ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/221536273

UML Class Diagram Syntax: An Empirical Study of Comprehension.

Conference Paper · January 2001

Source: DBLP

CITATIONS
38

READS
18

5 authors, including:

Matthew John McGill
University of New South Wales
15 PUBLICATIONS **382** CITATIONS

SEE PROFILE

Available from: Matthew John McGill
Retrieved on: 10 May 2016

UML class diagram syntax: an empirical study of comprehension

Helen C. Purchase[*], Linda Colpoys[*], Matthew McGill[*], David Carrington[*] and Carol Britton[†]

[*]School of Information Technology and Electrical Engineering
University of Queensland
St Lucia, Brisbane 4072, Queensland
[†]Faculty of Engineering and Information Sciences
University of Hertfordshire
Hatfield, England

{hcp, davec}@itee.uq.edu.au

Abstract

Despite UML being considered a software engineering standard, the UML syntactic notations used in texts, papers, documentation and CASE tools are often different. The decision as to which of the semantically equivalent notational variations to use appears to be according to the personal preference of the author or publisher, rather than based on any consideration of the ease with which the notation can be understood by human readers. This paper reports on an experiment that takes a human comprehension perspective on UML class diagram notational variants. Five notations were considered: for each, two semantically equivalent, yet syntactically different, variations were chosen from published texts. Our experiment required subjects to indicate whether a supplied specification matched each of a set of experimental diagrams. The results reveal that the best performing notation may depend on the task for which it is used, and that our personal, intuitive predictions intuitions (which were based in the complexity of the notation) were partly confirmed.

Keywords: UML class diagrams, notation, human performance.

1 Introduction

In recent years, the Unified Modelling Language (UML) has emerged as the defacto standard for the representation of software engineering diagrams (Rumbaugh et al. 1999). While adopted as a standard by the membership of the OMG (Object Management Group) in 1997 (Rumbaugh et al. 1999), a glance through different texts that use UML, or an investigation of current CASE tools, reveals a number of notational variations (Page-Jones 2000, Fowler et al. 2000, Evitts 2000, Rational Rose 2001). As a standard, UML is still evolving, and within each version of the standard are many semantically equivalent notational variations from which authors or publishers may choose, according to their personal preference. There seems to be no basis for the choice of one notational variation over another.

It may be that one notational variation is easier to comprehend than another. The research reported in this paper attempts to determine whether there are any comprehension differences between five different notational variations for UML class diagrams, through an empirical study of subjects' performance on a specification matching task. The notational variants may be equally as good as each other, in which case, it does not matter which one is used. But, if there are comprehension differences, the results of this study could assist in the definition of appropriate notational standards (from a human understanding point of view), and in determining the most suitable notation to use in UML texts, CASE tools, and practical software engineering tasks.

Some theoretical analytical work has been performed on software engineering notations and visual programming languages, but no empirical studies on human comprehension of these notations have been found. While software engineering notations (including UML) have been analysed and compared using the Cognitive Dimensions framework (Green and Petre 1996, Blackwell and Green 2000) this framework is theoretical, and these analyses, while providing interesting structured perspectives on the notational features, are not based on experimental data (Kutar et al. 2000, Cox 2000, Gurr and Stevens 1999, Gurr and Tourlas 2000). The experiment reported here complements such analytical work.

1.1 Experimental aim and definitions

Our previous study (Purchase, Allder and Carrington 2000) considered user preferences for some UML notations and layout features. The experiment reported here concentrates on user performance, rather than preference.

The aim of this experiment was to determine which variant of each of five notations used in UML class diagrams is the more suitable with respect to human performance. By asking subjects to perform comprehension tasks on a number of semantically equivalent UML class diagrams that vary with respect to the notations used, we aimed to identify the notational variants that resulted in the best performance.

Copyright © 2001, Australian Computer Society, Inc. This paper appeared at the Australian Symposium on Information Visualisation, Sydney, December 2001. Conferences in Research and Practice in Information Technology, Vol. 9. Peter Eades and Tim Pattison, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

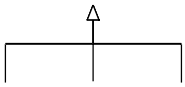
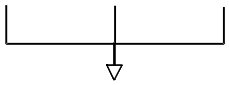
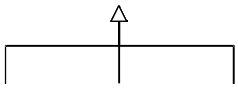
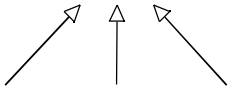
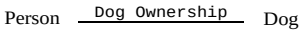
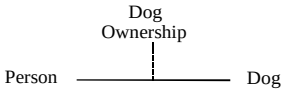
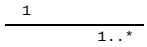
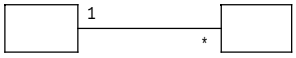
Notational Difference	Variation (a)	Variation (b)
Inheritance direction (N1)	 (Page-Jones 2000)	 (Purchase, Allder & Carrington. 2000)
Inheritance arcs (N2)	 (Page-Jones 2000)	 (Rumbaugh et al. 1999)
Association representation (N3)	 (Evitts 2000)	 (Evitts 2000)
Association names (N4)	(Evitts 2000)	(Gurr and Stevens 1999, Rational Rose)
Cardinalities (N5)	 (Rumbaugh et al. 1999)	 (Page-Jones, 2000)

Figure 1: The five notational variations.

1.2 UML class diagrams

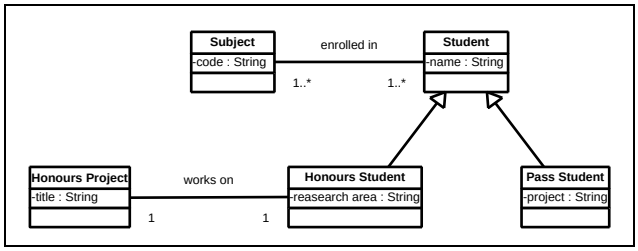


Figure 2: An example class diagram.

UML class diagrams are used to describe the static view of an application (Rumbaugh et al. 1999): the main constituents are classes and their relationships. A class is a description of a concept, and may have attributes and operations associated with it. Classes are represented as rectangles. A relationship between two classes is drawn as a line. Inheritance relationships indicate that attributes and operations of one class (the "superclass") are inherited by other classes (the "subclasses"), without needing to be explicitly represented in the subclasses themselves.

Figure 2 is an example of a small UML class diagram, showing the relationships between the classes in a simple university domain, including inheritance relationships between the student, pass student, and honours student classes.

1.3 Notational variations

Five independent notational variations for UML class diagrams were considered (see Figure 1): each notation (N1, N2, N3, N4, N5) has two variations (a) and (b), with identical semantics. All notational variations had been found in published documents on UML, or existing CASE tools, except N1(b), which was used in a prior experiment which considered user preference on notational variations (Purchase et al. 2000).

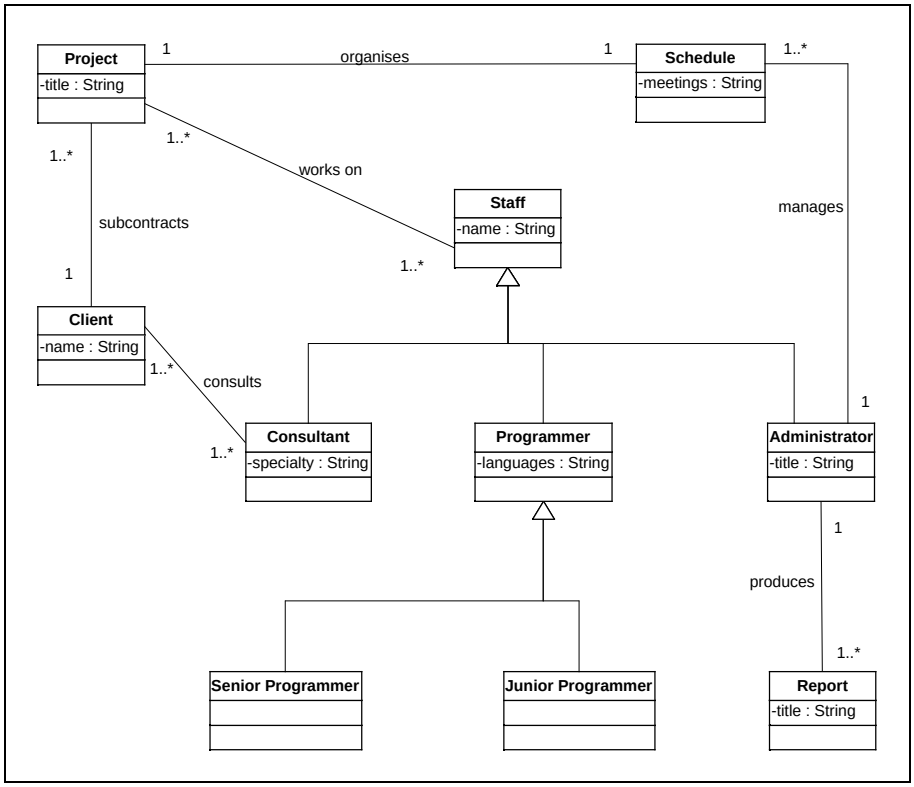


Figure 3: The application domain for the experiment.

Our informal predictions, based purely on our personal intuition and experience, were that, in each case, the (a) notation would produce better performance.

2 Experimental task:

The comprehension task was that of matching a given textual specification against a set of diagrams, indicating whether each diagram correctly matches the specification or not. The set of diagrams included both correct and incorrect diagrams.

2.1 Experimental materials:

2.1.1 The application domain

The class diagram used was based on a simple domain, which models a small Information Technology company that employs consultants, programmers and administrative staff to undertake projects for clients. The example includes 10 objects, 6 associations and 5 inheritance relations (see Figure 3).

A textual specification of this domain was produced in simple English. The subjects were asked to match the experimental diagrams against this specification.

2.1.2 UML tutorial and worked example

A tutorial sheet explained the meaning of UML class diagrams, and, using a simple example, described the semantics of all the notational variations. Subjects were not expected to have any prior knowledge of UML, and this tutorial provided all the UML background information they required for the experimental task. A worked example demonstrated the task that the subjects were to perform, by presenting a small specification with four different diagrams, and for each diagram indicating whether it matched the given specification or not. Care was taken to ensure that neither the tutorial nor the worked example would bias the subjects towards one notational variation over another.

2.2 The experimental diagrams

The UML diagram representing the experimental domain was drawn four times, in four differing layouts. Each layout was constructed with a view to minimising the potential for any confounding layout factors. Thus, each layout had a similar number of edge bends and sloping lines, no edge crosses, comparable orthogonality (the property of fixing nodes and edges to the intersections and lines of an invisible unit grid), and was of a similar size. Different layouts were required so that the subjects would not merely use visual pattern matching in performing the comprehension tasks: if all the diagrams had identical layout, the differences between them would be visually obvious and detectable without the subject needing to understand the information embodied in the diagram.

Using these four layouts of the diagram, the following experimental diagrams were produced:

2.2.1 Correct diagrams (24)

- 1 diagram using all the (a) notations, in four different layouts (N0t)
- 1 diagram using the (b) notation for N1, (a) notation otherwise, in four different layouts (N1t)

- 1 diagram using the (b) notation for N2, (a) notation otherwise, in four different layouts (N2t)
- 1 diagram using the (b) notation for N3, (a) notation otherwise, in four different layouts (N3t)
- 1 diagram using the (b) notation for N4, (a) notation otherwise, in four different layouts (N4t)
- 1 diagram using the (b) notation for N5, (a) notation otherwise, in four different layouts (N5t)

The naming convention of these diagrams is that the number (0-5) indicates which notation is being varied (where 0 represents that none of the notations is varied, that is, the (a) notation is used throughout), and the letter (t) indicates that these are true (correct) diagrams.

2.2.2 Incorrect diagrams (20)

- For N1: For two of the layouts of N0t, errors were introduced which affected the representation of inheritance (N0d1). For the same two layouts of N1t, the same errors were introduced (N1d). The error introduced was that a superclass was exchanged for a subclass.
- For N2: For two of the layouts of N0t, errors were introduced which affected the representation of inheritance (N0d2). For the same two layouts of N2t, the same errors were introduced (N2d). The error introduced was that one of the associations in the diagram was changed to be an inheritance relationship.
- For N3: For two of the layouts of N0t, errors were introduced which affected the representation of associations (N0d3). For the same two layouts of N3t, the same errors were introduced (N3d). The error introduced was that two of the association names were exchanged.
- For N4: For two of the layouts of N0t, errors were introduced which affected the representation of associations (N0d4). For the same two layouts of N4t, the same errors were introduced (N4d). The error introduced was that two of the association names were exchanged.
- For N5: For two of the layouts of N0t errors were introduced which affected the representation of cardinality (N0d5). For the same two layouts of N5t, the same errors were introduced (N5d). The error introduced was that all cardinalities on all associations were reversed.

The naming convention of these diagrams is that the letter (d) indicates that these are “dummy” (incorrect) diagrams. Diagrams beginning with N0 are those that use the (a) variation throughout, with the final number (1-5) indicating to which notation the diagram alteration relates. The other diagrams (N1-N5) indicate which notation has been varied to use (b) instead of (a).

2.3 Experimental procedure

2.3.1 Preparation

The students were given preparatory materials to read as an introduction to the experiment. These documents consisted of a consent form, an instruction sheet, a tutorial on UML class diagrams and notation, and a worked example of the experimental task.

As part of this document set, the subjects were also given the textual specification of the UML case study to be used in the experiment: this was the specification that they would need to match the experimental diagrams against. The subjects were asked to study this specification closely, and memorise it if possible,

The subjects were given 15 minutes to sign the consent form, read through and understand the materials, ask questions, take notes, or draw diagrams as necessary.

2.3.2 Online task

The subjects then used an online system to perform the experimental task. A copy of the text specification was placed in front of the computer for easy reference, and UML diagrams were presented in random order for each subject. The subjects gave a yes/no response to each presented diagram, indicating whether they thought the diagram matched the specification or not: two keys on the keyboard were used for this input.

16 practice diagrams (randomly selected from the 44 experimental diagrams) were presented first. The data from these diagrams was not collected, and the subjects were not aware that these diagrams were not part of the experiment. These diagrams gave the subjects an opportunity to practice the task before experimental data was collected.

The 44 experimental diagrams were then presented in a different random order for each subject, in blocks of eight, with a rest break between each block (the length of which was controlled by the subject). The final block was four diagrams in length.

Each diagram was displayed until the subject answered Y or N, or 40 seconds had passed. A beep indicated to the subject when the next diagram was displayed after a timeout (which was recorded as an error). The practice diagrams helped the subjects get used to the length of the allocated time period. The timeout period and the time needed for the subjects to prepare for the experiment were determined as appropriate through extensive pilot tests.

A within-subjects analysis was used to reduce any variability that may have been attributed to differences between subjects: thus, each subject's performance on one notation was compared with his or her own performance on the alternative notation. The practice diagrams and the randomisation of the order of presentation of the experimental diagrams for each subject helped counter the learning effect (whereby the subjects' performance on the task may improve over time, as they become more competent in the task).

2.3.3 Data collection

The response time and accuracy of the subjects' responses to the 44 experimental diagrams were recorded by the online system.

2.4 Subjects

The experiment was performed using 34 novice subjects and 5 expert subjects. The novice subjects were second and third year Computer Science and Information Systems students at the University of Queensland. The novice subjects were paid $15 for their time, and, as an incentive for them to take the experiment seriously, the best performer was given a CD voucher. The expert subjects were employees of the Distributed Systems Technology Centre, Brisbane, who volunteered their time.

3 Results

Both the speed and accuracy of each subject's responses were measured, enabling the analysis of two different measures of understanding. Analysis was performed on both the subjects' performance in identifying the correct diagrams and their performance in identifying the incorrect diagrams.

3.1 Novice Experiment:

3.1.1 Identifying the correct diagrams:

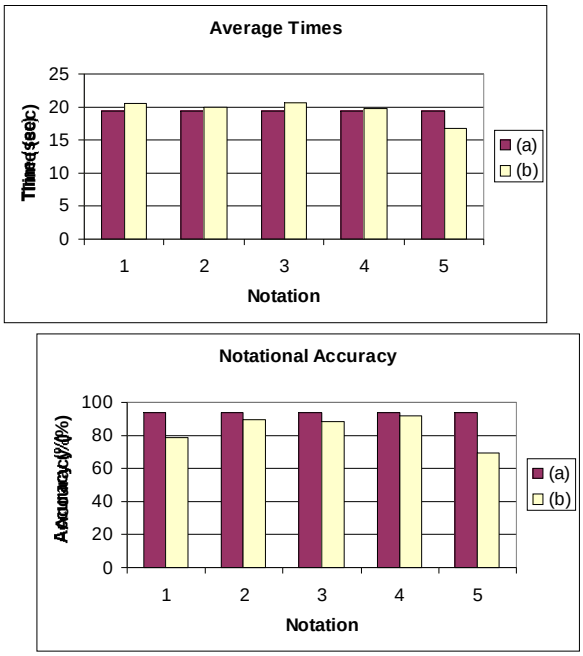


Figure 4: The response time and accuracy results novice users on correct diagrams.

3.1.2 Identifying the incorrect diagrams:

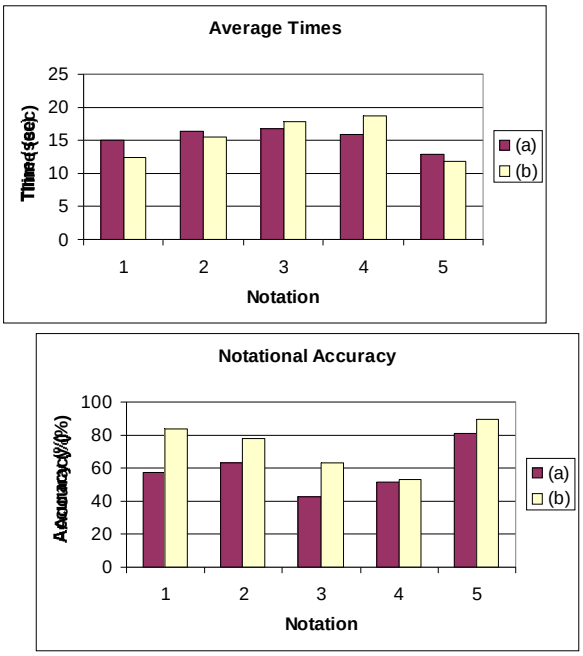


Figure 5: The response time and accuracy results novice users on incorrect diagrams.

Using a two-tailed t-test, the statistically significant results are:

- Notation 1: accuracy
  - for matching the specification to correct diagrams: (a) is better than (b) ($p < 0.05$)
  - for identifying an error in the diagrams: (b) is better than (a) ($p < 0.05$)
- Notation 1: time
  - for identifying an error in the diagrams: (b) is better than (a) ($p < 0.05$)
- Notation 2: accuracy
  - for identifying an error in the diagrams: (b) is better than (a) ($p < 0.05$)
- Notation 3: accuracy
  - for identifying an error in the diagrams: (b) is better than (a) ($p < 0.05$)
- Notation 4: time
  - for identifying an error in the diagrams: (a) is better than (b) ($p = 0.058$, approaching significance)
- Notation 5: accuracy
  - for matching the specification to correct diagrams: (a) is better than (b) ($p < 0.05$)
- Notation 5: time
  - for matching the specification to correct diagrams: (b) is better than (a) ($p < 0.05$)

3.2 Expert Experiment:

Five expert UML users performed the experiment; while both response time and accuracy were recorded, the small sample size meant that statistical analysis is impossible. However, we asked the five expert UML users to state, and explain, their preferences for the different notational variations.

3.2.1 Notation 1

Pointing the inheritance arrows upwards (variation (a)) was preferred by all the respondents, with the main reason being that it appears more natural to have the superclass (the more general object) above the subclass (the less general object) than to have the inheritance pointing downwards (variation (b)). The respondents revealed their assumptions underlying this view: that most people would read from top-to-bottom, and that it is important to identify the superclasses before the subclasses.

3.2.2 Notation 2

Four out of five respondents preferred the notation of joined inheritance lines (variation (a)) over the use of separate lines for each subclass (variation (b)).[1] The two main reasons for this choice was firstly that the joined inheritance notation demonstrates that the subclasses are on the same level of specialisation, and secondly, that in larger diagrams with more inheritance relationships, there is a potential for the diagram to "sprawl" and to look less "neat." The participant who preferred the (b) variation did so because "the diagonals require me to concentrate more when deciphering meaning."

3.2.3 Notation 3

All respondents preferred the (a) variation, where the name of each association is placed beside the line in the diagram, rather than in a separate association class linked to the line. The justifications of this choice related to the reduction of "clutter" ("(b) is too busy"), as well as the importance of making a clear visual distinction between objects and associations. One respondent pointed out that association classes are really only useful when there is additional information that needs to be recorded about the association: this was not the case in our small case study.

3.2.4 Notation 4

All respondents agreed that placing the label beside the line (variation (a)) is preferable to placing it over the line (variation (b)), for reasons of neatness and clarity ("(a) is more readable"). One respondent qualified his response by saying that if there were many more associations in the diagram, the (b) variation may be preferable, as it would be "clearer which text belongs to which line."

3.2.5 Notation 5

Variation (a) (making the cardinalities explicit) was preferred by all respondents, over the option of abbreviating the cardinalities (variation (b)), for the reason of improved clarity and reduction of ambiguity. There was confusion over whether the * meant 0..* or 1..*, the latter being what was stated in the experimental tutorial materials, while the former was what the experts assumed (and which is stated in Rumbaugh et al. 1999). Being explicit was strongly preferred ("It's clearer to put both upper and lower bounds to avoid this confusion").

4 Analysis

In performing the experimental task, the subjects would be looking for mistakes in each diagram, and, as soon as they identified a single mistake, would press the N key. The accuracy data therefore needs to be analysed with the following points in mind:

- We do not know the reason why subjects have rejected diagrams: it may be because they indeed have identified the planted errors relating to the notation in question, in which case, the rejection is valid.
- If subjects have a misunderstanding about one of the variations of a notation, then they may identify correct diagrams (incorrectly) as incorrect (by identifying what they believe to be an error), and incorrect diagrams (correctly) as incorrect (but for the wrong reason: an invalid rejection).
- As the (a) notations represented those variations that we considered to be intuitively the simpler, invalid rejections may be more likely to occur with the (b) notation, as this was the more complex.
- Invalid rejections relating to the (b) variation of any one notation would not affect the results for the other notations: the diagrams for each notation include their own variations, but use the (a) notation otherwise.

4.1 Notation 1

The experts preferred the (a) variation, and, when identifying correct diagrams, accuracy was significantly better with the (a) variation than the (b) variation.

When identifying inheritance errors in diagrams, however, the (b) variation produced both better accuracy, and a shorter response time. This is a surprising result, as we would expect the interrelation between accuracy and time to produce conflicting results (subjects are less likely to be accurate if they are working fast).

The (b) notational variant had been included to investigate the direction of flow in a class diagram: variation (a) follows the convention of reading from top to bottom, and places the superclass above the subclasses. The results appear to indicate that the upward flow of the arrows is more intuitive to understand, and, when this direction of flow is violated (as in variation (b)), the

[1] This result concurs with a result obtained as part of a previous study on user preferences of UML notation and layout (Purchase, Allder and Carrington 2000), when 76% of 70 subjects preferred notation (a) over notation (b): a two-tailed t-test significance of $p<0.05$.

subjects may be more careful and cautious when seeking errors.

As this notation variation did not actually change the syntax of the diagram, it is unlikely that there would have been a semantic misunderstanding that may have produced invalid rejections.

4.2 Notation 2

For identifying inheritance errors, (b) was more accurate than (a). Four out of five experts preferred variation (a).

The (b) notational variant had been included as it is a common layout method produced by automatic graph layout algorithms. The results suggest that when this (b) variation is used, subjects are more aware of inheritance errors; as for notation 1, the presence of a less intuitive notation may mean that the subjects are more diligent in identifying errors. The comment from the one expert who preferred the (b) notation supports this interpretation: "the diagonals require me to concentrate more when deciphering meaning."

As this notation variation changed the syntax only slightly, it is unlikely that there would have been a semantic misunderstanding that may have produced invalid rejections.

4.3 Notation 3

Variant (b) was more accurate for identifying errors in the associations than variant (a). All the experts preferred the (a) notation.

The syntax was changed very noticeably in this notation, so there is a possibility that there may have been some invalid rejections. These would most probably be on the more complex notational variant (b), and would help explain this high accuracy on the incorrect diagrams. However, the (b) variation highlights the association name in a separate rectangle on the diagram, rather than having the name of the association alongside the line, where it may be less obvious. This emphasising of the association name could also be the reason why the association errors were easier to identify.

As one of the experts pointed out, however, it would be rare to use association classes for such simple associations, which do not have additional attributes.

4.4 Notation 4

There were no significant results for this notation, although the time data for identifying an association error in the diagram approaches significance in favour of variation (a) being superior. All experts preferred the (a) notation.

This variation only affects association lines that are not represented horizontally. Therefore, only parts of the example diagram were affected in the different versions of the notation. This may have made the task for this notation of similar difficulty in all variants, as the parts of the diagram affected were easier to identify.

4.5 Notation 5

For identifying the correct diagrams, variation (a) was significantly more accurate than variation (b), but the response time for variation (a) was significantly greater than for variation (b). All experts preferred (a), where the cardinalities were made explicit, and there was no potential for ambiguity.

There is an obvious interaction between errors and time in the recognition of a correct diagram: faster response time meant more errors, and it is not surprising that the results are therefore conflicting.

Although the (b) notation performed better than the (a) notation with respect to the accuracy of identifying incorrect diagrams, this data was not statistically significant ($p=0.083$). It is very likely that the "invalid rejection" phenomenon affected these results. Although the semantic equivalence of the (a) and (b) variants was explained in the preparatory materials given to the subjects (and had been found in at least one UML text (Page-Jones 2000)), it appeared that the subjects found the (b) notation ambiguous. The (b) notation is less explicit than the corresponding (a) variant, and, as one of the experts noted, there was a potential semantic ambiguity in this variation, as, to some, * is semantically equivalent to 0..*, rather than 1..*.

5 Conclusions

The experts concurred with our intuitions that the (a) notations would produce better performance than the (b) notations. The quantitative data only support this with respect to the task of determining whether a diagram correctly represents a given specification. When identifying errors, the (b) notations tended to produce better performance: it appears that, when the notation is not intuitive, confusing, or ambiguous, subjects are more likely to identify errors. This may be because they feel less at ease with the unintuitive notation, and are therefore more diligent in seeking mistakes.

Of course, this experiment cannot be the final word on UML notational variations, and it is not intended to be so. Our aim has been to raise the issue of the lack of consistency between texts, propose a methodology for attempting to determine the most appropriate variation from the perspective of human comprehension, and to make some preliminary suggestions. The experiment has been run in a formal empirical manner, with constraints on the subjects, application domain and tasks. Case study investigations including, for example, the use of UML in a real world industrial application, or the learning of UML by students over an entire semester, would give greater insight into the suitability of the different notational variations from a human comprehension point of view. Such experiments may, in particular, relate the effectiveness of the variations according to the task for which UML is being used. It may be the case, for example, that some variations may be preferable for learning UML, but may need to be adapted for real world use on a multi-person project. In addition, the conventions and standards of different organisations may require use of one notation over another.

In depicting UML diagrams in CASE tools and UML texts, choices need to be made regarding which notation to use between semantically equivalent variations. Choosing the variation that most assists the users' comprehension can only enhance the value of the tool or text: only empirical studies can determine which of the variations are more suitable.

6 Acknowledgements

We are grateful to the students of the School of Information Technology and Electrical Engineering at the University of Queensland who willingly took part in the experiment, to members of the Distributed Systems Technology Centre at the University of Queensland for advice, and for acting as expert subjects, and to the Australian Research Council, which funded this research. Ethical clearance for this study was granted by The University of Queensland, 2001.

7 References

BLACKWELL, A. and GREEN, T. (2000): A cognitive dimensions questionnaire optimised for users. *Proceedings of the Twelfth Annual Meeting of the Psychology of Programming Interest Group*, Corigliano Calabro, Cosenza, Italy, 137-152, Memoria.

COX, K. (2000): Cognitive dimensions of use cases – Feedback from a student questionnaire. *Proceedings of the Twelfth Annual Meeting of the Psychology of Programming Interest Group*, Corigliano Calabro, Cosenza, Italy, 99-121, Memoria.

EVITTS, P. (2000): *A UML Pattern Language*. Indianapolis, Macmillan Technical Publishing.

FOWLER, M. and SCOTT, K. (2000): *UML Distilled*. Reading, Mass, Addison Wesley Longman Inc.

GREEN, T. and PETRE, M. (1996): Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages and Computing* **7**:131-174.

GURR, C. and STEVENS P. (1999): A cognitively informed approach to describing product lines in UML. unpublished. University of Edinburgh, Edinburgh, Scotland.

GURR, C. and TOURLAS, K. (2000): To the principled design of software engineering diagrams. *22nd International Conference on Software Engineering*, Limerick, Ireland, 509-518, ACM Press.

KUTAR, M., BRITTON, C. and WILSON, J. (2000): Cognitive dimensions: An experience report. *Proceedings of the Twelfth Annual Meeting of the Psychology of Programming Interest Group*, Corigliano Calabro, Cosenza, Italy, 81-98, Memoria.

PAGE-JONES, M. (2000): *Fundamentals of Object-Oriented Design in UML*. New York, Dorset House Pub.

PURCHASE, H., ALLDER, J. and CARRINGTON, D. (2000): User preference of graph layout aesthetics: A UML study. *Proceedings of the Graph Drawing Symposium 2000*, Colonial Williamsburg, USA, 5-18, Springer-Verlag.

PURCHASE, H., CARRINGTON, D. and ALLDER, J. (2000): Experimenting with aesthetic-based graph layout. *Proceedings of the Theory and Application of Diagrams Conference*, Edinburgh, Scotland 498-501, Springer-Verlag.

RATIONAL ROSE (2001) http://www.rational.com/products/rose/index.jsp, 23 October 2001.

RUMBAUGH, J. JACOBSON, I. and BOOCH, G. (1999): *The Unified Modeling Language Reference Manual*. Reading, Mass, Addison Wesley Longman Inc.

SCHMULLER, J. (1999): *Teach Yourself UML in 24 Hours*. Indianapolis, SAMS.