Managing Complexity in Object Oriented Analysis: Adapting UML for Modelling Large Scale Information Systems

Daniel L. Moody [1]
Guttorm Sindre [2]

[1] Department of Cybernetics
Czech Technical University
e-mail: dmoody@labe.felk.cvut.cz

[1] School of Business Systems,
Monash University
Melbourne, Australia
e-mail: dmoody@infotech.monash.edu.au

[2] Department of Computer and Information Science
Norwegian University of Science and Technology
e-mail: guttors@idi.ntnu.no

Abstract

Currently, the Unified Modelling Language (UML) lacks explicit abstraction mechanisms for managing complexity. Such constructs are essential for modelling problems of real world size and complexity, and in applying UML in large-scale commercial applications. Large class diagrams present major problems in understanding, particularly in the analysis stage, where diagrams need to be understood by a wide variety of stakeholders, many with little or no technical expertise. This paper proposes a refinement to UML which allows class diagrams to be represented at multiple levels of abstraction. The method is based on a solution to managing complexity of ER models, which has been successfully used in practice over a number of years. The method is soundly based on principles for organising knowledge drawn from systems theory, psychology and information science. Rather than formulating the refinement as an extension to UML (which we argue would be undesirable), we implement the approach by adapting existing language constructs.

Keywords

Complexity, package, information overload, decomposition, UML, class diagrams, ontology

INTRODUCTION

The Problem of Complexity in Object Oriented Analysis

Object modelling techniques in general, and UML in particular, currently lack effective mechanisms for dealing with the size and complexity of real world information systems (Berner et al, 1998; Engels and Groenewegen, 2000; Kobryn, 2000; Glinz et al, 2001; Opdahl and Henderson-Sellers, 2002). When UML is used to model large-scale commercial applications, the result is typically class diagrams of a hundred or more classes. In the absence of suitable mechanisms for complexity management, such models are represented as single, interconnected diagrams. Diagrams of this kind are difficult for stakeholders to understand and inhibit performance of analysts responsible for developing and maintaining these models. Experimental studies show that complexity has debilitating effects on end users' ability to comprehend and verify accuracy of IS models (Nordbotten and Crosby, 1999; Moody, 2001; Moody, 2002b; Shanks et al, 2002). Other studies have shown that complexity adversely affects analysts' ability to understand and maintain UML class diagrams (Genero et al, 2001).

Understandabiloity of Information Systems Models

Understandability is generally agreed to be an important determinant of the quality of an IS model (Roman, 1985; Mayer, 1989; von Halle, 1991; Batini et al, 1992; Levitin and Redman, 1994; Lindland et al, 1994; Kesh, 1995; Krogstie et al, 1995; Moody and Shanks, 1998; Schuette and Rotthowe, 1998; Witt and Simsion, 2000). IS models must be readily comprehensible so that they can be understood by all stakeholders, particular end users (Nordbotten and Crosby, 1999). If end users cannot effectively understand the model, they will also be unable to accurately verify whether it meet their requirements (Lindland et al, 1994). If the requirements as

expressed in the model are inaccurate or incomplete, the information system that is delivered will not satisfy users, no matter how well designed or implemented it is (Moody and Shanks, 1998). Empirical studies show that more than half the errors which occur during systems development are the result of inaccurate or incomplete requirements (Martin, 1989; Lauesen and Vinter, 2000). Requirements errors are also the most common reason for failure of systems development projects (Standish Group, 1995; 1996). This suggests that current methods for IS modelling are deficient for communication with users and that there is an urgent need to improve understandability of modelling notations.

Theoretical :Problems with UML: Ontological Incompleteness

Wand and Weber (Wand and Weber, 1990b; 1995; Weber, 1997) have proposed a theory of representation (often referred to as the Bunge-Wand-Weber or BWW ontology) which defines a comprehensive set of ontological concepts needed to represent the real world. This provides a theoretical basis for evaluating and comparing different modelling notations (Green and Rosemann, 2000). In applying this framework to evaluate UML, Opdahl and Henderson-Sellers (Opdahl and Henderson-Sellers, 2002) identified that it lacks constructs for representing *subsystems*, *decomposition* and *level structures* – these are precisely the constructs needed to manage complexity. This means that the ER Model is *ontologically incomplete*. Wand and Weber argue that these constructs are critical to the design and implementation of information systems and our ability to understand real world phenomena.

Research Objectives

Now that UML is becoming adopted in mainstream IS development, the issue of complexity is a practical problem that needs to be urgently addressed. Whereas in the past, UML has been applied mainly to small, specialist applications, it is now being increasingly applied in large scale commercial applications. Surprisingly, the issue of complexity management has received little attention in the software engineering and IS design literature. This may reflect a gap between research and practice: because it only arises when applying methods to problems of real world size and complexity, it may be an issue that academics are not aware of. This paper defines a method for representing large UML class diagrams in a way that maximises understanding and simplifies development and maintenance. This research incorporates both theoretical and practical objectives:

- To solve the practical problems in applying UML to model large scale information systems.
- To address the theoretical issue of ontological incompleteness in UML.

These objectives are complementary: by introducing constructs to address the problem of ontological completeness, the practical problems of complexity are resolved, as the missing constructs are precisely the ones needed to manage complexity.

THEORETICAL FOUNDATIONS

The Problem of Decomposition

The problem of representing large UML class diagrams can be considered as an instance of the *systems decomposition* problem (Weber, 1997). Decomposition is the process of breaking complex systems down into a set of smaller subsystems, and is one of the most common ways of dealing with complexity in large and complex systems (Flood and Carson, 1993). The major reason for decomposing large systems is to improve human understanding (Davis and Olson, 1985). A system which is too complex to be understood as a whole by the human mind can be broken down into a set of cognitively manageable units (Simon, 1996; Klir and Elias, 2003). Decomposition into subsystems also has advantages for development and maintenance, since subsystems can be added on, removed or modified relatively independently of each other (Wand and Weber, 1990a).

The ability to simplify a system using decomposition is based on the ability to divide the system into parts so that the relationships between the parts of the system can be shown at the outline level only (Flood and Carson, 1993; Klir and Elias, 2003). Detailed information about the relationships between elements belonging to different parts is likely to be lost. *Decomposable systems* are systems where the interactions between the parts are negligible and can be ignored for most purposes(Simon, 1996). *Nearly decomposable systems* are systems where the interactions between the parts are weak but not negligible(Simon, 1996). Such systems can be represented as hierarchies with the loss of comparatively little information. In studying the interaction between two molecules for example, we do not need to consider the individual interactions of nuclei of the atoms belonging to one molecule with the nuclei of the atoms belonging to the other.

In UML class diagrams this is not the case. A class diagram is a *non-decomposable system,* because no matter how it is divided into parts, relationships between classes cannot be ignored – these are critical for design

purposes. Network structured models such as class diagrams represent particular problems for decomposition, as there is no obvious way to divide them into subsystems without losing the connections between primitive level elements (in this case, classes). We cannot use a simple hierarchical representation to represent the model such as in Data Flow Diagrams(De Marco, 1978; Gane and Sarson, 1979; Page-Jones, 1988). The problem of non-decomposability explains why many network structured diagrams used in IS design lack effective mechanisms for complexity management. For example, Wand and Weber (1993), point out similar weaknesses in the Entity Relationship (ER) model despite the fact that this has been used in practice for more than 25 years.

A Street Directory as a Referent Problem

The problem of representing large and complex models is one which is faced in many other disciplines. Therefore a natural starting point in solving this problem is to look at how similar problems have been solved in other domains. This is an example of *analogical reasoning:* a problem solving approach in which a solution is found by adapting a solution from another (referent) domain (Gentner, 1983; Holyoak, 1985; Keane, 1985; 1988).

A street directory is an example of a successful solution to the problem of representing a large and complex network-structured model from another discipline. It provides a simple yet effective way of packaging a large amount of information in a way that people can easily understand. It has evolved over a long period of time, and has been proven to be highly effective in practice. Most people can use a street directory without explanation, and people are generally able to find the information they need, and to find it fairly quickly. The general form of a street directory is shown inFigure 1. The major structural components are:

- A *Key Map*, which provides an overview of the region covered by the directory.
- A set of numbered *Detail Maps*, each showing part of the region in full detail. These include *inter-map references*, which show how the maps fit together. There is partial overlap between detail maps to assist in navigation.
- A set of *Indexes*, listing roads, suburbs and other places of interest, together with their map reference.

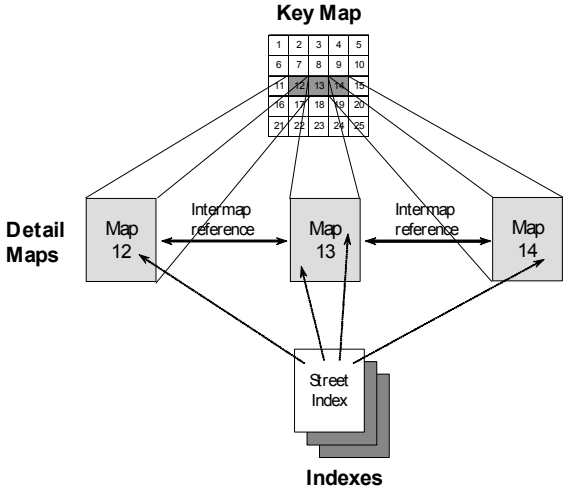

Figure 1. Street Directory Structure

In larger cities, multiple levels of key maps are often used, thus providing multiple levels of abstraction. The underlying model represented by the street directory is, at least conceptually, a single integrated map. However the street directory organisation defines a number of hierarchically linked, partially overlapping views, which make it easy to understand and reproduce in paper form. A street directory is a non-decomposable system, which suggests that the solution is likely to be transferable to the problem of decomposing network structured diagrams. In a street directory, it is critical to preserve the primitive level relationships between subsystems (streets which cross detail map boundaries) for it to be useful.

Levelled DataModels

Closely based on the street directory organisation, a method was developed for representing large ER models. The method was called Levelled Data Modelling, as it allows a large and complex data model to be represented at multiple levels of abstraction (Moody, 1997). This method has been successful applied in a wide range of industrial contexts, and has been extensively validated in laboratory and field studies (Moody, 1991; 1997;

1999; Moody, 2001; Moody, 2002c; 2002b; Moody, 2002a). Based on this solution, a set of general representational principles were proposed for incorporating complexity management mechanisms into IS modelling notations. Research is planned to apply these principles to a range of modelling notations, but UML class diagrams are a logical first step as they so closely resemble ER diagrams in structure.

PROPOSED REPRESENTATIONAL NOTATION

Incorporating Complexity Management in UML: Extension vs Adaptation?

There are two options for incorporating complexity management in UML:

- Adapt existing UML constructs
- Extend the UML to incorporate new constructs

There are obvious practical advantages of solving the problem within the current definition of UML. Firstly, the modelling language is already more than big enough to cause concern about stakeholders' ability to learn it (Krogstie, 2001; Siau and Cau, 2001). Secondly, significant changes are unlikely to be accepted by some of the language's stakeholders (e.g. tool vendors) (Dori, 2002). Hence, a suggested solution will have much better chance of being accepted in practice if it can utilise existing language constructs than if it introduces new special-purpose constructs as suggested by (Opdahl and Henderson-Sellers, 2002) or proposing a whole new modelling language as suggested by(Berner et al, 1998; Glinz et al, 2001).

In this paper, we propose to use the UML package construct as a complexity management mechanism. In the current definition of UML, this construct lacks clearly defined semantics and guidelines for how it should be used – this represents *construct ambiguity* (Weber, 1997). It has been also used to address a number of shortcomings in UML, to the point that it has become a *de facto* way of customising the language, which to some extent undermines the utility of UML as a universal standard. However the hierarchical structure of this construct (a package can consist of classes and/or other packages) makes it ideally suited for our purposes.

Overview of the Representation Method

In this section, we define a representation method for representing a large class diagram at multiple levels of abstraction, as a set of hierarchically linked diagrams. In this approach, large class diagrams are represented by the following components:

- A Context View at the top level (à la the key map), showing how the model is divided into subsystems or *Object Clusters*. Object Clusters are defined using the UML package construct.
- A set of bottom level Object Cluster Views (à la the detail maps), defining primitive level objects and relationships between them.
- Zero or more levels of Intermediate Level Views. The package construct is used recursively to create multiple levels of abstraction (i.e. a package can contain lower level packages). This means that a context view can be a view of lower level views, or of an object cluster. Any number of intermediate levels may be used, depending on the size of the underlying model.
- A set of indexes, telling in what part of the model any class (or other interesting construct, e.g., attribute, association, method) can be found.

Note that we talk about "views", not models. This is because we reserve the word model for the underlying model, which is still a single, interconnected network-structured model. What is presented here is thus not really a hierarchical structure of models, but of views filtered from a model of the problem domain. The representational approach does not change the underlying semantics of the model in any way, it just "packages" it in a way that is more easily understood. Importantly, the representation preserves the relationships between the primitive level elements of the model, which is a requirement of decomposing non-decomposable systems(Simon, 1996).

To illustrate the representational approach, we use an example. Figure 2 shows a UML class diagram of around twenty classes, which is quite small compared to the 100+ class models one would find in large industrial projects, but already beyond the comfortable limits of human information processing. Even with a diagram of this size, this would be quite difficult to understand if we had also included attributes for the classes and cardinalities and role names for the associations (which were omitted here to avoid eye strain for the reader).

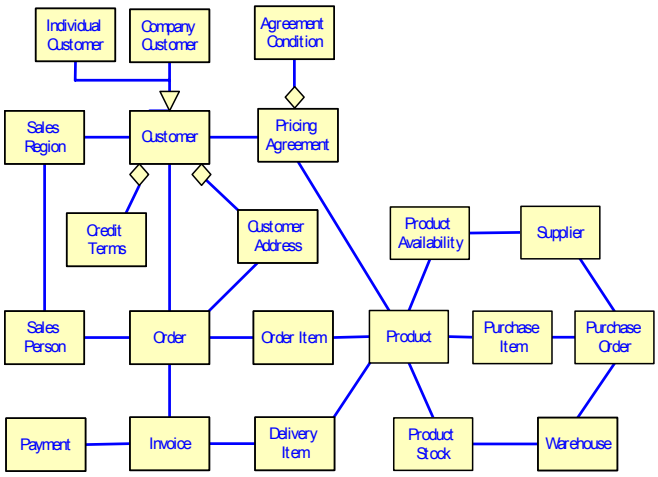

Figure2. Example Class Diagram

Context View

The first step in the method is to group the underlying classes into subsystems or Object Clusters. In simple terms, the rules for doing this are:

- Each class must be assigned to *one and only one* Object Cluster. The set of Object Clusters thus forms a *total partition* of the classes in the underlying model. The purpose of this is to avoid information loss and to minimise redundancy.
- Each Object Cluster should be cognitively manageable in size, following the "seven, plus or minus two" principle.
- Each Object Cluster should be named after one of the elements of the cluster, called the *central class*. This should be the class of central importance in the cluster.
- The number and strength of associations (coupling) between Object Clusters should be minimised in order to reduce dependencies between them.

A comprehensive set of decomposition principles together with associated metrics are defined in(Moody and Flitman, 1999) but are omitted here for brevity. Application of the above rules results in a decomposition of the example class diagram into three Object Clusters (Figure 3).

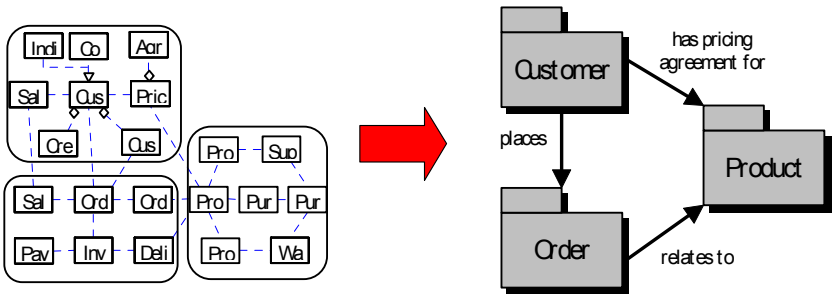

Figure 3. Class Diagram Clustered into Object Clusters and Context View

The right side of Figure 3 shows the Context View. This provides a high level overview of the model and how it is divided into subsystems, and is used to guide the reader in search for more detail. In the same way that a class diagram can be expanded to show internal components of classes (methods), so the Context View can be expanded to show components of Object Clusters (which will be either classes or lower level Object Clusters) (see Figure 4).

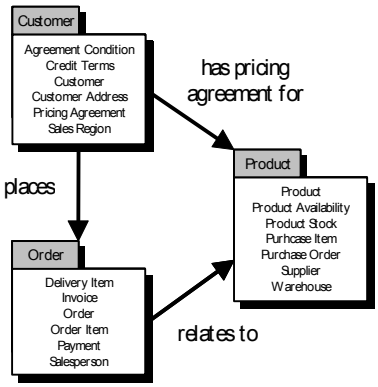

Figure 4. Context View (Glass Box View)

However we argue that it would be preferable for object clusters to be represented using a symbol more easily distinguishable from the class symbol – for example a circle or ellipse. It is important to clearly differentiate between "real" elements of the model (classes or ontological "things") and aggregations introduced for the purpose of complexity management. Both field studies (Moody, 2002c) and experimental studies (Nordbotten and Crosby, 1999) have shown that use of similar symbols to represent different semantic constructs leads to confusion in interpretation – diamonds are even confused with rectangles. However to do this would require a change to the standard representation of the package construct, which would reduce the chances of adoption in practice, at least in the short term. As a compromise, we have used the standard package symbol, but with shading to provide clear visual differentiation between object clusters and classes.

Object Cluster View

More detail about each Object Cluster in the Context View can be found in the Object Cluster Views. Each symbol on the top level diagram "explodes" to a lower level diagram (either a lower level Context View or an Object Cluster View). An example of an Object Cluster View is shown inFigure 6, showing what would be shown to the user when opening up the Customer package. Even if we had included attributes for the classes and cardinalities and role names for the associations, this diagram would still be relatively easy to read. The central class (Customer) is shown towards the centre of the diagram and is shown larger than all of the other classes in the cluster. The central class concept provides the mechanism for moving between different levels of abstraction ("drill up" and "drill down").

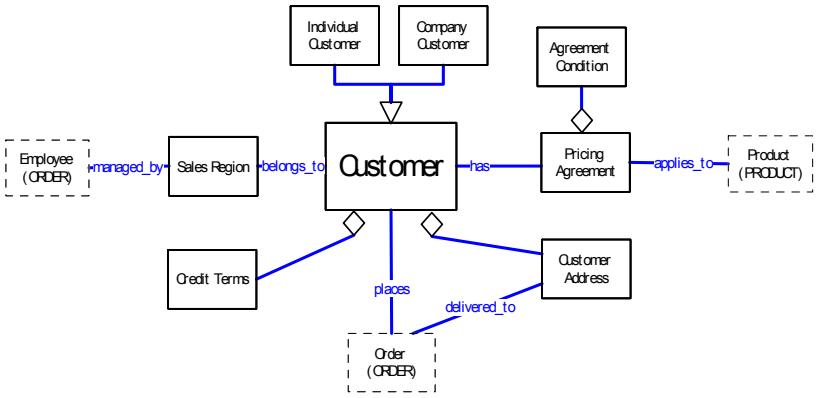

Figure 6. Customer Object Cluster View

In addition to showing the classes that were clustered into the Customer package in Figure 3, the Customer Object Cluster View also shows the object classes directly associated to any class within this package. These provide bi-directional navigational links between Object Cluster Views. The street directory analogy to duplication of classes in this way is that various map pages are normally presented with a slight overlap with the joining maps in each direction, to make it easier for the reader to see connections between maps. These extra classes in the view are called foreign classes, as they are analogous to the use of foreign keys in relational databases. An important difference is that while foreign keys only provide links in one direction (in the direction of referential integrity), foreign classes are shown in both directions. This is closer to the street directory analogy (if there is a reference from Map 4 to Map 5, there will be a similar reference in the reverse

direction) and is consistent with ontological principles for representing relationships(Weber, 1997; Shanks et al, 2002). Foreign classes are shown using dotted boxes, with the Object Cluster they belong to shown in parentheses.

A foreign class does *not* require a new modelling construct, as it is simply a diagrammatical convention to assist navigation between Object Cluster Views. An important property of the foreign class convention is that preserves relationships between primitive level classes, which preserves the non-decomposability property of UML class diagrams. Empirical studies have shown that this is the "optimal" level of redundancy between bottom level views – any more or less will reduce stakeholder understanding(Moody, 2001; Moody, 2002c).

Intermediate Level Views

Because we have used such a simple example, there is no need for intermediate level diagrams – these are only required when the number of Object Clusters exceeds the "seven plus or minus two" limit. Intermediate level diagrams are represented using the same conventions as the Context View. However there is no overlap between diagrams at intermediate levels – that is, there is no concept of "foreign" Object Clusters. The reason for this is that all horizontal relationships are defined in the bottom level diagrams (Object Cluster Views).

In general, models can be represented at any number of levels depending on the size of the underlying model. This results in a hierarchy of views, with higher levels representing higher levels of abstraction (Figure 7). At each level, elements of views are aggregations of elements at the next level down. This is called a *multi-level structure system* (Klir and Elias, 2003) or *level structure* (Weber, 1997). At the bottom level (Object Cluster Views), the elements are primitive level classes. The higher level views (Context and Intermediate Level Views) define hierarchically structured, non-overlapping views of the model, while the Object Cluster Views represent partially overlapping views of the model.

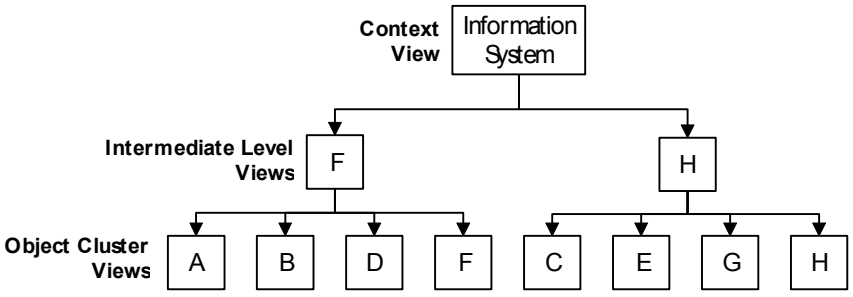

Figure 7. Level Structure

JUSTIFICATION OF THE METHOD

Theoretical Justification

The representation approach incorporates a number of principles for organising knowledge drawn from a range of disciplines, including psychology, systems theory and information science.

"Chunking"

A wide range of experimental studies have shown that humans organise items into logical groups or "chunks" in order to conserve memory (e.g. Miller, 1956; Cofer, 1965; Newell and Simon, 1972; Murdock, 1993; Baddeley, 1994). The ability to recursively develop information saturated chunks is the key to people's ability to deal with complexity on a day to day basis (Flood and Carson, 1993). As Miller (1956) says:

> *"Human beings have severe limitations on the amount of information they are able to receive, process, and remember. Their "span of immediate memory" or "channel capacity" is limited to approximately seven, plus or minus two, items. Through recoding items so that information to be remembered is placed in larger "chunks", it is possible to expand this channel capacity somewhat."*

UML class diagrams already provide one level of "chunking" through the use of classes. The mechanism of encapsulation is used to group atomic level attributes and processes (methods) into classes(Smith and Smith, 1977; Hull and King, 1987; Peckham and Maryanski, 1987). The chunking mechanisms provided by UML, while sufficient for small problems, are not enough to cope with models of real world size and complexity. Therefore there is a need for a mechanism for recursively grouping classes into higher level "chunks". The method described in this paper makes large UML class diagrams understandable to humans by organising them

into "chunks" of cognitively manageable size (seven plus or minus two classes) – this solves the problem of information overload.

Hierarchical Organisation

If a system is decomposed into subsystems, and these subsystems are decomposed into smaller subsystems, and so on, the resulting system most often forms a hierarchy or *level structure(Wand and Weber, 1990a). Through successive applications of the decomposition principle, the proposed method results in a hierarchy of models at different levels of abstraction: the Context View is at the highest level, Object Cluster Views are at the lowest level, and there may be any number of intermediate levels in between. Hierarchy is one of the most effective ways of organising complexity for the purposes of human comprehension (Flood and Carson, 1993; Klir and Elias, 2003). Simon (1996) points out the use of hierarchy in complex systems in organisations, societies, biological systems, physics and symbolic systems as a basis for suggesting it as a general architecture for complexity. Hierarchical organisation also supports top down understanding of the model, which has been shown experimentally to improve understanding of models (Nordbotten and Crosby, 1999).*

Redundancy

Perhaps the most difficult practical issue in hierarchically decomposing a UML class diagram is the need to preserve relationships between primitive level classes – this is a consequence of the non-decomposability of class diagrams. According to systems theory, the use of redundant elements (called *linking variables*) are necessary to preserve relationships between different parts of a system when it is decomposed(Klir and Elias, 2003). The street directory uses overlap at the boundary of maps and inter-map references to show relationships between detail maps. The method described in this paper uses redundancy in a similar way, by duplicating associations and the classes involved between related Object Clusters. Foreign classes are redundant elements introduced to preserve the relationships between classes. However the redundancy is strictly controlled – only associations which cross subsystem (Object Cluster) boundaries are duplicated between views.

Indexing

Indexing is one of the most common mechanisms used in information consolidation(Taylor, 1985). Indexes guide the search process by narrowing the information universe to a subset that has some probability of containing material that is needed (Saracevic, 1985).

Empirical Justification

The method described in this paper has not yet been tested in practice, although research is currently in progress in this area. However the fact that it is based on a successful solution to the problem in the ER domain gives some level of confidence that it should work. The Levelled Data Model method was applied in over twenty organisations over a seven year period as part of an ongoing action research programme(Moody, 2001). In addition, experimental analysis showed that the method improved end user comprehension and verification performance compared to the standard ER model by more than 50%(Moody, 2002b). This provides strong evidence for the representational principles in improving end user understanding of models. Given the similarities between class diagrams and ER diagrams,we argue by analogy that similar benefits should be found when a similar representation approach is applied to UML. However such claims represent *plausible presumptions* rather than *validated truths* (Rescher, 1977), and can only be proven by empirical testing.

CONCLUSION

Summary

Effective complexity management mechanisms are an essential requirement for applying any modelling technique to real world problems (Wand and Weber, 1993; Weber, 1997). UML currently lacks such a mechanism (Berner et al, 1998; Engels and Groenewegen, 2000; Kobryn, 2000; Glinz et al, 2001; Opdahl and Henderson-Sellers, 2002). In this paper, we have proposed a method for representing UML class diagrams as a set of hierarchically linked views, which addresses the deficiencies of UML for modelling large scale information systems. The approach uses only standard UML language constructs (though with some modifications to their diagrammatical representation and guidelines for use), which we argue increases the likelihood of acceptance in practice. Our approach represents a pragmatic compromise between theoretical considerations (e.g. ontological completeness and clarity) and practical considerations (e.g. ease of use, adoption in practice). We take the view that it is better to work within the existing standard rather than extending it or proposing a whole new language as others have suggested. The method described in this paper adapts a successful solution to a similar problem in the ER modelling field, which has been empirically proven in both field and laboratory studies. Surprisingly, there has been relatively little transfer of knowledge between OO and

ER research, despite the fact that there are clear similarities between the notations and many common problems faced in both fields.

Practical Significance

The practical contribution of this research is that it defines clear guidelines for practitioners for representing class diagrams of real world size and complexity in a way that maximises stakeholder understanding. This will help stakeholders to more effectively verify models and reduce the incidence of requirements errors in UML-based development. Another practical benefit of this work is that by defining clear guidelines for use of the package construct, it will help to prevent this construct being used to make unauthorised extensions to UML, which will strengthen its value as a universal standard.

Theoretical Significance

The theoretical contribution of this work is that it addresses the problem of ontological incompleteness (Wand and Weber, 1993; Weber, 1997) in UML by introducing constructs to represent the ontological concepts of subsystems, decomposition and level structures which are required for complexity management. In addition, by defining clear guidelines for the use of the package construct, it addresses the existing issues of construct ambiguity and construct overload with this construct. However for this to be effective, use of the package construct must be restricted to this purpose and no other.

REFERENCES

BADDELEY, A.D. (1994): "The Magical Number Seven: Still Magic After All These Years?", Psychological Review, 101, 2.

BATINI, C., CERI, S. and NAVATHE, S.B. (1992): Conceptual Database Design: An Entity Relationship Approach, Benjamin Cummings, Redwood City, California.

BERNER, S., JOOS, S., GLINZ, M. and ARNOLD, M. (1998): "A Visualization Concept for Hierarchical Object Models", Proceedings of the 13th IEEE International Conference on Automated Software Engineering (ASE-98),

COFER, C.N. (1965): "On Some Factors in the Organisational Characteristics of Free Recall", American Psychologist, pp. 261-272, April.

DAVIS, G.B. and OLSON, M.H. (1985): Management Information Systems: Conceptual Foundations, Structure, and Development, McGraw-Hill, New York.

DE MARCO, T. (1978): Structured Analysis And System Specification, Yourdon Press.

DORI, D. (2002): "Why Significant UML Change is Unlikely", Communications of the ACM, 45, 11, pp. 82-85.

ENGELS, G. and GROENEWEGEN, L. (2000): "Object-Oriented Modeling: A Roadmap". Future of Software Engineering, ACM Press, pp. 103-116.

FLOOD, R.L. and CARSON, E.R. (1993): Dealing With Complexity: An Introduction To The Theory And Application Of Systems Science, Plenum Press.

GANE, C. and SARSON, T. (1979): Structured Systems Analysis, Prentice-Hall.

GENERO, M., POELS, G. and PIATTINI, M. (2001): "Defining and Validating Measures for Conceptual Data Model Quality", 13th International Conference on Advanced Information Systems Engineering (CAiSE 2001), K.R. Dittrich, A. Geppert and M.C. Norrie, (Eds.), Interlaken, Switzerland, Springer, Lecture Notes in Computer Science 2068, June 4-8.

GENTNER, D. (1983): "Structure-Mapping: A Theoretical Framework For Analogy", Cognitive Science, 7.

GLINZ, M., BERNER, S., JOOS, S., RYSER, J., SCHETT, N. and XIA, Y. (2001): "The ADORA Approach to Object-Oriented Modeling of Software", Advanced Information Systems Engineering, Proceedings of CAiSE 2001, K.R. Dittrich, A. Geppert and M.C. Norrie, (Eds.), Interlaken, Switzerland, Lecture Notes in Computer Science Vol. 2068. Berlin: Springer,

GREEN, P. and ROSEMANN, M. (2000): "Integrated Process Modelling: An Ontological Evaluation", Information Systems Journal, 25, 2.

HOLYOAK, K.J. (1985): "The Pragmatics Of Analogical Transfer", The Psychology Of Learning And Motivation, 19.

HULL, R. and KING, R. (1987): "Semantic Data Models", ACM Computing Surveys, 19, 3, September.

KEANE, M. (1985): "On Drawing Analogies When Solving Problems: A Theory And Test Of Solution Generation In An Analogical Problem Solving Task", British Journal Of Psychology, 76.

KEANE, M. (1988): Analogical Problem Solving, Wiley, New York.

KESH, S. (1995): "Evaluating The Quality Of Entity Relationship Models", Information And Software Technology, 37, 12.

KLIR, G.J. and ELIAS, D. (2003): Architecture of Systems Problem Solving (2nd Edition), Plenum Publishing Corporation, New York.

KOBRYN, C. (2000): "Modeling Components and Frameworks with UML", Communications of the ACM, 43, 10, pp. 31-38.

KROGSTIE, J. (2001): "Using a Semiotic Framework to Evaluate UML for the Development of Models of High Quality". Unified Modeling Language: Systems Analysis, Design, and Development Issues, T. Halpin, (Ed.), IDEA Group Publishing, pp. 89-106.

KROGSTIE, J., LINDLAND, O.I. and SINDRE, G. (1995): "Towards a Deeper Understanding of Quality in Requirements Engineering", Proceedings of the 7th International Conference on Advanced Information Systems Engineering (CAISE), Jyvaskyla, Finland, June.

LAUESEN, S. and VINTER, O. (2000): "Preventing Requirement Defects", Proceedings of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2000), Stockholm, Sweden, June 5-6 2000.

LEVITIN, A. and REDMAN, T. (1994): "Quality Dimensions of a Conceptual View", Information Processing and Management, 31, 1.

LINDLAND, O.I., SINDRE, G. and SØLVBERG, A. (1994): "Understanding Quality In Conceptual Modelling", IEEE Software, 11, 2, pp. 42-49, March.

MARTIN, J. (1989): Information Engineering, Prentice Hall, Englewood Cliffs, N.J.

MAYER, R.E. (1989): "Models for Understanding", Review of Educational Research, Spring.

MILLER, G.A. (1956): "The Magical Number Seven, Plus Or Minus Two: Some Limits On Our Capacity For Processing Information", The Psychological Review, March.

MOODY, D.L. (1991): "A Practical Methodology For The Representation Of Large Data Models", Proceedings Of The Australian Database And Information Systems Conference, University Of N.S.W., Sydney, Australia, February.

MOODY, D.L. (1997): "A Multi-Level Architecture For Representing Enterprise Data Models", Proceedings Of The Sixteenth International Conference On Conceptual Modelling (ER'97), D.W. Embley and R.C. Goldstein, (Eds.), Los Angeles, November 1-3.

MOODY, D.L. (1999): "Bringing Enterprise Models To Life: Towards A More User-Oriented Representation of Data", Proceedings of the International Conference on Enterprise Modelling (IEMC'99), Verdal, Norway, June 14-16.

MOODY, D.L. (2001): Dealing with Complexity: A Practical Method for Representing Large Entity Relationship Models (PhD Thesis), Department Of Information Systems, University of Melbourne, Melbourne, Australia.

MOODY, D.L. (2002a): "Comparative Evaluation of Large Data Model Representation Methods: The Analyst's Perspective", 21st International Conference on Conceptual Modeling (ER'2002), S. Spaccapietra, S.T. March and Y. Kambayashi, (Eds.), Tampere, Finland, October 7-11.

MOODY, D.L. (2002b): "Complexity Effects On End User Understanding Of Data Models: An Experimental Comparison Of Large Data Model Representation Methods", Proceedings of the Tenth European Conference on Information Systems (ECIS'2002), Gdansk, Poland, June 6-8.

MOODY, D.L. (2002c): "Validation of a Method for Representing Large Entity Relationship Models: An Action Research Study", Proceedings of the Tenth European Conference on Information Systems (ECIS'2002), Gdansk, Poland, June 6-8.

MOODY, D.L. and FLITMAN, A. (1999): "A Methodology for Clustering Entity Relationship Models: A Human Information Processing Approach", Proceedings of the Eighteenth International Conference on

Conceptual Modelling, J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau and E. Metais, (Eds.), Paris, France, Springer, November 15-18.

MOODY, D.L. and SHANKS, G.G. (1998): "What Makes A Good Data Model? A Framework For Evaluating And Improving The Quality Of Entity Relationship Models", Australian Computer Journal, August.

MURDOCK, B.B. (1993): "TODAM3: A Model Of Storage And Retrieval Of Item, Associative And Serial Order Information", Psychological Review, 100, 2, pp. pp. 183-203.

NEWELL, A.A. and SIMON, H.A. (1972): Human Problem Solving, Prentice-Hall.

NORDBOTTEN, J.C. and CROSBY, M.E. (1999): "The Effect of Graphic Style on Data Model Interpretation", Information Systems Journal, 9.

OPDAHL, A.L. and HENDERSON-SELLERS, B. (2002): "Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model", Software and Systems Modelling, 1, 1, pp. 43-67.

PAGE-JONES, M. (1988): The Practical Guide to Structured Systems Design, Prentice Hall, Englewood Cliffs, N.J.

PECKHAM, J. and MARYANSKI, F. (1987): "Semantic Database Modelling: Survey, Application And Research Issues", ACM Computing Surveys, September.

RESCHER, N. (1977): Methodological Pragmatism: Systems-Theoretic Approach to the Theory of Knowledge, Basil Blackwell, Oxford.

ROMAN, G. (1985): "A Taxonomy of Current Issues in Requirements Engineering", IEEE Computer, April.

SARACEVIC, T. (1985): "Processes And Problems In Information Consolidation", UNESCO.

SCHUETTE, R. and ROTTHOWE, T. (1998): "The Guidelines Of Modelling: An Approach To Enhance The Quality In Information Models", Proceedings Of The Seventeenth International Conference On Conceptual Modelling (ER '98), Singapore, Elsevier Lecture Notes in Computer Science, November 16-19.

SHANKS, G.G., MOODY, D.L., NUREDINI, J., TOBIN, D. and WEBER, R.A. (2002): "Representing Things And Properties In Conceptual Modelling: An Empirical Evaluation", University of Melbourne Department of Information Systems Working Paper, December 1.

SIAU, K. and CAU, C. (2001): "Unified Modeling Language: A complexity analysis", Journal of Database Management, 12, 1, pp. 26-34.

SIMON, H.A. (1996): Sciences Of The Artificial (3rd edition), MIT Press.

SMITH, J.M. and SMITH, D.C.P. (1977): "Database Abstractions: Aggregation and Generalization", ACM Transactions on Database Systems, 2, 2.

STANDISH GROUP (1995): The CHAOS Report into Project Failure, The Standish Group International Inc. Available on-line at http://www.standishgroup.com/visitor/chaos.htm.

STANDISH GROUP (1996): Unfinished Voyages, The Standish Group International Inc. available on-line at http://www.standishgroup.com/visitor/voyages.htm.

TAYLOR, R.S. (1985): Value Added Processes In Information Systems, Ablex Publishing.

VON HALLE, B. (1991): "Data: Asset or Liability?", Database Programming and Design, 4, 7, July.

WAND, Y. and WEBER, R.A. (1990a): "A Model for Systems Decomposition", Proceedings of the Eleventh International Conference on Information Systems, J.I. DeGross, M. Alavi and H. Oppelland, (Eds.), Copenhagen, Denmark, December.

WAND, Y. and WEBER, R.A. (1990b): "An Ontological Model of an Information System", IEEE Transactions on Software Engineering, pp. 1282-1292, November.

WAND, Y. and WEBER, R.A. (1993): "On the Ontological Expressiveness of Information Systems Analysis and Design Grammars", Journal of Information Systems, October.

WAND, Y. and WEBER, R.A. (1995): "On the Deep Structure of Information Systems", Information Systems Journal, 5, pp. 203-223.

WEBER, R.A. (1997): Ontological Foundations Of Information Systems, Coopers And Lybrand Accounting Research Methodology Monograph No. 4, Coopers And Lybrand, Melbourne, Australia.

WITT, G.C. and SIMSION, G.C. (2000): Data Modeling Essentials: Analysis, Design, and Innovation, The Coriolis Group.

COPYRIGHT