1. Record Nr. UNINA9910855370303321

Autore	Meyer-Stabley Bertrand
Titolo	The French School of Programming / / edited by Bertrand Meyer
Pubbl/distr/stampa	Cham : , : Springer International Publishing : , : Imprint : Springer, , 2024
ISBN	9783031345180
Edizione	[1st ed. 2024.]
Descrizione fisica	1 online resource (451 pages)
Altri autori (Persone)	Meyer
Disciplina	005.1
Soggetti	Software engineering Computer science Programming languages (Electronic computers) Computer programming Machine theory Computer programs - Testing Software Engineering Theory of Computation Programming Language Programming Techniques Formal Languages and Automata Theory Software Testing
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di contenuto	The French School of Programming: A Personal View -- Part I: Software Engineering -- “Testing can be formal too”: 30 years later -- A Short Visit to Distributed Computing Where Simplicity is Considered a First Class Property -- Modeling: From CASE Tools to SLE and Machine Learning -- At the Confluence of Software Engineering and Human-Computer Interaction: a Personal Account -- Part II: Programming language mechanisms and type systems -- From Procedures, Objects, Actors, Components, Services, to Agents -- Semantics and syntax, between computer science and mathematics -- Some remarks about Dependent Type Theory -- Part III: Theory -- A Personal Historical Perspective on Abstract Interpretation -- Tracking Redexes in the

Lambda Calculus -- Confluence of terminating rewriting computations -- Part IV: Language design and programming methodology -- Programming with union, intersection, and negation types -- Right and wrong: ten choices in language design.

Sommario/riassunto

The French School of Programming is a collection of insightful discussions of programming and software engineering topics, by some of the most prestigious names of French computer science. The authors include several of the originators of such widely acclaimed inventions as abstract interpretation, the Caml, OCaml and Eiffel programming languages, the Coq proof assistant, agents and modern testing techniques. The book is divided into four parts: Software Engineering (A), Programming Language Mechanisms and Type Systems (B), Theory (C), and Language Design and Programming Methodology (D). They are preceded by a Foreword by Bertrand Meyer, the editor of the volume, a Preface by Jim Woodcock providing an outsider's appraisal of the French school's contribution, and an overview chapter by Gérard Berry, recalling his own intellectual journey. Chapter 2, by Marie-Claude Gaudel, presents a 30-year perspective on the evolution of testing starting with her own seminal work. In chapter 3, Michel Raynal covers distributed computing with an emphasis on simplicity. Chapter 4, by Jean-Marc Jézéquel, former director of IRISA, presents the evolution of modeling, from CASE tools to SLE and Machine Learning. Chapter 5, by Joëlle Coutaz, is a comprehensive review of the evolution of Human-Computer Interaction. In part B, chapter 6, by Jean-Pierre Briot, describes the sequence of abstractions that led to the concept of agent. Chapter 7, by Pierre-Louis Curien, is a personal account of a journey through fundamental concepts of semantics, syntax and types. In chapter 8, Thierry Coquand presents "some remarks on dependent type theory". Part C begins with Patrick Cousot's personal historical perspective on his well-known creation, abstract interpretation, in chapter 9. Chapter 10, by Jean-Jacques Lévy, is devoted to tracking redexes in the Lambda Calculus. The final chapter of that part, chapter 11 by Jean-Pierre Jouannaud, presents advances in rewriting systems, specifically the confluence of terminating rewriting computations. Part D contains two longer contributions. Chapter 12 is a review by Giuseppe Castagna of a broad range of programming topics relying on union, intersection and negation types. In the final chapter, Bertrand Meyer covers "ten choices in language design" for object-oriented programming, distinguishing between "right" and "wrong" resolutions of these issues and explaining the rationale behind Eiffel's decisions. This book will be of special interest to anyone with an interest in modern views of programming — on such topics as programming language design, the relationship between programming and type theory, object-oriented principles, distributed systems, testing techniques, rewriting systems, human-computer interaction, software verification… — and in the insights of a brilliant group of innovators in the field.