

Bertrand Meyer

Long biography (500 words)

Bertrand Meyer is an influential software pioneer whose work bridging academic theory and industrial practice has redefined how developers reason about software correctness, reuse, and design. As a passionate advocate for simple, elegant, and mathematically sound software architectures, his contributions span programming language design, formal verification, requirements engineering, and pedagogy.

Education and Early Career

Meyer's intellectual trajectory is marked by an unusually broad educational foundation. He graduated from École Polytechnique in Paris and followed with a Master of Science in Computer Science at Stanford University and an engineering degree from Télécom Paris. Demonstrating his wide-ranging intellectual curiosity, he also earned a degree in Russian language and literature from the Sorbonne. He completed his Doctorate in Computer Science at the University of Nancy.

His early professional career included nine years as a research engineer and manager at Électricité de France (EDF) and several years on the faculty at the University of California, Santa Barbara. These dual experiences in industrial development and academic instruction deeply informed his lifelong quest to elevate software engineering from ad-hoc craft to rigorous scientific discipline.

Eiffel and Design by Contract

In 1985, Meyer founded Interactive Software Engineering (now Eiffel Software) in Santa Barbara, California, where he set out to design a language that natively supported the tenets of object-oriented programming. The result was **Eiffel**, a highly elegant language structured around his ground-breaking concept of **Design by Contract (DbC)**. Under DbC, software elements govern their interactions using formal assertions—preconditions, postconditions, and class invariants—guaranteeing correctness at runtime and providing a clean path toward mathematical verification.

These theoretical and practical insights culminated in his masterpiece *Object-Oriented Software Construction*, Prentice Hall (1st edition 1988, 2nd edition 1997). This text became a cornerstone of the software literature, widely considered the definitive guide to object technology.

Academic Leadership and Later Works

In 2001, Meyer accepted the Chair of Software Engineering at the prestigious ETH Zurich, taking over from the Turing-Award-winner Niklaus Wirth. At ETH he directed research for fifteen years and served as Head of the Computer Science Department. He championed the “inverted curriculum” for introductory programming, arguing that students should learn to use sophisticated libraries before writing code from scratch. This pedagogy led to his textbook *Touch of Class: Learning to Program Well Using Objects and Contracts*, Springer, 2009. He continued his academic activity at the Constructor Institute of Technology as Professor of Software Engineering and Provost.

Beyond programming languages, Meyer has made lasting contributions to requirements engineering, software litigation, and process methodology. His critical gaze on industry trends is reflected in books including Bertrand Meyer: *Agile! The Good, the Hype and the Ugly*, Springer, 2014, and his exhaustive treatise on requirements engineering *Handbook of Requirements and Business Analysis*, Springer, 2022.

A highly decorated scholar, Meyer is the recipient of the ACM Software System Award, the IEEE Harlan D. Mills Award, the Dahl-Nygaard Prize, a Fellow of both ACM and IFIP, a member of the French National Academy of Technologies and the recipient of two honorary doctorates.