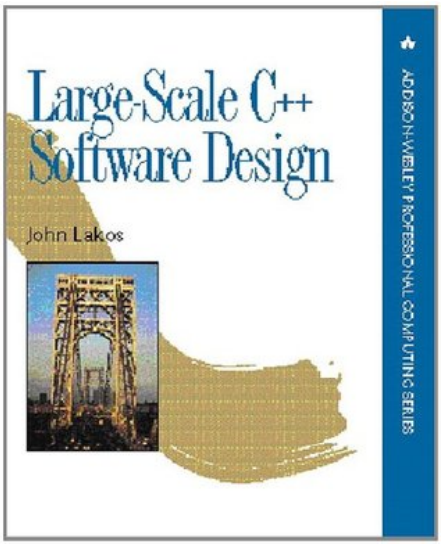

Large-Scale C++ Software Design

John S. Lakos

Large-Scale C++ Software Design

John S. Lakos

Large-Scale C++ Software Design John S. Lakos
This is the definitive book for all C++ software professionals involved in large development efforts such as databases, operating systems, compilers, and frameworks. It is the first C++ book that actually demonstrates how to design large systems, and one of the few books on object-oriented design specifically geared to practical aspects of the C++ programming language. In this book, Lakos explains the process of decomposing large systems into physical (not inheritance) hierarchies of smaller, more manageable components. Such systems with their acyclic physical dependencies are fundamentally easier and more economical to maintain, test, and reuse than tightly interdependent systems. In addition to explaining the motivation for following good physical as well as logical design practices, Lakos provides you with a catalog of specific techniques designed to eliminate cyclic, compile-time, and link-time (physical) dependencies. He then extends these concepts from large to very large systems. The book concludes with a comprehensive top-down approach to the logical design of individual components. Appendices include a valuable design pattern Protocol Hierarchy designed to avoid fat inte

Large-Scale C++ Software Design Details

Date	:	Published July 20th 1996 by Addison-Wesley Professional
ISBN	:	9780201633627
Author	:	John S. Lakos
Format	:	Paperback 896 pages
Genre	:	Computer Science, Programming, Software, Computers, Science

From Reader Review Large-Scale C++ Software Design for online ebook

Timothy Culp says

A classic text on modular design for large systems and the problems you run into when scaling to a million lines of code.

Ken says

Worth reading to discover the phrase "Escalate policy, demote infrastructure". Otherwise chapter 5 is interesting and the rest is eh.

Max Lybbert says

I believe I had picked this book up years ago, and dismissed it because the material is a little dated. However, after seeing https://www.youtube.com/watch?v=ASPj9... , I thought I'd give it another look.

And, yes, the book still is dated. There are several examples about iterators, but they're all pre-STL style iterators. The STL isn't mentioned until the last few chapters of the book. Some of the advice is explained as workarounds to CFront's implementation of specific language features. The book has several charts of performance data, but the systems in question are now 20 years old.

Even so, there is a lot to learn from here. I believe it's held up better than Design Patterns: Elements of Reusable Object-Oriented Software. Good design is timeless. And even where this book moves into design as influenced by the real world, it shows a great way to think. If you want to know the impact of a design decision, you could try to benchmark it as relentlessly as Lakos.

So, I would highly recommend watching Lakos' talk about hierarchical reuse. If you want to learn more, this book is the obvious next step.

Dax says

I wish this was required reading at my company. Instead of two-hour compile times every time you change a constant, this book shows you how to keep your codebase lean and mean.

Dave Peticolas says

A technical description of design problems and solutions for large C++ projects. In addition to logical design (functions, classes, etc.), this book focuses on physical design (files, directories, etc.) as an important aspect

of large software projects. Although C++ is used throughout, many, but not all, of the concepts apply to other environments. The prose of this manual is deadly boring, even for a technical work.