

Design of a Model-Generated Repository as a Service for USDL

Keith Duddy
Queensland University of
Technology
2 George St
Brisbane
QLD 4000
Australia
keith.duddy@qut.edu.au

Alejandro Metke Jimenez
Queensland University of
Technology
2 George St
Brisbane
QLD 4000
Australia
a.metke@qut.edu.au

Michael Henderson
Queensland University of
Technology
2 George St
Brisbane
QLD 4000
Australia
michael.henderson@qut.edu.au

Jim Steel
Queensland University of
Technology
2 George St
Brisbane
QLD 4000
Australia
james.steel@qut.edu.au

ABSTRACT

SAP and its research partners have been developing a language for describing details of Services from various view-points called the Unified Service Description Language (USDL) [12]. At the time of writing, version 3.0 describes technical implementation aspects of services, as well as stakeholders, pricing, lifecycle, and availability. Work is also underway to address other business and legal aspects of services. This language is designed to be used in service portfolio management, with a repository of service descriptions being available to various stakeholders in an organisation to allow for service prioritisation, development, deployment and lifecycle management.

The structure of the USDL metadata is specified using an object-oriented metamodel that conforms to UML, MOF and EMF Ecore. As such it is amenable to code generation for implementations of repositories that store service description instances. Although Web services toolkits can be used to make these programming language objects available as a set of Web services, the practicalities of writing distributed clients against over one hundred class definitions, containing several hundred attributes, will make for very large WSDL interfaces and highly inefficient "chatty" implementations.

This paper gives the high-level design for a completely model generated repository for any version of USDL (or any other data-only metamodel), which uses the Eclipse Modelling Framework's Java code generation, along with several open source plugins to create a robust, transactional repository running in a Java application with a relational datastore. However, the repository exposes a generated WSDL interface at a coarse granularity, suitable for distributed client code and user-interface creation. It uses heuristics to drive code generation to bridge between the Web service and EMF granularities.