

@make(article)
@device(postscript)
@heading(15-413: Project Description)
@heading(Interactive Maps)

@center(2 January 1991)

@section(Problem)

An interactive map allows a user to navigate through visual and textual representations of information, dynamically controlling the amount and nature of information that is displayed. For example, a topological map could be dynamically configured to show various combinations of rivers, trails, places of historical interest, emerging weather formations, previous annotations, campsites, favorite fishing holes.

This project would be a scaled-down version of this concept, applied to the School of Computer Science at Carnegie Mellon. For example, a user might navigate through a map of Wean and Doherty halls, "go into" various rooms, display varying amounts of information about the residents of particular rooms (e.g., mug shots, plan files), the availability of M&Ms, lecture halls, terminal availability, etc.

@section(Requirements Specification)

@subsection(Nonfunctional Requirements)

There are two high-level requirements for this project:

(1) Usability: The results must be usable by the campus community. There are two aspects to this. First, it should work as a stand-alone service that the University community will want to use. Second, ultimately we would like to integrate this project into the new version of the campus-wide information service managed and maintained by the library automation group at CMU. This service currently provides on-line access to library holdings, and is now being revised to take on an expanded role in providing more general information services. These services will access commercial databases and be accessible from X-based workstation environments. To support this requirement, an important byproduct of the project will be excellent documentation for users.

(2) Extensibility: The project must develop a framework for incrementally augmenting the services it provides. Again, there are two dimensions to this. First, it should be possible to incorporate new information sources into the interactive map. For example, supposing that an initial version supports accessing directory information and plan files for residents of Wean Hall, it should be possible to latter "plug" in a database of information about current events taking place in the various seminar rooms. Second, given that new information bases may be added later, it should be possible to extend the user interface so that the information in these added bases can be displayed appropriately. Continuing with the previous example, it may be necessary to use a new kind of graphical display for the "event" information -- say a calendar. To support the requirement of extensibility, an important byproduct of this project will be a document explaining how the system can be extended by system maintainers.

Thus you can view the project has having two kinds of clients. The

first kind of client is a system end user, who navigates around Wean Hall looking up information of interest. For this client the important properties of the system are appropriateness of information and user interface convenience. The second kind of client is a system maintainer, who has to extend the product by adding new data sources into the existing system. For this client the important properties of the system are the ease with which it is possible to add new information, without significantly modifying what is already there.

@subsection(Functional Requirements)

To be more concrete, the product should support access to three basic kinds of information: geographical, object, and temporal. Geographical information represents the map data, such as building floor plans, room location, campus building layout, etc. Object information describes the properties of entities in the system: contents of rooms, information about room occupants, etc. Temporal information represents event data, such as university holidays, class schedules, lecture schedules, etc.

For each of these three kinds of data we can distinguish between CORE and OPTIONAL aspects. The core aspects are those that must be included in the final product. Optional aspects are indicative of extensions that can be included, time and interest permitting.

The core requirements for the project are to be able to browse around a single floor of Wean Hall, obtaining information about each room, (such as whether it is an office or a lecture room). For rooms that are offices, it should be possible to find out who uses the office and some information about those people (title, phone number, department, etc.). For certain lecture halls, it should be possible to display a calendar of lectures for the room. In addition, it should be possible to access (more globally) schedules and information about people that work in Wean Hall. (eg., Where is Wing's office?, What is the class schedule for 15-413?)

Core requirements and optional extensions for this project are summarized in the table below:

@begin(example)

	CORE	OPTIONAL
Geographical	single floor WeH pan/zoom	multiple floors/buildings
Object	kind of room occupant info (title,phone,etc)	plan files of occupants room contents (MM,Coke,wkstations,etc)
Temporal	seminar schedule 15-413 schedule	personal calendar university events

@end(example)

As is evident from the description above access to information needs to occur in various combinations. Again, there are core and optional aspects. These are summarized below:

CORE:

@begin(itemize)

Object via Geographical: eg., access person information by clicking on

an office

Object via Global: eg., what is Bruegge's office?

Temporal via Geographical: eg., access lecture information by clicking
on a lecture hall

Temporal via Global: eg., show the schedule for this class

@end(itemize)

OPTIONAL:

@begin(itemize)

Other combinations of Global, Temporal, Object, and Geographical.

As an example of, Geographical via Global, have a map display the office
associated with Bruegge.

@end(itemize)

Finally, as indicated in the high-level requirements, a key aspect of
this project is extensibility. Specifically, the project should be
engineered in such a way that a new data source can be smoothly and
easily integrated into the existing product. To test this requirement,
you will be expected to add a new data source into your initial
version of the product, and to document the process that was required
to do it. That data source will not be known by you until you have
completed the initial version of the product.

@section(Design) @label(Overall Design)

A solution to the problem described above is complex and
requires a multi team effort. We have decomposed the problem
into four modules shown in figure @ref(Design).

@begin(figure)

@center<@graphic(width=4.375 inch, height=3.125 inch, PostScript =
"/storck/usrl/bob/se/SE91/overall_design.ps")>

@caption(Overall Design of the Interactive Maps System)

@tag(Design)

@end(figure)

@begin(itemize)

@c(User Interface): This module provides the look and feel, that is, the
command language, screen layout and error message format.

@c(Data Base Glue): This module provides an extensible query language used by
other modules
to access the multiple data bases.

@c(Maps): This module provides a map data base using geographical information.

@c(Calender): This module provides a calendar data base using temporal
information.

@end(itemize)

The full implementation of these modules consists of a core task
and an optional task:

@subheading(User Interface) @i(Core task):

@begin(itemize)

Graphics oriented user interface that displays a single floor and information

on a person/room when requested by the user.
@end(itemize)

@i(Optional tasks):
@begin(itemize)
Provide additional graphical capabilities for the @c(Maps) and @c(Calendar)
projects.
@end(itemize)

@subheading(Data Base Glue)
@i(Core task):
@begin(itemize)
Deal with global queries multiple data bases.

Provide for extensibility.
@end(itemize)

@i(Optional tasks):
@begin(itemize)
Implement the @b(third data base), a
data base to be chosen by or negotiated with the client. b@end(itemize)

@subheading(Maps)
@i(Core task):
@begin(itemize)
Display a single floor in Wean Hall and
provide information on a selected object (person/room).
@end(itemize)

@i(Optional tasks):
@begin(itemize)
Navigation through multiple floors/buildings.

Zoom in and out of objects.
@end(itemize)

@subheading(Calendar)
@i(Core task):
@begin(itemize)
Provide a facility to browse through a seminar schedule (15-413).
@end(itemize)

@i(Optional tasks):
@begin(itemize)
Provide a facility to setup a personal calendar (Appointments, Tobedone,
Reminders, Diary, Expenses).
@end(itemize)

@section(Project Plan)

Each module described in @ref(OverallDesign), also called group project in the following, is assigned to a group of people to be formed after the first lecture. The project evolves along several stages each of which results in a document produced by each group. At the end of the course the results of the four groups will be integrated into a single software system and associated documentation to be delivered to the client.

@subsection(Project Stages) @label(Documents)
The project stages are:
@begin(description)
@b(Project Plan):@A detailed schedule for individual milestones of the

module. Set up a schedule of activities and assign personnel@*
Document:
@begin(itemize)
Software Project Management Plan.
@end(itemize)

@b(Requirements):@\Describe the model of the proposed subsystem
in a data-flow diagram, the data dictionary, the activity specifications,
the structure of files, nonfunctional requirements and feasibility.
Document:
@begin(itemize)
Requirements specification document.
@end(itemize)

@b(Design):@\Describe the user interface and the software structure of the group
project.
Document:
@begin(itemize)
The Design Document consists of two parts
 @begin(itemize)
 The @i(Preliminary Design Document) describes the modules and
 data structures of the system.
 To ensure a smooth system integration,
 the data types and functions exported to the other groups
 are marked @c(PUBLIC).

 The @i(Detailed Design Document) is not really a new document.
 It expands the module headers produced during the
 preliminary design phase.
 @end(itemize)
@end(itemize)

@b(Coding and Unit Testing):@\Write the source code for each module and document it.
Provide a test bed that enables each group
to proceed independently from the progress of the other groups.*
Documents:
@begin(itemize)
Test case manual: Documents a set of test cases describing
how the group projects are to be tested.
@end(itemize)

@b(System Integration):
Integrate the individual projects into
a system and make sure it passes the client
acceptance test.*
Documents:
@begin(itemize)
 > Integration Test Manual: Describes how the overall system is
 to be tested.

 > User's Manual: Overview of the system from the end user's point of
 view, how to use the system.

 > Installation Manual: How to install the system,
 dependencies on the environment. Administrative details.
@end(itemize)
@end(description)

@section(System Integration) @label(CA)

In addition to the steps to be performed by the individual projects, two steps must be done together by all groups:

@begin(description)

@b(System Integration):@\\A set of tests to ensure correct performance of the whole system.

@b(Client acceptance test):@\\

The software system is acceptable, if it provides the core functionality described in section @ref(Overall Design) and all the documents listed in section @ref(Documents).

@end(description)

@subsection(Project Schedule)

The project schedule consists of phases and milestones. The overall project phases and milestones are as follows:

@begin(format, size = 9)

@tabclear()

@tabset(1 in, 4.5 in, 5.5 in)

Date@\\Phase@\\Milestone

Jan 17-Jan 30@\\Project planning phase.

Jan 22@\\@\\Project presentation by Sponsor.

Jan 31-Feb 14@\\Requirement phase.

Feb 12@\\@\\In-class Project meeting.

Feb 15-Mar 12@\\Design phase.

Mar 14@\\@\\@i(Project review with Sponsor).

Mar 13-Apr 4@\\Coding and Unit Testing.

Mar 21@\\@\\In-class Project meeting.

Apr 5 - Apr 30@\\System Integration.

May 2@\\@\\@i(Project acceptance by Sponsor).

@end(format)

@section(Organizational Interfaces)

The overall project managers are Bernd Bruegge, Rod Nord and Jeannette Wing.

The hardware configuration provided for the course consists of ? with ?.

@b{<<Bernd: Could we get a special Sun into the room (ask Wactlar, Stoltzfus), Get in touch with Jim Skees for the lab room.>>}

The software consists of the following components:

@begin(itemize)

IDE: Software through Pictures. A case tool for the requirements and design phases.

@b{<<Bernd: Send license and maintenance agreement to IDE,

Ask Allan Stoltzfus for purchase order (Coordinate with Nico).>>}

@b{<<Rod: Install IDE, once it is here (estimate: Jan 5 if everything goes well)>>}

Larch tools: ?

@b{<<Jeannette: Describe which ones>>}

@end(itemize)

@b{<<Bernd: Project meetings are to be held in room ?. Keys to the lab room are available from ?. Get in touch with Jim Skees for the lab room.>>}

There is an existing system that can be used as to get a feeling for the feasibility of the project. It is called CAMPUS and is available on Andrew machines.

@b{<<Rod: Please fill in the necessary information on information and how to run the CAMPUS system that you showed me and David.>>}

Several andrew bulletin boards @b(academic.cs.*) are available for @c(Interactive Maps).

@b{<<Rod: Ask gripe to install these bulletin boards. I believe there is another set of bulletin boards that gripe usually installs for each course>>}.}

@begin(description)

15-413.announce\Lecture and project announcements.

15-413.discuss@\Discussion of project related issues.

15-413.glue\Bulletin board for the Glue group.

15-413.ui\Bulletin board for the User Interface group.

15-413.maps\Bulletin board for the Maps group.

15-143.calendar\Bulletin board for the Calendar group.

@end(description)

The groups must meet weekly in the lab room. Each group has three main functions (project management, project leader, liaison) as well as three support functions (document editor, programmer and record keeper). The meeting times and internal group structure have to be decided by each group. We encourage the rotation of the project leader function among the team members, for example after the end of each project phase. Each member of the team has to be project leader at least once during the project. The other functions can be assigned on a permanent basis.