Simula: Mother Tongue for a Generation of Nordic Programmers

Yngve Sundblad

Simula: Mother Tongue for a Generation of Nordic Programmers

Yngve Sundblad

Human-Computer Interaction, Computer Science and Communication
KTH, 10044, Stockholm, Sweden
y@kth.se

Abstract. With Simula 67 Ole-Johan Dahl and Kristen Nygaard invented object-oriented programming. This has had an enormous impact on program development tools and methods in the world, well accounted in conferences and books, on programming languages and object-oriented programming, and on software pioneers. Early influenced were computer scientists in the Nordic countries who from about 1970 had Simula as the main programming tool, "mother tongue." This paper gives a first-hand account of experience of a unique early introduction of object-oriented programming for higher education in computer science and in computer programming, which provided powerful program development tools long before other educational institutions, especially as it coincided with the introduction of powerful interactive systems. The paper also challenges the misconception that Simula is primarily a tool for simulation by illustrating how it was used to teach general computer science and programming concepts with more general-purpose constructs than most contemporary languages, except perhaps Lisp.

Keywords: Computer science education, Nordic programmers, object-oriented programming, Simula

1 Introduction

"*From the cold waters of Norway comes Object-Oriented Programming*"; this is the first line in Bertrand Meyer's widely used textbook titled, "Object Oriented Software Construction" [1].

Based on the previous development of Simula I in 1961–65, Ole-Johan Dahl and Kristen Nygaard in 1967 introduced Simula 67, an extension of Algol 60 with the basic concepts of object-oriented programming [2, 3].

The development of those concepts and the language Simula, the contributions of Ole-Johan Dahl and Kristen Nygaard, and their impact are well described in history of computing publications. We find accounts in books resulting from the ACM conference on History of Programming Languages, 1979 [4], the IEEE conference on History of Object-oriented Programming, 1994 [5], and the SD&M conference on Software Pioneers, 2001 [6, 7].

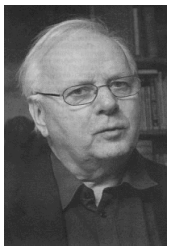

Fig. 1. O.-J. Dahl & K. Nygaard.

The development of computing education in the Nordic countries has been thoroughly described in several sessions at the History of Nordic Computing conferences in 2003 and 2007, as in [8].

Here, I will concentrate on the impact of Simula on computing education in universities in the Nordic countries. More specifically, as it affected computing education in Sweden and even more specifically at KTH and at Stockholm University.

Ole-Johan Dahl and Kristen Nygaard first presented Simula to us in 1968 at a seminar. As teachers of programming to engineering and science students mainly in the procedural paradigm of ALGOL-60 and to some extent Fortran, we saw Simula as a revelation. We immediately grasped and saw the power of the object-oriented concepts extending the syntax of Algol, which until then was our main programming tool (with some use also of Fortran and Basic).

We have kept contacts with the Scandinavian academic object-oriented programming environments, especially in Aarhus, Lund and Oslo, but the experience accounted for here could certainly be complemented by experience from other Nordic computer science educational institutions.

2 Simula Concepts

There is a misconception, especially in the US, that Simula was just a tool for simulation applications. Simula's main importance was the introduction of impressively powerful new concepts that are useful for and facilitate development of interactive and other computer applications as the predominance of the object-oriented concepts today shows. Someone coined that it is an acronym for "SIMple Universal Language." The main new concepts were the following.

- *Class* of similar Objects (in Simula declaration of CLASS with data and actions)
- *Objects* created as *Instances* of a Class (in Simula NEW object of class)
- *Data attributes* of a class (in Simula type declared as parameters or internal)
- *Method attributes* of, patterns of action (in Simula declared as PROCEDURE)
- *Message passing*, calls of methods (in Simula dot-notation)
- *Subclasses* that inherit from superclasses
- *Polymorphism* with several subclasses to a superclass
- *Co-routines* (in Simula Detach – Resume)

- *Encapsulation* of data supporting abstractions

3 Teaching Computer Science Concepts with Simula

Simula soon assumed a role as a second course language (after Algol or Fortran or Basic) in many academic programming environments. As an example, at KTH in Stockholm, all students from 1975 to about 1990 used the "Algol part" (the procedural subset) of Simula in their first programming course [9] while many students took a second course in object-oriented programming with the entirety of Simula [10]. At Stockholm University, Simula used teaching objects, program structures, and data structures until 1997, illustrated by a comprehensive textbook from 1993 [11]. Also at Lund University, Simula was used in education until 1997 as illustrated by a textbook from 1992 [12].

From 1972 onwards, this was a uniquely early introduction of object-oriented programming for education, applications and research, with continued development and spread to national industry and international academics environments.

Simula was also our tool for teaching program and data structures, with no real competitor except Lisp, which took over in the computer science specialist education in the late 1980s through the brilliant textbook and lectures on "Structure and Interpretation of Computer Programs" by Abelson and Sussman at MIT, using the Lisp dialect Scheme [13].

At the Stockholm Computer Centre, we taught an ongoing education one-week course in Simula twice a year from 1972 to about 1980, with about twenty participants from industry, academia, and defense research, mainly with backgrounds in Fortran. Thus, several hundred developers received an early introduction in object-oriented programming.

In computer science departments in the Nordic countries, there has been a variety of languages and systems used in teaching object-oriented programming since 1980, but the basic concepts go back to Simula and they were taught to our current generation of teachers by the older generation using Simula. We illustrate this by a list of Simula descendants used at KTH: Smalltalk, LOOPS, C++, Modula-2, now Java and Python.

We now take up some basic concepts that constitute a major computer science and programming education, introduced via Simula already in the early 1970s.

3.1 Data Structures

In the 1970s data structures were put on a firm theoretical footing by many researchers, e.g. Ole-Johan Dahl [14] and [15], Tony Hoare [14] and Nicolas Wirth [16], with the "formula" *Algorithms + Datastructures = Programs*.

In [6, p. 85], Dahl states, "The most important new concept was data structures with associated operators." That concept is very important and elegant in Simula but equally important are other new concepts in Simula such as class and inheritance.

The Simula reference variables that safely (strongly typed) can reference objects, with other references as attributes, give easy tools for building up stacks, queues, lists, trees, lattices, etc., that are proper data structures for different applications.

Together with the built-in framework SIMSET for building circular double-linked circular lists with head the references in Simula made systematic teaching of data structures powerful as illustrated in the textbooks [10–12, 14–18].

An elegant example is the following class for binary tree nodes with a method for scanning, which applied to the root makes a scan in post order. The "trick" is that it terminates because INSPECT-ing NONE has no effect.

```
CLASS node(left, right); REF(node) left, right;
BEGIN
   PROCEDURE scan(x); REF(node) x;
   INSPECT x DO BEGIN
      scan(left);
      scan(right);
      visit(THIS node);
   END of scan;
END of node;
```

3.2 Class and Subclass – Encapsulation and Inheritance

The encapsulation of data and algorithms (methods) into classes made it possible to structure programs in new ways, in many cases more appropriate and giving better overview than in the strictly procedural ways of the then predominant Algol and Fortran.

Subclass mechanisms allowed hierarchical structures and abstractions that also help students to structure and make less error-prone programs. The polymorphism, several subclasses of an abstract superclass, with rules for handling and using/overriding name collisions, gave the more advanced students powerful tools.

3.3 Co-routines and Discrete Event Simulation

An advanced feature of Simula, that allows defining the class of parallel processes and the SIMULATION framework, is the Detach-Resume-Call mechanisms.

The Detach and Resume() procedures in processes allow stopping and handing over control between them as does Detach and Call() between a class instance and the main program. These allow simulated parallelism for applications that are naturally described as co-routines and they were used as basis for true parallelism [19].

In the advanced Simula courses, SIMULATION was usually introduced, at KTH with a special textbook [20], while co-routines were treated for the most interested students.

3.4 Computer Graphics and Interaction, Model-View-Controller

We soon realized that computer graphics and interaction are very well suited for object-oriented structures and programming. It is natural to see the entities on the screen as objects, their actions and our interactions with them as invoking their methods.

In his brilliant work on Sketchpad, as early as 1963 [21] Ivan Sutherland invented several object-oriented concepts for handling the graphics on the screen such as objects, methods, and sub-objects although he does not develop it into a language but rather a hands-on system.

When introducing graphics into our courses in the 1970s, for plotters and vector and character terminals, Simula was natural to use for describing the objects and of keeping track of them is through display lists, which also is easy to implement with SIMSET in Simula [22].

Then, in the early 1980s, came the graphic workstations to us in academic Scandinavia, although developed in the US already in the mid-1970s, with Simula inspired tools such as Smalltalk.

The structuring of interactive graphic applications into MVC – Model-View-Controller, also has Nordic Simula origin, through Trygve Reenskaug, then visiting scientist at Xerox PARC [23].

MVC was introduced into our courses with Smalltalk around 1985 and is now commonly used in most programming tools for building interactive graphic applications.

4 Teaching Programming Practices with Simula

Here we look at some programming practices that emerged in the 1970s and were supported by Simula and its implementations.

4.1 Structured Programming and Programming Style

A buzz concept in the 1970s was structured programming with the “religions” hope of solving the “software crisis,” i.e. the galloping costs for development of large programs. Structured Programming also had a sound scientific basis formed in the early 1970s by theorists as Ole-Johan Dahl, Edsger Dijkstra and Tony Hoare with their ground-breaking book [14], Don Knuth and Niklaus Wirth, and practitioners as Kristen Nygaard, and Kernighan & Plaugher [24] at Bell labs.

These practices were taught extensively in our basic and advanced programming courses in Simula and survive today with other tools.

4.2 Dialog Programming Environment

A key factor in the success of using Simula in our education was its implementation on current computers. The first implementation we met in 1971 was on the IBM 360, but the real breakthrough came with the implementation for our first large dialog system, DEC-10. A Swedish team developed in the early 1970s a system for the DEC-system 10 [25], which was also basis for a dissertation at our department [26]. Ole-Johan Dahl states that "the DEC-system 10 implementation contributed considerably to the spread of Simula" [6, p. 85].

The experience of using Simula on text terminals led to abandonment of punched cards completely in 1976 and letting all our students take turns in using 30 terminals versus the DEC-10 at Stockholm Computer Centre (QZ). That led to slow response and complaints from other users. Hence, when they ported Simula to the DEC-20 and we were allowed to break the centralization dogma in Swedish academic computing, we acquired the department's own DEC-2020, "Nadja" in 1979, mainly for running Simula. The Algol part was taught to all KTH students and the entirety of Simula in second courses for students such as in engineering physics, electrical engineering, and mathematics. We soon had to expand with a DEC-2060 ("Vera") and one more DEC-2020 ("Venus"). The names represent Hope-Faith-Love in different languages.

The programming environment on the DEC computers had nice features such as the:

- SIMDDT debugger, [27], for inspection of variables at breakpoints and on forced interruptions when infinite loop is suspected, etc.
- SIMED program text editor, capitalising reserved words, making standard procedure names having first letter as capital and making proper indentations
- FQC collecting run time statistics
- SAFEIO, for safe input, not accepting input of "wrong" type

When the personal workstation revolution came in 1984, we moved some courses with Simula in computer science, programming, graphics, and interaction over to the Macintosh that survived until 1997, thereby abandoning the DEC computers in 1988. Developments and experiences in this area appear in [28–30].

Fig. 2. DEC-2020 computer "Nadja," used extensively for up to thirty parallel students' interactive development and running of Simula programs 1979–88. It was revived 2010, started at first effort, here with the proud author. Historically important as the first mainframe academic computer owned (and run) by a department (NADA, KTH) and not by the six centralized computer centers, one in each university region in Sweden.

4.3 Simula as an Inspiration for Smalltalk

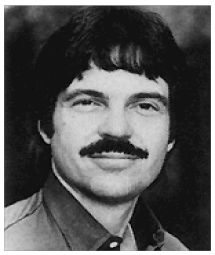

Fig. 3. Alan Kay & Adele Goldberg.

Alan Kay at Xerox PARC was one of the first American researchers that, by coincidence, learned about the Simula concepts and saw the power of object-orientation. Together with Adele Goldberg, he used it in the definition and development of the epoch-making Smalltalk language from 1972 onwards [31] and [32]:

> Then there was Simula, which the designers thought of as an extension of Algol. It was basically a preprocessor to Algol the way C++ was a preprocessor for C. It was a great concept and I was lucky enough to see it as almost a new thing. … If you combine Simula and Lisp – Lisp didn't have data structures, it had instances of objects – you would have a dynamic type system that would give you the range of expression you need. … It is not too much of an exaggeration to say that most of my ideas from then on took their roots from Simula – but not as an attempt to improve it. It was the promise of an entirely new way to structure computations that took my fancy. As it turned out, it would take quite a few years to understand how to use the insights and to devise efficient mechanisms to execute them.

In our courses on graphics and interaction, Smalltalk took over the role as a more powerful tool than Simula from about 1985, when we got the first decently efficient implementation on Sun and Tektronix stations, until about 2000. Now packages in Java play that role.

5 The Simula Heritage in Computer Science Education Today

Algol is "the Latin" of procedural programming, i.e. the dead language that most modern such languages build on. Similarly, Simula is "the Latin" of object-oriented programming.

Simula and its object orientation as inspiration in developing programming tools received recognition by the creators mentioned below:

- 1970s: Smalltalk (Alan Kay), Modula-2 (Niklaus Wirth), LOOPS (Dan Bobrow)
- 1980s: Eiffel (Bertrand Meyer), C++ (Bjarne Stroustrup), Objective C (Brad Cox), Object Pascal (Niklaus Wirth, Anders Hejlsberg)
- 1990s: Python (Guido van Rossum), Java (Jim Gosling), Objective Ada (Jean Ichbiah), C# (Anders Hejlsberg)

Some of these are extensions of existing programming languages such as general as LOOPS (of LISP), or procedural as C++, Objective C and C#, Objective Ada, Object Pascal, Java, leading to compromises and sometimes less "clean" constructions.

The extensions of procedural languages with stronger industrial backing than Simula such as C and Pascal gave object-oriented variants a much wider use. A notable example is C++, developed from C by Bjarne Stroustrup with a Scandinavian Simula background; it is still widely used in spite of its sometimes "dirty" constructions. Another is Java, where the strong backing by the Sun Corporation and the sensitivity for needs of portable devices such as mobile phones, has made it widely used. A third is C#, with its strong backing by Microsoft Corporation.

The other languages are new developments, often with elegant clean ways of expression in the Simula tradition: Smalltalk (with the "clean" paradigm that everything is objects), as well as Modula-2, Eiffel, and Python (with their strong influences of the later abstract data type and assertion theories). Some of these are quite broadly used in academic teaching but not dominant in industry.

In most academic computer science and programming educational environments as well as in the programming industry, these offsprings of Simula are the main tools used. Thus, Simula must be seen as a great success, although it is quite natural that new developments make old programming languages obsolete.

Acknowledgments. The experiences and lessons on teaching computer science and programming with Simula have been shared with many colleagues, especially those at the KTH department, NA (Numerical Analysis) from 1970 renamed into NADA (Numerical Analysis and Computer Science) in 1979; the trend also affected other Swedish and Nordic computer science departments. Stefan Arnborg, Serafim Dahl, Örjan Leringe, Kjell Lindqvist, and Staffan Romberger, all at NADA, Kalle Mäkilä, Mats Ohlin and Jacob Palme, all at FOA (now FOI), and Sten Henriksson and Boris Magnusson (Lund University) deserve special mention.

References

1. Meyer, B.: Object Oriented Software Construction. Prentice-Hall (1988)
2. Dahl, O.-J., Myhrhaug, B., Nygaard, K.: SIMULA Common Base Language. Norwegian Computing Centre (1968, 1970)
3. Birtwistle, G. M., Dahl, O.-J., Myhrhaug, B., Nygaard, K.: SIMULA BEGIN. Studentlitteratur/Auerbach, Philadelphia (1973)
4. Nygaard, K., Dahl, O.-J.: The Development of the SIMULA Language. Proceedings of the ACM SIGPLAN History of Programming Languages Conference, pp. 243–272, ACM (1979)
5. Holmevik, J.R.: Compiling SIMULA: A historical study of technological genesis. IEEE Annals of the History of Computing, 16 (4), pp. 25–37, IEEE (1994)
6. Dahl, O.-J.: The Roots of Object Orientation: The Simula Language. In: Broy, M., Denert, E. (eds.) Software Pioneers, pp. 78–90, Springer (2001)
7. Dahl, O.-J., Nygaard, K.: Class and Subclass Declarations. In: Buxton, J.N. (ed.) Simulation Programming Languages, North Holland, pp.158–174 (1967), reprinted in Broy, M., Denert, E. (eds.) Software Pioneers, pp. 91–107, Springer (2001)
8. Krogdahl, S.:The birth of Simula. History of Nordic Computing Conference 1, Oslo (2003), available as http://heim.ifi.uio.no/~steinkr/papers/HiNC1-webve
9. Romberger, S., Sundblad, Y.: Grundläggande Programmering i Simula. Teknisk Högskolelitteratur, Stockholm (1978)
10. Sundblad, Y., Romberger, S., Leringe, Ö.: Fortsatt Programmering i Simula. Teknisk Högskolelitteratur, Stockholm (1980)
11. Dahl S., Lindqvist, K.: Objektorienterad programmering och algoritmer i Simula. Studentlitteratur, Lund (1993)
12. Holm, P.: Objektorienterad programmering och Simula. KFS, Lunds Studentkår (1992)
13. Abelson, H. & Sussman, G.J.: Structure and Interpretation of Computer Programs. MIT Press (1985), available for non-commercial use at Creative Commons, http://mitpress.mit.edu/sicp/full-text/book/book.html
14. Dahl, O.-J., Dijkstra, E.W., Hoare, C.A.R.: Structured Programming. Academic Press, pp.175–220 (1972)
15. Dahl, O.-J., Belsnes, D.: Algoritmer og Datastrukturer. Studentlitteratur, Lund (1973)
16. Wirth, N.: Algorithms + Datastructures = Programs. Prentice-Hall. New Jersey (1975)
17. Pooley, R. J.: An introduction to programming in Simula. Blackwell Scientific Publications, Great Britain (1987)

18. Kirkerud, B.: Object-Oriented Programming with SIMULA. International Computer Science Series. Addison-Wesley Publishing Co. (1989)
19. Palme, J.: Making Simula into a programming language for real time. Management Informatics, vol.4, no.4, pp.129–137 (1975)
20. Siklósi, K.: Simula-Simulation. Teknisk Högskolelitteratur, Stockholm (1980)
21. Sutherland, I.: Sketchpad, a Man-Machine Graphical Communication System. Ph.D. thesis from MIT republished as Technical report no. 574 by Cambridge University (1963, 2003)
22. Haugen, Ø., Skifjeld, K.: Class graphics – A powerful tool in Interactive Computer Graphics. Proc. 10th Simula Users' Conference Norsk regnesentral (1982)
23. Reenskaug, T.: Models –Views – Controllers. Xerox PARC Learning Research Group internal memo (1979)
24. Kernighan, B.W., Plauger, P.J.: The Elements of Programming Style. McGraw-Hill, New York (1978)
25. Arnborg, S., Jones, R.W., Noble, K.H.M., Weston, P.J.: Simula 67 for the DEC System 10 Computer. FOA P Rapport C-8304-M3 (E4), Försvarets Forskningsanstalt, Planeringsbyrån, Stockholm (1971)
26. Arnborg, S.: Programming language implementation. Doctoral Thesis, Numerical Analysis, KTH, Stockholm (1972)
27. Palme, J., Wennersten, I.: SIMDDT – for conversational debugging of SIMULA programs. Simula Newsletter, vol.5, no.2 (1977)
28. Dahl, S., Sundblad, Y.: Mac Simula Application Framework. Proc 18th Simula Conference, Plzen (1990)
29. Sundblad, Y.: Teaching Object-Oriented User Interface Design. Proc Apple European Univ. Consortium, Heidelberg (1988)
30. Kjelldahl, L., Sundblad, Y.: Experience from ten years of student projects oriented towards graphic interaction, Computers and Graphics. vol.20 (1996)
31. Feldman, S.: A conversation with Alan Kay. Queue, vol.2, pp.20–30, ACM New York (2004)
32. Kay, A.: The Early History of Smalltalk. In: Bergin, Jr., T.J., Gibson, R.G. (eds.) History of Programming Languages – II. ACM Press, New York NY, and Addison-Wesley Publ. Co., Reading MA, pp. 511–578 (1996)