WSDL Workbench

Douglas Martin • Scott Grant • James R. Cordy

Alex Lau • Joanna Ng • Leho Nigul

The sparse syntax of Web Service Description Language (WSDL) makes it difficult to understand repositories of service descriptions. In this work, we propose a tool to explore and visualize these repositories using techniques derived from software design recovery to automatically analyze WSDL descriptions and construct a database for querying and exploring relationships found from this analysis.

Clone Detection

In our attempt to use clone detection to discover similar web service operations, we observed that WSDL operations consist of several pieces that are located in different parts of the description. This posed a challenge because we needed a set of continuous sequences of lines to give to the clone detector for comparison.

Using TXL (a source transformation language), we built an extractor that contextualizes each operation by extracting its associated pieces and consolidating them into a structured whole we call a *WSCell* (Web Service Cell).

An example of a set of WSCells can be seen in **Figure 1**.

We found that using WSCells was more effective for finding clones than only the `<operation>` elements.

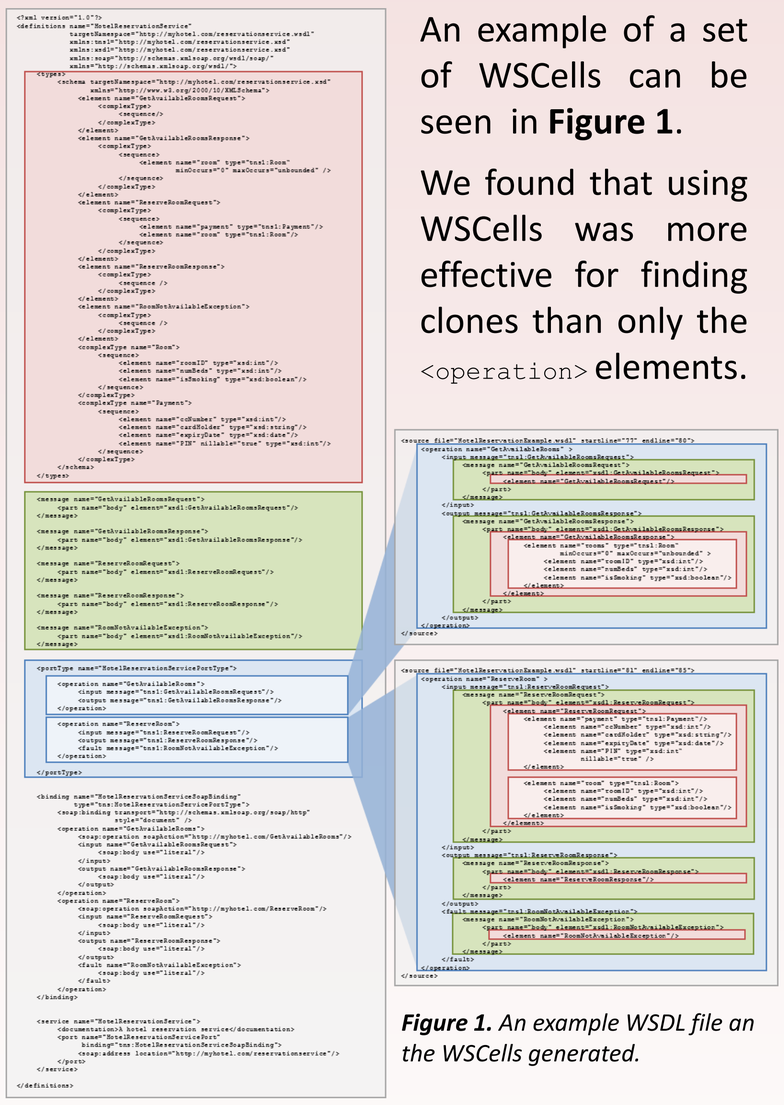

Figure 1. *An example WSDL file an the WSCells generated.*

If we extract and contextualize inputs and outputs separately, we can use clone detection to identify operations that could potentially be composed. If the output of one operation is a clone of the input of another, the output of the first operation may be able to serve as the input of the second (**Figure 2**).

Figure 2. *The output of one WSDL operation matches the input of another.*

Concept Location (LDA)

While clone detection is good at finding structural similarities of WSDL operations, it is not good at finding semantic relationships. For this we used *Latent Dirichlet allocation* (LDA) – a concept location technique. The problem we faced with clone detection was still present here because there is simply not enough information in each `<operation>` element to make any meaningful connections. However, the added context that WSCells bring to WSDL operations solves this problem. We used a tool called *POCO* (Pairwise Observation of Concepts) to see local similarities (**Figure 3**) and a visualization called *Bluevis* to see the global structure (**Figure 4**) of the repository.

Figure 3. *A list of related operations (expressed as WSCells) found using LDA and a tool called POCO.*

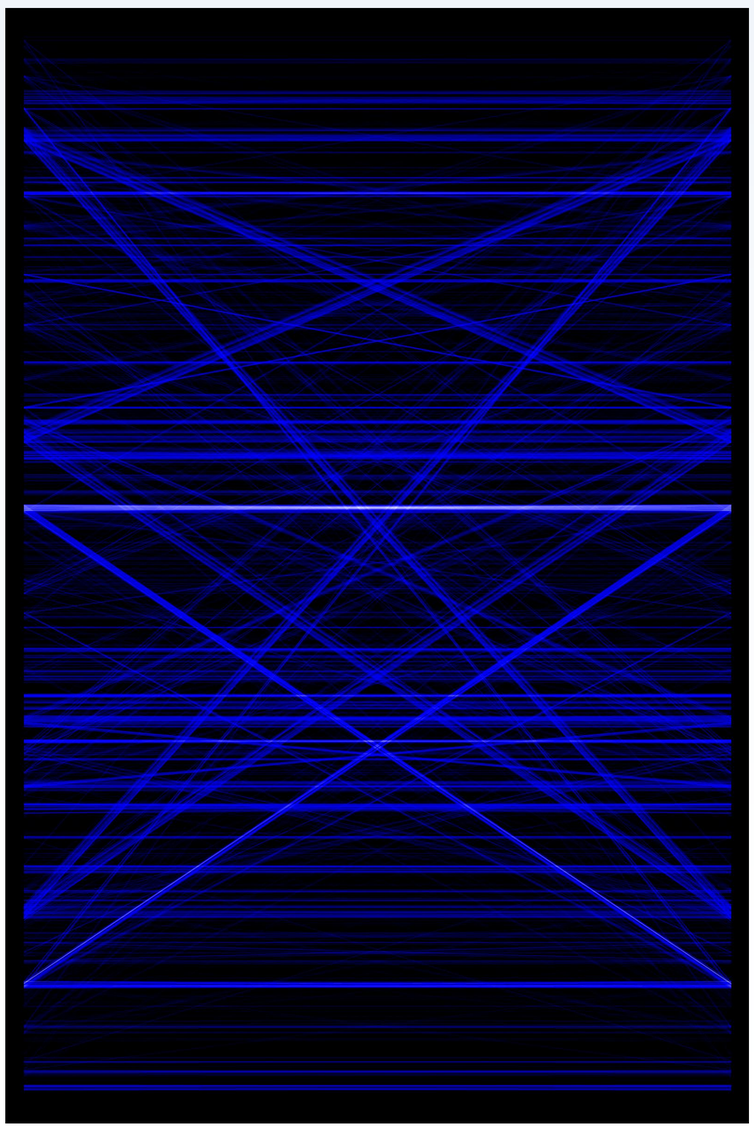

Figure 4. *A visualization called Bluevis. The blue lines connect operations with shared topics. The deeper the blue the stronger the connection.*

WSDL Workbench

Our next objective is to put all of this (and more) together into a single web-based interface to give an overview of a service repository and provide tools to help developers migrate their RPC-based services to a RESTful architecture. This includes parsing WSDL descriptions and storing them in a database along with the results of these various software design recovery algorithms that can be easily queried and explored.

Currently planned features:

- text-based and similarity searches
- filtering based on common names, types, service providers, etc.
- visualizations like Bluevis and word clouds
- smart resource and HTTP method suggestions (for migrating to REST)

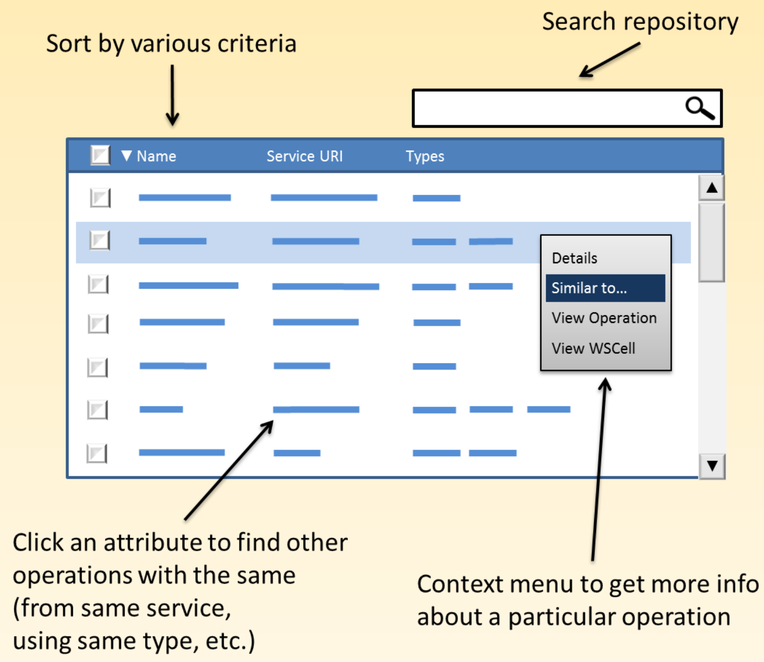

Figure 5. *A mockup of the proposed WSDL Workbench.*

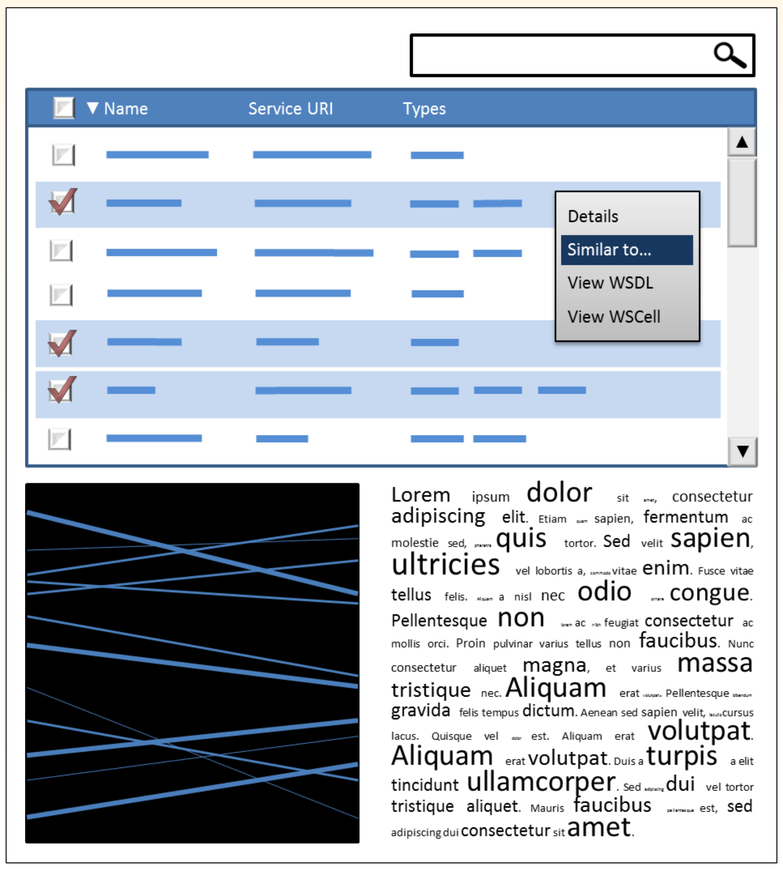

Figure 6. *A mockup of the WSDL Workbench showing visualizations.*