SERVICE ENGINEERING LAB, QUEENSLAND UNIVERSITY OF TECHNOLOGY

Requirements for the Frond End of Service Integration Accelerator

Fuguo Wei

August 9, 2015

1 THE OVERVIEW OF THE PROJECT

This project proposes a novel framework and algorithms for the analysis of Web service interfaces to improve the efficiency of application integration in wide-spanning business networks. Our approach addresses the notorious issue of large and overloaded operational signatures, which are becoming increasingly prevalent on the Internet and being opened up for third-party service aggregation. Extending upon existing techniques used to refactor service interfaces based on derived artefacts of applications, namely business entities, we propose heuristics for deriving relations between business entities, and in turn, deriving permissible orders in which operations are invoked. As a result, service operations are refactored on business entity CRUD which then leads to behavioural models generated, thus supportive of fine-grained and flexible service discovery, composition and interaction. A prototypical implementation and analysis of web services, including those of commercial logistic systems (FedEx), are used to validate the algorithms and open up further insights into service interface synthesis. This prototype analyses service interfaces and generates BE data models for CRUD of a business entity. This prototype has two parts: the back

end interface analysis and front end visualisation. The former is implemented in Java and the later is in Java Script, CSS and HTML.

2 Requirements

The home page allows users to load a service specification (i.e., a WSDL file), and passes it to the back end system for analysis. The results from the back end indicate the operations revealed in the specification, and the front end should then list the name of these operations under the name of the service. Users can then either right-click the name of the service or or each operation. A menu should pop up when right-clicking and it should have two options: hierarchical structural view and business entity data model. When users choose hierarchical structural view, a graph like Figure 2.1 should be rendered, whereas the business entity data model leads to a graph like Figure 2.2. These two graphs can be generated using D3[1] and source codes for these two graphs can be found in the attached treeView.html and test.html. In the business entity data model view, users should be able to edit the nodes, i.e., they can delete and rename a business entity, and they can also change the relationships between two entities. There are four relationships (exclusive containment, inclusive containment, association, and specialisation), and they should be represented using different types of lines. When users click one business entity, the system should allow them to select a detailed business entity data model or a behavioural model of the entity. For the former, the system should bring them to a detailed view as shown in Figure 2.3. The view also allows users to edit so that they can change the name, attributes, and relationships. For the latter, users can view the protocols of creating, updating, deleting, and reading a business entity like graph in Figure 2.4. There should an system configuration page that allows users to configure an ontology repository and the details are to be designed.

[1] `https://github.com/mbostock/d3/wiki/Gallery`

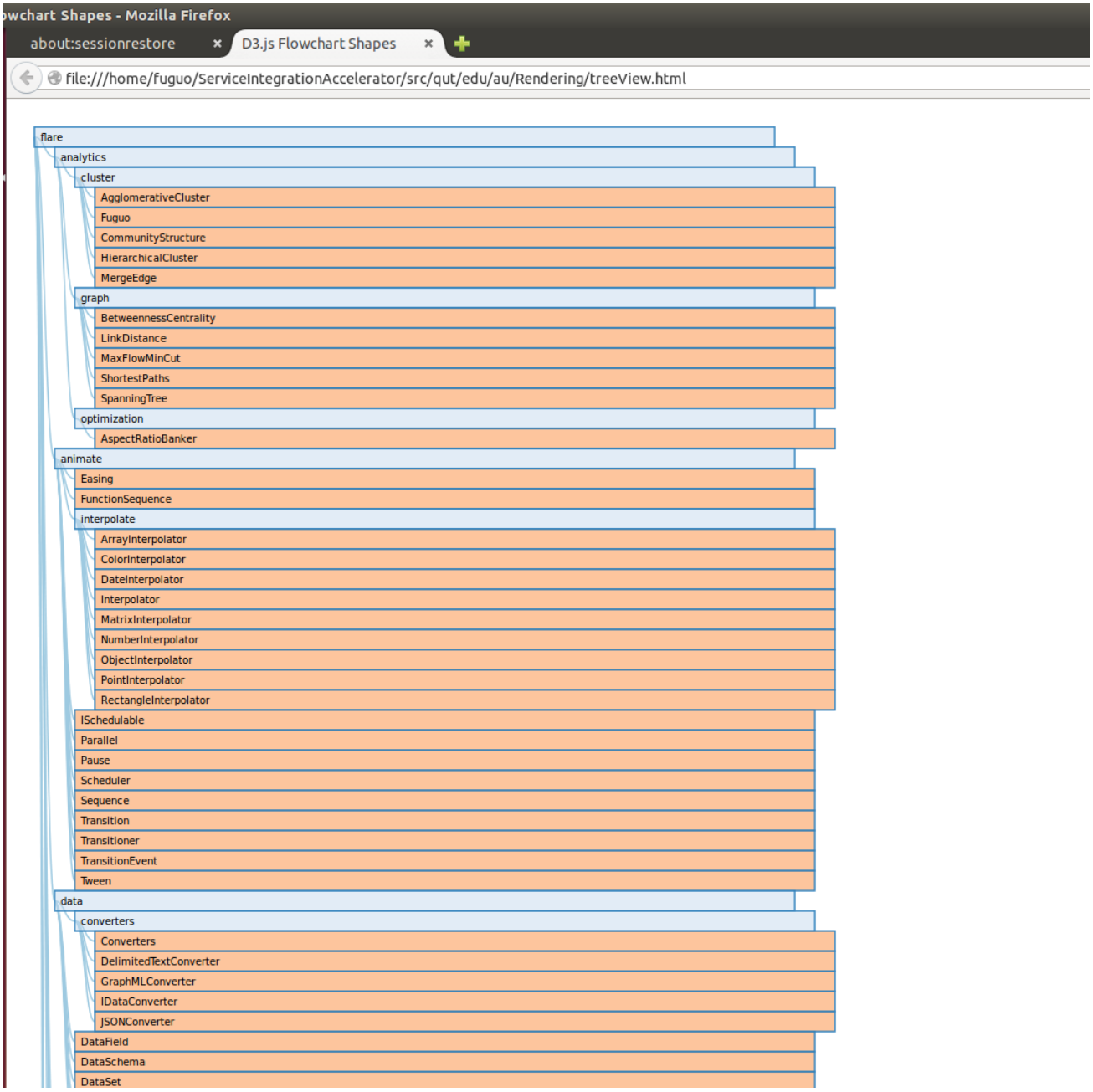

Figure 2.1: Tree view structural interface.

Figure 2.2: BE data model.

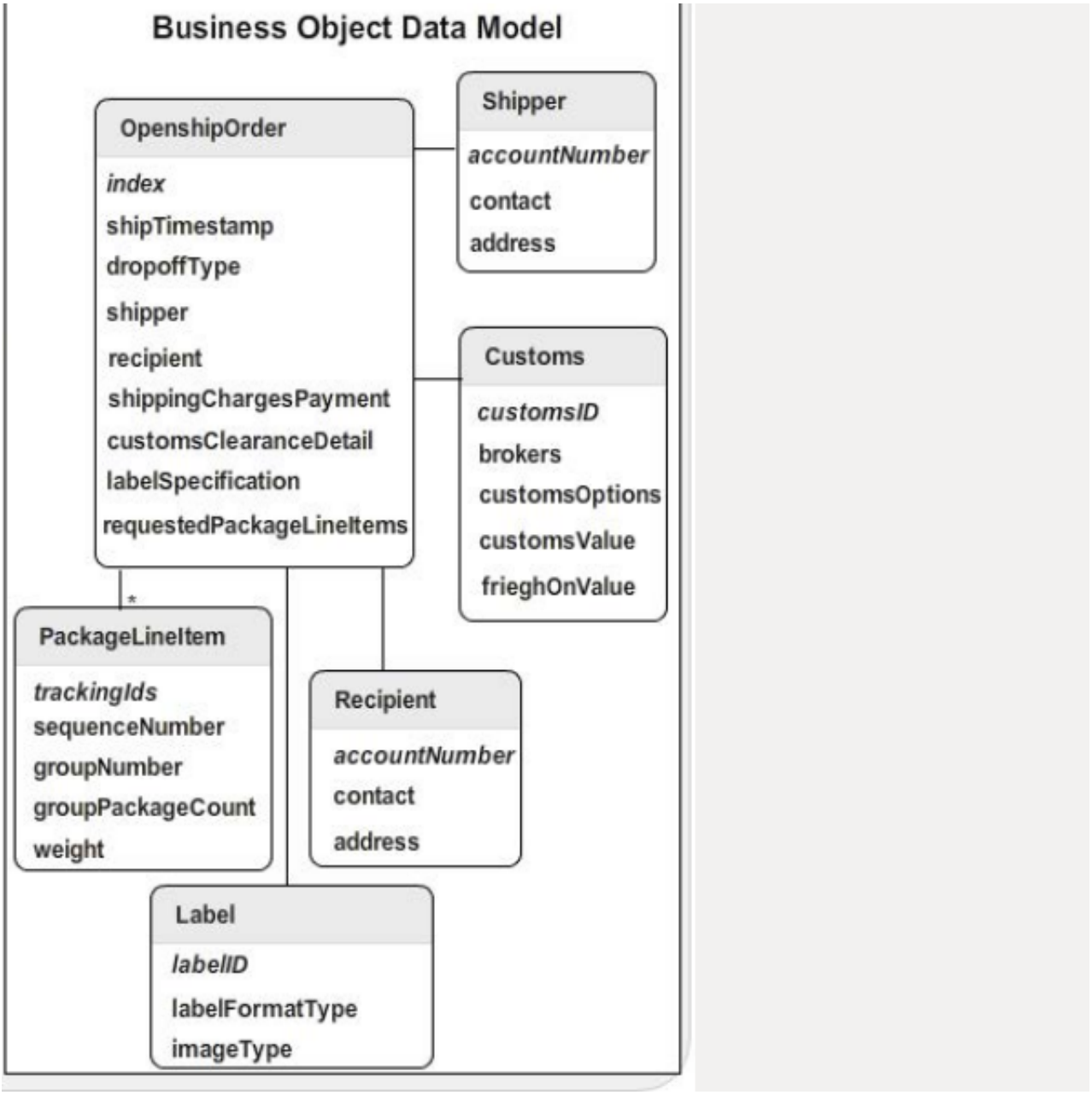

Figure 2.3: Detailed BE model.

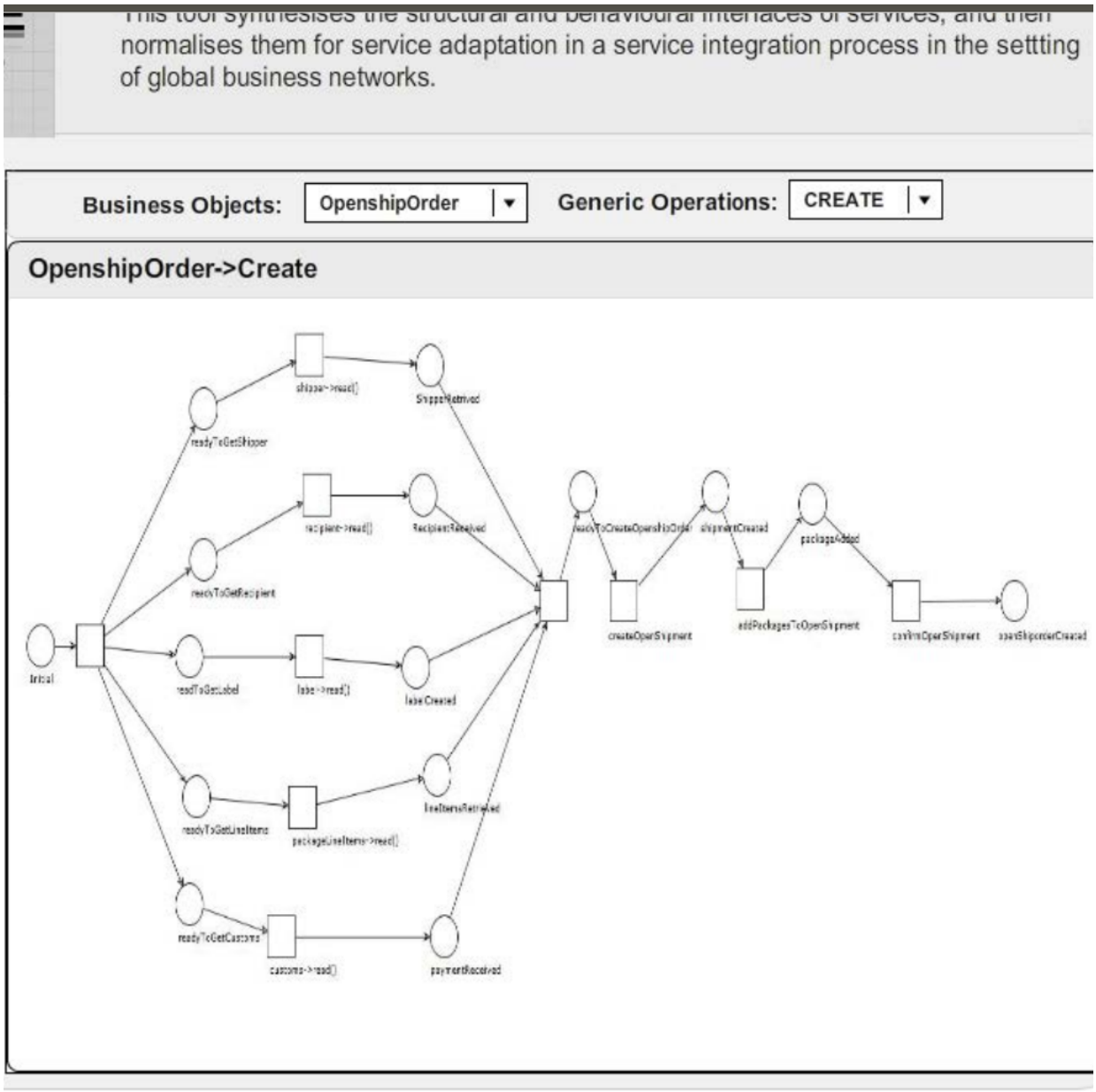

Figure 2.4: Detailed BE model.