We live in a world of digital data, but to use the data often requires writing complex code. My work combines human-computer interaction, end-user programming and software engineering research to create tools that make data and databases more useful and usable.

Spreadsheet Programming for Using Online Data

The Internet has all kinds of data, from public data such as sport stats, stock prices and social network feeds, to private data such as personal calendars, location check-ins and business sales records. Many data sources provide web services to let people access their data for custom uses, such as building custom applications or performing data analysis. However, to use data from web services currently requires significant programming efforts. Even to acquire the desired data from web services involves writing lengthy code to deal with network requests and parse the returned documents. To further analyze or create applications and visualizations with the retrieved data requires more programming to manipulate the data and create views and visualizations that are interactive. Research shows that even professional programmers find those tasks challenging, in part because they require learning new programming languages and libraries [1]. For end users who need to work with data or to create interactive presentations of data, the programming barriers are often too difficult to overcome, so they need professional programmers to help, which can be a costly and time-consuming process [2].

My dissertation investigates a new way to create applications that use web services and other hierarchical data using the familiar spreadsheet languages and novel interaction techniques without having to write conventional code. Spreadsheets have been shown to be popular among users who are at all programming levels [3]. By leveraging the spreadsheet model, my research will benefit a wide range of users, from empowering spreadsheet users to use web service data and create database-like applications, to helping professional programmers work more efficiently with data analysis and visualization tasks.

Gneiss

I created a tool called Gneiss (http://www.cs.cmu.edu/~shihping/gneiss.html). Gneiss incorporates multiple innovations to the conventional spreadsheet model in spreadsheet languages, interaction techniques and spreadsheet user interfaces to support new programming activities, including:

• Programming two-way data flows between a spreadsheet and multiple, dynamically chosen REST web services [4].
• Creating interactive web applications that can dynamically consume and modify spreadsheet data, which can come from local or remote data sources [5].
• Exploring and analyzing hierarchical data such as reshaping, regrouping and joining data from multiple hierarchical documents and calculating summaries [6]. and
• Using streaming data and programming live analysis and visualizations [7].
Exchanging data with web services

I designed a “source pane” (Figure 1 at the left) in Gneiss where users can enter a web API in the URL bar to load JSON data from a web service or a local file. Users can extract any desired fields from the returned data to the spreadsheet by selecting each of them in the source pane and dragging it to a spreadsheet column. Using the user’s selection as an example, the system automatically collects other similar fields to fill the rest of that column. Users can then specify sorting and filtering rules in the spreadsheet to further refine the collected data. Those interaction techniques replace writing query languages (such as SQL or XPath) to get the desired data.

Sending data to web services leverages the spreadsheet language and constraint evaluation model. The user can send data in any spreadsheet cell to a web service by replacing any part in a web API in the source pane with the cell’s name using the syntax {{cellName}} (e.g., Figure 1 at 1). Web service data extracted to the spreadsheet are dynamic: when a web service call uses the value of a spreadsheet cell, every time the cell changes its value, the system will send a new API request to the web service using that new value, retrieve new data, and in turn update the corresponding data in the spreadsheet. Behind the scenes, Gneiss leverages the spreadsheet’s constraint evaluation model to handle different states of an asynchronous network call for the users. For example, it passes a special “Loading” value among spreadsheet cells that use data from a web service call while waiting for the call to return, which still allows computations in other cells to run normally in parallel. Once the call returns, all relevant cells automatically refresh with the latest data without needing any extra maintenance from the users.
With Gneiss, users can create reusable and interactive spreadsheet programs, as executing a new query only requires editing a cell without having to change all the computational logic that was set up previously. Users can also send data retrieved from one data source to another to create complex cascading calls that otherwise would require writing multiple nested asynchronous callbacks (see [4] for details).

Creating interactive, data-driven applications

My thesis further innovates new ways to program web applications that use backend data. One popular pattern for such applications is model-view-controller (MVC). I designed Gneiss to support programming data-driven applications in an MVC fashion but using only spreadsheet languages and interaction techniques.

In Gneiss, the spreadsheet serves as an intermediate place to hold the data (the “model”), which can be both remote data retrieved from web services and local data entered by the user. Gneiss has a right pane that is a web interface builder where the user can create web pages by drag-and-dropping UI elements (the “view”) from the sidebar, and editing properties of an element in a property sheet (see Figure 1). Each web UI element in Gneiss has several “interactive properties” that change values when the user interacts with it. For example, a checkbox has a “Checked” property that dynamically changes between “true” and “false” when the user toggles it in the web page. In Gneiss, web UI element properties can be referenced using spreadsheet language syntax and used in formulas in spreadsheet cells. This allows users to create links between the spreadsheet and the web interface and program interactive behaviors in a web application that modify spreadsheet data using a “pull-based” mechanism consistent with the spreadsheet model without the need of conventional event-based (“push-based”) programming.

For example, in Figure 1, cell A1 is set to =TextBox1!Value (similar to using the value of a cell in a spreadsheet) which will evaluate to whatever the user types in the text box in the web page. A1 is then used in a web API as the query value. Every time that the user enters a new value in the text box in the web page, it changes A1’s value and triggers a new request to be sent to the web service, which in turn updates columns B–D in the spreadsheet with the latest search results returned from the web service. In the web page, the value of the text label in the first list item is set to =B1, the name of the first search result item. The text labels in the rest of the list items are automatically populated with the rest of the data in column B. The list adjusts its length automatically based on the number of search items returned (the number of non-empty rows in column B).

With a few lines of spreadsheet code, the user sets up the data bindings between the UI and the spreadsheet (the “controller”) and creates a custom search web application. The spreadsheet’s constraint evaluation model maintains the data bindings live without the user having to write callbacks to monitor events and DOM changes. With Gneiss, the user can program many interactive behaviors commonly used in database applications using only the spreadsheet language. The interfaces can take advantage of checkboxes and sliders to sort and filter data, interactive charts and maps based on the data, and even multi-page applications where the contents in the next page are generated dynamically by what the user clicks on the current page (see [5] for details).
Using hierarchical and streaming data

As web databases and web services become more and more popular, there is more and more data in hierarchical formats such as JSON and XML. I extended spreadsheets to support using hierarchical data [6]. My thesis introduces a new method to visualize hierarchical data dynamically as nested tables using the relative hierarchical relationships between data in two adjacent spreadsheet columns. This method allows the user to flatten, reshape and regroup a hierarchical JSON document using any fields by simply moving spreadsheet columns through drag-and-drop (Figure 2). I also extended spreadsheet languages and sorting, filtering and auto-filling to support manipulating hierarchical data and create summaries of data using familiar functions. With these novel features, Gneiss becomes an environment where users can explore and analyze hierarchical data retrieved from web services or databases using the familiar spreadsheet mechanisms.

Streaming data is another type of data that has become increasingly popular. I extended spreadsheets to support streaming data [7], letting users set a column to periodically pull data from a data source. The column will grow over time with the latest data stacked on top of the older data. Temporal information of when the data arrived becomes metadata of the spreadsheet cells that can be used to manipulate spreadsheet data, such as to filter to see only data from a certain time period. Since the spreadsheet is a live environment, programming real-time analysis becomes natural because all the computations in the spreadsheet will automatically update when new values come in. Combined with my work on using spreadsheets to program data-driven applications, the user can even create web applications that show alerts based on the latest data values or that have a real-time visualization that tracks live data.

Facilitating Entering and Using Data on Mobile Devices

Another branch of my research explores ways to facilitate entering data and using databases on mobile devices. I designed Spinel [8], an architecture for Android applications that lets people add a new data source (a new web service) to an Android application by installing a plugin. Spinel includes a GUI tool that lets users create the required plugins without writing code, and an Android library that helps developers create an extendable application that can accept such plugins. For example, I made a map application where the location data sources are plugins and can be added and removed at runtime as needed. Another project, Listpad [9], innovated an unstructured, notepad-like interface for defining database schema and entering data to databases on mobile devices. The system can then use the types and structures they defined in data to provide to mobile users au-
to complete suggestions using relevant data sources, such as using Yelp’s web service to provide autocomplete suggestions when the user is entering data in a “restaurant” list, or to visualize the data using different formats, such as a timeline, a calendar or a map.

### Future Research Agenda

I look forward to continuing my research on facilitating the use of data and creating data-driven applications through designing new tools.

**Creating mobile data applications**

Programming a mobile data application could be quite different from building a desktop data application, as developers would deal with new types of input, such as gestures and voice, and new types of data sources, such as GPS and other contextual data. Programming environments popular on desktop computers, such as spreadsheets, encounter many usability issues when transferred to mobile devices including phones and tablet computers due to the smaller screen size and the lack of a physical keyboard. I am interested in the design of both mobile data applications that enable end-users to easily acquire, manipulate, explore and store data on the go and new programming environments that help developers more easily create such applications.

**Empowering end-users to work with data**

While languages such as SQL and R are powerful tools for using and analyzing data, learning those languages remains a barrier for end-users. I am also interested in continuing my research on empowering end-users to work with data. This could include conducting studies to understand the state of the art and the challenges end users face, extending familiar programming models to support new activities, and designing new ways to construct queries that are more intuitive and learnable for end-users.

**Programming streaming data**

I am also interested in designing new tools for using and publishing streaming data. This could include designing new ways to facilitate manipulating data by time, to provide a live programming environment for using live data, and to make creating real-time data analysis and visualizations easier.

### References


