
This is a collection of papers on design methodology. It includes three articles with Christopher Alexander as at least one author. My favorite paper in the collection (I’ve not read them all) is "Planning Problems are Wicked Problems" by Horse Rittel and Melvin Webber. We should all understand the meaning of "Wicked Problems" -- I think if you look, you’ll find "Wicked" was a term current at the time in describing what are now called "Chaotic" systems. Herbert Simon’s paper "The Structure of Ill-structured Problems" is also included.


Category: Patterns


This is the first of Alexander’s books I knew, and seems to be from his PhD thesis. Here he describes a view of architecture that resonates with many software designers. The description of what he’s trying to do seems particularly useful and certainly changed the way I thought when I encountered this book in grad school. Later works define a different approach to design, Patterns, but the philosophy here, in the Preface to the Paperback Edition, Part I, and the epilogue to Part II, is very important reading for designers. Take note, too, of the way the diagrams in Appendix I compose -- These are the original "sketches" that are so important in Patterns, and differ significantly from those typically seen in current software patterns. These sketches could not usefully be done in UML -- (And, I would argue, neither can those that usually are.)

Category: Patterns


Category: Patterns


Provides some good design concepts but the structure is sometimes confusing.


This book illustrates the use of C++ to build matrix-like classes to solve scientific and numerical problems. The book shows many advanced idioms that use C++ parameterized types. Category: Language.


Original description of Class-Responsibilities-Collaborations (CRC Cards) technique.


Summary of Booch’s notation


Summary of Booch’s notation


Often cited work on object-oriented design, particularly for the notation. This book describes the Booch methodology, which is one of the more popular object oriented analysis and design methods. Category: Techniques.


This is one of the best books on object-oriented design. Mr Booch thoroughly covers the ins and outs of it. If you read nothing else, read this book


This book is filled with practical rules and guidelines for object oriented project management. Booch talks a lot about the use of iterative and incremental software development to minimize the risks in a software development project. Category: Management.


This book discusses architecture patterns (such as Pipes and Filters, Blackboard, and Model-View-Controller) and design patterns (such as Proxy and Publisher-Subscriber). This book extends some of the original design pattern work contained in the book by Erich Gamma, et. al. Category: Patterns


C++ professionals from a variety of backgrounds recommend their favorite "must have" C++ books from all publishers.


This book contains that rare and useful information--examples of bad code with analyses. Like the original Elements of Programming Style, this book is quite helpful because it shows what not to do


This book presents many issues in the creation of reusable C++ class libraries: correct use of multiple inheritance, source file organization, error handling, initialization of objects, and other technical C++ topics. Category: Language.


An introduction to object-orientation. It discusses how analysis can be improved through the use of object-oriented techniques describes an object-oriented analysis method, and provides some thoughts on how to get started.

A follow on to the analysis book. It discusses how design can be improved through the use of object-oriented techniques, describes an object-oriented design method, and provides some thoughts on how to get started.


The third book in the series. It completes the discussion of Coad and Yourdon’s object-oriented approach.


This is an excellent practical manager’s guide to the introduction of object oriented technology. This book gives a bunch of practical advice to managers on how to select the right languages and tools, how to do iterative and incremental project scheduling, how to bring in training, and how to measure your success. Category: Management.


This book describes the HP Fusion methodology, which was originally developed at HP Labs. It is a simple methodology that is based on ideas taken from the Booch, Shlaer-Mellor, and Jacobson OOSE methodologies. Said to be a next generation of object-oriented methods incorporating the best features of other object-oriented methods, including "use cases" and CRC Cards. Category: Techniques.


Lots and lots of useful techniques. However, it can be fairly dense at points, it isn’t well organized, and some of the examples run counter to rules in this and other books.


This book is a great introduction to Java for C and C++ programmers. It gives excellent examples of good Java usage. The book is much less negative about other programming languages than most of the books written by people from Sun Microsystems. Category: Language.


This book has been supplanted by the current working paper for the draft ANSI C++ standard. It reflects the state of the language at the start of the standardization process a few years ago. Neither this book (abbreviated as the ARM) nor the working paper are light bedtime reading. However, it’s essential to have one of them close at hand if you hope to make optimum use of C++. If you can possibly stay awake, read one all the way through. If you can get a copy of the working paper, that is preferable to using the ARM.


This book is the best concise reference book on the Java programming language. It contains a complete reference to the Java 1.1 standard library classes. The example programs are very useful -- they illustrate many of the common uses of Java in building real applications. Category: Language.
   A collection of essays on Software development. Good discussion of what Patterns are and how they
   relate to software design, also a good discussion of some issues in software design, particularly how they
   relate to people. I particularly recommend his final essay "Money Through Innovation Reconsidered"
   which looks at development in the real world, where "Worse is Better".

34. Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable
   This is the famous "Gang of Four" design patterns book. An excellent source of design patterns. Useful
   for application architects. It contains descriptions of 23 design patterns and shows examples of their use in
   many different kinds of applications. Category: Patterns

35. Goldberg, Adele and Kenneth S. Rubin, Succeeding with Objects: Decision Frameworks for Project
   This book is a good overview of object oriented project management issues, with some discussion of a

36. Goldstein, Neil and Jeff Alger, Developing Object-Oriented Software for the Macintosh, Addison-Wesley,
   This book provides another excellent perspective on object-oriented design. It explodes some common
   myths. Don’t let the title fool you--it’s really not that specific to the Macintosh.


   1987.
   Description of Harel’s state transition diagrams, developed for use in large systems, especially responsive
   systems (real-time systems).

39. Harmon, P. and D. Taylor, Objects in Action: Commercial Applications of Object-Oriented Technologies,
   Results of real-world experience with object-oriented projects as submitted to OMG. The book is organized
   into sections for industry segments. Reasonable material but almost all the projects are successful and very
   few problems are cited in the lessons learned section.

40. IBM Object Oriented Technology Center, Developing Object-Oriented Software: an Experience-Based
    This is a very detailed handbook of object oriented techniques from IBM’s Object Oriented Technology
    Center. They give good descriptions of the myriad software work products that have been used in various
    parts of IBM, without coming out as a firm advocate of any one analysis and design methodology. Category: Techniques.

    Standard for Information Systems--Programming Language C++.
    This is essentially the same as the ANSI X3/J16 working paper for programming language C++.

    C.
    This is essentially the same as ANSI C, X3J11/88-159

43. Jacobson, Ivar, Magnus Christerson, Patrik Jonsson, and Gunnar Overgaard, Object Oriented Software
    ISBN 0-201-55435-0
    The original textbook for scenario, or use-case, based design. Category: Techniques.

   Good treatment of object database issues, features, and implementations. Somewhat academic in places.


   Brief article outlining some complexity metrics for object-oriented projects.


   This book discusses large scale software development issues in C++: how to design software to minimize recompilation and relinking time when changes are made in the middle of a design, building layers of classes to reduce circular dependencies in compilation and testing, and component design principles. Lakos summaries his experiences in using C++ into a well-defined set of design rules, guidelines, and principles. Category: Language.


   Category: Languages, C++


   It's got lots of good observations by someone who's been through the wringer a few times.


   This book gives a good introduction to the problems posed by the concurrent programming facilities that are built into the Java programming language. Doug explains many important tips to follow to write good multithreaded Java applications. Category: Language.


   This book describes general object oriented modeling techniques, with some cross-references to C++ implementation of object oriented models. The book uses UML notation throughout, but it doesn't follow any of the "standard" object oriented analysis and design methodologies. Category: Techniques.


   There are other papers on this as well. Outlines the "Law of Demeter." which is a set of rules for structuring objects so that the resulting implementations are easy to maintain and support re-use. All are available on the Internet.


   A gentler tutorial for C++ than *The C++ Programming Language*, 2nd Edition. The second edition of Stroustrup’s book is much better organized and is more approachable, lessening the need for this book. Also, this book does not cover classes until the later chapters.


   Real world experiences with implementing systems using an object-oriented approach.

Another object-oriented approach. Seems to spend a lot of time in terminology. Relates OO to Information Engineering approach.


The original source for information on the contracting approach to design. The contracting approach provides precision of specifications and a way to trace through from analysis to design to construction. The book uses the Eiffel language for examples.


This is an advanced-level follow-on to Scott Meyers’ Effective C++ book. It covers several advanced features of C++, including exception handling, efficiency issues for multiple inheritance and runtime type identification, and abstract classes. Category: Language.


This is the most important intermediate-level C++ book for anyone who plans to do object oriented programming in C++. Scott covers all of the important topics in C++ as a series of 50 short guidelines. Category: Language.


This book presents 50 good rules for designing and writing with C++. The intent of Meyers is not unlike that of Taligent’s Guide to Designing Programs.


Many useful tips and techniques for using C++. Gives more detailed examples of some of the techniques discussed in this document.


This is a useful intermediate C++ book that illustrates many important C++ class design issues. Rob Murray gives many examples of good library class design throughout. Category: Language.


This book describes the Standard Template Library (STL), a set of C++ container classes that have been added to the ANSI C++ standard. This book shows how to use the STL classes and functions, and it also shows how to extend them to new container types and algorithms. Category: Language.


If you’re a designer and you’ve not read this, you probably should. To me, it’s all about the Quality Without A Name, others see is as about motorcycles and/or mental illness (depends on what your focus is). If you’d like to see an amazing collection of reactions to the book head for www.amazon.com and look it up there -- they have reviews/comments from many people -- some love it, some hate it. I suspect it’s related to personality type, and my prejudice is the designer type is generally in the "I love it" camp.

   This book discusses a series of 61 object-oriented design heuristics. It explains the most important design tradeoffs to consider when doing object-oriented design. Category: Techniques.


   A difficult book to read but thorough. Rumbaugh’s model is widely supported with tools. A criticism of the method is that the model components are not well integrated. This book describes Object Modeling Technique (OMT), an object-oriented analysis and design methodology first developed at General Electric. This method is supported by many CASE tools and is the subject of many commercial training courses. Category: Techniques.


   This book is a compendium of articles by one of the original creators of the OMT methodology. Rumbaugh collects here a series of articles that he wrote on OMT for the Journal of Object-Oriented Programming, which describe some of the newer items that have been added to the OMT methodology, such as use cases. Category: Techniques.


   This book describes the Real-Time Object-Oriented Modeling (ROOM) methodology, which is supported by the commercial CASE tool ObjectTime. Category: Techniques.


   Well presented and easy to read. Material will be easily understood by those with a data modeling background.


   Well presented, easy to read. This small book provides a methodology from initial discussion to the beginning of physical design.


   A paper giving an overview of a transformational method of physical design. Physical design is to be the topic of the third book in Schlaer/Mellor’s series.


   In addition to being the best introduction and reference to the language (excepting the ANSI draft specification), it includes excellent chapters on object-oriented design and management of object-oriented projects. These chapters are full of experience from the man who has seen more C++ projects than anyone else.

   Of special note in the 3rd edition are chapters on Design Using C++: this part presents C++ and the techniques it supports in the larger picture of software development. The focus is on design and the effective realization of design in terms of language constructs. Category: languages.


   Overview material, reasonable for concepts
   Presents a full analysis and design methodology using the contracting approach.
   Includes bibliographical references (p. 241-250) and index.
   This is a timeless book on the activities of programming. It is useful to read this periodically. Note especially the parable of the Coke machine. Note also the parable of the special I/O system. Some choice bits about what drives programmers to worry about efficiency. Category: General
   This book is an easy introduction to the CRC card process. It works through one example from early requirements to C++ code, with lots of false starts and wrong turns. It is very easy to read, but it also explains a lot of subtle points in the use of CRC cards. Category: Techniques.
   Well presented, easy to read book on object-oriented design. This is the original textbook for responsibility driven design. This book is on the Responsibility Driven Design (RDD) methodology. This method focuses on assigning responsibilities to classes before creating a complex data model. Category: Techniques.