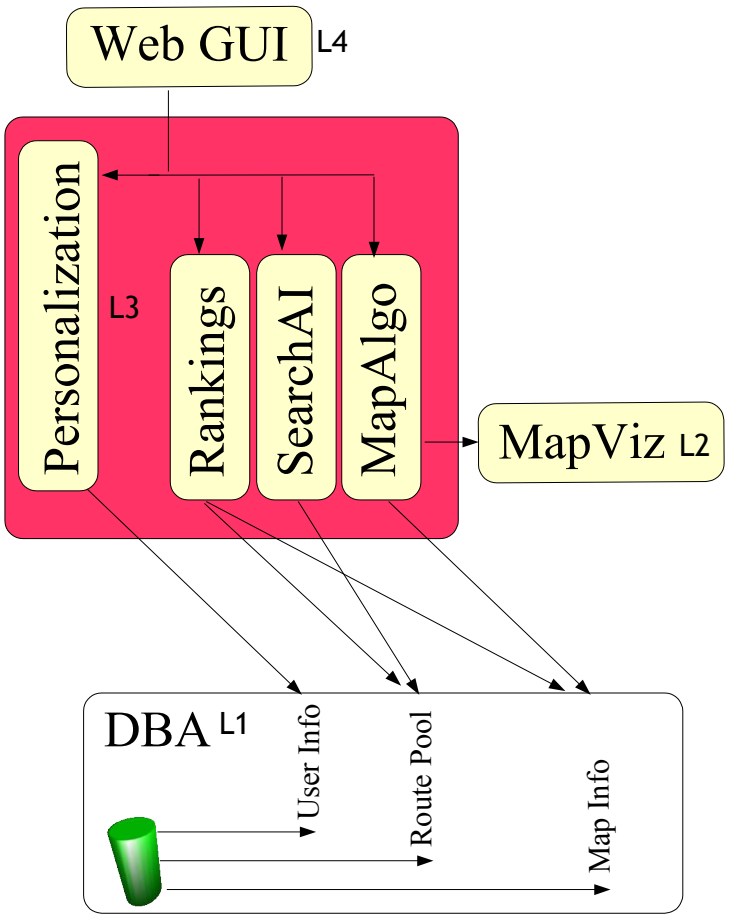
Top-Level Design Document

Redha Elminyawi <relminya>
CS190: Software System Design
2/24/2005

High-Level Component Diagram



Component Diagram Explained

Map Database (DBA) Level One

Data-store and interfaces to it. It's information effects the Mapping and Search algorithms. Receives user rating information. After GIS data is parsed into our proprietary format it will drive the Map Algorithm. DBA is also responsible for user tables, and a pool of stored routes that can very quickly be searched.

Map Visualization Level Two

Takes raw points from mapping algorithm, in combination with our map database, returns a picture of the route desired, with text description. Which type of visualization is a detailed design question.

Map Algorithm Level Three

Takes in search terms from UI, and searches the database for the correct route. Then sends the relevant information to MapVis. This will likely be some sort of A* search.

Ranking Level Three

Takes in user ratings data, and persists it to the database. Secondly, it has a loose connection to the Map Algorithm, in that it could effect route data if we change our Map data based on user feedback. Loosely coupled to Search algorithm, in that the search algorithm depends on the data persisted by UF. This is almost a user interface essentially. Since we are ranking route sections separately, this has become a user interface issue, and combining the UI and Ranking roles seems prudent.

Route Search Level Three

Will return search results to UI based on information input by the user. Also will receive information from BikeQuest database, and personalization package. Search rank algorithm will change based on personalization.

Personalization Level Three

This package receives and sends information from the front end. This information is detailed in the features section. It also sends some information to the search package, to alter search based on user. Since the myRoutes feature will be implemented, it will persist data to the map DB, in user tables.

Web UI Level Four

View for whole software, takes in all info from users and sends commands to process data to Mapping and Search algorithms. It receives data mainly from MapVis and Search. This will be written in PHP or a mix of HTML and CGI scripting.

External Dependencies

BikeQuest depends on user input. We will have to recruit a base of users to help test, and put ranking info into the system.

BikeQuest will have to have a system administrator. Physical road changes will necessitate changing GPS database information. The search algorithm will also need to be changed if it is gamed by users. Since creating an administration interface may seriously put the team in the hole time-wise, the administrator will need to know a lot about the system (through the documentation) and also about C++ and SQL. An excellent candidate for this would be Mark Dietrich.

Students will have to learn GPS format RIGIS and convert it into our own database, which may be tedious and time consuming. Besides obtaining this data (GPS), dependencies include finding web server space, database, code repository and bug tracking software. These four items are readily found within the CS department.

Task Breakdown/Group Organization

Group Manager (1) This is what Brooks terms the Surgeon's role in Mythical Man Month. During the course of the project the Group Manager will ensure that each component is on schedule, organize meetings, and monitor the overall process of creating BikeQuest. It is the Group Manager's responsibility to clear all road blocks the developers face, to maximize productivity.

Recommendations: dkarr, vywu

Architect (1) I am proposing splitting Brooks' Copilot role into two. The first, the CS190 Architect, will be have the final say on design changes to the logic components in BikeQuest. These logic components include DB, Search and Map Algorithms, Personalization and Ranking (PR). Architect will also help the DBA and Algorithms coder.

Recommendations: dspinosa, wcabral

UI Architect (1) The second copilot, the CS190 UI Architect, will be have the final say on design changes to the interface components in BikeQuest. The responsibilities will include user issues from ensuring the format results are printed out on is bike friendly to helping users test. The UI Architect will sign off all design changes to the web GUI, and Map Visualization components. As with the Architect, the UI Architect will aid in coding.

Recommendations: dspinosa, htse, relminya

Development Team (3)

DBA / Tools The database programmer will also take on the Brooks' Toolsmith role. In addition to creating readily usable interfaces to the database, this role includes setting up all tools such as a web server, database software, code repository, and bug reporting software. This role will prospectively be done early, so this coder may be recruited in helping the Algorithms coder.

Recommendations: crschmid, htse, relminya, vywu

UI / PR Will work closely with the UI Architect to ensure that the web GUI, map visualization, and personalization and ranking components reach completion.

Recommendations: aavila, crschmid, dkarr, htse, relminya,

Algorithms Will work to complete the most important BikeQuest code. This will include the search and map algorithms.

Recommendations: dkarr, dspinosa, wcabral

Technical Test (1): This person will write test scripts for all checked in code, report the bugs, and in some cases fix them. This includes all code separately, and a final integration test.

Recommendations: aavila, htse, relminya

Documentation / User Test (1): This person will maintain all documentation for the project. In addition to this, they will coordinate external users to test the program and report the bugs from these test sessions.

Recommendations: aavila, dkarr, vywu

Schedule

Friday March 4th: Suggested Roles

Hand in three preferences for roles within BikeQuest software engineering team.

Friday March 11th: Final Design Documents

Revised top level design document which includes finalized specifications and requirements, testing strategy, schedules, roles, and high level component diagrams. I propose that this document also include E-R diagrams for the database. Toolsmith should begin setting up aforementioned tools. Group Manager and Architect should begin investigating GPS data source.

Wednesday March 16th: Interface Proposals

Each component should hand in .H files that correspond to the design. Tools should be set up, and the GPS data source should be nailed down, allowing the toolsmith to start working on conversion into our database, based on agreed upon design.

Monday March 21st: Interface Comments

Each component should hand in comments on interface proposals that their component is dependent on.

Friday March 25th: Finalized Interfaces

Interfaces are changed based on comments and finalized. Discussion begins on low level design.

Friday April 8th: Detailed Designs

Design of each component is due. Any design change proposals hereafter will be met with an extremely concerning eye (namely both Architects). This begins the intensive coding session, lasting until the 18th. Each component is responsible for testing code before checking it in, reducing the amount of bugs that could get by the technical tester during this period.

Monday April 18th: Initial Integration

All modules should be complete at this time, and ready to integrate. Technical tester will first run tests on full system, and when deemed ready, the user tester will bring

Friday April 29th: Full Integration / In Class Demos

Integration, integration tests, and user tests should be nearly done here. Minor bugs should be all that exist in the in-class demo product.

Wednesday May 4th: Public Demos

Show nearly finished product to the world.

Friday May 16th: Final Demos & Hand-Ins

All bugs are gone, and all documentation is done at this stage. BikeQuest will have all proposed requirements (from Final Design document) done.

Spec Clarifications

My specifications document is at http://www.cs.brown.edu/courses/cs190/2005/asgns/2-11/relminya.pdf . There are three major clarifications needed to the preceding document.

1) How to rank roads

When the initial idea of ranking a route as a whole fell under scrutiny it proved to be a useless idea. Suppose a route is made up of 20 intersection to intersection segments. If there is heavy traffic on only one segment, then ranking the whole route as a 1 for traffic will negatively affect 19 possibly non-congested road sections. Rankings will have to be done segment by segment, and a clever user interface concept will have to be created to make ranking road segments easy, and thus make ranking something a user would desire to do.

2) Personal database

When I proposed the initial user click through, one of my reviewers pointed out that if a user wanted to return to rank a route once he biked it, he would have to enter fake ranking numbers and then add it to the BikeQuest-wide pool of routes. A compromise between this is to allow personal space for a few routes, which a user may return to, rank the route, and then add it to the database.

3) Advanced search page

The advanced search page that I propose will simply add the following search quantifiers to the basic search page: Traffic, Scenery, Road Condition, Hazard Conditions, and Gradation. Each of these will allow the user to further qualify their search terms on a 1-5 scale. For example, if a user enters a 5 for scenery, the only routes that will be returned will be routes whose sections average a score of 5 in scenery.