Implementing QuantLib

Luigi Ballabio

1 Introduction

The whys and wherefores

WITH THE ENTHUSIASM of youth, the QuantLib web site [22] used to state that QuantLib aimed at becoming "the standard free/open-source financial library." By interpreting such statement a bit loosely, one might say that it has somewhat succeeded—albeit by employing the rather devious trick of being the first, and thus for some time the *only* open-source financial library[1].

Standard or not, the project is thriving; at the time of this writing, each new release is downloaded a few thousand times, there is a steady stream of contributions from users, and the library seems to be used in the real world—as far as I can guess through the usual shroud of secrecy used in the financial world. All in all, as a project administrator, I can declare myself a happy camper.

But all the more for that, the lack of proper documentation begins to show. Although a detailed class reference is available (that was an easy task, since it can be generated automatically) it doesn't let one see the forest for the trees; so that a new user might get the impression that the QuantLib developers share the views of the Bellman from Lewis Carroll's *Hunting of the Snark*:

> "What use are Mercator's North Poles and Equators,
> Tropics, Zones and Meridian Lines?"
> So the Bellman would cry: and the crew would reply,
> "They are merely conventional signs!"

The purpose of this book is to fill a part of the existing void. It is a report on the design and implementation of QuantLib, alike in spirit—but, hopefully, with less frightening results—to the *How I did it* book[2] prominently featured in Mel Brooks' *Young Frankenstein*. If you are—or want to be—a QuantLib user, you will find here useful information on the design of the library that might not be readily apparent when reading the code. If you're working in quantitative finance, even if not using QuantLib, you can still read it as a field report on the design of a

[1] A few gentle users happened to refer to our library as "*the* QuantLib." As much as I like the expression, modesty and habit have so far prevented me from using it.

[2] In this case, of course, it would be "how *we* did it."

financial library. You will find that it covers issues that you might also face, as well as some possible solutions and their rationale. Based on your constraints, it is possible—even likely—that you will choose other solutions; but you might profit from this discussion just the same.

In my descriptions, I'll also point out shortcomings in the current implementation; not to disparage the library (I'm pretty much involved in it, after all) but for more useful purposes. On the one hand, describing the existing pitfalls will help developers avoid them; on the other hand, it might show how to improve the library. Indeed, it happened already that reviewing the code for this book caused me to go back and modify it for the better.

For reasons of both space and time, I won't be able to cover every aspect of the library. In the first half of the book, I'll describe a few of the most important classes, such as those modeling financial instruments and term structures; this will give you a view of the larger architecture of the library. In the second half, I'll describe a few specialized frameworks, such as those used for creating Monte-Carlo or finite-differences models. Some of those are more polished than others; I hope that their current shortcomings will be as interesting as their strong points.

The book is primarily aimed at users wanting to extend the library with their own instruments or models; if you desire to do so, the description of the available class hierarchies and frameworks will provide you with information about the hooks you need to integrate your code with QuantLib and take advantage of its facilities. If you're not this kind of user, don't close the book yet; you can find useful information too. However, I'll drop a hint here: another QuantLib administrator has repeatedly expressed the intent to write a companion *Using QuantLib* book. By all means, go and nag him until it gets written.

And now, as is tradition, a few notes on the style and requirements of this book.

Knowledge of both C++ and quantitative finance is assumed. I've no pretense of being able to teach you either one, and this book is thick enough already. Here, I just describe the implementation and design of QuantLib; I'll leave it to other and better authors to describe the problem domain on one hand, and the language syntax and tricks on the other.

As you already noticed, I'll write in the first person singular. True, it might look rather self-centered—as a matter of fact, I hope you still haven't put down the book in annoyance—but we would feel rather pompous if we were to use the first person plural. The author of this book feels the same about using the third person. After a bit of thinking, I opted for a less formal but more comfortable style (which, as you noted, also includes a liberal use of contractions.) For the same reason, I'll be addressing *you* instead of the proverbial acute reader. This will also help to avoid confusion; when I use the plural, I describe work done by the QuantLib developers as a group.

I will describe the evolution of a design when it is interesting on its own, or relevant for the final result. For the sake of clarity, in most cases I'll skip over the blind alleys and wrong turns taken; put together design decisions which were made at different times; and only show the final design, sometimes simplified.

I will point out the use of design patterns in the code I describe. Mind you, I'm not advocating cramming your code with them; they should be applied when they're useful, not for their own sake.[3] However, QuantLib is now the result of several years of coding and refactoring, both based on user feedback and new requirements being added over time. It is only natural that the design evolved toward patterns.

I will apply to the code listings in the book the same conventions used in the library and outlined in appendix B. I will depart from them in one respect: due to the limitations on line length, I might drop the std and boost namespaces from type names. When the listings need to be complemented by diagrams, I will be using UML; for those not familiar with this language, a short introduction by examples is given in appendix C.

Well, this is it. Let's dive.

[3] A more thorough exposition of this point can be found in [17].