1. Record Nr. UNISA996465445703316

Autore	Rosenberg Doug
Titolo	Parallel Agile – faster delivery, fewer defects, lower cost [[electronic resource] /] / by Doug Rosenberg, Barry Boehm, Matt Stephens, Charles Suscheck, Shobha Rani Dhalipathi, Bo Wang
Pubbl/distr/stampa	Cham : , : Springer International Publishing : , : Imprint : Springer, , 2020
ISBN	3-030-30701-8
Edizione	[1st ed. 2020.]
Descrizione fisica	1 online resource (XIX, 221 p. 120 illus.)
Disciplina	005.1
Soggetti	Software engineering Management information systems Computer science Software Engineering/Programming and Operating Systems Management of Computing and Information Systems
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di contenuto	1. Parallel Agile Concepts -- 2. Inside Parallel Agile -- 3. CodeBots: From Domain Model to Executable Architecture -- 4. Parallel Agile by Example: CarmaCam -- 5. Taking the Scream Out of Scrum -- 6. Test Early, Test Often -- 7. Managing Parallelism: Faster Delivery, Fewer Defects, Lower Cost -- 8. Large-Scale Parallel Development -- 9. Parallel Agile for Machine Learning -- Appendix A. The Scream Guide -- Appendix B. Architecture Blueprints.
Sommario/riassunto	From the beginning of software time, people have wondered why it isn’t possible to accelerate software projects by simply adding staff. This is sometimes known as the “nine women can’t make a baby in one month” problem. The most famous treatise declaring this to be impossible is Fred Brooks’ 1975 book The Mythical Man-Month, in which he declares that “adding more programmers to a late software project makes it later,” and indeed this has proven largely true over the decades. Aided by a domain-driven code generator that quickly creates database and API code, Parallel Agile (PA) achieves significant schedule compression using parallelism: as many developers as necessary can

independently and concurrently develop the scenarios from initial prototype through production code. Projects can scale by elastic staffing, rather than by stretching schedules for larger development efforts. Schedule compression with a large team of developers working in parallel is analogous to hardware acceleration of compute problems using parallel CPUs. PA has some similarities with and differences from other Agile approaches. Like most Agile methods, PA "gets to code early" and uses feedback from executable software to drive requirements and design. PA uses technical prototyping as a risk-mitigation strategy, to help sanity-check requirements for feasibility, and to evaluate different technical architectures and technologies. Unlike many Agile methods, PA does not support "design by refactoring," and it doesn't drive designs from unit tests. Instead, PA uses a minimalist UML-based design approach (Agile/ICONIX) that starts out with a domain model to facilitate communication across the development team, and partitions the system along use case boundaries, which enables parallel development. Parallel Agile is fully compatible with the Incremental Commitment Spiral Model (ICSM), which involves concurrent effort of a systems engineering team, a development team, and a test team working alongside the developers. The authors have been researching and refining the PA process for several years on multiple test projects that have involved over 200 developers. The book’s example project details the design of one of these test projects, a crowdsourced traffic safety system.