> Semantic Web Use Cases and Case Studies

Case Study: Open Services Lifecycle Collaboration framework based on Linked Data

Arnaud Le Hors, IBM, and Steve Speicher, IBM

November 2012

General Description

Introduction

The Rational group in IBM has for several years been employing a read/write usage of Linked Data as an architectural style for integrating a suite of applications, and we have shipped commercial products using this technology. We have found that this read/write usage of Linked Data has helped us solve several perennial problems that we had been unable to successfully solve with other application integration architectural styles that we have explored in the past. The applications IBM Rational has integrated are primarily in the domain of Application Lifecycle Management (ALM) but other groups such as IBM Tivoli are following suit in domains such as Integrated Service Management (ISM). We believe that our experiences using read/write Linked Data to solve application integration problems could be broadly relevant and applicable within the IT industry.

The integration challenge

IBM Rational is a vendor of industry leading system and software development tools, particularly those that support the general software development process such as bug tracking, requirements management and test management tools. Like many vendors who sell multiple applications, we have seen strong customer demand for better support of more complete business processes - in our case system and software development processes - that span the roles, tasks and data addressed by multiple tools. While answering this demand within the realm of a single vendor offering made of many different products can be challenging it quickly becomes unmanageable when customers want to mix in products from other vendors as well as their own homegrown components.

These problems, encountered in many application domains, have existed for many years, and our industry has tried several different architectural approaches to address the problem of integrating the various products these complex scenarios require. Here are a few:

1. Implement some sort of Application Programming Interface (API) for each application, and then, in each application, implement "glue code" that exploits the APIs of other applications to link them together.
2. Design a single database to store the data of multiple applications, and implement each of the applications against this database. In the software development tools business, these databases are often called "repositories".
3. Implement a central "hub" or "bus" that orchestrates the broader business process by exploiting the APIs described in option 1 above.

While a discussion of the failings of each of these approaches is outside the scope of this document it is fair to say that although each one of them has its adherents and can point to some successes, none of them is wholly satisfactory. So, we decided to look for an alternative and embarked in a search that led to us Linked Data.

The solution: Using read-write Linked Data as an application integration architecture

We wanted an architecture that is minimalist, loosely coupled, had a standard data representation, kept the barriers to entry low and could be supported by existing applications implemented with many implementation technologies. We realized that one such solution already existed: Linked Data.

Linked Data, as defined by Tim Berners-Lee, builds on the same principles that have made the World Wide Web so successful: it works in terms of protocols and resource formats rather than application specific interfaces. The web of documents uses HTTP and HTML. Linked Data uses HTTP and RDF. Both use URIs to identify resources allowing to easily express arbitrary linkage between resources, independently of their domain.

Applying Linked Data to the ALM domain integration problem meant thinking in terms of domain specific resources, such as requirements, change requests, and defects, and access to these resources rather than in terms of tools. We stopped thinking of the applications as being the central concept of the architecture and instead started to focus on the resources.

We redesigned our architecture around a web of resources from the various application domains – in our case, change management or quality management etc. -, in which applications are mere handlers of HTTP requests for those resources.

We used RDF to model the different types of resources we needed and the relationships between them. Thus a change request became a resource exposed as RDF that can be linked to the defect it is to address, and a test to use to validate the change to be made. With Linked Data the change management, defect management, and test management tools no longer connect to each other via specific interfaces but simply access the resources directly, following the Linked Data principles.

Open Services Lifecycle Collaboration (OSLC)

To support integration between products from various vendors, rather than defining a proprietary solution, IBM launched the Open Services Lifecycle Collaboration initiative and with others we have developed a set of specifications that build on Linked Data and provide a framework for application integration. Started with a focus on ALM OSLC now extends in other

domains such as ISM and Product Lifecycle Management (PLM).

Figure 1 depicts OSLC and the IBM Rational Jazz platform on which the Rational products build to provide a more collaborative, productive and transparent software and systems delivery, through integration of information and tasks across the phases of the lifecycle..

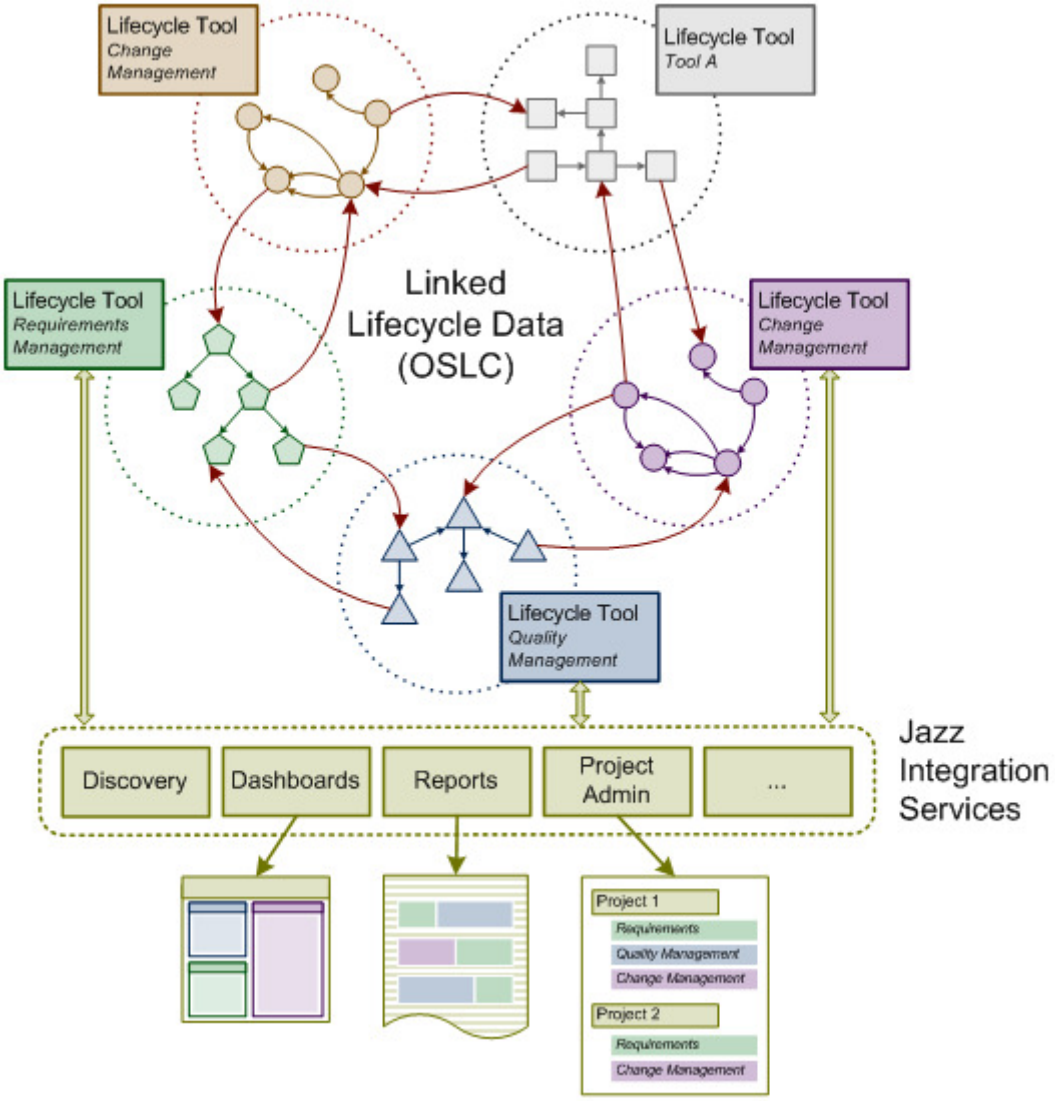

Figure 1: Open Services Lifecycle Collaboration and IBM Rational's Jazz Platform

Conclusion

We have shipped a number of products using the Linked Data technology as a way to integrate ALM products and are generally pleased with the result. We now have more products in development that use these technologies and are seeing a strong interest in this approach in other parts of our company.

As more data gets exposed using Linked Data we believe we will be able to do even more for our customers, with a set of integration services with richer capabilities such as traceability across relationships, impact analysis and deep querying capabilities. Additionally, we will be able to develop higher level analytics, reports, and dashboards providing data from multiple products across different domains. We will be able to answer questions such as: what

enhancements in today's build address requirements that need to be tested with certain test cases?

We believe that Linked Data has the potential to solve some important problems that have frustrated the IT industry for many years, or at least make significant advances in that direction, and we are looking forward to being able to leverage the outcome of the W3C Linked Data Platform Working Group.

Key Benefits of Using Semantic Web Technology

- RDF provides a data model that is very flexible, enables interoperability and extensibility.
- Following the basic principles of the World-Wide Web, Linked Data enjoys the same characteristics: distributed, scalable, reliable, extensible, simple, and equitable.
- RDF provides a foundation for rich capabilities such as impact analysis, reporting, and deep querying across multiple domains.