References

Aaron, S. (2016). Sonic pi–performance in education, technology and art. *International Journal of Performance Arts and Digital Media*, 12(2):171–178.

Abadi, M. and Cardelli, L. (1996). *A Theory of Objects*. Springer Science & Business Media.

Abadi, M., Gardner, P., Gordon, A., Mardare, R., and (eds.) (2014). Essays for the Luca Cardelli Fest. Technical Report MSR-TR-2014-104.

Abbate, J. (2012). *Recoding Gender: Women's Changing Participation in Computing*. MIT Press.

Abbott, R. J. (1983). Program design by informal English descriptions. *Communications of the ACM*, 26(11):882–894.

Adam, A. (2005). Hacking into hacking: Gender and the hacker phenomenon. In Adam, A. (ed.), *Gender, Ethics and Information Technology*, pages 128–146. Springer.

Agaram, K. (2020). Bicycles for the mind have to be see-through. In *Programming'20: 4th International Conference on the Art, Science, and Engineering of Programming, Porto, Portugal, March 23–26, 2020,* pages 173–186.

Allen, C. D., Chapman, D. N., and Jones, C. B. (1972). A formal definition of Algol 60. Technical Report 12.105, IBM Laboratory Hursley.

Allman, E. (2004). A conversation with James Gosling: James Gosling talks about virtual machines, security, and of course, Java. *Queue*, 2(5):24–33.

Altenkirch, T., Gaspes, V., Nordström, B., and von Sydow, B. (1994). A user's guide to ALF. Technical Report, Chalmers University of Technology, Sweden.

Altenkirch, T., McBride, C., and McKinna, J. (2005). Why dependent types matter. *Manuscript*, 235, available online, www.e-pig.org/downloads/ydtm.pdf.

Amin, N. and Tate, R. (2016). Java and Scala's type systems are unsound: The existential crisis of null pointers. *ACM Sigplan Notices*, 51(10):838–848.

ANSI/IEEE (1983). IEEE standard for software test documentation. Technical Report 829-1983, Institute of Electrical and Electronics Engineers.

ANSI/IEEE (1986). IEEE standard for software verification and validation plans. Technical Report 1012-1986, Institute of Electrical and Electronics Engineers.

ANSI/IEEE (1987). IEEE standard for software unit testing. Technical Report 1008-1987, Institute of Electrical and Electronics Engineers.

Ardis, M., Basili, V., Gerhart, S., Good, D., Gries, D., Kemmerer, R., Leveson, N., Musser, D., Neumann, P., and von Henke, F. (1989). ACM forum. *Communications of the ACM*, 32(3):287–ff.

Armstrong, J. (2007). A history of Erlang. In *Proceedings of the Third ACM SIGPLAN Conference on History of Programming Languages*, HOPL III, 1–26. New York. ACM Publications.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Ashenhurst, R. L. (1972). Curriculum recommendations for graduate professional programs in information systems. *Communications of the ACM*, 15(5):363–398.

Astarte, T. K. (2017). Towards an interconnected history of semantics. In *Fourth International Conference on the History and Philosophy of Computing*, 1–5, Brno, Czech Republic.

Astarte, T. K. (2019). Formalising Meaning: A History of Programming Language Semantics. PhD thesis, Newcastle University.

Astarte, T. K. and Jones, C. B. (2018). Formal semantics of Algol 60: Four descriptions in their historical context. In De Mol, L. and Primiero, G., (eds.), *Reflections on Programming Systems: Historical and Philosophical Aspects*, pages 71–141. Springer.

Atchison, W. F. (1985). The development of computer science education. *Advances in Computers*, 24:319–377.

Atkinson, B. (2016). Bill Atkinson, interviewed by Leo Laporte. https://twit.tv/shows/triangulation/episodes/247.

Augustsson, L. (1998). Cayenne: A language with dependent types. In Doaitse Swierstra, S., Oliveira, J. N. and Henriques, P. R. (eds.), *International School on Advanced Functional Programming*, pages 240–267. Springer.

Backus, J. W. (1980). Programming in America in the 1950s: Some personal impressions. In Metropolis, N., Howlett, J. and Rota G.-C. (eds.), *A History of Computing in the Twentieth Century*, pages 125–135. Elsevier.

Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Naur, P., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., et al. (1960). Report on the algorithmic language Algol 60. *Numerische Mathematik*, 2(1):106–136.

Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., and Woodger, M. (1963). Revised report on the algorithm language Algol 60. *Communications of the ACM*, 6(1):1–17.

Backus, J. W., Beeber, R. J., Best, S., Goldberg, R., Herrick, H. L., Hughes, R. A., Mitchell, L. B., Nelson, R. A., Nutt, R., Sayre, D., Sheridan, P. B., Stern, H., and Ziller, L. (1956). *Fortran: Automatic Coding System for the IBM 704 EDPM*, IBM Corporation.

Backus, J. W., Herrick, H., and Ziller, I. (1954). *Specifications for the IBM Mathematical Formula Translating System*, Fortran. preliminary report, programming research group. Applied Science Division, IBM Corporation.

Baker, T. F. and Mills, H. D. (1973). Chief programmer teams. *Datamation*, 19(12):58–61.

Bank, D. (1995). The Java Saga. *Wired*. www.wired.com/1995/12/java-saga/.

Barber, R. J. (1975). The advanced research projects agency, technical report published by ARPA, number AD-A154 363. URL: https://apps.dtic.mil/sti/citations/ADA154363. 1958–1974.

Bardini, T. (2000). *Bootstrapping: Douglas Engelbart, Coevolution, and the Origins of Personal Computing*. Stanford University Press.

Bartik, J. J. (2013). *Pioneer Programmer: Jean Jennings Bartik and the Computer That Changed the World*. Truman State University Press.

Bawden, A., Greenblatt, R., Holloway, J., Knight, T., Moon, D., and Weinreb, D. (1974). Lisp machine progress report. Technical Report AIM-444, MIT.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Beck, K. (2003). *Test-Driven Development: By Example*. Addison-Wesley Professional.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development. URL: https://agilemanifesto.org/.

Beck, K. and Cunningham, W. (1989). A laboratory for teaching object oriented thinking. In *Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications*, OOPSLA '89, pages 1–6, New York. ACM.

Beeler, M., Gosper, R. W., and Schroeppel, R. (1972). HAKMEM. Technical Report, AI Memo 239, MIT.

Benington, H. D. (1983). Production of large computer programs. *Annals of the History of Computing*, 5(4):350–361.

Bissell, C. (2007). Historical perspectives: The moniac a hydromechanical analog computer of the 1950s. *IEEE Control Systems Magazine*, 27(1):69–74.

Black, A. P. (2013). Object-oriented programming: Some history, and challenges for the next fifty years. *Information and Computation*, 231:3–20.

Blackwell, A. F. and Collins, N. (2005). The programming language as a musical instrument. In *Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group, PPIG 2005, Brighton, UK, June 29–July 1, 2005*, page 11. Psychology of Programming Interest Group.

Bobrow, D. G., Darley, D. L., Deutsch, L. P., Murphy, D. L., and Teitelman, W. (1967). The BBN 940 LISP system. Technical Report, Bolt Beranek and Newman, Inc.

Bobrow, D. G. and Winograd, T. (1976). An overview of KRL, a knowledge representation language. Technical Report AIM-293, Stanford Aritifical Intelligence Laboratory.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5):61–72.

Böhm, C. and Jacopini, G. (1966). Flow diagrams, Turing machines and languages with only two formation rules. *Communications of the ACM*, 9(5):366–371.

Booch, G. (1983). *Software Engineering with Ada*. Benjamin Cummings.

Booch, G. (1990). *Object-Oriented Analysis and Design with Applications*. Benjamin Cummings.

Bornat, R. (2009). Peter Landin: A computer scientist who inspired a generation. *Higher-Order and Symbolic Computation*, 22(4):295–298.

Borning, A. (1981). The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 3(4):353–387.

Borning, A. H. (1986). Classes versus prototypes in object-oriented languages. In *Proceedings of 1986 ACM Fall Joint Computer Conference*, pages 36–40.

Bosak, R., Clippinger, R. F., Dobbs, C., Goldfinger, R., Jasper, R. B., Keating, W., Kendrick, G., and Sammet, J. E. (1962). An information algebra: Phase 1 report – language structure group of the CODASYL development committee. *Communications of the ACM*, 5(4):190–204.

Boyer, C. (2004). *The 360 Revolution*. IBM. Online at https://archive.org/details/h42_The_360_Revolution_Chuck_Boyer.

Boyer, R. S. and Moore, J. (1983). On why it is impossible to prove that the BDX90 dispatcher implements a time-sharing system. Technical report published by NASA document number: 19840017251. URL: https://ntrs.nasa.gov/citations/19840017251.

Boyer, R. S. and Moore, J. S. (1988). *A Computational Logic Handbook*. Academic Press Professional.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Brady, E. (2017). *Type-Driven Development with Idris*. Manning Publications Company.

Brand, S. (1972). Spacewar: Fanatic life and symbolic death among the computer bums. *Rolling Stone*, (123) December 1972, 52–57.

Brand, S. and Crandall, R. (1988). The media lab: Inventing the future at MIT. *Computers in Physics*, 2(1):91–92.

Brooks, F. (1987). No silver bullet: Essence and accidents of software engineering. *Computer*, 20(4):10–19.

Brooks Jr, F. P. (1975). *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley.w

Brown, H. (1886). Receptacle for storing and preserving papers. US Patent 352,036.

Brümmer, L. (2021). *The Hub: Pioneers of Network Music*. Kehrer Verlag.

Budd, T. A., Lipton, R. J., DeMillo, R. A., and Sayward, F. G. (1978). The design of a prototype mutation system for program testing. In Ghosh, S. P. and Liu, L. Y., (eds.), *American Federation of Information Processing Societies: 1978 National Computer Conference, June 5–8, 1978, Anaheim, CA, USA*, volume 47 of AFIPS Conference Proceedings, pages 623–629. AFIPS Press.

Burstall, R. (2000). Christopher Strachey: Understanding programming languages. *Higher-Order and Symbolic Computation*, 13(1–2):51–55.

Bush, V. (1945). As we may think. *The Atlantic Monthly*, 176(1):101–108.

Buxton, J., Randell, B., and Committee, N. S. (1970). *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee*. NATO Science Committee; available from Scientific Affairs Division, NATO.

Byous, J. (1998). Happy 3rd birthday! Sun Microsystems, archived at http://web.archive.org/web/19990223195009/http://java.sun.com/features/1998/05/birthday.html (Retrieved on 19 Jan 2024).

Campbell-Kelly, M. (1980). Programming the EDSAC: Early programming activity at the University of Cambridge. *Annals of the History of Computing*, 2(1):7–36.

Campbell-Kelly, M. (1992). The airy tape: An early chapter in the history of debugging. *IEEE Annals of the History of Computing*, 14:16–26.

Campbell-Kelly, M. (2004). *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. MIT Press.

Campbell-Kelly, M. (2011). From theory to practice: The invention of programming, 1947–51. In Jones, C. B. and Lloyd, J. L. (eds.), *Dependable and Historic Computing*, pages 23–37. Springer.

Campbell-Kelly, M. and Aspray, W. (1996). *Computer: A History of the Information Machine*. Basic Books.

Cardelli, L. (1984). A semantics of multiple inheritance. In Kahn, G., MacQueen, D. B. and Plotkin, G. (eds.), *Proc. of the International Symposium on Semantics of Data Types*, pages 51–67, Berlin, Heidelberg. Springer-Verlag.

Cardelli, L. and MacQueen, D. (1983). Polymorphism: The ML/LCF/Hope newsletter. *Polymorphism*, 1(1), http://lucacardelli.name/Papers/Polymorphism%20Vol%20I,%20No%201.pdf.

Cardelli, L. and Wegner, P. (1985). On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys*, 17(4):471–523.

Cardone, F. and Hindley, J. R. (2006). History of lambda-calculus and combinatory logic. *Handbook of the History of Logic*, 5:723–817.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Cavanaugh, R. (2018). Type-checking unsoundness: Standardize treatment of such issues among typescript team/community? [Online: https://github.com/Microsoft/TypeScript/issues/9825\#issuecomment-234115900. Accessed September 10, 2018].

Ceruzzi, P. (1988). Electronics technology and computer science, 1940–1975: A coevolution. *Annals of the History of Computing*, 10(4):257–275.

Ceruzzi, P. E. (2003). *A History of Modern Computing*. MIT Press.

Chandrasekaran, S. K., Leijen, D., Pretnar, M., and Schrijvers, T. (2018). Algebraic effect handlers go mainstream (dagstuhl seminar 18172). In *Dagstuhl reports*, volume 8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. https://drops.dagstuhl.de/entities/document/10.4230/DagRep.8.4.104.

Chang, H. (2004). *Inventing Temperature: Measurement and Scientific Progress*. Oxford University Press.

Chang, H. (2012). *Is Water H2O? Evidence, Realism and Pluralism*, volume 293. Springer Science & Business Media.

Chang, H. (2017). Who cares about the history of science? *Notes and Records: The Royal Society Journal of the History of Science*, 71(1):91–107.

Cheatham, T. E. (1969). Motivation for extensible languages. *SIGPLAN Notices*, 4(8):45–49.

Chris Brown and John Bischoff (2002). Indigenous to the net: Early network music bands in the San Francisco Bay area. http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html [Accessed December 7, 2021].

Church, A. (1940). A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68.

Cihon, P. (2019). Standards for AI governance: International standards to enable global coordination in AI research & development. Technical Report, Future of Humanity Institute. University of Oxford.

Clarke, D. G., Potter, J. M., and Noble, J. (1998). Ownership types for flexible alias protection. In *Proceedings of the 13th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 48–64.

Cohn, A. (1989). The notion of proof in hardware verification. *Journal of Automated Reasoning*, 5(2):127–139.

Colburn, T. R., Fetzer, J. H., and Rankin, T. L., (eds.). (2012). *Program Verification: Fundamental Issues in Computer Science*, volume 14. Springer Science & Business Media.

Conger, R. A. (1962). Certification of algorithm 58: Matrix inversion. *Communications of the ACM*, 5(6):347.

Cook, S. (1988). Impressions of ecoop'88, *Journal of Object-Oriented Programming*, 1(4):42–43. SIGS Publications Inc., New York.

Cook, W. and Palsberg, J. (1989). A denotational semantics of inheritance and its correctness. In *Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications*, OOPSLA '89, page 433–443, New York. ACM.

Cook, W. R. (2009). On understanding data abstraction, revisited. In *Proceedings of the 24th ACM SIGPLAN Conference on Object-Oriented Programming Systems Languages and Applications*, pages 557–572.

Coquand, C., Takeyama, M., and Synek, D. (2006). An emacs interface for type directed support constructing proofs and programs. In *European Joint Conferences on Theory and Practice of Software, ENTCS*, volume 2.

Coquand, T. (2018). Type theory. In Zalta, E. N., (ed.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2018 edition.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Cox, B. J., Naroff, S., and Hsu, H. (2020). The origins of Objective-c at PPI/Stepstone and its evolution at NeXT. *Proceedings of the ACM on Programming Languages*, 4(HOPL).

Crawford, K. and Paglen, T. (2019). Excavating AI: The politics of images in machine learning training sets. *AI and Society*, 36(4): 1105–1116.

Dahl, O.-J. (1969). Programming languages as tools for the formulation of concepts. In Aubert, K. E. and Ljunggren, W., (eds.), *Proceedings of the 15th* Scandinavian Congress Oslo 1968, pages 18–29, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dahl, O. J., Dijkstra, E. W., and Hoare, C. A. R., (eds.). (1972). *Structured Programming*. Academic Press.

Dahl, O.-J., Myhrhaug, B., and Nygaard, K. (1968). Some features of the SIMULA 67 language. In *Proceedings of the Second Conference on Applications of Simulations*, pages 29–31. Winter Simulation Conference.

Dahl, O.-J. and Nygaard, K. (1966). SIMULA: An Algol-based simulation language. *Communications of the ACM*, 9(9):671–678.

Dahl, O.-J. and Nygaard, K. (2002). *Class and Subclass Declarations*, pages 91–107. Springer.

Damouth, D., Urbach, J., Mitchel, J. G., and Kay, A. (1971). Parc papers for pendery and planning purposes. Technical Report, Xerox PARC.

Dastin, J. (2018). Amazon scraps secret AI recruiting tool that showed bias against women. In Martin K. (ed.), *Ethics of Data and Analytics*, pages 296–299. Auerbach Publications.

DeMillo, R. A., Lipton, R. J., and Perlis, A. J. (1979). Social processes and proofs of theorems and programs. *Communications of the ACM*, 22(5):271–280.

De Mol, L. and Bullynck, M. (2018). Making the history of computing: The history of computing in the history of technology and the history of mathematics. *Revue de Synthèse*, 139(3–4):361–380.

Dechesne, F. and Nederpelt, R. (2012). N.G. de Bruijn (1918–2012) and his road to automath: The earliest proof checker. *The Mathematical Intelligencer*, 34(4):4–11.

DeMillo, R. A., Lipton, R. J., and Perlis, A. J. (1977). Social processes and proofs of theorems and programs. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '77, page 206–214, New York. ACM.

DeMillo, R. A., Lipton, R. J., and Sayward, F. G. (1978). Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41.

Depaz, P. (2023). The Role of Aesthetics in Understanding Source Code. PhD thesis, Université Sorbonne Nouvelle, ED120 - THALIM.

DeRemer, F. and Kron, H. H. (1976). Programming-in-the-large versus programming-in-the-small. *IEEE Transactions on Software Engineering*, SE-2(2):80–86.

Deutsch, L. P. and Lampson, B. W. (1967). An online editor. *Communications of the ACM*, 10(12):793–799.

Deutsch, L. P. (1967). Preliminary guide to the LISP editor. Technical Report Project Genie Document W-21, University of California, Berkeley.

Di Sessa, A. A. (1985). A principled design for an integrated computational environment. *Human-Computer Interaction*, 1(1):1–47.

Dijkstra, E. W. (1968). Letters to the editor: Go to statement considered harmful. *Communications of the ACM*, 11(3):147–148.

Dijkstra, E. W. (1970). Notes on Structured Programming. circulated privately.

Dijkstra, E. W. (1972). The humble programmer. *Communications of the ACM*, 15(10):859–866.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Dijkstra, E. W. (1977). A position paper on software reliability. *ACM SIGSOFT Software Engineering Notes*, 2(5):3–5.

Dijkstra, E. W. (1978). On a political pamphlet from the middle ages. *ACM SIGSOFT Software Engineering Notes*, 3(2):14–16.

Dijkstra, E. W. (1980). America's programming plight. Technical Report EWD 750, The University of Texas at Austin.

Dijkstra, E. W. (1982). How do we tell truths that might hurt? In Dijkstra, E. W. (ed.), *Selected Writings on Computing: A personal Perspective*, pages 129–131. Springer.

Dijkstra, E. W. (1988). On the cruelty of really teaching computing science. Technical Report EWD 1036, The University of Texas at Austin.

Dijkstra, E. W. (1993). In reply to comments. Technical Report EWD 1058, The University of Texas at Austin.

Dijkstra, E. W. (2002). Ewd 1308: What led to 'Notes on structured programming'. In Broy, M. and Denert, E. (eds.), *Software Pioneers*, pages 340–346. Springer.

diSessa, A. A. and Abelson, H. (1986). Boxer: A reconstructible computational medium. *Communications of the ACM*, 29(9):859–868.

Dyer, M. and Mills, H. D. (1981). Cleanroom software development. In *Sixth Annual Software Engineering Workshop*. NASA.

Ellis, T. O., Heafner, J. F., and Sibley, W. L. (1969). The GRAIL project: An experiment in man-machine communications. Technical Report, RAND Corporation, Santa Monica, CA.

Engelbart, D. C. (1962). Augmenting human intellect: A conceptual framework. Technical Report AFOSR 3223, Stanford Research Institute.

English, J. (1998). The story of the Java platform. Archived at http://web.archive.org/web/19990218201604/http://java.sun.com/nav/whatis/storyofjava.html (Retrieved on 19 Jan 2024).

Ensmenger, N. (2010). Making programming masculine. In Misa, T. J., *Gender Codes: Why Women Are Leaving Computing*, pages 115–141. IEEE Computer Society and John Wiley & Sons.

Ensmenger, N. (2012). *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. MIT Press.

Ensmenger, N. (2015). 'Beards, sandals, and other signs of rugged individualism': Masculine culture within the computing professions. *Osiris*, 30(1):38–65.

Evans, T. G. and Darley, D. L. (1965). Debug–an extension to current online debugging techniques. *Communications of the ACM*, 8(5):321–326.

Evans, T. G. and Darley, D. L. (1966). On-line debugging techniques: A survey. In *Proceedings of the November 7-10, 1966, fall joint computer conference*, pages 37–50. ACM.

Fedorkow, G. C. (2021). Recovering software for the whirlwind computer. *IEEE Annals of the History of Computing*, 43(1):38–59.

Fetzer, J. H. (1988). Program verification: The very idea. *Communications of the ACM*, 31(9):1048–1063.

Feyerabend, P. (1975). *Against Method*. Verso.

Finney, J. W. (1975). Safeguard ABM system to shut down: 5 billion spent in 6 years since debate. *The New York Times*.

Fisher, D. A. (1970). Control structures for programming languages. PhD thesis, Carnegie Mellon University.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Fisher, K. D., Carr, C. J., and Talbot, J. M. (1975). Computer applications in the biological sciences. Technical Report AD-A012 589, Air Force Office of Scientific Research, Advanced Research Projects Agency.

Floyd, C. (1987). Outline of a paradigm change in software engineering. In Bjerknes, G., (ed.), *Computers and Democracy: A Scandanavian Challenge*, pages 191–210. Brookfield, Gower Publishing Company.

Floyd, C., Mehl, W.-M., Resin, F.-M., Schmidt, G., and Wolf, G. (1989). Out of Scandinavia: Alternative approaches to software design and system development. *Human-Computer Interaction*, 4(4):253–350.

Floyd, R. W. (1967). Assigning meanings to program. In *Proc. Symposia in Applied Mathematics, 1967*, volume 19, pages 19–32.

Forrester, J. W. and Everett, R. R. (1990). The whirlwind computer project. *IEEE Transactions on Aerospace and Electronic Systems*, 26(5):903–910.

Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.

Fowler, M. (2001). The new methodology. *Wuhan University Journal of Natural Sciences*, 6(1):12–24.

Fox, P. (1960). LISP I programmers manual. Internal paper, MIT, Cambridge.

Freiberger, P. and Swaine, M. (1984). *Fire in the Valley: The Making of the Personal Computer*. McGraw-Hill.

Gabriel, R. P. (1996). *Patterns of Software: Tales from the Software Community*. Oxford University Press.

Gabriel, R. P. (2012). The structure of a programming language revolution. In *Proceedings of the ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2012, pages 195–214, New York. ACM.

Gabriel, R. P., Bourbaki, N., Devin, M., Dussud, P., Gray, D., and Sexton, H. (1990). Foundation for a c++ programming environment. *Proceedings of C++ at Work*, 90.

Gabriel, R. P. and Frost, M. E. (1984). A programming environment for a timeshared system. In *Proceedings of the First ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, SDE 1, page 185–192, New York. ACM.

Galison, P. (1997). *Image and Logic: A Material Culture of Microphysics*. University of Chicago Press.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson.

Gelperin, D. and Hetzel, B. (1988). The growth of software testing. *Communications of the ACM*, 31(6):687–695.

Genuys, F., (ed.). (1968). *Programming Languages: NATO Advanced Study Institute*. Academic Press.

Gilmore, Jr., J. T. (1958a). A functional description of the TX-0 computer. Technical Report 6M-4789-1, Massachusetts Institute of Technology.

Gilmore, Jr., J. T. (1958b). TX-0 direct input utility system. Technical Report 6M-5097-1, Massachusetts Institute of Technology.

Gold, B. and Simons, R. (2008). *Proof and Other Dilemmas: Mathematics and Philosophy*. MAA spectrum. Mathematical Association of America.

Goldberg, A. (2002). Oral history of Adele Goldberg, interviewed by Janet Abbate, 3 July, 2002. Technical Report Interview 591, IEEE History Center.

Goldberg, A. and Kay, A. (1976a). *Smalltalk-72: Instruction Manual*. Xerox Corporation Palo Alto.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Goldberg, A. and Kay, A. (1976b). Smalltalk-72: Instruction manual. Technical Report SSL 76-6, Xerox Corporation, Palo Alto.

Goldberg, A. and Robson, D. (1983). *Smalltalk-80: The Language and Its Implementation*. Addison-Wesley Longman Publishing.

Goldberg, J., Kautz, W. H., Melliar-Smith, P. M., Green, M. W., Levitt, K. N., Schwartz, R. L., and Weinstock, C. B. (1984). Development and analysis of the software implemented fault-tolerance (sift) computer. Technical Report NASA Contractor Report 172146, SRI International.

Gombrich, E. H. (1961). *Art and illusion*. Pantheon Books.

Goodenough, J. B. (1975a). Exception handling: Issues and a proposed notation. *Communications of the ACM*, 18(12):683–696.

Goodenough, J. B. (1975b). Structured exception handling. In Graham, R. M., Harrison, M. A., and Reynolds, J. C., (eds.), *Conference Record of the Second ACM Symposium on Principles of Programming Languages, Palo Alto*, pages 204–224. ACM Press.

Goodenough, J. B. and Gerhart, S. L. (1975). Toward a theory of test data selection. In *Proceedings of the International Conference on Reliable Software*, pages 493–510, New York. ACM.

Gordon, M. (2000). From LCF to HOL: A short history. In Plotkin, G., Stirling, C. P. and Tofte, M. (eds.), *Proof, Language and Interaction*, pages 169–186.

Gosling, J. and McGilton, H. (1995). The Java language environment: A whitepaper. Technical Report, Sun Microsystems Computer Company.

Grad, B. (2007). The creation and the demise of VisiCalc. *IEEE Annals of the History of Computing*, 29(3):20–31.

Greelish, D. (2013). An interview with computing pioneer Alan Kay. *Time Magazine*.

Greenblatt, R. (2005). Oral history of Richard Greenblatt, interviewed by Gardner Hendrie, January 12, 2005. Technical Report X3056.2005, Computer History Museum.

Gresham-Lancaster, S. (1998). The aesthetics and history of the Hub: The effects of changing technology on network computer music. *Leonardo Music Journal*, 8(1):39–44.

Gries, D., (ed.). (1978). *Programming Methodology: A Collection of Articles by Members of IFIP WG 2.3*. Springer-Verlag, Berlin, Heidelberg.

Gschwind, M. (1952). The moniac. *Fortune Magazine*, 100–101.

Gugerty, L. (2006). Newell and Simon's logic theorist: Historical background and impact on cognitive modeling. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):880–884.

Habermann, A. N. (1969). Prevention of system deadlocks. *Communications of the ACM*, 12(7):373–ff.

Hacking, I. (1983). *Representing and Intervening: Introductory Topics in the Philosophy of Natural Science*. Cambridge University Press.

Haigh, T. (2010). Dijkstra's crisis: The end of Algol and beginning of software engineering, 1968-72. [Online; accessed August 27, 2021].

Haigh, T. and Ceruzzi, P. E. (2021). *A New History of Modern Computing*. MIT Press.

Haigh, T., Priestley, P. M., Priestley, M., and Rope, C. (2016). *ENIAC in Action: Making and Remaking the Modern Computer*. MIT Press.

Halpern, M. (1965). Computer programming: The debugging epoch opens. *Computers and Automation*, 14(11):28–31.

Halvorson, M. J. (2020). *Code Nation: Personal Computing and the Learn to Program Movement in America*. ACM Books.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Harel, D. (1980). On folk theorems. *Communications of the ACM*, 23(7):379–389.

Harrison, J., Urban, J., and Wiedijk, F. (2014). History of interactive theorem proving. *Computational Logic*, 9:135–214.

Henhapl, W. and Jones, C. B. (1978). *A formal definition of Algol 60 as described in the 1975 modified report*, pages 305–336. Springer.

Hetzel, B. (1988). *The Complete Guide to Software Testing*. QED Information Sciences, 2nd edition.

Hetzel, W., (ed.). (1973). *Program Test Methods*. Prentice-Hall.

Hicks, M. (2010). Meritocracy and feminization in conflict: computerization in the British government. In Misa, T. J. (ed.), *Gender Codes: Why Women Are Leaving Computing*, pages 95–114. IEEE Computer Society and John Wiley & Sons, Inc., Hoboken, New Jersey.

Hicks, M. (2017). *Programmed Inequality: How Britain Discarded Women Technologists and Lost Its Edge in Computing*. MIT Press.

Hicks, M. (2020). Built to last. *Logic Magazine*, (11).

Hiltzik, M. A. et al. (1999). *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age*. HarperCollins Publishers.

Hoare, C. A. (1971). Proof of a program: Find. *Communications of the ACM*, 14(1):39–45.

Hoare, C. A. R. (1965). Record handling. *Algol Bulletin*, 21:39–69.

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580.

Hoare, C. A. R. (1972). Notes on data structuring. In Dahl, O. J., Dijkstra, E. W. and Hoare, C. A. R. (eds.), *Structured Programming*. pages 83–174. Academic Press.

Hoare, C. A. R. (1981). The emperor's old clothes. *Communications of the ACM*, 24(2):75–83.

Hoare, C. A. R. (1996). How did software get so reliable without proof? In Gaudel, M.-C. and Woodcock, J., (eds.), *FME'96: Industrial Benefit and Advances in Formal Methods*, pages 1–17, Springer.

Hodges, W. (2001). A history of british logic. http://wilfridhodges.co.uk/history01.pdf.

Hohpe, G. (2005). Revenge of the nerds – OOPSLA 2005. www.enterpriseintegrationpatterns.com/ramblings/37_oopsla.html.

Holden, D. and Kerr, C. (2023). *./code-poetry*. Broken Sleep Books.

Holmevik, J. (1994). Compiling simula: A historical study of technological genesis. *IEEE Annals of the History of Computing*, 16(4):25–37.

Honda, K. (1993). Types for dyadic interaction. In Best, E. (ed.), 4th International Conference on Concurrency Theory (CONCUR' 93), Hildesheim, Germany, 23–26 August, 509–523. Springer.

Hooker, S. (2021). Moving beyond 'algorithmic bias is a data problem'. *Patterns*, 2(4):100241.

Hoyningen-Huene, P. and Sankey, H., (eds.). (2001). *Incommensurability and Related Matters*. Dordrecht: Kluwer.

Huang, J. C. (1975). An approach to program testing. *ACM Computing Surveys*, 7(3):113–128.

Hudak, P., Hughes, J., Peyton Jones, S., and Wadler, P. (2007). A history of Haskell: Being lazy with class. In *Proceedings of the Third ACM SIGPLAN Conference on History of Programming Languages*, HOPL III, page 12–1–12–55, New York. ACM.

Hunt, J. (1997). *Smalltalk and Object Orientation: An Introduction*. Springer Science & Business Media.

IBM (1967). IBM System/360: PL/I subset reference manual. Technical Report S360-29, IBM Systems Reference Library.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

IBM (1960). *General Information Manual: IBM Commercial Translator*. IBM. Online at http://bitsavers.org/pdf/ibm/7090/F28-8043_CommercialTranslatorGenInfMan_Ju60.pdf.

Ichbiah, J. D., Krieg-Brueckner, B., Wichmann, B. A., Barnes, J. G. P., Roubine, O., and Heliard, J.-C. (1979). Rationale for the design of the Ada programming language. *SIGPLAN Notices*, 14(6b):1–261.

Ingalls, D. (2020). The evolution of Smalltalk: From Smalltalk-72 through Squeak. *Proceedings of the ACM on Programming Languages*, 4(HOPL).

Ingalls, D., Kaehler, T., Maloney, J., Wallace, S., and Kay, A. (1997). Back to the future: The story of Squeak, a practical Smalltalk written in itself. In Berman, A. M. (ed.), *Proceedings of the 12th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA '97, page 318–326, New York. ACM.

Ingalls, D., Palacz, K., Uhler, S., Taivalsaari, A., and Mikkonen, T. (2008). The lively kernel a self-supporting system on a web page. In Hirschfeld, R. and Rose, K. (eds.), *Self-Sustaining Systems: First Workshop, S3 2008 Potsdam, Germany, May 15–16, 2008 Revised Selected Papers*, pages 31–50. Springer.

Ingalls, D. H. H. (1978). The Smalltalk-76 programming system design and implementation. In Szymanski, T. G. (ed.), *Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '78, page 9–16, New York. ACM.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley.

Jacobson, I., Ng, P.-W., McMahon, P. E., Goedicke, M., et al. (2019). *The Essentials of Modern Software Engineering: Free the Practices from the Method Prisons!* Morgan & Claypool.

Jakubovic, J., Edwards, J., and Petricek, T. (2023). Technical dimensions of programming systems. *The Art, Science, and Engineering of Programming*, 7(3):13–1.

Johns, A. (2010). *Piracy: The Intellectual Property Wars from Gutenberg to Gates*. University of Chicago Press.

Johnson, J., Roberts, T. L., Verplank, W., Smith, D. C., Irby, C. H., Beard, M., and Mackey, K. (1989). The Xerox star: A retrospective. *Computer*, 22(9):11–26.

Jones, A. K. and Liskov, B. (1976). An access control facility for programming languages. Technical Report AFOSR-TR 76-0886, Massachusetts Institute of Technology, Laboratory for Computer Science.

Jones, C. B. and Astarte, T. K. (2016). An Exegesis of Four Formal Descriptions of Algol 60. Technical Report CS-TR-1498, Newcastle University School of Computer Science.

(abstract). In Silberber, M. Y. (ed.), *Proceedings of the November 16-18, 1971, Fall Joint Computer Conference*, AFIPS '71 (Fall), page 395–396, New York. ACM.

Kay, A. (1972b). A personal computer for children of all ages. In Donovan, J. J. and Shields, R. (eds.), *Proceedings of the ACM annual conference-Volume 1*.

Kay, A. (1974). Summary of Smalltalk message forms and intentions. Technical Report, Xerox Palo Alto Research Center.

Kay, A. (1997). The computer revolution hasn't happened yet (keynote). The 12th annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications, Atlanta, USA.

Kay, A. (1998). prototypes vs classes was: Re: Sun's hotspot. http://lists.squeakfoundation.org/pipermail/squeak-dev/1998-October/017019.html. E-mail sent to the squeak-dev mailing list, Accessed: July 25, 2023.

Kay, A. (2003). Clarification of 'object-oriented'. http://userpage.fu-berlin.de/~ram/pub/pub_jf47ht81Ht/doc_kay_oop_en. E-mail exchange with Stefan Ram, Accessed: July 25, 2023.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Kay, A. and Goldberg, A. (1977). Personal dynamic media. *Computer*, 10(3):31–41.

Kay, A. C. (1968). FLEX, a flexible extendable language. Master's thesis, Utah University.

Kay, A. C. (1969). The reactive engine. PhD thesis, Utah University.

Kay, A. C. (1996). The early history of Smalltalk. *History of programming languages—II*, pages 511–598.

Kell, S. (2014). In search of types. In Bruegge, B. and Krishnamurthi, S. (eds.), *Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software*, Onward! 2014, pages 227–241, New York. ACM.

Kell, S. (2017). Some were meant for C: The endurance of an unmanageable language. In Torlak, E., van der Storm, T., and Biddle, R., (eds.), *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Onward!* Vancouver, BC, October 23–27, 2017, pages 229–245. ACM.

Kell, S. (2018). Unix, Plan 9 and the lurking Smalltalk. In De Mol, L. and Primiero, G. (eds.), Philosophical Studies Series, vol 133. *Reflections on Programming Systems: Historical and Philosophical Aspects*, Philosophical Studies Series, vol 133, 189–213.

Kelty, C. M. (2008). *Two Bits: The Cultural Significance of Free Software*. Duke University Press.

Kennedy, A. J. (1996). Programming languages and dimensions. PhD thesis, University of Cambridge, Computer Laboratory.

Kiczales, G., Des Rivieres, J., and Bobrow, D. G. (1991). *The Art of the Metaobject Protocol*. MIT Press.

Kidwell, P. A. (1998). Stalking the elusive computer bug. *IEEE Annals of the History of Computing*, 20(4):5–9.

Kleiman, K. (2022). *Proving Ground: The Untold Story of the Six Women Who Programmed the World's First Modern Computer*. Hurst Publishers.

Klein, G., Andronick, J., Fernandez, M., Kuz, I., Murray, T. C., and Heiser, G. (2018). Formally verified software in the real world. *Communications of the ACM*, 61(10):68–77.

Kleiner, D. (2010). *The telekommunist manifesto*, volume 3. Institute of Network Cultures Amsterdam.

Knuth, D. (1968). *The Art of Computer Programming*, volume 1–4. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Knuth, D. E. (1973). A review of 'structured programming'. Technical Report STAN-CS-73-371, Computer Science Department, Stanford University.

Kotok, A. (2005). Oral history of Alan Kotok, interviewed by Gardner Hendrie, November 15, 2004. Technical Report X3004.2005, Computer History Museum.

Kristensen, B. B., Madsen, O. L., and Møller-Pedersen, B. (2007). The when, why and why not of the BETA programming language. In Ryder, B. and Hailpern, B. (eds.), *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, pages 10–1.

Kuhn, T. (1962). *The Structure of Scientific Revolutions*. University of Chicago Press.

Kusner, M. J., Loftus, J., Russell, C., and Silva, R. (2017). Counterfactual fairness. *Advances in Neural Information Processing Systems*, 30.

Laan, T. D. L. (1997). The evolution of type theory in logic and mathematics. PhD thesis.

Lakatos, I. (1976). *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge University Press.

Lampson, B. (1972). Why Alto? Technical Report, Xerox PARC. www.microsoft.com/en-us/research/wp-content/uploads/2016/11/Acrobat-1.pdf.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Landin, P. J. (1964). The mechanical evaluation of expressions. *The Computer Journal*, 6(4):308–320.

Landin, P. J. (1965a). Correspondence between Algol 60 and Church's lambda-notation: Part i. *Communications of the ACM*, 8(2):89–101.

Landin, P. J. (1965b). A correspondence between Algol 60 and Church's lambda-notations: Part ii. *Communications of the ACM*, 8(3):158–167.

Landin, P. J. (1966a). A formal description of Algol 60. In Steel, T. B. Jr. (ed.), *Formal Language Description Languages for Computer Programming*, pages 266–294. North Holland Publishing Company, Amsterdam.

Landin, P. J. (1966b). The next 700 programming languages. *Communications of the ACM*, 9(3):157–166.

Latour, B. (1987). *Science in Action: How to Follow Scientists and Engineers through Society*. Harvard university press.

Lauer, P. E. (1968). Formal definition of Algol 60. Technical Report 25.088, IBM Laboratory Vienna.

Leavenworth, B. M. (1972). Programming with(out) the goto. *SIGPLAN Not.*, 7(11):54–58.

Lee, G. (2021). The three amigos, among others. *De Programmatica Ipsum*, (39).

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076.

Lehman, M. M. (1993). An interview conducted by William Aspray, IEEE History Center, September 23, 1993. Technical Report 178, The Institute of Electrical and Electronics Engineers, Inc.

Lehman, M. M. and Belady, L. A. (1985). *Program Evolution: Processes of Software Change*. Academic Press.

Lennon, B. (2019). Foo, bar, baz…: The metasyntactic variable and the programming language hierarchy. In Petricek, T., Durnova, H. and Priestley, M. (eds.), *Philosophy & Technology*. 34(1):13–32.

Leroy, X. (2009). Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115.

Levy, S. (2010). *Hackers: Heroes of the Computer Revolution*. O'Reilly Media.

Licklider, J. C. (1969). Underestimates and overexpectations. *Computers and Automation*, 18(9):48–52.

Licklider, J. C. R. (1960). Man-computer symbiosis. *IRE Transactions on Human Factors in Electronics*, HFE-1(1):4–11.

Licklider, J. C. R. (1965). *Libraries of the Future*. MIT Press.

Lindsey, C. H. (1996a). *A History of Algol 68*, page 27–96, New York. ACM.

Lindsey, C. H. (1996b). *A History of Algol 68*, page 27–96, New York. ACM.

Liskov, B. (1993). A history of Clu. In Bergin, T. J. and Gibson, R. G. (eds.), *The Second ACM SIGPLAN Conference on History of Programming Languages*, HOPL-II, pages 133–147, New York. ACM.

Liskov, B. and Zilles, S. (1974). Programming with abstract data types. In Leavenworth, B., (ed.), *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages*, pages 50–59, New York. ACM.

Livingston, E. (1999). Cultures of proving. *Social Studies of Science*, 29(6):867–888.

Livshits, B., Sridharan, M., Smaragdakis, Y., Lhoták, O., Amaral, J. N., Chang, B.-Y. E., Guyer, S. Z., Khedker, U. P., Møller, A., and Vardoulakis, D. (2015). In defense of soundiness: A manifesto. *Communications of the ACM*, 58(2):44–46.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Llewelyn, A. and Wickens, R. (1968). The testing of computer software. In Naur, P. and Randall, B. (eds.), *Software Engineering, Report on a conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch, Germany*, pages 7–11.

MacKenzie, D. (2004). *Mechanizing Proof: Computing, Risk, and Trust*. Inside technology. MIT Press.

MacKenzie, D. (2005). Computing and the cultures of proving. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 363(1835):2335–2350.

MacKenzie, D. (2014). A sociology of algorithms: High-frequency trading and the shaping of markets. https://c.mql5.com/forextsd/forum/169/algorithms25.pdf.

MacQueen, D., Harper, R., and Reppy, J. (2020). The history of Standard ML. In Steele, G. L. Jr. (ed.), *Proceedings of the ACM on Programming Languages*, 4(HOPL):1–100.

Madsen, O. L. and Møller-Pedersen, B. (2022). What object-oriented programming was supposed to be: Two grumpy old guys' take on object-oriented programming. In Scholliers, C. and Singer, J., (eds.), *Proceedings of the 2022 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, Onward!* Auckland, New Zealand, December 8–10, 2022, pages 220–239. ACM.

Magnusson, L. and Nordström, B. (1993). The ALF proof editor and its proof engine. In Barendregt, H. and Nipkow, T. (eds.), *International Workshop on Types for Proofs and Programs*, pages 213–237. Springer.

Mahoney, M. S. (1988). The history of computing in the history of technology. *Annals of the History of Computing*, 10(2):113–125.

Mahoney, M. S. (1992). Computers and mathematics: The search for a discipline of computer science. In Echeverría, J., Ibarra, A. and Mormann, T. (eds.), *The Space of Mathematics. Philosophical, Epistemological, and Historical Explorations. Berlin: Springer*, pages 349–363.

Mahoney, M. S. (1997). *Computer Science: The Search for a Mathematical Theory*. Taylor and Francis.

Mahoney, M. S. (2005). The histories of computing(s). *Interdisciplinary Science Reviews*, 30(2):119–135.

Markoff, J. (2005). *What the dormouse said: How the sixties counterculture shaped the personal computer industry*. Viking Press.

Martin, W. and Hart, T. (1964). Time sharing LISP. Technical Report AIM-67, MIT.

Martini, S. (2016). Several types of types in programming languages. In Gadducci, F. and Tavosanis, M., (eds.), *History and Philosophy of Computing*, pages 216–227. Springer International Publishing.

McBride, C. (2002). Faking it simulating dependent types in Haskell. *Journal of Functional Programming*, 12(4–5):375–392.

McBride, C. and McKinna, J. (2004). The view from the left. *Journal of Functional Programming*, 14(1):69–111.

McBurney, P. (2009). Guerrilla logic: A salute to Mervyn Pragnell. https://vukutu.com/blog/2009/09/guerrilla-logic-a-salute-to-mervyn-pragnell/.

McCabe, T. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320.

McCarthy, J. (1961). A basis for a mathematical theory of computation, preliminary report. In Bauer, W. F. (ed.), *Papers presented at the May 9–11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, pages 225–238.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

McCarthy, J. (1962). Computer programs for checking mathematical proofs. In Dekker, J. C. E. (ed.), *Proceedings of Symposia in Pure Mathematics, vol 5*. American Mathematical Society.

McCarthy, J. (1963). Towards a mathematical science of computation. In Popplewell, C. M., (ed.), *Information Processing*, pages 21–28. North-Holland Publishing Company.

McCarthy, J. (1964). A formal description of a subset of Algol. In Steel, T. B. (ed.), *Formal Language Description Languages for Computer Programming*, pages 1–12. North-Holland Publishing Company.

McCarthy, J. (1978). History of LISP. In Wexelblat, R. L. (ed.), *History of Programming Languages*, pages 173–185, ACM.

McCarthy, J. (1992). Reminiscences on the history of time-sharing. *IEEE Annals of the History of Computing*, 14(1):19–24.

McCarthy, J., Abrahams, P. W., Edwards, D. J., Hart, T. P., and Levin, M. I. (1962). *LISP 1.5 Programmer's Manual*. MIT Press.

McCracken, D. (1973). Revolution in programming: An overview. *Datamation*, pages 50–52.

McGonagle, J. D. (1972). Notes on the computer program test methods symposium. *SIGPLAN Notices*, 7(5):8–12.

McKenzie, J. A. (1974). TX-0 computer history. Technical Report 627, MIT RLE.

McKinsey (1968). Unlocking the computer's profit potential. Technical Report, McKinsey & Company, Inc.

Meerts, J. and Graham, D. (2010). The History of Software Testing. www.testingreferences.com/testinghistory.php.

Meyer, B. (1992). *Eiffel: The Language*. Object-Oriented Series, Prentice Hall.

Meyer, E. A. (2002). 'Considered Harmful': essays considered harmful. https://meyerweb.com/eric/comment/chech.html.

Mills, H., Dyer, M., and Linger, R. (1987). Cleanroom software engineering. *IEEE Software*, 4(5):19–25.

Milner, R. (1972). Logic for computable functions: Description of a machine implementation. Technical Report, Stanford, CA, USA.

Milner, R. (1978). A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17(3):348–375.

Milner, R. (1979). LCF: A way of doing proofs with a machine. In Bečvář, J., (ed.), *Mathematical Foundations of Computer Science 1979*, pages 146–159. Springer.

Milner, R. (2003). An interview with Robin Milner, interviewed by Martin Berger, September 3, 2003. https://users.sussex.ac.uk/~mfb21/interviews/milner/.

Milner, R., Tofte, M., and Harper, R. (1990). *Definition of Standard ML*. MIT Press.

Milner, R. and Weyhrauch, R. (1972). Proving compiler correctness in a mechanized logic. *Machine Intelligence*, 7(3):51–70.

Mindell, D. A. (2011). *Digital Apollo: Human and Machine in Spaceflight*. MIT Press.

Misa, T. J. (2011). *Gender Codes: Why Women Are Leaving Computing*. John Wiley & Sons.

Mitchell, J. C. (2003). *Concepts in Programming Languages*. Cambridge University Press.

Montfort, N., Baudoin, P., Bell, J., Bogost, I., and Douglass, J. (2014). *10 PRINT CHR (205.5+ RND (1));: GOTO 10*. MIT Press.

Moore, J. S. (2019). Milestones from the pure LISP theorem prover to ACL2. *Formal Aspects of Computing*, 31(6):699–732.

Morris, J. H. (1973a). Protection in programming languages. *Communications of the ACM*, 16(1):15–21.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Morris, Jr., J. H. (1973b). Types are not sets. In Ullman, J. D. (ed.), *Proceedings of the 1st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '73, pages 120–124, New York. ACM.

Morris Jr, J. H. (1969). Lambda-calculus models of programming language. PhD thesis, Massachusetts Institute of Technology.

Mosses, P. D. (1974). The mathematical semantics of Algol 60. Technical Report Technical Monograph PRG-12, Oxford University Computing Laboratory, Programming Research Group.

Mugridge, R. (2003). Test driven development and the scientific method. In Cockburn, A. (ed.), *Proceedings of the Agile Development Conference, 2003. ADC 2003*, pages 47–52. IEEE.

Murhpy, E. M. (2013). In the Matter of Knight Capital Americas LLC Respondent. Technical Report SEC Release No. 70694, File No. 3-15570, Securities and Exchange Commission.

Musa, J. D. (1975). A theory of software reliability and its application. *IEEE Transactions on Software Engineering*, 1:312–327.

Myers, G. J., Badgett, T., and Sandler, C. (1979). *The Art of Software Testing.* John Wiley & Sons, pages 22041–3467.

Nake, F. (1971). There should be no computer art. In Metzger, G. (ed.), *Bulletin of the Computer Arts Society*, PAGE 18:1–2.

Nance, R. E. (1993). A history of discrete event simulation programming languages. In *The Second ACM SIGPLAN Conference on History of Programming Languages*, HOPL-II, page 149–175, New York. ACM.

National Bureau of Standards. (1984). Guideline for lifecycle validation, verification, and testing of computer software. Technical Report, U.S. Dept. of Commerce, National Bureau of Standards.

National Museum of American History (2024). Log book with computer bug. https://americanhistory.si.edu/collections/nmah_334663.

Naur, P. (1966). Proof of algorithms by general snapshots. *BIT Numerical Mathematics*, 6(4):310–316.

Naur, P. (1978). *The European Side of the Last Phase of the Development of Algol 60*, page 92–139, New York. ACM.

Naur, P. (1993). The place of strictly defined notation in human insight. In Colburn, T. R., Fetzer, J. H. and Rankin, T. L. (eds.), *Program Verification*, pages 261–274. Springer.

Naur, P., Randell, B., Bauer, F., and Committee, N. S. (1969). *Software engineering: report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968*. Scientific Affairs Division, NATO.

Neff, G. and Nagy, P. (2016). Talking to bots: Symbiotic agency and the case of Tay. *International Journal of Communication*, 10:4915–4931.

Newell, A. and Simon, H. (1956). The logic theory machine: A complex information processing system. *IRE Transactions on Information Theory*, 2(3):61–79.

Noble, S. U. (2018). *Algorithms of Oppression*. New York University Press.

Nofre, D., Priestley, M., and Alberts, G. (2014). When technology became language: The origins of the linguistic conception of computer programming, 1950–1960. *Technology and Culture*, 55(1):40–75.

Noll, A. M. (2016). The Howard Wise gallery show computer-generated pictures (1965): A 50th-anniversary memoir. *Leonardo*, 49(3):232–239.

Nooney, L. (2023). *The Apple II Age: How the Computer Became Personal*. University of Chicago Press.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Nordström, B., Petersson, K., and Smith, J. M. (1990). *Programming in Martin-Löf's Type Theory, Volume 200*. Oxford University Press, Oxford.

Norell, U. (2007). Towards a practical programming language based on dependent type theory. PhD thesis.

November, J. (2004). Linc: Biology's revolutionary little computer. *Endeavour*, 28(3):125–131.

Nygaard, K. and Dahl, O.-J. (1978). The development of the SIMULA languages. *SIGPLAN Notices*, 13(8):245–272.

O'Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Broadway Books.

Ornstein, S. (2002). *Computing in the Middle Ages: A View from the Trenches*. AuthorHouse.

Papert, S. (1980). *Mindstorms: Computers, Children, and Powerful Ideas*. Basic Books.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058.

Parnas, D. L. (1985). Software aspects of strategic defense systems. *Communications of the ACM*, 28(12):1326–1335.

Passani, L. (1996). Java naysayer or realist? (letter). *Dr. Dobb's Journal*, (251):10–11.

Paulson, L. C. (1993). Designing a theorem prover. In Abramsky, S., Gabbay, D. M. and Spurr, J. (eds.), *Handbook of Logic in Computer Science (vol. 2) Background: Computational Structures*, pages 415–475.

Pelaez Valdez, M. E. (1988). A gift from Pandora's box: The software crisis. PhD thesis, University of Edinburgh.

Perlis, A. J. (1978). *The American Side of the Development of Algol*, page 75–91, New York. ACM.

Perlis, A. J. and Samelson, K. (1958). Preliminary report: International algebraic language. *Communications of the ACM*, 1(12):8–22.

Petricek, T. (2015). Against a universal definition of 'type'. In Murphy, G. C. and Steele, G. L. Jr. (eds.), *2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, Onward! 2015, pages 254–266, New York. ACM.

Petricek, T. (2017). Miscomputation in software: Learning to live with errors. *The Art, Science, and Engineering of Programming*, 1(2):14.

Petricek, T. (2018). What we talk about when we talk about monads. *The Art, Science, and Engineering of Programming*, 2(3):12.

Petricek, T. (2020). The lost ways of programming: Commodore 64 basic. Presented at HaPoC 2020.

Petricek, T. (2023). The rise and fall of extensible programming languages. Abstract presented at 7th International Conference on the History and Philosophy of Computing.

Petricek, T., Orchard, D., and Mycroft, A. (2014). Coeffects: A calculus of context-dependent computation. In Chakravarty, M. (ed.), *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, ICFP '14, pages 123–135, New York. ACM.

Pierce, B. (2002). *Types and Programming Languages*. MIT Press.

Pirsig, R. M. (1999). *Zen and the Art of Motorcycle Maintenance:An Inquiry into Values*. Random House.

Pitts, A. M. (2011). Lecture notes on types for Part II of the computer science tripos. www.cl.cam.ac.uk/teaching/1112/Types/typ-notes.pdf.

Plauger, P. J. (1993). *Programming on Purpose*. PTR Prentice Hall.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Pleasant, James, C. (1989). The very idea (technical correspondence). *Communications of the ACM*, 32(3):374–381.

Plotkin, G. and Pretnar, M. (2009). Handlers of algebraic effects. In Castagna, G. (ed.), *European Symposium on Programming*, pages 80–94. Springer.

Polanyi, M. (1958). *Personal Knowledge*. University of Chicago Press.

Polanyi, M. (1967). *The Tacit Dimension*. University of Chicago Press.

Postley, J. A. (1960). Letters to the editor. *Communications of the ACM*, 3(1):0.06–.

Priestley, M. (2011). *A Science of Operations: Machines, Logic and the Invention of Programming*. Springer.

Priestley, M. (2017). AI and the origins of the functional programming language style. *Minds and Machines*, 27(3):449–472.

Primiero, G. (2019). *On the Foundations of Computing*. Oxford University Press.

PROGRAMme (2022). *What is a computer program?* In De Mol, E. (ed.), title and publisher to be added at proof stage.

Project MAC (1967). Project MAC progress report III – July 1965 to July 1966. Technical Report DTIC AD-648346, MIT.

Pym, D., Spring, J. M., and O'Hearn, P. (2019). Why separation logic works. *Philosophy & Technology*, 32(3):483–516.

Radin, G. (1978). The early history and characteristics of PL/I. *SIGPLAN Notices*, 13(8):227–241.

Randell, B. (1975). System structure for software fault tolerance. *IEEE Transactions on Software Engineering*, SE-1(2):220–232.

Rankin, J. L. (2018). *A People's History of Computing in the United States*. Harvard University Press.

Raymond, E. S. (1996). *The New Hacker's Dictionary*. MIT Press.

Raymond, E. S. (1997). *The Cathedral and the Bazaar*. O'Reilly Media.

Raymond, E. S. (2003). *The Art of UNIX Programming*. Addison-Wesley Professional.

Redmond, K. C. and Smith, T. M. (2000). *From Whirlwind to MITRE: The R&D Story of the SAGE Air Defense Computer*. MIT Press.

Reenskaug, T. M. H. (1981). User-oriented descriptions of smalltalk systems. *Byte, 6*, 8.

Rentsch, T. (1982). Object oriented programming. *SIGPLAN Notices*, 17(9):51–57.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11):60–67.

Reynolds, J. C. (1974). Towards a theory of type structure. In Robinet, B., (ed.), *Programming Symposium*, pages 408–425, Springer.

Rheingold, H. (2000). *Tools for Thought: The History and Future of Mind-Expanding Technology*. MIT Press.

Richards, G., Nardelli, F. Z., and Vitek, J. (2015). Concrete Types for TypeScript. In Boyland, J. T., (ed.), *29th European Conference on Object-Oriented Programming (ECOOP 2015)*, volume 37, pages 76–100, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Rochester, N. (1960). Foreword. In Rochester, N. (ed.), *Papers Presented at the December 13-15, 1960, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '60 (Eastern), page 1–9, New York, ACM.

Rosenwasser, D. (2018). Announcing TypeScript 3.0. https://blogs.msdn.microsoft.com/typescript/2018/07/30/announcing-typescript-3-0/.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Ross, D. T. (1961). A generalized technique for symbol manipulation and numerical calculation. *Communications of the ACM*, 4(3):147–150.

Royce, W. W. (1970). Managing the development of large software systems. In *Proceedings IEEE WESCON*, pages 1–9.

Rushby, J. M. and Von Henke, F. (1993). Formal verification of algorithms for critical systems. *IEEE Transactions on Software Engineering*, 19(1):13–23.

Russell, B. (1903). Appendix B: The doctrine of types. In Russell, B. (ed.), *Principles of Mathematics*, pages 523–528. Cambridge University Press.

Russell, B. (1908). Mathematical logic as based on the theory of types. *American Journal of Mathematics*, 30(3):222–262.

Russell, S. (2017). Interview with Stephen (Steve) 'Slug' Russell, conducted by Christopher Weaver, January 8, 2017. Technical Report, Video Game Pioneers Oral History Collection, Archives Center, National Museum of American History, Smithsonian Institution, Washington, DC.

Russell, S. R. (1963). Improvements in LISP debugging. Technical Report AIM-10, Stanford Artificial Intelligence Project.

Ryder, B. G., Soffa, M. L., and Burnett, M. (2005). The impact of software engineering research on modern progamming languages. *ACM Transactions on Software Engineering and Methodology*, 14(4):431–477.

Sack, W. (2019). *The Software Arts*. MIT Press.

Sack, W. (2022). Miniatures, demos and artworks: Three kinds of computer program, their uses and abuses. In De Mol, L. and Petricek, T. (eds.), *Fifth Symposium on the History and Philosophy of Programming*, HaPoP 2022, pages 21–24. HAPOC.

Sammet, J. E. (1961). Detailed description of cobol. In Goodman, R., (ed.), *Annual Review in Automatic Programming*, volume 2 of *International Tracts in Computer Science and Technology and Their Application*, pages 197–230. Elsevier.

Sammet, J. E. (1969). *Programming Languages: History and Fundamentals*. Prentice-Hall, Inc.

Sammet, J. E. (1978). The early history of COBOL. *SIGPLAN Notices*, 13(8):121–161.

Samson, P. (2017). Interview with Peter Samson, conducted by Christopher Weaver, January 9, 2017. Technical Report, Video Game Pioneers Oral History Collection, Archives Center, National Museum of American History, Smithsonian Institution, Washington, DC.

Schank, R. and Riesbeck, C. (1981). *Inside Computer Understanding: Five Programs Plus Miniatures*. Lawrence Erlbaum Associates.

Schuman, S. A. and Jorrand, P. (1970). Definition mechanisms in extensible programming languages. In Brown, D. (ed.), *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, AFIPS '70 (Fall), pages 9–20, New York. ACM.

Schwaber, K. (1997). Scrum development process. In Sutherland, J., Casanave, C., Miller, J., Patel, P. and Hollowell, G. (eds.), *Business Object Design and Implementation*, pages 117–134. Springer.

Scott, D. S. (1993). A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121(1):411–440.

Seemann, M. (2021). *Code That Fits in Your Head: Heuristics for Software Engineeing*. Addison-Wesley Professional.

Shapin, S. and Schaffer, S. (2011). *Leviathan and the Air-Pump: Hobbes, Boyle, and the Experimental Life*. Princeton University Press.

Simondon, G. (2016). *On the Mode of Existence of Technical Objects*. Univocal Publishing.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Skelton, M., Pais, M., and Malan, R. (2019). *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. IT Revolution Press.

Slayton, R. (2013). *Arguments That Count: Physics, Computing, and Missile Defense, 1949-2012*. Arguments that Count. MIT Press.

Smith, B. C. (1985). Limits of correctness in computers. Technical Report CSLI-85-36, The Center for the Study of Language and Information, Stanford, CA.

Smith, D. A., Raab, A., Reed, D., and Kay, A. (2002). Croquet: The user manual (draft revision 0.1). Technical Report, Viewpoints Research Institute, Glendale.

Smith, D. C. (1977). *Pygmalion: A Computer Program to Model and Stimulate Creative Thought*. Birkhauser.

Smith, R. B. and Ungar, D. (1995). Programming as an experience: The inspiration for Self. In Tokoro, M. and Pareschi, R. (eds.), *European Conference on Object-Oriented Programming*, pages 303–330. Springer.

Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M. L., Minsky, M., Papert, A., and Silverman, B. (2020). History of Logo. *Proceedings of the ACM on Programming Languages*, 4(HOPL):79:1–79:66.

Stallman, R. M. (1986). What is the free software foundation? *GNU Bulletin*, 1(1).

Star, S. L. and Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907–39. *Social Studies of Science*, 19(3):387–420.

Steele, G. L., (ed.). (1983). *The Hacker's Dictionary*. Harper & Row.

Steele, G. L. and Gabriel, R. P. (1996a). The evolution of LISP (draft). www.dreamsongs.com/Files/HOPL2-Uncut.pdf.

Steele, G. L. and Gabriel, R. P. (1996b). The evolution of Lisp. In *History of Programming Languages—II*, pages 233–330. ACM

Steenson, M. W. (2022). *Architectural Intelligence: How Designers and Architects Created the Digital Landscape*. MIT Press.

Stockham, T. G. and Dennis, J. B. (1960). FLIT – Flexowriter Interrogation Tape: A symbolic utility program for the TX-0. Technical Report M-5001-23, MIT.

Stroustrup, B. (1986). *The C++ Programming Language*. Addison-Wesley.

Stroustrup, B. (1988). What is object-oriented programming? *IEEE software*, 5(3):10–20.

Stroustrup, B. (1994). *The Design and Evolution of C++*. Pearson Education.

Stroustrup, B. (1996). A history of C++ 1979–1991. In *History of Programming Languages—II*, pages 699–769. ACM.

Stroustrup, B. (2007). Evolving a language in and for the real world: C++ 1991–2006. In Ryder, B. and Hailpern, B. (eds.), *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, pages 4–1.

Sutherland, I. E. (1963). Sketchpad: A Man-Machine Graphical Communication System. PhD thesis, MIT.

Sutherland, W. R. (1966). On-line Graphical Specification of Computer Procedures. PhD thesis, MIT, Lincoln Labs Report TR-405.

Syme, D. (2020). The early history of F#. In Steele, G. L. Jr. (ed.), *Proceedings of the ACM on Programming Languages*, 4(HOPL):1–58.

Syme, D., Petricek, T., and Lomov, D. (2011). The F# asynchronous programming model. In Rocha, R. and Launchbury, J. (eds.), *International Symposium on Practical Aspects of Declarative Languages*, pages 175–189. Springer.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Takeuchi, H. and Nonaka, I. (1986). The new new product development game. *Harvard Business Review*, 64(1):137–146.

Talpin, J. and Jouvelot, P. (1994). The type and effect discipline. *Information and Computation*, 111(2):245–296.

Tan, C. (2020). The poetics of computer code: Tracing digital inscription in Ada Lovelace's England. *Digital Studies/Le champ numérique*, 10(1).

Tedre, M. (2014). *The Science of Computing: Shaping a Discipline*. CRC Press.

Teitelman, W. (1965). EDIT and BREAK functions for LISP. Technical Report AIM-84, MIT.

Teitelman, W. (1966). PILOT: A Step toward Man-Computer Symbiosis. PhD thesis, MIT.

Teitelman, W. (1974). INTERLISP reference manual. Technical Report, Xerox Palo Alto Research Center.

Teitelman, W. (1984). The Cedar programming environment: A midterm report and examination. Technical Report, Xerox Palo Alto Research Center.

Thacker, C. P., MacCreight, E., Lampson, B. W., F, S. R., and Boggs, D. R. (1982). *Alto: A Personal Computer*, chapter 33. McGraw-Hill.

Thagard, P. and Croft, D. (1999). *Scientific Discovery and Technological Innovation: Ulcers, Dinosaur Extinction, and the Programming Language Java*, pages 125–137. Springer.

The Free Software Foundation (2022). What is free software? www.gnu.org/philosophy/free-sw.en.html, Retrieved July 3, 2022.

Thomas, D. A. (1995). Travels with Smalltalk. web.archive.org/web/20130612055149/www.mojowire.com/TravelsWithSmalltalk/DaveThomas-TravelsWithSmalltalk.htm.

Tofte, M. and Talpin, J.-P. (1997). Region-based memory management. *Information and Computation*, 132(2):109–176.

Tomayko, J. E. (1988). Computers in spaceflight. Technical Report, NASA contractor report CR-182505, National Aeronautics and Space Administration, Scientific and Technical Information Division, Washington, DC, USA.

Tozzi, C. (2017). *For Fun and Profit: A History of the Free and Open Source Software Revolution*. MIT Press.

Tung, C. C. (1974). The 'personal computer': A fully programmable pocket calculator. *Hewlett-Packard Journal*, 25(9):2–20.

Turing, A. (1949). Checking a large routine. In *Report of a Conference on High Speed Automatic Calculating Machines*, pages 67–69, Cambridge. University Mathematical Laboratory.

Turner, F. (2010). *From Counterculture to Cyberculture*. University of Chicago Press.

Ungar, D. and Smith, R. B. (2007). Self. In *Proceedings of the third ACM SIGPLAN Conference on History of Programming Languages*, pages 9–1.

Univac (1957). Introducing a new language for automatic programming univac flow-matic. Computer History Museum, catalog no. 102646140.

Urquhart, A. (1988). Russell's zig-zag path to the ramified theory of types. *Russell: The Journal of Bertrand Russell Archives*, Johns Hopkins University Press, 8(1–2), summer/winter 1988, 82–91.

van Wijngaarden, A., Mailloux, B. J., Peck, J. E. L., and Koster, C. H. A. (1969). *Report on the Algorithmic Language Algol 68*, pages 80–218. Springer.

Vekris, P., Cosman, B., and Jhala, R. (2016). Refinement types for TypeScript. In Berger, E. (ed.), *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, pages 310–325, New York. ACM.

Victor, B. (2013). The future of programming. DBX Conference. https://worrydream.com/dbx/.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

von Neumann, J. and Goldstine, H. (1947). Planning and coding of problems for an electronic computing instrument, Part II, vol. 1. Technical Report, Institute for Advanced Study, Princeton.

Wadler, P. (2015). Propositions as types. *Communications of the ACM*, 58(12):75–84.

Wadler, P. and Findler, R. B. (2009). Well-typed programs can't be blamed. In Castagna, G. (ed.), *European Symposium on Programming*, pages 1–16. Springer.

Waldrop, M. M. (2001). *The Dream Machine: J.C.R. Licklider and the Revolution That Made Computing Personal*. Viking Penguin.

Ward, A., Rohrhuber, J., Olofsson, F., McLean, A., Griffiths, D., Collins, N., and Alexander, A. (2004). Live algorithm programming and a temporary organisation for its promotion. In Goriunova, O. and Shulgin, A. (eds.), *Proceedings of the README Software Art Conference*, volume 289, page 290.

Ware, W. H. (1966). *Future Computer Technology and Its Impact*. RAND Corporation.

Weinreb, D. and Moon, D. (1980). Flavors: Message passing in the LISP machine. Technical Report AIM-602, MIT.

Whitaker, W. A. (1993). Ada—the project: The DoD high order language working group. *SIGPLAN Notices*, 28(3):299–331.

Wilkes, M. (1985). *Memoirs of a Computer Pioneer*. MIT Press.

Wilkes, M. V., Wheeler, D. J., and Gill, S. (1951). *The Preparation of Programs for an Electronic Digital Computer: With Special Reference to the EDSAC and the Use of a Library of Subroutines*. Addison-Wesley Press.

Winberg, G. M. (1971). *The Psychology of Computer Programming*. Van Nostrand Reinhold.

Wirfs-Brock, A. (2013). Smalltalk at Tektronix. STIC'13. https://wirfs-brock.com/allen/files/tek/Tek-Smalltalk-history-slides.pdf.

Wirfs-Brock, A. (2020). The rise and fall of commercial Smalltalk. https://wirfs-brock.com/allen/posts/914.

Wirfs-Brock, R. and Wilkerson, B. (1989). Object-oriented design: A responsibility-driven approach. In Bosworth, G. (ed.), *Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications*, OOPSLA '89, page 71–75, New York. ACM.

Wirfs-Brock, R., Wilkerson, B., and Wiener, L. (1990). *Designing Object-Oriented Software*. Prentice-Hall, Inc.

Wirth, N. (1971). The programming language Pascal. *Acta Informatica*, 1(1):35–63.

Wirth, N. (1996). *Recollections about the Development of Pascal*, page 97–120, New York. ACM.

Wlaschin, S. (2018). *Domain Modeling Made Functional: Tackle Software Complexity with Domain-Driven Design and F#*. Pragmatic Bookshelf.

Wright, A. K. and Felleisen, M. (1994). A syntactic approach to type soundness. *Information and Computation*, 115(1):38–94.

Xi, H. and Pfenning, F. (1999). Dependent types in practical programming. In Appel, A and Aiken, A. (eds.), *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 214–227.

Yau, S. S. and Cheung, R. C. (1975). Design of self-checking software. In Shooman, M. L. and Yeh, R. T. (eds.), *Proceedings of the International Conference on Reliable Software*, pages 450–455, New York. ACM.

Yourdon, E. (1975). *Techniques of Program Structure and Design*. Prentice-Hall.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009

Yushchenko, Y. (2022). Pointers in programs on the computer MESM. For the European Virtual Computer Museum of the History of Information Technologies in Ukraine. www.icfcst.kiev.ua/MUSEUM/TXT/YuriYushchenko.pdf.

Zachary, G. P. (2018). *Endless Frontier: Vannevar Bush, Engineer of the American Century*. Simon and Schuster.

Zilles, S. N. (1973). Procedural encapsulation: A linguistic protection technique. In Graham, R. M. and Schroeder, M. D. (eds.), *Proceeding of ACM SIGPLAN - SIGOPS Interface Meeting on Programming Languages - Operating Systems*, page 142–146, New York. ACM.

Zmölnig, J. and Eckel, G. (2007). Live coding: An overview. In Jensen, K. K. (ed.), *Proceedings of the 2007 International Computer Music Conference, ICMC 2007, Copenhagen, Denmark, August 27–31, 2007*. Michigan Publishing.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 11 Nov 2026 at 12:08:12, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/9781009492379.009