Re: Recommended books on Top Down Design

Source: http://coding.derkeiler.com/Archive/General/comp.programming/2004−11/0420.html

---

From: Phlip (*phlip_cpp_at_yahoo.com*)
Date: 11/09/04

Date: Tue, 09 Nov 2004 14:30:51 GMT

Mike Pro wrote:

> *I've been practicing Pascal at college for the last 3 months now, I was*
> *wondering if anyone could suggest some good books or url's regarding TDD*
or
> *screen layout design...*

"Top down design" still happens, but books these days don't promote it. You need to learn an object−oriented language, such as Delphi or Kylix. Then you need to learn "iterative design", which means patterns of design growth that allow you to frequently finish a deliverable version. Studies have shown that the longer a project goes without delivering a useful application, the less likely it ever will release one.

Then you need to read all these books. They introduce some topics they probably are not teaching you in school:

* Design Patterns: elements of reusable object−oriented software by Gamma, Johnson, Helm, & Vlissides
* Refactoring: Improving the Design of Existing Code by Martin Fowler
* Refactoring to Patterns by Joshua Kerievsky
* Smalltalk Best Practice Patterns by Kent Beck
* Domain Driven Design: Tackling Complexity in the Heart of Software by Eric Evans
* The Art of Computer Programming by Knuth

Books about the culture of programming around the source code:

* Lean Software Development: An Agile Toolkit for Software Development Managers by Mary Poppendieck and Tom Poppendieck
* Extreme Programming eXplained: Embrace Change, 2nd Edition, by Kent Beck
* Agile Development: Principles Practices and Patterns by Robert C. Martin

* AntiPatterns: refactoring software, architectures, and projects in crisis by Brown, Malveau, McCormick & Mowbray
* The Pragmatic Programmer: From Journeyman to Master by Andy Hunt & Dave Thomas
* Rapid Development: Taming Wild Software Schedules by Steve McConnell

How to preventing long open−ended debugging sessions:

* Test−Driven Development: By Example by Beck
* Code Complete 2nd Edition by Steve McConnell
* Test Driven Development: A Practical Guide, by Dave Astels
* How to Break Software: A Practical Guide to Testing by James A. Whittaker
* Working Effectively with Legacy Code by Mike Feathers

The TDD (Test−Driven Development) books are especially noteworthy because, as you start writing programs larger than a couple modules, you will discover that most programmers spend most of their time debugging. The solution to this connundrum is so simple it shouldn't need a book, but our industry has a lot of "antipatterns" to overcome. The solution is to write tests for every kind of behavior you need, to only add code when you can get a test to fail, and to use Undo if a test fails unexpectedly (and you don't feel like applying a light amount of debugging). Replacing our advanced interactive debuggers with an Undo button helps code's behavior never depart from a known state. Prevention is better than a cure.

The Refactoring books are significant because, no matter how smart you are, you will always write code whose design could be improved after you read it. You could fight this effect, by planning on paper for a long time, or you can leverage it. Refactoring fits TDD like a glove.

Smalltalk (and Python & Ruby) is a strongly typed dynamic language. C++ is an almost−strongly typed static language that introduces generics to cover some dynamic typing abilities. Curiously, the higher level your code, the more dynamic typing you need. Dynamic typing means that foo.bar() will compile and execute no matter what type 'foo' is, so long as it has a method 'bar()'. This works as if all objects inherited a common interface, Object, that magically declared every possible method.

Curiously, the big software vendors keep pushing statically typed languages, Java & C#, while high−level code, such as GUI code or business−logic code, typically must bend over backwards to provide dynamic typing. Free and powerful languages, such as Ruby, supported entirely by their own programmer communities, are going to teach these big software vendors a thing or to.

−−
Phlip
http://industrialxp.org/community/bin/view/Main/TestFirstUserInterfaces