END OF STUDIES PROJECT REPORT IMAFA2 2013/2014

LIBRAIRIE QUANTLIB 1.3: DESIGN AND IMPLEMENTATION OF NEW OPTIONS' PRICING ENGINES.

Presented by:

Ahmed AYADI

Nathan KRUCK

Nolan POTIER

Supervised by:

Anne Marie HUGUES
&
Luigi BALLABIO

Index

Table of Figures

Chapter I Introduction

I. Overview

QuantLib is a C++ library that eases writing applications for quantitative finance. In its turn, it uses the **Boost** library **[**1**]**. Since its publication in November 2000, it has continued to grow in popularity in the financial sector. QuantLib is an open source project, it has many advantages. First of all, it is free. It is also thoroughly tested. This is due to the collective intelligence of all users and developers. All errors are quickly detected and corrected. Finally, it is easy to extend because the source code is freely available, the developer can learn from the implementation of its functionality. QuantLib is structured into modules [2]. Each module covers a distinctive aspect of all features, and contains a set of components.

II. Problem Statement

In fact, price calculation is not available for every type of financial product, in QuantLib, it is still some further work to do. So in the first part of our project, we have been given the Partial-Time Barriers as an exotic type of options, in order to design and implement a Pricer for it. In the second part, we were assigned to design classes for two other types of exotic options: Complex Chooser Options and Extendible Options.

In fact, there are many types of theoretical solution formulae to calculate the price of an option, in this project we have been given the analytical solution formula mentioned in the book: **"The Complete Guide To Option Pricing Formulas" by Espen Gaarder Haug.**

III. Purpose of the project

Our work involves the mastery of the use of existing pricing engines and components of QuantLib, in the first place, and the design and development of others, in the second. So our planning for the project will cover a learning period including documentation reading and testing of pricers, then comes the other part, the implementation of our solution.

So in the beginning we configured our work environment in order to compile the library, taking care of all dependencies problems.

We will use the IDE Visual Studio as development environment and Visual Paradigm 8.0 as UML designing tool.

Our work will deliver a source code able to calculate the price for the different variants of the Partial-Barrier Call options, Complex Chooser Options and Extendible Options, using the available theoretical formulae.

Chapter II QuantLib Architecture

The QuantLib architecture has been made carefully, making it easy to use and extend by adding new pricing functionalities. For this it is necessary to know the two main classes of the library.

On one hand, the abstract class « Instrument » that allows to generalize financial instruments like Vanilla Option with its different parameters (volatility, strike, maturity …). On the other hand, we have the « PricingEngine » abstract class that allows to calculate the instrument price.

It's necessary to understand the use of these two classes for any given instrument. Notice that it is not always the case that a unique pricing method is used; one might want to use multiple methods for the same instrument.

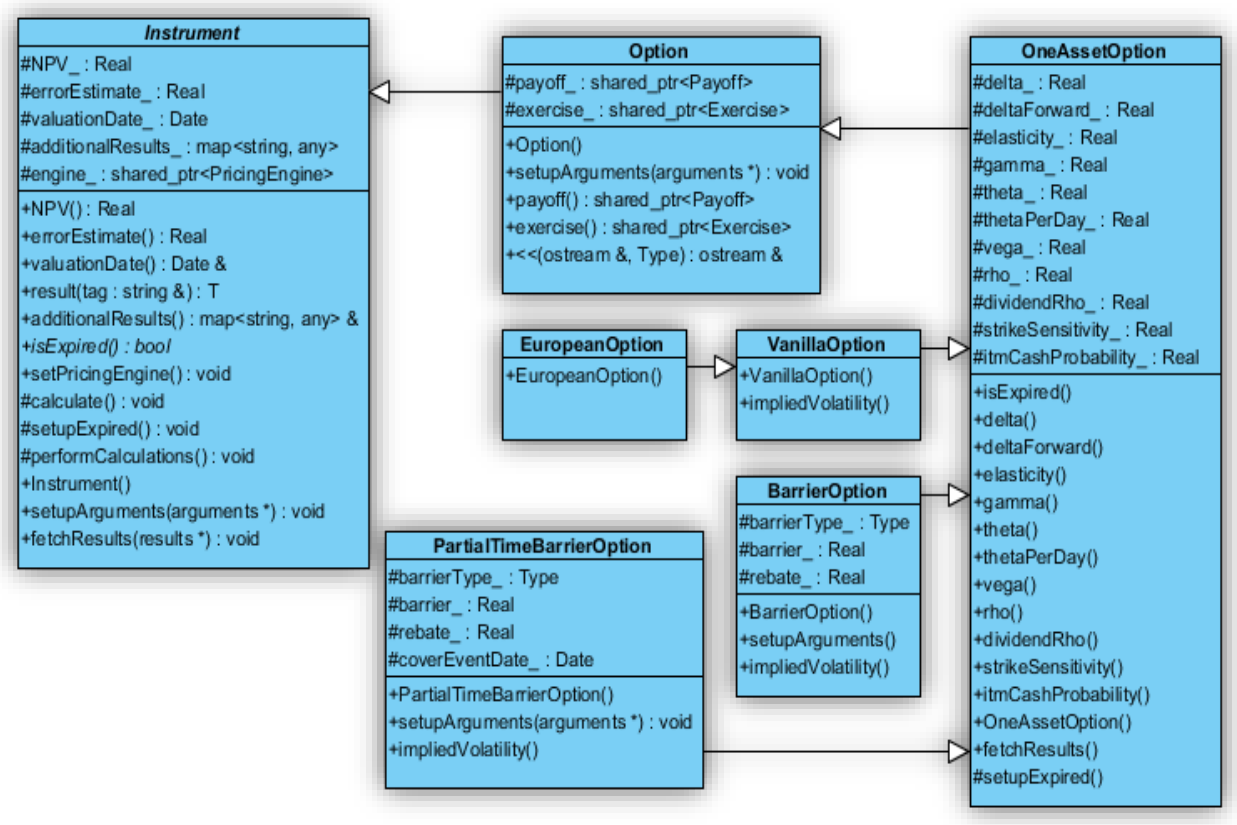

Figure 1 Instrument Class Inheritance

The figure 1-(Elaborated using UML tool Visual Paradigm)-shows a part of QuantLib Class Inheritance Diagram.

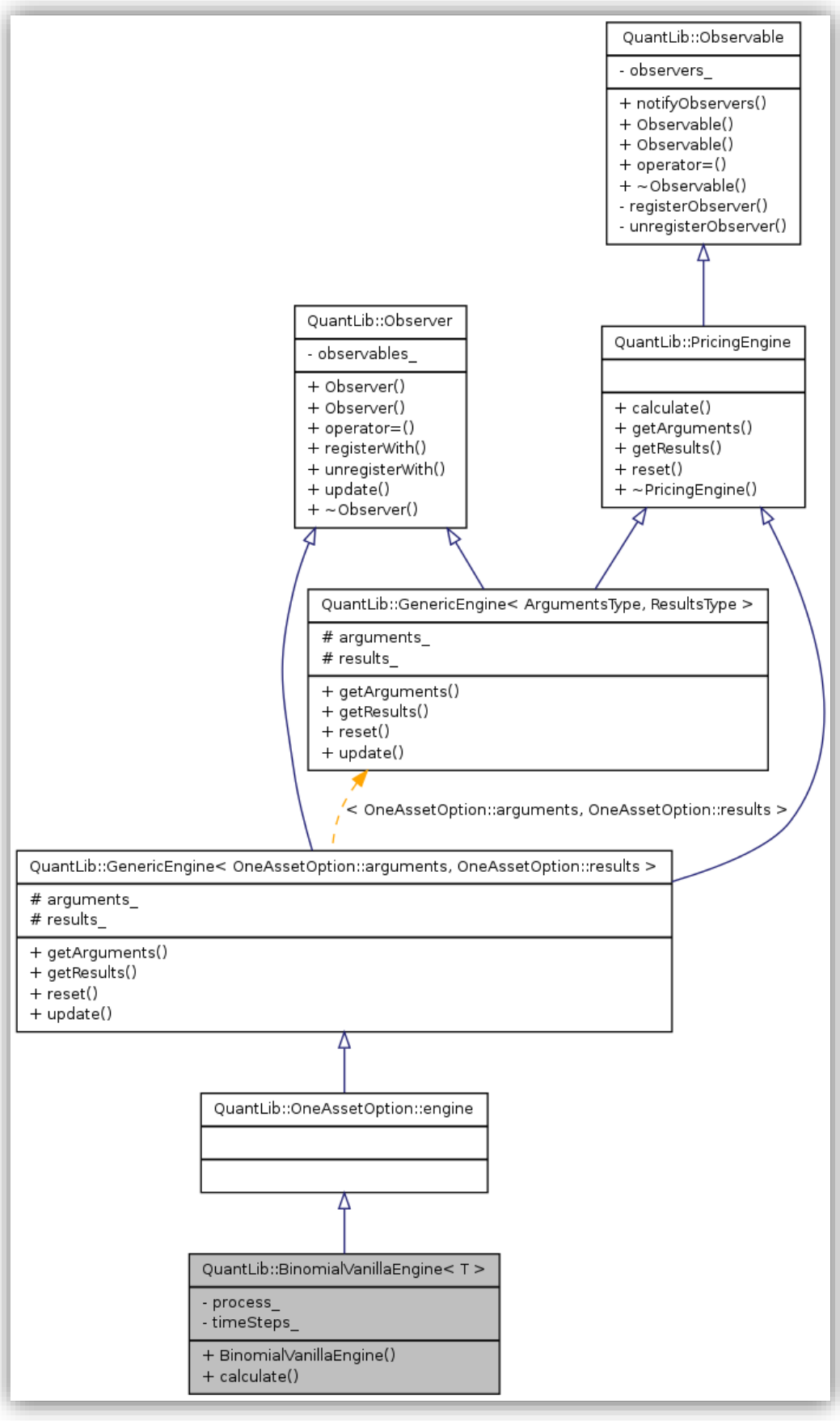

Figure 2 PricingEngine Inheritance Graph; Example Binomial Vanilla Option Engine [6]

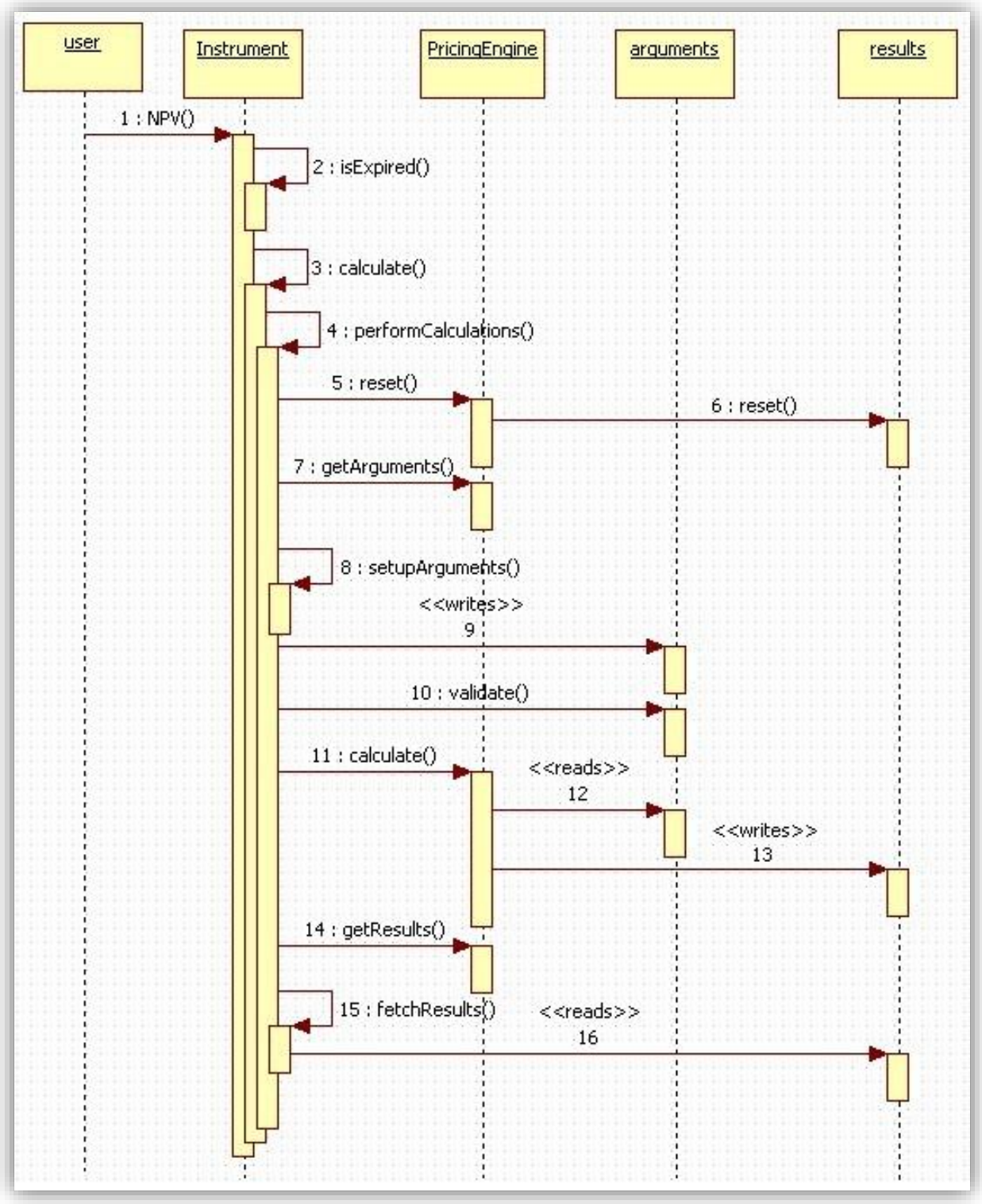

Figure 3 Net present value calculus Sequence Diagram [7]

The figure 3 shows the different exchanges between the objects to calculate the net present value (NPV) of an instrument.

Chapter III Barrier Options:

I. Overview:

Barrier option is a path-dependent option where the price is based on the fluctuations in the underlying's value during all or part of the contract term. It was traded in the late 60s and it has been used to manage the risk. It comes in many flavors and forms, but its key characteristic is that this type of option is either initiated or exterminated upon reaching a certain barrier level; that is, it is either knocked in or knocked out.

> "A basic American option is one type of **path dependent option**. Because it can be exercised at any time prior to expiration, its value will change as the underlying asset's value changes. An Asian option, also called an average option, is another type of path dependent option, because its payoff is based on the average price of the underlying asset during the contract term. Similarly, a barrier option would be considered a path dependent option because its value changes if the underlying asset reaches or surpasses a specified price. The lookback option and Russian option are also path-dependent options."[3]

In our work we consider the most basic type of barrier option; "**the single barrier**". This option comes in 4 types where each type can be a Call or a Put option:

- ✓ Up & In
- ✓ Up & Out
- ✓ Down & In
- ✓ Down & Out

An "In" barrier option, for example, means that it becomes active once crossing the barrier level. If the underlying asset fails to cross the barrier, the Barrier Option you bought becomes worthless piece of paper upon expiration even if the underlying asset is trading above (call option) or below (put option), its strike price!

So it's clear, at first thoughts, that Barrier Options are more dangerous than normal plain vanilla options. Then why would investors use them? Basically because they carry a much **lower extrinsic value** than plain vanilla options. [5].

	Advantage	Disadvantage
Knock-In Barrier Options	1. Cheaper, resulting in higher profits 2. Ideal for speculating huge moves	1. Higher risk of loss if underlying asset moves moderately 2. Commonly traded for forex, not stocks.
Knock-Out Barrier Options	1. Cheaper, resulting in higher profits 2. Ideal for speculating small moves	1. Higher risk of loss if underlying asset rallies 2. Commonly traded for forex, not stocks.

Figure 4 Barrier Options versus Vanilla Options

There are various barriers types including Parisians, double barriers, and partial time barriers. Here, we discuss partial-time barrier options.

II. Partial-Time Barrier Options

The watching period for a barrier crossing is restricted to a fraction of the option's lifetime. Here we have two types: **partial-time-start** and **partial-time-end**.

The monitoring period of **Partial-time-start** barrier options starts at time zero and ends at a random date before expiration. As for **partial-time-end** barrier options, they have the monitoring period start at an arbitrary date before expiration and end at expiration.

There are two classes of **partial-time-end** barrier options [4]: Type **B1** is defined such that only a barrier hit or crossed causes the option to be knocked out, and hence there is no difference between up and down options. Type **B2** options are defined such that a down-and-out call is knocked out as soon as the underlying price is below the barrier. Similarly, an up-and-out call is knocked out as soon as the underlying price is above the barrier.

III. Implementation

The first phase of our work, concerned the reading of the available library documentation and its installation and configuration on our computers. Also we had to read about Barrier Options.

To implement the barrier option our first idea was to reuse the class "OptionBarrier" included in the library and combined it with a "VanillaOption". But after we understood the architecture of the library it was easier to implement the solution directly with "PartialTimeBarrierOption" class inheriting from the "OneAssetOption" class. So we elaborated a class diagram to represent the new class and hence to explicit the relations with the other classes needed for the solution (see Figure 1: Instrument Class Inheritance).

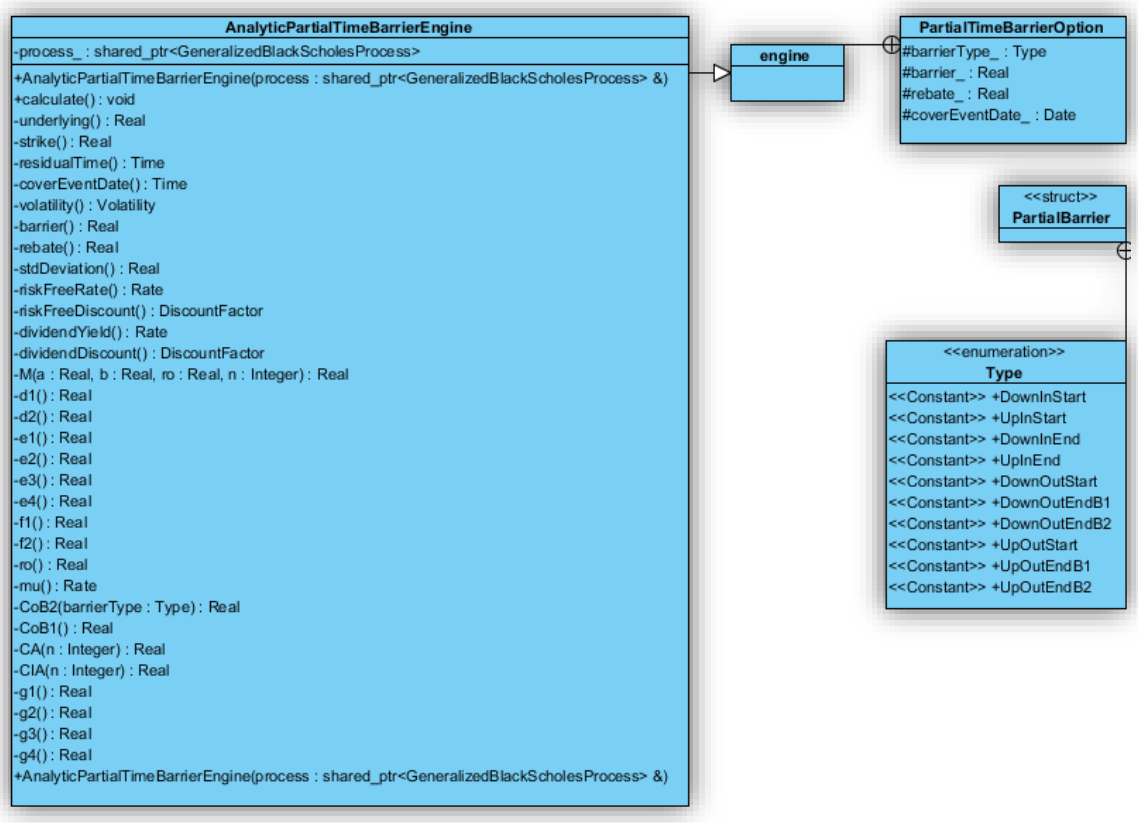

Figure 5 Partial-Time-Barrier Option class and Analytical Engine

Once the architecture was respected we had to implement the pricing method to the Partial-Barrier Option. For this we have used formulas from the book: "The Complete Guide To Option Pricing Formulas by Espen Gaarder Haug".

Example of formulas implemented:

S: underlying
H: barrier
X: Strike
T2: expiring date
t1: barrier window start or stop

b: cost of carry (here q-r)
q: Dividend yield
r: interest rate
σ: volatility

Partial-Time-Start-Out Options

$$c_A = Se^{(b-r)T_2}\left[M(d_1, \eta e_1; \eta\rho) - \left(\frac{H}{S}\right)^{2(\mu+1)} M(f_1, \eta e_3; \eta\rho)\right]$$
$$- Xe^{-rT_2}\left[M(d_2, \eta e_2; \eta\rho) - \left(\frac{H}{S}\right)^{2\mu} M(f_2, \eta e_4; \eta\rho)\right], \qquad (4.59)$$

where $\eta = -1$ for an up-and-out call (c_{uoA}) and $\eta = 1$ for a down-and-out call (c_{doA}), and

$$d_1 = \frac{\ln(S/X) + (b + \sigma^2/2)T_2}{\sigma\sqrt{T_2}}, \quad d_2 = d_1 - \sigma\sqrt{T_2}$$

$$f_1 = \frac{\ln(S/X) + 2\ln(H/S) + (b + \sigma^2/2)T_2}{\sigma\sqrt{T_2}}, \quad f_2 = f_1 - \sigma\sqrt{T_2}$$

$$e_1 = \frac{\ln(S/H) + (b + \sigma^2/2)t_1}{\sigma\sqrt{t_1}}, \quad e_2 = e_1 - \sigma\sqrt{t_1}$$

$$e_3 = e_1 + \frac{2\ln(H/S)}{\sigma\sqrt{t_1}}, \quad e_4 = e_3 - \sigma\sqrt{t_1}$$

$$\mu = \frac{b - \sigma^2/2}{\sigma^2}, \quad \rho = \sqrt{t_1/T_2}$$

Figure 6 Partial-Time-Start-Out Barrier Option formula

IV. Results:

Expected results from “The Complete Guide To Option Pricing Formulas by Espen Gaarder Haug":

TABLE 4-16

Partial-Time-End Barrier Call Type B1 Option Values

$(H = 100, r = b = 0.1, \sigma = 0.25, T_2 = 1)$

		Barrier Monitoring Time t_1				
S	X	0	0.25	0.5	0.75	1
95	90	0.0393	6.2747	10.3345	13.4342	17.1612
95	110	0.0000	3.7352	5.8712	7.1270	7.5763
105	90	9.8751	15.6324	19.2896	22.0753	25.4213
105	110	6.2303	9.6812	11.6055	12.7342	13.1376

Figure 7 Expected results for Partial-Time Barrier

The figure below shows our results, they are as expected in the book but we were not able to calculate the value when $t_1 = 0$ and $t_1 = 1$ because of some transitional calculation where these values imply to divide by 0. We are very satisfied with the accuracy of our results even if we did not conduct a large and extensive range of tests on all options due to a lack of theoretical data which are difficult to find.

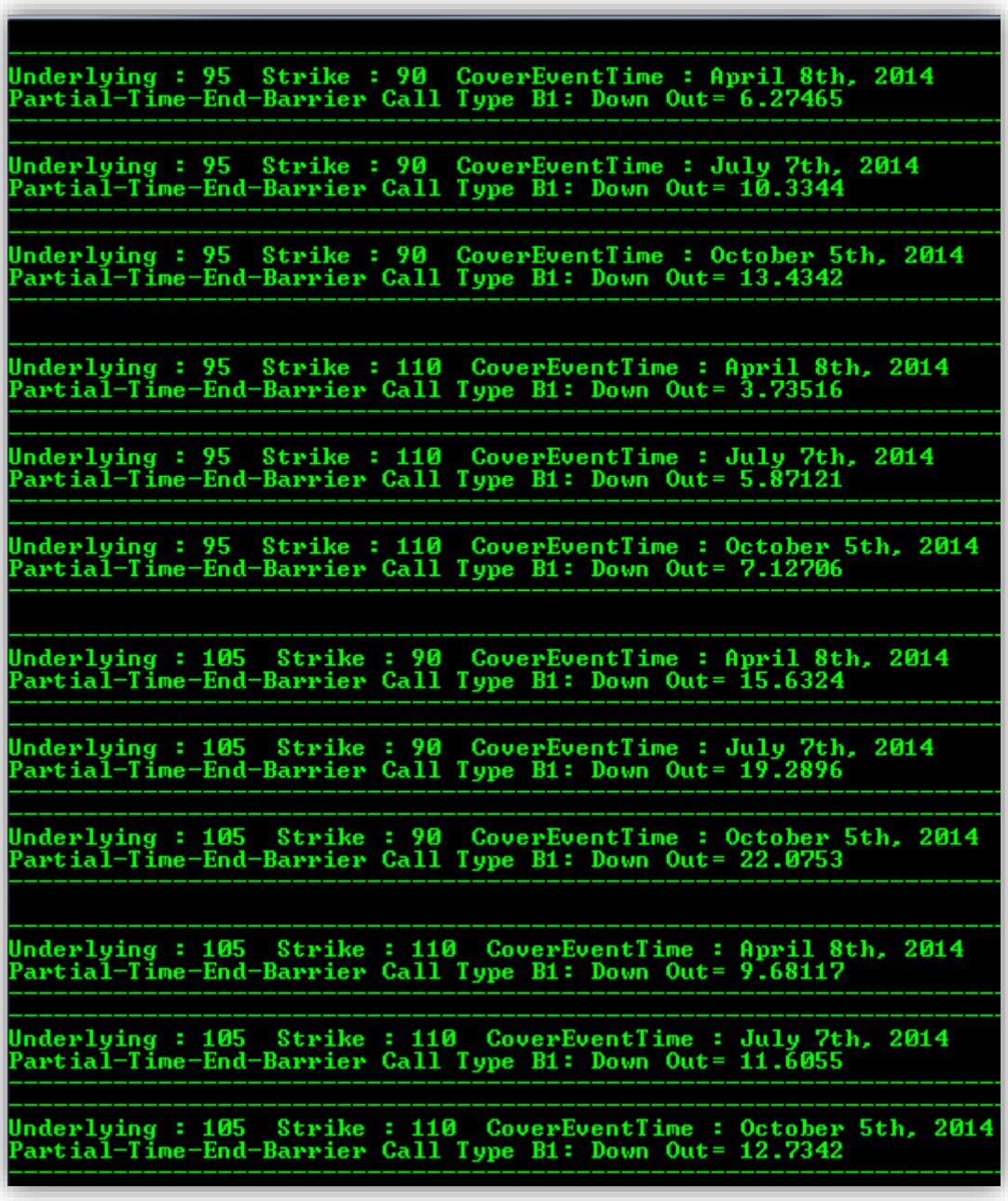

Figure 8 Calculated results, using our pricing engine

Chapter IV Complex Chooser Options

I. Overview:

The common name for this type of options is “Chooser Option”, but it is also called "as-you-like-it option”. It specifies an exercise price “**X**” on the date of issue, and allows the holder to decide after a certain period “**[0, t_1]”**, determined at the beginning ("The choosing date”), whether the option is to be converted as call or put. After deciding to convert the "chooser option", its profile becomes that of a standard option (call or put) with the specified exercise price and expiry date T_2 (already mentioned in the contract). During the first period, the investor waits for the uncertainty generating event to be resolved and choose at the beginning of the second period, the optimal type of option.

The premium for the "chooser option" is higher than Call or Put, but much cheaper than the cost of a “**Long Straddle**" (premium (Call) +premium (Put), with the same maturity and the same strike). The Chooser Option will be a losing choice comparing to a Long Straddle, if after the "choosing date" $\mathbf{t_1}$ and the conversion to a Call or a Put, other events accrue and reverse the trend in the opposite direction to the one already chosen.

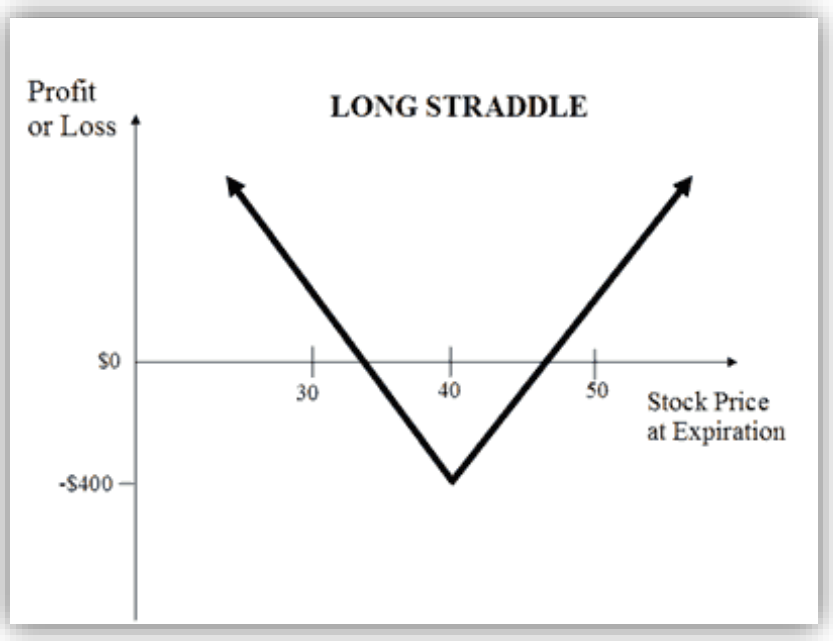

Figure 9 Long Straddle P&L [9]

We can distinguish two types of Chooser Options, the "Simple" and "Complex" Chooser. The next section will deal with Complex Chooser Options.

The next figure shows the price of a simple chooser option in function of choosing date, which explains logically the importance of the advantage such option gives to its holder:

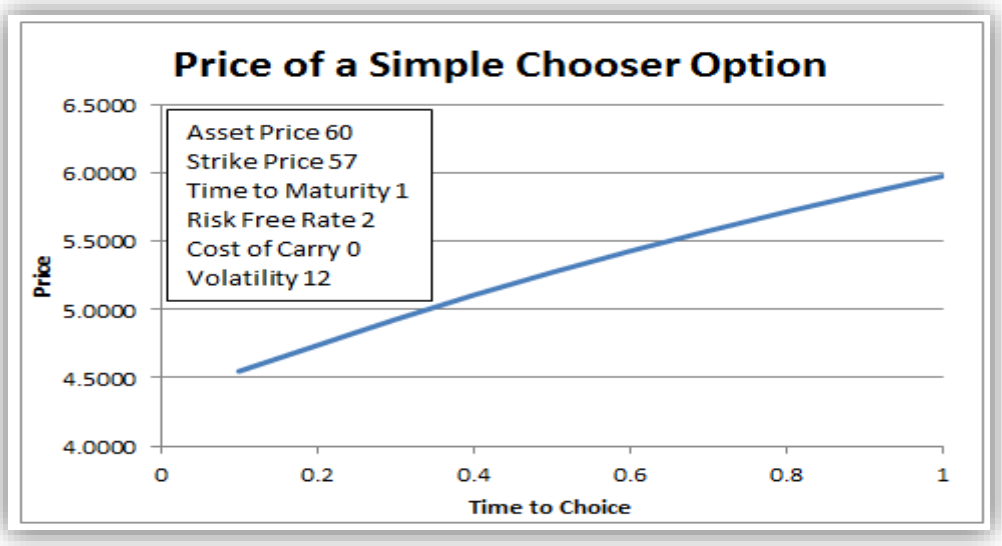

Figure 10 Simple Chooser Option Price = f (choosing date) [10]

II. Complex Chooser Options:

For Simple Chooser Options, comparing to a standard option, we have only to specify the Choosing date t1. But for Complex Chooser Options we have to specify two strikes X_c and X_p, and two expiry dates T_c and T_p.

In times of great uncertainty about the future price of assets, many investors choose to stay out of the market. We can cite, for example, the case of financial crisis, or even the event of a simple, economic press release. In these scenarios, investors believe that these events will have a significant impact on the value of a stock share, causing very high fluctuations up or down. A "chooser" is appropriate for this type of uncertain situation, allowing the investor to defer decisions on asset coverage or speculation, for a defined period.

III. Implementation:

To implement the Complex Chooser Option in QuantLib we chose to make an inheritance from the One Asset Option Class and add the specific attributes for this type of option. So we elaborated a class diagram to represent the new class and hence to explicit the relations with the other classes needed for the solution.

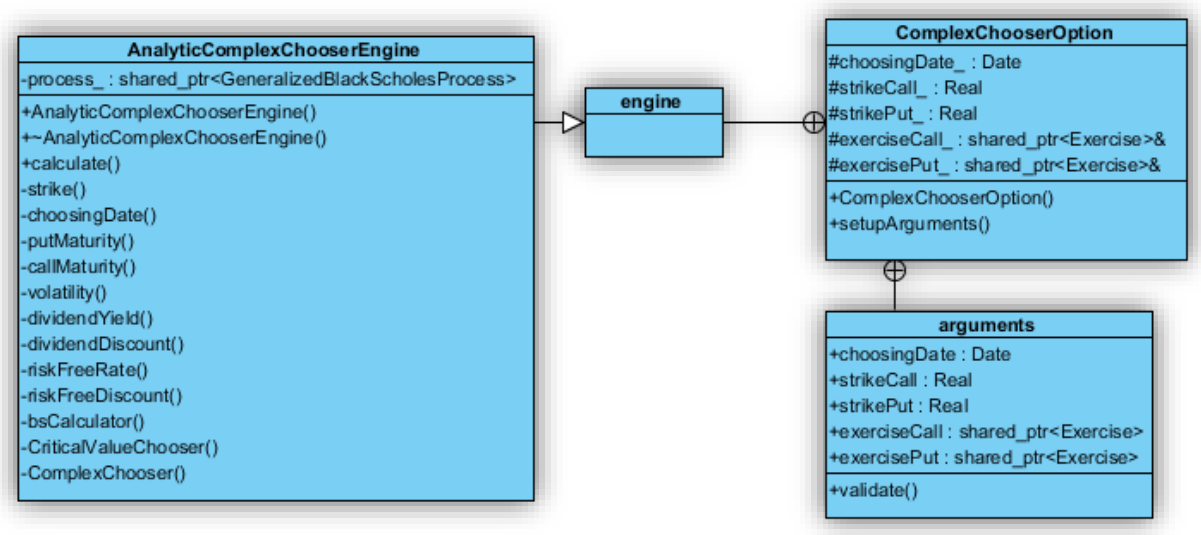

Figure 11 Complex Chooser Option Class and Analytical Engine

Once the architecture was respected we had to implement the pricing method to the Complex Chooser Option. For this, we have used formulas of the book: "The Complete Guide To Option Pricing Formulas by Espen GaarDer Haug"

Example of formulas implemented:

$$w(S, X_c, X_p, t, T_c, T_p) = \max[c_{BSM}(S, X_c, T_c), p_{BSM}(S, X_p, T_p)],$$

where $c_{BSM}(S, X, T)$ and $p_{BSM}(S, X, T)$ are the BSM call and put formulas, respectively.

$$\begin{aligned} w = {} & Se^{(b-r)T_c} M(d_1, y_1; \rho_1) \\ & - X_c e^{-rT_c} M(d_2, y_1 - \sigma\sqrt{T_c}; \rho_1) - Se^{(b-r)T_p} M(-d_1, -y_2; \rho_2) \\ & + X_p e^{-rT_p} M(-d_2, -y_2 + \sigma\sqrt{T_p}; \rho_2), \end{aligned} \tag{4.27}$$

where T_c is the time to maturity on the call, and T_p is the time to maturity on the put.

$$d_1 = \frac{\ln(S/I) + (b + \sigma^2/2)t}{\sigma\sqrt{t}} \qquad d_2 = d_1 - \sigma\sqrt{t}$$

$$y_1 = \frac{\ln(S/X_c) + (b + \sigma^2/2)T_c}{\sigma\sqrt{T_c}} \qquad y_2 = \frac{\ln(S/X_p) + (b + \sigma^2/2)T_p}{\sigma\sqrt{T_p}}$$

$$\rho_1 = \sqrt{t/T_c} \qquad \rho_2 = \sqrt{t/T_p},$$

and I is the solution to

$$Ie^{(b-r)(T_c-t)}N(z_1) - X_c e^{-r(T_c-t)}N(z_1 - \sigma\sqrt{T_c - t}) +$$
$$Ie^{(b-r)(T_p-t)}N(-z_2) - X_p e^{-r(T_p-t)}N(-z_2 + \sigma\sqrt{T_p - t}) = 0$$

$$z_1 = \frac{\ln(I/X_c) + (b + \sigma^2/2)(T_c - t)}{\sigma\sqrt{T_c - t}} \qquad z_2 = \frac{\ln(I/X_p) + (b + \sigma^2/2)(T_p - t)}{\sigma\sqrt{T_p - t}}$$

Figure 12 Complex Chooser Option formula

IV. Results:

Expected results from “The Complete Guide To Option Pricing Formulas by Espen Gaarder Haug":

Example
Consider a complex chooser option that gives the holder the right to choose whether the option is to be a call with six months to expiration and strike price 55, or a put with seven months to expiration and strike price 48. The time to choose between a put or call is in three months, the underlying stock price is 50, the risk-free interest rate per year is 10%, the dividend yield is 5% per year, and the volatility per year is 35%. $S = 50$, $X_c = 55$, $X_p = 48$, $T_c = 0.5$, $T_p = 0.5833$, $t = 0.25$, $r = 0.1$, $b = 0.1 - 0.05 = 0.05$, and $\sigma = 0.35$.

$$d_1 = \frac{\ln(50/51.1158) + (0.05 + 0.35^2/2)0.25}{0.35\sqrt{0.25}} = 0.0328,$$

where a Newton-Raphson search gives the solution to the critical value $I = 51.1158$, and

$$d_2 = d_1 - 0.35\sqrt{0.25} = -0.1422$$

$$y_1 = \frac{\ln(50/55) + (0.05 + 0.35^2/2)0.5}{0.35\sqrt{0.5}} = -0.1604$$

$$y_2 = \frac{\ln(50/48) + (0.05 + 0.35^2/2)0.5833}{0.35\sqrt{0.5833}} = 0.3955$$

$$\rho_1 = \sqrt{0.25/0.5} = 0.7071 \qquad \rho_2 = \sqrt{0.25/0.5833} = 0.6547$$

$$M(d_1, y_1; \rho_1) = 0.3464 \qquad M(d_2, y_1 - 0.35\sqrt{0.5}; \rho_1) = 0.2660$$

$$M(-d_1, -y_2; \rho_2) = 0.2725 \qquad M(-d_2, -y_2 + 0.35\sqrt{0.5833}; \rho_2) = 0.3601$$

$$\begin{aligned} w = {} & 50e^{(0.05-0.1)0.5} M(d_1, y_1; \rho_1) \\ & - 55e^{-0.1\times 0.5} M(d_2, y_1 - \sigma\sqrt{T_c}; \rho_1) \\ & - 50e^{(0.05-0.1)0.5833} M(-d_1, -y_2; \rho_2) \\ & + 48e^{-0.05\times 0.5833} M(-d_2, -y_2 + \sigma\sqrt{T_p}; \rho_2) = 6.0508 \end{aligned}$$

Figure 13 Expected results for Holder-Chooser Option

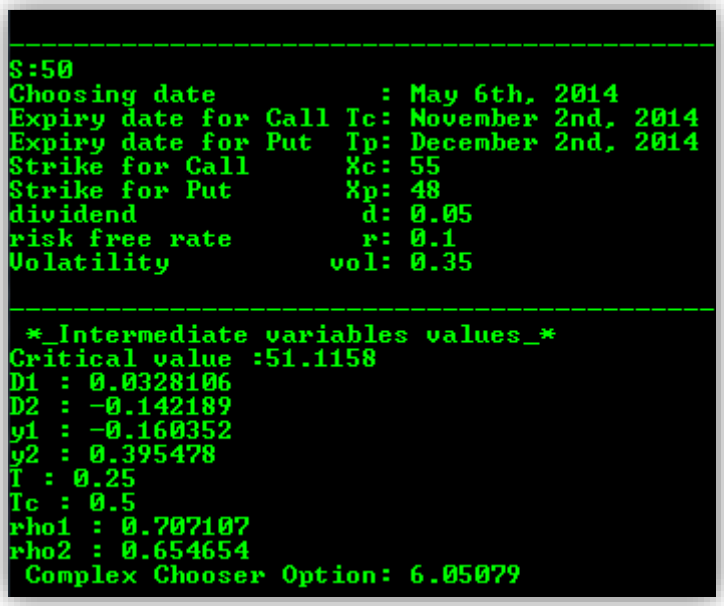

Figure 14 Empiric results for Holder-Chooser Option using our pricing engine

Chapter V Extendible Options

I. Overview:

For standard option the holder has the right to exercise the contract at a specific date (maturity). For Extendible Options, in addition to that right, there is the right to extend the expiry date. So, this type of options may be extended by the writer or the holder as they can choose the new expiry date with new strike.

Extendible options are generally classified as Holder-Extendible options and Writer-Extendible options. Hence, a Holder-Extendible option allows the holder to decide whether to extend the contract or not. On the other hand, he has to compensate the writer with an additional premium. Writer-Extendible options, however give seller the right to extend the option without paying the holder any extra amount, or refunding any percentage of the original premium, in order to extend the option.

Predictably, the option will be extended only if it is out of the money, as there is little value in extending the option if it is already in the money. Extendible options come in two forms: extendible calls and extendible puts. [8]

II. Implementation:

To implement the Extendible Option in QuantLib we chose the same as for the two other options mentioned above, to make an inheritance from the One Asset Option Class and add the specific attributes for this type of option. Hence our OO solution is based on a specific design in order to satisfy QuantLib architecture constraints.

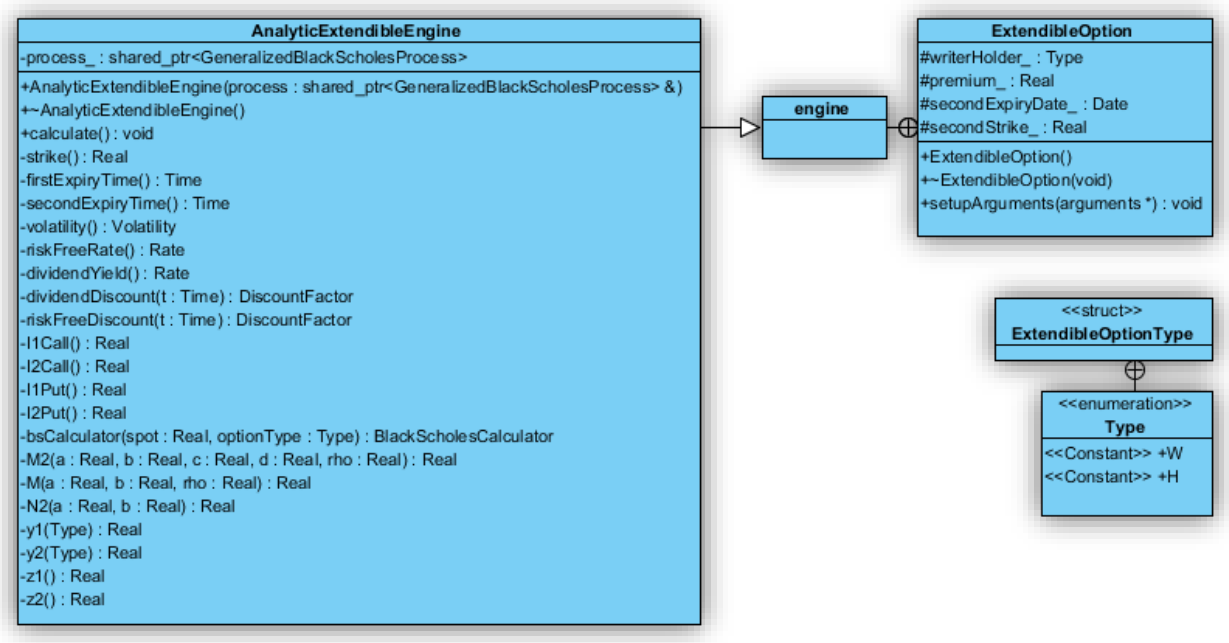

Figure 15 Extendible Option Class and Analytical Engine

Once the architecture was respected, we had to implement the pricing method to the Extendible Option by developing its pricing engine. For this, we have used formulas of the book: "The Complete Guide To Option Pricing Formulas by Espen GaarDer Haug".

Example of formulas implemented:

Writer-Extendible Call

$$c = c_{BSM}(S, X_1, t_1) + Se^{(b-r)T_2} M(z_1, -z_2; -\rho)$$
$$- X_2 e^{-rT_2} M(z_1 - \sigma\sqrt{T_2}, -z_2 + \sigma\sqrt{t_1}; -\rho) \quad (4.37)$$

Writer-Extendible Put

$$p = p_{BSM}(S, X_1, t_1) + X_2 e^{-rT_2} M(-z_1 + \sigma\sqrt{T_2}, z_2 - \sigma\sqrt{t_1}; -\rho)$$
$$- Se^{(b-r)T_2} M(-z_1, z_2; -\rho) \quad (4.38)$$

$$y_1 = \frac{\ln(S/I_2) + (b + \sigma^2/2)t_1}{\sigma\sqrt{t_1}} \qquad y_2 = \frac{\ln(S/I_1) + (b + \sigma^2/2)t_1}{\sigma\sqrt{t_1}}$$

$$z_1 = \frac{\ln(S/X_2) + (b + \sigma^2/2)T_2}{\sigma\sqrt{T_2}} \qquad z_2 = \frac{\ln(S/X_1) + (b + \sigma^2/2)t_1}{\sigma\sqrt{t_1}}$$

$$\rho = \sqrt{t_1/T_2},$$

Figure 16 Writer-Extendible Option formula

III. Results:

Chapter VI Conclusion

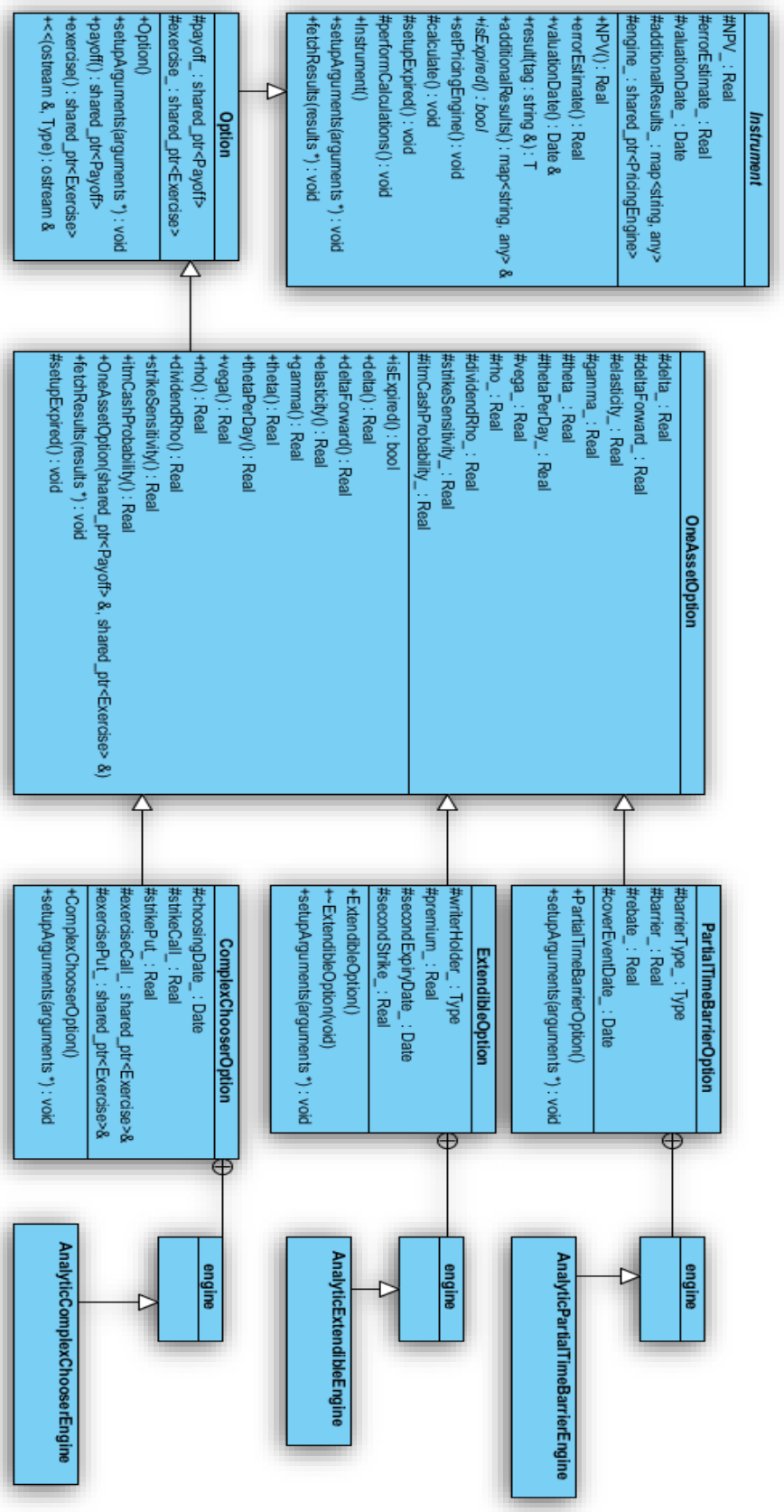

Figure 17 Complete classes integrated in QuantLib

So as the last figure presents, we managed to propose our design solution with respect to QuantLib architecture and constraints. We integrated our solution by inheriting from the OneAssetOption class and specifying every child class with its extra parameters.

Most facilities we exploited using QuantLib was the elegant inheritance vision from the beginning, it made it easier for us to extend it with our solution once we worked hard on understanding the mechanism with which the Library works. Second thing was the template design pattern that helps the integration of as many pricing engines as we want. For our project we had the chance to prove our skills for designing object oriented solutions and compute analytical pricing formulas with the sufficient functional and theoretical knowledge. Our work also gave us a great chance to contribute in QuantLib library and put another brick on the wall of financial IT.

Chapter VII Netography

[1] Boost Library: http://www.codeproject.com/Articles/4496/An-Introduction-to-Boost

[2] QuantLib1.3 modules: http://quantlib.org/reference/modules.html

[3] http://www.investopedia.com/terms/p/pathdependentoption.asp

[4] http://hosho.ees.hokudai.ac.jp/~kubo/Rdoc/library/fExoticOptions/html/BarrierOptions.html

[5] http://www.optiontradingpedia.com/barrier_options.htm

[6] http://quantlib.sourcearchive.com/documentation/1.11/classQuantLib_1_1BinomialVanillaEngine.html

[7] http://quantlife.tistory.com/category/quantlib?page=2

[8] http://www.investment-and-finance.net/derivatives/e/extendible-option.html

[9] http://www.theoptionsguide.com/images/long-straddle.gif

[10] http://investexcel.net/wp-content/uploads/2012/02/PriceofSimpleChooserOption.png