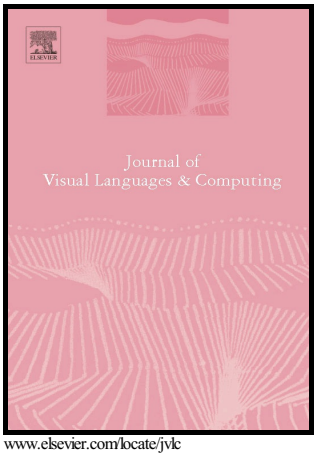
# Gneiss:Spreadsheet Programming Using Structured Web Service Data

Kerry Shih-Ping Chang, Brad A. Myers




www.elsevier.com/locate/jvlc

PII: S1045-926X(16)30099-4
DOI: http://dx.doi.org/10.1016/j.jvlc.2016.07.004
Reference: YJVLC752

To appear in: *Journal of Visual Language and Computing*

Received date: 4 September 2015
Revised date: 3 February 2016
Accepted date: 6 July 2016

Cite this article as: Kerry Shih-Ping Chang and Brad A. Myers Gneiss:Spreadsheet Programming Using Structured Web Service Data, *Journa of Visual Language and Computing,* http://dx.doi.org/10.1016/j.jvlc.2016.07.00

# Gneiss: Spreadsheet Programming Using Structured Web Service Data

Kerry Shih-Ping Chang, Brad A. Myers

Human-Computer Interaction Institute Carnegie Mellon University Pittsburgh, PA, USA

kerrychang@cs.cmu.edu
bam@cs.cmu.edu



**Web services offer a more reliable and efficient way to access online data than scraping web pages. However, interacting with web services to retrieve data often requires people to write a lot of code. Moreover, many web services return data in complex hierarchical structures that make it difficult for people to perform any further data manipulation. We developed Gneiss, a tool that extends the familiar spreadsheet metaphor to support using structured web service data. Gneiss lets users retrieve or stream arbitrary JSON data returned from web services to a spreadsheet using interaction techniques without writing any code. It introduces a novel visualization that represents hierarchies in data using nested spreadsheet cells and allows users to easily reshape and regroup the extracted structured data. Data flow is two-way between the spreadsheet and the web services, enabling people to easily make a new web service call and retrieve new data by modifying spreadsheet cells. We report results form a user study that showed that Gneiss helped spreadsheet users use and analyze structured data more efficiently than Excel and even outperform professional programmers writing code. We further use a set of examples to demonstrate our tool's ability to create reusable data extraction and manipulation programs that work with complex web service data.**

**Keywords. spreadsheet; mashup; end-user programming**

## I. INTRODUCTION

The Internet is full of data. While many data are presented in the form of web pages, more and more data sources now provide web services that allow data to be retrieved using web APIs. Compared with scraping web pages, web services offer a more efficient and reliable way to access online data, as they do not rely on parsing web page layouts that might change frequently. Many data sources also support more sophisticated searches, and return more detailed data in their web services than on their websites. As RESTful web services have now become the mainstream[1], web APIs are often accessed through URLs that are designed to be human-readable [1] and can be directly used in a web browser. The availability and variety of web services make them good resources for people who want to collect online data and perform custom data manipulations such as combining or comparing data.

However, working with web service data remains difficult for many people. The majority of modern web services return structured data, such as JSON and XML documents. The documents often contain large and complex hierarchies and are difficult for people to use without writing conventional code. Moreover, while it is common for people to collect data from multiple sources in daily tasks, working with data from multiple web services requires stitching together multiple web APIs, extracting desired parts from the returned data, and then performing further data operations. This usually requires people to write a significant amount of surprisingly intricate code that deals with asynchronous network calls which may fail to return, often requiring complex and sometimes nested call-backs [2]. Research has found that the majority of web API users are tech-savvy developers, and even they encounter many programing difficulties when trying to integrate multiple web services [3]. Data flow language tools such as Yahoo Pipes [4] and Microsoft PopFly [5] can let people wire up web services without writing conventional code, but studies have found that the dataflow concept is often difficult for people to comprehend [6], [7], and these tools have not been particularly popular or successful with end-users.

[1] According to ProgramambleWeb (http://programmableweb.com/), there are over thirteen thousand web APIs on the Internet, and 70% of them are REST APIs.