1. Record Nr. UNINA9910821368103321

Titolo	Static analysis of software [[electronic resource]] : the abstract interpretation / / edited by Jean-Louis Boulanger
Pubbl/distr/stampa	Hoboken, N.J., : Wiley, 2012
ISBN	1-118-60286-2 1-118-60295-1 1-118-60284-6 1-299-18778-1
Edizione	[1st ed.]
Descrizione fisica	1 online resource (347 p.)
Collana	ISTE
Altri autori (Persone)	BoulangerJean-Louis
Disciplina	005.1/4
Soggetti	Computer software - Testing Debugging in computer science Computer software - Quality control
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Description based upon print version of record.
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Cover; Title Page; Copyright Page; Table of Contents; Introduction; Chapter 1. Formal Techniques for Verification and Validation; 1.1. Introduction; 1.2. Realization of a software application; 1.3. Characteristics of a software application; 1.4. Realization cycle; 1.4.1. Cycle in V and other realization cycles; 1.4.2. Quality control (the impact of ISO standard 9001); 1.4.3. Verification and validation; 1.5. Techniques, methods and practices; 1.5.1. Static verification; 1.5.2. Dynamic verification; 1.5.3. Validation; 1.6. New issues with verification and validation; 1.7. Conclusion 1.8. BibliographyChapter 2. Airbus: Formal Verification in Avionics; 2.1. Industrial context; 2.1.1. Avionic systems; 2.1.2. A few examples; 2.1.3. Regulatory framework; 2.1.4. Avionic functions; 2.1.5. Development of avionics levels; 2.2. Two methods for formal verification; 2.2.1. General principle of program proof; 2.2.2. Static analysis by abstract interpretation; 2.2.3. Program proof by calculation of the weakest precondition; 2.3. Four formal verification tools; 2.3.1. Caveat; 2.3.2. Proof of the absence of run-time errors: Astree; 2.3.3. Stability and numerical precision: Fluctuat

2.3.4. Calculation of the worst case execution time: aiT (AbsInt GmbH)
2.4. Examples of industrial use; 2.4.1. Unitary proof (verification of low level requirements); 2.4.2. The calculation of worst case execution time; 2.4.3. Proof of the absence of run-time errors; 2.5. Bibliography; Chapter 3. Polyspace; 3.1. Overview; 3.2. Introduction to software quality and verification procedures; 3.3. Static analysis; 3.4. Dynamic tests; 3.5. Abstract interpretation; 3.6. Code verification; 3.7. Robustness verification or contextual verification; 3.7.1. Robustness verifications
3.7.2. Contextual verification3.8. Examples of Polyspace® results; 3.8.1. Example of safe code; 3.8.2. Example: dereferencing of a pointer outside its bounds; 3.8.3. Example: inter-procedural calls; 3.9. Carrying out a code verification with Polyspace; 3.10. Use of Polyspace® can improve the quality of embedded software; 3.10.1. Begin by establishing models and objectives for software quality; 3.10.2. Example of a software quality model with objectives; 3.10.3. Use of a subset of languages to satisfy coding rules; 3.10.4. Use of Polyspace® to reach software quality objectives
3.11. Carrying out certification with Polyspace®3.12. The creation of critical onboard software; 3.13. Concrete uses of Polyspace®; 3.13.1. Automobile: Cummins Engines improves the reliability of its motor's controllers; 3.13.2. Aerospace: EADS guarantees the reliability of satellite launches; 3.13.3. Medical devices: a code analysis leads to a recall of the device; 3.13.4. Other examples of the use of Polyspace®; 3.14. Conclusion; 3.15. Bibliography; Chapter 4. Software Robustness with Regards to Dysfunctional Values from Static Analysis; 4.1. Introduction; 4.2. Normative context
4.3. Elaboration of the proof of the robustness method

Sommario/riassunto

The existing literature currently available to students and researchers is very general, covering only the formal techniques of static analysis. This book presents real examples of the formal techniques called ""abstract interpretation"" currently being used in various industrial fields: railway, aeronautics, space, automotive, etc. The purpose of this book is to present students and researchers, in a single book, with the wealth of experience of people who are intrinsically involved in the realization and evaluation of software-based safety critical systems. As the authors are people curr