

CS345 - Software Engineering

Group Project Description

Version 1.2

Spring, 2006

N.B. This is a living document, check <http://cs.earlham.edu/courses/cs345> to make sure you are looking at the current version.

1 Overview

Design, build, and test a mashup based on the Google Maps API. The domain is mapping social, demographic, and environmental data, for example:

1. Square foot of housing
2. Percent owned housing vs rented housing
3. Food/diet (possibly based on Federal lunch program data)
4. Energy consumption
5. Water consumption
6. Wind speed
7. SuperFund sites
8. Application of Federal spending
9. Application of Federal earmarks
10. Deaths due to terrorism

This is a preliminary set of metrics. Metrics have associated dimensions, for instance SuperFund sites can also have the size of the problem in acres and/or dollars. Some are raw values, some percentages, and there may be other ways of quantifying the data. Many of the elements under consideration could be represented as per capita, *i.e.* units per person or per thousand persons, over a given geographic area. They could also be represented by geographic area, and not normalized to per person. We'd like the option of doing either, and within per capita by individual or *e.g.* per thousand people. For each metric we will decide what the appropriate dimensions are.

Some of these metrics may not be available in a form which can be used, or available at all. Part of your work in this project is to find credible sources for each of the metrics, perform a certain amount of vetting on the source, and document the source as part of your meta-database so that it can be displayed through the user interface.

The interface should support viewing the data at various resolutions, for example:

1. ZIP code (U.S. only)

2. county (U.S. only)
3. state (U.S. only)
4. region (TBD, or is there a "standard"? U.S. only)
5. country
6. continent
7. world

The essence of this project is two-fold: 1) to design, build, and test an API that provides a consistent interface to a variety of data stores and is easy to integrate with the Google Maps API, and 2) to design, build, and test a web browser based user interface which supports visualizing the data and is built on your API and the Google Maps API.

All of this is to be done in the context of the software engineering practices, techniques, etc. that we are studying: XP, clarity, simplicity, generality, automation, *etc.*

Chris and Charlie are the customers for this project. We are in effect "buying" both the API and the user interface and as such will have "control" over a subset of XP's project variables at any given point.

2 Deliverables

At the high level there are five deliverables for this project:

1. An API designed to wrap a variety of data sources, *e.g.* U.S. Census data and C.D.C. data, in such a way as to make that information easy to combine with the Google Maps API.

Programmer and user documentation for the API.

Test harness (with instructions) for the API.

The API is due at the end of March. The meta-database should support at least 5 metrics (of your choosing) at this point, it does not need to include the full set until the completion of the project at the end of May.

2. A user interface based on a web browser that uses your API and the Google Maps API to visualize the data at hand.

Programmer and user documentation for the user interface.

Test harness (with instructions) for the user interface.

The user interface is due at the end of April.

3. A critical write-up describing the software engineering processes and techniques, both technical and organizational, that you employed.

4. A public presentation on your project at 10:30a on Monday May 1st.

(a) Introduction (5 minutes, charlie)

(b) Overview of the problem space and solution architectures (10 minutes, chris)

(c) Group Fu (15 minutes)

(d) Group Bar (15 minutes)

(e) Questions

5. An individual reflection about the process, the product, and your role in all of it.

The specifics for what needs to be included, addressed, *etc.* for each is contained below in the *Specifications* section.

3 Groups

For a variety of pedagogical and logistical reasons we have divided you into two groups for this project. Each of the groups will do the entire assignment.

Team Foo:

- Will
- Dave
- Skylar
- ColinK
- Shawn

Team Bar:

- Aybars
- Toby
- Alex
- ColinC
- Kevin

Each team will name themselves so that it's easy for us to refer to you. Chris and Charlie will be the clients for both groups, interchangeably. Both groups will work from a common, evolving, specification (which is this document).

Two significant problems typically encountered in the course of software engineering projects are procrastination and poor group organization. The first problem manifests itself in the form of late work, missing/reduced/un-tested functionality, and frustration. The second problem relates to group's inability to distribute tasks reasonably to members of the group. Please be mindful of these potential problems as you plan and organize your work.

We're very concious of the fact that the people in this class represent a wide range of backgrounds and experience with this material. There are many roles in a group project, it's up to each group to make sure that all members are participating fully and to the extent of their abilities.

Groups need a leader. You should choose one sooner rather than later. We're not suggesting you give them regal powers *ala* W but rather that they provide leadership in the traditional sense of the word. Being strong technically does not necessarily equate to being a good leader, many good leaders are not true geeks. That said, some of the best leaders I've worked with combine the two skill sets.

We are more than happy to give advice in these areas, just ask if you feel you need it.

4 Guidelines

This list is not yet exhaustive but it provides a good starting point for how this project should be approached.

- Use XP practices. Document how you adopted each of the principles/practices or defend why you didn't.
- Use a wiki to organize all the non-code aspects of your work, *e.g.* documentation, to do lists, developer's journals, meeting notes, *etc.*
- Use source code control.
- Intentionally choose a license for your data API and user interface.
- Use Bugzilla (on cluster.earlham.edu) for tracking bugs.
- Not all data sources will be perfect, build-in a "background" link and information for the sources used for each metric. This will show what the underlying data source is and provide a direct reference to it.
- Do a careful survey of other mashups built on the Google Maps API. What works? What doesn't? Is anyone doing something similar? Document what you find.
- Handle missing information well. Not all metrics will be available for all locations and/or resolutions.
- The total number of sources will probably be between 5 and 15, plan to accommodate a wide range of input formats and harvesting techniques.
- At least one person in each group should consider Tufte's books and be prepared to incorporate his ideas into your visualizations.
- Automate your testing. The API deliverable and final deliverable both include your test harness and instructions for using it.

5 Code

Each group is free to choose the languages and existing APIs to use for both the data API and the user interface. You should document *why* you made the choices you did in your project documentation.

You should adopt, document, and follow coding standards.

6 Specifications

This is a particularly dynamic section of this document, at least until mid-March for the API and mid-April for the user interface components. It is not the sole source of requirements for this project, consider this entire document when checking your lists.

6.1 Data API

Data sources and elements, *i.e.* metrics. The list in the introduction will be our starting point. The documentation should include a data dictionary listing all of the metrics available through the API. This information should be embedded in the API's source code and automatically extracted into the documentation. This should include all the source information, dimensions, *etc.* for each metric.

Functionality

How to add new data sources

Verification and validation. In a nutshell; validation is checking to make sure the specification accurately represents the thing being measured, verification is checking to make sure the software/hardware system is correctly implementing the specification.

6.2 User Interface

The User Interface should allow a user to quickly and intuitively view and compare the various data sources made available via your API. There should be two main elements of the interface: the control area, and the map/data display area.

The control area should allow the user to select one of the metrics available from your API for display in the map display area. Each data source listed should have links to information about its source, scope (*i.e.* for what regions it is available), last updated timestamp, and any other relevant notes. The control area should indicate what data source is being used for the current map display, if any.

The map display part of the interface should essentially be a Google map widget with your data source displayed. Depending on the nature of the data sources, you may want to use individual map points, polygonal regions, or both. Make use of colors, size and other interface cues to differentiate the data and indicate its value relative to other markers; make sure you have a legend where appropriate. Rolling over or clicking on a marker/region should show the quantitative data for that point in all the dimensions it is available. The display maps one metric at a time, overlays are not required.

Beyond the control and map display information, the user interface should have clear and concise user interface instructions, with "help" details available for any components that might not be completely intuitive. Assume that your user is unfamiliar with other kinds of map display features, and is only initiated at a basic level to the capabilities of your system (*i.e.* they know that it does something with displaying data on a map, but they don't know what or how).

The User Interface should work on a variety of modern web browsers; certainly Mozilla Firefox 1.5, Internet Explorer 6.0 and Netscape 7.0. It should be W3C standards compliant for XHTML 1.0 Transitional.

Your user interface delivery should accompany documentation that instructs other programmers on how to incorporate new metrics into the interface (assuming they are retrieved using your existing API). It should include an automated test suite that verifies the basic functionality of the interface: that the site loads, the availability of the data sources, *etc.*

6.3 Approach and Process

Survey of similar user interfaces. Include the ones identified for the exercise due February 14th.

Which languages and APIs you used and why.

What license did you choose and why.

How did you organize yourselves? What roles did you establish? Who filled those roles?

What worked well? What didn't work well and why?

6.4 Presentation

6.5 Individual Reflection

7 Process

After a couple of longer sessions initially we'll continue to devote a certain amount of in-class time to sorting-out questions, specifications, etc. but will probably limit it to about 10 minutes per class meeting. Chris and Charlie will also be happy to meet with groups outside of class to refine specifications, give solicited and un-solicited advice, etc.

You should organize your work so that a significant proportion of the time you spend on this project is spent in your bullpen working with some number of your group-mates.

Chris and Charlie will maintain this document and keep a current version of it published on the course website. We will make sure that any items that are addressed in class, via email, or in person are reflected here as necessary.

8 Open Items

We'd like to maintain flexibility for a bit, both to accurately simulate reality and to be able to accommodate ideas that you folks may bring to the project. Below is a list of things that we know have to be worked-out at some point.

- What are the appropriate dimensions for each of the data elements?
- Can we control the projection type, *e.g.* Mercator or equal-area? Doubtful.
- How to do customer reporting? Customer meetings?
- The assumption is that at least some of the underlying data stores will be cached locally. How much space will we need?
- Setup auto notification of changes to this document.
- Consider Prototype, <http://prototype.conio.net/>, for working with Javascript.