

Advanced Modelling in RiskAnalytics

Case Study on the GIRA Model



PillarOne Community Event Berne, September 2nd, 2011

Stefan Kunz, Intuitive Collaboration



- Defining a model
- Complexity drivers
- Examples based on model General Insurance RiskAnalytics (GIRA)
- Migration

- Which decisions should be supported by the model?
 - include all possible stakeholders to collect their needs and to make sure they understand *their* model
 - needed results and their granularity
 - can the result key figures be derived from the input parameters?
 - is the available input data sufficient to determine the necessary input parameters?
 - derive required time concept
period, cash flows, valuation dates ([separate presentation](#))

- Draw a model graph – the upcoming model builder might help!
 - define component functionality
 - required parameters and information of preceding nodes
 - produce results and information for following nodes
 - super vs. many small and reusable components
 - providing flexibility for the model user

Defining a Model: One Model only?

- a chain of models: discounted cashflows ,post processed` in a follow up model
- visibility of parameters and results of a model?
 - model composition including cross model dependency modelling might be a practical approach
 - access restrictions might influence a model
- is there a possibility to automatically retrieve parameters from other models/IT systems (accounting, RMS)

- traverse the graph from the **end** in order to define the granularity of required information for every node
- unique **names** for modelling items and parameters
missing names, unclear names are warning signs that should not be ignored!



- **Models** define the directed graph by defining a number of components and their relations. Traversing the graph can be data driven. A model can be stochastic and/or deterministic.
- A model itself is not run-able as it contains no **parameters**. Parameters can trigger the behaviour of a model by changing the number of components and applying different strategies (i.e. distributions, r/i contracts) within a component.
- **Result templates** define paths and granularity (single, aggregate, drill down) of data collected. Simulation runtime highly depends on the amount of collected data.

Depending on used components models tend to be quite flexible and their behaviour varying a lot depending on a parameterization. Therefore some users tend to talk of model templates and use the term model for parameterizations.

- Reduction of complexity already during specification
 - granularity should be adequate given the required decision support
 - look at effective risk exposure
 - consistent across different areas (claims, premium modelling)
 - simplify whenever extra complexity is not really providing deeper insights

- Refactoring and migration

- Reduction of complexity, safety measures
 - hierarchies, nesting
 - design by contract applied to components
incoming and outgoing data has to fulfil a contract ([GContracts](#))
 - don't provide functionality increasing complexity before it is fully specified. Keeping it simple makes refactoring and adding additional functionality afterwards easier
 - avoid loops
loops within the graph are allowed only for allocation purposes
- Complexity is increased by
 - true multi period models with carry-overs as data structures become much more complex
 - date calculations (lapse years, ...)

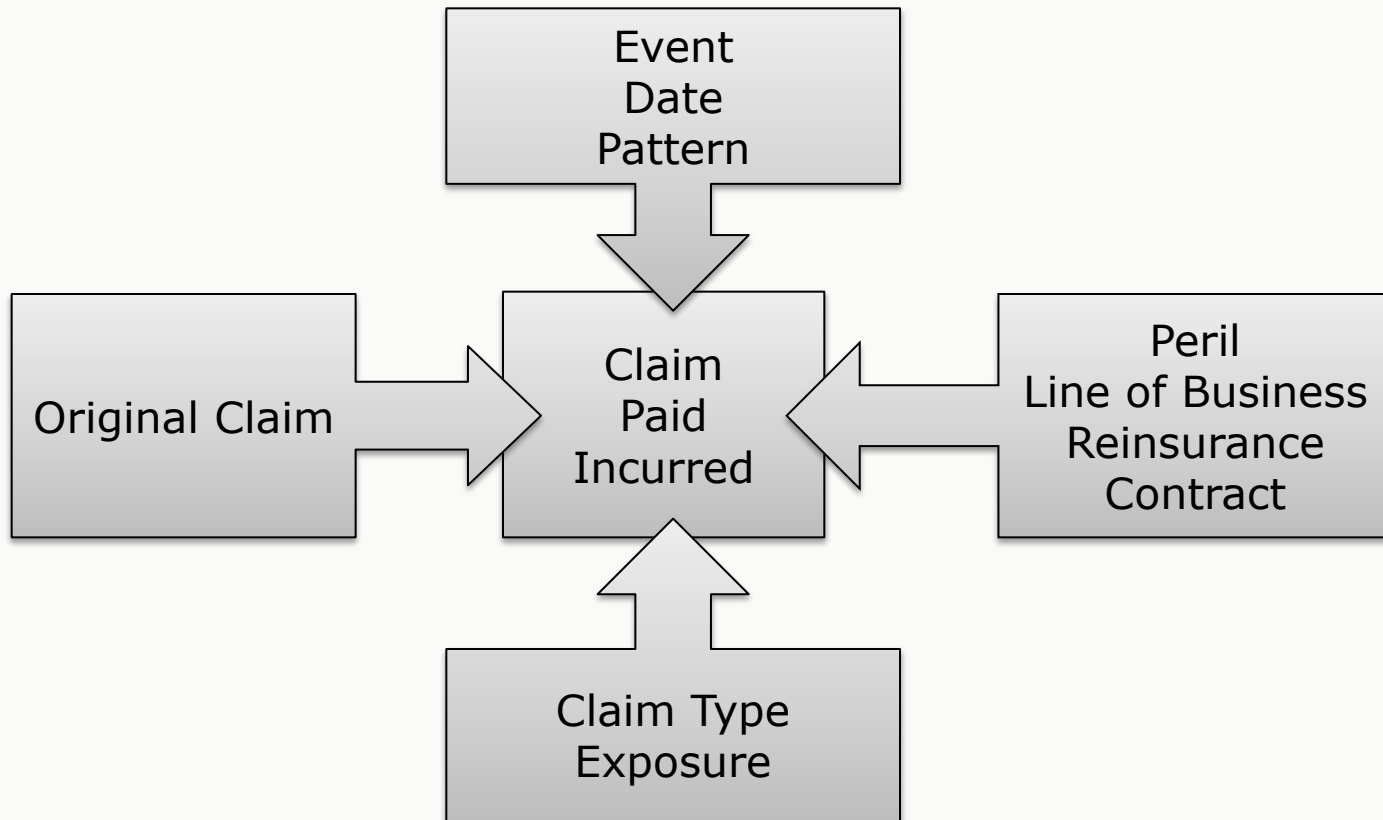
- Cashflow modelling: patterns, indices and yield curves for discounting
- Premium cycles, business planning using indices
- Multi legal entities including simple default modelling capabilities
- Multi period
- Multi period reinsurance including
 - stabilization of XL contracts
 - multiple counterparties
 - inward/outward modelling
- Increased dependency modelling capabilities
 - mixing global and local effects
 - correlating frequencies, events, indices

- There are several updates of a claim in different periods
- Conflicting interests: ultimate generation approach, stochastic development
- R/I contracts need to know the whole history of a claim:
 - did it already exceed the threshold in a former period, is the per claim cover for this claim already used up?
 - stabilization is done on the full history
 - acting independently on ultimate, reported and paid claim as claims of different sources will have different patterns
 - first come/first served: applying cover according to updated claim information
 - analysis of contract covered splitted up by legal entity, segment and peril

First Come/First Served

- Illustration of first come/first served and acting independently on ultimate and paid

Time						
[months]						
		6	12	24	48	
	Occurrence	Ultimate	Paid			
Pattern A			0.5	0.2	0.2	0.1
Pattern B			0.3	0.6	0.1	0
Peril A	01.01.2011	1000	500	200	200	100
Peril B	01.04.2011	1000	300	600	100	0
XL 400 xs 600						
Peril A ceded		400	0	100	0	0
Peril B ceded		0	0	300	0	0



- Please have a look at Arnold Wassmer's slides to get the full details.
- Using indices, payout and reported patterns lead to an over-determined system. Therefore patterns are expressed before application of indices and effectively applied patterns will be different.
- Indices can be specified with specific dates, patterns on a monthly basis. This requires special aggregation logic for a claim packet.

- Models are not developed for eternity
- Challenges
 - parameters should be transformable to new model versions without conflicting with process features
 - what happens with a parameterization in the workflow state 'in production'?
 - results: a new model version will most probably produce different paths and values. Users need support to easily identify differences and decide if they are intended or not
 - comments pointing suddenly to nowhere
 - auditing by logging automated migration steps
- Migration support by a suitable workflow

- The specification process has to consist of several elements:
 - regular workshops to define the big picture and review detailed specification
 - specification in spreadsheets format don't expose all thoughts to the developers: especially rejected solution ideas are missing and they might be interesting for a full understanding of a solution
 - correct impact identification of specification changes
 - keeping a model consistent always
 - avoid shortcuts in the development process

- <http://www.pillarone.org>
 - presentations
 - screen casts
 - try it online
 - manual
- Issuetracking <https://issuetracking.intuitive-collaboration.com/jira>
- Buildsystem <https://build.intuitive-collaboration.com/jenkins/>
- Code Repository <https://github.com/pillarone/>

Thanks!



Stefan Kunz; Partner, Actuarial Tools
+41 44 926 14 07; +41 76 370 31 86
stefan.kunz@intuitive-collaboration.com
Skype: stefan-kunz

Intuitive Collaboration AG; Seestrasse16; CH-8712 Staefa