Mo.net
Lowering the Modelling Platform Migration Hurdle

November 2020
Revision 8

Background

The early-1990s saw a revolution in personal computing, with computers appearing on desks in the office and in the home. Activities which had historically been the preserve of men in white coats were for the first time within reach of mere mortals.

The actuarial community was quick to appreciate the potential opportunities afforded by personal computers, and numerous actuaries took their first tentative steps into the world of programming. Using relatively exotic languages like APL, FORTRAN77 and COBOL ported from mainframe environments, these budding developers started to build the first generation of financial models.

Software vendors and consultants also spotted an opportunity to capitalise on the emerging power of desktop computing by developing proprietary software specifically for those actuaries without the inclination to code models from scratch. This led to the appearance of the first generation of financial modelling platforms in the mid-1990s. Most of these platforms still exist to this day, with largely the same architecture, although the vendors themselves have changed through various mergers & acquisitions.

Legacy Debt

Unfortunately, the first generation of financial modelling software required offered little in the way of model development governance, enterprise integration or operational robustness. The actuarial modelling community was focused more on building ever-more elaborate solutions to solve the latest business or regulatory reporting challenge than making sure those solutions were scalable, transparent and resilient. The result was, and still is for many organisations, a complex web of opaque & in many cases stovepipe modelling code supporting an increasingly demanding business environment.

Even in today's highly governed & regulated reporting environment, the underlying financial models are typically just evolutions of the first-generation models. This presents insurers with a range of potential problems.

- Fragmentation of models, data, and associated systems / processes
- Complexity of the modelling estate - increased cost of change
- Layering and complexity of often tightly coupled processes
- Variety of approaches, methods, standards
- Dependency on still relatively exotic skills or specific implementation knowledge
- Limited scalability and opportunity to embrace new technology paradigms / design patterns

Stickiness

The inherent complexity of the existing financial modelling estate in use at most insurers, coupled with the almost continuous regulatory change agenda over the last 20 years, has provided little or no opportunity for refinement / optimisation.

Furthermore, the inherent cost & risk of change required to unlock what are assumed to be marginal benefits in control, performance, and productivity has allowed the first-generation modelling platforms to remain very sticky components of the actuarial enterprise. Any modelling platform migrations that have taken place have typically been carried-out in the context of model consolidation initiatives, usually as the result of M&A activity, where cost or resource reduction is the primary consideration.

The consequence of this is that modelling technology & solutions originally developed more than two decades ago is still in widespread use across the life insurance industry today. And with most modelling software vendors having little incentive to innovate, there is now little to differentiate the financial modelling platforms from one another.

Despite this, there are significant potential benefits on offer for those who are willing to challenge accepted wisdom and understand the true cost & opportunity of financial modelling platform migration.

Migration Dimensions

While the core projection models are usually the focus of people's attention when considering a modelling platform migration, in reality this element is just one of many often tightly coupled dimensions to consider. In many cases, the models themselves are actually the most discrete / portable component and are therefore the least complex, costly and risky to migrate to an alternative platform.

Other parts of the modelling ecosystem are typically owned & supported by other business functions outside the direct control / influence of the actuarial function. Getting buy-in for any change beyond the scope of the core modelling functionality may therefore be difficult and / or costly. As a result, the development of any business case for migration often stalls before the potential benefits are determined.

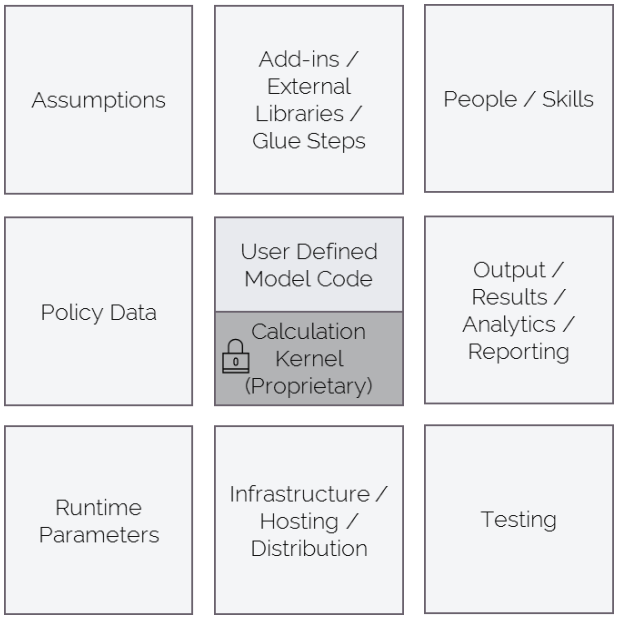


Figure 1 - Typical Dimensions of Financial Modelling Platform Migration

An Alternative Paradigm

The loosely coupled, open architecture of Mo.net makes migration from competitor platforms a realistic & tangible possibility. While each of the competitors obviously include a sizable element of proprietary code, which cannot be accessed / migrated directly, the majority of this is simply an encapsulation of actuarial, statistical or financial mathematics. Any remaining vendor code is largely to provide generic features & functionality such as read / writing files, transforming data, distribution of workload, visualisation of results, etc. The rest of the model code is yours, and you are free to do with it as you wish, including migrate to an alternative platform.

The Mo.net architecture & specifically the calculation kernel enables models developed in competitor platforms to be completely consumed with a modest amount of effort. In reality, the fundamental components of the Mo.net platform are very similar to competitor platforms, albeit implemented using a more powerful, accessible & modern technology stack.

Furthermore, the flexibility of the Mo.net platform – specifically connectivity to legacy data sources / targets - allows all other components of the modelling ecosystem to be left largely unchanged, at least initially. This enables a focus on core model migration while leaving existing data, assumptions, output, and associated processes / dependencies untouched.

Even without optimising other components of the modelling ecosystem, simply porting core model logic across to Mo.net will usually unlock immediate performance, transparency and flexibility benefits, simply because of the advantages of the modern, loosely coupled platform architecture.

Code-Based vs Library-Based

The universe of financial modelling platforms polarises into two distinct groups, each with their own benefits & shortcomings. Code-based platforms provide almost infinite modelling possibilities, but require users / modellers to develop models from scratch or from "accelerator" template models On the other hand, library-based systems provide a range of standard functionality "out of the box", but are difficult to customise for specific modelling needs and require the bulky libraries to be carried around for even the most simplistic modelling applications.

Suggested Approach

Our advice is to focus on porting projection models that are stable and will not require enhancement / modification for the duration of the migration. While there is often a desire to improve / enrich models during any change or migration programme, this makes testing / reconciliation a challenge.

Mo.net includes numerous features that help ensure the migration of models is as efficient and well controlled as possible. These include, but aren't limited to:

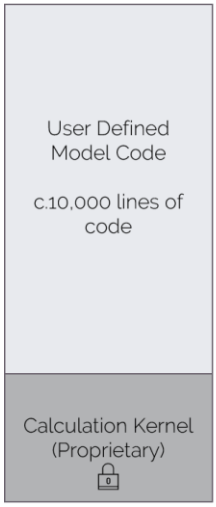
- Huge range of core / specialist modelling functions, implemented on top of the highly performant & scalable .NET Framework foundation
- Best in class interactive model debugging
- Code / object reuse
- Near-instantaneous model rebuilds
- Visualisation of model structure
- Code completion technology

Once the core models have been ported across to Mo.net, reconnected to other elements of the modelling ecosystem, and proven in an operational environment, other components of the landscape can be refined / optimised / upgraded to unlock the real potential offered by the Mo.net platform core.

Sizing-up the Problem & the Opportunity

Before embarking on any model migration programme it's worth understanding the size, not only of the challenge ahead, but also of the potential benefits on offer. The example below outlines the potential level of effort (and indicative cost) required to port a model from a code-based competitor platform to Mo.net. These metrics are based on a real-world migration project carried out recently by one of our existing clients.

Note: The costs below do not include the additional effort & associated costs of Mo.net platform licenses, platform integration, testing or training. The exact cost of these will be dependent on the specific needs / existing capabilities of the existing modelling environment.



Size of Source Model User Defined Code Base (C++)	- 10,000 lines of code
Assume 1 line of original model source code translates to 1 line of Mo.net (VB.NET) code	
Translation / porting rate	- 250 lines of code per day
Time to port	- 10,000 / 250 - 40 days
Cost of user defined code port	- £0.35 per line of code (offshore)
Total cost of user defined code port	- 10,000 x £0.35 - £3,500

Additional tasks / costs (example)
- Mo.net Platform Licenses
- Target Model Design
- Platform Integration
- Testing
- Documentation
- Training

Further Information

For more information regarding the Mo.net Financial Modelling Platform and to discuss the potential benefits of migrating your legacy modelling platform to Mo.net, please get in touch

Software Alliance Limited
30 Stamford Street, London, SE1 9LQ
Tel: +44 (0) 20 3964 2755
www.softwarealliance.net
hello@softwarealliance.net