Rethinking the Modeling Ecosystem: A Modest Proposal

BY RYAN SCHULZ, JONATHAN 'WES' WEST, AND STEVE TURNER

EXECUTIVE SUMMARY

For banks in many countries the current model development and validation process is broken: it takes too long, there are too many nonstandard elements, there is too much rework, and it introduces material risk ("model risk risk", if you will) when critical and time-sensitive models fail validation. It also costs an exorbitant amount relative to the benefit it provides to shareholders. In our experience, the major challenges underlying this are threefold:

- Developers must create an entire evaluation framework of analyses, stats tests, and assumptions under which to evaluate models before they can begin modeling
- Developers and validators focus on different things: developers are attuned to the competing priorities of business rationality, applicability, and statistical validity whereas validators' primary focus is on statistics
- Validation expectations are articulated generically and rarely explicit. The different focus of the groups frequently results in disconnects where developers either focus on validators' lower-priority issues or perform tests at a level of rigor different than what is expected

There must be a better way! A better way would overcome these challenges where Validation takes on separate roles at the front and the back of the development process. That is, Validation establishes a new engineering role along with its current quality control responsibilities. This would reduce failure risk by:

- Streamlining the validation process: validators would need less per-model review time if they were already familiar with the standard processes used to test the models
- Accelerating the development timeline: developers provided with a standard set of baseline evaluation criteria can more confidently develop models without expending material time and energy building an artisanal evaluation framework

OBSERVATIONS ON THE CURRENT MODEL DEVELOPMENT AND VALIDATION PROCESS

The model development process has evolved significantly since the global financial crisis and banks have achieved a high level of confidence in what Validation produces. Many of the models and model development processes were created to meet regulatory requirements that have been promulgated through international bodies like the Basel Committee for Banking Supervision and in US laws like Dodd–Frank. In these and many similar cases, management direction was to spare no resources, get it done, and make sure requirements are met.

Under these conditions there were not a lot of questions on, "Is the process efficient?", "Are our resources being used most effectively?", "Are common practices being established for similar modeling types?", and so on. Which leaves us where we are today where the problems with the current process of validating models are prodigious: it takes too long from beginning through validation and into implementation; there's too much back and forth with many models having to be reworked after submission to Validation; and the engagement model is fundamentally broken as it was not designed within the right context.

The core problem resulting from this evolution is that the development team is expected to create, whole-cloth, an evaluation framework in which models will be built and judged. Validation then reviews the framework — in toto — and identifies every insufficiency in the modeling universe, from unclear data controls to insufficient theoretical support for statistical testing to model accuracy analysis. The validation review is presented to the development team, often with the explicit instruction to effectively start over, but this time with some incomplete negative guidance (i.e., guidance on what not to do) which is used to try an alternative approach. In CCAR modeling that means there can often be 200+ pages of documentation per model and eight or more weeks of back-and-forth to support a single equation that isn't more complicated than "balance growth is a function of GDP and the 3M Treasury rate". This cannot be the best use of scarce bank resources.

In this approach, the developers create not just the equation, but also the evaluation framework under which that equation will be judged. This means that the developer needs to:

- Identify the problem to be solved
- Figure out when, where, and how to bring the problem's business owner into the process
- Determine the right modeling (or other analytic) approach to answer that problem
- Identify the necessary statistical tests to verify proper application of the approach
- Settle upon acceptance criteria and thresholds for these tests and figure out work-arounds or mitigants if statistical tests fail
- Create an accuracy testing framework and similarly define acceptance rules and mitigants

These decisions, among others, occur before a single test equation is even developed. Conversely, these choices are made after a definition of "data reconciliation" is created; modeling data sources are identified; and reconciliation to a "source of truth" is conducted.

This development process — which can take well upwards of six months — culminates in the validation review, where the development package is reviewed from start to finish: every one of those decisions, choices, analyses, and assumptions is scrutinized to determine if appropriate diligence was applied in the creation of an equation. In nearly every case, validators find at least one thing wrong with the models (we have yet to see a development effort where every "t" is crossed and every "i" is dotted). If the criticism is strong enough that a model is a no-go, and the weakness in process is sufficiently up-stream, then every piece of work made after that decision over the last three, five, or more months is wasted: the entire effort needs to be reconsidered.

What we have most commonly seen is that Validation has a set of tests they apply and a set of heuristics to those tests to ensure the model passes muster. However, the conveyance of these tests and heuristics to the development team happens sporadically, opaquely, and too late to influence the outcome, resulting in failed models and consequent rework required to bring the models to muster.

Too often, these mistakes are the result of an inconsistent view of importance: the model fails a statistical test that has no bearing on practical applications; the data is the best available but fails to tie out at some degree of precision to a verifiable source of truth; acceptance criteria for model performance changed between the start and the end of the development window; documentation is unclear.

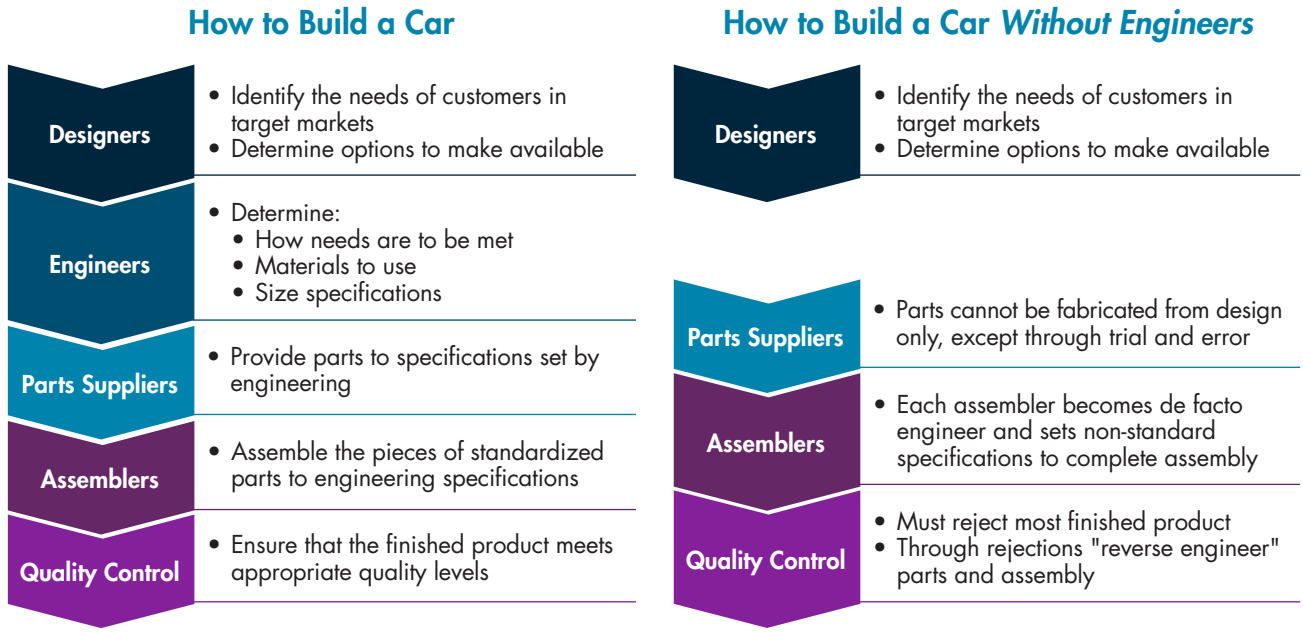
CHANGING THE PERSPECTIVE

What if bankers had started from a different perspective by thinking about the problem not as a statistical modeling problem but as a business problem that needed to be resolved efficiently and effectively. What bankers would then do is look at each stage of the planned process, determine what needs to be done, figure out ways to do it with the least effort, set rules on data source usage, and establish common development standards. Bankers would look at information needed in the early stages of the process and see where that information was appearing and, if out of alignment, change when and where the information surfaces.

To us, this sounds like a complex production process which can be visualized as a "model factory". If the process is viewed from a model factory perspective, there would be different levels of attention (read: money) paid to each stage of the process with "designers", "engineers", "parts suppliers", "assemblers", and "quality control" each having a role different than is what occurs today.

FIGURE 1: The Necessity of an Engineer

Lessons from the auto industry.



DESCRIBING THE SOLUTION

What we see as the solution is for Validation to take on two roles—engineer at the front of the process and quality control at the end — resulting in a redefinition of the guidance that Validation provides to developers.

With these roles filled, banks should see fewer costly restarts due to "incorrect" early decisions, which currently waste months of developer time. Weeks or months of a validator's time that were previously wasted inefficiently replicating or evaluating the basic standards of the modeling framework are now free to focus on complex problems. Elimination of delays driven by rework that could cause severe risks like rushed models or simple assumptions going into CCAR submissions, failure to meet promised results to regulators, or — getting away from regulatory application — the continued use of out-of-date risk or trading models.

Banking: How to Build a Model with Engineers

Designers	Identify the business problem to be resolved and the non-statistical considerations that will impact a model
Engineers	Define the modeling techniques and model evaluation criteria, along with setting data hygiene and documentation standards. These rules should be finalized before modeling commences. This role is currently held by modelers but should be held by Validation
Parts Suppliers	Provide clean, serviceable, and for-purpose data on which models will be built. In the absence of Validation-led rule setting, though, inconsistencies emerge in decisions that range from very big (acceptable data sources) to very small (should models be built on end-period or month-average observations)
Assemblers	Execute against a set-in-stone list of design and delivery guidelines to develop models that address the business problem. Without Validation pre-committing to evaluation criteria, however, work at this stage is often wasted
Quality Control	Double-check that the work created by the assemblers appropriately uses the inputs provided by the parts suppliers while adhering to the rules set forth by the designers and engineers. In today's model, however, validators review the acceptance criteria at the same time they review whether the models adhere to the acceptance criteria. As noted above, if Validation and developers cannot agree on the rules of the road, all subsequent work is wasted

Most of the failures identified above are driven by developers and validators speaking different "languages", with developers driving to business objectives and with validators focusing on statistical best practices. To bridge that divide, we think it necessary for Validation to offer a stronger hand in defining the development approach: how should one validate data against a source of truth; which statistical tests are necessary versus nice to have when building models of certain types; how is accuracy defined?

Validation should provide both the tests or methods that should be used, but also a prescription for what is "good enough".[1] To be specific, consider how accuracy tests are determined today compared with a situation where engineering requirements are set upfront. Once a model has been built, its accuracy needs to be assessed. However, the specific method for evaluating accuracy is an open field. We have seen the following (and more):

FIGURE 2: Validation Choices

Sample range of accuracy measurement methodologies.

METRICS	MEASURED OVER TIME PERIOD	AT GRANULARITY OF
R^2	Full modeling dataset	Components (e.g., acquisitions vs. runoff)
Adjusted R^2	"Training" window	Sub-products, single model
MAPE	"Holdout" window	Products, aggregation of multiple models
MAE	Rolling 9Q basis	LOBs, aggregation of multiple models
Etc.	Crisis window	
	Etc.	

Any of these are "right", although they all have their advantages and disadvantages. For example, MAPE or MAE are excellent at evaluating level of accuracy, but can change wildly based on the granularity of the series being measured: a MAPE of 30% may be very good when predicting monthly account originations, but is offensively bad when predicting an LOB's aggregate deposit balance. The same is true for R^2 and other goodness-of-fit metrics (see the Appendix for additional examples).

To that end, we push validation units to think through the levels at which models could be developed and to customize measures accordingly:

VALIDATION CHARACTERISTIC	POTENTIAL OPTION 1	POTENTIAL OPTION 2	POTENTIAL OPTION "N"
Accuracy Timeframe	Full observed history	Crisis period	…
Accuracy Measure	MAPE	MAD	…
Threshold	3% MAPE aggregated at total product level	50% R^2 when number of accounts >500,000	…

The common refrain is that this sort of validator direction would violate the group's independence. While this is a legitimate concern, we encourage a more practical interpretation:

- Many validation groups already have rules under which every model is evaluated, but only share these as outcomes of the model review process. The explication of these already-existing decision heuristics should have no impact on validation groups independence
- Large swathes of model development are comprised of low-value analyses or tests. In these cases, it should be acceptable to pre-define acceptance/failure rules without threatening the validity of the model and therefore Validation's independence

[1] While not explored in this paper, we also think it would be useful for validation units to more prescriptively outline the background information they need in describing the business problem, impacted products, and other qualitative contextual factors: continuing with the theme of speaking different languages, developers often over-estimate the prevalence of background knowledge about the business problems they are trying to resolve.

- Pre-committing to acceptance criteria only resolves the tests and acceptance criteria: it remains in the developer's hands to describe mitigants in cases of failure

Note that the core thinking of whether the modeling approach chosen is right for the business question, that model drivers are conceptually sound, and that critical statistical features are adhered to must all get attention. To achieve higher efficiency this should occur through streamlining of low-value-add time vortexes that hurt both the development and validation stages of model creation. Standardization benefits validators and developers alike: a consistent look and feel for reporting model development milestones vastly simplifies the cognitive burden on both sides of the table in building acceptable models. Developers no longer need to create an artisanal and ever-expanding universe of evaluation tests and validators no longer need to learn the intricacies of the framework being applied to each model in isolation.

Importantly, not all models will fit into cleanly pre-defined categories. Some models will naturally require more bespoke attention from both developers and Validation — especially as the bank explores challenger modeling approaches. By streamlining the 60%—80% of models that do fit the framework, both teams free up more time and cognitive resources for the truly interesting and fulfilling aspects of the work — solving complex modeling challenges.

CONCLUDING COMMENTS

This approach challenges the common wisdom that the independence of Validation is paramount — the risks of missteps are too great and too predictable for the status quo to remain. Banks could establish a separate engineering function independent from Validation. However, this group would either conflict with Validation or, more likely, codify Validation's criticisms and apply these in a model development rulebook. A better approach is for Validation, which has all of the required knowledge to fulfill an engineering role, to establish separate groups—one to set standards and the other to ensure that those standards are met.

HOW NOVANTAS CAN HELP

What we have outlined here is controversial. That said, a growing number of banks are asking for our help to think through how they can streamline the validation process. We have worked with many of the largest banks in North America, Australia, and Europe to build and evaluate time series, credit risk, and survival models over the last three years across a number of applications, and this broad exposure has given us hands-on insight to the vast majority of validation-related challenges. We have seen what works, what fails, and what fails statistically but still works pretty well at addressing business problems. From this experience, we have developed "cookbooks" for modeling families which document the tests we think are important and those which are perfunctory, and detail every analysis, statistical technique, acceptance criterion, and assumption we plan to apply when building models.

We are now beginning to work with banks' validation teams to define model segmentations, and craft their own "cookbooks" within these segmentations; reviewing the development process and developing pre-commitment guidelines to eliminate points of friction.

Ryan Schulz
Principal, New York
rschulz@novantas.com

Jonathan 'Wes' West
Managing Director, New York
jwest@novantas.com

Steve Turner
Managing Director, New York
sturner@novantas.com

APPENDIX ONE: COMMON ISSUES AND SOLUTIONS

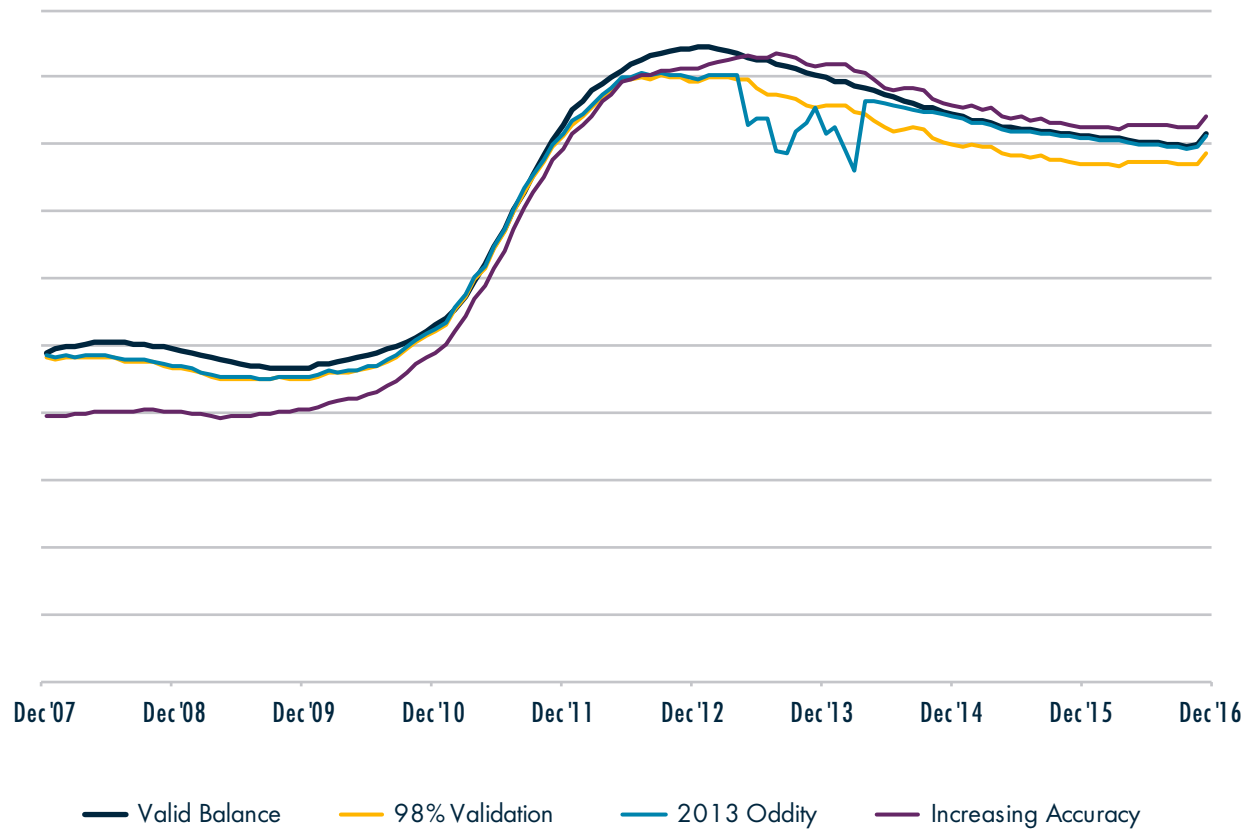
As an example, we use straightforward time series modeling supporting CCAR, since most are familiar with its intricacies, and its rigid annual deadlines provide strong motivation for minimizing rework.

Before any modeling can begin, the modeler must ensure the data they use is complete, correct, and valid. Ideally, the data used should reconcile directly with the General Ledger ("G/L"). However, the data rarely tie perfectly: the modeling dataset may cross multiple systems or bank mergers, reflect outdated product mappings, or be at a processed level of granularity (e.g., customer-level), among other potential permutations, any one of which can preclude perfect reconciliation.

The developer therefore must engage in a reconciliation exercise where they determine what will be reconciled, over what time period, and at what frequency. They must also determine what is an acceptable error threshold. Examine the following graph, outlining three common problems we see: one line is consistently ~98% accurate with the "source of truth"; another is nearly perfectly, except for (weirdly) 2013, when it is off by 10%; another exhibits historic inaccuracy that gets better over time. Which one is good enough for modeling?

FIGURE 3: Deposit Balances Example

Alternative equations for estimating deposit balances.



Source: Novantas Analytic Results

In the current development and validation framework each of these decisions is up to the developer to justify. While the list of potential exceptions are nearly infinite, the Developers should at least be given guideposts by Validation for what is necessary. For example (and note: these are purely illustrative):

VALIDATION CHARACTERISTIC	POTENTIAL OPTION 1	POTENTIAL OPTION 2	… OR SOMETHING ELSE
Frequency	Yearly snapshots as of December	Monthly, if the models are monthly	…
Granularity	LOB-level	LOB-Product-level	…
Accuracy	Average all observations; average inaccuracy 3%	No one period >5% inaccurate	…
Veto Criteria	Failure is "yellow light" with additional analyses proving validity	Failure is "red light" and dataset cannot be used	…

APPENDIX TWO: SERIAL CORRELATION TESTING

When developing a model with time series elements, a critical consideration is whether the model suffers from serial correlation: the property that performance in the prior periods is influencing performance in the current period.

While there have been many papers[2] written regarding the appropriate treatment of serial correlation, there are a handful of camps:

- Include statistically significant exogenous autoregressive ("AR") error terms
- Include every exogenous AR error term going back n periods
- Demonstrate the model passes serial correlation when AR error terms are included, but then exclude them from the final "champion" model
- Evaluate the model using Newey—West Standard Errors, sidestepping the question of serial correlation entirely

We have been witness to holy wars at more than one bank regarding which of the above approaches is the "best" and there is no consensus. It is for that reason alone that the validation unit should help the development team, since the evaluation procedures are entirely different depending on the path one takes and a developer who chooses "incorrectly" can waste substantial effort on models which will ultimately fail.

VALIDATION CHARACTERISTIC	POTENTIAL OPTION 1	POTENTIAL OPTION 2	… OR SOMETHING ELSE
Inclusion of AR Error Term	Include; only if statistically significant	Exclude; test with Newey—West	…
AR Error Sensitivity	Generate forecast with/without AR terms; measure base-case forecast gap after 9Q	Sum AR term coefficients; "yellow light" if sum >0.7	…
Inclusion of AR Error Terms in Actual Deployment	Include AR terms: that was the "champion" model and therefore must be used	Exclude AR terms: the influence is normally small and they are computationally difficult to manage	…

[2] Including our own, "Autoregressive Error Terms in PPNR Balance Modeling", published Feb 9, 2016