SEI Series in Software Engineering

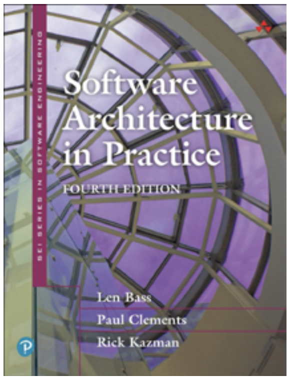
Software Architecture

FEBRUARY 2025

Carnegie Mellon University
Software Engineering Institute

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Software Architecture in Practice, Fourth Edition



Available at InformIT

Published: August 2021
ISBN: 9780136886099
Hardback, 464 pages

With 11 new chapters, *Software Architecture in Practice, Fourth Edition* remains the core book in the SEI's software architecture curriculum. The book explores what software architecture is, why it's important, and how to design, instantiate, analyze, evolve, and manage it in disciplined and effective ways.

The authors cover the entire software architecture lifecycle, presenting practical guidance, expert methods, and tested models for use in any project, no matter how complex. Offering insights on how to use architecture to optimize key quality attributes (e.g., performance, modifiability, security, availability, testability, usability, and deployability), this guide explains how to manage and refine existing architectures, transform them to solve new problems, and build reusable architectures that become strategic business assets.

Readers will learn how to use architecture to address accelerating growth in requirements, system size, and abstraction and to manage emergent quality attributes as systems are dynamically combined in new ways. Additional topics include

- how architecture influences (and is influenced by) technical environments, project lifecycles, business profiles, and individual practices
- leveraging proven patterns, interfaces, and practices for optimizing quality through architecture
- architecting for mobility, the cloud, machine learning, and quantum computing
- designing for increasingly crucial attributes, such as energy efficiency and safety
- scaling systems by discovering architecturally significant influences, using DevOps and deployment pipelines managing technical debt
- understanding architecture's role in the organization in order to deliver more value

Designing Software Architectures, Second Edition

Available at InformIT

Published: June 2024
ISBN: 9780138108021
Paperback, 336 pages

Created for any professional software engineer, *Designing Software Architectures, Second Edition* provides a practical, step-by-step approach for architecture design, with structured methods, reusable chunks of design knowledge, and rich case studies that demonstrate how to use the methods.

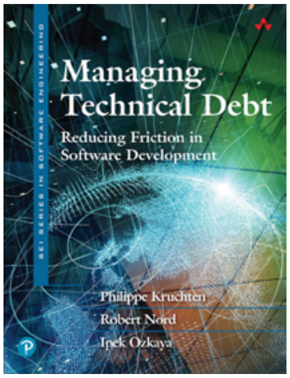
The Attribute-Driven Design method may not have changed since this book's first printing, but the industry has. This edition has new chapters on supporting business agility through API-centric design, deployability, cloud-based solutions, and technical debt in design.

The authors explore best practices that guide designers on how to make design decisions that are systematic, repeatable, and cost-effective while designing complex systems.

The new edition includes

- a clear explanation of the Attribute-Driven Design method
- new chapters focused on the technical environments and contexts of contemporary design
- two new case studies on the Hotel Pricing System and Digital Twin Platform
- discussion of current architecture topics like cloud computing, DevOps, and large-scale systems
- methods to make architecture design agile and achievable

Managing Technical Debt: Reducing Friction in Software Development


SEI SERIES IN SOFTWARE ENGINEERING
Managing Technical Debt
Reducing Friction in Software Development
Philippe Kruchten
Robert Nord
Ipek Ozkaya

Available at InformIT

Published: April 2019
ISBN: 9780135645888
Paperback, 272 pages

Technical debt occurs when earlier design or code decisions made in the context of budget or schedule constraints increasingly impede evolution and innovation as software systems mature. *Managing Technical Debt* explores integrated, empirically developed principles and practices that any software professional can use to gain control of technical debt in any software system.

Using real-life examples, the authors explain the forms of technical debt that afflict software-intensive systems, their root causes, and their impacts. They introduce proven approaches for identifying and assessing specific sources of technical debt, limiting new debt, and "paying off" debt over time. They describe how to establish managing technical debt as a core software engineering practice in an organization.

Readers will

- discover how technical debt damages manageability, quality, productivity, and morale—and what can be done about it
- clarify root causes of debt, including the linked roles of business goals, source code, architecture, testing, and infrastructure
- identify technical debt items and analyze their costs in order to prioritize action
- choose the right solution for each technical debt item: eliminate, reduce, or mitigate
- integrate software engineering practices that minimize new debt

Documenting Software Architectures: Views and Beyond, Second Edition

SEI SERIES IN SOFTWARE ENGINEERING
Documenting Software Architectures
Views and Beyond
SECOND EDITION
Paul Clements • Felix Bachmann • Len Bass
David Garlan • James Ivers • Reed Little
Paulo Merson • Robert Nord • Judith Stafford

Available at InformIT

Published: September 2010
ISBN: 0321552687
Hardback, 592 pages

Software architecture—the conceptual glue that holds every phase of a project together for its many stakeholders—is widely recognized as a critical element in modern software development. Without an architecture that is appropriate for the problem being solved, a project will stumble along or, most likely, fail. However, effectively communicating the architecture is as important as designing it; if the architecture is misunderstood, the project is unlikely to succeed.

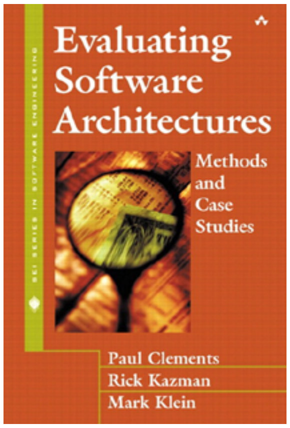
This book provides the most complete guidance, independent of language or notation, on how to capture an architecture in a commonly understandable form. How do software architects create *just enough* documentation to communicate with their stakeholders? The authors, first, help readers decide what information to document. Then, with guidelines and examples, they show readers how to express an architecture so that others can successfully build, use, and maintain a system from it.

The book features rules for sound documentation, the goals and strategies of documentation, architectural views and styles, documentation for software interfaces and software behavior, and templates for capturing and organizing information to generate a coherent package.

Topics include

- coverage of architectural styles such as service-oriented architectures, multitier architectures, and data models
- guidance for documentation in an Agile development environment
- documentation of rationale that reflects best industrial practices
- templates and documentation layouts, reflecting years of use and feedback
- reference guides for three important languages: UML, AADL, and SySML

Evaluating Software Architectures: Methods and Case Studies


SEI SERIES IN SOFTWARE ENGINEERING
Evaluating Software Architectures
Methods and Case Studies
Paul Clements
Rick Kazman
Mark Klein

Available at InformIT

Published: October 2001
ISBN: 9780201704822
Hardback, 368 pages

This book is a comprehensive, step-by-step guide to software architecture evaluation, describing specific methods that can quickly and inexpensively mitigate enormous risk in software projects. The methods are illustrated by both case studies and sample artifacts put into play during an evaluation: presentation outlines, scenarios, and final report templates—everything you need to evaluate an architecture in your own organization.

In particular, the book presents three important evaluation methods:

- Architecture Tradeoff Analysis Method® (ATAM®)
- Software Architecture Analysis Method (SAAM)
- Active Reviews for Intermediate Designs (ARID)

Detailed case studies demonstrate the value and practical application of these methods to real-world systems, and sidebars throughout the book provide interesting background and hands-on tips from the trenches.

All software engineers should know how to carry out software architecture evaluations. *Evaluating Software Architectures* is the chance to get up to speed quickly by learning from the experience of others.

® Architecture Tradeoff Analysis Method and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Related Web Sites

sei.cmu.edu/architecture

informit.com

For Course Registration

sei.cmu.edu/education-outreach/courses

About the SEI

Always focused on the future, the Software Engineering Institute (SEI) advances software as a strategic advantage for national security. We lead research and direct transition of software engineering, cybersecurity, and artificial intelligence technologies at the intersection of academia, industry, and government. We serve the nation as a federally funded research and development center (FFRDC) sponsored by the U.S. Department of Defense (DoD) and are based at Carnegie Mellon University, a global research university annually rated among the best for its programs in computer science and engineering.

Contact Us

CARNEGIE MELLON UNIVERSITY
SOFTWARE ENGINEERING INSTITUTE
4500 FIFTH AVENUE; PITTSBURGH, PA 15213-2612

sei.cmu.edu
412.268.5800 | 888.201.4479
info@sei.cmu.edu

Copyright 2025 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific entity, product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute nor of Carnegie Mellon University—Software Engineering Institute by any such named or represented entity.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. Requests for permission for non-licensed uses should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM25-0236