Model Risk Management Group

# Model risk mitigation and cost reduction through effective design*

*connectedthinking

PRICEWATERHOUSECOOPERS

# Table of contents

# The heart of the matter

The current financial crisis has drawn increased scrutiny to model risk management. In particular, government regulators, boards of directors, internal and external auditors, and other constituents are demanding more accountability and rigor from financial institutions when it comes to the use of models to guide business decisions, manage risk, prepare financial statements, or produce regulatory reports.[1]

Most of the attention to date has been focused on the independent validation of models.[2] However, the purpose of this paper is to shed light on how certain design choices in the model development process – in particular, choices in the design of the model production system – can significantly reduce model risks with minimal investment. By considering these recommendations during the model design stage, not only can a company reduce a model's overall risk profile and, therefore, the likelihood of model error, it can also reduce the costs associated with model validation, documentation, maintenance, and operational controls (such as SOX 404 controls).

---

[1] See, for example, "Will Regulatory Scrutiny of Model Risk Management Intensify?" PwC FS Regulatory Brief, April 2009.

[2] See, for example, "Model Risk Management: Key Considerations for Challenging Times," Bank Accounting & Finance, June 2008.

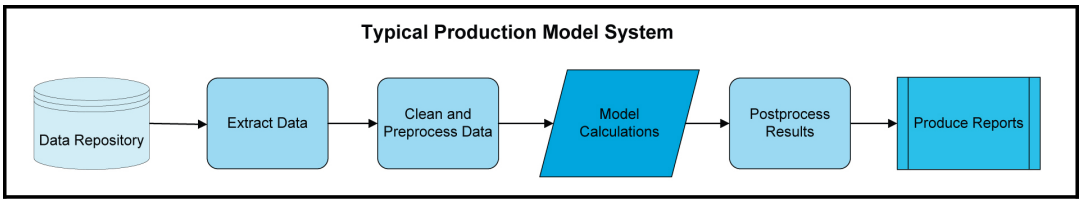
## What this means for your business

# Overview of model application risks

A model is an approximation of a complex real-world process. An effective financial model reduces these complex economic or financial relationships into their most important elements – typically represented by mathematical, statistical, or logical processes and algorithms – in order to estimate a particular outcome or value.

Implementation of sophisticated models frequently occurs within highly complex computer-based production systems consisting of a number of interconnected components. For example, as shown in the diagram below, a production model might be comprised of the following modules:

- A data extraction module responsible for obtaining model input data from a variety of sources – including corporate data warehouses, stand-alone data files, and third-party data feeds.
- A data cleaning, filtering, and pre-processing module responsible for preparing the model input data for consumption by the core model logic. Some of the functions of such a module would include infilling missing or invalid data values, or creating new data inputs by combining multiple data fields based on pre-defined business logic.
- A core model processing module that implements the model logic and calculations – effectively transforming the data inputs and assumptions into model estimates.
- Post-processing and reporting modules responsible for converting model outputs into a final user-required format.




In general, model production systems of the type described above typically expose a company to the following types of model risks:

- Errors in the implementation of the model logic/algorithms into the production system
- Incomplete processing of records
- Materially inaccurate data input values

- Run-time errors or execution failures leading to incomplete or incorrect model results
- Use of incorrect or outdated model parameters and assumptions
- Incorrect or incomplete implementation of approved model changes
- Unauthorized changes to the model
- Use of unapproved or outdated version of the model

Realization of any of these risks can lead to adverse consequences for the company, some of which may be substantial – for example, costly mistakes in critical business decisions, significant underestimation of risk positions, or large financial misstatements, to name a few. While robust model governance and operational controls can usually mitigate these risks, these control activities may come at a significant cost to the company due to the complexity of the model production system. As such, in an environment of tight budgets and scarce resources, we offer a perspective on how these risk areas can be mitigated in a more cost-effective manner through up-front model design choices that reduce the overall risk profile of the model production system. These recommendations are based on our extensive experience validating a wide range of models used in the financial services industry and, in this article, are focused on those models implemented within complex production systems of the type described above.

# Model system design recommendations

Model risk mitigation considerations should be incorporated into the model system design stage since certain model system structures and attributes can help decrease the risk of model execution failures, incorrect model calculations, unauthorized changes to the model logic, and other potential issues. By reducing these risks, they can also help decrease the costs of performing model validation tasks and ongoing operational control activities. In the remaining sections of this article, we focus on the following six specific model system design recommendations:

- Modular system/computational structure
- Centralized updates of model parameters and assumptions
- Centralized automated model execution
- Increased transparency of intermediate calculations
- Detailed execution reporting
- Use of a robust computing platform

## Modular system/computational structure

As mentioned in the introduction, a typical model system incorporates a number of functions, including data extraction and transformation, model calculations, and output post-processing and reporting. Our first recommendation to reduce model risk is to separate these functions into clearly defined distinct system modules. For example, for a model system implemented in the SAS or C++ programming language, this means coding each of these functionalities into separate programs, as opposed to including all functionalities into one long program.

While this may seem like an obvious suggestion for someone formally trained in software development, in practice model production systems are frequently designed by economists and financial engineers outside of an IT general controls environment and without formal software development training.

This modular system architecture offers the following model risk mitigation benefits:

- **Reduced change management risks** – When implementing an approved model/system change (for example, a change to the model pre-processing code), the risk of an inadvertent change to other areas of the model production system (for example, the model calculation logic or the model post-processing module) is greatly reduced when they are separated into independent modules. Only the module(s) requiring modification are accessed, thus shielding the other modules from exposure to accidental changes. This modular structure also makes change management control activities (for example, ensuring lack of unauthorized changes) more efficient for third party reviewers – such as a model validation unit or an internal or external auditor.
- **Reduced system maintenance costs** – When individual model system functions are clearly delineated, implementing model changes is faster, and error detection and correction is less complicated. Because each module can be executed independently, the task of isolating model issues or errors can be performed more efficiently and with a higher degree of confidence. Similarly, approved changes to specific functions only require accessing the specific module related to the function, lessening the effort necessary to implement and test the changes. The ability to isolate the changed module for testing also allows the model users to better evaluate the impact of the changes on the model results.
- **Reduced code documentation costs** – The modular structure, by design, organizes the model into distinct logical sub-processes, providing a strong foundation for the model system documentation layout.

## Centralized updates of model parameters and assumptions

Almost every model system has some parameter values and assumptions that have to be updated periodically. These can include locations and names of the input and output files, date parameters, prepayment/default/loss severity rates and other behavioral assumptions, interest rate forecasts, house price appreciation forecasts, unemployment rates, and other economic forecasts.

Frequently these parameters and assumptions are manually entered into the model system by making edits to the system's programming code. In many cases, the same parameter or assumption values may be used in multiple locations throughout the model code – thereby requiring the model owner/user to update these values in many different locations. This type of programming code design increases the risk that certain model parameter or assumption values will not be updated completely or accurately.

To reduce this model risk, we recommend:

- Structuring the programming code such that the parameter and assumption values only need to be updated in one program location – which is then referenced in all subsequent programming statements relying upon the parameter or assumption value.
- Consolidating all of the periodically adjusted parameters and assumptions into one centralized location in the model system's programming code.

Not only do these recommendations reduce the risks of incomplete or inaccurate model parameter/assumption updates, they also significantly increase transparency of the model production code as a third-party can see in one location all of the model parameter and assumption values – rather than hunting through the entire programming code to find them. Executing approved changes is now simplified for the person making the changes and validation of the changes by a supervisor or third-party is more effective and efficient.

Finally, we also recommend creating detailed documentation for each model parameter and assumption value to further mitigate the risk of errors. This documentation should include the purpose of each parameter and assumption value, how frequently these values should be updated, the required format of these values, and a catalog of program line numbers where each value is used throughout the program. This documentation aids in the training of new personnel and reduces key-person risks. Such documentation can either be a part of the model system operating procedures or programming code, or both.

## Centralized automated model execution

When the model production system consists of several independent modules, there is a risk of inaccurate or incomplete hand off of data between the modules if the system is executed manually one stage at a time. For example, consider a system consisting of several modules that each has to be manually executed by the model user. If the Data Pre-processing Module generates a dataset that is supposed to be consumed by the Core Model Processing Module, there is a risk that this dataset may be unintentionally corrupted, moved, or renamed by the model user after it is produced by the Preprocessing Module, but before the Core Model Processing Module is executed. To mitigate the risk of incomplete or incorrect data hand-offs, we recommend eliminating manual model execution in favor of an automated execution sequence.

## Increased transparency of intermediate calculations

The sequence of calculations within the core model logic of a model system can sometimes be very complex. For example, a discounted cash flow valuation model might simulate the performance of a loan portfolio for every single month of its remaining life, which might involve a number of intermediate steps such as:

- Calculations of contractual cash flows
- Estimation of prepayment probabilities
- Estimation of default probabilities
- Estimation of the loss severity rates
- Estimation of expected cash flows
- Discounting of expected cash flows

While, ultimately, the user may only care about the final model output such as the expected value, it is critically important to design the model production system in such a way that the results of all the intermediate calculations like those mentioned above are accessible.

Exposing the intermediate results facilitates effective and efficient validation of the model and its implementation. Our experience testing a large number of complex production model systems shows that the level of effort required to validate a model that does not capture and expose intermediate calculations is dramatically higher than the one that does.

Concerns about extra IT costs or sacrifice of the model system performance speed by saving intermediate outputs can be addressed by incorporating a switch into the model system that can turn the intermediate output on or off.

## Detailed execution reporting

A number of model operational risks can be monitored by designing the model production system to produce summary reports containing run-time information that can be used to identify various model processing errors and other potentially abnormal conditions. Such reports are frequently referred to by different names such as LOG files, exceptions reports, or control reports. In our view, the design and production of these reports should be included in the design of all model production systems. The types of information produced by these reports should include the following:

| Type of information | Model risks addressed |
|---|---|
| • Number of records, or sum of balances, processed by the model run. For model systems consisting of several distinct modules, additional reports showing the number of records, or sum of balances, entering and exiting each module.<br>• Number of records, or sum of balances, that could not be processed, as well as the reasons why the records could not be processed. | • The model did not process the population of records completely.<br>• The model run did not complete successfully. |
| • Number of duplicate records.<br>• Number of records with missing values.<br>• Number of records with invalid data values. | Data inputs are not materially accurate. |
| Breakdown of the number of records that were affected by the preprocessing filters and data cleaning overrides. | • Data filtering/cleaning rules become obsolete or otherwise invalid.<br>• Data inputs are not materially accurate. |
| Information about any errors or other abnormal conditions encountered during the model run. | • Model run did not complete successfully.<br>• Model logic has become obsolete or otherwise invalid.<br>• Data inputs are not materially accurate. |

| Type of information | Model risks addressed |
|---|---|
| Summary of inputs, assumptions, and parameters used in the run. | Incorrect/unapproved parameters, assumptions, or data inputs are used by the model. |
| Listing of any external files imported during the run. | Incorrect data inputs were used by the model (e.g., using a March loan file when the April file should have been used). |
| • Execution date/time stamps for each module run – including the execution start date/time and execution finish date/time.<br>• Unique label such as a Run ID. | • Model run did not complete successfully.<br>• Model modules were not executed in a correct sequence.<br>• Model users consumed results of the wrong model run (for example, a preliminary run). |
| Unique digital fingerprint of model production code (e.g., an MD5 checksum of the production executable). | • Model version in the run is not the official approved version |

## Use of a robust computing platform

Use of a robust computing platform for a model production system should be a critical element of a model risk mitigation strategy. What do we mean by “robust”? It is quite simply such a platform that would allow the user to implement the design recommendations set out earlier in this article, such as:

- Modular structure
- Centralized mechanism for updating model parameters and assumptions
- Centralized model execution mechanism
- Increased transparency of intermediate calculations
- Detailed execution reporting

This definition of a robust computing platform would include programming systems such as SAS, C++, and Java, but would generally exclude “desktop” applications such as spreadsheet or database tools (“desktop” term refers to applications running on Windows personal computers). While cheaper to acquire and easier to use, desktop spreadsheet and database applications are limited in their ability to accept the types of design recommendations discussed above and, as such, models deployed within these platforms typically expose the institution to heightened model

risks and costlier operational controls, model validation, and maintenance. Key limitations associated with desktop application-based model systems are:

- Lack of a mechanism for producing detailed model execution reports and audit trails
- Lack of a robust error-reporting mechanism
- Reduced transparency of model calculations
- Increased risk of human error (e.g., errors associated with copying and pasting data or copying formulas)
- Difficulty of identifying differences between versions of the same model leading to increased change management risks
- Difficulty documenting model logic within the application. While SAS, C++, and any other programming language allow the user to insert extensive comments within the code to document the logic, the mechanism for adding such comments to formulas within spreadsheet applications is very cumbersome
- Limitations on the size of the problems that can be handled by these applications, and lack of scalability
- Vulnerability to viruses and security breaches

While some desktop spreadsheet and database applications may offer an internal programming language, even those desktop-based model systems that take advantage of this capability are subject to many of the above limitations.

We recognize that use of desktop applications as a platform for model production systems is sometimes unavoidable because of the need for fast model deployment, or because of the shortage of personnel qualified on other computing platforms. In these circumstances we recommend that companies consider developing plans for transitioning such models from desktop applications to more robust platforms when it becomes practical.

# Summary

While most of the recent efforts by financial services companies, their regulators, boards of directors, and other constituents focused on strengthening independent model validation programs, this paper offers ideas on how to combat model risk at an earlier stage of the model life cycle – the design phase. We provide practical recommendations on how to design model production systems in a manner that minimizes model risks and reduces model validation and maintenance costs. Most of the recommendations cost little to nothing to implement, but offer significant potential long term benefits.

www.pwc.com/modelrisk

Contacts:

PricewaterhouseCoopers offers a full range of Advisory services to assist you in identifying and managing the complex risks associated with the development, deployment, and maintenance of complex models used for risk management, valuation, and financial/regulatory reporting purposes. Please contact:

Ric Pace
(703) 918-1385
ric.pace@us.pwc.com

Jason Dulnev
(703) 918-1371
jason.dulnev@us.pwc.com

Andrés Palacio
(703) 918-3660
andres.palacio@us.pwc.com

Renato Torrijos
(703) 625-8549
r.torrijos@us.pwc.com