Portlet Development in Exo

Practice and Experience: Developing and Deploying Tools and Services in the Emerging Portal Frameworks

VRE Developers Workshop 18 - 19 January 2006

Vikas Deora, Arnaud Contes and Omer Rana
Cardiff University

Contents

- EU Provenance Project
- Why Portal Framework
- Comparison of Portal Framework
- Portlet developed using Exo
- Demo
- Problems
- Lessons learnt
- Conclusion

EU Provenance Project

- **IBM United Kingdom Limited**
  - Alexis Biller
  - John Ibbotson
  - Neil Hardman
- **University of Southampton**
  - Luc Moreau
  - Paul Groth
  - Simon Miles
  - Victor Tan
- **University of Wales, Cardiff**
  - Arnaud Contes
  - Omer Rana
  - Vikas Deora
- **German Aerospace Center (DLR)**
  - Andreas Schreiber
  - Guy K. Kloss
- **Universitat Politecnica de Catalunya**
  - Javier Vazquez
  - Sergio Alvarez
  - Steven Willmott
- **MTA SZTAKI Computer and Automation Research Institute, Hungarian Academy of Sciences**
  - Arpad Andics
  - Laszlo Varga
  - Tamás Kifor

EU Provenance Project

- Today's grid architectures suffer from limitations, such as lack of mechanisms to trace results and infrastructures to build up trusted networks.

- Provenance enables users to trace how a particular result has been arrived at by identifying the individual and aggregated services that produced a particular output.

Project Requirement

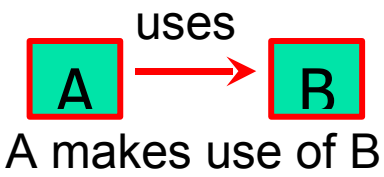
- In order to achieve the provenance project aim, following support was required:
  - a scalable and secure architecture
  - an open proposal for standardising the protocols and data structures
  - a set of tools for configuring and using the provenance architecture
  - an open source reference implementation
  - a deployment and validation in industrial context.

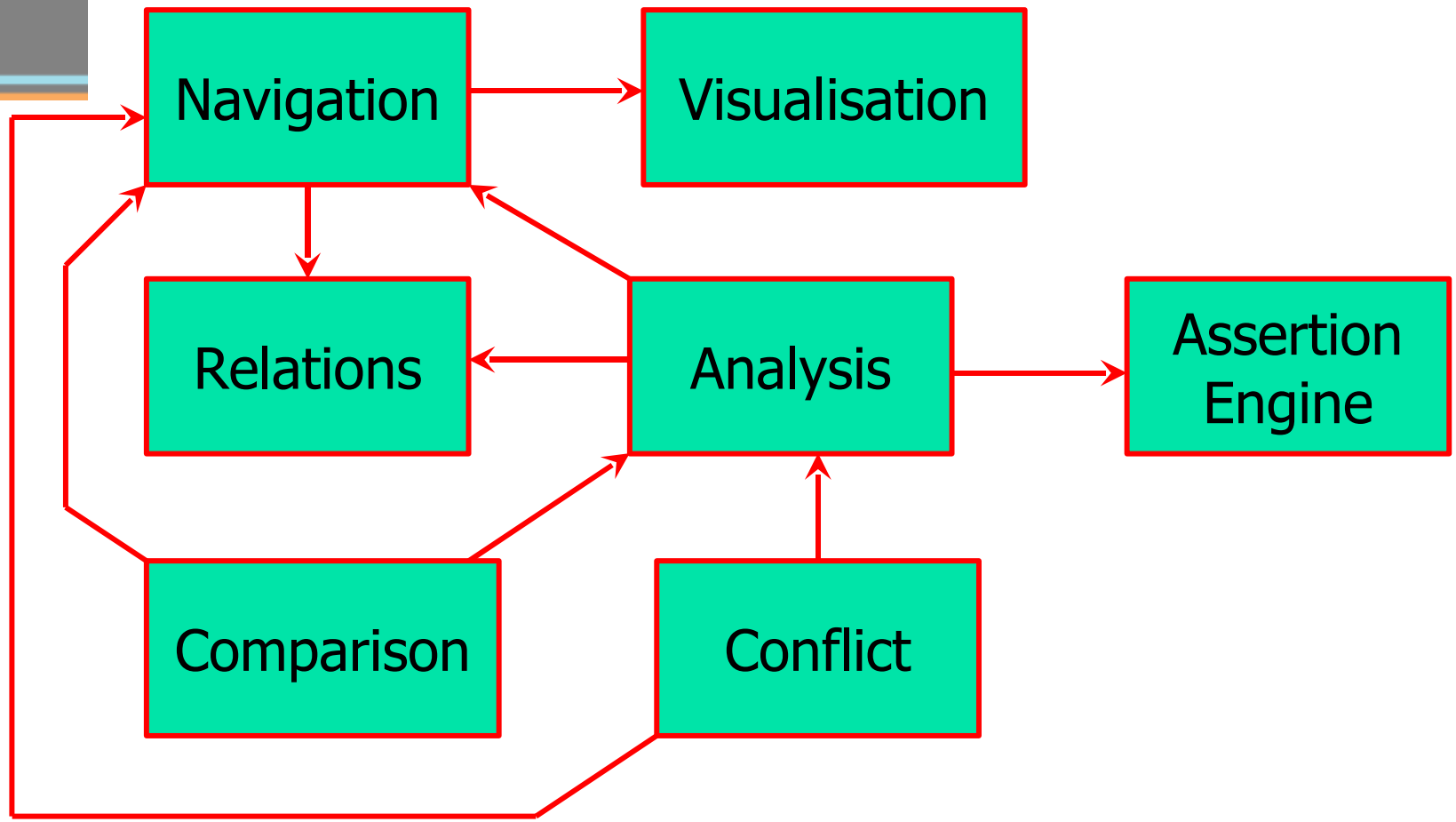
Project Tool Suite

- The aim of the tool suite developed at Cardiff was to allow the information's history, including the processes that created and modified it, to be inspected, validated and reasoned about by authorised users.

- The target applications
  - aerospace engineering and
  - organ transplant management

Tools overview





How to implement Tool suite?

- We considered following interfaces to implement the tool suite:
  - Standalone Java Swing programs
  - HTML and JSP providing browser based access
  - JAVA Applets providing browser based access
  - Portal and Portlets providing browser based access

Requirements

- The implementation of the tool suite was dependent on various application requirement, most importantly which included:
  - A access mechanism that allows application users to gain access to distributed provenance sources seamlessly. For example, in organ transplant application a patient records could be held at different hospitals. This needs to be aggregated and displayed quickly and reliably.
  - Provide customised content to the application users. For example in organ transplant application a doctor performing the organ transplant surgery needs different information than a administration staff at the hospital.

Why Portal Framework

- The decision to choose portal framework was based on the capability of portal framework to fit rightly with our application requirements. Most importantly, the portal framework provided us with following features:
  - Customization: Application users (for example doctors) get straight to the information they need, thus resulting in quick decision making.
  - Aggregation: Aggregates information from different sources to be displayed in one place. For example, the information from different hospital about a patient, logistic information and other information can all be aggregated and displayed in one web page.

Why Portal Framework

- Single sign-on: Allowed access to distributed information sources without having to authenticate again. This provided with increased efficiency in decision making for application users (doctors for example). For example a doctor can have access to patient records, blood banks, logistic data from different systems without having to authenticate again and again.
- Personalisation: Allowed application users to personalise information according to their needs. For example a doctor performing organ transplant might require different information than a doctor performing routine medical tests.

JSR 168 and WSRP

- Java Specification Request (JSR) 168
  - JSR 168 enables interoperability among portlets and portals.
  - JSR 168 establishes a standard API for creating portlets thus enabling them to be deployable under any JSR 168 compliant container.
- Web Services for Remote Portlets (WSRP)
  - Enables interactive, presentation-oriented web services to be easily plugged into standards-compliant portals (http://www.oasis-open.org/)
  - The standard allows a portlet to be published as a WSRP producer, which can then be consumed by one or many compliant portals (WSRP consumers).
  - The standared provides ease of adding remote portlets on a portal page just like adding any other local portlet.

Portal framework

- In order to select a portal framework to use, we analysed different Portal frameworks based on a set of custom criteria that represented requirements suited to the project needs and ease of development.
- Many portal framework exists both open-source and commercial based, our project required use of open source portal framework, thus only open source portal frameworks were compared.
- Also many open source portal frameworks exists and due to time constraints only few were selected to be evaluated for selection purposes.

Portal selection matrix

Criteria	eXo	Liferay	Gridsphere	Jetspeed1	uPortal	JBoss
1. JSR 168 and WSRP compliance	☺☺☺	☺☺☺	☺ (no WSRP compliance)	☺ (no WSRP compliance)	☺☺☺	☺ (not truly compliant)
2. Ease of initial setup and documentation – ease of initial start-up from download to first running of portal	☺☺☺	☺ Difficult to setup initially and lack of documentation	☺☺ Average	☺☺ Average	☺ Difficult to setup	☺ Setup is easy but lack of documentation
3. Community – how popular the portal tool is in various forums and blogs to share knowledge	☺☺☺	☺☺	☺☺	☺☺☺	☺☺	☺
4. IDE integration – this allows ease of development and debugging	☺☺☺	☺	☺	☺	☺	☺☺
5. Sample portlets – this is quite essential due to lack of documentation linked to most free portal software	☺	☺☺☺	☺	☺	☺	☺☺

Average - ☺ Good - ☺☺ Very Good - ☺☺☺

Portal selection matrix

Criteria	eXo	Liferay	Gridsphere	Jetspeed 1	uPortal	JBoss
6. Ease of developing new portal pages and linking to a navigation portlet	☺☺☺	☺☺☺	☺☺	☺☺☺	☺	☺☺
7. Ease of portlet deployment – A interface to achieve this can considerably save development time, specially during debugging stage	☺☺☺	☺	☺	☺	☺	☺
8. Dependency on servlet container – compatibly with only single server environment is very restrictive for web deployment	☺☺	☺☺☺	☺☺	☺☺	☺	☺
9. Performance	☺☺	☺ (crashes a lot)	☺	☺☺	☺☺	☺☺
10. Security setup	☺☺☺	☺☺☺	☺☺☺	☺☺☺	☺☺☺	☺☺☺
11. Portal management tasks	☺☺☺	☺☺	☺☺	☺☺☺	☺☺	☺☺
Final Conclusion	**29**	**23**	**18**	**22**	**18**	**18**

Portal framework selection

- We decided to implement a portal framework based on eXo portal framework.

- The first implementation decision was to create a simple navigation tool portlet to provide some of the navigation capabilities required out of the navigation tool part of the tool suite.

Navigation Portlet

- Navigation portlet consist of two portlets currently (1) the tree navigation portlet and (2) the XPath navigation portlet
- Tree Navigation portlet currently provide capabilities to accept provenance assertions (in form of xml) to create a tree graph object to help visual navigation through provenance store.
- XPath navigation portlet provides an interface for Xpath query navigation, which helps analyse Xpath expressions.

Workflow re-construction portlet

- A workflow re-construction portlet is being developed to inspect the provenance information's history, including the processes that created and modified it.
- The workflow portlet would also assist in visual validation of how a piece of provenance information was derived and if it was indeed the right means of achieving it.
- This portlet is currently in a early development stage.

Demo

Problems faced in Portlet development

- Directory Structure: How the resources that a portlet need to access are placed is very important in order for the portlets to work as required.

- Database Connection: There is a problem in accessing provenance data stores, as the applet can only talk to the web server that holds the applet. Any communication with the outside web server is restricted.

Problems faced in Portlet development

- Inter-portlet communication: Found inter-portlet to be a problem while using applet based interface. Still needs to answer bigger questions, such as how dependencies between portlets are resolved when one or the other portlet seize to exist.

- Visual interface of exo portal: We found it difficult to manipulate the portal interface that exo provides for displaying portlets.

Lessons learnt

- Database Connection: Using a intermediate Java program that have access to the required resource OR by using applet certification both at the client and server end, we were able to get around the problem.
- Directory Structure: By placing the resources that portlets need to access outside the WEB-INF folder of the application folder we were able to solve the resource access problem.
- Inter-portlet communication: A simple approach to allow independent portlets to exist by providing input medium of required data was implemented.

Conclusion

- We have taken a step into creating a implementation of tools for provenance information inspection and validation using the emerging portal framework, we hope to in future exploit the features that the portal framework provides us into further improving the tools and interface provided to application users of provenance system.