



Case Study: An integrated platform for assessing commodities market risk

Introduction

Brown Study Ltd was approached by Trafigura Ltd, one of the largest independent companies trading commodities, to redesign a legacy commodities market risk assessment system. The existing system suffered from a number of deficiencies, notably unacceptably long processing time and a lack of reliability and transparency: it was very difficult to detect and diagnose processing failures, which were occurring often.

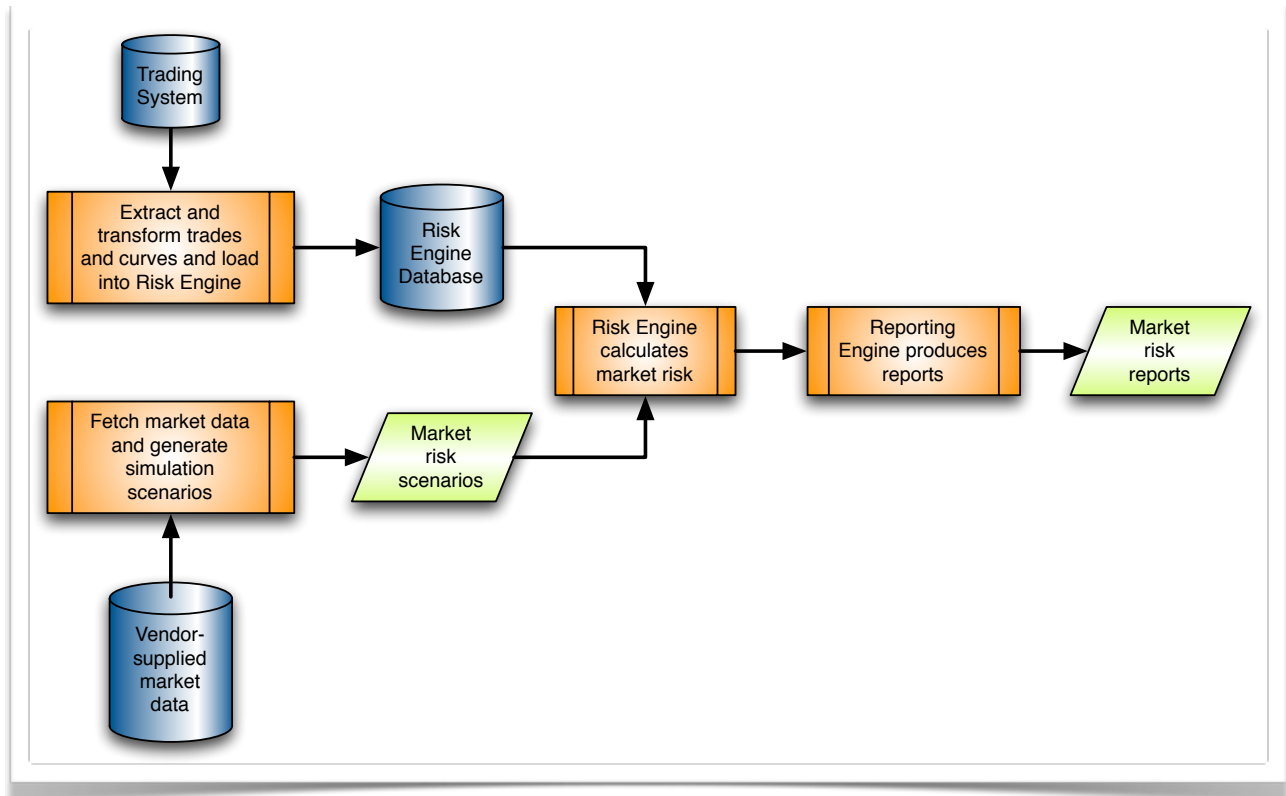
The goal of the project was to create a platform capable of producing a number of sophisticated commodities market risk reports on a daily basis, in a reliable and timely manner. The platform was intended to be simple to maintain and extend as the company grows and trades new classes of financial instruments.

Key Requirements

- ★ **Reliability:** Above all, the new platform was required to be robust. Immediate notification must be provided if processing breaks down, with rich diagnostic information. It must be possible to attach debuggers to running processes if necessary.
- ★ **Transparency:** Operators should be able to inspect the state and data of current and historical processes. All relevant data must be logged and warehoused to allow for technical and financial analysis.
- ★ **Performance:** The aim was to reduce the processing window from 10 hours to 1 hour and enable parallel processing where possible.
- ★ **Standards-based:** The platform would adopt proven open standards for data and communications.
- ★ **Tools:** A tool suite to perform visual, historical analysis of risk analytics would be provided to satisfy internal investigation and regulatory requirements.
- ★ **Portal:** A web portal would be created to facilitate interaction with the platform, including process management and report publishing.
- ★ **Documentation:** All aspects of the design, implementation and maintenance of the system were to be documented in a hyper-linked knowledge management system.

Process Overview

The ultimate goal of the platform is to produce a number of risk analysis reports for each trading entity. To accomplish this, a master process is used to co-ordinate a number of of complex sub-processes that are responsible for the extraction, transportation and transformation of various data.



Architecture

Many of the legacy system's failings stemmed from a haphazard underlying architecture: a multitude of undocumented scripts and script languages were being used to manipulate unstructured data in a point-to-point communications topology.

An early and key decision in the project was to adopt a Service Oriented Architecture (SOA) for the new platform, which would manipulate data structured according to bespoke XML¹ data and messaging schemas. All platform services were to be offered using SOAP² web services.

Proven open-source components were selected to build the platform infrastructure, allowing for complete control over all aspects of processing and providing flexible deployment. Where this was not feasible, best-of-breed proprietary solutions were selected, on the basis that they had adopted open standards for integration and data exchange.

¹ **XML** is the eXtensible Mark-up Language, which allows the creation of structured text documents that can carry rich business semantics.

² **SOAP** is a protocol for exchanging XML-based messages over computer networks. It forms the foundation layer of the web services protocol stack.

Service Implementation

Bespoke Schemas & Adaptors

All data and messages flowing through the platform are represented in a *canonical* form, whose structure and semantics are defined by a central XML Schema³. XSLT⁴ style sheets are used as adaptors: they transform data and messages between the canonical form and proprietary forms used by trading systems and risk analysis components.

This leads to a de-coupling of platform components: any one of the platform's major components can be exchanged without affecting the other components and only the adaptor need be rewritten.

Web Services

All platform services are provided by SOAP web services, based on Apache Tomcat/Axis2. This Java technology stack has the following characteristics:

- ★ **Open-source, open standards:** Apache Tomcat and Axis2 are both lightweight open-source components, based on open standards, such as SOAP. This allows for simple, flexible deployment and integration.
- ★ **Model driven:** Apache Axis2 offers the *Axis2 Databinding Framework* (ADB), which enables automatic code generation from the bespoke schemas. Manipulating and updating the domain model is simplified.
- ★ **WS-I compliant:** Axis2 is compatible with the WS-I Basic profile, ensuring a good level of Web Services interoperability.
- ★ **WS-Addressing:** Axis2 supports the WS-Addressing standard for routing return messages. This is important as many of the services offer long-running operations, which need to perform an asynchronous call-back on completion.



Business Process Engine

A Service Oriented Architecture requires a number of key services. One approach is to implement these as custom web services but these tend to be rigid, requiring effort to change and maintain. An alternative approach is to use a BPEL engine to supply the necessary platform services and perform process co-ordination.

This is a good fit, as producing risk analytics typically involves co-ordinating a number of complex business processes. This approach allows processes to be developed quickly, maintained easily, executed reliably and yields a visual record of each process.



After careful consideration, the *ActiveVOS*⁵ BPEL⁶ engine was selected, as it offers the following features:

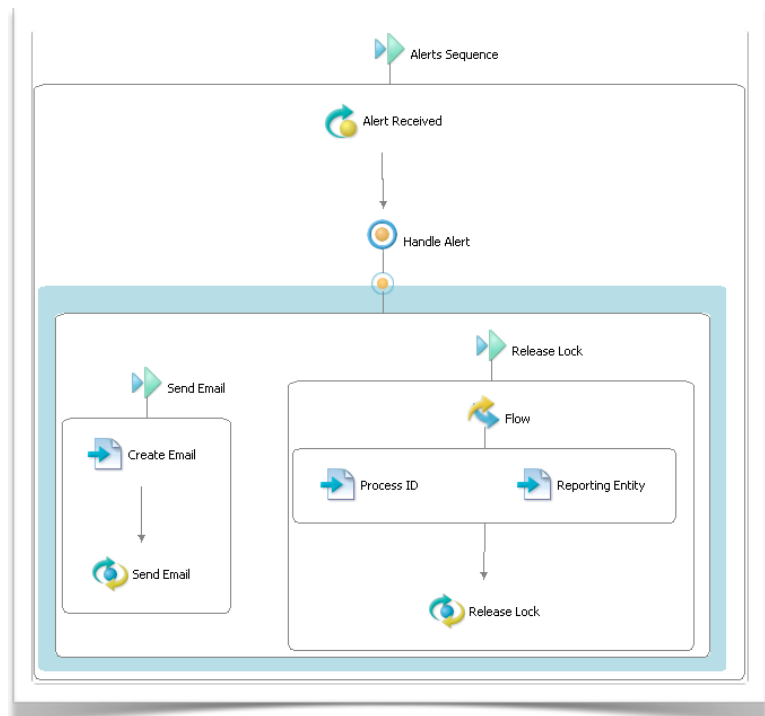
³ An **XML Schema** defines an XML language used to describe the rules governing the structure and content of an XML document.

⁴ **XSLT** is eXtensible Stylesheet Language Transformations, a programming language specialised for the transformation of XML documents.

⁵ **ActiveVOS** is the Active Visual Orchestrator, an implementation of a BPEL engine provided by the vendor Active Endpoints.

⁶ **BPEL** is the **B**usiness **P**rocess **E**xecution **L**anguage, an XML-based language for the formal specification of business processes.

- ★ **Validation services:** Bespoke schemas referenced by WSDL⁷ service definitions are automatically added to the BPEL engine. ActiveVOS will optionally validate all SOAP messages against matching XML Schemas, ensuring that all data flowing through the platform is properly formed.
- ★ **Routing services:** ActiveVOS offers a number of routing abstractions, such as URN⁸s and dynamically generated Partner Definitions that allow sophisticated message routing at runtime. This was a key feature for taking advantage of parallel processing in the Risk Engine, as the BPEL engine was used to dynamically assign work to separate hosts.
- ★ **Transformation services:** All data adaptors are hosted and executed by the BPEL engine, which uses the *Xalan* XSLT transformer. In the case where transformation logic was too complex for XSLT, ActiveVOS supplies a framework for invoking custom Java functions, which was needed to express complex financial business logic. Where simple transformation logic was required, the BPEL engine offers XPath⁹ and XQuery¹⁰ statement execution directly in the process.
- ★ **Service co-ordination:** ActiveVOS excels at the creation of complex business processes. An Eclipse-based designer allows for visual construction and debugging of processes and process Unit tests can be used to drive Test Driven Development. The runtime BPEL engine allows for process versioning, so migrating processes between versions is simple.
- ★ **Service deployment:** ActiveVOS designer provides one-click packaging and deployment of processes to the server, which facilitates an iterative approach to process development.
- ★ **Process versioning:** All processes deployed to the BPEL engine are versioned, which greatly facilitates up and downgrading of processes.
- ★ **Process logs:** The state of all processes and variables are persisted in a database, allowing inspection during and after processing.
- ★ **Fault handling:** ActiveVOS allows processes to be suspended when they fault, allowing for human intervention when necessary and provides an email alerting facility to notify operators of the failure.



⁸ A **URN** is a **Universal Resource Name**, which allows service endpoints to be declared at runtime.

⁹ **XPath** is an XML language that identifies parts of an XML document to be processed.

¹⁰ **XQuery** is a query language (with some programming language features) that is designed to query collections of XML data.

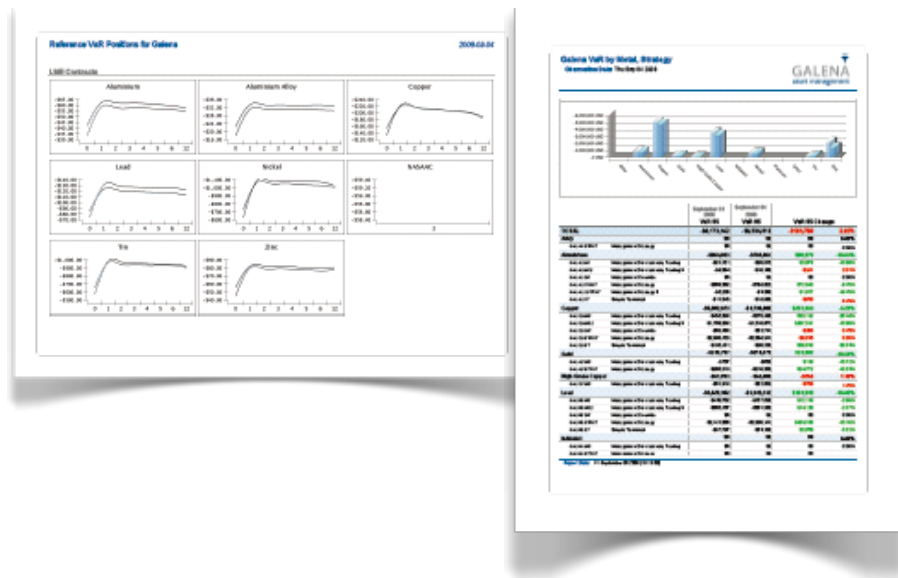
Risk Analysis Engine

The core of the risk analysis platform consists of a number of components supplied by a third party vendor who specialise in assessing market risk. These components are not cohesive and demand interaction via proprietary, legacy mechanisms (CSV¹¹ files, command-line scripts).

Consequently, the components were wrapped with web services, to provide an abstraction layer that allowed them to be integrated using open standards such as XML for data interchange and SOAP for communications.

Reporting Engine

The platform takes advantage of the Eclipse Business Intelligence and Reporting Tools (BIRT) framework to produce a series of sophisticated visual reports in both Excel and PDF formats. The BIRT Engine is integrated into the platform using a custom SOAP web service.



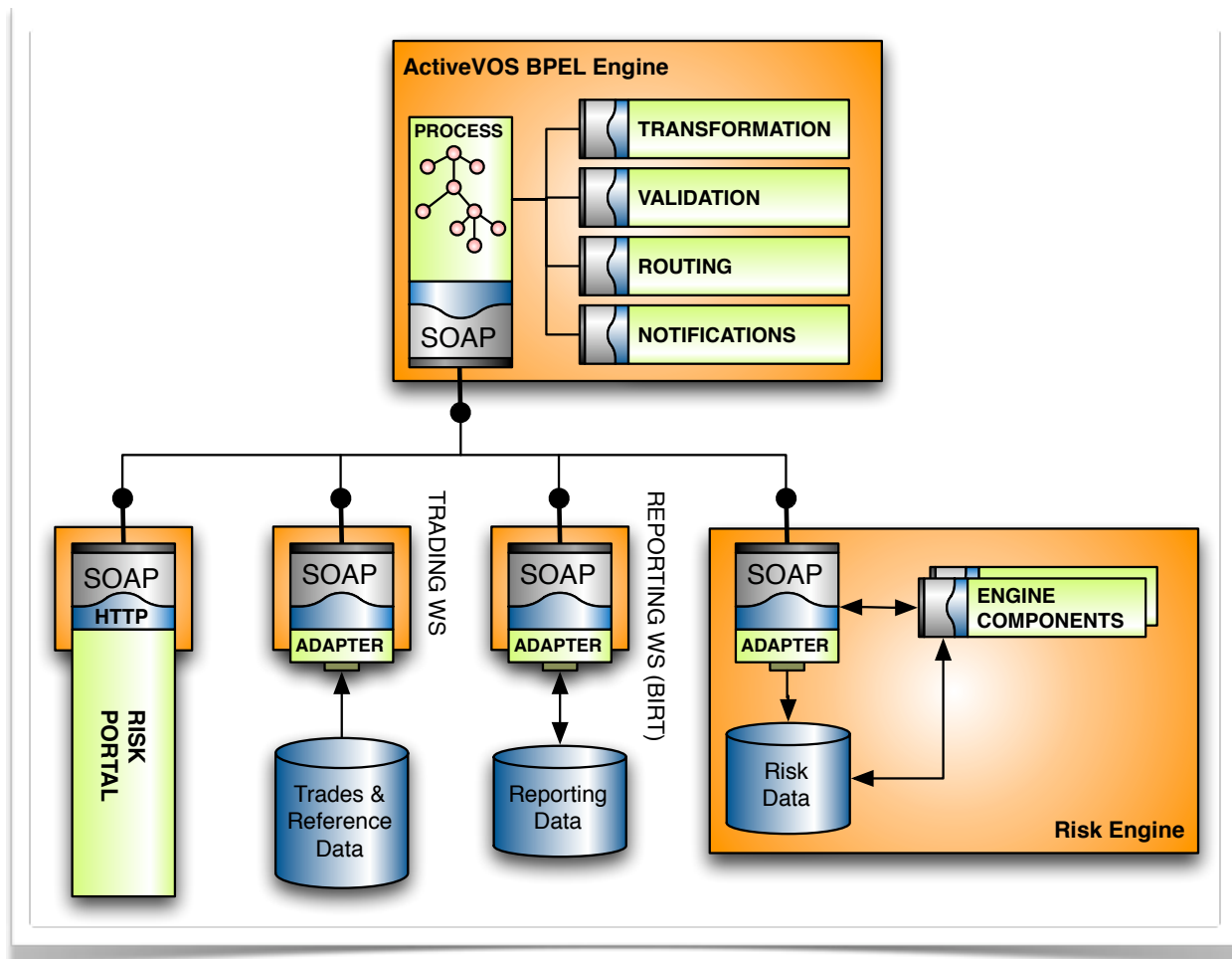
Service Orchestration

As early versions of the platform services began to take shape, it became possible to create rudimentary BPEL processes to integrate and test them. Each service is represented as an abstract endpoint (or *partner*) in the BPEL engine. Endpoint definitions are supplied to the engine in a declarative manner, allowing these to be updated at runtime. This proved particularly useful for maintaining multiple environments for testing and production deployment.

The ActiveVOS Designer also allows a process developer to visually simulate processes using predefined XML data structured according to the schemas, which is indispensable for testing process logic. In addition, BPEL Unit tests (BUnit) were supplied, to allow for regression testing of processes.

Some of the sub-processes need to invoke long-running service methods. To accomplish a true asynchronous service invocation, the BPEL engine can be instructed to produce *WS-Addressing* SOAP headers. These headers include a 'ReplyTo' address, which allows the service to call back to the BPEL engine when it has finished processing. This prevents the process failing due to a broken HTTP connection.

¹¹ CSV are Comma Separated Values, which are used to create unstructured text documents.



Project Review

The platform is now running successfully in a production environment. All of the key requirements have been met and the implementation has proved to be highly robust. The following benefits are being enjoyed by the client:

- ★ Processing time has been drastically reduced. Market risk reports are being produced in 5 to 30 minutes, so they are more relevant.
- ★ Traders are now able to assess unit risk (i.e. how much risk is involved in purchasing a ton of LME Copper in July?), which is driving trading decisions.
- ★ Interacting with the platform is intuitive and pleasant, thanks to the simple Risk Portal.
- ★ Producing the daily risk reports is now stress-free for the operators. Even in the case where the process breaks down, email notifications are received quickly and faulting processes can be inspected visually. Process failure can often be resolved in a matter of minutes.
- ★ The platform is now easy to extend: a new hedge fund can be hooked into the platform in a day or two, rather than the month that was previously required.