Nygaard, K. (1963b) April 17. Letter to C. A. Christopher, Director of Procurement, Univac (C).
Nygaard, K. (1963c) May 18. Letter to A. M. Paster, Manager Systems Research, Univac, New York. (C).
Nygaard, K. (1963d). A Status Report on SIMULA—A Language for the Description of Discrete-event Networks. In *Proceedings of the Third International Conference on Operational Research,* pp. 825–831. London: English Universities Press. (P).
Nygaard, K. (1963e) September 19. *Opparbeidelse av kompetanse innenfor Real-time Systemer* (Building up competence on real-time systems). Note, in Norwegian. (D).
Nygaard, K. (1965a) August 15. *The Software Contract between UNIVAC and the Norwegian Computing Center.* NCC Doc. (D).
Nygaard, K. (1965b). *Report on the use of SIMULA up to December 1965.* NCC Doc. (D).
Nygaard, K. (1966) February 17. Letter to S. M. Haffter, Univac Sperry Rand Corporation, Lausanne. (C).
Nygaard, K. (1967) November 3. Letter to C. A. R. Hoare, Elliott-Automation Computers, Herts., England. (C).
Nygaard, K. (1968a) January 29. Letter to Niklaus Wirth, Rechenzentrum der Universität, Zürich. (C).
Nygaard, K. (1968b) April 2. *En redegjørelse for samarbeidet mellom Det russiske vitenskapsakademi og Norsk Regnesentral om bruk av programmeringsspråket SIMULA i Sovjet* (An account of the cooperation between the Russian Academy of Science and the Norwegian Computing Center on the use in the Soviet Union of the programming language SIMULA). Note, in Norwegian. (D).
Nygaard, K. (1968c) September. *Oversikt over NR's SIMULA-engasjement* (Survey of NCC's commitment to SIMULA). Note, in Norwegian. (D).
Nygaard, K. (1968d) September. *Markedsføring av SIMULA 67* (Marketing of SIMULA 67). Note, in Norwegian. (D).
Nygaard, K. (1969) September 26. Letter to Peter Weil, Manager, Contracts and Pricing, Univac, England. (C).
Nygaard, K., and Dahl, O.-J. (1965). SIMULA—A Language for Describing Discrete Event Systems. In *Proceedings of the IFIP Congress, 65,* Vol. 2, pp. 554–555. Washington, D.C.: Spartan Books; New York: Macmillan. (P).
Palme, J. (1968). A comparison between SIMULA and FORTRAN. *BIT* **8**:203–209. (P).
Paster, A. M. (1962) October 12. Letter to Kristen Nygaard, NCC. (C).
Reitan, B. (1969) September 4. Letter to Kristen Nygaard, NCC. (C).
Roach, (1963) July 3. Telegram to Kristen Nygaard, NCC. (C).
Ross, D. T., and Rodriguez, J. E. (1963). Theoretical Foundations for the Computer-aided Design System. In *Proceedings of the SJCC,* p. 305. (P).
SIMULA Standards Group (1968).*Report from the meeting of the SIMULA Standards Group, held in Oslo, Norway, February 10, 1968.* (D).
Statutes (1967) May 23. *Statutes for the SIMULA Standards Group.* NCC Doc. (D).
Stevenson, F. (1967) November. *LOGIC, A computer program for simulation of digital logic networks.* NCC Doc. (D).
Tocher, K. D. (1963). *The Art of Simulation.* London: English Universities Press. (P).
Wang, A., and Dahl, O.-J. (1971). Coroutine Sequencing in a Block Structured Environment. *BIT* **11:** 425–449. (P).
Wegner, P. (1976) December. Programming Languages—The first 25 years. *IEEE Transactions on Computers* C-**25**(12): 1207–1225. (P).
Weizenbaum, J. (1962) March. Knotted List Structures. *CACM* **5**(3): 161–165. (P).
Wirth, N. (1968) February 14. Letter to Kristen Nygaard, NCC. (C).

TRANSCRIPT OF PRESENTATION

BARBARA LISKOV: Our speaker on SIMULA will be Kristen Nygaard. At the time that SIMULA was developed, Kristen Nygaard was the Director of Research at the Norwegian Computer Center, and apart from his work on SIMULA he was also responsible for building up the NCC, as it's called, as a research institute. He had been working in computing since 1948, and in Operations Research since 1952. Today, Kristen Nygaard, con-

tinues as Director of Research at NCC, but directs only his own projects. He also holds the rank of Professor at the University of Oslo, and his current research interests include programming languages and the social aspects of computing.

KRISTEN NYGAARD: During the writing of the paper for this conference, the authors had an abundance of letters from Jean Sammet and the language coordinators, but little communication between themselves. As it turns out, we have each chosen our own style. Those of you who have read our paper will know that it at least partially is in the form of an "action thriller." We are not certain whether it is fun to read or not, but it was on the point of becoming funnier. Two slides illustrate this.

Frame 1 shows a typical passage from the paper—from the "Greek nightclub episode." Observe the words "watching a beautiful belly dancer."

Quotation from the Greek night club episode (final version):

"While they were listening to bouzouki music, watching a beautiful belly dancer, Nickitas presented the following informal proposal,...."

Frame 1

Now observe the next to final version of the manuscript, (corrected at the last minute) in which we are not content with passive observation [Frame 2].

Also, our very competent typist felt that the sex angle was not sufficiently well handled. Her feeling materialized in terms of a very resolute insistence on writing SINSCRIPT instead of SIMSCRIPT (a prominent language which to our regret is not presented at this conference).

When one looks at one's own work at a distance, one is able to sort out the essentials and talk briefly. Having had the rather large job of writing our paper, we are in trouble because the SIMULA development once more has become very close. To sum up seven years of rather hard work in 25 minutes has become a difficult task; therefore, you must excuse me for using a manuscript. I will *close* my lecture with acknowledgments of some persons and one important activity outside of our own effort.

I will *start* by stating that SIMULA was, and is, a collective effort by many persons. SIMULA exists as a living language with an active user community, and the reason is a series of good implementations by some exceptionally competent teams. I admired the way in which John Backus brought his team members into the picture, and I wish that I

Quotation from the Greek night club episode (original version):

"While they were listening to bouzouki music, washing a beautiful belly dancer, Nikitas presented the following informal proposal,...."

Frame 2

had been able to do the same thing. I want to mention one name, however, that of Bjørn Myhrhaug, who designed SIMULA's string-handling and input–output; Ole-Johan Dahl and I very much would have liked to see him here.

SIMULA did not start as a programming language, and during all of its development stages reasoning outside traditional programming played an important part in its design. The first ideas came from operational research. My own work in that field told me that we needed a set of concepts and an associated language in terms of which one could understand and describe the complexity of the systems one had to deal with. It was also evident that simulation had to be the main tool for analysis. Consequently, from the very start in 1961, SIMULA was labeled both a system description language and a simulation programming language.

I was working within operations research, and had at the time lost contact with programming after leaving that field in 1954. I realized that I was not competent to design such a language alone, and since I knew Ole-Johan from my time at the Norwegian Defense Research Establishment, I tried to get him interested. I succeeded, and from then on, and through all the important development stages of SIMULA we worked together. It is impossible, even just between us, to find out who was responsible for which language concept. This may sound like a beautiful pastoral scene of peaceful cooperation. It was not. And the following story is true:

In the spring of 1967 a new employee at the Norwegian Computing Center came running into the telephone exchange and, very shocked, told the operator: "Two men are fighting in front of the blackboard on the first floor corridor!" The operator went out of her cubicle, listened for a few seconds, and said, "Relax—it's only Ole-Johan and Kristen discussing SIMULA!"

When we started, Ole-Johan knew much about programming and next to nothing about systems thinking. I knew something about systems and next to nothing about programming. Today Ole-Johan knows much about systems thinking.

SIMULA started from a mathematically oriented concept of a network consisting of passive customers flowing through a network of active stations. However, we realized that the processing and decision rules to be described and simulated made it necessary that the language, quoting from a very early document, "had to include a general algorithmic language such as ALGOL or FORTRAN." John Backus said in his speech yesterday that language design was a relatively easy part of the task. I understood him that way. That was not the case with the two SIMULA languages. It was a long and tedious process of working and reworking our concepts before we gradually arrived at the SIMULA of today.

At this point I want to mention another important aspect of our method of working. We always discussed how to implement as we went along, and never accepted anything unless we knew it could be implemented efficiently.

Our network concept dissolved for two reasons: we discovered that we could regard the networks as consisting of active customers and passive stations equally well as the opposite, what we labeled a "dual view." We then realized that an in-between approach in many situations was very useful, and also discovered many situations which couldn't be described well by the network concept.

At this stage, the influence of ALGOL 60 became more and more prominent. It had at earlier stages been both an inspiration and an obstacle. We found the stack to be the obvious way of organizing a system component's action sequence. We believed, and it is spe-

cifically stated in our SIMULA contract with Univac, that we should implement SIMULA through a preprocessor to ALGOL 60.

In the spring of 1963 we were almost suffocated by the single-stack structure of ALGOL. Then Ole-Johan developed a new storage management scheme in the summer and autumn of 1963. The preprocessor idea was dropped, and we got a new freedom of choice. In February 1964 the process concept was created, which is SIMULA 67's class and object concept (but integrated at that time in simulation facilities, and without a subclass feature).

Frame 3 sums up some essential facts about SIMULA I. When we moved on to SIMULA 67 we had realized that SIMULA also was a powerful general programming language. Sometimes it is remarked that SIMULA's usefulness as a tool for implementing data types was a welcome lucky coincidence. I do not agree because many of the uses to which SIMULA has been put are the logical and necessary consequences of the system approach married to the ALGOL block concept.

It is true, however, that we did not grasp all of the implications of SIMULA at the time of creating its concepts. The transition from SIMULA I to SIMULA 67 is described in our paper, and I only want to summarize this by the next slide [Frame 4]. When we developed SIMULA we were not concerned with most of the questions which were essential to the ALGOL 60 fathers. We more or less took the algorithmic capabilities of ALGOL 60 for granted. Alan Perlis said to me yesterday that we should have developed SIMULA as the common superstructure for both ALGOL, FORTRAN, and possibly other languages. That was also our initial idea. But as our insight in what we regarded as a proper systems approach increased, this became more and more impossible. The ALGOL block concept which, with its integration of data, patterns for actions (procedure declarations) and actions became the cornerstone of our thinking.

At this stage, I want to illustrate what I mean by "system thinking." Our main reference frame was a set of examples of systems in the world around us: job shops, airports, epidemics, harbors, etc. For this reason, the dynamic systems created by the program execution was first and foremost a model of the system described by the program. This reasoning was carried over to our understanding of more traditional parts of programming as well. Let us examine this approach in a little more detail.

In Frame 5 a sketch of an ALGOL program is showed on the left and a simplified model of the corresponding program execution on the right. What is physically generated is organized as a stack. Systems which may be usefully thought of as such stacks, are described conveniently by ALGOL. Frame 6 states this with other words.

Moving on to SIMULA, each component, now called an "object" in our model system (the program execution) is itself a stack. And therefore what is SIMULA? Here we have a

SIMULA I

developed:
  June 1961–March 1964

design objectives:
  system description
  simulation programming

basic feature:
  the process concept

Frame 3

SIMULA 67

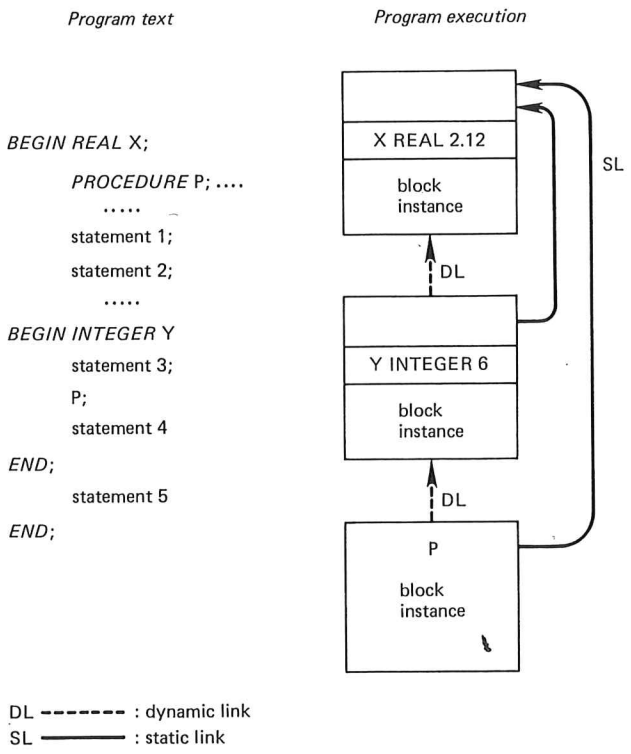
developed:
December 1966–January 1968

design objectives:
system description
high level programming
application languages
(e.g. simulation)

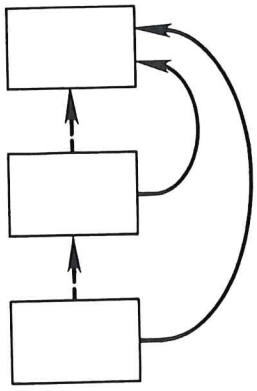
basic features:
the object/class concept
prefixing and subclasses
the virtual concept

Frame 4

number of stacks with the static enclosures, and SIMULA is the world regarded as a nested collection of interacting stacks [Frame 7]. When we examine what was new in SIMULA 67, two main and interrelated features were prefixing and virtual procedures. The prefixing with block concatenation made it possible for us to use Tony Hoare's ideas of reference qualification, and still keep the flexibility we wanted. It also provided the possibility of building up hierarchies of concepts [Frame 8]. As you know, using prefixing by Class *A*, an object of Class *B* contains an *A*, and its prefix and main parts are glued together as one integrated block instance.



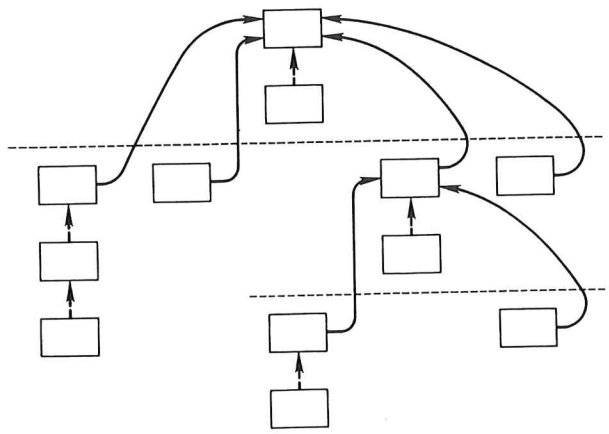
Frame 5



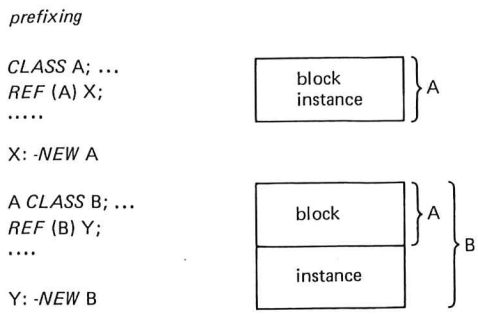
Frame 6. ALGOL 60—the world regarded as a stack of block instances.

One thing which should be mentioned in passing is that we tried by a last minute desperate effort to get a type concept into SIMULA 67 (in terms of in-line types). We did not succeed for reasons which are explained in our paper.

My last visit to the U.S. was in 1970. At that time the class concept only had a certain curiosity and entertainment value. I except people like Don Knuth, John McNeley, Bob Bemer, Al Paster, and some others. Today it's interesting and pleasant to observe that the situation is different. But—and there is a "but"—I still think that many people who refer to SIMULA only grasp parts of it. In fact, those who understand SIMULA best are not the people in programming research, but rather SIMULA's simulation users. The computer scientists like SIMULA as a vehicle for implementing data types. But many of them have never discovered the use and implication of the class/subclass feature. If they have, most have not exploited the virtual concept. And only very few, including Tony Hoare and Per Brinch Hansen, have realized what I feel is the most important property of SIMULA, that it is a multistack language. The computer-based systems we now have to



Frame 7. SIMULA—the world regarded as a nested collection of interacting stacks.



Frame 8

implement are networks of human beings, production equipment and computing equipment. For these systems I'm convinced that SIMULA's multistack framework is useful.

When I planned this presentation, I expected to use much of my time in discussing SIMULA politics. As I wrote along, my mind changed, as you have observed. Politics was, however, an essential part of the SIMULA venture, and it is described in detail in our paper. I will conclude my speech by pointing out what I feel were the most essential political elements of that venture.

When SIMULA 67 was developed, the Norwegian Computing Center employed approximately 120 persons. Of these 120, three persons could be assigned to language development. Backus told us that he got the resources he wanted. Grace Hopper's situation was more like ours. [Refer to the FORTRAN and Keynote presentations. Ed.] In 1962 we were told that (1) there would be no use for SIMULA; (2) there would be use, but it had been done before; (3) we would not succeed; (4) we should not make such efforts in a small and unimportant country like Norway.

In 1967 we were told that SIMULA was wonderful, but the lifetime of a programming language was only three to five years, and we have to earn back our expenses during that time period.

We had very small resources, and we had to fight for them. We wanted SIMULA to be an "existing" language, and our definition of that term is given on Frame 9.

These were our ambitions. To achieve these objectives, we needed the compilers to be of "high standard," a term which is defined on Frame 10.

The period from the spring of 1968 until the summer of 1970 was a crucial phase in SIMULA 67's life. The Control Data implementations were on the way, but SIMULA would not exist unless we got it onto IBM and Univac, and this was something which we

"Existing" language:

— available on most of the major computer systems

— being used over a long period of time by a substantial number of people throughout the world

— having a significant impact upon the development of future programming languages.

Frame 9

Condition for "existence"—
"high standard":

— Compilation and run time execution speeds comparable with best ALGOL 60 compilers.

— Availability of comprehensive and well written documentation and educational material.

— The existence and effective operation of distribution and maintenance organizations for the compilers.

Frame 10

had to do at the Computing Center. And we were in a difficult situation at that time, described in the paper.

In the summer of 1970 the NCC compilers had passed their "point of no return"; it was more costly to drop them than to complete them. Still, our resources were limited, as shown on Frame 11.

For this reason it was essential for SIMULA's success that we already in 1967 had designed an organizational strategy which was carried out during the subseqent five years [Frame 12].

The Norwegian Computing Center was at all stages, except a brief interval, very loyal to the SIMULA effort. And when you are shown loyalty, you also feel loyalty in return. Captain Grace Hopper concluded her opening address by expressing her gratitude and dedication to the organization she had served. I feel the same towards the Norwegian Computing Center and our comrades there. I could stop here, but I have to thank some of those outside the Norwegian Computing Center who were essential to the success of SIMULA. What I'll present to you is my own "short list" of names. Ole-Johan's may be slightly different, so I'll give it to you as my list.

First of all—and underlined—is Jan V. Garwick, the father of computer science in Norway. He was Ole-Johan's and my own first boss, and we are in great professional debt to him. Then follows a series of people at that time associated with Univac. Without Univac's immediate interest at an early stage, SIMULA would at least have been seriously delayed.

Stig Walstam brought us in contact with key Univac people in May 1962, as, e.g., Bob Bemer who listened for twenty minutes and then interrupted me by stating "Why don't you go to Munich [the IFIP World Conference in Munich] and discuss it?" And he was interested in discussing negotiations with Univac. But then Jim Nickitas, the man who got the whole thing started within Univac, the man who had faith in us at the Computing Cen-

Norwegian Computing Center:
120 persons

IBM SIMULA team:
7 persons (peak)

Univac SIMULA team:
2 persons

Frame 11. Resources.

Strategy:

— getting SIMULA 67 implemented on IBM, Univac and Control Data computers

— SIMULA Common Base Conference (June 1967) and SIMULA Common Base Language

— SIMULA Standards Group (SSG)

— Association of SIMULA Users (ASU)

— SIMULA Newsletter

Frame 12

ter, and was material in the Computing Center getting the 1107 computer. Al Paster, our main contact during our work with the concept. Bernie Hausner, staying with us for a while, and telling us about SIMSCRIPT. We learned from it; we copied it to a slight extent, and we learned what we wanted to be different. Joe Speroni behind our ALGOL compiler for our SIMULA I; Don Knuth, John McNeley—who very generously supported us. Eugen I. Yakovlev and Kirill S. Kusmin in the Soviet, at the Zentralniya Ekonomika Mathematicheskiya Institut in Moscow. Then Tony Hoare who is the single person whose ideas have meant most to us. Jean Ichbiah has done a big job in telling other people about SIMULA. Robin Hills, the first chairman of the Association of SIMULA Users. Jacob Palme . . . (I understand from the Johnny Carson show you have a big research project going on here in the states: trying to develop a T-shirt without anything printed on it! Henry Tropp has asked us to give references. I'm loyal to that. But it has not been developed in Sweden, and Jacob Palme is a person who has SIMULA 67 on his T-shirt.) But he has done very very much more. There are many others, and I'll use my leeway by ending up and saying—above all—well, it is at the bottom, but it is in fact, on top—ALGOL 60. Our gratitude first and foremost goes to ALGOL 60, and to all of the elegance and clearness which this language has, and which we have tried to carry over in SIMULA. Thank you.

TRANSCRIPT OF DISCUSSANT'S REMARKS

BARBARA LISKOV: As I'm sure you're all aware, SIMULA was in fact a two-man project. The other member of this team was Ole-Johan Dahl, and he's going to be a discussant for SIMULA today. At the time that the ideas for SIMULA came up, he was working for the Norwegian Defense Research Establishment for Jan Garwick. As a matter of fact, he was working on an implementation of an ALGOL-like high-level language that was going to be implemented—if I have this correctly—on a machine with 1000 words, and a drum of 16K words. Anyway, in 1962, Ole-Johan Dahl moved to the Norwegian Computing Center and started working full-time on SIMULA. Today he's Professor of Informatics at the University of Oslo and his primary research interest is in program specification and verification.

OLE-JOHAN DAHL: I would like to use another few minutes in talking about ALGOL and its surprisingly many kinds of applications, especially of course the block concept of

ALGOL, which turned out to be able to model all we felt that we needed for simulation models—that was the first SIMULA language. It also could be extended to the more general purpose class concept of SIMULA 67.

I have listed five different kinds of uses of blocklike constructs in SIMULA 67, all under the disguise of what we call a *class* or *class body*. I know that SIMULA has been criticized for perhaps having put too many things into that single basket of class. Maybe that is correct; I'm not sure myself. But it was certainly great fun during the development of the language to see how the block concept could be remodeled in all these ways.

So, the first and simplest instance of class is of course the pure data structures that you have in languages such as Pascal, ALGOL 68 and others; recordlike things, which you get out of an ALGOL block simply by deleting the block tail, keeping the declarations of variables and arrays.

The next example can be called "generalized data objects." You get them by including not only variable declarations in your class body block head, but also procedure declarations. The discovery was made in 1964–1965 that these so-called "procedure attributes" could be useful. And people like Tony Hoare have since shown us that what is really lying there is the concept of an abstract data object, with the procedures as abstract operators. The only thing we needed to do to ALGOL in order to make blocks look like data objects, was simply to devise a naming mechanism for block instances. And a mechanism of looking into blocks from the outside.

Next on my list is the process concept. It merely required us to include some very simple mechanisms for coroutinelike sequencing in addition to ALGOL's procedure activation mechanism. Having that, we had all the power of ALGOL programs going in quasi-parallel.

Then, the fourth on the list is the class prefixes. Now this was, as Kristen mentioned, a less trivial addition. The idea of prefixing one class by another one, and thereby composing a composite object consisting of two layers. But this enabled us to use the class mechanism for a kind of abstraction, collecting common properties of different kinds of objects into a single class, and making it available for use at later times as a kind of plug-in unit, where a given class could be extended into more concrete classes at later times by adding more properties.

And finally, the prefix mechanism could be used for ordinary in-line blocks too. This turned out to be a very interesting application, rather like collecting together sets of interrelated concepts into a larger class, and then using that larger class as a prefix to a block. The class would function as a kind of predefined context that would enable the programmer to program in a certain style directed toward a certain application area. Simulation, of course, was one of the ideas that we thought of, and we included a standard class in the language for that particular purpose.

There are two more points on my list of mechanisms that we proposed for inclusion in the language at the Common Base Conference which was held in June or late May of 1967. To use these class objects as generalized variables so that the concept of a class could be used rather like a generalized type concept, like integer and real. And finally we also saw the possibility of unifying the class and procedure concepts.

Now, as I said, it was great fun to see how easily the block concept could be remodeled and used for all these purposes. It is quite possible, however, that it would have been wiser to introduce a few more specialized concepts, for instance, a "context" concept for the contextlike classes. As a matter of fact, one of our disappointments with the usage of

SIMULA during the past years is the relatively little usage of the context-building capability of the language. Maybe this is also due to the fact that it is not at all an easy task to make good contexts. Thank you.

TRANSCRIPT OF QUESTION AND ANSWER SESSION

BARBARA LISKOV: There have been a number of questions, and I'll start with this one from Richard Miller: "Do you think the fact you were in a small group and country working on a project that many other groups were also working on helped push you on to the high quality work that you wanted?"

KRISTEN NYGAARD: Yes. I think that we benefitted from the fact that we had very complete control in a small group of what we were doing. I believe in team development of such a project. I'm skeptical of committee projects in designing languages. ALGOL 60 succeeded. They had a number of very brilliant people. They succeeded also because they had an apparently innocent but in fact very cunning secretary with devious manners, Peter Naur, to help the whole thing succeed. I address this comment also to the television cameras, which (we are told) records for history.

LISKOV: I have a question from Richard Nance at Virginia Tech: "Using the Kiviat–Lackner classification of simulation languages, both SIMULA and GPSS are described as process interaction languages. GPSS is narrowly focused on the interaction of temporary objects with permanent objects. Did SIMULA evolve from the same view, and how well can SIMULA treat interaction among permanent objects?"

OLE-JOHAN DAHL: I think it is fair to say, looking back now, that the conceptual origins are much more similar than we thought they were at the time. Our starting point was also the concept of a network through which things were flowing, and as I have understood, that is also the basic view of GPSS. About the distinction between temporary and permanent objects, I really can't see any great distinction between them. To me, a permanent object is an object that gets created at the start of an execution and lasts the whole time.

LISKOV: Here's another question from Richard Miller: "Could you comment further on the possible use of parallelism in SIMULA programs? When was this idea thought of?"

NYGAARD: I guess that you are talking about real physical parallelism as opposed to quasi-parallelism. Quasi-parallelism, to portray parallelism, was of course essential from the very outset. It occurred to us that this could be carried over to true parallelism, and in fact, there exists a note, in Norwegian, from 1963, about developing SIMULA into a real-time language. When we discovered it, among much dust, I reread it with quite some nervousness. But it turned out that the ideas there were not too bad, but of course we didn't at that time at all really understand all the problems related to physical parallelism. I think that we *could* incorporate such features in SIMULA, but I don't think it *will* be done. There are versions of SIMULA now with extended operating systems for running programs in parallel, but not in the nature of, say, concurrent Pascal.

LISKOV: Thank you very much.

FULL TEXT OF ALL QUESTIONS SUBMITTED

ULF BEYSCHLAG

SIMULA turned out to be a successful, nearly profitable programming language project. This without the strong backing of a manufacturer or a widespread academic community. What problems did you face and how do you see the chances for further projects with these characteristics?

PER BRINCH HANSEN

Many computer scientists now refer to SIMULA as the origin of the idea of abstract data types, that is, as a program module that makes a clear distinction between externally available abstract operations and internal implementation details of these abstract concepts. Was this viewpoint evident to you and Dahl from the beginning or was it a happy (unexpected) consequence of the SIMULA 67 class concept?

RICHARD MILLER

The class concept of SIMULA has become the prototype of the data abstraction techniques now coming into vogue in languages such as CLU and ALPHARD. Was this usage of the class apparent during the design? When did it become apparent?

(For Professor Dahl in particular, but. . . .) Could you describe your attempts (and eventual success) in developing the multiple stack concept for SIMULA. In particular, what options were available to you at the time and how were they used or eliminated?

Do you think the fact that you were in a small group and country working on projects that many groups were also working on helped push you on to the high quality work that you wanted?

RICHARD NANCE

Using Kiviat–Lackner classification of simulation languages, both SIMULA and GPSS are described as process interaction languages. GPSS is narrowly focused on the interaction of temporary objects with permanent objects. Did SIMULA evolve from the same view? How well can SIMULA treat interaction among permanent objects?