7CCSMSUF Tutorial 1: Requirements analysis

December 23, 2017

Here is an initial requirements statement about a bond analysis system:

> The system will assist investors in evaluating investments in *bonds*. A bond has a *term* (the number of years to its expiry), a *coupon* (the percentage of the investment, paid to investor at regular intervals), and a $frequency$ of payments. Bonds have names, eg. "UK government bond".
>
> For this system, it is assumed that $frequency = 1$ per year, and initial investment = 100.
>
> Investors receive back the invested sum at the term of the bond, together with the final coupon payment.
>
> Eg., invest £100 in bond b paying 10% coupon annually for 5 years. 4 payments of £10, one of £110.
>
> Bonds have a *price*: the investor pays this with the sum invested.
>
> The system should compute and display the *payout* of each bond: the sum of its payments. Eg., £150 in the example.
>
> The system should compute and display each bond $value(r)$: the sum of *discounted* payments, using inflation rate r, where payment X is discounted after N years of inflation r to $\frac{X}{(1+r)^N}$.
>
> Eg., £10 1 year in future, with 5% inflation, is worth 10/1.05 today = £9.52.
>
> For each bond, system should calculate and show its *Macaulay duration* for rate r:
>
> $$duration = (\Sigma_{p:payments} \frac{p.time * coupon}{(1+r)^{p.time}} + \frac{term * 100}{(1+r)^{term}}) / value(r)$$
>
> Where $p.time$ is time of p, in years from start of bond.
>
> For each bond, system should compute and show its *internal rate of return*, accurate to within 10^{-3}: this is the rate r such that
>
> $$price = \Sigma_{p:payments} \frac{coupon}{(1+r)^{p.time}} + \frac{100}{(1+r)^{term}}$$
>
> Ie., such that $price = value(r)$.

1. List the *functional* requirements from this statement.
2. List the *non-functional* requirements from the statement.

3. Are there any additional requirements/issues that should be included? Are there any ambiguities and unclear aspects?

4. Draw a use case diagram for the system, showing the main functionalities as use cases.