

Industrial Placement Projects Presentation

Alex Michael

Contents

- UBS – The company
- Market Risk IT
- Projects
 - Arisk Testing Framework – Informatica
 - Separation of a code base
 - Buzzy
- Thoughts
- Conclusion

UBS – The company

- Global firm headquartered in Zurich and Basel.
- Established in 1998 from the merge of the Union Bank of Switzerland with the Swiss Bank Corporation, but its roots date as back as the second half of the 19th century.
- Employs more than 64000 people and provides a variety of financial services to clients worldwide.
- The operational structure of UBS comprises of the Corporate Centre and four business divisions
 - Wealth Management & Swiss Bank
 - Wealth Management Americas
 - Global Asset Management
 - Investment Bank

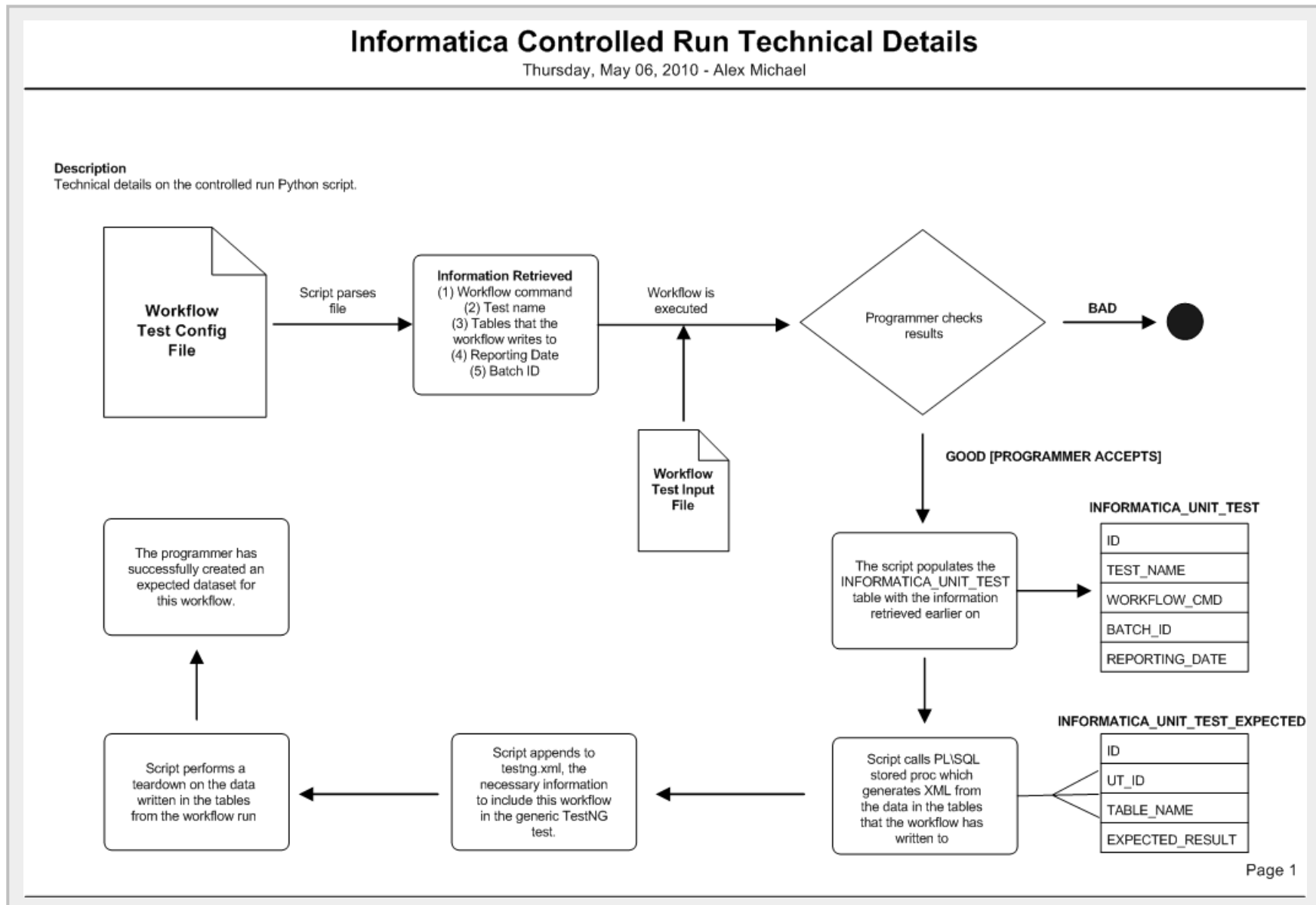
Market Risk IT

- Market Risk is the risk of losing money from your portfolio, due to changes in the market.
- Main development effort of Market Risk IT is ARISK.
- ARISK is an aggregation platform that aims to be a single source repository for all front office FICC and Equities risk information. This will enable middle office to calculate the risk that UBS is exposed to and produce reports for the business side.
- My projects revolved around ARISK.

[Project 1] ARISK Testing Framework – Informatica

- Informatica PowerCenter is an enterprise application for performing ETL functions on any set of data. ETL stands for Extract, Transform and Load. It is mainly used for data warehousing
- The files produced by the PowerCenter developer are called “workflows” and the purpose of this project was to devise a testing approach for them.
- There is no standardized way of testing workflows in the enterprise world. Testing is mostly done manually by the developer by examining the generated dataset.
- I tried to automate this process so it can be integrated into a testing framework for the whole ARISK project.
- Checking the correctness and integrity of the newly generated dataset is done by comparing and contrasting against an expected dataset.
- Reports are produced so the developer can have a detailed view of what’s gone wrong.

Creating a test for a workflow (the Controlled Run)



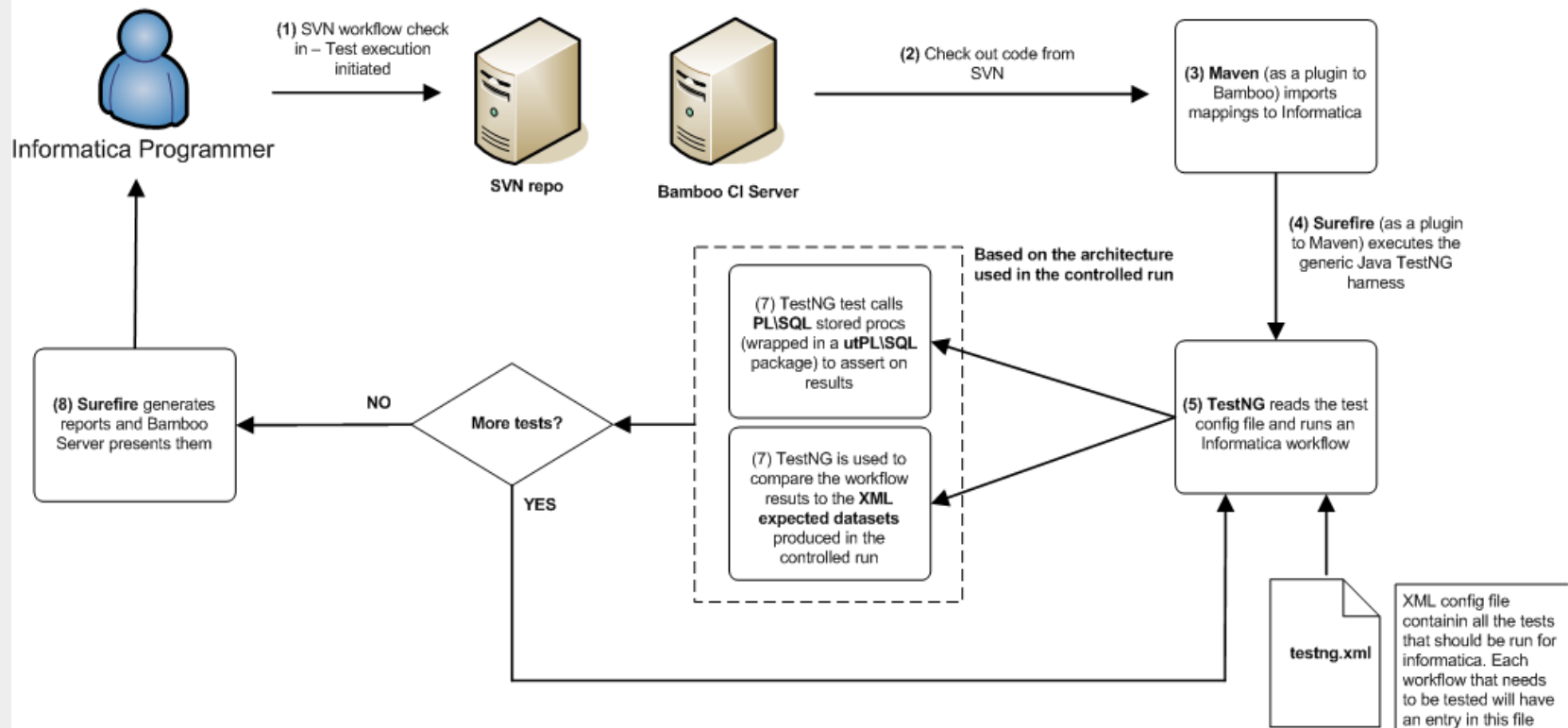
Testing a workflow as part of the framework

Informatica CI Unit Testing Overview

Friday, April 30, 2010 - Alex Michael

Purpose

To automate the testing of Informatica workflows through a Continuous Integration plan. The process is mediated by Maven and the testing is controlled by a generic Java TestNG harness which will invoke the workflow, and assert on the execution and results produced.



Page 1

[Project 2] Separation of a code base

- Loader project: Acts as a controller for the ARISK framework. Receives feeds from the front office applications, registers them with the database and delegates work to other components.
- Transformation project: Receives a slice ID, lookups the necessary transformations for that slice in the database and performs those transformations in priority order.
- A slice is a term used to group related risk information.
- Clearly, the two projects perform different tasks so they should not be under the same project.
- Performance is another reason.

The solution

- Since the two projects will run as two distinct applications, there needs to be a mechanism of communication with each other. Messaging is an ideal fit here.
- After research, I decided that the best solution was to use a message queue. The Loader will leave a message to the queue for the Transformation to pick it up and proceed.
- The Transformation project was moved in its own package and an interface for picking up messages from the queue was created.
- Similarly the Loader code was refactored to send messages to the queue instead of calling the Transformation service directly. The interface for sending messages was made as generic as possible, to allow for future communication with other types of applications and not just the Transformation.
- The queue solution left the team happy since it has two major advantages.
 - Scaling and performance
 - Message persistence and hence greater reliability and fault-tolerance.

[Project 3] Buzzy – A simple JMS web interface

- Buzzy is a lightweight and simple web application built on Spring Web MVC, useful for sending JMS messages to pre-defined and custom destinations.

Buzzy

Send

Browse

Choose a destination

ruff.dev.rollrequest.queue

Your message

Custom

Destination name: test.testQueue

Destination type: Queue

Destination url: tcp://localhost:61616

View messages

☐ Autorefresh

Message ID	Destination	Reply To	Priority	Timestamp
ID:wldn0173417-1722-1281522557361-0:0:1:1:1	queue://test.testQueue	null	4	Wed Aug 11 11:29:17 BST 2010

Message:
This is a testing message

Thoughts

- Overall, a very enjoyable and rewarding experience
 - Taste of working in the industry
 - Met new people
 - Been to new places
- Improved my technical skills...
 - oracle databases, unit and integration testing on large software
- .. and soft skills
 - teamwork, collaboration, communication.
- Helpful DoC courses
 - Software Engineering
 - Distributed Systems
 - Databases

Conclusion

- Working in Market Risk IT
- ARISK platform
- Projects:
 - ARISK testing framework – Informatica PowerCenter
 - Separation of a code base
 - Buzzy
- **Questions?**