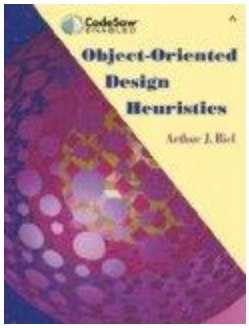
Object-Oriented Design Heuristics by Arthur J. Riel



Challenge What You Know About Ooa/D

Here is the first object-oriented development book to provide specific experience-based guidelines to help developers make the right design decisions. This book offers the next step for readers that know the basics of object-oriented development and now need to know if they are doing it right and making the right choices.

My Personal Review:
I have been studying the object oriented methodology for some time now. I felt that I had a good understanding of what OOP was all about. I have studied OOA/D and design patterns from numerous sources. All of my sense for OOA/D knowledge changed completely when I read this book. This book really showed me that I was stuck somewhere in the middle of the paradigm shift between action oriented programming (aka procedural programming) and object oriented programming. After reading this, I feel like my knowledge in OOA/D has truly advanced to the next level.

Are you the type of person that knows what OOP is? I mean, if youve studied up on OOP then you are probably aware of what an abstract class is. You know what interfaces, inheritance, polymorphism, information hiding (...etc) are. You may have a sense in when you should use inheritance and when you should use containment. You probably follow certain OOP practices like keeping all of your variables private, hiding secrets from other objects (information hiding). This may all make sense to you but are you also the type of person that just never feels comfortable about your designs? Do you look at your classes and just get a sense that something doesnt seem right, yet you just cant figure out what it is even if your software system is running fine? I am willing to wager that you are in the middle of a paradigm shift. You are probably taking the route that a lot of developers take when they shift from thinking in a procedural fashion (action-oriented design) into object oriented design. There is nothing wrong with this, but if youre like a lot of developers you will have a long hard journey utilizing a lot of experience before you really make that shift. This

book is an essential tool that will help you make that shift a LOT faster. After reading this book you will see why you felt your designs weret quite right.

One of the first topics that really hit home for me was when the author Arthur Riel talks about God classes in chapter 3. God classes are classes that have too much implementation in them. Most of the complexity of a piece of software resides in these classes. They are the all-knowing classes that delegate messages between the much smaller, less complex classes. Signs of God classes are classes that have words in their name such as Manager or System in them. This one hit home because there are numerous classes in the software im working on now with the name Manager in them. For example one of our classes is called the BiDirectional_Dataflow_Manager. This is definitely a God class through and through. While I was reading about the disadvantages of these types of classes I couldnt help but agree with everything Arthur was saying. I began to see the light already and I was just on chapter 3. There are 59 other Heuristics, all equally important in this book.

Most books that teach OOA/D seem to really only teach the definition of OOA/D and perhaps clue you in to the whole idea. You learn the terminology well and you see a few examples (Im sure youve seen an animal hierarchy a time or two), but you dont really gain a solid understanding in how you actually think in objects. This book will bridge that gap. This is the best book ive read by far on OOA/D. This book will apply to you no matter what your skill level is in OOA/D, unless youre a complete beginner then you might find yourself a little bit lost. If you are brand new to OOA/D then you should probably read a short book on OOP, just to gain the basic concepts first. Object Oriented Thought Process might be a good start as its short and sweet, then you should move on to this book. If you are advanced then you may know a lot of this information, but this book will probably help tweak your OOA/D skills; helping you become an even more solid developer. But for you guys and gals out there that know what OOP is and read a few books on it, but still dont feel quite right about your designs, this book is essential. You guys out there are the sweet spot for a book like this. Thats how I was. Now I feel so much better, I feel like ive gained more knowledge in OOA/D with this book then all other books on OOA/D and OOP that ive read combined - and then some.

Arthur Riel is a very talented programmer and author. He is able to communicate ideas to you that are sure to hit home, as if hes right there with you and understands your problems in OOA/D. This book is densly packed. Not including the bibliography and index this book is a mere 367 pages. Even more, if you dont include the example code at the end of the book (all C++ code) this book is only 243 pages. The real meat of this book is in the first 9 chapters (where he talks about all of the heuristics), which totals 182 pages. After that he talks about topics such as handling memory leaks and such. Most of the dim lights will shine brightly after a mere 182 pages! This may sound too good to be true, but as I said earlier Arthur is

VERY talented in communicating his ideas. You just have to read this book very carefully, dont skim! Because its so dense, it may take a couple of passes before you really get the idea but once you understand it you will surely belt out a resounding AH HA!. This book is 10 years old at the time of this review, but the information inside is far from being outdated.

To conclude this lengthy review (sorry about that) I would like to say that I give this book my highest recommendation. In fact, this may be the best book on software development that ive ever read! This book has influenced my software development more then any other book ive read and thats a fact. This is truly a rare gem. The only downside (not this books fault) is that its become a bit harder to work on the software that im currently working with because I now see where all of the pitfalls are. My co-workers think im just being anal about design now, but you dont have to be like them. Step up, become the best software developer that you can be. Just read this book and you will take a giant leap forward in your OOA/D understanding, especially if youre stuck in a paradigm shift like I was. Thank you very much Arthur!

4 OOPDTool OO design heuristics, design patterns and anti-patterns represent ways of capturing OO design expertise. Current design tools do not provide support to the reuse of this kind of knowledge. Therefore, our goals are twofold: First, to support design expertise reuse.Â These facts are stated in predicates corresponding to constructions defined by a metamodel for object oriented software. This metamodel was defined based on UML semantics metamodel [1] and on other works in this field [7], [11]. This metamodel defines the entities and relationships that are relevant to design patterns and anti- patterns identification, including not only structural elements, but also dynamic elements such as object instantiation and method calls, for instance. 2 Design Heuristics Object-Oriented Design Heuristics by Arthur Riel, Addison-Wesley, 1996. 3 Goal Insights into oo design improvement. More than sixty guidelines are languageindependent and allow one to rate the integrity of a software design. The heuristics are not written as hard and fast rules; they are meant to serve as warning mechanisms which allow the flexibility of ignoring the heuristic as necessary. 4 Classes and Objects: The Building Blocks of the Object-Oriented Paradigm. 5 Hidding Data Heuristic #2.1 All data should be hidden within its class.