Renamed into Invisibility

The Intellectual Genealogy of Simula

Intellectual Genealogies Project

August 2026 — draft v0.1

Concept by Carlos Baquero · Execution by Claude agents

Seed paper: Ole-Johan Dahl and Kristen Nygaard. "SIMULA — an ALGOL-Based Simulation Language." *Communications of the ACM* 9(9), Sept. 1966, pp. 671–678 — with the 1967 "Class and Subclass Declarations" and the Simula 67 Common Base as seed cluster.

An intellectual genealogy follows ideas through history, not just papers through a citation network.

Abstract. In September 1966, Communications of the ACM published an eight-page paper by Ole-Johan Dahl and Kristen Nygaard, edited by D. E. Knuth, describing SIMULA, a "simulation language" extending ALGOL 60. Under the simulation vocabulary, the paper carries the semantics of what would later be called object-oriented programming. A companion paper of 1967 renamed everything in it. Sixty years later the concepts run in essentially every mainstream language, uncredited, and the citation record of the source, counted by decade, reads 68, 171, 209, 203, 278, 142, 14: a peak in the 2000s, then collapse — the first declining curve in this series. Across the eleven intermediary works we measured, between 87.5% and 99.3% of their citers never reach back to the seed. This genealogy traces how that happened. The short version is that the ideas conquered the world by shedding every name that could be cited. The authors renamed them first, their successors renamed them again, and total victory ended citation.

1 The Original Intervention

There is a line of code in the September 1966 issue of Communications of the ACM that hires a secretary: "Pat := new secretary (true, 10);" (Dahl and Nygaard 1966). A little further on, the paper explains what happens when the program drops the last reference to Pat: "she is deleted." (Underneath sits reference counting with garbage collection, with a reference to Weizenbaum. The 1960s were direct about these things.) To any programmer today the line is unremarkable — a constructor call, a fresh instance, a reference that keeps the instance alive. In 1966 there were no words for what it was doing. The paper never uses class, object, inheritance, subclass, or virtual as technical terms. None of them existed yet. Pat is created, referenced, shared, and finally collected: a complete object lifecycle, described with the pronouns of a personnel department.

The paper is an eight-page description of SIMULA by Ole-Johan Dahl and Kristen Nygaard, edited for CACM by D. E. Knuth. Its declared audience is not programmers in general but one profession: the language was "designed to provide a systems analyst with unified concepts which facilitate the concise description of discrete event systems." It is as much a notation for thinking as a tool for computing: "System descriptions should be easy to read and print and hence useful for communication." The authors are also explicit about what they consider the heart of the contribution: "The most important new concept is that of quasi-parallel processing." Note what is being claimed and what is not. The claimed novelty is a control structure. The material that would conquer the world is treated as support.

Read with modern eyes, the supporting material translates itself. "The concept of an 'activity,' which is a class of processes described by the same declaration, is distinguished from the concept of a 'process,' which is one dynamic instance of an activity declaration." An activity is a class; a process is an instance. A process "is a data carrier and it will execute actions", and further, "a process is thus a referenceable data structure" — data together with the actions that operate on it, held through references. The fields are called attributes, which is still the word we use. Access to another process's attributes goes through inspect and connection blocks, a clumsiness the authors already knew how to remove: in "the next version of the language, now being implemented", remote access "will be written '(element expression).(identifier)' ". Dot notation, announced a version early. The paper's second worked example is an epidemic, with sick persons as processes infecting one another. (The reader of the 2020s needs no help seeing the point of that example.)

The paper is also honest about its ancestors — more honest, as we will see, than its descendants would be about it. It is a true extension of ALGOL 60 (Naur 1963). It points to the record proposal of Hoare and Wirth: "It is worth noticing the similarity" (Hoare and Wirth 1966). It credits list processing: "Many of the ideas presented in this section were inspired by the SLIP system" (Weizenbaum 1963). Conway's coroutines are acknowledged (Conway 1963). So is SIMSCRIPT, through Bernard Hausner, "whose experience with SIMSCRIPT had no little impact at an early stage" (Markowitz et al. 1963). Even the practical scaffolding is on record: a UNIVAC contract, the Case ALGOL compiler. Every borrowed part is labelled with its origin. Keep that in mind; the rest of this report is about how the favor was not returned.

Then, within months, the authors renamed their own invention. Dahl's testimony, given in 2001 (Dahl 2001): "At the Vilard-de-Lans Summer School Tony Hoare had put forth a proposal for 'record handling' with record classes and subclasses ... Attribute accessing was by dot notation ... We chose the terms 'class' and 'objects' of classes for our new Simula." The names came from Hoare's records. The one mechanism with no ancestor was their own: "The solution came with the idea of class prefixing: using C as a prefix to another class, the latter would be taken to be a subclass of C inheriting all properties of C." Dahl gives the date: "The breakthrough happened in January of 1967." And he pushes the underlying concept back before both papers: "the idea of data objects with associated operators was under way already in 1965." The concepts, by the inventor's own account, predate both sets of names.

The renaming carried a bibliographic price that is still being paid. The 1967 paper, "Class and Subclass Declarations" (Dahl and Nygaard 1968) — the paper that gave programming the words class, object, and subclass —

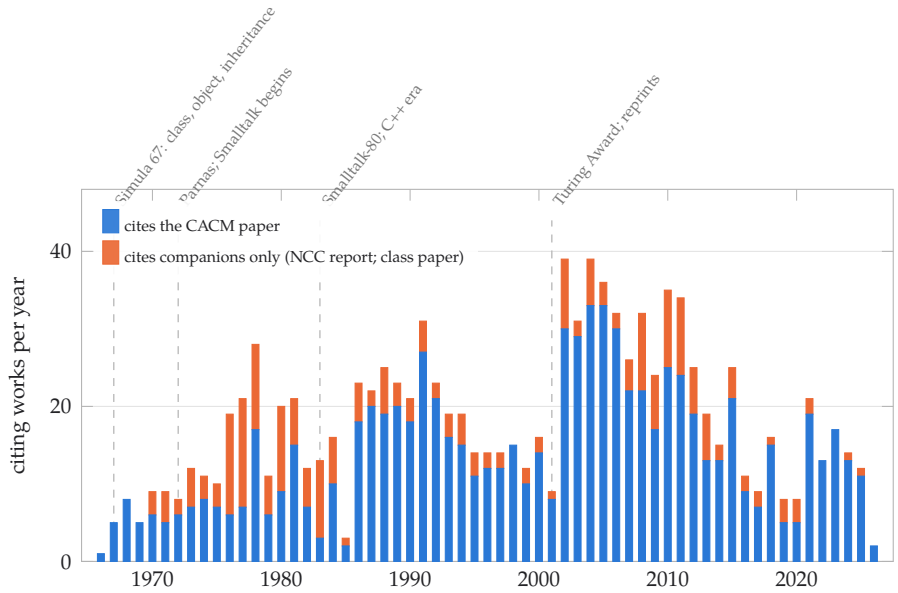

Figure 1: Citing works per year for the merged Simula cluster (1,085 works, six OpenAlex records), 1966–2026. Blue: works citing the CACM 1966 paper; orange: works citing only the companions (the NCC Common Base report and the 1967 class paper's reprint records). Markers are intellectual events from the genealogy's timeline. 2026 is a partial year.

exists in OpenAlex only through its 2001 and 2002 reprint records, with 50 citations between them. The 1966 CACM paper has 862. The Simula 67 Common Base report (Dahl et al. 1968) has 172. To build the corpus for this genealogy we had to merge six bibliographic records of what is intellectually a single contribution, arriving at 1,085 unique citing works (Figure 1). Bibliometric databases count records, not contributions, and splitting one contribution across six records is already a mild form of forgetting. The seed was fragmented before the world even began to forget it.

2 The First Descendants

Who picked the paper up? The canon did. The earliest and most-cited citers of the seed read like a syllabus in programming languages: Parnas's 1972 information-hiding paper (Parnas 1972), which cites the seed while 98.7% of its own 4,701 citers do not; CSP; Liskov and Zilles's 1974 paper on abstract data types (Liskov and Zilles 1974), which cites the Simula 67 Common Base rather than the CACM paper; KRL in 1977 (Bobrow and Winograd 1977); ThingLab in 1981 (Borning 1981); Stefik and Bobrow's 1986 survey of object-oriented programming (Stefik and Bobrow 1986),

which cites the seed; the 1987 paper on computational reflection (Maes 1987); Cardelli in 1988 (Cardelli 1988). The ideas were received at the top of the field, immediately, and by people whose own work would go on to eclipse the source. Notice also, in Liskov and Zilles, the fragmentation of Section 1 already at work: the descendants could not even agree on which record of the seed to cite.

What did the citers think they were citing? We collected 948 citation entries; 318 of them carry citation sentences, around 3,700 sentences in all, with 50 flagged as influential. Of the classified intents, 213 are background, 71 methodology, and exactly 1 result. This is a sparse base for a paper of 1966 vintage — context extraction reaches thinly into old literature, and we flag the limitation — but it is enough to watch the reframing happen in the data. Consider what the intent distribution says: among hundreds of citers, exactly one used the paper for a result. The seed is not a quarry that anyone still mines. It is a place to point at. Background is the citation class of monuments.

The pointing changed direction over time (Figure 2). Before 1980, 0% of the context-bearing citers describe the paper in object-oriented terms, for the good reason that the term was not yet in circulation. In the 1980s, 35% do; in the 1990s, 44.2%; in the 2000s, 29.4%; since 2010, 27.0%. The simulation framing never leaves: 35.0%, 55.0%, 32.7%, 42.1%, and 32.0% across the same decades. The paper kept a double identity to the end — a simulation tool to one community, an origin story to another — and only 12.9% of citers frame it as an origin, a first, or a precursor at all. A paper with two identities has two literatures, and each literature could assume the other was keeping the flame. Most telling is what else appears inside the seed's own citation contexts: Smalltalk, 45 mentions; Java, 22; C++, 20; SIMSCRIPT, 13. People citing the 1966 paper are usually talking about other languages. The ancestor is cited as a footnote to its descendants.

And the counts fall. Sixty-eight citations in the first decade, then 171, 209, 203, a peak of 278 in the 2000s, then 142, then 14. Every earlier seed in this series shows a curve that rises or holds. Simula's is the first that declines. We will argue below that the decline and the ubiquity are the same phenomenon seen from two sides.

3 The Branching

We map nine branches, which is a lot of branches for one eight-page paper. That is the point. One caution before the map: branch membership here is established at survey depth — from surveys, titles, and selective full-text reading, not from a full-text verdict on every member. It is a sketch of the river system, not a property registry (Table 1).

The object-oriented language line — Simula 67 to Smalltalk to C++ to Java,

	emerged	relation	eclipse of the seed*
SOURCES OF THE SEED			
ALGOL 60 — a true extension	1960	documented	
SIMSCRIPT — Hausner's "no little impact"; ancestor-sibling	1963	documented	
SLIP lists; Conway's coroutines	1963	documented	
Hoare's record classes — Villard-de-Lans	1966	documented	
BRANCHES OF THE LINEAGE			
the simulation line — the stated purpose; DES onward	1966→	documented	
coroutines — threads, async, generators	1966→	documented	
the OO language line — Smalltalk, C++, Java. . .	1972→	documented	92–98%
abstraction & software engineering — Parnas, ADTs	1972→	documented	93–99%
Scandinavian school — Simula Begin, BETA; keeps the memory	1973→	documented	95%
AI / knowledge representation — frames, KRL	1977→	documented	94%
the authors' own 1978 history (HOPL)	1978	documented	88–91%
type theory — detached at Cardelli–Wegner	1985→	indirect	96%
learning objects, education — documented gateway	2000→	indirect	99.3%
DORMANT			
the Simula language itself — faded by the 1990s	—	dormant	
"quasi-parallel" as a term	—	dormant	
OUTSIDE THE LINEAGE			
GPSS — rival simulation tradition	1961	no descent claimed	

Table 1: The genealogy at a glance — a view of the evidence ledger. Relations use the ledger's own vocabulary, not citation links. *Eclipse of the seed: the share of the branch canon's citers that does not cite the seed cluster (bar drawn at the range's upper bound). Every measured branch sits between 88% and 99.3% — the table is the thesis: the lines that carried the ideas no longer cite the source. The Simula language itself appears under dormant — the rare case of the artifact dying on a victorious tree.

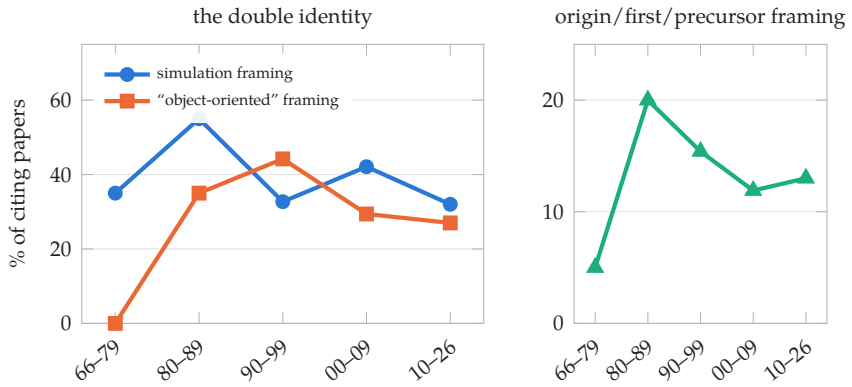

Figure 2: Framings of the seed in its citation contexts, by era (share of context-bearing citing papers; the context sample is sparse — 318 citers — and percentages are portraits, not measurements). Left: the double identity — the simulation framing never leaves; the "object-oriented" framing is 0% before 1980, since the term did not yet exist, and peaks in the 1990s. Right: explicit origin/first/precursor framing — the monument function.

C#, and onward. The main line, and the strangest one: it is carried by languages, not by citations. Each language hands the concepts to the next in syntax and semantics, and the papers stop mattering. A programmer can spend a career inside this line and never encounter a reference to its origin.

Abstraction and software engineering — Parnas's information hiding, abstract data types, encapsulation, and the methodology industry built on top of them.

Type theory — detached early. Cardelli and Wegner's 1985 survey (Cardelli and Wegner 1985) has 61 references and not one belongs to the Simula cluster; 96.2% of its 1,680 citers skip the cluster as well. By 1985 the theory of objects no longer needed the history of objects.

AI and knowledge representation — frames, KRL, and the Stefik–Bobrow bridge back into object-oriented programming.

The Scandinavian school — the Simula Begin textbook (Birtwistle et al. 1979), BETA, and Nygaard's participatory-design side. Of all the branches, this one keeps the memory best.

The simulation line — the stated purpose of 1966: DEMOS, process-oriented discrete-event simulation, and onward into distributed discrete-event simulation. This branch shares a node with Report No. 1 of this series; we return to it in Section 8.

Coroutines — the other life of quasi-parallelism: threads, async, generators. The concept the authors called their most important survives here, unnamed as such.

Learning objects — an unexpected crossing into education technology,

documented in the next section.

The dormant line — the language itself, dead by the 1990s; "quasi-parallel" as a term; the SQS machinery. Branches die even on a victorious tree.

4 Migration and Transformation

The decisive moves in this genealogy are hops between communities, and the same pattern repeats at each hop, so we state it once: the people who moved the ideas said in so many words where they came from, and the communities that received the ideas did not repeat the acknowledgement. Testimony gets stronger while citation gets thinner.

The Smalltalk hop. Kay's history of Smalltalk for HOPL-II (Kay 1993) cites the seed. Of Kay's own 272 citers, 91.9% do not follow the link. The Smalltalk-80 book (Goldberg and Robson 1983), the artifact that actually carried the concepts to a generation of programmers, has 3,909 citers, of whom 95.3% never cite Simula — and its OpenAlex record carries no reference list at all, so we cannot even measure what the book itself acknowledged. That gap is documented in our ledger rather than papered over; where the bibliographic record fails, testimony has to substitute.

The C++ hop. Here the testimony could not be plainer. Stroustrup (Stroustrup n.d.): "I wanted to write efficient systems programs in the styles encouraged by Simula67". And his definition of the paradigm itself: object-oriented programming is "a style of programming originating with Simula (more than 40 years ago!) relying of [sic] encapsulation, inheritance, and polymorphism". The designer of the industry's workhorse names Simula as the origin, in writing, with an exclamation mark. His Design and Evolution book (Stroustrup 1994) has 392 citers; 98.2% of them do not cite the seed. The origin claim is transmitted; the citation is not. A reader can learn from Stroustrup that everything began with Simula and still never look up what Simula was.

The learning-objects hop. In 2000, Wiley's text on learning objects (Wiley 2000) cites the 1966 paper directly and builds the new field's central metaphor on the object of object-oriented programming. The text gathered 1,890 citers, with a median year of 2011; 99.3% of them skip the seed. We verified that this is real cross-disciplinary descent — the education-technology object really does descend from the programming object — and not a collision of words. A 1966 simulation paper has grandchildren in pedagogy journals, and almost none of them know it. It is the deepest eclipse in our table.

The last transformation is the quietest: absorption into specification culture. Language specifications define class, object, and inheritance the way dictionaries define words — without attribution, because the genre

keeps no bibliography for the obvious. No dictionary credits the first user of a word, and no language standard credits the first language to have classes. We checked examples, not the full population of specifications, so this claim stays at the strength of a sampled observation. But it names the end state: vocabulary so thoroughly won that defining it requires no history. The 1966 paper wanted system descriptions to be "useful for communication"; the wish was granted, minus the return address.

Two boundary decisions complete the map. GPSS is a rival simulation tradition, not a descendant, and stays off the tree. SIMSCRIPT is a documented ancestor-sibling — Hausner carried its experience into the Simula effort — and stays off the tree from the other side. A genealogy is defined as much by who is excluded as by who is included.

5 The Lost Citation Chain

Our method reserves a full stage for transmission without citation, because citations are evidence of transmission, not its definition. In most genealogies that stage recovers a few broken links. In this one it recovers almost the entire picture. A genealogy of Simula that only walked the citation graph would conclude, absurdly, that the paper had modest influence and lost it.

We took the eleven measured intermediaries — works that demonstrably carried Simula's concepts onward — and asked what fraction of each one's citers ever cite the seed cluster. From deepest eclipse to shallowest: Wiley 99.3%, Parnas 98.7%, Stroustrup's Design and Evolution 98.2%, Cardelli–Wegner 96.2%, the Smalltalk-80 book 95.3%, the Simula Begin textbook 95.3%, Stefik–Bobrow 93.6%, Liskov–Zilles 93.3%, Kay's history 91.9%, and the authors' own 1978 HOPL history of SIMULA (Dahl and Nygaard 1978) at 87.5% and 90.9% across its two bibliographic records. Even the paper in which Dahl and Nygaard tell their own story loses roughly nine of every ten onward readers. The median citation years of these intermediaries run from 1988 to 2012: the chain did not break recently. It broke decades ago and stayed broken.

Numbers this uniform point to structure, not accident. When one intermediary eclipses its source, the cause can be local — a feud, a paywall, an unlucky name. When all eleven do, at 87.5% or worse, the cause is the transmission mechanism itself. Programming concepts travel in artifacts: languages, books, specifications. Artifacts do not carry bibliographies forward. Programmers will recognize the mechanism from their own tools: a compiled binary keeps none of the comments of its source. What survives a hop is what the artifact needs in order to function, and no language needs a reference list to run. Every hop resets the bibliography to empty, and there were many hops. Citation analysis can only see the

hops that happen on paper.

The series gives the comparison. The intermediaries of Report No. 0 eclipsed their seed at 22–54%. Those of Report No. 1 ranged from 24% to 99.8%. Simula's run from 87% to 99%. The eclipse deepens with the age of the seed, and Simula, the oldest seed so far, shows the deepest. The floor of Simula's range sits above the ceiling of Report No. 0's.

6 Convergence, Fusion, and Reinterpretation

Three renamings, in sequence, and each one is a fusion with someone else's vocabulary.

The first happened at the source. Activity and process became class and object; the names were imported from Hoare's record handling, and the dot notation came along with them; only inheritance — class prefixing — was named at home. The authors did this to their own paper while it was still fresh in print. It was the right technical decision, and it orphaned the 1966 vocabulary on the spot. No one would ever again search the literature for "activities".

The second renaming happened in the Smalltalk circle, which coined "object-oriented" and thereby named the paradigm — a name Simula received only retroactively. The retro-reading numbers of Section 2 measure this reinterpretation directly: no citer before 1980 could describe the paper in words that did not yet exist, and from the 1980s onward between a quarter and nearly half of them describe it in exactly those words. A 1966 simulation paper was re-filed, in place, under a category invented a decade later. The paper did not move; the shelf labels around it did. Meanwhile the simulation community kept citing the same paper for its stated purpose, in every decade, at comparable rates. The double identity was never resolved; the two readerships simply stopped meeting.

The third renaming was the industry's, and it dissolved the name entirely. Class and object became specification vocabulary, defined in every language standard and attributed in none. Each renaming was also a fusion: with Hoare's records, with Smalltalk's paradigm, with the anonymous prose of standards. At every step the ideas gained reach and lost address. There is no fourth renaming, because there is nothing left to rename.

7 Decline, Survival, or Canonization

The honors arrived exactly as the citations left. In 2001 Dahl and Nygaard received the Turing Award, "for ideas fundamental to the emergence of object oriented programming, through their design of the programming

languages Simula I and Simula 67" (*ACM A.M. Turing Award 2001: Ole-Johan Dahl and Kristen Nygaard* n.d.). In 2002 came the IEEE von Neumann Medal. Both authors died in 2002. ACM SIGPLAN reprinted the 1967 paper in 2001 and 2002, and those reprint records are, as we saw, the paper's entire bibliographic presence today. The award citation itself performs the retro-reading of Section 2: it honors the languages for what they became the origin of, not for what their authors said they were. The peak citation decade is also the award decade. Whether the prize refreshed the field's memory or only marked it, the sequence in the counts is 278, 142, 14. The monument was completed just as the visitors stopped coming.

We have met this monument function in both earlier reports: canonization converts a living reference into a plaque. Simula adds the purest case so far of the artifact parting ways with the idea. The language was dead by the 1990s. The semantics became the industry default, running in essentially every mainstream language. Simula stands to today's languages roughly as Latin stands to the Romance languages: no one speaks it, and everybody speaks it. And unlike the seeds of Reports No. 0 and No. 1, this one had no founding school propelling its citations forward. The Scandinavian school kept the memory — Simula Begin, BETA, the participatory-design tradition — but memory and citation are different currencies, and only one of them shows up in the counts. Simula's survival does not pass through its literature at all. It passes through every `new` expression executed today.

8 The Intellectual Legacy

Genealogists of human families know the pattern well: the hardest ancestor to trace is the one who changed names at the border. Simula's ideas crossed three borders and changed names at every one. The authors renamed activity and process into class and object in 1967, taking Hoare's terms. The Smalltalk circle renamed the whole enterprise "object-oriented", a name that stuck to the successors first and to the source only in hindsight. The industry renamed once more, into the anonymous vocabulary of specifications, where class and object are defined the way integer is defined — as if they had always existed.

The measurements agree with the story. This is the deepest eclipse in the series: 87–99% across all eleven intermediaries, against 22–54% for Report No. 0 and 24–99.8% for Report No. 1; the depth grows with the seed's age. And it is the series' first declining citation curve, falling while the concepts approached total ubiquity. These two facts are one fact. A citation needs a name to attach to, and the ideas won by shedding every name that could be cited. Total victory ended citation.

The genealogy also connects outward. The simulation branch —

the stated purpose of 1966, easy to forget under the object-oriented retrospective — continues into distributed discrete-event simulation, which is where the subject of Report No. 1, Jefferson's Virtual Time, lives. Two genealogies in this series meet at that node. And the recurring structures recur: the monument function, with awards and reprints standing where readers used to be, and the two-tier memory, with a small school that remembers everything and a world that remembers one line, appear in all three reports. They are starting to look less like accidents of particular papers and more like the normal way this field remembers. If the pattern holds, eclipse is not a failure mode of scientific memory. It is what complete adoption looks like in the record.

The 1966 paper set out to give systems analysts a concise way to describe the world as interacting processes, and demonstrated the idea on a secretary and an epidemic. The description language succeeded beyond any reasonable ambition: it became the way software describes everything. The price was the description of its own descent. Ideas that win completely stop needing their names, and then remembering them becomes someone's job. That is the job of genealogy.

Provenance

This report is built on 13 lineage claims recorded in the genealogy's evidence ledger, all of which passed verification: 5 at documented strength, 8 at strong. Four hedges qualify the narrative. Branch membership in Section 3 is established at survey depth, not by full-text reads of every member. The citation-context sample behind Section 2 is sparse for a paper of 1966 vintage — 318 context-bearing citers. The specification-culture claim of Section 4 is based on sampled examples, not an exhaustive sweep of language standards. And the Smalltalk-80 and Design and Evolution book records carry no reference lists in OpenAlex; the gap is documented in the ledger, and author testimony substitutes for bibliography at those two links. The full research notes, data pulls, and evidence ledger accompany this document as research provenance. Margin ledger references can be rendered by compiling with the `provenance` option.

Sources

ACM A.M. Turing Award 2001: Ole-Johan Dahl and Kristen Nygaard (n.d.). Citation: "for ideas fundamental to the emergence of object oriented programming, through their design of the programming languages Simula I and Simula 67." Accessed 2026-08-20. URL: https://awards.acm.org/award-recipients/dahl_6917600.

Birtwistle, Graham M. et al. (1979). *Simula Begin*. Studentlitteratur.

Bobrow, Daniel G. and Terry Winograd (1977). "An Overview of KRL, a Knowledge Representation Language". In: *Cognitive Science* 1.1, pp. 3–46.

Borning, Alan (1981). "The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory". In: *ACM TOPLAS* 3.4, pp. 353–387. DOI: 10.1145/357146.357147.

Cardelli, Luca (1988). "A semantics of multiple inheritance". In: *Information and Computation* 76.2–3, pp. 138–164.

Cardelli, Luca and Peter Wegner (1985). "On understanding types, data abstraction, and polymorphism". In: *ACM Computing Surveys* 17.4, pp. 471–523. DOI: 10.1145/6041.6042.

Conway, Melvin E. (1963). "Design of a separable transition-diagram compiler". In: *Communications of the ACM* 6.7, pp. 396–408.

Dahl, Ole-Johan (2001). *The Birth of Object Orientation: the Simula Languages*. Manuscript, June 2001; Univ. of Oslo Dahl bibliography.

Dahl, Ole-Johan, Bjørn Myhrhaug, and Kristen Nygaard (1968). *SIMULA 67 Common Base Language*. Tech. rep. S-22. Norwegian Computing Center, Oslo.

Dahl, Ole-Johan and Kristen Nygaard (1966). "SIMULA — an ALGOL-Based Simulation Language". In: *Communications of the ACM* 9.9, pp. 671–678. DOI: 10.1145/365813.365819.

— (1968). "Class and Subclass Declarations". In: *Simulation Programming Languages (IFIP Working Conf., Oslo 1967)*. Reprinted 2001/2002; the reprint records carry the paper's bibliographic presence. North-Holland.

— (1978). "The development of the SIMULA languages". In: *History of Programming Languages (HOPL)*. ACM, pp. 439–480. DOI: 10.1145/800025.1198392.

Goldberg, Adele and David Robson (1983). *Smalltalk-80: The Language and its Implementation*. Addison-Wesley.

Hoare, C. A. R. and Niklaus Wirth (1966). "A Contribution to the development of ALGOL". In: *Communications of the ACM* 9.6, pp. 413–432.

Kay, Alan C. (1993). "The early history of Smalltalk". In: *History of Programming Languages—II (HOPL-II)*. ACM, pp. 511–598. DOI: 10.1145/154766.155364.

Liskov, Barbara and Stephen Zilles (1974). "Programming with abstract data types". In: *ACM SIGPLAN Symp. on Very High Level Languages*, pp. 50–59. DOI: 10.1145/800233.807045.

Maes, Pattie (1987). "Concepts and experiments in computational reflection". In: *OOPSLA*, pp. 147–155.

Markowitz, Harry M., Bernard Hausner, and Herbert W. Karr (1963). *SIMSCRIPT: A Simulation Programming Language*. Prentice-Hall.

Naur, Peter (ed.) (1963). "Revised report on the algorithmic language ALGOL 60". In: *Communications of the ACM* 6.1, pp. 1–17.

Parnas, David L. (1972). "On the criteria to be used in decomposing systems into modules". In: *Communications of the ACM* 15.12, pp. 1053–1058. DOI: 10.1145/361598.361623.

Stefik, Mark and Daniel G. Bobrow (1986). "Object-oriented programming: Themes and variations". In: *AI Magazine* 6.4, pp. 40–62.

Stroustrup, Bjarne (n.d.). *Bjarne Stroustrup's FAQ*. Accessed 2026-08-20. URL: https://www.stroustrup.com/bs_faq.html.

— (1994). *The Design and Evolution of C++*. Addison-Wesley.

Weizenbaum, Joseph (1963). "Symmetric list processor". In: *Communications of the ACM* 6.9, pp. 524–544.

Wiley, David A. (2000). "Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy". In: *The Instructional Use of Learning Objects*. AIT/AECT.