Requirements Eng (2004) 9: 229–237
DOI 10.1007/s00766-004-0203-7

ORIGINAL ARTICLE

Keng Siau · Lihyunn Lee

Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML

Received: 15 July 2003 / Accepted: 25 May 2004 / Published online: 7 October 2004

Abstract Despite the status of united modeling language (UML) as the de facto standard for object oriented modeling, it has received controversial reviews. The most controversial diagram in UML is the use case diagram. Some practitioners claim that use case diagrams are not valuable in requirements analysis and some have even argued that use case diagrams should not be part of UML. This research examined the *values* of use case diagram in interpreting requirements when use case diagrams are used in conjunction with class diagrams. In other words, the study investigated the possible synergetic values and relationships between the use case and class diagrams in the context of requirements analysis. This study used theories from cognitive psychology as its theoretical and conceptual foundation. The data collection utilized the verbal protocol technique in which subjects were asked to think aloud as they interpreted the use case and class diagrams. The results show that the use case diagrams were more completely interpreted than the class diagrams. The presence or absence of one diagram when interpreting another diagram had no effect on the outcome of the interpretation. From the results, we argue that the use case diagrams and class diagrams depict different aspects of the problem domain, they have very little overlap in the information captured, and both are necessary in requirements analysis.

Keywords Requirements analysis · Requirements engineering · Conceptual modeling · Unified modeling language · Use case diagram · Class diagram · Experimental study

K. Siau (✉) · L. Lee
Department of Management, University of Nebraska-Lincoln,
209 College of Business Administration,
Lincoln, NE, 68588-0491, USA
E-mail: ksiau@unl.edu

1 Introduction

Since the late 1990s, unified modeling language (UML) has emerged as the software industry's dominant modeling language. However, the use case centric approach of the UML modeling language has been controversial. The rational unified process (RUP) for UML has three defining features: use case driven, architecture-centric, as well as iterative and incremental processes. The use case driven feature is the core of the RUP. Use cases surfaced as an excellent way to drive requirements capture, analysis, and high-level design [5]. The use case centric approach is, however, often challenged.

The literature reveals that there are critical opinions and disapproving views besetting the role of use cases. For instance, Krogstie [14] questioned the domain appropriateness of use cases. Dobing and Parsons [9] highlighted the controversy between the "naturalness" of the object models involved and the idea of use cases facilitating communication and requirement verification. They also pointed out that advocates of use cases do not offer empirical evidence of use cases being "good" mechanisms for communicating with users. "Goodness of use cases could be established in relative sense by comparing them to other mechanisms for communicating the same information with users" [9].

This research investigated the roles and values of the use case diagram in understanding requirements depicted by the use case and class diagrams. It should be noted that the goal of the research is not to compare use case and class diagrams. They have different information content and they cannot be made informationally equivalent for comparison purposes [23]. Rather, the objective of this research is to investigate whether these two diagrams are able to complement each other in the context of understanding system requirements.

The rest of the paper is organized as follows: Sect. 2 provides a literature review on the use case and class diagrams. Section 3 motivates the research and lists the research questions. Section 4 discusses the theoretical foundation for this research and proposes the research

hypotheses. Section 5 describes the research design and the research procedure. Section 6 presents the results and discusses the findings. The last section, Section 7, discusses the implications of the findings and concludes the paper.

2 Literature review

2.1 Use case diagram

The main purpose of use case diagrams is to visualize use cases in the system. Elements of the use case diagram include actors, use cases, and relationships. The use case diagram shows not a single use case, but a series of use cases for a given system, and each use case can be further annotated with textual description. The use case diagrams combine both UML notations and narrative text. In general, use case diagrams distinguish themselves as a highly text-based communication tool, which focuses on transactions from the user's perspective [9]. Use cases, nevertheless, are requirements analysis and modeling tools that should describe **"what"**, not **"how"** [9].

In RUP, use cases serve as a guiding technique to deal with all types of user aspects [12]. Use cases provide an inventory of the kinds of interactions that could occur between users and a system, thus providing a forum for domain experts, end users, and developers to communicate with one another [6].

Nevertheless, the use of use cases and use case diagram is controversial. Rosenberg and Scott [19] stressed that one of the early steps in object modeling is building a use case model. Kulak and Guiney [15] argued that use cases are the drivers for the rest of the UML diagrams. Maciaszek [16] stated that use case diagrams and class diagrams are the most important specification techniques in object-oriented analysis.

On the other hand, Evans [11] argued that use cases were not part of the design process and that implementation could not be done on the basis of use cases solely. Dobing and Parsons [9] identified several problems with both the application and the theoretical underpinnings of use cases. For example, they highlighted three potential issues: (1) a high degree of variety in the level of abstraction of use cases, (2) the controversy regarding the "naturalness" of object models, and (3) the ideas of use cases facilitating communication and requirements verification with users. Thus, the roles and values of the use case diagram are unclear and debatable.

2.2 Class diagram

A class is a description of a group of objects with similar properties, common behaviors, common relationships, and common semantics [19]. Class diagrams represent the classes (or modules) that comprise a system and the functions supporting the system, excluding dynamic information, by showing what classes they are and how they are related, but not how they interact to achieve particular behaviors [18]. In short, a class is the descriptor of a set of objects with the same attributes and operations. It specifies three aspects; state, behavior, and object state changes.

Class diagrams describe a system domain in terms of the kinds of objects within a domain, the attributes, the behavior that objects can exhibit, and the associations among these kinds of objects. As Jacobson et al. [13] noted, "people regard their environment in terms of objects." Therefore, it is simple to think in the same way when designing a model. A model designed using an object-oriented approach is often easy to understand, as it can be directly related to reality. Thus, with such a design method, only a small semantic gap will exist between reality and the model. Booch [4, p. 39] further stressed that "in quality object oriented system, you will find many classes that speak the language of the domain expert." These rationales are centered on an avowal of the naturalness and the ease of understanding of classes.

Class diagrams, however, also have their critics. For instance, Maciaszek [16] pointed out that classes are chronically difficult to find, and the properties of classes are not always obvious. Dobing and Parsons [9] noted that "there are few empirical studies addressing the ability of users to understand class models." They also postulated that class diagrams alone might not be appropriate for communicating and verifying requirements. However, they added that it might be appropriate to use class diagrams directly as a mechanism for communicating and verifying the structure of application domain with users. Vessey and Conger [28] also argued that objects might not provide such a "natural" way of thinking about a problem domain.

3 Research questions

The literature review clearly shows that there exist controversial views regarding the role of class diagrams and use case diagrams during requirements analysis. One of the key controversies is the *values* of use case diagrams in requirements analysis. Are use case diagrams valuable in requirements analysis? This question, however, cannot be settled by argument. Empirical research is warranted to provide evidence and shed light on this issue.

This study addresses the following research questions:

1. What are the informational roles and values of use case diagrams and class diagrams in the context of requirements analysis?
2. Is there any complementary effect between use case diagrams and class diagrams, i.e., does the usage of both models result in a more complete understanding of the problem domain?

4 Theoretical foundation and research hypotheses

4.1 ACT-R theory

This study finds its theoretical underpinnings in cognitive psychology that is proposed as a reference discipline for systems analysis and design research by Siau [21–23], Siau et al. [26], and Siau and Tian [25]. A key model in cognitive psychology is the well-known human information-processing model ACT-R [1], a revised version of the ACT model. ACT-R consists of a theory of the nature of human knowledge, and how this knowledge is deployed and acquired [2]. In brief, ACT-R consists of *declarative* and *procedural* long-term memory, and a *goal stack*. Figure 1 shows the ACT-R model.

A reference theory related to the ACT-R model is the theory of schemata and scripts, which falls under the dichotomy of propositional-based theories. Propositional-based theories posit that knowledge is represented in terms of propositions. Incoming new facts derived or inferred from the propositions are compared with stored knowledge. Subsequently, any similarity between these two knowledge dimensions is accessed. The theory of schemata and scripts is commonly used to provide context-dependent problem-solving explanations for human cognitive processes [29].

4.2 Schemata and scripts

Anderson [1] pointed out that "ACT can simulate the operation of any schema by the operation of some production set." Velenueve and Fedorowicz [29] also wrote "declarative knowledge can be said to correspond to schemata and procedural knowledge to scripts." Schemata and scripts represent different knowledge structures. In general, schemata are static representations of concepts that model the world, while scripts are structures that put schemata into action. Schemata embody prototypical expectations about objects, situations, events, and actions. Schemata serve not only as representation of knowledge but also act as "filters" to extract knowledge from memory. Via schemata, individuals can make accurate inferences and predictions by capitalizing on the regularities of situations. Schemata are powerful because they help bring in knowledge relevant to the current environmental situation. In the context of schemata, instead of accessing many highly specific pieces of knowledge separately, one can easily find a single schema that contains sufficient information to permit adequate interpretation of the situation [29]. All schemata share a common feature—encompassing variables and representing knowledge on all levels of abstraction. According to Smith [27], a schema has an attribute-value format and for each attribute, there are possible values that instances of the concept can assume. Within each schema, relations between attributes are specified. Also, each schema indicates the type or superset from which the concept is derived.

Scripts are sequences of episodes that take place over time. Scripts essentially submit to spatial and temporal dimensions. They imply interpretation of events executed at the moment of their instantiation. Due to the nature of event sequencing, scripts support "story-telling" behaviors. Scripts also complement a feature lacking in schemata—incomplete structures. Schemata

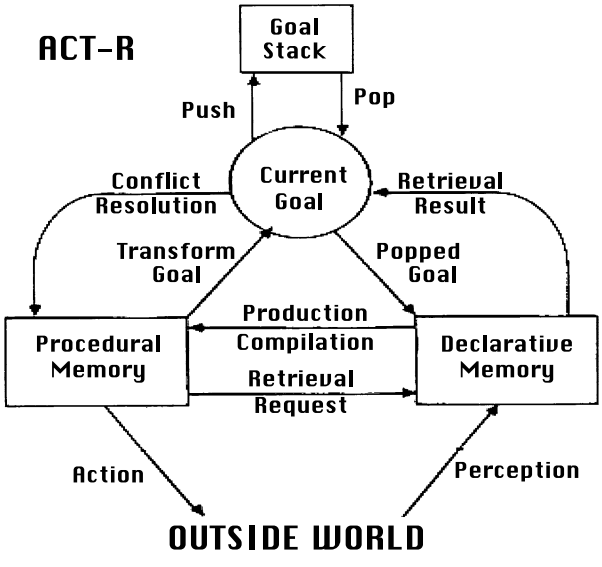


Fig. 1 ACT-R model

Table 1 Summary of schemata and scripts structures

Cognitive structure	Features
Schemata	Hierarchically organized Attribute-value format Attributes have a list of possible values (value domain) Each attribute has a default value Some attributes can be interdependent Each attribute has a weight indicating the attribute's relative importance A schema has an indication of its class (concept) membership Schemata bear a notion of contextual information Schemata filter incoming information when instantiated Schemata represent knowledge at all levels of abstraction Schemata can embed in each other (subschemata) Abstract schemata exist at the top of the hierarchy and have leaner attribute lists
Scripts	Action oriented knowledge structures Scripts sequence the steps in mental processing Some are pure action scripts that automate psychomotor behavior Main focus is on the sequencing of events Role is to put schemata into action Scripts' temporal scope is large (temporal units are large) Scripts help in selecting the most appropriate schemata Scripts help in confirming selected schemata

Table 2 Comparison of schemata and class diagrams

Key concepts	Schemata	Class diagrams
A single occurrence	Schema	Instance
A collection of occurrences with common features	Class	Class
A collection of occurrences with different features	Composite schema	Aggregation
A descriptive feature	Attribute	Attribute
Inheritance relationship	Inheritance	Generalization
Links between objects	Embedded schema	Associations

Table 3 Comparison of scripts and use case diagrams

Key concepts	Scripts	Use case diagrams
Action oriented knowledge structures	Scripts	Use case diagrams
Sequencing of events	Spatial-temporal dimension of scripts	Connection among use cases
Put schemata into action	Spatial-temporal dimension of scripts	Connection among use cases

categorize knowledge but tend not to support action. The spatial-temporal dimension of scripts promotes change and helps to depict concepts in ''action''. They subsequently adjust solutions to the problem accordingly.

A summary of schemata and scripts is produced in Table 1. Also, a comparison of Schemata and class diagrams is presented in Table 2. Table 3 shows a comparison of Scripts and use case diagrams.

Although the structures of knowledge are different, schemata and scripts complement each other well. Combined, both schemata and scripts represent a powerful learning mechanism. Villeneuve and Fedorowicz [29] posited that both schemata and scripts are necessary and they complement one another. As seen from Tables 2 and 3, schemata correspond closely to class diagrams and the functionalities of scripts are similar to those of use case diagrams. Thus, it can be argued that use case diagrams and class diagrams, like schemata and scripts, should complement one another.

4.3 Problem space theory

Problem-space theory is also relevant to explaining the cognitive activities involved in interpreting information system models. A problem-space representation allows the description of a problem, using a search strategy (mean–ends analysis is the most popular strategy). This approach is powerful, yet simple, for it corresponds to human problem-solving domains while fitting well with computer implementation [17]. We subscribe that the problem space of a problem domain is broader when either diagram (use case diagram or class diagram) is used alone. Supplementing one diagram with another naturally shrinks the problem space, therefore making the interpretation easier and more precise because the problem space is smaller. The reduced problem space also lowers the probability of making false problem-solving solutions, while improving the accuracy of interpretation. The problem-space theory again points to the complementing effects of use case diagram and class diagram when these diagrams are used together.

4.4 Research hypotheses

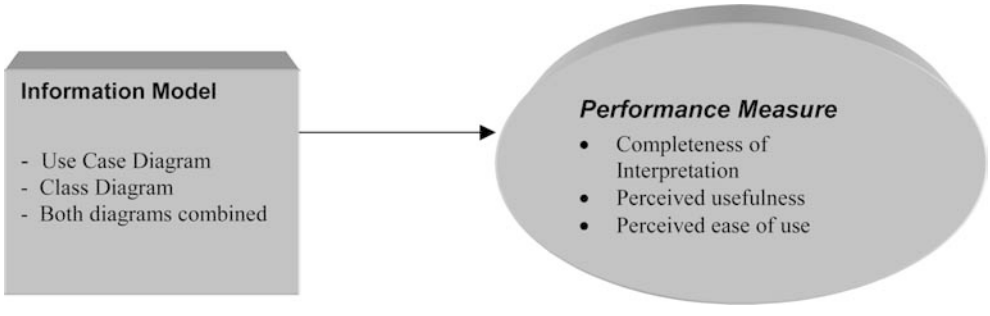
Based on the above discussion, we put forth the following hypotheses:

- H1: The completeness of interpreting class diagrams and use case diagrams is different.
- H2a: The inclusion of use case diagrams affects the completeness of the problem domain interpretation using class diagrams.
- H2b: The inclusion of class diagrams affects the completeness of the problem domain interpretation using use case diagrams.
- H3: The sequence combination of the diagrams affects the completeness of the problem domain interpretation.
- H4: Perceived Usefulness is different between use case diagrams and class diagrams.
- H5: Perceived Ease of Use is different between use case diagrams and class diagrams.

5 Research design and framework

In this research, we focused on model interpretation. Siau et al. [26] argued that a model interpretation task is a task of considerable, though manageable, complexity and is reasonably within the range of analytic tools we

Fig. 2 Research design



have available from behavioral research; because of the intrinsic importance of the task itself, and its tractable complexity, studies on information model interpretation are a natural starting point in the behavioral study of information modeling. The research design is shown in Fig. 2.

5.1 Research methodology

Experiments capturing subjects' performance via questionnaires and process-tracing method [10] were carried out. Process-tracing techniques have been commonly used in information requirements specification studies (e.g., [21, [28]). The key advantage of using process-tracing techniques is that the data captured is far richer than data from input–output analysis [28]. Toward the end of the experimental session, subjects were asked to fill up questionnaires on the perceived usefulness and perceived ease of use [8] of the use case diagrams and class diagrams.

5.2 Subjects

The subjects were university student volunteers who had completed at least one object-oriented UML course. Thirty-one subjects were recruited to participate in the study. These subjects were randomly assigned to one of the two treatment groups during the experiment. Due to a technical problem during an experimental session (audio recorder malfunctioning), the verbal protocol of one subject was not properly captured and the data from that subject was discarded from analysis.

A scatter plot diagram and a histogram were created to test for sample normality. The results indicated a non-normal distribution. Chebyskev's Rule was applied to determine possible outlier(s) in the data set. According to Brightman and Schneider [7], for non-normal data, data would rarely fall more than four standard deviations from the group mean. No subject fell outside the four standard deviations although one subject was very close. As such, a total of 30 subjects were used for the data analysis.

Table 4 Experimental design

Treatment	Domain	
	Domain A (ATM)	Domain B (music club)
Treatment no. 1	Sequence 1: CD → CD + UC	Sequence 2: UC → UC + CD
Treatment no. 2	Sequence 2: UC → UC + CD	Sequence 1: CD → CD + UC

Domain A is a bank ATM case
Domain B is a music club
UC stands for use case diagram
CD stands for class diagram

5.3 Experimental design

The independent variables are: the models (use case diagram, class diagram, and both diagrams). The dependent variable is the subject's performance in interpreting the requirements depicted by the models. Table 4 depicts the experimental design.

5.4 Data collection methods and data analysis

Verbal protocol analysis [10], in which subjects were asked to verbalize their analysis, was administered. Subjects were asked to think aloud as they interpreted the use case diagrams and class diagrams. Verbal utterances of the subjects during the experiment were the main source of data collection and subsequent analysis. The experimental session was audiotaped and later transcribed, coded, and analyzed. Specifically, the taped protocol analysis processes of the subjects were transcribed to MS word format and itemized.

Itemized information chunks were then coded against a listing of information chunks depicted on the use case diagrams and class diagrams. This listing was produced based on the case analysis of the original source and verified by two experienced UML instructors. Mapped chunks of the diagrams were quantified, counted, and normalized to enable a standardized comparison of the final data scores.

5.5 Task domain

Two problem domains, each varying in complexity, were used in the experiment. Variation of the problem domain served to achieve generalization of the analysis results across varying problem domains in practice. Problem domain A was a system analysis case for an ATM banking system. Problem domain B was a system analysis case for a music club. Problem domain B was more complex than problem domain A. System models, i.e., the use case diagram and the class diagram, for these two domains were adopted from existing published materials to enforce internal reliability of the experimental design. The problem domain A was adopted from *Object Oriented Systems Development using the Unified Modelign Language* by Bahrami [3]. The problem domain B was adopted from *System Analysis and Design Methods* (5th Edition) by Whitten et al. [30].

5.6 Perceived usefulness and perceived ease of use

There are many variables that affect the acceptance or rejection of information technologies or systems. Two determinants that are especially important are perceived usefulness and perceived ease of use. First, perceived usefulness refers to the tendency of the people using a system to believe it will help them perform their job

Table 5 Results of hypotheses testing

Hypothesis	*p*-value	Supported
H1: The completeness of interpreting class diagrams and use case diagrams is different	<0.001	Y
H2a: The inclusion of use case diagrams affects the completeness of the problem domain interpretation using class diagrams	>0.05	N
H2b: The inclusion of class diagrams affects the completeness of problem domain interpretation using use case diagrams	>0.05	N
H3: The sequence combination of the diagrams affects the completeness of the problem domain interpretation	>0.05	N
H4: Perceived Usefulness is different between use case diagrams and class diagrams	>0.05	N
H5: Perceived Ease of Use is different between use case diagrams and class diagrams	>0.05	N

better [8]. It is defined as "the degree to which a person believes that using a particular system would enhance his or her job performance" [8, p. 320]. Second, perceived ease of use refers to "the degree to which a person believes that using a particular system would be free of effort" [8, p. 320]. Subjects were asked to respond to the questionnaires reflecting their responses on perceived usefulness and perceived ease of use for the two diagram types (i.e., class diagram and use case diagram) at the end of the experimental session.

Both measures of "perceived usefulness" and "perceived ease of use" use a Likert-type seven-point response format where 1 = "strongly agree," 2 = "moderately agree," 3 = "slightly agree," 4 = "neutral," 5 = "slightly disagree," 6 = "moderately disagree," and 7 = "strongly disagree."

The perceived usefulness and perceived ease of use questionnaires consist of six questions each. For perceived usefulness, the questions are (1) accomplishes requirement analysis more quickly, (2) improves requirement analysis performance, (3) increases productivity in requirement analysis, (4) enhances effectiveness in requirement analysis, (5) makes it easier to do requirement analysis, and (6) useful in requirement analysis. For perceived ease of use, the six questions are (1) need to consult modeling manual and/or reference, (2) easy to model what I want to, (3) easy to understand, (4) rigid and inflexible to understand, (5) easy to remember how to do requirement analysis, and (6) easy to use.

6 Results and discussions

The final data was obtained from counts of matching information elements identified by the subjects during protocol analysis. To assure an identical measuring standard, the counts were normalized—denominators were total counts of information elements for the independent problem domains. These normalized data were then collapsed into a single data count for the respective diagram type—class diagram or use case diagram. As the percentiles of information elements' counts were normalized per diagram, the basis of comparison should be equivalent. These percentiles became the performance scores and were used for the statistical analysis. ANOVA and *T*-test analysis were used for the statistical testing. Table 5 summarizes the statistical results.

6.1 Effect of the diagram type analysis

The analysis of the effect of the diagram types on the completeness of diagram interpretation was done using an ANOVA analysis, and it indicated a significant statistical difference of 0.000 ($p < 0.001$). This means that there is a significant difference between the two diagram types. Hence, the result supported Hypothesis H1: The completeness of interpreting class diagrams and use case diagrams is different. Since the use case diagram has a higher sample mean than class diagram (0.8655 vs. 0.6464), it means that use case diagrams were interpreted more completely than class diagrams.

One explanation of the results is the complexity of class diagrams. The complexity analysis study conducted by Siau and Cao [24] indicated that class diagram is the most complex diagram in UML. Specifically, class diagrams have the highest complexity in terms of relationship types, property types per technique, and role types. Cognitive propositions on human memory infer that the complexity of a problem affects the immediate memory capacity involved in solving the problem. As a result, class diagrams may not be as completely interpreted as use case diagrams.

6.2 Effect of diagram types combination

Two scenarios were examined. The difference scores between class diagrams alone and class diagrams when combined with the subsequent use case diagram were computed i.e., (CD − CD in (CD + UC)). The same applies to use case diagrams when response scores for use case diagrams used alone were compared to that of use case diagrams when class diagrams were added i.e., (UC − UC in (UC + CD)). The Paired Samples *T*-test explored the effect of the two scenarios proposed: (1) response scores for the class diagrams alone versus the

response scores for class diagrams after subsequent addition of use case diagrams, (2) response scores for use case diagrams alone versus the response scores for use case diagrams after subsequent addition of class diagrams. Contrary to initial expectation, neither of the paired samples showed a significant statistical difference. The comparison of class diagrams alone versus class diagrams after the inclusion of use case diagrams has a significance value of 0.129 ($p > 0.05$) while the comparison of use case diagrams alone versus the use case diagrams after the inclusion of class diagrams has a significance value of 0.095 ($p > 0.05$). Hence, we rejected the hypothesis H2a: the inclusion of use case diagrams affects the completeness of the problem domain interpretation using class diagrams, and hypothesis H2b: the inclusion of class diagrams affects the completeness of the problem domain interpretation using use case diagrams. Specifically, the inclusion of a subsequent diagram type does not affect the completeness of interpretation rendered by the initial diagram type. The complementary effect of the inclusion of either diagram type is not significantly different.

The results show that the inclusion of an additional diagram did not help the subjects to gain significantly more information on the given diagram. Thus, it appears that the information depicted by the two diagram types is sufficiently different and not overlapping. From the Problem Space Theory perspective, the problem space for use case diagram and the problem space for class diagram have little overlap or no overlapping. As a result, the additional information provided by the inclusion of another diagram did not help in interpreting the given diagram. From hindsight, this is not surprising as the use case and class diagrams are designed to capture different aspects of a problem domain.

6.3 Sequence combination effect

Sequence 1 employed the class diagram interpretation prior to the subsequent combination with the use case diagram (CD $\rightarrow$ CD + UC) while sequence 2 employed the use case diagram prior to the combination with the class diagram (UC $\rightarrow$ UC + CD). The ANOVA test on the effect of sequence combination indicates an insignificant statistical difference of 0.684 ($p > 0.05$). Hence the hypothesis H3: the sequence combination of the diagram interpretation affects completeness of the problem domain understanding is not supported.

Dobing and Parsons [9] stated that the process for moving forward from the use case diagram to identify classes is neither universally accepted, even among use case adherents, nor does it appear to be clearly defined or articulated. This argument is partially supported in our study as the sequence appears to have no effect on the interpretation of the diagrams.

6.4 Cronbach's alpha for perceived usefulness and perceived ease of use

Cronbach's Alpha was computed for the Perceived Usefulness and Perceived Ease of Use questionnaires. Prior to the item adjustment, the Perceived Usefulness questionnaire had a Cronbach's Alpha of 0.9462. However, Cronbach's Alpha reliability estimate for the Perceived Ease of Use questionnaire was not as high. In fact, it showed a somewhat low reliability of 0.6560. Based on the reliability analysis for Perceived Ease of Use questionnaire, it is justifiable to delete item (4), because the item-total correlation is as low as 0.1571 and the alpha will increase to 0.7115 if this item is deleted. Deflation of the item-total correlation for item (4) may have occurred due to its reversal phrasing. Item (4) is a reversal item and would have possibly confused the subjects. Subsequently, final scores for the five remaining questions and the final total scores for the Perceived Ease of Use questionnaire were revised after deletion of item (4).

6.5 Perceived usefulness

The ANOVA analysis shows an insignificant value of 0.474. Hence, there is no statistical difference between class diagrams and the use case diagrams for perceived usefulness. This does not support the hypothesis H4: Perceived Usefulness is different between use case diagrams and class diagrams. The scores of perceived usefulness are mostly three or below (i.e., a lower number represents a higher perceived usefulness). In other words, the subjects perceived both the use case and class diagrams to be useful in helping them to understand the problem domain.

6.6 Perceived ease of use

The ANOVA test shows a nonsignificant level of 0.613 at p-level > 0.05. In other words, there is no significant difference in the perceived ease of use between class diagrams and use case diagrams. hypothesis H5: Perceived Ease of Use is different between use case diagrams and class diagrams is not supported.

This result is somewhat surprising because the use case diagram is less complex than the class diagram [24]. Also, the interpretation of the use case diagram is more complete than the class diagram (i.e., Hypothesis 1). One possible explanation is that responses are perceptual in nature and the result may be caused by limitation of self-reported responses. Most subjects scored three or less on a seven-point scale (i.e., a lower number represents a higher perceived ease of use). This may indicate a possible ceiling effect—both diagrams were perceived to be very easy to use and the scale could not capture the difference. The ceiling effect is another possible explanation. As these were subjects with at least a semester of

training in UML, they might have perceived both diagrams as easy to use.

6.7 Limitations of the study

Due to the labor-intensive nature of protocol analysis, a small sample size was used. The majority of verbal protocol studies have utilized a relatively small sample size [20]. A small sample size reduces statistical power for comparison. However, the statistical power problem was partly alleviated by testing the subjects with two pairs of diagrams. The order of presenting the diagrams was alternated to reduce any possible order bias. Another limitation associated with the protocol analysis is the coding process. The coding process was not automated but was performed manually by human coder(s). Biases could have been introduced into the study results although care was taken to ensure objectivity in the coding process. Despite these potential limitations, analysis of protocol data has proved to be effective in providing us useful data to investigate the research questions.

7 Implications and conclusions

This research centers on investigating the roles of use case diagrams and class diagrams in requirements analysis. The findings of this research have some implications for practitioners. The results of the study show that use case diagrams were interpreted more completely than class diagrams. This seems to imply that use case diagrams are easier to interpret than class diagrams, thus enabling the subjects to attain a more complete understanding of the model. The study by Siau and Cao [24] also found that class diagrams and use case diagrams differ in their diagrammatic complexity. Class diagrams have a higher diagrammatic complexity. Use case diagrams are comparatively less complex and presumably easier to understand. This suggests that it is probably better to discuss systems requirements with end users using use case diagrams rather than class diagrams. Kulak and Guiney [15] also stressed that the simplicity of use case diagrams makes them a great communication tool. The lower complexity level of use case diagrams will facilitate end users' understanding of the systems.

For instructors who are teaching UML, the results of this study suggest that it may be good to start off with teaching the use case diagram as it is probably easier for students to comprehend than the class diagram.

For researchers in UML, we suggest that they consider replicating this study or carrying out similar studies using different research methodologies. For example, similar studies based on novice and experienced users should be carried out to validate the outcome of the present findings. It would also be interesting to consider investigating the rest of the modeling diagrams in UML, such as activity diagram, sequence diagram, and statechart diagram. Determining the core UML diagrams and the core constructs in each diagram are other interesting topics in the area. As UML becomes more and more complex with each version, there is a need to focus on the core diagrams and the core constructs—particularly in teaching. Research aiming to identify the core of UML may also help to contribute to the development of UML standard.

The insignificant statistical results of some of the hypotheses may point to the fact that class diagrams and use case diagrams capture different aspects and views of the problem domains. As such, we assert a need for the coexistence of class diagrams and use case diagrams for effective requirements analysis. Given that there is no significant difference between the sequences of interpretation, we may argue that the order in which the diagrams are used or constructed during requirements analysis may not be important. One may even argue that both diagrams may need to be constructed concurrently and modified iteratively.

Finally, we believe that both use case and class diagrams are valuable in the requirement analysis process. As the information content of use case and class diagrams are different and arguably complementary, two is better than one!

References

1. Anderson JR (1983) The architecture of cognition. Harvard University Press, Cambridge
2. Anderson JR, Lebiere C (1998) The atomic components of thought. Erlbaum, Mahwah
3. Bahrami A (1999) Object oriented systems development using the unified modeling language. Irwin McGraw-Hill, Boston
4. Booch G (1996) Object solutions: managing the object oriented project. Addison-Wesley, Reading
5. Booch G (1999) UML in action. Commun ACM 42(10):27–28
6. Booch G, Rumbaugh J, Jacobson I (1999) The unified modeling language user guide. Addison-Wesley, Reading
7. Brightman H, Schneider H (1992) Statistics for business problem solving. South-Western Publishing Co, Cincinnati
8. Davis F (1989) Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q September :319–339
9. Dobing B, Parsons J (2000) Understanding the role of use cases in UML: a review and research agenda. J Database Manag 11(4):28–36
10. Ericsson K, Simon H (1993) Protocol analysis: verbal reports as data, rev edn. MIT Press, Cambridge
11. Evans GK (1999) Why are use cases so painful? Thinking Objects 1(2). http://evanetics.com/articles/Modeling/UCPainful.htm. Cited on 30 September 2004
12. Hesse W (2000) RUP—a process model for working with UML? Critical comments on the rational unified process. In: Siau K, Halpin T (eds) Unified modeling language: systems analysis, design, and development issues. Idea Group Publishing, Hershey
13. Jacobson I, Christerson M, Jonsson P, Overgard G (1992) Object oriented software engineering: a use case driven approach. Addison-Wesley, Reading
14. Krogstie J (2000) Using a semiotic framework to evaluate UML for the development of models of high quality. In Siau K, Halpin T (eds) Unified modeling language: systems analysis, design, and development issues. Idea Group Publishing, Hershey

15. Kulak D, Guiney E (2000) Use cases—requirements in context. Addison Wesley, Reading
16. Maciaszek LA (2001) Requirements analysis and system design. Developing information systems with UML. Addison-Wesley, Reading
17. Mayer R (1991) Thinking, problem solving, cognition. W.H. Freeman
18. Pooley R, Stevens P (1999) Using UML: software engineering with objects and components. Addison-Wesley, Harlow
19. Rosenberg D, Scott K (1999) Use case driven object modeling with UML: a practical approach. Addison-Wesley, Reading
20. Schenk KD, Vitalari NP, Davis KS (1998) Differences between novice and expert system analysts: what do we know and what do we do? J Inf Syst 15(1):9–50
21. Siau K (1996) Empirical studies in information modeling: interpretation of the object relationship. Unpublished PhD Dissertation, University of British Columbia
22. Siau K (1999) Information modeling and method engineering: a psychological perspective. J Database Manag 10(4):44–50
23. Siau K (2004) Informational and computational equivalence in comparing information modeling methods. J Database Manag 15(1):73–86
24. Siau K, Cao Q (2001) Unified modeling language—a complexity analysis. J Database Manag 12(1):26–34
25. Siau K, Tian Y (2001) The complexity of unified modeling language—a GOMS analysis. In: 14th international conference on information systems (ICIS'01), New Orleans, 16–19 December 2001, pp 443–448
26. Siau K, Wand Y, Benbasat I (1997) The relative importance of structural constraints and surface semantics in information modeling. Inf Syst 22(2/3):155–170
27. Smith EE (1989) Concepts and inductions. In: Posner MI (ed) Foundations of cognitive science. MIT Press, Cambridge
28. Vessey I, Conger S (1994) Requirements specification: learning object, process, and data methodologies. Commun ACM 37:102–113
29. Villeneuve A, Fedorowicz J (1997) Understanding expertise in information systems design, or, what's all the fuss about objects? Decis Support Syst 21:111–131
30. Whitten JL, Bentley LD, Dittman KC (2001) System analysis and design methods. McGraw-Hill Irwin, Boston