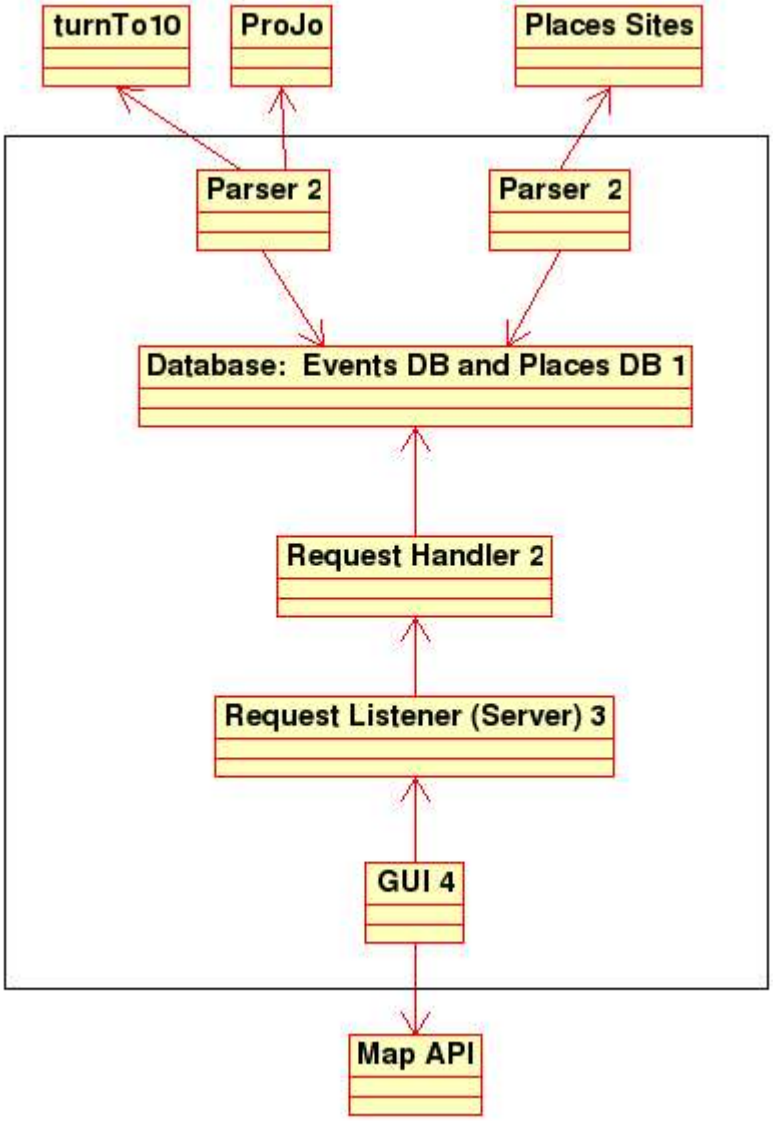
GeoEvents

Top-Level Design

Haley Allen
<haley>
2/24/2006

I. Components



A. Database

i. Events Database

The main part of the database, the Events Database will store an event's name, location, time period, description (or URL to such), and URL to where the event was parsed from.

ii. Places Database

The solution to the issue of many places being the named the same (or similar) name. Will contain places in the Providence area, their location, phone number, website (if possible), as well as any nicknames or abbreviations commonly used to name them.

B. Parser(s)

i. Events Parser

It will access the HTML for sites we determine to have useful events to parse, and parse them accordingly for useful information (as specified above).

ii. Places Parser

It will access and parse the HTML and RSS feeds from sites we determine to have accurate, relevant information regarding locations in Providence that may have events.

C. GUI

This will be the actual web site on which a user can search for local events. It will not (and cannot) be written in C++, although the language it will be written in is yet to be determined. It will include a searching interface (place, time, category, name, etc.), a map of possible events, a link to the event's website (or website from which we parsed the information), a link to directions to this location, etc. It may also include some potential add-ons including a calendar, weather information, and many others.

D. Request Listener

This will serve as the C++ interface to the GUI. It will wait for requests from the GUI and then pass them along to the Request Handler.

E. Request Handler

The Request Handler will be in charge of taking a search request from the request listener and figuring out what it needs to search for in the Database(s). It will then take that information, format it, and return it to the Request Listener which will pass it along to the GUI.

II. External Dependencies

A. TurnTo10.com and ProJo.com

There are many websites from which we could get information regarding upcoming local events. Two such examples are TurnTo10.com, a TV news site, and ProJo.com, the local newspaper site. Other such sites include providence.citysearch.com and local.yahoo.com which may

prove easier to parse then the other two. One additional feature of local.yahoo.com is that it provides an RSS feed of today's events happening around Providence.

B. Places Websites

These websites will serve as validators to the location information our map API provides. As there are many places in Providence with the same or similar names, these sites will provide information regarding their potential relevance to our searches, that is, how likely they are to host an event. For example, Dunkin Donuts and Dunkin Donuts Center are similar in name and yet it's far more likely for there to be an event in the latter than in the former.

C. Map API

From this, our GUI will be able to display maps of locations and their events, as well as provide a tagging system on said map and a search engine. Although this external dependency is a risk, relative to the other two (which are pretty huge risks), this seems to be one that is managable.

III. Task Breakdown and Group Organization

A. Project Manager – Haley

The Project Manager will be responsible for the project. In addition, she will keep track of the schedule and where everyone is on that schedule. She will arrange and manage regular meetings and provide assistance where there is trouble.

B. Coder(s)

i. **Database – Yotsawan**
ii. **Parser(s) – Andy, Toby**
iii. **Request Handler – Kaveh**
iv. **Request Listener – Sam**
v. **GUI – Peter**

The coders will be in charge of their specific components: in their design, adherence to the interfaces, and their completion. They will also be in charge of the component testing on their code.

C. Architect – Sam

He will be in charge of having a good overall picture of the design, each component, and how they all fit together. If coders need implementation-specific feedback, he/she should see Sam.

D. Tools Guru – Haley

She will setup and maintain the tools the group will need. If any of the other team members have questions regarding the usage of certain tools, she will provide assistance.

E. Documentation Czar – Amy

She will keep track of all paperwork during the course of the semester and will also be in charge of taking minutes at each of the meetings as well as distributing them afterwards. She will also be in charge of maintaining a basic web page will relevant information to the group. Finally, she will be in charge of the README document that gets attached to the final product for general consumption.

F. Integration Manager – Amy

She will be in charge of the schedule for the integration of various components in addition to actually managing these integrations. She will also lead the integration testing.

G. Tester – Toby

He will be in charge of testing, and delegating testing, from the user's point of view.

H. Cheerleader – Peter

He will be in charge of resolving conflicts and acting as a support system for anyone who is feeling discouraged. He will be generally friendly and approachable.

IV.Schedule

3/10: Final design document complete. Begin interfaces.
3/24: Interfaces and updated final design document complete. Begin coding and component testing.
4/7: Basic functionality complete, detailed designs complete. Continue coding and component testing.
4/14: Initial codebase complete. Begin integration and integration testing.
4/19: Initial integration complete. Continue integration testing and begin user testing.
4/28: Premlim demo ready. Continue debugging and user testing.
5/19: Final demo ready and all documentation complete. Continue user testing.

V. Assumptions

My main assumption was regarding Andy's component diagram, the one from which I based my design. His diagram included most of the components I think are necessary but connected them according to data flow as opposed to actual calls being made. As a result, I had to make some assumptions about which way the calls would be directed from his (sometimes) two-way arrows. I also made the assumption that there would need to be some C++ interface (here, the Request Listener) that listened for search events in the GUI and then would pass them along to the Request Handler he initially specified.