Kristen Nygaard is internationally acknowledged as the co-inventor with Ole-Johan Dahl of object-oriented programming, through their design of the programming languages Simula I and Simula 67.

Kristen Nygaard, Norway - 2001

With Ole-Johan Dahl, for ideas fundamental to the emergence of object oriented programming, through their design of the programming languages Simula I and Simula 67.

One exception to the broad adoption of SIMULA concepts is the notion of an active object with its own action sequence, which, strangely enough, has not been adapted to other languages. For Dahl & Nygaard, having active objects was an essential facility to be able to simulate concurrent processes from the real world.

Before the concept of object-orientation became popular, SIMULA influenced the development of new abstract data types. As a result of these ideas, Simula was extended with constructs such as public, private and protected modifiers, originally proposed by Jakob Palme.

Nygaard's original field was operations research. Early in his career he realized that computer simulations would be a useful tool. He collaborated closely with Ole-Johan Dahl who, in Kristen's words, "had an exceptional talent for programming". This collaboration led to the first Simula language, SIMULA I, based on the language ALGOL-60. SIMULA I was originally considered a system description and simulation language, not a general programming language. Dahl & Nygaard quickly realized that its simulation concepts could be applied for programming in general. As a result of this insight they designed Simula 67, later just called SIMULA. It is a general purpose programming language and, like SIMULA I, it contains Algol-60 as a subset.

SIMULA contains the core of the concepts now available in mainstream object-oriented languages such as C++, Eiffel, Java, and C#.

1. Class and object. The class concept as a template for creating instances (objects).
2. Subclass. Classes may be organized in a classification hierarchy by means of subclasses.
3. Virtual methods. A SIMULA class may define virtual methods that can be redefined in subclasses.
4. Active objects. An object in SIMULA may be the head of an active thread, technically it is a co-routine.
5. Action combination. SIMULA has an inner-construct for combining the action-parts of a class and its subclass.
6. Processes and schedulers. SIMULA makes it easy to write new concurrency abstractions, including schedulers.
7. Frameworks. SIMULA provided the first object-oriented framework in the form of Class Simulation-the mechanism it used to implement its Simulation.
8. Automatic memory management, including garbage collection.

One unexpected result was that people often found making a model in Simula to be more useful than the actual simulation results. The process of describing the application provided a valuable insight in itself. This led Kristen to formulate one of his favorite aphorisms: To program is to understand. He thought programming should not be considered a low-level technical discipline designed just to accomplish a specific task, but that writing a program should enhance understanding of the problem domain and the solutions.

As a result of these ideas, Simula was extended with constructs such as public, private and protected modifiers, originally proposed by Jakob Palme.

One unexpected result was that people often found making a model in Simula to be more useful than the actual simulation results. The process of describing the application provided a valuable insight in itself. This led Kristen to formulate one of his favorite aphorisms: To program is to understand. He thought programming should not be considered a low-level technical discipline designed just to accomplish a specific task, but that writing a program should enhance understanding of the problem domain and the solutions.

Nygaard's next advance was the development, with Peiler Händlykken and Erik Holbrek-Hansen, of DELTA. DELTA was not a programming language, but rather a system description language used to aid in modeling real world systems. Ideas intended for collaborative use by developers and users together-delta means participate in...
Kristen Nygaard - A.M. Turing Award Winner

Norwegian. DELTA was based on SIMULA, but extended with equations for describing both discrete state changes and continuous changes over time.

When Kristen was a visiting professor in Aarhus, Denmark, he initiated work with Bent Bruun Kristensen, Ole Lehmann Madsen and Birger Möller-Pedersen on the BETA programming language. BETA is a language for describing models of the real world, but, in the tradition of SIMULA, it was also to be used as an implementation language. In the design of the BETA programming language, a criterion for adding any new construct to the language was that it be meaningful both for modeling and for programming.

In the late sixties, the Norwegian Iron and Metal Workers Union contacted Kristen for help with new computing technologies. Kristen and others developed courses and books about information technologies for the Union. The project developed the first "data agreement" between a union and a company, and elements from this were later included in Norwegian legislation.

The Iron and Metal project was the first of a series of Scandinavian research projects involving users in the design of IT systems. Nygaard was motivated by a desire to empower users to have more influence on these designs. Introducing user participation in systems development was, for many years, considered to be political. Today, however, companies realize that by directly involving users in the design process, the resulting systems are often better. Methods for involving users in the design of IT systems are now known as participatory design, and are taught and practiced by many groups around the world.

Kristen was also one of the few philosophers of informatics, stemming from his early work on simulation. To be able to create a model of real-world phenomena, it is necessary to have a strong conceptual framework to understand and organize knowledge. Modeling capabilities were always central to his design of languages. One famous example of this is the subclass mechanism, developed to represent domain concept specialization hierarchies.

Kristen was thrilled by the enormous success of the object-oriented approach to programming. During a visit to Xerox PARC, Alan Kay demonstrated the Smalltalk system, and Kristen was very impressed to see how the ideas from SIMULA had inspired the Smalltalk team. He was happy to see these ideas further adapted by languages such as Flavours, Loops, C++, Eiffel, Java, C# and many others. He never participated in the critique of possible shortcomings of other languages; on the contrary, he respected their creators and acknowledged their influence on the development of object-orientation.

Hundreds of books were written on object-orientation, but Kristen found that most of these books did not do a good job in teaching the fundamental concepts. He was very concerned with education, but frustrated by what he considered a lack of quality. The goal of his last project (COOL: Comprehensive Object-Oriented Learning) was to develop first-class teaching material on object-oriented programming. He had just set up an international team of participants and was ready to start the work when he died in 2002.

Kristen initiated research on participatory design and object-oriented programming at the University of Aarhus. At the University of Oslo he initiated research on participatory design; Ole-Johan Dahl had already established a research group on object-orientation. His work on system development and on the social impacts of computing technology became the foundation of the Scandinavian School in System Development and the field of participatory design.

But Kristen was not just a pioneer and researcher in informatics. He was an engaged social and political citizen, involved in several aspects of society, including politics. During the intense political fight before the 1972 Referendum on whether Norway should become member of the European Common Market, he worked as coordinator for the large majority of youth organizations that worked against membership. He also was the leader (1990-1995) of Norway's No to the EU movement, which argued against Norwegian membership of the European Union and led to victory in the 1994 referendum.

Further information on Kristen Nygaard can be obtained from the following:

1. Memorial site for Kristen Nygaard has further information.

Author: Oie Lehrman Madsen