Paper 179-29

A Day in the Life of an Analytical Warehouse

Julian Anderson, American Electric Power
Greg Barnes Nelson and Jeff Wright,
ThotWave Technologies, Cary, North Carolina

Introduction

A decision support system that doesn't have the trust of the decision makers is worthless. To earn trust, quality must be built into the design and construction of the system, but that isn't enough: It's just as necessary to assure quality as new data is loaded into the system. Data warehouses normally involve integrating data from multiple source systems, so the classic maxim "garbage in, garbage out" applies. The quality challenge is compounded if the processing also includes sophisticated (i.e., time consuming) analytic modeling. How do we insure that errors from bad input data are not published to users? Can we recover from bad input data without re-running the analytic models? How do we allow new business activities to be modeled without having them affect production reports?

This paper describes experiences from an enterprise implementation of SAS® software (using SAS Risk Dimensions). While the applications for risk management are appropriate, the audience should gain practical knowledge that is appropriate for any large scale implementation of an analytic engine. In this case study, analytic results (output from Risk Dimensions) were fed back to a data warehouse to provide a historical record of market exposures and sensitivities. We first describe how the SAS analytics are scheduled and monitored to make sure that the technology has done what it was built to do. Then we describe the features that were implemented so that business users can make experimental runs, verify results, and approve them for production reports.

Background

American Electric Power (AEP), with the help of ThotWave Technologies, LLC., recently developed an enterprise-wide risk management system to measure risks associated with energy contracts. These measures help project the possible loss in value of a portfolio due to movements in financial markets. The system, developed at AEP's Columbus, Ohio headquarters, provides AEP with improved analytical capabilities for their North American Energy Trading and Global Treasury risk management groups.

The Enterprise Risk Measurement Application (ERMA) project's goal was to replace a legacy risk engine and reporting system with a robust, automated, and secure application – to make assessments of their overall risk. The application enables users to access reports via a web browser to gain better, faster access to underlying data for more detailed analysis. ERMA includes various real time and near real time components developed by ThotWave.

To solve this enterprise-wide need of aggregating various risk measures, we needed to combine data from various operational and planning systems throughout the organization and bring them all together each day to feed into Risk Dimensions—the analytical workhorse that performs the calculations. Then, information could be surfaced to users in a secure way. One of the most important features of this system was derived from the need to officially "publish" the numbers for the day after someone had reviewed them.

Market Risk Management

Energy companies are required to manage a large portfolio of contracts for energy products such as natural gas, coal, and power. A typical portfolio will include derivative contracts such as options and swaps as well as simple futures. The purpose of these deals may be to guarantee a supply of fuel, to provide products to customers, to hedge against market uncertainties, or just to take advantage of a perceived market opportunity.

These contracts can be valued against daily market conditions. For example, an option to buy a certain number of MMBTUs of natural gas in February 2005 can be valued based on today's price for natural gas in February 2004, the volatility of the February 2005 prices based on recent market history, and the interest rate.

The goal of market risk management is to provide financial oversight for potential losses due to market changes to insure that operations are within the company's risk tolerance. In the energy industry, the Committee of Chief Risk Officers (CCRO) has published recommendations for how to mathematically measure and report market risk [CCRO]. One commonly used measure is Value-at-Risk (VaR). VaR is the maximum expected loss over a defined period of time with a defined confidence interval. For example, a company may want to measure the one-day VaR with 95% confidence, and to report that daily for different breakdowns of the portfolio: By organization/book, by geographic regions, by commodities, by forward time periods, etc.

Information technology plays a key role in enabling risk management. SAS Risk Dimensions delivers a full range of modern credit, market and operational risk analysis techniques including:

- Mark-to-Market
- Scenario Analysis
- Profit/Loss Curves & Surfaces
- Sensitivity Analysis
- Delta Normal VaR
- Historical Simulation VaR
- Monte Carlo VaR
- Current Exposure
- Potential Exposure
- Credit VaR
- Back Testing
- Optimization

Risk System Architecture

ERMA has to run flawlessly every day as the highest level executives and directors in AEP's trading organization rely on this data to make fundamental business decisions. The ERMA application is comprised of four major functional elements – each responsible for a specific task or series of tasks within the application framework. These are:

- Data Warehousing
- Risk Modeling and Analytics
- Reporting and End User Access
- Infrastructure and Support Systems

Each of these areas is tied together loosely through a logical infrastructure and are housed on physically separate machines and distributed appropriately throughout the organization.

- **Data Warehousing:** The data warehouse architecture plays a pivotal role in the deployment of this application as it provides the driver for many of the exploitation tools that sit on top of the data store. The data warehousing components both provide input to the analytics and manage results. The information architecture is described in more detail in the following section.
- **Risk Modeling and Analytics:** The risk modeling and analytics tool is based on SAS Institute's Risk Dimensions product. This tool is fed by the data warehouse as well as updates to the data warehouse with model output. The analytics were implemented in a highly data-driven manner, driven by reference data such as portfolio structure, risk factors and utilizing run parameters from SAS data sets.
- **Reporting and End User Access:** The end user access tool has two components. The secure Dashboard is geared towards widespread access throughout the enterprise. This web-based tool utilizes Java technology and HTML/DHTML to enable end users the ability to view reports containing interactive, drillable charts and tables. Access is password protected and reports have built-in data security constraints so that users only see data that matches their level of responsibility. The second component is for a more specialized audience that needs full access to the modeling and analytics toolset described above. These users access the data mart for analytics through the SAS RD client, Enterprise Guide or other desktop tools.
- **Infrastructure and Support Systems:** The ability to manage an enterprise-class application relies on good administrative tools. In these kinds of applications there is always the need to manage metadata (such as lookup tables), security (users, groups and roles) as well as the ability to monitor events in the system. These events are usually technical in nature like source system problems or notification of jobs being completed successfully.

The entire application employs an n-tier framework to serve up the risk management reports (i.e., Positions, VaR, and EaR). The thin-client (browser) accesses a web server to access the application front-end. From there, all of the application logic and database access is executed on a second (logical) server (SAS application server). A simple diagram of the application infrastructure is defined below.

Information Architecture

The Information Architecture has several key components that make up its framework. The diagram below outlines ERMA as an integrated system, complete with management (administrative and operations components) as well as information services (metadata) that help tie the system together.

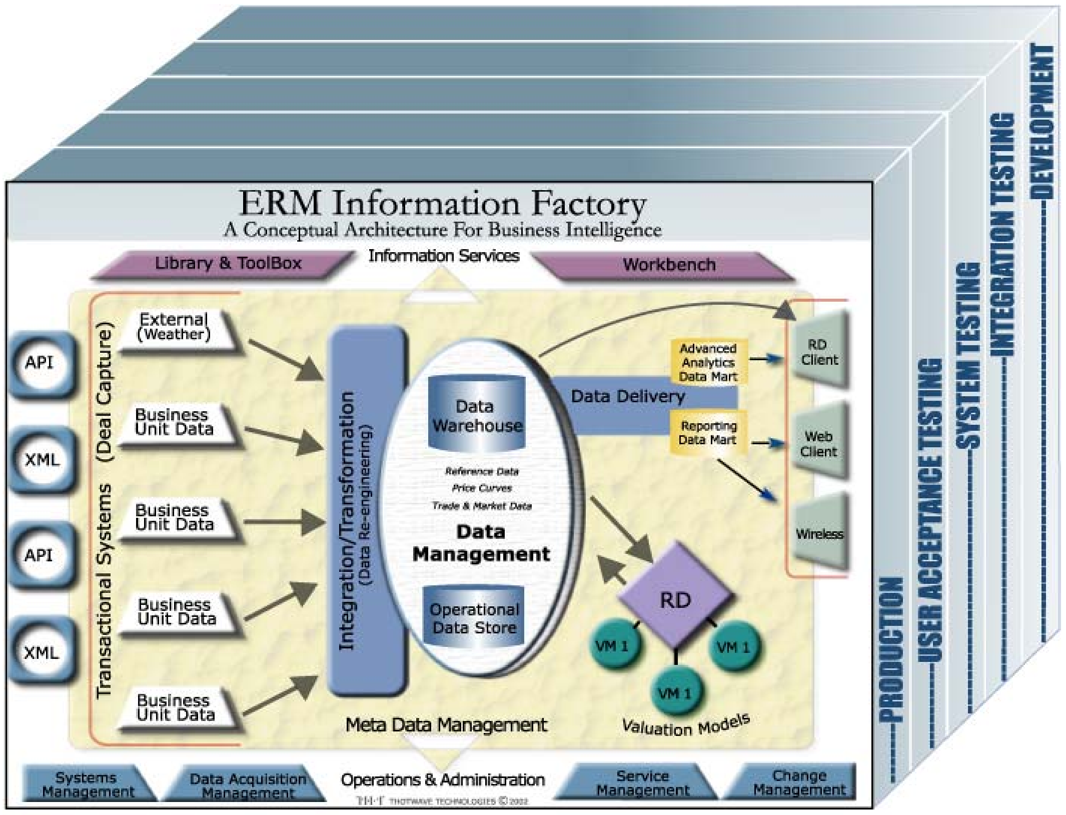


Figure 1 – Conceptual Design Diagram

Each of these systems or services supports the business systems to ensure data quality and repeatable processes which include:

- **Transactional Systems** are the internal and external core systems that run the day-to-day business operations. They are the source of data for the data warehouse. Most often, these represent deal capture systems used by the business units (e.g., ZAI*NET, OpenLink).
- **Integration/ Transformation** is the set of processes that captures, integrates, transforms, cleanses, reengineers and loads source data into the data warehouse and operational data store. Data reengineering is the process of investigating, standardizing, and providing clean consolidated data. (This is the handshake between the business unit integration and the operational data store).

- **Data Management** is the set of processes that manage data within and across the Data Warehouse and Operational Data Store. It includes processes for backup and recovery, partitioning, summarization, aggregation, and archival and retrieval data from near-line or off-line storage.
- **An Operational Data Store** is a subject-oriented, integrated, current, volatile collection of data used to support the tactical decision-making process for the enterprise and is the central point of data integration for business management. The operational data store delivers a common view of enterprise data. In a Risk Dimensions application, this data is primarily used to serve the immediate loading of the risk engine.
- **The Data Warehouse** (proper) is a subject-oriented, integrated, time-variant, non-volatile collection of data used to support the strategic decision-making process for the enterprise and is the central point of data integration for business intelligence. The data warehouse is the source of data for the Data Marts and delivers a common view of enterprise data.
- **A Data Mart** is customized and/or summarized data that is derived from the Data Warehouse (and sometimes the Operational Data Store) and tailored to support the specific analytical requirements of a given business unit or business function. It utilizes a common enterprise view of strategic data and provides business units more flexibility, control and responsibility. The Data Mart may or may not be on the same server or location as the Data Warehouse.
- **Data Delivery** enables end users and their supporting IT staff to build and manage views of the data warehouse within their data marts. It involves a three-step process of filtering, formatting and delivering data from the Data Warehouse to the Data Marts. Data Delivery is provided through two separate, but complementary tools: Reporting and advanced analytics (exploration and what-if processing).
- **Metadata Management** is the process for managing the information needed to promote data legibility, use and administration. Contents are described in terms of data about data, activity and knowledge. Usually, three types of metadata are of use to consumers: Technical, business and process metadata.
- **Information Services** are the facilities that optimize use of the entire information architecture by organizing its capabilities and knowledge and assimilating them into the business process.
- **Operations and Administration** is the set of activities required to ensure three objectives: Smooth daily operations, resource optimization, and management of growth. Operations and Administration can also be referred to as Production Support. Activities in this area support the management and administration of the application in terms of its

data (reference data maintenance), data movement (error and event notification), servers and services (Server Management), security sub-system and technical support metadata.

Technical Assurance

The ERMA system executes a nightly job to pull data from source systems, perform risk analytics, and prepare results for reporting. This process involves a number of technical "moving parts" that must all be working correctly to get to the reports that management expects every morning. This section explains how job flow control and the event system work together to assure that the moving parts have done what they were built to do every night.

Job Flow Control

Ant is, simply put, a Java-based build automation tool [ANT]. Unlike other “inherently evil” build tools, as the Apache Ant Project puts it, (such as *make*, *gnumake*, *nmake*, *jam, etc*), users provide instructions to Ant using an XML build file that contain *targets* and *tasks*. Instead of shell-based commands, each task corresponds to a Java object, either a built-in Ant task or a custom extension. The Ant approach is not only easier to use, but it employs the same principles as the Java language itself, mainly platform independence and extensibility.

Unix administrators have used *make* for years to automate many tasks above and beyond software compiles. In that spirit, ThotWave elected to use the Ant tool to control the job flow for the nightly run. An XML configuration file is used to define the targets, the dependencies, and the order in which they should run. In AEP’s case, targets are defined as UNIX system calls to SAS programs; each target executes a SAS program as its own process. Targets can be configured to run in parallel or sequentially. The type of run depends on the jobs that are grouped together. For instance, suppose you have the following configuration:

```
<target name='energy_setup' description='Runs extract for energy initalization'>
   <antcall target='000_extract_etvar_ref_tables'/>
   <parallel>
      <antcall target='000_extract_zainet_ref_tables'/>
      <antcall target='000_extract_openlink_ref_tables'/>
   </parallel>
   <antcall target='initialization'/>
 </target>
```

In this example, the target name is “energy_setup”. This target doesn’t directly execute a task. Instead, it calls other targets, in the following order:

1) '000_extract_etvar_ref_tables'

2) '000_extract_zainet_ref_tables' AND '000_extract_openlink_ref_tables', which are executed simultaneously (because they are enclosed within the <parallel> </parallel> tags), and finally,

3) 'initialization'

In order to execute parallel tasks, the <parallel> tag has to be explicitly stated. By contrast, sequential execution of targets is implicit.

Here's an example of a target that actually performs a task:

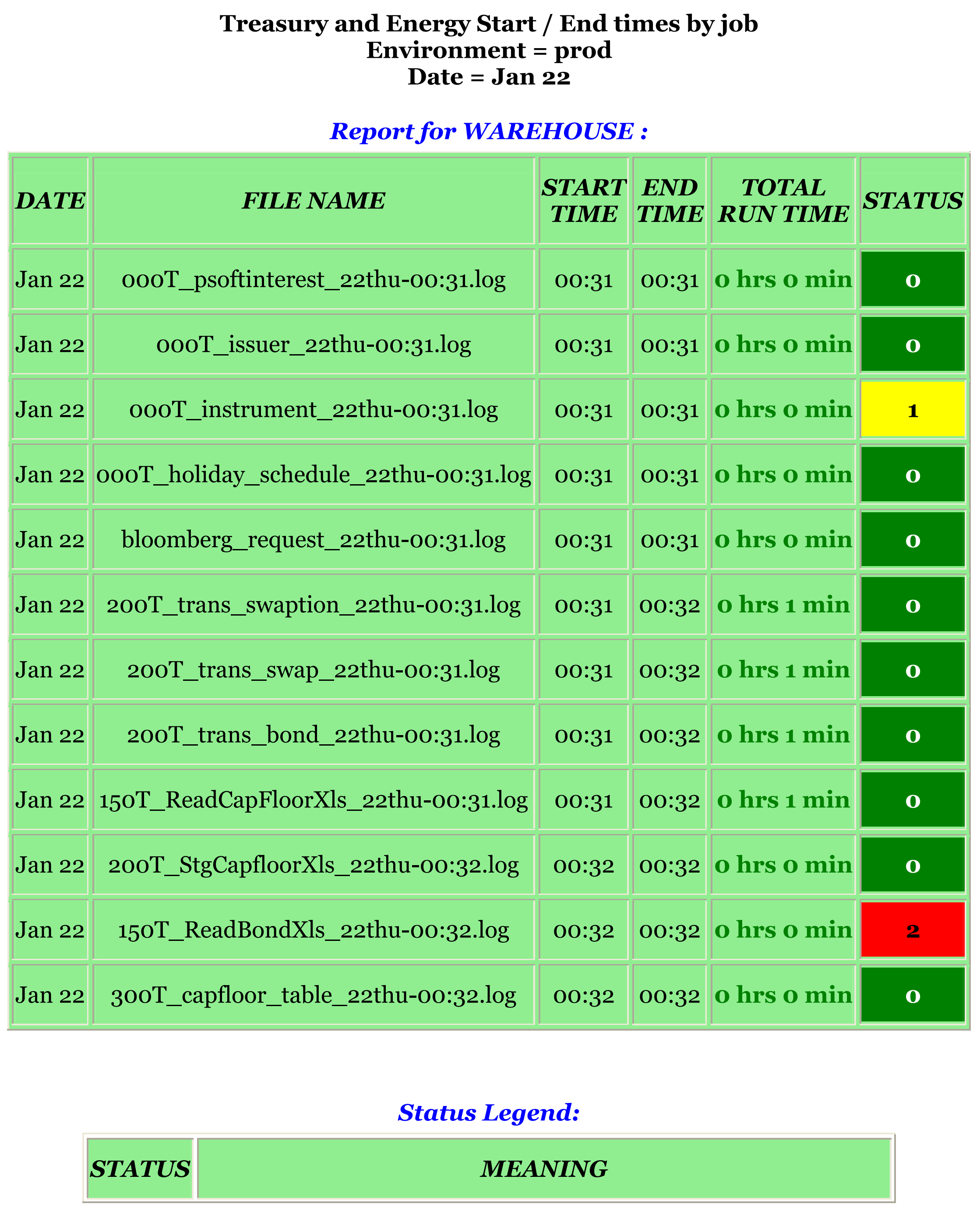
```
<target name='000_extract_zainet_ref_tables'

        depends='init'

        description='get Zainet reference information' >

       <exec dir='${basedir}'

             executable='${script.file}'

             failonerror='${dofail}'>

               <arg line='${MODE} ${SASENV}
               warehouse/programs/000_extract_zainet_ref_tables ${BATCH}'/>

       </exec>

 </target>
```

The `<exec>` tag defines the base directory of the program, what program to run, and what to do in case it fails (stop the job or continue). In this case, the `${dofail}` attribute is a flag set previously to 'true', to stop the batch run from running any tasks after this one.

Using other Ant attributes, we can also set a target to run only a subset of tasks, as opposed to the entire process. For instance, if we have a set of targets that extract data from different sources (target group 1), another set of targets that processes the data and puts them into a format understood by Risk Dimensions (target group 2), and another set of targets that actually run the Risk Dimension engine (target group 3), we can create a "sub" target that only runs one of the groups. This option is also very helpful when we want to run only parts of the process, either for testing purposes, for reruns, or adjustments (discussed later in more detail).

Each program defined in the XML configuration file is instrumented with one or more asserts, as defined by developers, to catch different errors or report on different states (e.g., data integrity, timestamp, number of observations). An assert is a way of checking whether a given condition is true or not; it logs the event using the Event System (described in more detail below). If a condition tested by an assert is not as expected, the program is terminated and returns an error status to Ant. Ant will then abort the job flow.

A *Run Summary Report* is created to analyze the time it took each job in the overnight batch to run and present the exit status of each step. A Korn shell script parses the master log of the nightly process and individual logs for each job to create this summary report. The script can create the report in HTML format or as a comma-separated file for viewing in Excel. System Administrators, developers, and other contributors can examine the report and, if they see that a job took longer to complete than usual, analyze the process in detail and come up with a solution to decrease execution time. Below is a sample run summary report:



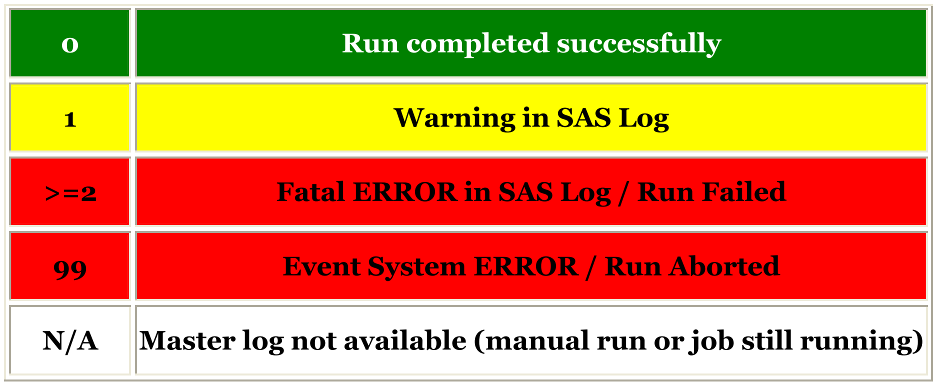
Treasury and Energy Start / End times by job
Environment = prod
Date = Jan 22

Report for WAREHOUSE :

DATE	*FILE NAME*	*START TIME*	*END TIME*	*TOTAL RUN TIME*	*STATUS*
Jan 22	000T_psoftinterest_22thu-00:31.log	00:31	00:31	**0 hrs 0 min**	**0**
Jan 22	000T_issuer_22thu-00:31.log	00:31	00:31	**0 hrs 0 min**	**0**
Jan 22	000T_instrument_22thu-00:31.log	00:31	00:31	**0 hrs 0 min**	**1**
Jan 22	000T_holiday_schedule_22thu-00:31.log	00:31	00:31	**0 hrs 0 min**	**0**
Jan 22	bloomberg_request_22thu-00:31.log	00:31	00:31	**0 hrs 0 min**	**0**
Jan 22	200T_trans_swaption_22thu-00:31.log	00:31	00:32	**0 hrs 1 min**	**0**
Jan 22	200T_trans_swap_22thu-00:31.log	00:31	00:32	**0 hrs 1 min**	**0**
Jan 22	200T_trans_bond_22thu-00:31.log	00:31	00:32	**0 hrs 1 min**	**0**
Jan 22	150T_ReadCapFloorXls_22thu-00:31.log	00:31	00:32	**0 hrs 1 min**	**0**
Jan 22	200T_StgCapfloorXls_22thu-00:32.log	00:32	00:32	**0 hrs 0 min**	**0**
Jan 22	150T_ReadBondXls_22thu-00:32.log	00:32	00:32	**0 hrs 0 min**	**2**
Jan 22	300T_capfloor_table_22thu-00:32.log	00:32	00:32	**0 hrs 0 min**	**0**

Status Legend:

STATUS	*MEANING*



0	**Run completed successfully**
1	**Warning in SAS Log**
>=2	**Fatal ERROR in SAS Log / Run Failed**
99	**Event System ERROR / Run Aborted**
N/A	**Master log not available (manual run or job still running)**

Figure 2 – Run Summary Report

Event System

The other component to insuring that the ERMA technical infrastructure is performing as expected is the Event System. The Event System is responsible for collecting events from system operations and communicating them to affected parties. Here are some examples of events that require attention in the ERMA system:

- The nightly ERMA analytics rely on scheduled data extracts from source systems. If any of these extracts fail, an administrator should be notified so that corrective action can be taken.
- As will be described below, the typical morning routine is for an expert business user to review the nightly analytic results before approving them for view by the enterprise. The user responsible for approving data needs a notification that the analytics have completed and that the pending data has been loaded.
- The reporting data mart uses "slowly changing dimension" logic to enable historical reporting. The number of changed dimension rows is captured as an event, because a spike in the number of changes may indicate a data quality problem in one of the source data systems.

The Event System is a Java sub-system built by ThotWave, design to capture events from the SAS data management and analytic processes as well as the Java reporting dashboard. SAS code tests for events using an *assertion-based* API. In other words, the SAS code contains calls that assert the condition that is expected, and if that condition does not hold true an event is recorded. The event information includes a free form event description and contextual information such as the process step that failed and SAS program/macro name where the event occurred. In addition to events that are explicitly checked with assertions, all SAS errors and warnings are also captured as events.

The Event System handles storage and notification tasks for events. Events are recorded in an event database, which for ERMA consists of SAS data sets accessed via SAS/Share. New events are compared to user subscriptions. Users are able to create event subscriptions based on the type, level (severity), and category (process step) of event. A notification email is delivered to every user with a matching subscription. ERMA system administrators have subscribed their cell phone email addresses to errors that cause the overnight run to fail in order to provide 24x7 support.

The Event System is described more fully in a separate paper [EVENT].

Business Assurance

The preceding section described how the technical workings of the ERMA system are scheduled and monitored. However, before the nightly results are made available to users during the day, a business assurance process must execute to assure that the analytic results are credible. In our terminology, the *publish and approve process* refers to the procedures that control which risk system results are made available to the enterprise in the production reports on the Dashboard. These procedures verify that the source data, data handling processes, and analytic models all worked together in a way to produce measures that are consistent and credible. This section describes the publish and approve process and the tools that support it.

Publish and Approve Process

The basic requirements for the publish and approve process are easy to understand:

- The results from a nightly analytic run should initially be saved in a Pending state not visible to production reports.
- A member of the Market Risk Oversight Operations group will use a variety of tools to review the Pending results. If they are judged accurate, the approve operation is used to make the results visible to production reports (the run is moved to the Approved state).
- Alternately, a manual reject operation deletes the Pending run.

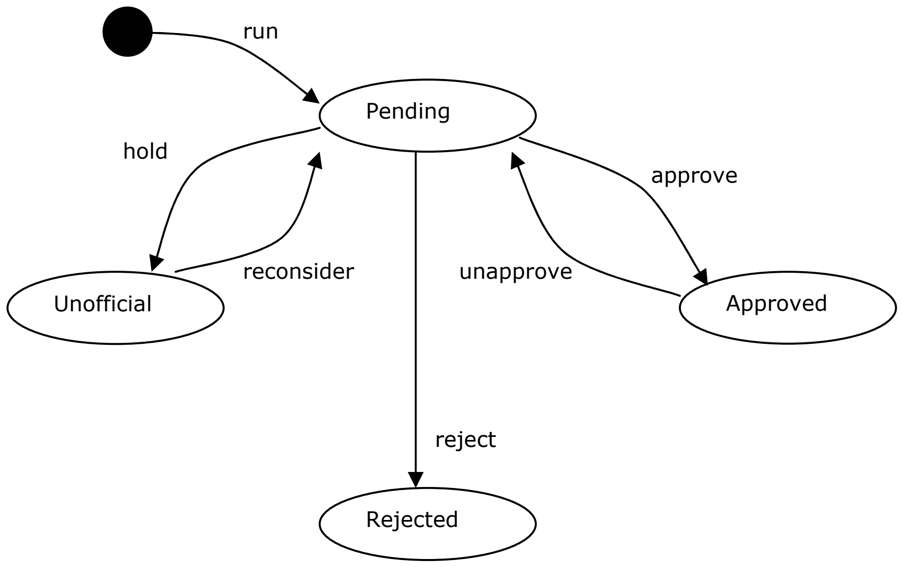
The features just listed might be relevant for nearly any data warehouse. However, the analytics-intensive nature of an enterprise risk management system demand sophisticated tools for managing risk analytic results. There are a number of opportunities for processing problems that can cause incorrect results. Many discrepancies result from recording contracts or market prices incorrectly in the source systems. New types of business ventures and changes in how similar deals are bucketed together present special challenges. These cases sometimes require several experimental runs in order to get results to a point where they are deemed appropriate for publishing to the organization. Because market risk measures are very visible to executive management, and are becoming increasingly part of financial disclosures even outside the corporation, it is very important that the results be stated precisely.

When issues are detected in the review of new pending results, most problems require significant repeats of the analytic calculations after adjusting for the data problems. Since many risk measures are non-additive, it is not possible to add an after-the-fact adjustment.

These considerations led us to enhance the initial publish and approve process to allow several additional features:

- Should be able to make multiple Pending runs before approving one of them. The multiple Pending runs may include different adjustments, risk bucket mappings, or other run parameter changes.
- Should be able to recover if a run is approved by mistake (operator error picking run to approve, or failure to notice a data problem until after run approved).
- An Unofficial run state should be available for runs that are neither approved nor rejected for production Dashboard reports. These are experimental results that need to be saved for ad hoc analysis.

These features led to an implementation of four states for runs. The following state transition diagram shows these states represented by ovals. Arrows represent the operations used to manage run state. The dark circle is the starting point:



Implementing the six operations (run, approve, unapproved, hold, reconsider, and reject) involved some complexity. In particular, insuring that the slowly changing dimension rules are correctly applied to the reporting data mart's dimensional data model and that all updates can be undone were significant challenges both for initial software development and on-going administration of the system. However, the end result is a set of tools that cover the needs of the typical business scenarios and operational issues. The next section will present these tools in more detail, as experienced by the administrative users.

Publish and Approve Tools

ANALYSIS OF PENDING RUNS

The nightly process finishes around 5 AM, a short time before the Operations group starts their morning analysis. The first step in the process is to check that a completion email was received from the Event System, reporting that the nightly process ran to completion. If this email is not in the inbox when the analyst begins, there will typically be an email from an administrator who is working to correct a technical problem with the run.

The nightly process includes a number of Enterprise Guide projects that support reconciling the nightly results. The positions and risk measures are checked for internal consistency and also for consistency against source systems (using direct access or summary spreadsheets received from trading mid-office).

The morning review is also part of the risk oversight function. The Operations analyst reviews key risk measures and creates a daily commentary on risk, explaining changes from the previous day. Trading positions are compared to limits (e.g., notional, loss), and communication is initiated when violations occur.

ADJUSTMENTS

The morning analysis may indicate that the source data need adjustment. An example of an adjustment is positions volume. If a certain position came in with the wrong volume during the overnight process, that position has to be changed to reflect the real volume; the rest of the process that uses that position also needs to run to ensure data integrity in the Reporting Data Mart. The overnight run either remains in pending mode or it is rejected, and the new run is approved, upon verification. The adjustments can be of type *change, delete,* or *add*. Typically we exercise an *add* adjustment, to balance the initial volume. For instance, if a volume for one position came in as 500, but in reality it was 125, a *change* adjustment would modify that record and change its value from 500 to 125. A *delete* and an *add* adjustment can be used in combination to first remove the first record (500) and then add the correct one (125), or an *add* adjustment can be used to enter a new record with the value –375, which would bring the total to 125. This type of adjustment is preferred because we can keep track of the adjustments made, the original value, and what was done to fix that problem.

The Enterprise Risk Management Application provides the ability to create these adjustments and rerun only the portions of the process pertinent to these adjustments. This is a significant timesaving feature, as it may save anywhere from 20 to 60 minutes, depending on the type and volume of adjustments. These time savings are important because any adjustment will delay the availability of approved results to users in the morning.

There are also instances when the entire process needs to run again because some of the source data in the overnight process was either missing or out of date. The source system data might have been fixed or it was loaded later, after the extract process started, and it is available in the morning. This process is usually referred to as a *rerun*.

A *rerun* can be either a full run or a partial extract of the source data but a full run after the extract steps.

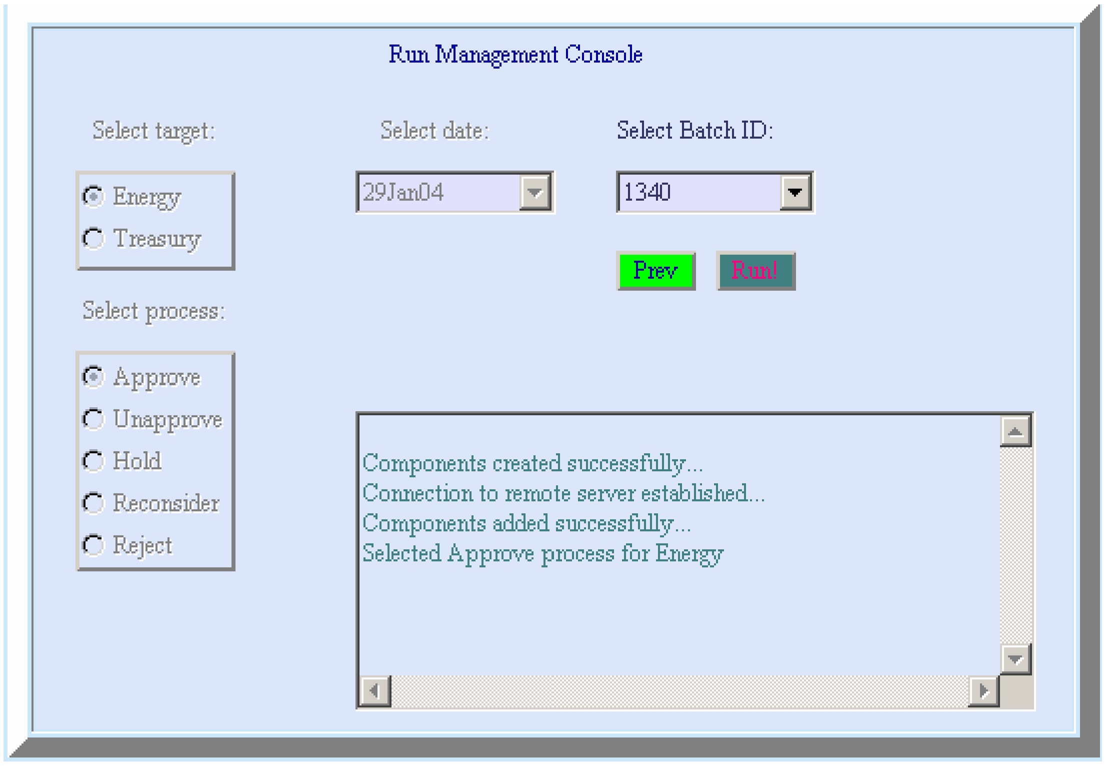
RUN MANAGEMENT USER INTERFACE

Initially, the system was built with the assumption that the publish and approve process would be an administrative task. As such, command line tools were developed and the system administrators ran the process, as requested by the Operations group, after the morning analysis was completed. We discovered, however, that this process involved too many middle-people and it would make more sense to allow authorized users to directly execute these processes. Further, the decision is made solely by the Operations group within Market Risk Oversight. The people in this group are highly qualified business analysts who understand the data and make crucial decisions based on the results produced by the nightly run.

In order to simplify the process, the system administrator created a number of interfaces, one of them shown in the figure below. All of the interfaces are web based and available only to users who have authority to see them. The security mechanism is part of the reporting system. The interfaces are written in Java and they use servlets and RMI in the back-end and applets and servlet-generated HTML in the front-end. Database connection is achieved using a JDBC driver.

The Run Management Interface allows users to place a current pending run in one of the five possible states. The interface has built-in logic to allow for the appropriate operation – one can approve a pending run, but cannot unapprove a pending run, for instance. As shown in the state transition diagram above, a run can only be approved, placed on hold, or rejected when the run is in its initial pending state. In order to execute the reconsider command, the run in question must be in a hold state; a run can only be unapproved if its current state is approved. There is logic to handle remote RMI and JDBC connections, either built directly into the interface or indirectly through the processes invoked by the interface; and informative popup messages inform users about the status of a submitted request.

Should an error occur while the process is running, the user is given an option to view the details associated with that error. When this option is executed, a process queries the *occurred events* table from the Event System and it brings back the events of type *ERROR* associated with that command. The user can then contact the system administrator and provide full information about the process, including detailed error messages received. However, since all the problems are logged in the Event System, system administrators will normally receive the messages automatically through subscriptions.



```
Components created successfully...
Connection to remote server established...
Components added successfully...
Selected Approve process for Energy
```

Figure 3 – Run Management Interface

Other user interfaces were developed to allow various tasks such as updating various notes and news within the reporting application. This allows the Operations group to incorporate comments on the previous day's results into the Dashboard.

Administrative tasks have also been improved by creating an interface to manage users, groups, roles, as well as the default reports settings for individual users and a password reset facility.

Summary

Now that we've covered all the moving parts, we can review a day in the life our analytical warehouse. The day begins in the afternoon. As the business day ends, all new contracts and new market data are recorded in the source deal capture systems. Some nonstandard deals are recorded in spreadsheets and communicated to the Market Risk Oversight organization so that adjustments can be prepared to reflect these positions in the risk system.

The nightly process begins in the early morning hours. The Ant-based Job Flow Control system sequences the SAS programs that extract source data, perform risk analytics, and prepare results for reporting. These steps are monitored by the Event System, so that a system administrator can be notified immediately if there is an error.

When the nightly process finishes, the Event System sends a completion notification to the Operations group. When Operations comes in before 7:00 AM, they begin reviewing the pending results. Reconciliation tests are performed to verify the accuracy of the results. If there are problems, adjustments are created and necessary steps are re-run.

The initial (human) analysis of the run is entered into the system as comments on various reports. Then the pending run is approved, making its risk measures visible to all users of the web-based Dashboard.

As the business users begin their work day, the daily cycle is already half over for the risk system, ERMA. All that's left is to keep serving up the various slice and dice views of the data, until the afternoon comes and the whole process commences again.

References

[ANT] The Apache Ant Project Group, http://ant.apache.org.

[CCRO] Committee of Chief Risk Officers, "Valuation and Risk Metrics White Paper", http://www.ccro.org, 2002.

[EVENT] Barnes Nelson, G.S. Wright, Jeff. “Automated Testing and Real-time Event Management: An Enterprise Notification System“. Presented at the SAS users Group International Conference, Montréal, Canada. May, 2004.

Contact Information

Your comments and questions are valued and encouraged. Please feel free to contact the authors at:

jtanderson@aep.com

greg@thotwave.com

jwright@thotwave.com