

Case Study: **FORTIS**

The Environment

Fortis is an international financial services provider engaged in banking and insurance. With a market capitalization of EUR 38.7 billion (30/04/2006), Fortis ranks among the twenty largest financial institutions in Europe.

Fortis occupies a leading position in banking and insurance in the Benelux (Belgium, Netherlands, Luxemburg) countries. It offers internationally operating companies throughout Europe an integrated network and provides wealthy individuals and business people with advanced services based on a unique set of competences. Fortis's expertise in niche markets such as shipping, commodities, export and project finance and fund administration has made it a regional or world leader in those areas. Fortis also provides retail-banking services in France, Poland and Turkey. Lastly, Fortis successfully combines its banking and insurance expertise in growth markets in Europe and Asia and leads the bancassurance markets in Spain and Portugal.¹

With several hundred financial software applications being developed every day, and new demands being added all the time, Fortis software development processes have had to be fast and productive while still being managed in a secure and auditable way.

The Challenge

As Fortis continued to grow and expand across Europe and beyond, so too did the scale and complexity of managing their software development processes. To complicate matters, years of mergers, acquisitions and management transitions created fragmented Software Change and Configuration Management (SCCM) processes in the various regions Fortis occupies. From a business perspective, having one common set of SCCM and build

management procedures for the entire organization became a requirement. The centralization and protection of valuable software assets and the streamlining of developing mission-critical applications is a top priority. Using proven existing technologies to assist in this standardization, as opposed to developing time consuming and costly in house solutions, was the only feasible way to meet the challenge.

Fortis' SCCM restructuring initiative led to major centralization in Brussels, Belgium of standards and procedures for the entire Benelux region. While much of their SCCM processes had already been centralized using Computer Associates' CA SCM for software change and configuration management, an equal level of centralization was needed in order to replace the "black box" of Ant/XML and Make scripts that were used to configure and execute the application build process.

Often a disparate, developer driven, scripted process, the compilation and assembly of application build components needed to be brought under central administrative control for the entire Benelux region. Fortis SCCM Belgium understood that the application build process pieces together all of the developed components and brings to realization an application's changes. Unless builds were centrally managed and their procedures standardized, the integrity of the full end to end SCCM process would never be realized.

As part of this streamlining process, Fortis' SCCM team in Belgium recognized the immense benefits associated to entirely eliminating the hard coded ad-hoc scripted build processes. With agile development requirements and quick to market demands of their end users, Fortis SCCM team in Belgium identified the "black box" of build scripts as being a

primary bottleneck in streamlining, measuring, and managing their development to release process.

The Solution

Having already successfully implemented Catalyst's Openmake Build Management solution in Brussels for all their Java J2EE builds, it was decided to bring the same level of build standardization to the other parts of the organization in Luxemburg and the Netherlands. This, they decided would not only allow Fortis to have one maintainable build solution for the entire enterprise, but they also knew that moving forward, should any new build process be required (for example, .NET), Openmake, with its support of 200+ languages out of the box, would be able to handle the new requirements.

"Although Openmake Meister already offers, on a daily basis, increased productivity and quality to the Fortis WebSphere Competence Centers (150 persons), compared to manually controlled builds, Openmake also completes the configuration management process through its central build capabilities. Owing to this, Fortis fully controls all internal and external software assets completely within the very fast evolving J2EE world," says Matthias Pyck.

OpenMake Meister version 6.3 had already eliminated the need for scripted build procedures for Brussels development. All J2EE builds were managed on one centralized Knowledge Base Server in Brussels by the SCCM team. Now they had to do the same for the other countries. Additionally, the new features of 6.4.1, with its built in Lifecycle Application Management features further assisted them with reducing the scripting before and after the build. In order to successfully implement OpenMake Meister across all three

¹ overview information from Fortis website <http://www.fortis.com/General/brief.asp> 06/15/2006

countries, Fortis knew that they needed to get "buy in" from the other development teams. Using Openmake Meister in conjunction with CA SCM they believed they could create the level of automation and ease of use required for mass acceptance. They also wanted to allow teams in all three countries to run their builds using Openmake Meister's Remote Build (RB) Server technology. The RB machines meant that at any moment, remote builds could be executed from one machine to another across all three countries. Still under central administrative control, however, these builds could be monitored and maintained in real time by the SCCM team in Brussels, using OpenMake Meister's Build Manager console.

Fortis's ultimate objective was to complete end to end automation of the checkout, build and check-in, executed from a UDP (User Defined Procedure) in CA's Software Configuration Manager (CA SCM) In addition, they eventually would do away with any scripted procedures, using OpenMake Meister's activity management features offered to manage the end to end process. They also would use OpenMake Meister's built in scheduling to automate their builds at regular intervals, catching errors as they occur. Importantly, this would all need to be centrally managed with build and audit reports, emails of build results and real time monitoring available to the SCCM team in Brussels.

The Results

Because of impending rollout deadlines within Fortis, an aggressive schedule of 2 weeks was designated for laying the groundwork for the migration from OpenMake Meister 6.3 to 6.4.1 in Brussels and abroad. This meant taking an existing build ready application in 6.3, migrating it to 6.4.1 and building it in all three countries on different build machines against one centralized KB Server in Brussels. Since OpenMake Meister was not currently in use in Luxemburg and the Netherlands, fresh installations were required.

Using a well thought out plan developed by Fortis and OpenMake Software, that included travel to all

three countries, an implementation incorporating all of 6.4.1's advanced build management features was achieved. In order to accomplish this mission, the OpenMake Software expert and the Fortis' SCCM specialists worked closely as a team to tackle each challenge. The first challenge, addressed before the OpenMake Software consultant was onsite was to ensure the required ports were opened in the network firewalls between the different countries. This allowed the various, Meister Remote Agents, Clients and Meister Knowledgebase (KB) Server components to talk to each other and share information while executing end to end builds across all countries. Once networkability was guaranteed, installation in all three countries began.

The first component installation was the KB Server. Although OpenMake Meister is "out of the box" ready for installation with an Apache Tomcat layer, because it is a J2EE compliant application, it can also be deployed to other J2EE application servers. Fortis' standard was WebSphere Application Server (WAS) 5.1.1.9. With some minor configuration using the WAS administrative console, the KB Server EAR file was successfully deployed in Brussels to both DEV and QA Meister testing states.

After Meister's "backbone" was in place, Remote Agents and Build Manager Clients could now be installed. A total of eight Remote Agents were added across all three countries and installed to both Unix and Windows servers. In each case, where a Remote Agent was installed, so was a Meister Client.

Following installation, without any Ant or Make scripting required, a test application was built in exactly the same way on all eight build servers. These builds were all executed from one machine in Brussels and while the build was running, real time monitoring was sent to all Meister Clients, providing useful information if and when any compile errors occurred. Additionally, all build logs and audit reports were stored centrally on the KB Server regardless of the location of the build execution. Now that the compilation portion of the end to end build was successfully

implemented, it was time to incorporate the advanced Lifecycle Automation features offered in 6.4.1. This meant complete centralized control of all processes around the build. The following Lifecycle Automation activities were successfully incorporated across the entire cross boarder network and could be executed from inside the CA SCM Workbench:

- Prebuild processing, including the creation of a build area and the checkout of a desired CA SCMPProject and State.
- Generation of a dynamic Build Control File containing all Target and Dependency information.
- Real time dependency analysis and build execution, compiling and reporting of Java application components.
- Post build processing, including the check-in of all successfully built components.
- Email notification sent to team members with a hyperlink to a detailed build results log stored on the KB Server.

In addition, all of the activities configured in Meister were conditionally related, so that successive steps could be contingent on the success, failure or completion of dependent activities, leaving nothing to surprise. Scheduling functionality was also used to eliminate all manual intervention of the end to end build.

Moving forward, Fortis plans to separate and add even more pre and post processing activities to the mix. In addition they expect to be building non Java components in the near future. With OpenMake Meister 6.4.1, they now can feel confident that no matter what language they need to compile, what platform they need to compile on and what country they need to run their full end to end builds in, they have a build management solution scalable to fit all configurations – no scripting required!



www.openmakesoftware.com
(800) 359-8049 • (312) 440-9545