Algorithmica

ARMS RiskEye

High-performance, enterprise-wide, market and counterparty risk for traders and risk managers

Contents

WHO SHOULD READ THIS PAPER?

If you are responsible for asset and investment risk management, or accountable for any related risk-supervisory or risk-governance role, this paper is for you.

If you work in leveraged finance, asset management, investment management, custodianship or pensions management, this paper is for you.

ABOUT US

Founded in 1994, Algorithmica Research is dedicated to creating innovative software for quantitative financial analysis. Now part of the highly respected FCG group – the Nordic leader in financial risk, analytics and compliance advisory – Algorithmica is honored to serve an extensive blue-chip client list, including banks, hedge funds, asset and fund managers, insurance firms, and exchanges.

Algorithmica successfully combines leading-edge competence in quantitative finance and software development, and partners with its clients to provide financial technology expertise, offering enterprise solutions for:

- **Data management**
- **Risk Management**
- **Quantitative analysis**

The company sponsors academic research and supports courses in quantitative finance, and invites clients and partners to join the mission to improve all aspects of financial management.

Executive Summary

Enterprise Risk Management in Real Time

Today's risk managers deal with unprecedented complexity and dynamism. Managing counterparty, credit and market risk relies on capturing, maintaining, and analyzing vast amounts of historic data, while simultaneously gaining real-time insight based on live feeds.

Many institutions rely on legacy risk management systems with limited analysis capabilities, few configuration options, and no real-time analysis. As a result, those institutions will gradually fall further behind while markets – and competitors – race into the future. For example, in today's world, all new trading should be instantly integrated into current risk analysis in order to monitor limit breaches and potential risk/reward ratios. Similarly, for investment managers running risk-budgeted mandates, the ability to run what-if scenarios on top of the current portfolio is integral. These capabilities are often near-impossible to achieve with legacy systems. In addition, investors and government authorities require asset managers and financial institutions to manage market and credit risks with methods such as multi-factor grid stress-testing and value-at-risk. Extending legacy systems to meet new regulatory demands can be costly, demanding high capital investment.

Algorithmica Risk Management System (ARMS) RiskEye is a groundbreaking new solution, able to perform risk analysis in real-time using in-memory technology – previously only possible through end-of-day batch calculations – and equips risk managers with powerful tools to handle market and counterparty risk. Using the latest analytics technologies, together with a comprehensive collection of instruments and validated historic data, RiskEye supports a wide range of methodologies, enabling risk to be viewed and analyzed from multiple perspectives – such as by instrument groups, by risk class, by regions, diversification, portfolio risk, and more. RiskEye enables real-time risk analysis of new trading, and provides what-if capabilities across all instruments and portfolios.

To enable rapid deployment, RiskEye offers turn-key interfaces for popular front-office systems, with connections to live market data feeds such as Bloomberg, Reuters and more. Its flexible system architecture supports rapid configuration for regulatory changes, and the use of multiple risk management methodologies to enable legal compliance.

RiskEye offers a modern, highly flexible risk analytics solution that enables managers and institutions to gain competitive advantage, enhance decision quality, and improve risk management.

What is ARMS RiskEye?

Algorithmica Risk Management System (ARMS) RiskEye is a fast, robust, and scalable risk calculation and aggregation framework. It has an in-memory datastore for storing complex pre-calculated data that enables real-time aggregation from intra-day transactions and updated market data.

What sets RiskEye apart is its open architecture, with an easy-to-use compiled language, Qlang, that can easily be extended with in-house models or external calculation libraries. As part of its modern design, RiskEye provides multi-threaded calculation, aggregation and API-access, offering maximum performance on inexpensive hardware. As RiskEye provides all three processing layers - integration, calculation, and aggregation – within the single solution, it offers considerable infrastructure and management cost savings while delivering outstanding risk management capabilities.

What does ARMS RiskEye do?

RiskEye performs five key functions:

1. RiskEye loads financial transaction data, instrument meta data, and risk factor timeseries data using a variety of loaders and integrated connections. It can consolidate and validate transaction data coming directly from front office systems, and loads both transaction and instrument definitions into an in-memory datastore. All transactions and instrument data can be accessed and updated using APIs.
2. RiskEye provides a flexible hierarchical calculation tree-graph that can hold simple or complex relations between nodes. As transactions and instrument data resides on the leaf-level, a chain of calculations can be triggered throughout the entire tree as soon as any of the information on the atom level changes. This ensures that updates to transaction volumes or market data will propagate instantly and update all measures and aggregations that are dependent on that information.
3. RiskEye computes relevant risk measures out-of-the box. Whether the platform is used for market valuations, value-at-risk, profit and loss, or XVAs, RiskEye will compute lowest-level measures on the transaction and risk factor level, and store the result vectors in-memory. Custom measure can easily be created and integrated on the same terms as the built-in functions, both if written in Qlang™ and if taken from in-house or external libraries. All built-in valuation functions can also be viewed in plain text and modified by the user. As all complex and non-linear calculations are performed at the atom-level, aggregation of risk measures is linear – and extremely fast.
4. RiskEye can pre-aggregate or dynamically aggregate data depending on preference. For frequently used measures such as Present Value or PnL, all nodes in the hierarchical tree can have pre-aggregated values stored in the rCube™ (see panel) ready to be queried by the user. For ad-hoc queries over multiple aggregation criteria, dynamically triggered aggregation will be performed for the specific user request. Both pre-aggregated and dynamically triggered aggregation will be updated incrementally when the underlying positions, instruments, or classifications change. This ensures only re-aggregation of small parts of the entire tree structure and updating of the datastore.
5. RiskEye performs what-if analysis in many dimensions. For example, users may enter a set of potential trades and see the effect on risk, limits, and capital charges for those trades; or users may want to see the effects that an adverse risk factor or curve will have on the portfolio metrics. Naturally, multiple what-if changes can be performed simultaneously, and viewed side-by-side with the original data. What-if results can be shared with other users, and accessed through the available APIs.

Informed Business Decisions

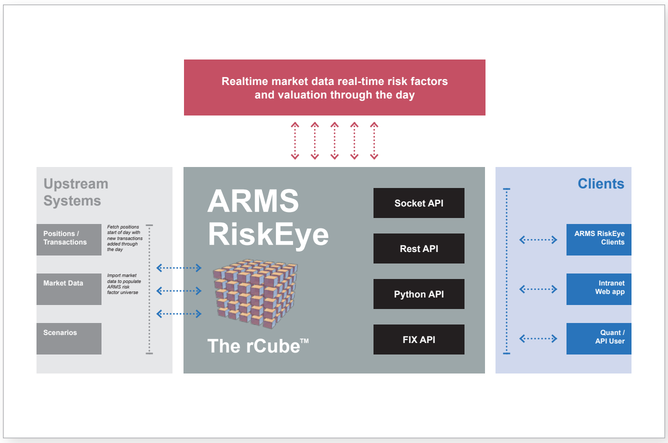
To enable timely and informed business decisions, having near- to real-time transaction data and risk calculations is crucial. Legacy solutions that build on down-stream aggregation of pre-calculated data struggle to handle many of types of what-if calculations. For example, performing a simple +100 basis point parallel shift of a yield curve would, to be handled correctly, trigger a complete re-calculation in all systems having transactions dependent on such a yield curve.

RiskEye achieves complex what-if scenario calculations by using a combination of pre-calculated and aggregated data, with incremental updates in the hierarchical tree and the inclusion of the complex calculations themselves, all in one platform.

The RiskEye architecture is inherently flexible, open to customer extensions, and able to meet business and regulatory changes. In-house users can benefit from having official risk measures pre-calculated, as well as permitting user overrides to obtain alternate risk measures.

RiskEye flexibly allows customers to:

- Load many alternate hierarchical portfolio trees
- Run incremental loading and calculation of positions and measures when available from the business systems
- Internally define measures to be pre-calculated and pre-aggregated for maximum user performance
- Add position and risk-factor classifications to use for aggregation across the tree
- Allow tiered access control of positions and calculated measures
- Ensure coherence and integrity of the calculated results, ensuring that all users, whether using the RiskEye solution or any of the APIs, will see the same results
- Keep an intra-day updated set of transactions and valuations ready for inspection if the system is connected to real-time sources such as Bloomberg or Refinitiv and relevant front-office systems.


Realtime market data real-time risk factors and valuation through the day
Upstream Systems
Positions / Transactions
Fetch positions start of day with new transactions added through the day
Market Data
Import market data to populate ARMS risk factor universe
Scenarios
ARMS RiskEye
The rCube™
Socket API
Rest API
Python API
FIX API
Clients
ARMS RiskEye Clients
Intranet Web app
Quant / API User

Loading Transactions & Risk Factors

RiskEye loads transactions and risk factors either from a standard relational database, using xml-based files, or directly using a streaming API.

At start-up, a server can be directed to load a large set of quality assured transactions directly into memory. As soon as the start-up sequence is finished, configured measures are calculated and made available to the user interfaces.

After loading is complete, the raw xml-formatted transaction / position as well as the risk-factor mapped transaction will be available to users in both the UI, and in the Python and REST APIs. This process enables updated what-if scenarios to be included as loaded transaction and instrument definitions. Late additions or changes can be loaded to the running server, and these will trigger recalculations for the parts of the tree that have become “dirty”.

Through-out the trading day, new transactions can be loaded seamlessly into the rCube memory (see panel) using the APIs. Risk factor series, curves and surfaces can also be updated during the day to provide a current view on market values and risk measures.

RiskEye includes adapters for well-known front office systems, further simplifying integration between trading platforms and independent risk measurement.

Numbers, Measures, and Measure Settings

Using a combination of transaction data, instrument definitions, and mapped market data many “numbers” can be calculated, referred to as “measures.”

Examples of measures:

- Present Value of a position in a predefined base currency, for a given set of market data
- Scenario Value of a position using a predefined market value scenario, either historical or user-defined
- Value-at-Risk at a given confidence interval using a pre-set number of historical risk sample days, weighting scheme, horizon days etc.
- Profit and Loss historical samples (as an intermediary measure format for Value-at-Risk)
- Sensitivity measures by group of risk factors such as one basis point curve shifts

Users can easily extend the set of available measures and/or change settings to configure and adapt calculations to meet specific regulatory and in-house demands.

For example, Value-at-Risk can be pre-configured to calculate an institution’s official risk number, and the measure settings can be user-defined in the UI and the Python and REST APIs. By simply asking for another confidence interval or a different trading horizon, the server will calculate and present that to the user with a minimum of recalculations.

In practice, all profit and loss samples used in a value-at-risk calculation per each position and risk factor group are stored in the rCube memory (see panel). This ensures that measures that can be derived from a given set of parameters such as number of samples, confidence interval, expected shortfall, time weighting and more, and can be specified by the user without requiring recalculation. All the samples, both at the position and portfolio level, can be viewed and exported using the APIs for further analysis.

What is the rCube?

RiskEye includes an in-memory datastore called the rCube, a combination of a hierarchical tree graph and an OLAP cube. The result is that a financial organization is typically represented as a business-line tree structure, with risk taking (positions/transactions) at the bottom and the grand total business at the top. Between the tree’s leaves and the top, rCube includes numerous business lines, geographical partitions, treasury business, trading books, and more.

In order to cater for many different stake holders, any number of hierarchies – including all or subsets of the positions – can be represented. By combining built-in classifications (for example currency, instrument type, portfolio, trader, counterparty) with user-defined classifications (for example reporting codes, product codes), a multi-dimensional cube with a sub-set of positions can be represented at each node in the hierarchical tree.

The rCube is then the representation of the results for each desired sub-set of measure results, at all levels of the tree. For example, the rCube on the global swap trading book level node will include all historical PV-samples for each risk factor group and classification grouping in a matrix structure, as well as the aggregated market value grouped by all classifications defined in the system.

The rCube structure enables users (in the UI or using APIs) to answer questions such as, “What is the Value-at-risk @97,5% confidence level of the global swap trading book, coming from only from interest rate swaps denominated in EUR, counting only the effect from ESTR discounting curve risk factors?”

The rCube dimensions in historical space, as well as the classifications, are only limited by the memory available in the RiskEye server. For example, a tree with a thousand tree nodes, 2.5 million unique positions with storage of 250 risk days, and with value-at-risk per risk factor groups and several classification groups, can be run using less than 300 GB internal memory.

Aggregation of Results

RiskEye stores the measure results on an atomic level that enables simple summation for most aggregated measure results. This includes calculation of value-at-risk on all levels of the hierarchical tree. For parametric value-at-risk, deltas and gammas are stored in the rCube and aggregated using the method of moments; more-complex aggregation methods can also be defined between the nodes in the tree if necessary.

In the hierarchical tree graph, RiskEye stores both the transformation formula between the nodes (the aggregation method of data) and stores the dependency structure. This process ensures that if any part of the tree becomes dirty either by user interaction override or by updates of intra-day transactions, only the relevant parts of the aggregation need to be recalculated.

An optimal path with minimum number of recalculations is stored in the server. For example, if a user changes the maturity of a single fx-forward, the following steps are performed:

- Trigger the create function of a new in-memory binary position from the raw transaction representation (that also resides in-memory)
- Trigger all measures calculations associated with a binary position, such as its present value, all historical PnL samples, sensitivities etc.
- Lowest-level rCube members that are changed now trigger their dependent nodes using the relevant aggregation functions, until affected nodes are re-calculated all the way to the top node
- User interfaces connected to the rCube will listen to any nodes that have changed and instantly be updated with the new data, and highlighted that to the user(s) in color-blinks.

If only the notional value of the fx-forward changes, then the system triggers only the calculation of measures, which does not require the create function to be activated. Similarly, any changes to higher level node-results do not trigger “downward” looking numbers. This data structure is sometimes referred to as a Directed Acyclic (tree) Graph (DAG).

What-if Analysis

One of the main features of the RiskEye platform is its ability to perform fast what-if simulations that propagate all the way to the top level of the hierarchical tree structure. Using either the RiskEye client or Python and REST APIs, users can access and change any part of the transaction or instrument definitions of existing positions. Further, users can build an overlay of one or many new positions as if they were recently traded.

In combination with modifying transactions, the risk factor and market data can be what-if simulated.

Users could create what-if scenarios to answer questions such as: “What happens to the risk and limits if I trade a new interest rate swap with counterparty A and the discount curve in EUR jumps 54 basis points?”

For each connected user, a separate rCube memory space is allocated when running what-if simulations that can be paired with the original information to enable displaying “before and after” results in the user interface or using the APIs.

Client User Interface

Even though the RiskEye server is intended to be used entirely as an in-house or cloud-based calculation server, accessed using official APIs only, the solution includes a light-weight user interface. The RiskEye client User Interface (UI) uses the raw socket-based API to present data to the end-user. This approach ensures that even if portfolios contain hundreds of thousands of positions and multiple measures, only the part being viewed on-screen is passed over the network connection and displayed on the desktop, resulting in very fast response times. Thousands of users can run RiskEye clients, connected to the same server(s), without causing network congestion and without slowing the server down.

Additionally, the RiskEye client runs a separate thread that listens to updates on the server. When nodes or cells in the server become dirty, this immediately triggers a “push” update to the RiskEye client, and there is no need for the user to refresh the display as new data is shown as color blinks. Again, this design reduces network traffic and ensures optimal server performance even on modest hardware.

Connections are created as RiskEye client logins or using windows authentication, enabling only permitted server data to reach user displays. When using the RiskEye client for what-if simulations, a unique ID is automatically created and a secured simulation space on the server is allocated for the what-if simulation results. Results can be private or public, and shared with other users.

The principal RiskEye client view is a vertical and horizontal pivot table. Measures with user-defined settings can be attached and saved as workspaces. User-defined columns can be created by combining columns and performing simple arithmetic.

For what-if purposes, a set of predefined position templates are available directly in the UI by simply right-clicking on a portfolio and adding a simulated position into the pivot tree.

QUERYING THE rCUBE

Data placed in the rCube at any level, regardless of datatype, can be queried by users. Examples include:

- List of the portfolios belonging to a tree node
- Raw position and instrument definition data
- List of external position identifications for a given portfolio
- Market value for a given position / portfolio / node for a given date
- Profit and loss sample vector for a given position / portfolio / node
- Aggregated values anywhere in the tree
- Risk factor values for a given date
- In addition, interaction with data, such as updating positions and performing what-if simulations, can be completed using the same query tools.

For rCube queries, the RiskEye client user interface (see below) relies on socket-based communication, which can also be accessed via Qlang and C#. A set of Python functions, as well as REST API functions, can be used to perform most common queries to the rCube.

Conclusion

A Modern, Highly Flexible Solution

In essence, ARMS RiskEye solves the central challenges facing risk managers: real-time analysis of risk by any chosen dimension. The solution architecture delivers high performance even on modest hardware platforms, with powerful data management provided by the rCube technology. The light-weight user interface and comprehensive Qlang and API capabilities offer easy integration and extension, able to handle new regulations and business requirements.

In brief, RiskEye offers a modern, highly flexible solution that enables managers and institutions to gain competitive advantage, enhance decision quality, and improve risk management.