# Virtual Classes and Their Implementation

Ole Lehrmann Madsen

Computer Science Department, Aarhus University,
Åbogade 34, DK-8200 Århus N, Denmark
Ole.L.Madsen@daimi.au.dk

**Abstract**. One of the characteristics of BETA [4] is the unification of *abstraction* mechanisms such as class, procedure, process type, generic class, interface, etc. into one abstraction mechanism: the *pattern*. In addition to keeping the language small, the unification has given a systematic treatment of all abstraction mechanisms and leads to a number of new possibilities.

One of the interesting results of the unification is the notion of *virtual class* [7,8], which is the BETA mechanism for expressing genericity. A class may define an attribute in the form of a virtual class just as a class may define an attribute in the form of a virtual procedure. A subclass may then refine the definition of the virtual class attribute into a more specialized class. This is very much in the same way as a virtual procedure can be refined - resulting in a more specialized procedure. Virtual classes can be seen as an object-oriented version of generics. Other attempts to provide genericity for OO languages has been based on various forms of parametric polymorphism and function application rather than inheritance. Virtual classes have been used for more than 15 years in the BETA community and they have demonstrated their usefulness as a powerful abstraction mechanism. There has recently been an increasing interest in virtual classes and a number of proposals for adding virtual classes to other languages, extending virtual classes, and unifying virtual classes and parameterized classes have been made [1,2,3,13,14,15,16,17].

Another distinguishing feature of BETA is the notion of *nested class* [6]. The nested class construct originates already with Simula and is supported in a more general form in BETA. Nested classes have thus been available to the OO community for almost 4 decades, and the mechanism has found many uses in particular to structure large systems. Despite the usefulness, mainstream OO languages have not included general nesting mechanisms although C++ has a restricted form of nested classes, only working as a scoping mechanism. Recently nested classes has been added to the Java language.

From a semantic analysis point of view the combination of inheritance, and general nesting adds some complexity to the semantic analysis, since the search space for names becomes two-dimensional. With virtual classes, the analysis becomes even more complicated – for details see ref. [10].

The unification of class and procedure has also lead to *an inheritance mechanism for procedures* [5] where method-combination is based on the **inner**-construct known from Simula. In BETA, patterns are *first-class values*, which implies that procedures as well as classes are first-class values. BETA also supports the notion of *class-less* objects, which has been adapted in the form of anonymous classes in Java. Finally, it

R. Wilhelm (Ed.): CC 2001, LNCS 2027, pp. 1–2, 2001.

might be mentioned that BETA supports *coroutines* as well as *concurrent active objects*. For further details about BETA, see [6,9,11]. The Mjølner System is a program development environment for BETA and may be obtained from ref. [12].

## References


1. Bruce, K., Odersky, M., Wadler, P.: A Statically Safe Alternative to Virtual Types, In: Jul, E. (ed.): 12th European Conference on Object-Oriented Programming, Brussels, June 1998. Lecture Notes in Computer Science, Vol. 1445. Springer-Verlag, Berlin Heidelberg New York, 1998.
2. Ernst, E.: Propagating Class and Method Combination, In: Guerraoui, R. (ed.): 13th European Conference on Object-Oriented Programming, Lisbon, June 1999. Lecture Notes in Computer Science, Vol. 1628. Springer-Verlag, Berlin Heidelberg New York, 1999.
3. Igarashi, A., Pierce, B.: Foundations for Virtual Types, In: Guerraoui, R. (ed.): 13th European Conference on Object-Oriented Programming, Lisbon, June 1999. Lecture Notes in Computer Science, Vol. 1628. Springer-Verlag, Berlin Heidelberg New York, 1999.
4. Kristensen, B.B., Madsen, O.L., Møller-Pedersen, B. Nygaard, K.: Abstraction Mechanisms in the BETA Programming Language. In Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages. Austin, Texas, January 1983.
5. Kristensen, B.B., Madsen, O.L. Møller-Pedersen, B., Nygaard, K.: Classification of Actions or Inheritance also for Methods. In: Bézivin, et al. (eds.): 1st European Conference on Object-Oriented Programming, Paris, June 1987. Lecture Notes in Computer Science, Vol. 276. Springer-Verlag, Berlin Heidelberg New York, 1987.
6. Madsen, O.L., Møller-Pedersen, B., Nygaard, K.: Object-Oriented Programming in the BETA Programming Language. Addison Wesley/ACM Press, Wokingham, England, 1993. (Out of print, but a reprint can be obtained from [12]).
7. Madsen, O.L., Møller-Pedersen, B.: Virtual Classes, a Powerful Mechanism in Object-Oriented Languages. In: Proc. OOPSLA'89, New Orleans, 1989.
8. Madsen, O.L., Magnusson, B., Møller-Pedersen, B.: Strong Typing of Object-Oriented Languages Revisited. In: Proc. OOPSLA'90, Ottawa, Canada, 1990.
9. Madsen, O.L.: Open Issues in Object-Oriented Programming — A Scandinavian Perspective. Software Practice and Experience, Vol. 25, No. S4, Dec. 1995.
10. Madsen, O.L.: Semantic Analysis of Virtual Classes and Nested Classes. In: Proc. OOPSLA'99, Denver, Colorado, 1999.
11. Madsen, O.L: Towards a Unified Programming Language. In: Bertino, E. (ed.): 14th European Conference on Object-Oriented Programming. Lecture Notes in Computer Science, Vol. 1850. Springer-Verlag, Berlin Heidelberg New York, 2000.
12. The Mjølner System: http://www.mjolner.com
13. Shang, D.: Subtypes and Convertible Types, Object Currents, 1(6), June 1996.
14. Thorup. K.K.: Genericity in Java with Virtual Classes. In: Aksit, M., Matsouka, S. (eds.): 11th European Conference on Object-Oriented Programming. Lecture Notes in Computer Science, Vol. 1241. Springer Verlag, Berlin Heidelberg New York, 1997.
15. Torgersen, M.: Virtual Types are Statically Safe, 5th Workshop on Foundations of Object-Oriented Languages, January 1998.
16. Thorup, K.K., Torgersen, M.: Structured Virtual Types, informal session on types for Java, 5th Workshop on Foundations of Object-Oriented Languages, January 1998.
17. Thorup, K.K., Torgersen, M.: Unifying Genericity — Combining the benefits of Virtual types and parameterized types. In: Guerraoui, R. (ed.): 13th European Conference on Object-Oriented Programming, Lisbon, June 1999. Lecture Notes in Computer Science, Vol. 1628. Springer-Verlag, Berlin Heidelberg New York, 1999.