Concepts

Contents

1. Overview

The objective of this document is to explain the different components and concepts that make up mguchiQ.

mguchiQ is a financial and statistical forward-looking calculation engine that calculates values for time periods out into the future.

The central component of mguchiQ is a **Model**. A mguchiQ **Model** comprises the data specification and calculations necessary to compute the results you require. A **Model** is built by first creating a prototype in Excel and then importing this into mguchiQ.

A **Run** is the process of calculating Model results from a set of data inputs. mguchiQ comprises a variety of Calculation Types that calculate and present results in different ways. mguchiQ can compute both Deterministic and Stochastic results.

mguchiQ is web-based and can be scaled indefinitely.

2. How Results are Calculated

mguchiQ produces results along the following lines:

For every **item** in a given data set *(e.g., for every policy in a book of policies)*

For each **time period** from 1 to n *(e.g., a number of months into the future)*

For each value you require to be **calculated**

Calculate the **value**

3. Models

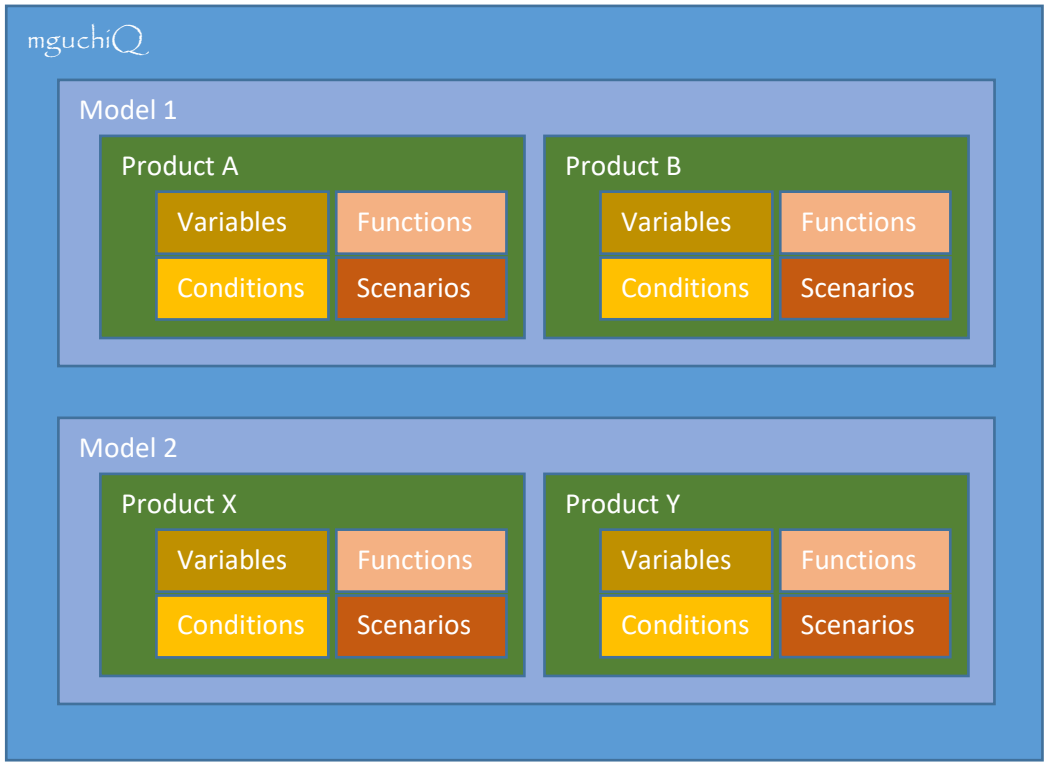
Building a **Model** is the first step in being able to use the functionality of mguchiQ. Once a model is built it can then be run against a set of data to produce results.

Building a **Model** requires that you define the data requirements and calculations that represent the financial **Products** you wish to model. A **Model** is made up of one or more **Products** – think of **Products** as financial instruments such as life policies, annuities, fixed deposits, etc. A **Model** can be simple, representing only a few calculations for a single **Product** or can be very complex, representing a large set of calculations for a multitude of **Products**. The output of the **Model** building process is an actual piece of compiled computer code that is then executed when running the model against a set of data.

Each **Product** within a **Model** is described by a set of **Variables** (data) and a set of **Functions** (calculations on the data).

In addition, a **Model** can contain a set of **Scenarios** that describe changes to **Variables** so that you can see how this data change will affect your results. An example of a scenario could be a change in our investment assumptions (possibly represented by a change in the yield curve), and / or a change in our mortality assumptions.

Lastly a **Model** can contain a set of **Conditions** by which you can optionally partition your results. An example of a **Condition** could be to partition our results by all policy holders under the age of 50 and all policy holders over the age of 50.



3.1. Products

Products represent financial instruments by describing the data (**variables**) required to perform calculations and the calculations (**functions**) themselves.

A life insurance product, for example, may require data specific to each policy, such as the *premium* and *sum insured* – this type of data is called **instance variables** as it is specific to each policy instance. Data not specific to each policy, such as a *yield curve*, may also be required – this type of data is called **non-instance variables**.

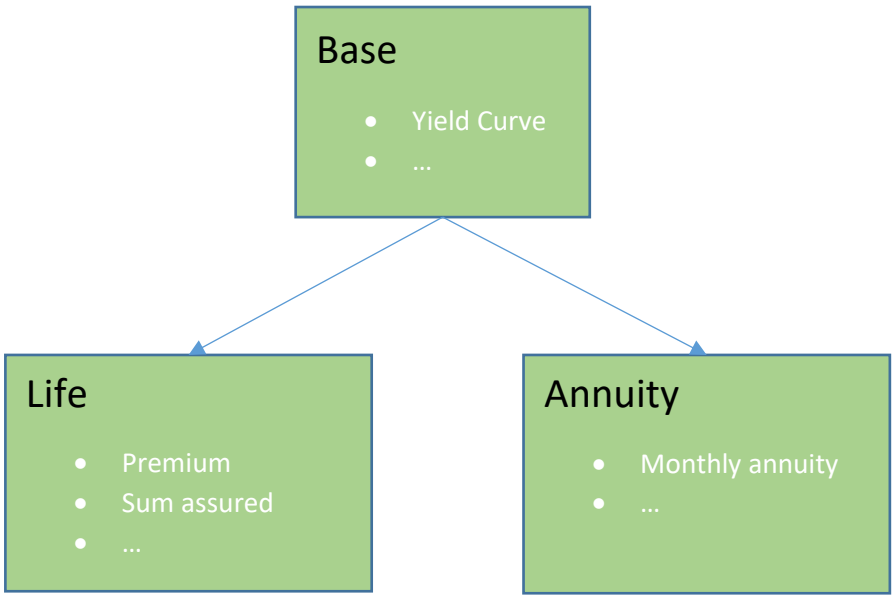
Our ultimate aim may be to calculate the *Best Estimate Liability (BEL)* of a set of life insurance policies – *BEL* would be an example of a **function** of the product. We may require other **functions** to aid in the calculation of *BEL*, such as the number of *Deaths* and the *Net Cashflow* per period into the future.

A model can be simple and consist of only one product.

Alternatively a model could consist of more than one unrelated products – by unrelated we mean that they share nothing on common.

More likely, if you build a model with more than one product, those products will share some information – this is achieved by the concept of product *inheritance*.

Inheritance allows you to share information in a product hierarchy in order to avoid data and function duplication. As an example, if we have 2 products in our model, say a *life insurance product* and an *annuity*, and we want to calculate the *present value of all future cashflows* for these 2 products, it is likely that we will need a *yield curve* for both the *life insurance product* and the *annuity*. The *yield curve* can be defined in a *base* product from which both the *life insurance product* and the *annuity* products are descended.



3.1.1. Constants

Constants are merely used to improve readability of formulae.

3.1.2. Variables

Variables describe the data required by a product necessary for the calculations (functions) of that product.

There are various types of variables that can be used to describe the data of the products in a model.

3.1.2.1. Instance Variables

Instance variables are assigned to each specific instance of a product. Examples could be the *sum assured* of a life policy, the *interest rate* of a fixed deposit, etc.

Instance variables, sometime referred to as model point data, are generally supplied in a data source (file / database) comprising a download of products from a source system, for example, current inforce life assurance policies, or current inforce annuities.

3.1.2.2. Non-instance Variables

Non-instance variables are data shared by all instances of a product, i.e. they are not specific to any one particular product instance. Examples could be a *yield curve* by which to discount cashflow, *mortality tables*, etc.

There are 3 types of Non-instance variables:

Variable type	Description
Single Variables	Single Variables are singular, as opposed to array, variables. An example could the monthly *Expense* incurred to maintain each policy.
Series Variables	Series Variables are variables made up of a series of values, essentially a one-dimensional array. Examples could be a *Yield Curve* or a *Lapse Rate Curve*.
Table Variables	Tables Variables are variables made up of a table of values, essentially a two-dimensional array. An example could be a *Mortality Table*, giving the probability of survival for a given policy holder age. The table could have 4 columns representing a combination of gender (male / female) and smoker status (smoker / non-smoker).

3.1.2.3. Derived Variables

Derived Variables are variables that are derived, via a formula, from other variables. There are at least 2 reasons for creating Derived Variables:

- They simplify the formulae of the Functions of our products.
- They improve efficiency by evaluating the formula only once, rather than being evaluated for each product instance.

There are 4 types of Derived Variables:

Derived Variable Type	**Description**
Derived Instance Variables	A *Derived Instance Variable* is a variable that is derived, via a formula, from other *Instance Variables*.
Derived Single Variables	A *Derived Single Variable* is a variable that is derived, via a formula, from other *Single Variables*.
Derived Series Variables	A *Derived Series Variable* is a variable that is derived, via a formula, from other *Series Variables*.
Derived Table Variables	A *Derived Table Variable* is a variable that is derived, via a formula, from other *Table Variables*.

Examples of Derived Instance Variables could be:

- a series of *Discount Factors* derived from a *Yield Curve* (series variable).
- a series of future *Escalated Expenses* derived from the current *Expense* (single variable) and an *Inflation Curve* (series variable).

3.1.3. Functions

Functions are calculated values of a product, represented by formulae. As stated at the beginning of this document:

> *"mguchiQ is a forward-looking calculation engine that calculates the values of the functions of a product for time periods out into the future"*

Each function is therefore related to a time period t.

Examples of functions could be:

- **CurrentPolicyHolderAge**(t) = FLOOR(**AgeAtInception** + ((**DurationInForce** + **t**) / 12))

  Where:

  - **AgeAtInception** is an Instance Variable representing the policy holders' age when they took out the policy.
  - **DurationInForce** is an Instance Variable representing how long (months) the policy has been in force a time t = 0.

- **EscalatedAnnuity**(t) = **Annuity** * ((1 + **AnnuityEscalation**) ^ **PolicyAge**(t))

  Where:

  - **Annuity** is an Instance Variable representing the base guaranteed Annuity.
  - **AnnuityEscalation** is an Instance Variable representing by how much an Annuity will escalate annually.
  - **PolicyAge**(t) is another Function of the product representing the age of the policy at time t.

3.1.4. Conditions

Conditions allow us to create additional result sets that will only include portions of our data set based on **conditions** that we specify in our model.

An example of a **Condition** could be to partition our results by all policy holders under the age of 50 and all policy holders over the age of 50.

3.2. Scenarios

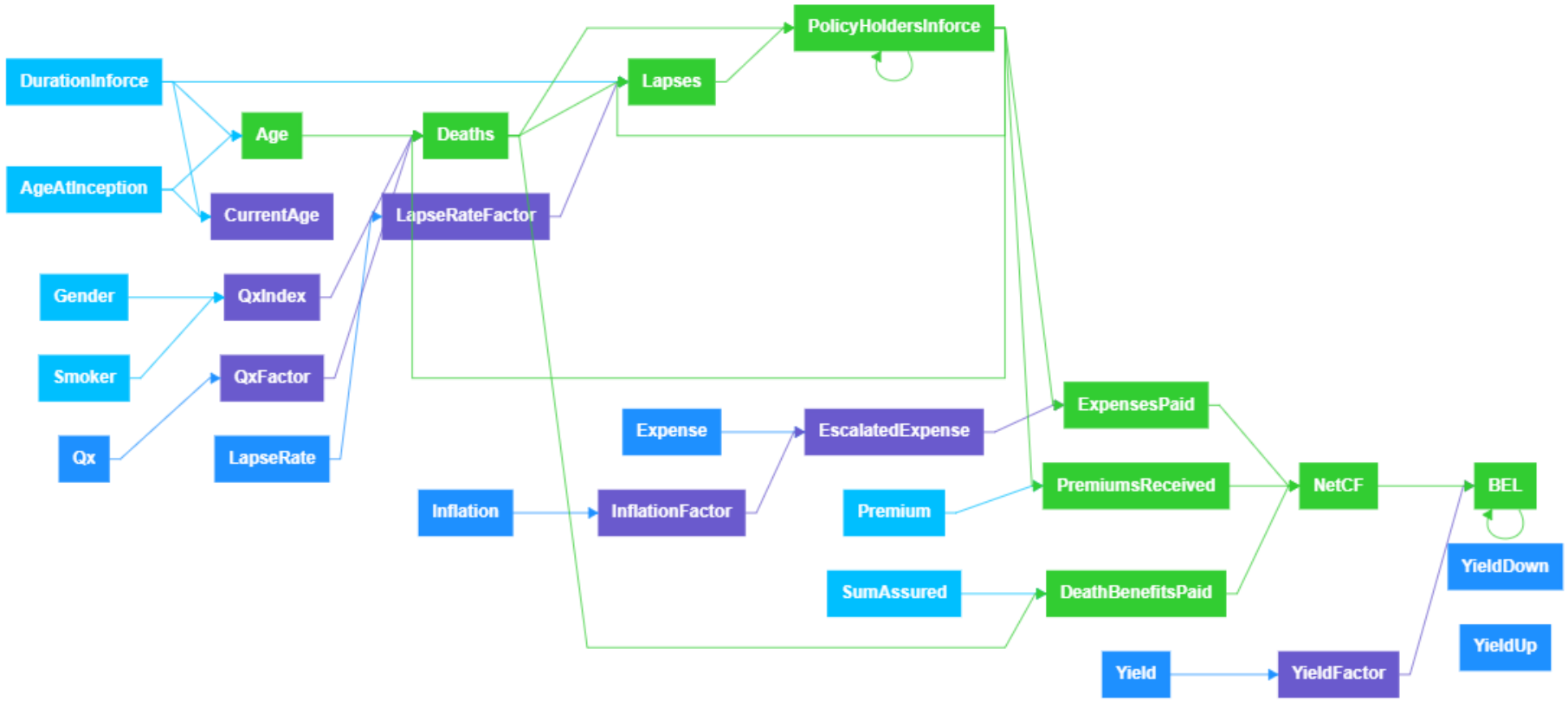
Scenarios describe changes to **Non-Instance Variables** so that you can see how these changes affect your results. An example of a scenario could be a change in our investment assumptions (possibly represented by a change in the yield curve), and / or a change in our mortality assumptions.

Scenarios consist of a set of **Scenario Adjustments** – each **Adjustment** indicates how to change a specific **Variable**.

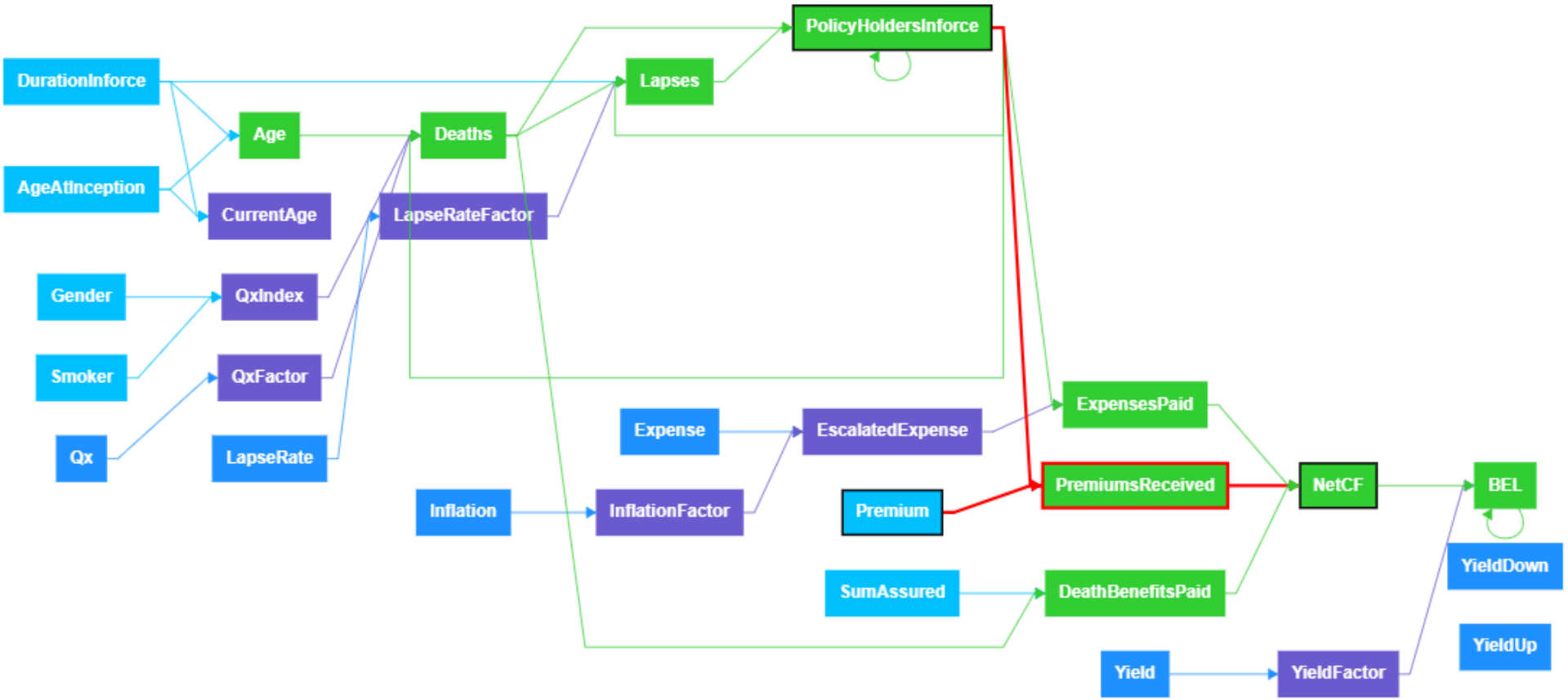
As part of defining a scenario we can also specify additional properties of **functions** that determine how results are aggregated for the scenario. An example could be to only aggregate results for *positive* scenario variances.

3.3. Model Visualisation

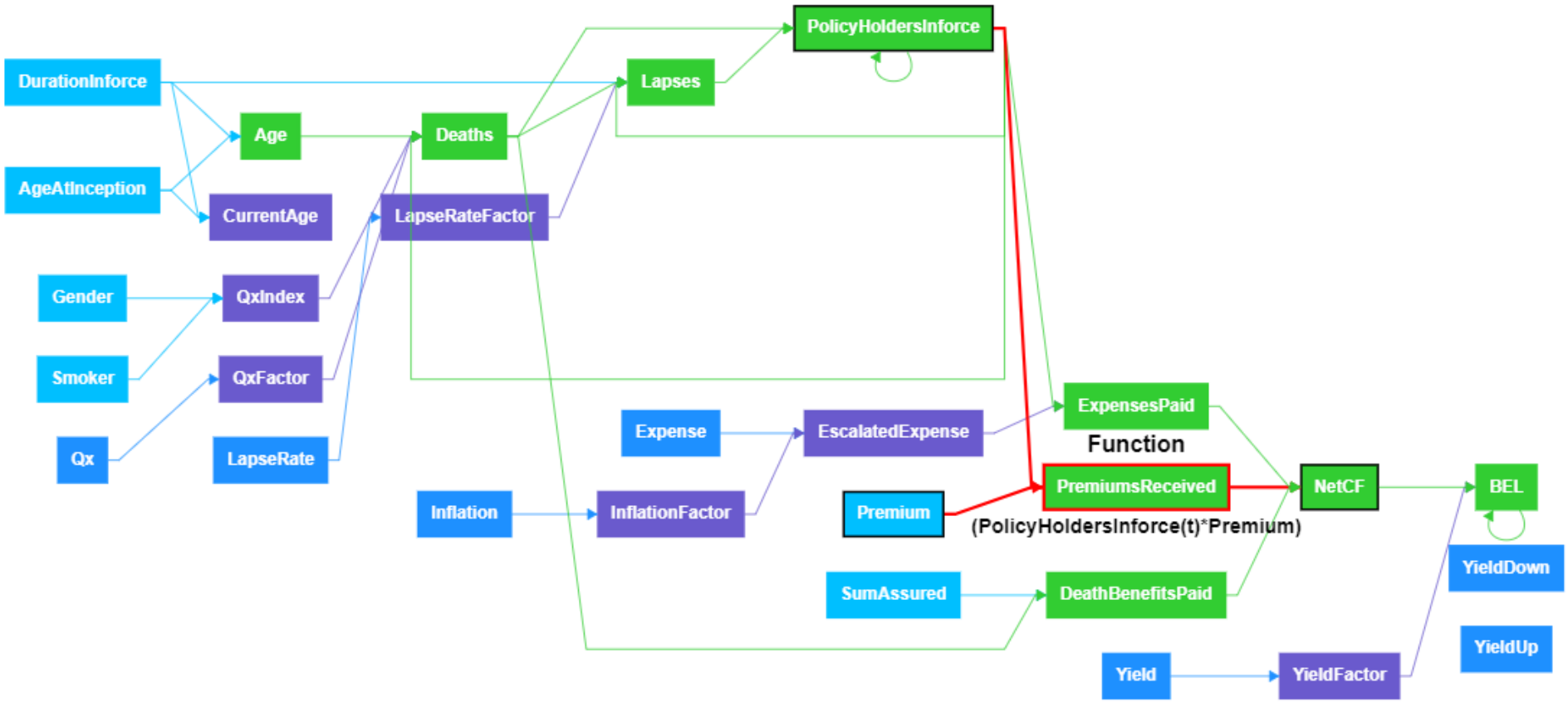
Once a model has been successfully built, mguchiQ will allow you to see a visual representation of the model which can be useful in understanding and debugging a model. Below are some screenshots of the visualisation of a model:



Model visualization - different items types (variables, functions, etc.) are coloured differently



Clicking on an item will highlight all the flows into and out of that item



Hovering over an item will bring up useful information about that item, such as its formula

4. Calculation Types

mguchiQ can currently create the following different calculation types:

Type	Calculation	Description
General	Check	Checks the integrity of the data.
Deterministic	Aggregate	Performs calculations on individual product items (e.g. policies) for a range of time periods (t) and aggregates the results into a result set. The aggregated (book level) results are stored.
	T	Performs calculations for a given time period (t) and stores results per individual product item (e.g. policy). This allows an expansion per product item of the values calculated in the **Aggregate** result set for a chosen time period (t).
	Product	Performs calculations for the specified product item (e.g. policy). Also creates an Excel spreadsheet of the calculations for the specified product item. Note that the Excel spreadsheet is populated with actual formulae (not values) so that calculations can be easily audited.
	Goal seek	Goal seeks a particular **Function** for each individual product item at a given time period (t).
Stochastic	Aggregate	Performs an aggregate calculation for each scenario in a stochastic data set, with each scenario's values being stored. Results can be extracted either as the average of all the scenarios or by selecting different percentiles of the calculated data.

5. Data

When executing an mguchiQ Run it is necessary to supply data that will be fed into the calculations you require. There are generally 2 types of data requirements:

- **Instance Data** - the individual product data items, such as a *book of insurance policies* (sometimes referred to as *model point data*).
- **Non-instance Data** - supplementary data that is not specific to a product instance, such as *mortality tables* or *yield curves*.

mguchiQ allows for various ways of specifying data for a Run:

- Via an **Excel** spreadsheet – this makes it very easy to test a Model independently of having access to any external data sources.
- via an **SQL** query from an SQL data source.

Any combination of Excel and SQL data sources can be combined in specifying the data for a Run.

6. Runs

A **Run** is the process of submitting a request to mguchiQ to calculate some values. The parameters of the **Run** are specified via an Excel spreadsheet that contains information such as:

- The **Model** to be used.
    - Which **Functions** of the Model to calculate.
    - Which **Conditions** of the Model to calculate.
    - Which **Scenarios** of the Model to calculate.
- The **Result Type** required.
    - Any specific details pertaining to the Result Type requested.
- The **Instance Data**, such as a set of policies. This data can be included in the Excel spreadsheet or specified externally.
- The N**on-Instance data**, such as yield curves and mortality tables. This data can be included in the Excel spreadsheet or specified externally.

The Excel spreadsheet containing all the Run details is upload to mguchiQ via the mguchiQ website.

Calculations occur in the background, and once complete, results are available from the mguchiQ website.

7. MaxT

To recap:

mguchiQ is a forward-looking calculation engine that calculates the values of the functions of a model for time periods out into the future. It produces results along the following lines:

For every **item** (e.g., policy) in a given data set *(e.g., a book of policies)*

For each **time period** (t) from 1 to **n** *(e.g., a number of months)*

For each **function** in the model

Calculate the **value** of that function

There are therefore the following potential number of calculations in a result set:

Number of calculations = Number of items in given data set * **n** * number of functions

So, for a data set of 100,000 items (e.g., policies), a **n** of 1,200 (e.g., 1200 months, or 100 years out into the future), and 50 functions in our model we would potentially have:

100,000 * 1,200 * 50 = 6 billion calculations

This is a large number of calculations and for efficiency we must make every attempt to reduce the number of calculations without affecting our result set, thus the value of **n** becomes very important. The value of **n** needs to be such that when calculating future values all **significant** future values are incorporated. To be safe we could attach a value of 150 *12 = 1800 to **n** as no one has ever lived longer than 150 years. But this will be hugely inefficient as we would be doing a large number of calculations that have no real significance to our answers. Possibly a value of 100 * 12 = 1200 would be more appropriate. This may be the right value for a policy holder than is currently 20 years old but for an 80 year-old policy holder we would again be calculating a lot of irrelevant information above, say, **n** = 480 (40 years), where the policy holder would be 120 years old.

In mguchiQ we refer the value of **n** as **MaxT**, the maximum number of future time periods t.

We unfortunately do not know the exact age profile of our policy holders before we initiate a run so determining the correct value for **MaxT** is not straightforward. When specifying a model **MaxT** is specified per product type, meaning you can have a different **MaxT** for different product types. So, we could, for example, have a **MaxT** of 1200 (100 years) for a **Life** product (as we may expect to have policy holders from 20 years old) and a **MaxT** of 840 (70 years) for an **Annuity** product (as we may expect our minimum age for **Annuity** policy holders to be 50 years old).

An even more efficient way of specifying **MaxT** is via a formula. If **MaxT** is specified via a formula then it is calculated per policy instance, meaning each policy has a different value of **MaxT**. An example formula could be something along the lines of:

*MaxT = MAX(1,(101-CurrentAge)*12)*

A **MaxT** specified in this way would run all policies up to each policy holder being 101 years of age.

Quick Reference Guide

mguchiQ Item	Description
Model	A description of the data and calculations required to represent one or more financial products.
Products	A financial product, such as a *life policy*.
Constants	Used to make formulae more readable.
Variables	Data required for calculations.
Instance Variables	Data specific to a particular instance of a product, such as the *premium* of a policy.
Non-instance Variables	Data external to any particular product instance, such as a *yield curve*.
Single Variables	A singleton data item, such as the current *prime interest rate*.
Series Variables	A one-dimensional array of data, such as a *yield curve*.
Table Variables	A two-dimensional array of table, such as a *mortality table*.
Derived Variables	A value derived from one or more Instance or Non-Instance Variables.
Derived Instance Variables	A singleton variable derived from other instance and / or non-instance variables.
Derived Single Variables	A singleton variable derived from other non-instance variables.
Derived Series Variables	A one-dimensional array of data derived from other non-instance variables.
Derived Table Variables	A two-dimensional array of data derived from other non-instance variables.
Functions	Formulae representing the calculations you require.
Conditions	A grouping of product instances for result aggregation purposes.
Scenarios	A change to the non-instance input variables of a run, such as a *shift* in the *yield curve*.
Scenario Adjustments	The specific non-instance variable changes that make up a scenario.
Calculation Types	The different calculations that can be performed.
General	
Check	Checks the integrity of instance data.
Deterministic	Deterministic run types
Aggregate	Performs calculations on individual product items (e.g. policies) for a range of time periods (t) and aggregates the results into a result set. Only the aggregated (book level) results are stored.
T	Performs calculations for a given time period (t) and stores results per individual product item (e.g. policy). This allows an expansion per product item of the values calculated in the Aggregate result set for a chosen time period (t).
Product	Performs calculations for the specified product item (e.g. policy). Also creates an Excel spreadsheet, with formulae, of the calculations for the specified product item enabling all calculations to be easily audited.
Goal Seek	Goal seeks a particular function for each individual product item at a given time period (t).
Stochastic	Stochastic run types.
Aggregate	Performs an aggregate calculation for each scenario in the stochastic data set, with each scenario's values being stored. Results can be extracted either as the average of all the scenarios or by selecting different percentiles of the calculated data.
MaxT	The maximum number of future periods to use for calculation purposes.