The Effective Software Development Series

The **Effective Software Development Series** (ESDS) provides expert advice on all aspects of modern software development. Books in the series are well-written, technically sound, of lasting value, and tractable length. Each describes the critical things the experts almost always do — or almost always avoid doing — to produce outstanding software.

Scott Meyers (author of the *Effective C++* books and CD) conceived of the ESDS and acts as its consulting editor. Authors accepted into the series work with Meyers and with Addison Wesley's professional editorial staff to create essential reading for software developers of every stripe.

Overview

The series summary above is filled with buzzwords. This is what they mean:

- **Expert Advice** — Effective software development calls for knowing what to do, how to do it, and why to do it that way. Books in the ESDS address all these questions, but the focus is on *advice*. Series authors expose the tricks of the trade: unlikely-looking things that work, likely-looking things that don't, and critical insights that explain how to tell one from the other. Reading an ESDS book is like attending a master workshop with an expert in the field.
- **All Aspects** — Most book series focus on a particular area of software development, e.g., coding in Java or programming for Windows. The ESDS has a broader vision. Requirements, design, coding, QA, maintenance, user interfaces, security, performance, etc., they're all valid topics. If professional software developers deal with it, the series covers it.
- **Well-written** — The series places great emphasis on readability. Prose is clear, direct, and straightforward. Names and examples are well-chosen. The technical material in the books may be challenging, but the explanations are not.
- **Technically Sound** — Technical material in ESDS books is based on experience, not hearsay. It's verified through detailed technical reviews by outside experts. Where technologies are evolving, ESDS authors explain how to produce systems that both work now and are likely to continue working in the future.
- **Lasting Value** — The information in ESDS books is rarely introductory, but it is always fundamental. That makes it less susceptible to changes in technology. In accord with this increased lifespan, page layouts are designed with ongoing reference in mind. Indices and cross references are given special attention. An online search engine makes it possible to perform full-text searches through all ESDS books.
- **Tractable Length** — Both authors and readers are best served by books that aren't too long. Shorter books are more likely to be read in their entirety, thus increasing the chances that readers get the "full story" the authors want to convey. Shorter books can be completed sooner, and it's easier to ensure the quality of shorter books. Books in the Effective Software Development Series generally run no more than 300-400 pages.
- **Outstanding Software** — Too much software is best described as "serviceable." Users can get it to do what they want it to do, and software developers can enhance it in modest ways, but neither users nor software developers find working with it particularly rewarding. Outstanding software does what it does so well, users *enjoy* working with it. Software developers, too, are enthusiastic, because the software is easy to modify, easy to extend, and elegant in how it accomplishes its tasks. Books in the ESDS explain how to write *outstanding* software.

Example Topics

The Effective Software Development Series embraces all aspects of software development. Topics well suited to the ESDS include (but are not limited to) these:

- Use of particular programming languages (e.g., C++, Java, C#, Visual Basic, Python, etc.).
- Programming for particular platforms or platform types (e.g., Unix, .NET, PalmOS, embedded systems, handheld or wireless devices, etc.).
- Programming for particular application types (e.g., web services, games, real-time systems, etc.).
- XML and associated technologies (e.g., XSLT, XPath, etc.).
- Improving performance (e.g., in computations, in database queries, in network communications, etc.).
- Managing memory (e.g., to improve performance, to facilitate debugging, to avoid leaks, etc.).
- Application of development tools and methodologies (e.g., UML, XP, etc.).
- User interface design and implementation.
- Development of concurrent systems, including parallel, multithreaded, and distributed systems.
- Building secure systems.
- Library design and development (both language-dependent and language-independent).
- Maintaining and modifying existing systems (including refactoring, etc.).
- Quality assurance (including code reviews and walkthroughs, testing strategies and methodologies, design for testability etc.).
- Patterns and architectures.
- Debugging techniques.
- Internationalization and cultural adaptation.
- Specification, development, and use of databases and query languages.
- Requirements analysis.
- Managing the development of software systems (including “peopleware” issues, creating and meeting schedules, etc.).

Style

Scott Meyers' books exemplify the pragmatic, advice-driven nature of books in the ESDS, but publications in the Series need not be organized or written in Meyers' style, nor must they have “Effective” in the title. Different authors and different topics lend themselves to different ways of conveying meaningful guidance. The ESDS welcomes all approaches that give software developers the advice they need.