype system to understand the extent to which consistency can be verified. Following a design science approach (Hevner, March, Park, & Ram, 2004), we propose and build a knowledge-based system that is capable of analyzing the consistency of a given object model against a set of given use case narratives and providing recommendations for improving consistency of the model against those narratives. Although there are several types of inter-model consistency (Mens, Van der Straeten, & Simmonds, 2005), this study addresses the inter-model consistency of object models evolved from use case narratives during the analysis phase of systems development.

The remainder of this paper is organized as follows. Section 2 provides background information on the challenges faced in object modeling and the available mechanisms for consistency validation. This section also summarizes the capabilities and shortcomings of commonly used UML modeling environments and model checking tools. Section 3 outlines the proposed methodology for assistance in developing object models consistent with a set of use case narratives. Section 4 describes the design and implementation of the prototype system. In Section 5, we report a study conducted to evaluate the prototype system to assess its efficacy in improving the consistency of the object models created. Section 6 discusses the results of the evaluation process. Finally, Section 7 concludes the paper.

2. Background and motivation

Although UML models are widely used in systems analysis and design, creating high quality models has always been a challenge. While different techniques exist within UML for modeling various aspects of systems, many researchers have documented the difficulties and complexities associated with using UML. In this section, we provide a brief review of object modeling, the support provided by UML modeling environments for consistency management, and then discuss support mechanisms that are currently available.

2.1. Object modeling

Object models, comprising classes, the attributes and operations of classes, and the relationships among the classes in the problem domain, have had a relatively long history of use in the fields of object-oriented programming and systems development (Moritz & Blank, 2005). Thus, object models are now widely considered to be the most useful tools for clarifying and understanding the static structure of a system (Eriksson et al., 2004). Object models matching the requirements specifications expressed in use case narratives can minimize communication problems and misunderstandings of requirements, thereby reducing the effort required to accommodate changes at later stages of system development; as a result, object models are an integral part of UML modeling. The object modeling process includes identifying classes, the attributes and operations of classes, and the relationships among the classes in the problem domain, based on requirements that are typically captured in use cases and associated diagrams. The resulting object models are graphically depicted as class diagrams. However, various difficulties encountered in identifying the required model elements (classes, attributes, operations, and relationships) often affect how closely the resulting models reflect the requirements of the system under consideration.

As UML modeling (and, in particular, object modeling) is a cognitively intensive activity, several commercial and open source UML modeling tools have been developed to support analysts in the "mechanical" aspects of creating UML models (a partial list of the existing tools and their features can be obtained from Wikipedia, 2011). Most of these tools provide a number of basic functionalities that are common in systems modeling, such as diagram editors that assist in building models of systems. However, the current tools are vastly different in regards to ease of use and the level of support they provide in assisting analysts in the development of high quality models. In supporting different tasks, the current tools make use of a variety of features, including visual and declarative editors, support for business process modeling and metadata management, context sensitive entry, syntax coloring, and customizable design elements. Although these tools focus on the entire lifecycle of systems development, they are extremely complex and demand a steep learning curve. As a result, they do not provide adequate support for analysts to quickly develop high quality object models.

Typically, the modeling tasks are iterative processes, and the validation of the models as they are created is an important step in improving the quality of the final models. Because UML models represent system requirements from different perspectives, maintaining consistency across different artifacts is a major concern. Moreover, maintaining consistency between multiple UML diagrams that represent the static and dynamic aspects of a system is a non-trivial task, particularly when the models are frequently changed (Kotb & Katayama, 2005). Although existing tools perform reasonably well in checking individual models, such as in verifying UML activity diagrams (Eshuis & Wieringa, 2004), they have been shown to be cumbersome in checking for consistency and providing quality and timely feedback (Egyed, 2006). Hence, Lange (2006) advocates the use of metrics and visualization techniques to improve the quality of UML models during development. Similarly, Egyed (2004) discusses a tool called a UML/Analyzer that automatically determines the most appropriate consistency checking rules to execute during model changes to detect inconsistencies and provide instant feedback to the designer. Yet, those tools primarily focus on syntactic checking. Two notable exceptions are the tools described by Campbell, Cheng, McUmber, and Stirewalt (2002) and Nentwich, Capra, Emmerich, and Finkelstein (2002). The tool described by Campbell et al. is capable of performing structural and behavioral analyses to detect errors, which it presents in terms of the original UML diagrams. In addition, it performs intra- and inter-diagram consistency checks as part of the analysis process. The tool described by Nentwich et al. provides an incremental way of finding inconsistencies in UML diagrams represented as XML files. Despite these efforts, in their comprehensive review of research prototypes for maintaining UML model consistency, Lucas, Molina, and Toval (2009) highlight a lack of research devoted to tackling inter-model consistency problems and integrating consistency management features within CASE tools.

Creating quality object models goes beyond the mechanical aspects of drawing syntactically correct diagrams. In particular, although the quality of a model is heavily dependent on the extent to which it captures a system's requirements, UML modeling tools do not provide explicit mechanisms to help systems analysts validate that the models they create truly capture the requirements specified. Because an object model is a form of conceptual model, the challenges associated with object modeling are comparable to those associated with conceptual modeling. When developing conceptual models, novice or inexperienced systems analysts tend to encounter difficulties in the areas of domain-specific knowledge and problem-structuring, as well as in the cognitive processes associated with identifying model elements (Schenk, Vitalari, & Davis, 1998); especially for novice analysts, conceptual modeling is a complex task to perform efficiently and effectively, because of the absence of standardized validation procedures (Shanks, Tansley, &