EVOLUTION OF OO METHODS: THE UNIFIED CASE

Matti Rossi
University of Jyväskylä
Department of Computer Science and Information Systems
P.O. Box 35
SF-40351 Jyväskylä
Finland
Internet: mor@jyu.fi
Fax: 358-41-603011

ABSTRACT

This paper takes an evaluative look into OO methods and especially the evolution of the new Unified method from its ancestors, OMT and OODA. The paper tries to classify the components of the earlier methods and identify the parts that have been taken into the Unified method. The research applies the method metrics approach. For the sake of compactness we limit ourselves to the class diagram technique of all methods. We make observations about the number of concepts in each variation and show how the metrics can be used to analyse the changes in the techniques.

INTRODUCTION

Recent years have witnessed the appearance of new systems development paradigms and methods. Examples of these are object-oriented analysis and design methods (eg see Booch (1991), Rumbaugh, Blaha, Premerlani, Eddy and Lorensen (1991), Booch and Rumbaugh (1995) etc.). However, we feel that there is need for improvement in the analysis of these methods and in understanding their use and functionality. Although some attempts have been made to compare existing methods Champeaux (1991), Iivari (1994), Olle, Sol and Verrijn-Stuart (1982) and numerous others, the studies lack rigour and a sound conceptual foundation, and are mostly based on ad hoc feature analysis techniques. Some recent attempts have proposed more systematic approaches, based on a common formal metamodelling language to describe methods (e.g. see Hong, Goor and Brinkkemper (1992), Song and Osterweil (1992). In this paper we investigate a few "generations" of OO methods and identify their differences and similarities. The aim of the study is to highlight the changes in the methods and discuss the features of the evolved methods. Especially we investigate the changes in the definitions of the core method concepts by analysing the metamodels of the methods.

We are interested in the properties of the methods and the representation of different methods within a common formalism. Through the modelling of methods we hope to gain more understanding about their nature. In general several metamodel based comparison strategies can be used. Here we distinguish three ways, namely comparing, combining and statistical measuring. First one, the comparison strategy, is most obvious way to utilise metamodel in which we use two or several metamodels and compare them directly to each other (i.e. two by two or all together). This allows us to find commonalties and obscurities in methods concept structures and seek for different connectivity approaches between modelling languages. Second, the combination approach applies several metamodels in order to build a method that include some (best) parts from each method. Combination can be done either by building up a super method (collecting all of aspects or elements defined) as in Hong, Goor and Brinkkemper (1992), selecting a minimum set of common aspects, or through a hybrid approach using both of the previous ways. Third, the measurement approach in Rossi and Brinkkemper (1995) collects knowledge from all the metamodels and tries to find for example regular or divergent structures. In this paper we apply the last approach by making some observations with metrics on metamodels.

We try to measure the complexity of learning and applying the methods. We restrict the complexity of techniques and methods to two aspects. On the one hand we try to measure the complexity of learning and understanding the technique, which is related to the number of different concepts and constructs (say, object types and relationship types) used in the technique. On the other hand it is desirable to get insight in the complexity of the internal structure of the models resulting from applying the technique. This complexity is dependent on the number of describing properties of the technique's objects and relationships.

This paper is organised as follows. In the next section we present the metamodeling method used to describe the knowledge of methods. In section 3 we present the resulting metamodels of the three selected methods. These metamodels capture both the static concept structure part of the method and the dynamic part of the method use process. In section 4 we compare these methods by obtained results of metrics. Finally, section 5 summarizes research outcomes and presents future directions for this kind of evaluation.

THE METHOD DESCRIPTION LANGUAGE

Techniques of interest here consist of traditional graphical formalisms. These techniques describe the object systems by objects and their relationships. In many cases the relationships and objects can have attributes or properties. The techniques usually describe only one aspect of an object system (such as data flows or state changes etc.). To be able to compare and analyse techniques, we describe their structure using one common language to define the metamodels of the methods. We use here the OPRR (Object, Property, Relationship, Role) methodmodelling language, as presented by Smolander (1990) to model the techniques and methods. The use of one methodmodelling language gives us a neutral basis to compare the properties of techniques, and it provides a common background for the formulation of metrics.

It is important to notice, however, that there are a few factors that can bias the results. First, the precision of the method description is dependent on the quality of the technique's description in the textbook. Some authors present detailed formal descriptions of their techniques, and others define their techniques more vaguely. Secondly, the experience and personal preferences of the method modeller affect the model. For example, a given concept might be metamodelled either as a property or as a relationship. In this particular case all of the techniques have been modelled by one experienced method engineer, and we can thus expect that they have been modelled with a consistent style.

The acronym OPRR comes from the words Object, Property, Role, and Relationship which are the *meta-types* in OPRR. Welke (1988) defines the meta-types in the following way:

- *Object* is a "thing" which exists on its own. Examples of objects are process, flow, store, source, module, etc.

Properties are the describing or qualifying characteristics associated with the other meta-types. Typical properties include name, description, definition, etc.

- *Relationship* is an association between two or more objects. For example, there may be a relationship between a source and a process meaning that the process *uses* the source.
- *Role* is the name given to the link between an object and its connection with a relationship. From the example above, the process would be the *user* and the source would be the *origin* of the data.

In MetaEdit's CAME environment we have extended OPRR by defining explicit mappings from these concepts onto their representations. We have used the following notation: an object type is represented by a rectangle, a property type by an ellipse, a role type by a circle and a relationship type by a diamond. The name of each object, property, role or relationship type is written inside its symbol. An example of such a graphical OPRR model is in Figure 1.

TECHNIQUES COMPARED

All the methods use a number of individual techniques. As most of them apply quite similar state diagrams and the main techniques are the class diagrams, that describe the class structure of the system under development, we restrict ourselves to the CDs. In this chapter we describe the three techniques used for the comparison. For the sake of compactness we use only the first published version of OODA and OMT class diagrams and the Unified version 0.9. For each of the techniques we describe its objects and relationships and show its graphical OPRR model. The models can be found from Rossi and Tolvanen (1994), except for Unified CD, which was modelled for this project.

Technique: Class diagram

This is the (1991) version of quite popular Booch method for object-oriented systems development Booch (1991). It describes a system by its class structure with special emphasis to groupings of the classes.

Technique's concepts

Class, Class category, Class utility

Relationships of the technique

Class category visibility, Inherits (compatible type), Inherits (new type), Instantiates (compatible type), Instantiates (new type), MetaClass, Undefined, Uses (for implementation), Uses (for interface)

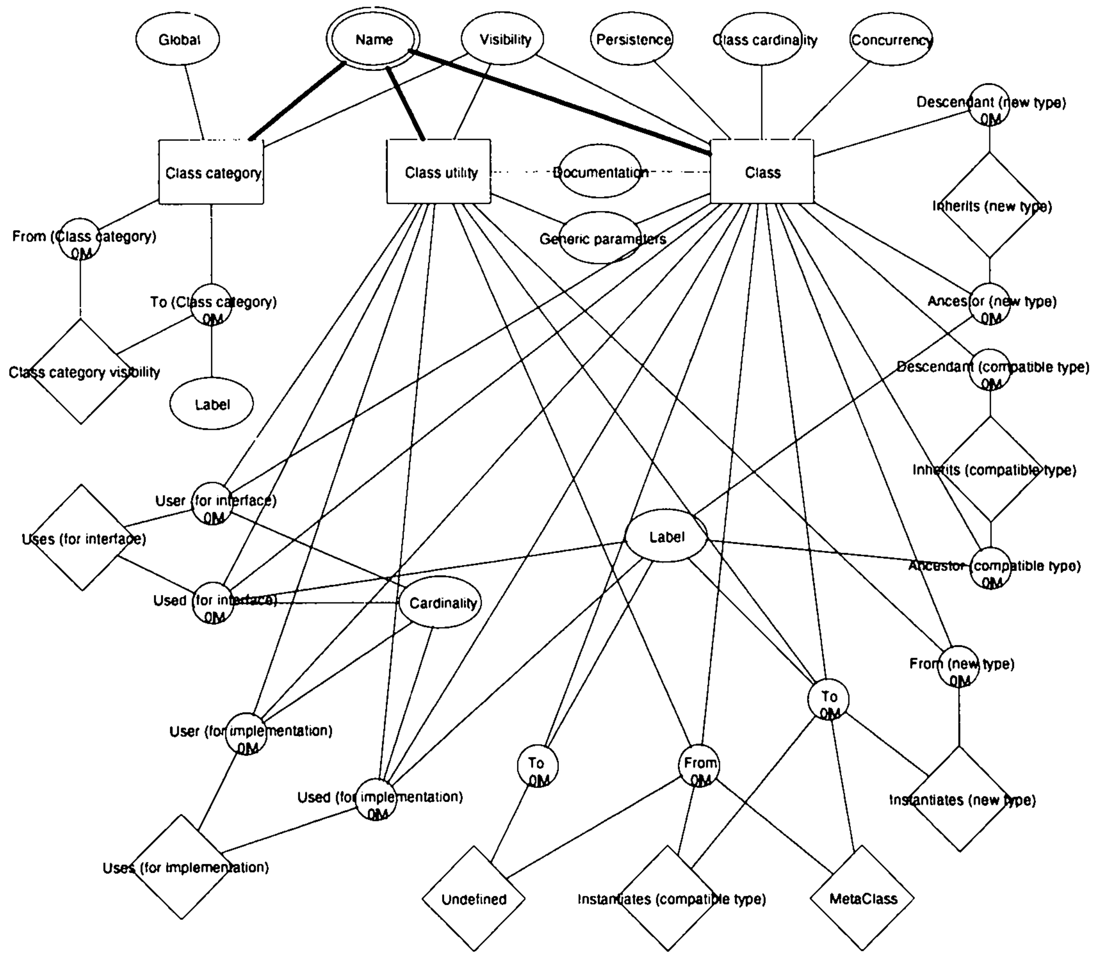

Figure 1. OPRR model of OODA CD

Technique: OMT Class Diagram

The OMT Class Diagram describes the static structure of the objects in a system and their relationships Rumbaugh, Blaha, Premerlani, Eddy and Lorensen (1991).

Technique's concepts

Class, Disjoint divider, Divider, Object, Subclass group

Relationships of the technique

Aggregation 1 to 1, Aggregation 1 to M, Association (optional to 1), Association (optional to many), Association (optional to optional), Association 1 to 1, Association 1 to many, Association many to many, Generalisation, GeneralisationSpecialisation, Instantiation, Qualified association 1 to 1, Qualified association 1 to many, Qualified association many to many, Specialisation

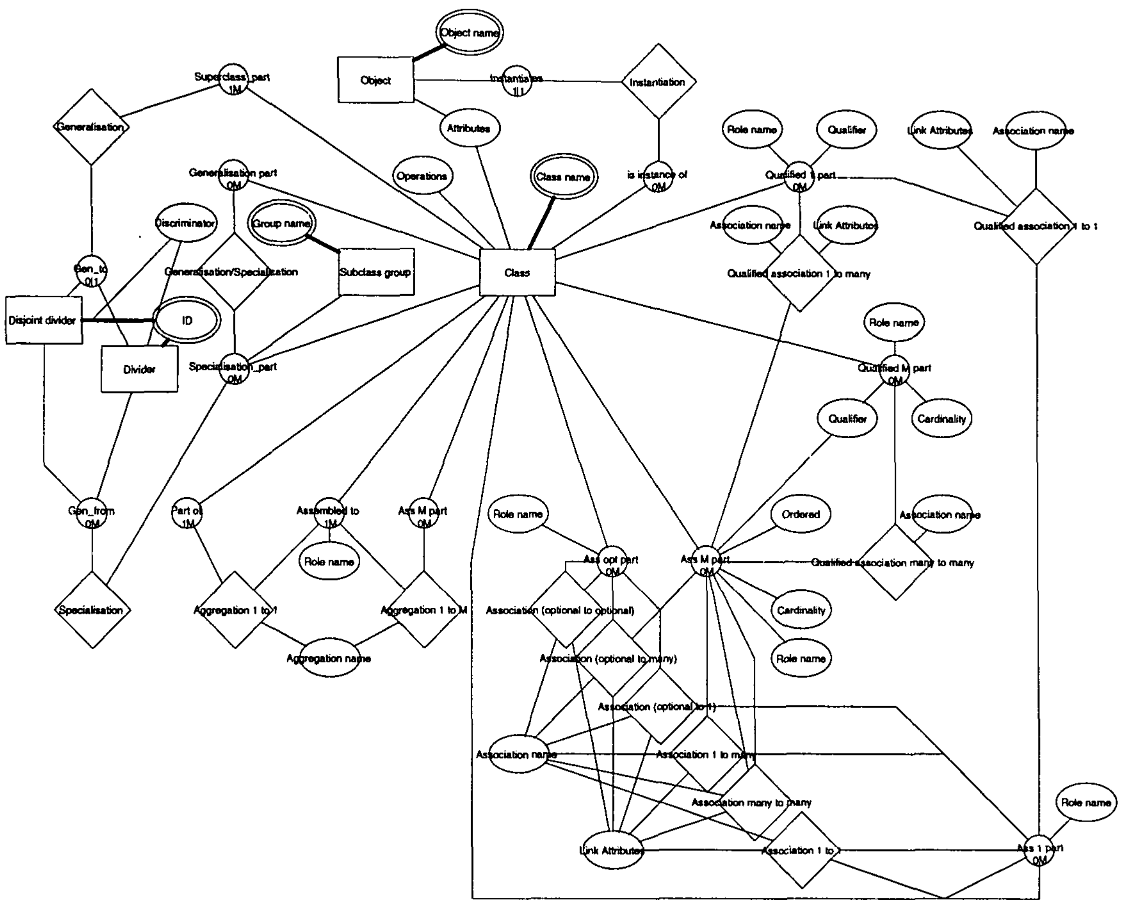

Figure 2. OPRR model of OMT CD

Technique: Unified Class Diagram

The Unified method can be seen as a direct descendant of the two above mentioned methods and especially the Class Diagram technique seems to be an amalgamation of their concepts. The Unified Modeling Language, or UML, is claimed to be a third-generation object-oriented modeling language. At present UML is being pushed to the Object Management Group in the hope that it will become a standard modeling language for OO development. The model here is based on the preliminary 0.8 version of the technique Booch and Rumbaugh (1995).

Technique's concepts

Class, Object

Relationships of the technique

Aggregation, Association, Inheritance, Instantiation, Qualified association

Remarks about the techniques

There are noticeable similarities between the concepts of the techniques. OMT and Unified distinguish the OO notions of Class and instance (Object), but OODA uses Classes and their categories. OODA uses a different technique for modelling the instances. The relationships in the methods differ noticeably. We can argue, that the OODA relationship types come from the ADA programming language, whereas the OMT CD relationships come from semantic modelling camp. The Unified CD's have mainly adopted, but simplified, the OMT relationships. The properties of the concepts are mostly identical.

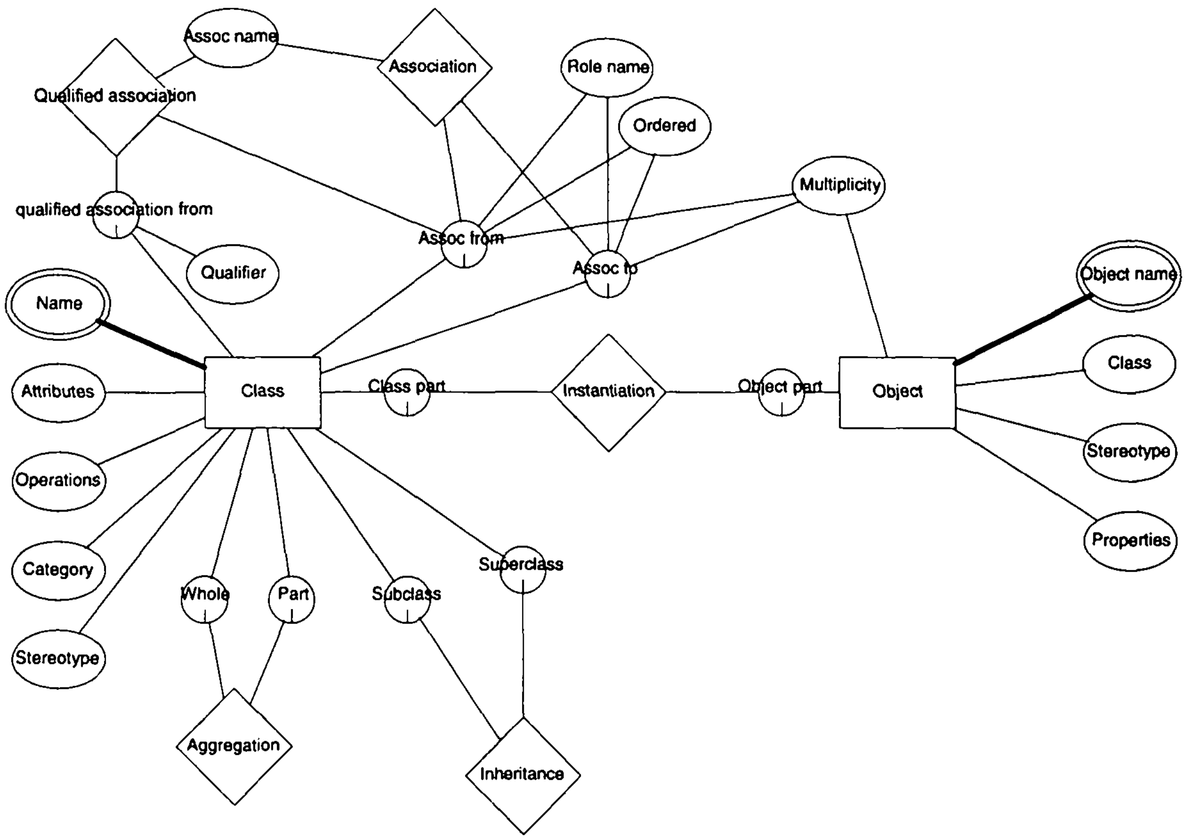

Figure 3. OPRR model of Unified CD

ANALYSIS OF THE METHODS

In this chapter we present the analysis of the techniques based on method metrics and concept classification. We then discuss shortly the findings. The metrics used here are described in greater detail in Rossi and Brinkkemper (1995). We use only a few of the metrics of the suite presented there to show how they can be applied when analysing the evolution of techniques.

Independent measures

1. $n(O_T)$

The *count of object types per technique*. This metric shows the number of individual object types used to specify object systems.

2. $n(R_T)$

The *count of relationship types per technique*. This is the number of concepts, which are used for describing connections between objects.

3. $n(P_T)$

The *number of property types per technique*. The reader should notice, that this metric measures the total number of property types in the whole technique, whereas the following metrics count properties of individual object types or relationship types.

Aggregate metrics

The independent metrics above described the individual characteristics of techniques. In this section we propose some aggregate metrics, that can be used to measure the overall complexity of the technique.

4. $$C(M_T,o) = \frac{P_o(M_T,o)}{\sum_{\{R_1 \in R : \exists b \in r(R_1 = b \cdot i) \wedge (\exists \sigma \in (b \cdot j) : (o \in \sigma \cdot j))\}} P_R(M_T,e)}$$,where $i = (1,0)$ and $j = (0,1)$

5. $$\overline{C}(M_T) = \frac{1}{n(O_T)} \sum_{o \in O_T} C(M_T,o)$$

The quotient (formula 4) shows the division of work in this technique, i.e. are things described by their internal properties, or by external connections. The quotient will get higher values if there are many properties and a few relationship types with a few properties. Formula 5 gives the average for the whole technique.

$$6.\ C'(M_T)=\sqrt{n(O_T)^2+n(R_T)^2+n(P_T)^2}$$

The total conceptual complexity of a technique is the complexity vector in a three-‑dimensional co-ordinate system. The vector can be compared with other techniques. The idea of using the complexity vector is, that one can see the complexity of the technique by looking at how long the vector is and at~~in~~ the same time one can see the "style" of the technique by looking at in which direction the vector goes.

Comments on metrics

The values obtained from the different Class Diagrams are shown in Table 1. We can shortly notice, that all techniques use a relatively small number of object types, but interestingly the new Unified Class Diagram uses the smallest number of them. The same applies to relationships as well, however it is important to notice, that the huge number of relationships in OMT CD comes from the representational variations of the association cardinalities. The number of properties is, interestingly enough, such that the Unified falls between its two ancestors.
The Formula 5 shows, that there is a clear tendency to move from the relationship orientation to more property oriented style in Unified. We claim that this tendency comes from the difficulties to apply the relationships in practise.

Technique	**1**	**2**	**3**	**5**	**6**
OODA Class Diagram	3,00	9,00	10,00	,70	13,78
Class Diagram	5,00	15,00	14,00	,91	21,12
Unified Class Diagram	2,00	5,00	13,00	2,65	14,07

Table 1. Metric values obtained from the techniques

In the Figure below we present the boxplots of the first three formulas, which is a five number summary of
smallest observed value that isn't outlier, lower quartile, median, higher quartile and largest observed value that isn't outlier. Notice that extreme values (values which are more than 1.5 times the difference between the lower and upper quartile outside of the quartiles) are plotted separately.

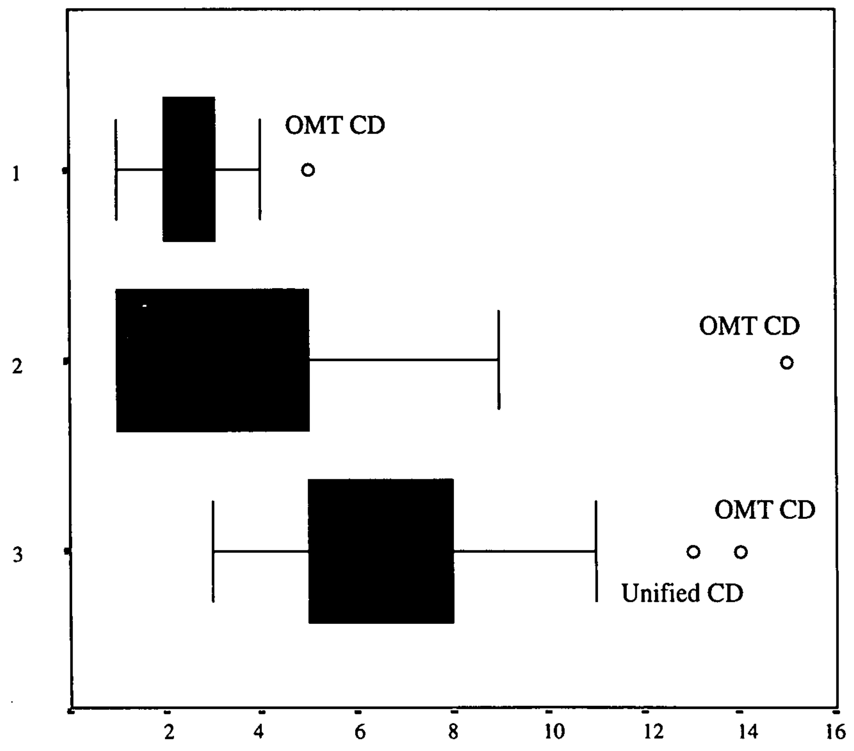

Figure 4. Boxplots of Formulas 1, 2 and 3

It is interesting to see, that OMT CD is an outlier in all of the formulas. OODA has values, that are near the maximum line, but not above it. Unified CDs use a "normal" number of relationships and roles, but a big number of properties.

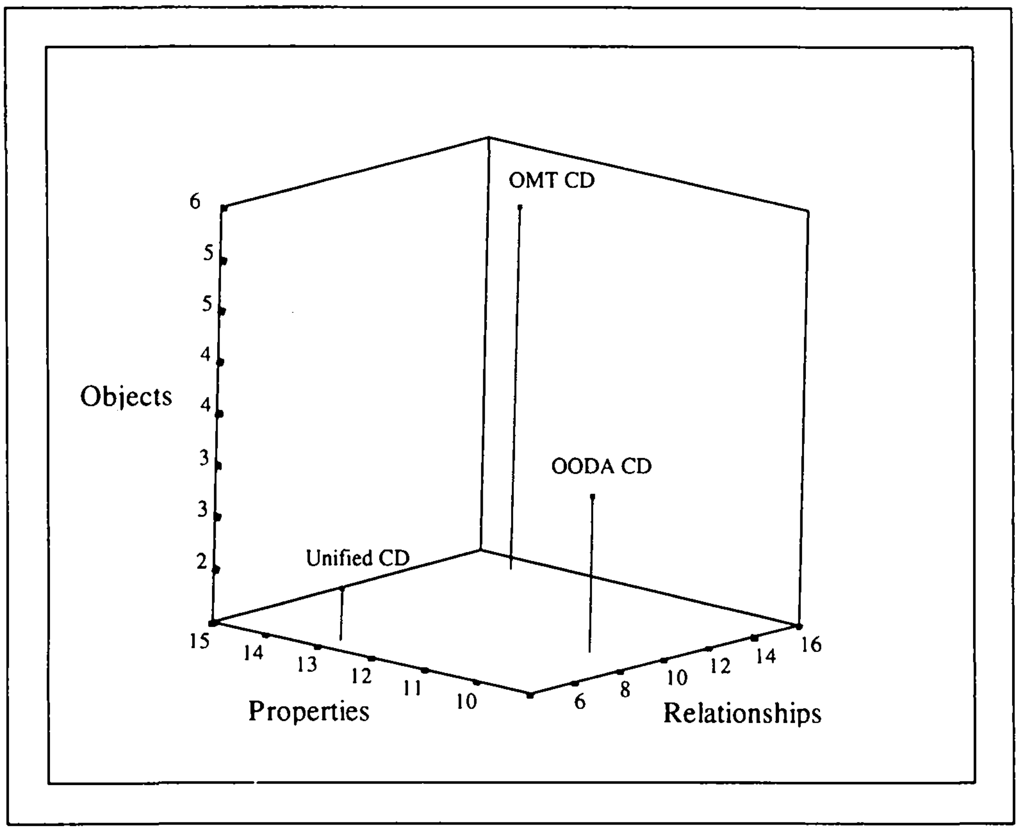

Figure 5. Technique cube

The technique cube in Figure 5 is used to show the complexity vectors of the techniques. The Class Diagram technique is the most complex by this measure as it uses properties and relationships extensively, but it contains an average number of objects. Unified CD's differ in style from the others, as the have a small number of objects and relationships, but a lot of properties. To summarise our findings, we can say, that it is interesting to notice, that by a metrical analysis the techniques evolution doesn't seem to necessarily lead all the time to more complex techniques.

CONCLUSIONS

In this paper we have compared generations of separate techniques (OMT and OODA class diagrams) and their merger, Unified class diagram. Our goal was to show how the method metrics can be used to analyse different techniques. To obtain a thorough understanding, one needs to use these metrics together with other comparison aids such as Iivari's classification hierarchies Iivari and Kerola (1983) and tabular feature comparisons Hong, Goor and Brinkkemper (1992). The proposed metrics are relatively simple to understand and easily implemented in a tool. The comparison of the techniques showed, that the authors have merged their previous techniques in such a way, that the total complexity has actually gone down.
In the future we shall expand this research to contain all techniques of the methods and we shall also use qualitative analysis of the individual components to show how their definitions have evolved.

REFERENCES

Booch, G. (1991), **Object-Oriented Design with Applications**, BenjaminCummings, Redwood City, California.

Booch, G., J. Rumbaugh (1995), **Unified Method for Object-Oriented Develoment**, Rational Software Corporation, Santa Clara, US.

Champeaux, D. de (1991), **A comparative study of Object Oriented Analysis Method**, Research Report, HP Laboratories.

Hong, Shuguang, Geert van den Goor and Sjaak Brinkkemper (1992), **A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies**, pp. 689-698 in

Proceedings of the 26th Hawaii International Conference on Systems Science, J. Nunamaker, R. Sprague and S. Hong (Ed.), IEEE Computer Society Press, USA.

Iivari, J., P. Kerola (1983), **A sociocybernetic framework for the feature analysis of information systems development methodologies**, pp. 87--139 in Information Systems Methodologies: A Feature Analysis, T. W. Olle, H. G. Sol and C. J. Tully (Ed.), North--Hollland, Amsterdam.

Iivari, Juhani (1994), **Object-oriented information systems analysis: A comparison of six object-oriented analysis methods**, pp. 85--110 in Methods and Associated Tools for the Information Systems Life Cycle *(A-55)*, A. A. Verrijn-Stuart and T. W. Olle (Ed.), Elsevier Science B.V. (North-Holland).

Olle, T. W., H. G. Sol and A. A. Verrijn-Stuart (1982), **Proceedings of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies**, North-Holland, Amsterdam.

Rossi, M., J.-P. Tolvanen (1994), **Metamodeling approach to method comparison: A survey of a set of ISD methods**, Working Paper, University of Jyväskylä, Jyväskylä.

Rossi, M., S. Brinkkemper (1995), **Metrics in Method Engineering**, pp. 200-216 in Advanced Information Systems Engineering, Proceedings of the 7th International Conference CAiSE'95, J. Iivari, K. Lyytinen and M. Rossi (Ed.) No. 932, Springer-Verlag, Berlin.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen (1991), **Object-Oriented Modeling and Design**, Prentice--Hall, Englewood Cliffs, NJ, USA.

Smolander, K. (1990), **OPRR - A model for modeling systems development methods**, Proceedings of the Second Workshop on The Next Generation of CASE Tools, University of Jyväskylä.

Song, X., L. Osterweil (1992), **Towards Objective and Systematic Comparisons of Software Design Methodologies**, IEEE Software 18(5) pp.43--53.

Welke, R. J. (1988), **METABASE A Platform for the Next Generation of Meta Systems Products**, Proceedings of CASE Studies Conference, Meta Systems, Ann Arbor, Michigan.