

Preface

The Software Engineering Handbook

A Judgment and Decision-Making Approach

James C. Davis

Front matter of the full book · Edition 1 · 2026

Preface

Once upon a time, engineering was taught via apprenticeship. A novice worked alongside experienced practitioners, first observing their decisions and then making increasingly consequential decisions under supervision. The apprentice learned the field's formal knowledge, but also something harder to write down: which questions to ask, which details matter, when a familiar solution applies, when it does not, and how to make a defensible choice when no answer is obviously correct. Modern engineering education cannot reproduce that arrangement at scale. We teach principles, methods, tools, and examples instead. These are valuable, but they can leave students knowing about engineering without yet knowing how an engineer thinks.

Generative AI makes that gap harder to leave for later. For a long time, industry could complete much of the apprenticeship. New engineers spent years implementing, debugging, testing, reviewing, and maintaining systems while learning from more experienced practitioners. Much of their professional judgment developed through doing that work. Increasingly capable AI can now perform precisely this routine work. Students still need the foundations of software construction, but we can no longer assume that years spent practicing them in industry will supply the judgment that education leaves implicit. As organizations delegate more routine work to AI, fresh graduates who cannot yet exercise judgment over that work may struggle to contribute enough value to begin the apprenticeship at all. They may be asked to direct and evaluate machine-generated work before they have accumulated the experience from which such judgment once emerged.¹

Welcome to your apprenticeship. This book teaches software engineering through the judgments its practitioners must make. Why choose one development process rather than another? What should we promise to build? Which distinctions belong in a specification, and which choices should remain open? Where should an architectural boundary go? When is additional analysis worth its cost? This book teaches you to recognize these decisions, identify the alternatives and consequences, and exercise engineering judgment. As implementation becomes cheaper, making and defending such decisions becomes more rather than less important. AI did not create these software-engineering questions. It has made them harder to postpone.

This handbook is also a practical companion to [Model-Based Agentic Engineering](#) (MAGE). MAGE argues that engineering with capable agents depends on making consequential knowledge explicit, important obligations enforceable, and recurring judgment durable. Applying those ideas requires knowing what is consequential in the first place. An engineer must decide what should be modeled, which obligations deserve enforcement, what can safely remain free, and when experience should become a rule that future work

¹**Small butterflies.** For many years, we could graduate caterpillars and let industry shape them into butterflies. Now we need to graduate small butterflies that can grow into big ones. Graduates need enough engineering judgment to contribute from the start, with professional experience deepening that judgment over time.

inherits. MAGE provides a theory for engineering with increasingly capable agents. This handbook develops the judgment needed to put that theory into practice.

So the chapters that follow are deliberately organized around decisions. Requirements engineering asks what we should promise. Specification asks what exactly that promise means. Architecture asks how competing obligations can coexist in one system. Design asks how the resulting parts should actually work. The same pattern continues throughout software engineering: identify the engineering problem, understand the choices, reason about their consequences, make a decision, and learn when reality tells you to reconsider it. No handbook can substitute for working beside excellent engineers. But we can no longer leave all of their most important lessons for later. This handbook makes some of those lessons explicit, so that experience can deepen engineering judgment rather than being expected to create it from scratch. That is the apprenticeship this handbook offers.