IPRO 305

The Applications of Pervasive Computing

Spring 2005 Final Report

May 6, 2005

“Things that think want to link”

- ***Nicholas Negroponte, Media Labs, MIT***

Table of Contents

List of Figures

[1] Background

[a] What is Pervasive Computing?

Pervasive computing is a broad new research field in computer science. Whereas traditional interfaces with computers have been through a keyboard and monitor, pervasive computing seeks to weave computers seamlessly into our daily lives. In a world with pervasive computing you'll be able to go shopping and automatically get more information about a product that is tailored to your preferences. You'll be able to open your notebook computer up and automatically discover network services that are present in your area. You'll be able to start complex computing jobs from your handheld device and not have to worry about where the results are going to be stored. These are just a few of the things that will come as a result of living in a truly pervasive world.

[b] Problem

Pervasive computing applications, while greatly enhance our lives, must be deployed to be useful. The problem is that with so many different devices and platforms available today, an architecture is needed that can handle this variety and be used in any environment. In addition, a real world application is needed to validate the architecture.

[c] Solution

Our solution is two fold. It includes an infrastructure and a proof-of-concept application. The HawkTour infrastructure employs primarily Web services to provide an extensible, mobile, context aware, content delivery system. It can run on laptops, tablet PCs, and in the future, on PDA's and mobile phones. The HawkTour application is a proof-of-concept prototype designed to exhibit pervasive computing in the real-world and validate the effectiveness of our infrastructure.

ARCHITECTURE

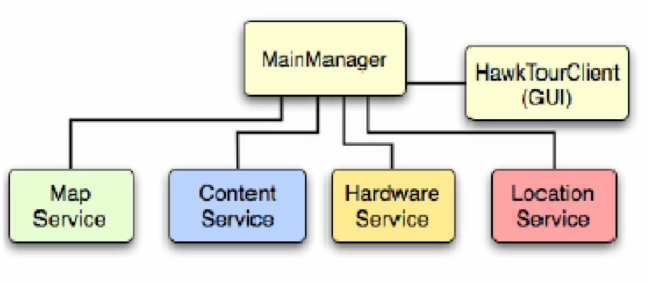


Fig.1 Architecture of HawkTour – An Application of Pervasive Computing

HawkTour is a module based, context aware, content delivery application used as an interactive student guide. The architecture has various modules enabling complete scalability and extensibility of any module without affecting any other module. For example, the range of the Map Service can be expanded from the MTCC to the entire IIT area without affecting the Hardware Service. Also, the application is entirely written in Java and hence enabling portability of the application between different platforms.

LOCATION SERVICE

The Location Service is a Web Service implemented in Java and deployed on an Apache Tomcat server. The Location Service itself serves as a public gateway to an Ekahau positioning engine which provides location data based on wireless signal strength from IEEE 802.11 access points. The Ekahau backend is accessed by the application via the exposed methods of the Java web service via the industry standard Simple Object Access Protocol (SOAP).

The Location Service gateway is multithreaded, which allows it to handle multiple client connections. The Location Service is implemented as a "request-response" or "pull" service, which means that the application desiring information from the service is responsible for connecting to it and requesting the appropriate information. It is also responsible for reporting back to the application the current map information for which the location information applies. This allows the application to provide sub-maps of greater detail in appropriate areas.

CONTENT SERVICE

The Content Service, like the Location Service, is implemented in Java and deployed as a web service on a Tomcat server running Apache SOAP, and is also a passive service. The Content Service itself serves as a public gateway to a MySQL database, much as the Location Service serves as a gateway to a backend Ekahau positioning engine. Communication between the service and the database is done via the MySQL Java Database Connectivity (JDBC) connector. Therefore, the Content Service and database can be located on totally different servers, since communication between the service and the database can be done remotely.

The database makes use of the InnoDB database type, which allows for foreign key constraints, and is searchable in a variety of different ways. The database contains URL references to content, which the Content Service returns to the application. The Content service only provides references to the content; it does not deliver the content itself. This prevents the need to marshal binary data, such as pictures and audio, into SOAP envelopes, and allows the actual content to be located in multiple locations.

HARDWARE INTERFACE

In order to make use of hardware devices that might be available to the tour-taker, the HawkTour team has introduced the Hardware Interface. This feature is currently implemented for display devices, such as televisions and computer monitors. As the user equipped with a HawkTour device approaches a large display, the HawkTour application will prompt the user to interact with the display. Possible functions that the display can provide are as follows:

- Greet guests as they enter the building
- Display large campus and building maps
- Display videos

The Hardware Interface is built in Java and C++. It utilizes a layered structure which allows Java and C++ code to communicate with each other. This is made possible through the Java Native Interface (JNI). The Bluetooth hardware device is controlled using C++, which is then wrapped in JNI and Java and made available for the Java HawkTour application to utilize.

HAWKTOUR APPLICATION

The HawkTour application is the actual application run on a tablet PC by the user taking the tour. The application consists of five major components: the Main Manager, the Location Manager, the Content Manager, the Map Manager, and the Graphical User Interface (GUI). The Location, Map, and Content Managers are responsible for all functions relating to their name. For example, the Map Manager is responsible for all map related functions, such as coordinate translation, map display, and managing the GUI map elements.

Each manager manages the GUI elements that belong to its functional area. For example, the Content Manager contains QuickTime GUI elements within itself that it passes to the GUI in order to create the GUI. The GUI itself is only responsible for displaying the interface to the user, and passing user input back to the appropriate manager. The GUI itself is not responsible for executing anything other than simple method calls on the appropriate manager.

All communication is funneled through the Main Manager, which instantiates the client and facilitates communication between all the other managers. This architecture allows for a unified communication pathway between the managers, and creates code that is more easily debugged. The HawkTour application is implemented in Java, and is responsible for retrieving data from the Content and Location Services, and displaying this data. The graphical user interface is designed using Java's Abstract Windowing Toolkit (AWT), and is currently tailored to fit Tablet PC displays. The AWT design will allow for simpler translation of the interface to smaller devices, such as PDA's or cell phones.

[d] Fall 2004 Conclusions

APPLICATION GROUP

- Further Improve the Application
  - Make Maintainable/Readable
  - Implement the Re-Design
  - Improve Stability
- Increase the Scope of HawkTour
  - More Site Surveys
  - More Hotspots
  - More Multimedia Content
- Integrate and Test the New/Improved Modules

DESIGN GROUP

- Research Data Synthesis
  - Brain Storming
  - Exploring Concepts
- Implementation
  - Prototype Interface
  - Simplified Maps
  - Panoramic Photos

HARDWARE GROUP

- Software Design
    - Requirements
    - System Design
    - Interface Design
- Implemented
    - Service Broadcasting
    - Device Discovery

[2] Objectives for Spring 2005

The Spring 2005 IPRO 305 team will be continuing the efforts of ongoing research at IIT in the realm of Pervasive Computing. The team will be focusing on the challenge of integrating Pervasive Computing into a specific aspect of the IIT campus. The overall vision of the IPRO 305 team is to continue efforts to develop an application called HawkTour – a virtual Illinois Institute of Technology tour guide. HawkTour will provide a completely new approach to the campus tour at IIT.

The application has been designed to run on Tablet PC's or other devices with similar computing capabilities, and provides the user with general campus information while guiding the user around campus and maintaining complete awareness of the user's current location and intent, thereby adapting the tour to the user's own personal preferences. The objective for this semester's team is to refine the application and come up with a new version with an optimized content database, a more sophisticated hardware interface, improve the mapping methodology, work on the user interface and scale of the application to simpler devices such as cell phones and personal digital assistants.

MAJOR DESIGN GOALS

- Understand the current location and provide surrounding campus information. This includes respective building information and history about the building.
- Provide campus information on demand and at all times.
- User-friendly interface in navigation through the campus and in the campus buildings.

- The application should be easily extensible so that new campus tour features can easily be added.
- To replicate the application development process horizontally.

[3] Team Organization

The team is divided into six sub-teams, based on a controlled de-centralized team organization approach. Each team will each focus on one of the major aspects of the project. Responsibilities are not defined on an individual level; rather, the activities are assigned to a sub-team as a whole. The sub-group is assigned a leader who is responsible for delegating tasks among the sub-group members, and ensuring assigned tasks are accomplished. As the goals for each team are accomplished and the needs of the project change, members may be reassigned to different teams, old teams may dissolve, and new teams may be formed.

- **DATABASE TEAM**

  The entire content database will be analyzed and new schemas will be developed so that the information is used to its fullest potential. Also, the application will allow customization based on a pre-tour questionnaire.

  Members of this team

  Satish Thomas, Rohit Murali, Sangmin An

- **HARDWARE INTERFACE TEAM**

  The hardware interfacing with the high definition televisions will be worked on. Significant results will include muting of the television's audio while playing hawktour's videos, automatic detection of the video display settings, streaming media, improve the interface application on the client and work on error codes.

  Members of this team

  Santhosh Meleppuram, Marcin Jastrzebski, Olumayowa Jenyo, Richard Kodamanchili

- **MAPPING TEAM**

  The mapping methodology of the system is going to be changed from rectangles to more generic polygons.

Members of this team

Robert Brozyna, David Dixon, Jodel Charles

- **MICRO TEAM**

  The application will be scaled accordingly to port it onto smaller devices such as cell phones and personal digital assistants.

  Members of this team

  Michael Sepcot, Nicu Ilea

- **USER INTERFACE TEAM**

  The user interface of the application will be revised to look more like the prototype. User surveys and user diaries will also be included in the application.

  Members of this team

  Tyler Butler

- **TESTING TEAM**

  The testing process will begin right away. Test cases and methodologies will be developed by consulting with each team and every build will be tested during the build after that.

  Members of this team

  Tyler Butler, Rohit Murali, Olumayowa Jenyo

[4] Progress Overview

MAPPING TEAM

- Implemented polygon hotspots.
- Implemented mini-map.
- Analyzed mapping sorting and other algorithms for efficiency.

DATABASE TEAM

- Came up with schema's for new database.
- Re-deployed existing DB on new server.
- Integrated schema changes into Content Service.

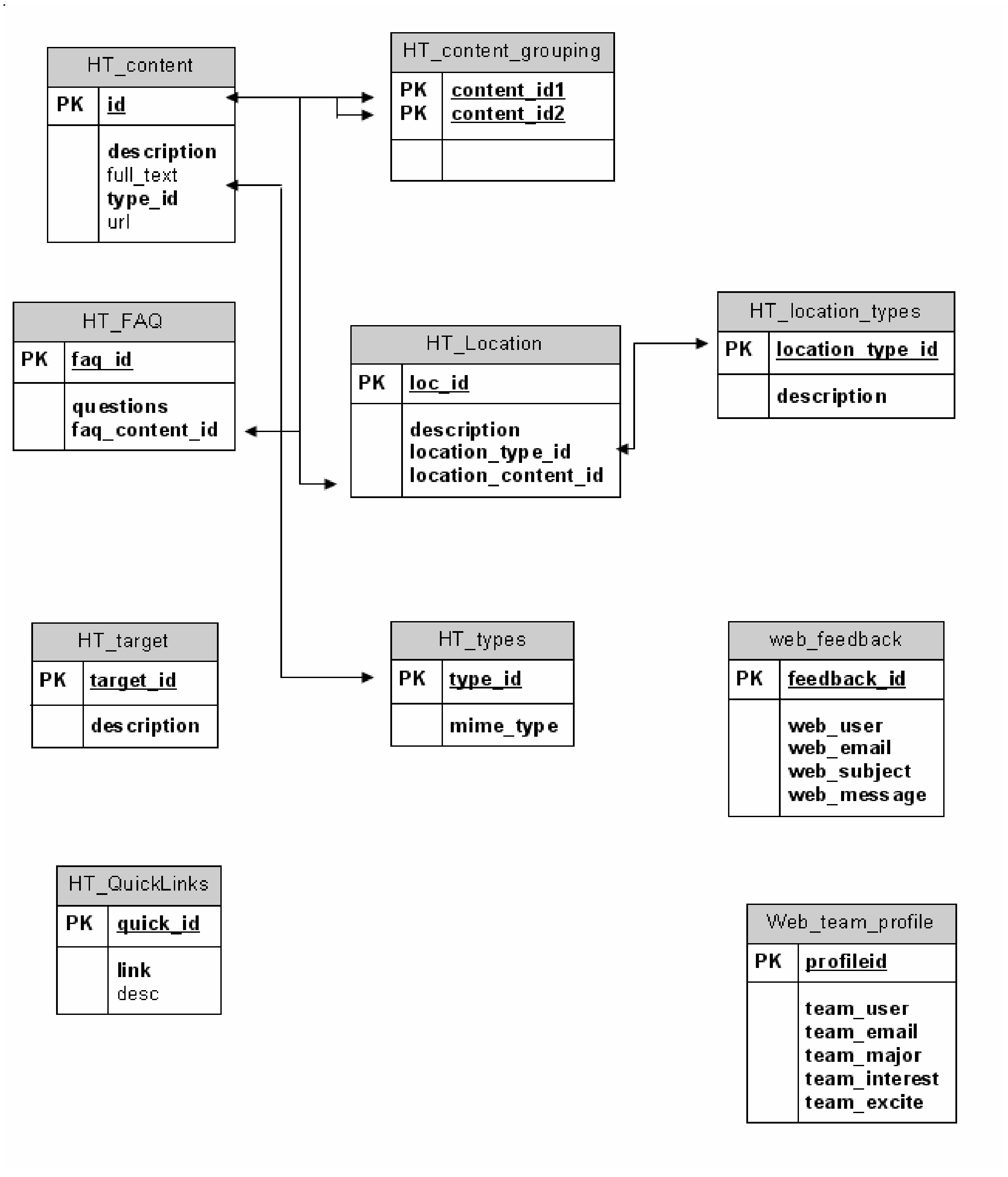


Fig.2 Table relationships of the revised HawkTour Database

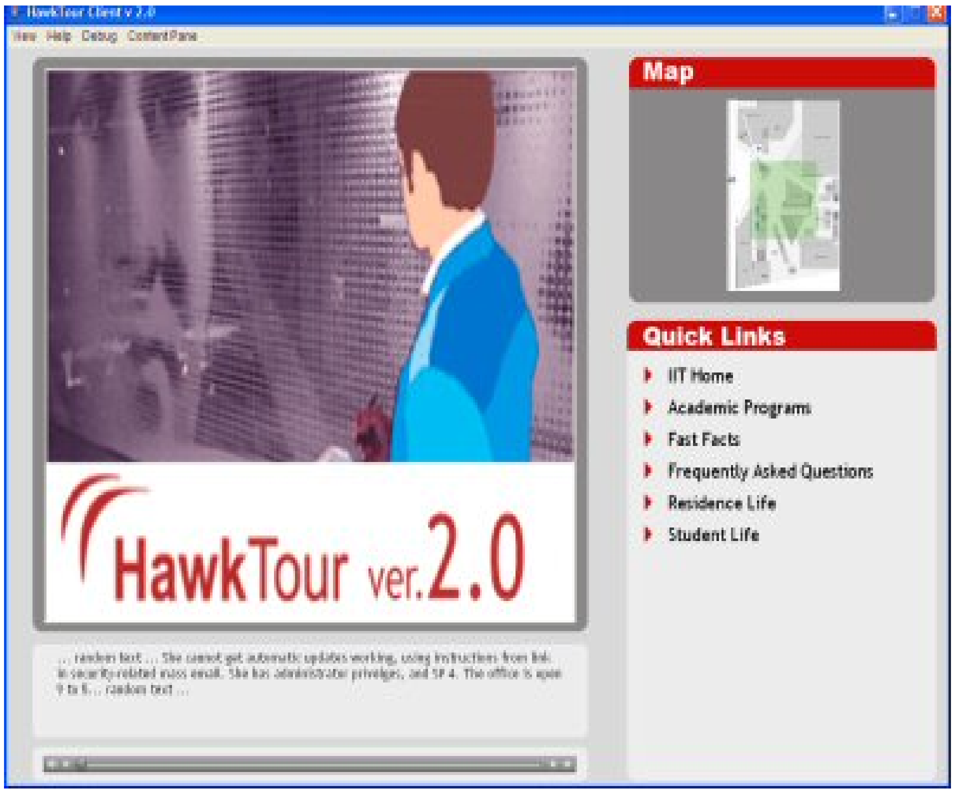
HARDWARE INTERFACE

- Developed and Integrated code for muting audio output.
- Performed dynamic window resizing for playing HawkTour videos and images.
- Built a fully functional Digital Media Receiver (DMR).
- Re-designed the HI Panel.
- Developed code to automate starting the of the TV tuner software on the DMR (Digital Media Receiver) when the DMR server application is started.
- Researched software and hardware methods to decrease the range of Bluetooth devices used by the Hardware Interface.

USER INTERFACE

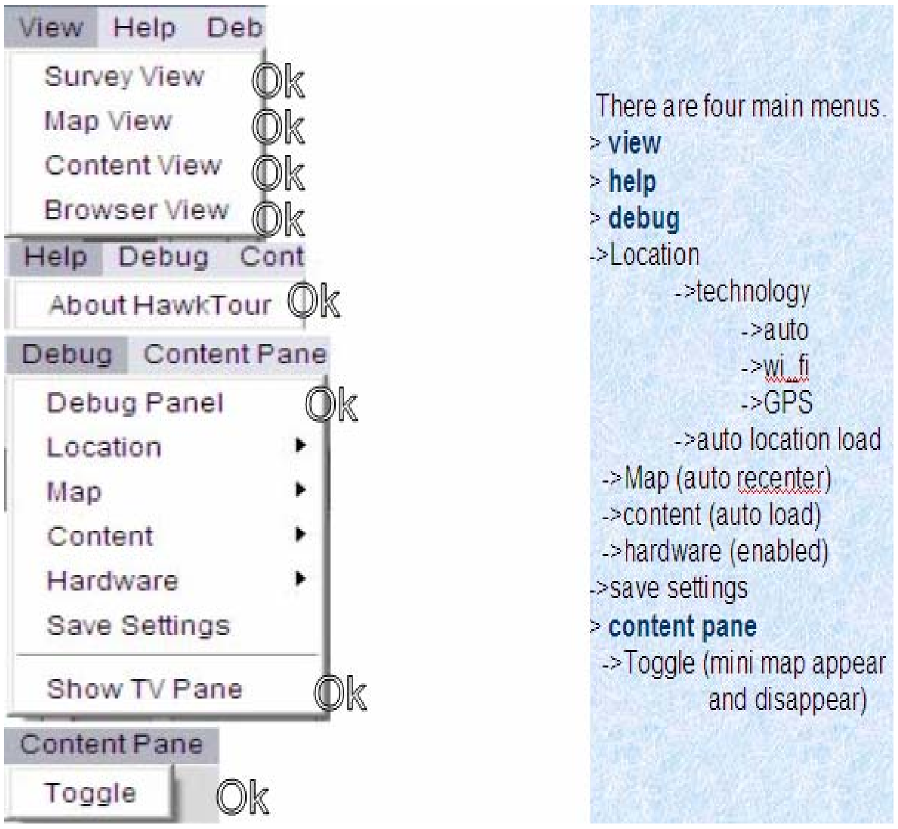
- Implemented user surveys in the application.
- Implemented user diaries in the application.

SCREEN SHOTS



... random text ... She cannot get automatic updates working, using instructions from link in security-related mass email. She has administrator privileges, and SP 4. The office is open 9 to 5... random text ...

Fig.3 HawkTour Main User Interface



There are four main menus.
> **view**
> **help**
> **debug**
->Location
->technology
->auto
->wi_fi
->GPS
->auto location load
->Map (auto recenter)
->content (auto load)
->hardware (enabled)
->save settings
> **content pane**
->Toggle (mini map appear and disappear)

Fig.4 HawkTour Menu Tree

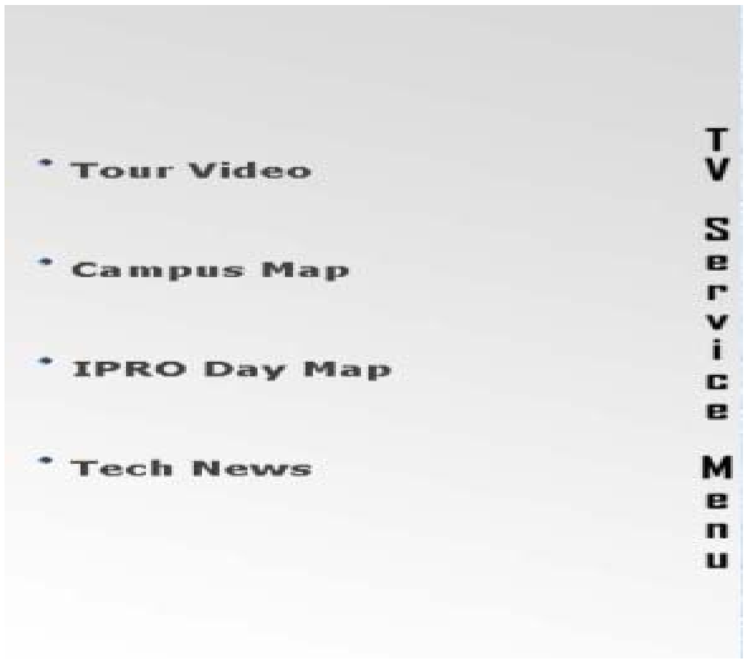
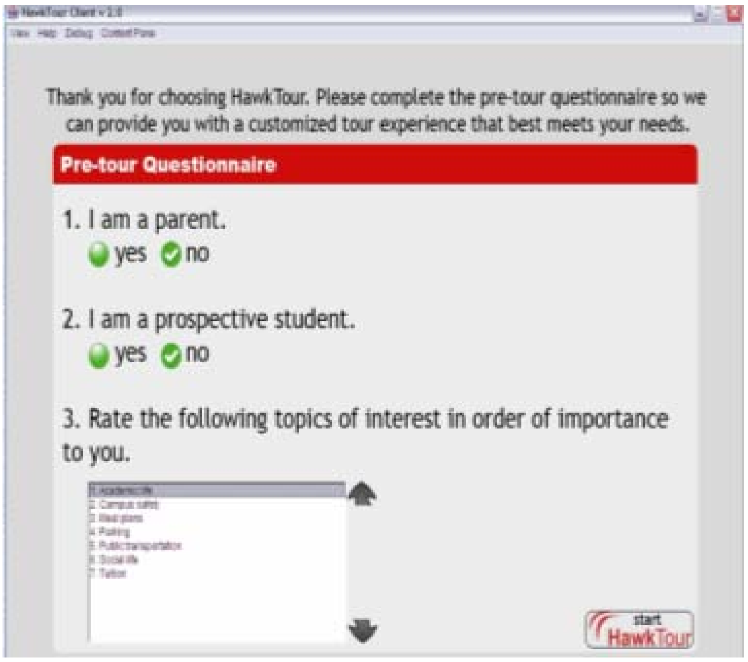


Fig.5 HawkTour TV Panel



Thank you for choosing HawkTour. Please complete the pre-tour questionnaire so we can provide you with a customized tour experience that best meets your needs.

Pre-tour Questionnaire

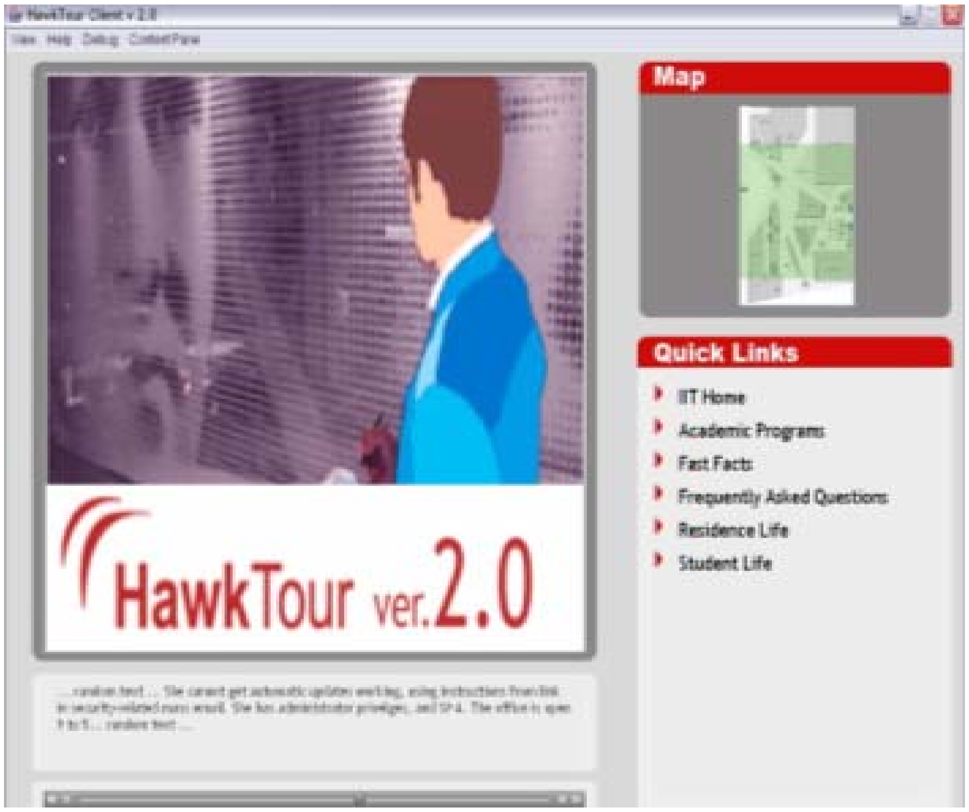
1. I am a parent.
   yes no

2. I am a prospective student.
   yes no

3. Rate the following topics of interest in order of importance to you.

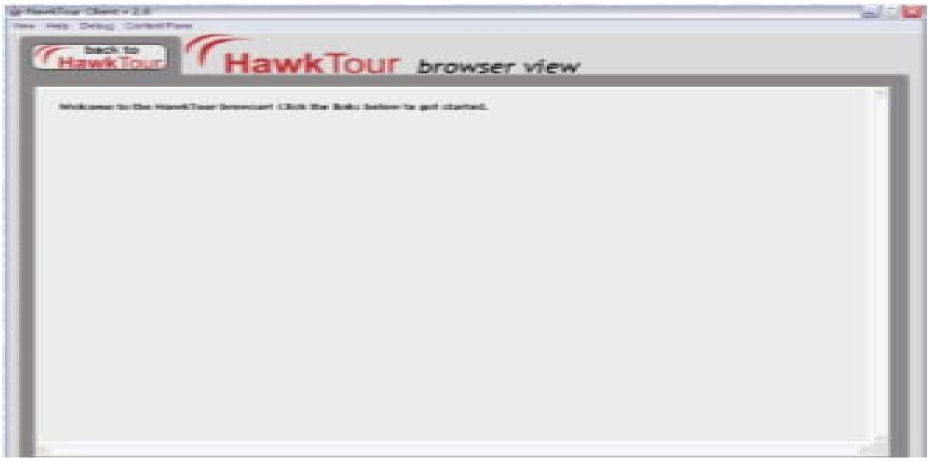
   1 Academic life
   2 Campus safety
   3 Meal plans
   4 Parking
   5 Public transportation
   6 Social life
   7 Tuition

Fig.6 HawkTour User Survey Panel



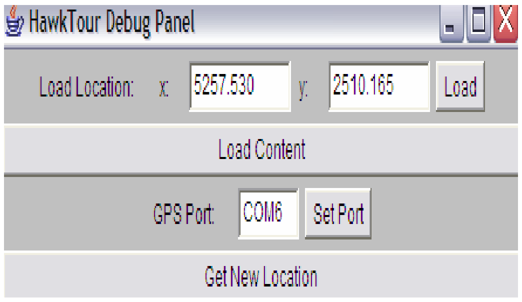
... random text ... [illegible] ... random text ...

Fig.7 HawkTour Content Panel



Welcome to the HawkTour browser! Click the links below to get started.

Fig.8 HawkTour Browser Panel



Load Location: x: 5257.530 y: 2510.165

GPS Port: COM6

Fig.9 HawkTour Debug Panel

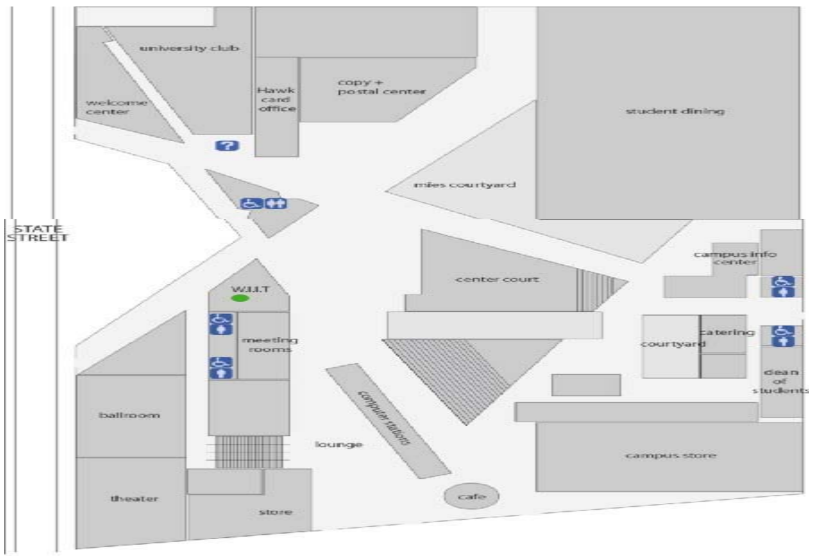


Fig.10 HawkTour Map

[5] Results and Conclusions

The goal of the continuing work is to extend the underlying technologies used in the tour application, and create a new application with additional functionality. Possible extensions to the project would be integration of the tour system with other devices within a building, implementing path-finding algorithms, so that users could request the application to direct them to specific locations, increasing support for locations not covered by wireless networks, which would necessitate the intelligent caching of content so that it could be delivered without network access (using a HawkTour on a Segway), and the creation of an administrative interface which would allow the tracking of all devices in the system at a central location. Such an application has uses in the fields of security and healthcare.

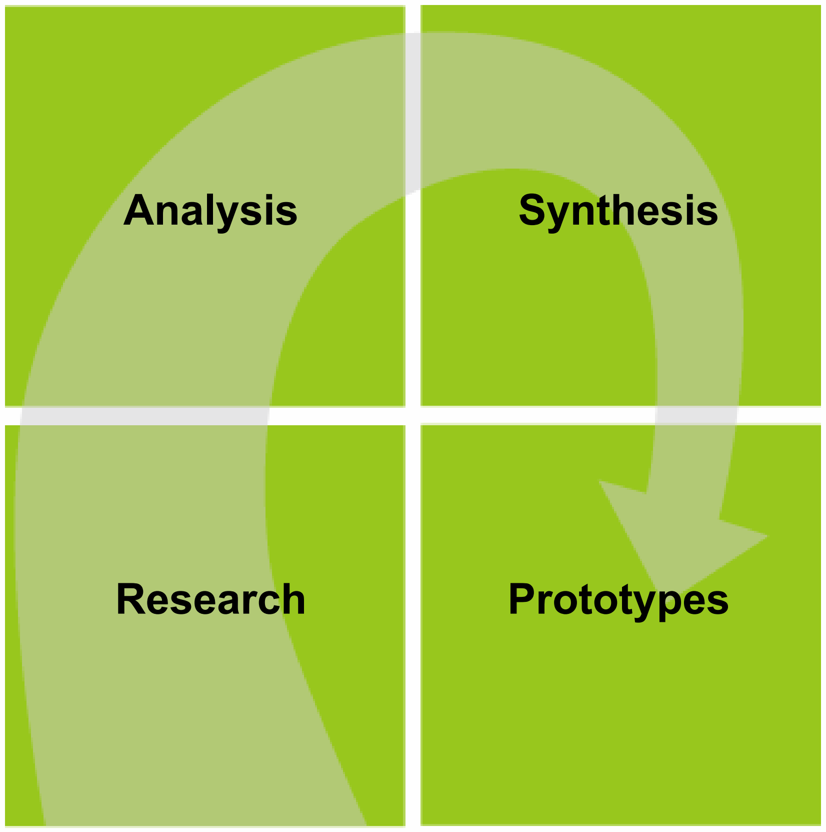
In addition, continuing integration of the underlying technology with other Pervasive Computing technologies, such as the cross-network phone system developed by an Illinois Institute of Technology graduate student, would be beneficial and challenging. For the coming semesters, a new project that uses the concept of Pervasive Computing will be developed and work will be started. However, HawkTour also will be continued over the next few semesters and improvements will be done to produce a robust, fully functional final product.

HawkTour:
User Research and Analysis

Prepared by Wonsuk Chung and Andrew Kim
Institute of Design, IIT
November 2004

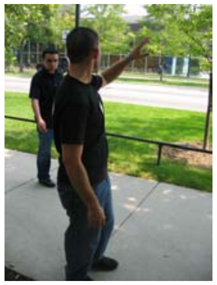
ILLINOIS INSTITUTE OF TECHNOLOGY

Innovation Process



User Research 1

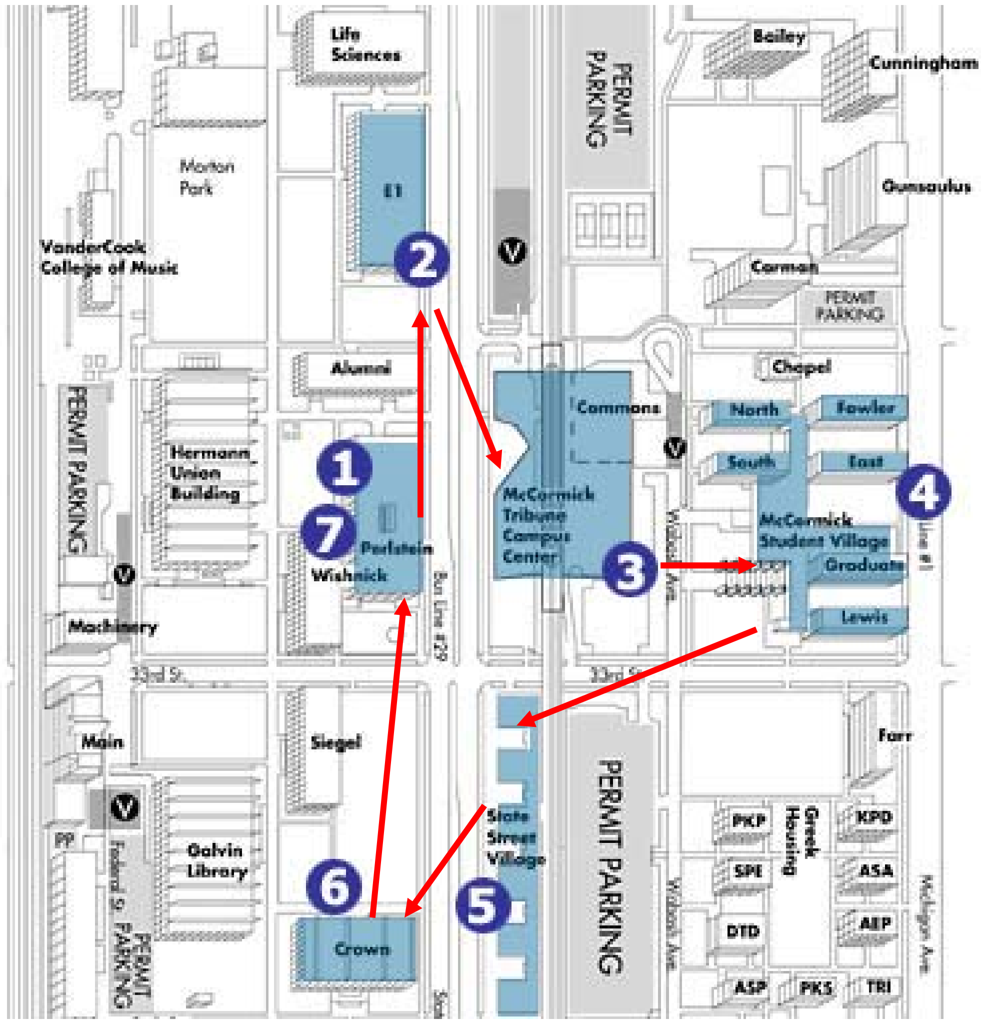
We Observed a Campus Tour… (Sep. 10, 2004)



- Friday at noon
- 1 hour tour
- Single student guide
- Group – 3 students
- Appointment suggested
- Offered once or twice a day

User Research 1

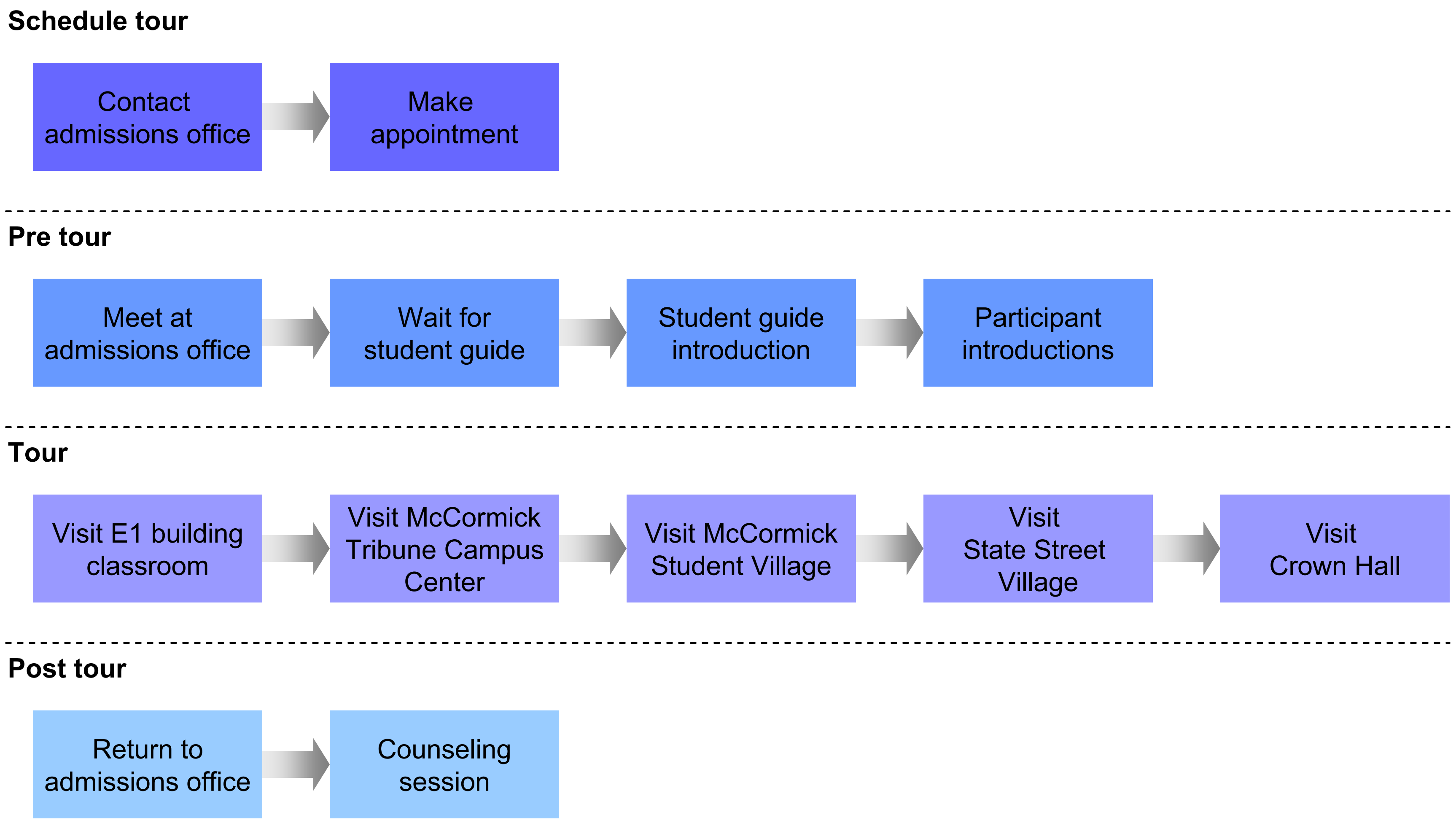
Campus Tour Route



1. Admissions office
   - meet guide
2. E1 Building
   - classroom
3. McCormick Campus Center
   - orange walkway
   - wall clock
   - post office
   - grill
   - student offices
4. McCormick Student Village
   - lounge
   - model room
5. State Street Village
   - lounge
   - balcony
6. Crown Hall
7. Return to admissions office

User Research 1

Campus Tour Activity Map


Schedule tour
Contact admissions office
Make appointment
Pre tour
Meet at admissions office
Wait for student guide
Student guide introduction
Participant introductions
Tour
Visit E1 building classroom
Visit McCormick Tribune Campus Center
Visit McCormick Student Village
Visit State Street Village
Visit Crown Hall
Post tour
Return to admissions office
Counseling session

User Research 1

After the Tour, We Interviewed the Student Guide…

Zachary Hartnett

• Second year business major
• Soccer athlete

Comments/suggestions

• Students typically joined by their parents and sometimes their siblings
• Only graduate and transfer students come alone
• A good guide:
  – reacts to the tour group
  – knows what the group wants
• Most people ask questions about:
  – campus safety
  – class load and difficulty
  – tuition

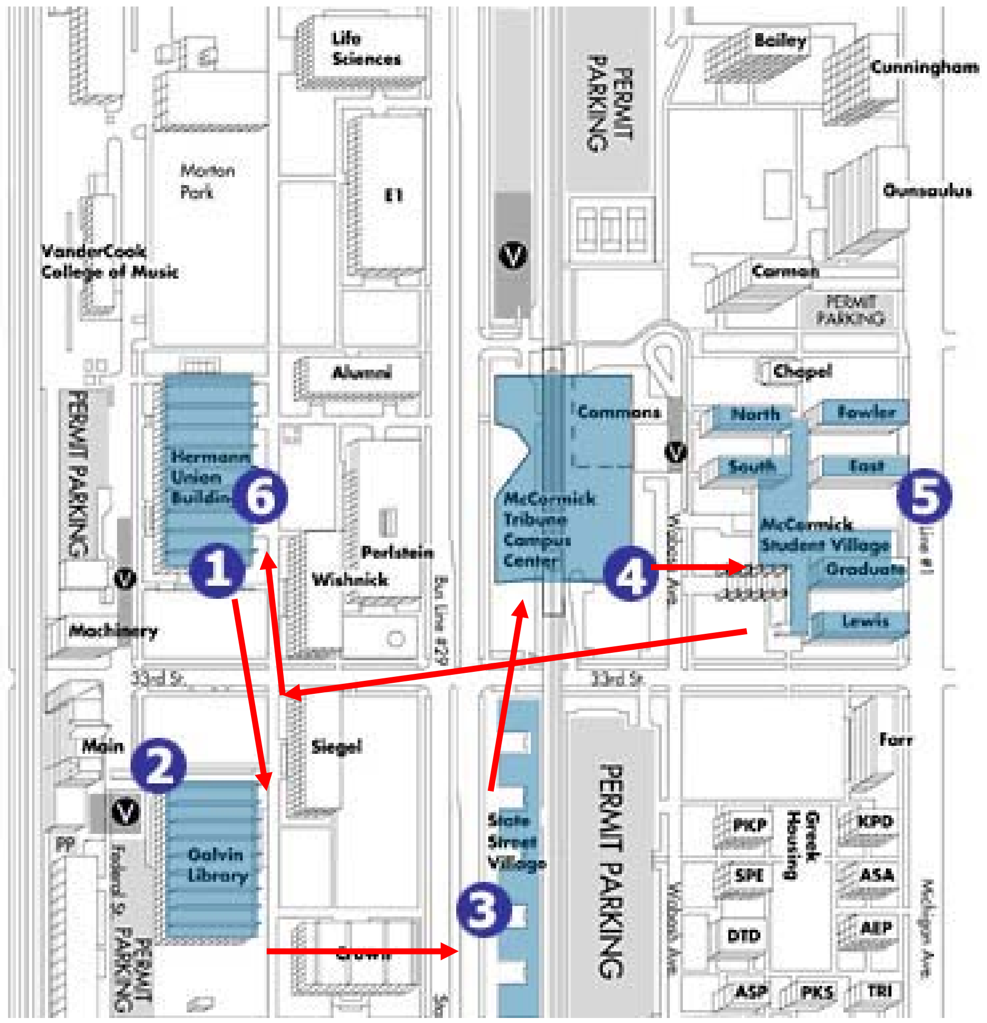
User Research 2

We Observed an Open House Tour… (Oct. 16, 2004)

- Part of weekend open house
- Saturday, multiple times
- 1 hour tour
- Multiple student guides for each group
- Groups of 20 to 30 students and parents

User Research 2

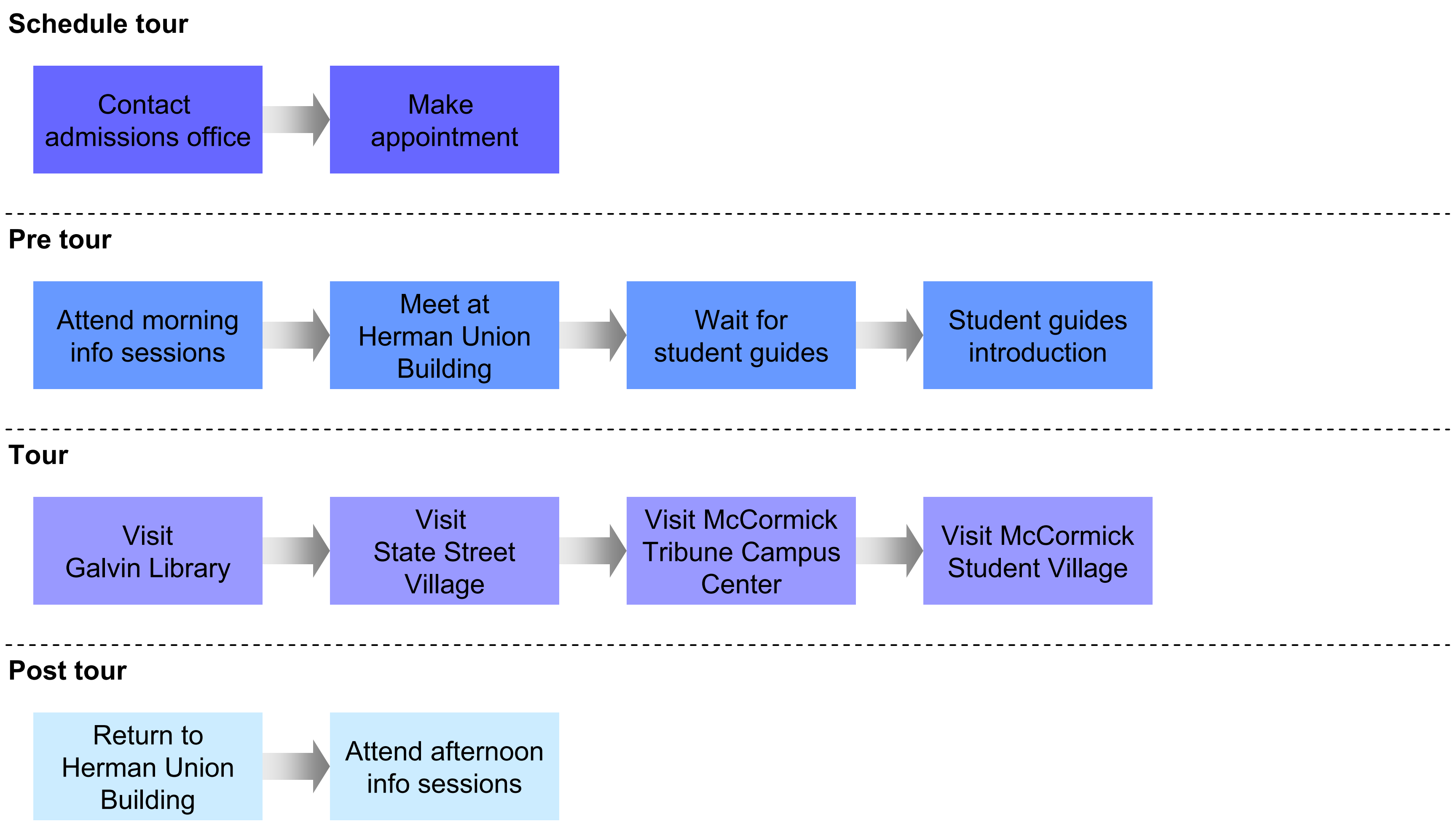
Open House Tour Route



1. Herman Union Building
   - divide into groups
   - meet guides
2. Galvin Library
   - resource center
3. State Street Village
   - lounge
   - model room
4. McCormick Campus Center
   - cafeteria
5. McCormick Student Village
   - model room
6. Return to Herman Union

User Research 2

Open House Tour Activity Map and Era Analysis


Schedule tour
Contact admissions office
Make appointment
Pre tour
Attend morning info sessions
Meet at Herman Union Building
Wait for student guides
Student guides introduction
Tour
Visit Galvin Library
Visit State Street Village
Visit McCormick Tribune Campus Center
Visit McCormick Student Village
Post tour
Return to Herman Union Building
Attend afternoon info sessions

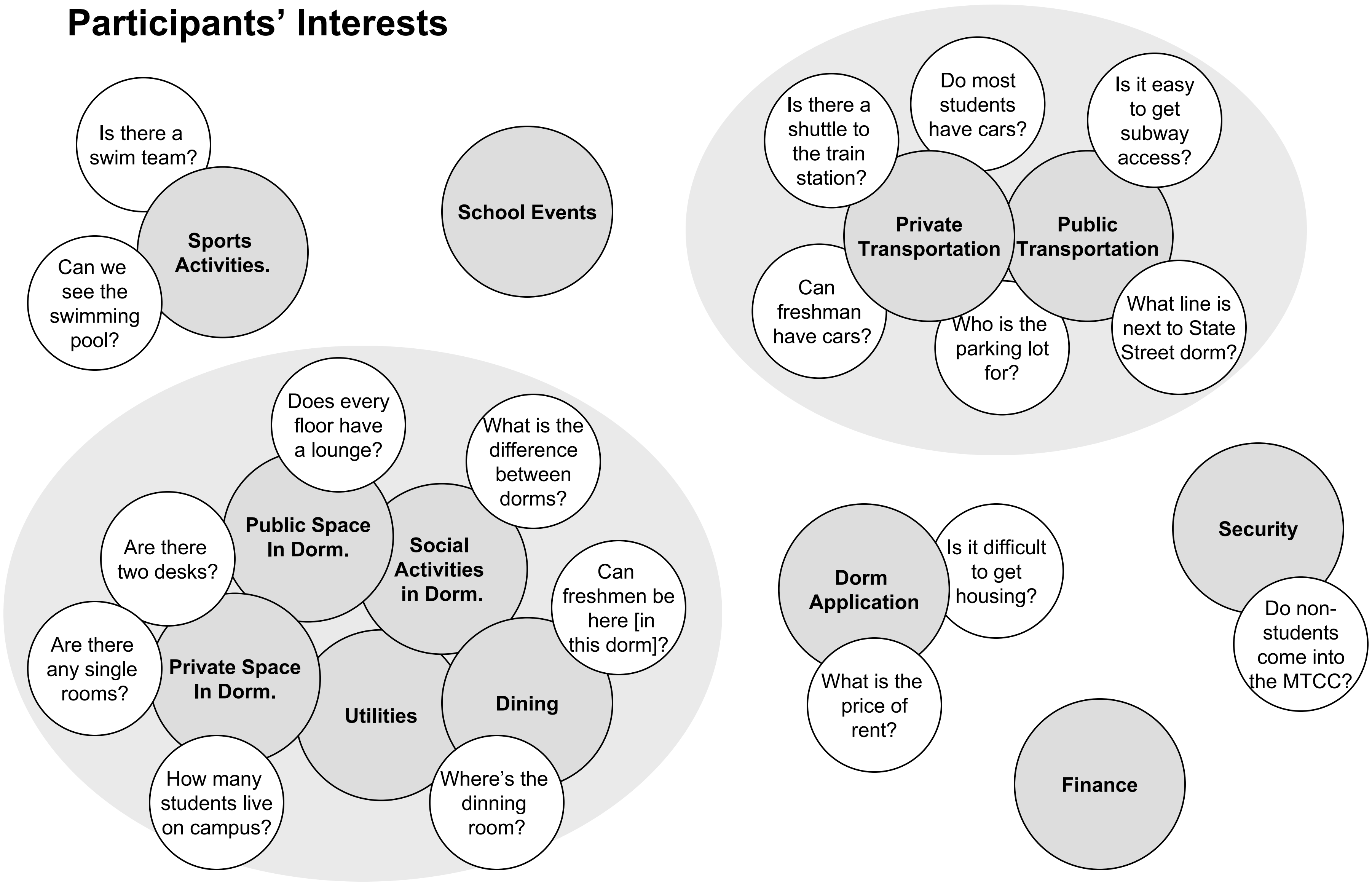
Analysis

We Conducted a Cluster Analysis…

- Collected photos
- Wrote down observations
- Grouped photos and observations into meaningful clusters
- Created group and subgroup headings

Analysis

Participants' Interests


Is there a swim team?
Sports Activities.
Can we see the swimming pool?
School Events
Is there a shuttle to the train station?
Do most students have cars?
Is it easy to get subway access?
Private Transportation
Public Transportation
Can freshman have cars?
Who is the parking lot for?
What line is next to State Street dorm?
Does every floor have a lounge?
What is the difference between dorms?
Public Space In Dorm.
Social Activities in Dorm.
Are there two desks?
Can freshmen be here [in this dorm]?
Are there any single rooms?
Private Space In Dorm.
Utilities
Dining
How many students live on campus?
Where's the dinning room?
Dorm Application
Is it difficult to get housing?
What is the price of rent?
Security
Do non-students come into the MTCC?
Finance

Analysis

Parents ask more questions than students

16 to 2

The number of questions asked by parents versus students on a single tour

Analysis

Key Discoveries Drawn From the Cluster Analysis

1. Tour
   - Rich interaction between student guides and tour group participants
   - More time spent in dormitories than in the student center
   - Limited amount of time spent in McCormick Student Center
2. Guide
   - Student guides customize experience
   - Student guides’ use of gestures
3. Participants
   - Prospective students usually accompanied by their parents and sometimes siblings
   - Tour participants asked many questions about:
     - dorm life
     - transportation
     - tuition
   - Parents ask many more questions than prospective students
   - High number of questions asked as the tour group is moving from point to point

Analysis

Insights Drawn From Key Discoveries

1. Tour
   - Successful group interaction starts with knowing participant’s interests
2. Guide
   - A positive tour experience depends on the guides’ engagement with the participants
3. Participants
   - Tour participants are most interested in day to day campus life and
   - Students are intimated from asking questions in the tour group setting
   - The tour was more parents driven

Synthesis

HawkTour Opportunities

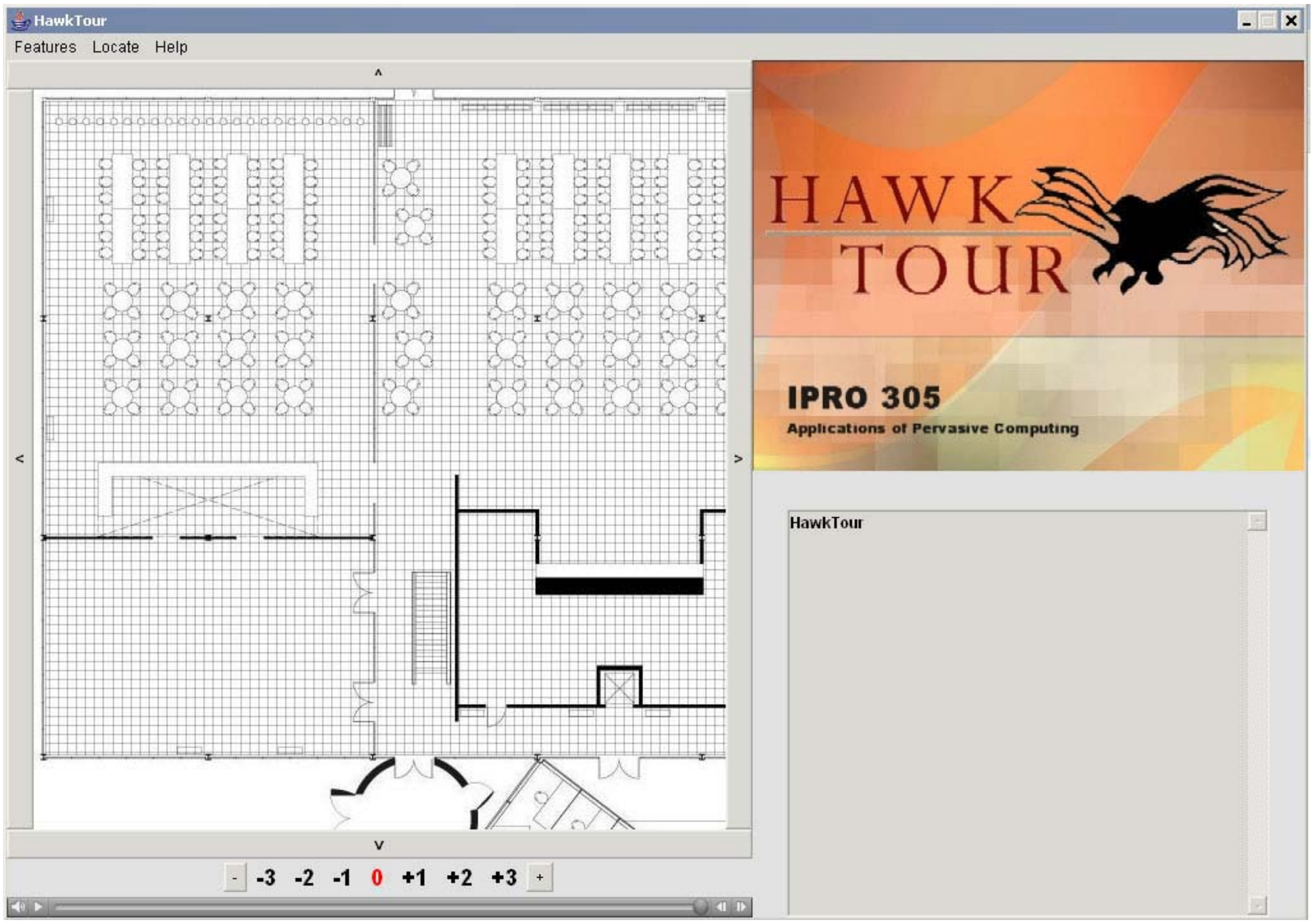
1. Interaction
   - Pre-tour questionnaires
   - Chatting/text messaging with students/admissions office
2. Engagement
   - Participants have the ability to choose the tour topic
3. Contents
   - User generated contextual content
     - Current IIT Students
     - Tour participants
   - Student/Parent Diaries
   - Day to Day Campus Life

Synthesis

HawkTour Open Issues

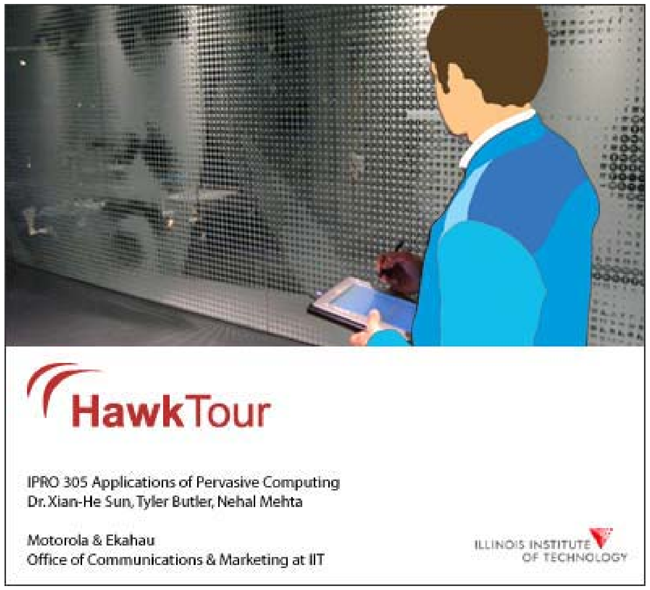
- Is HawkTour a prospective student or architecture tour application?
- Is HawkTour for prospective and/or currents students?
- Can HawkTour be used for guided led tours?

Old Hawk Tour Interface



Prototype

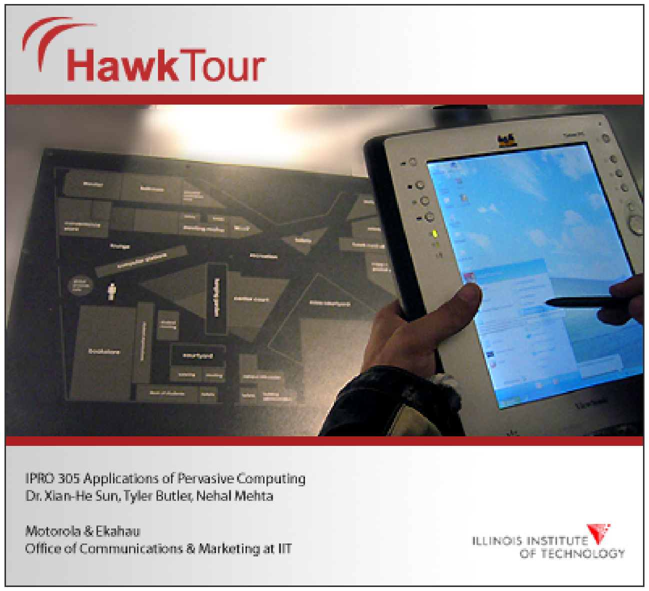
Start-up screen options as client application is loading


HawkTour

IPRO 305 Applications of Pervasive Computing
Dr. Xian-He Sun, Tyler Butler, Nehal Mehta

Motorola & Ekahau
Office of Communications & Marketing at IIT

ILLINOIS INSTITUTE OF TECHNOLOGY


HawkTour

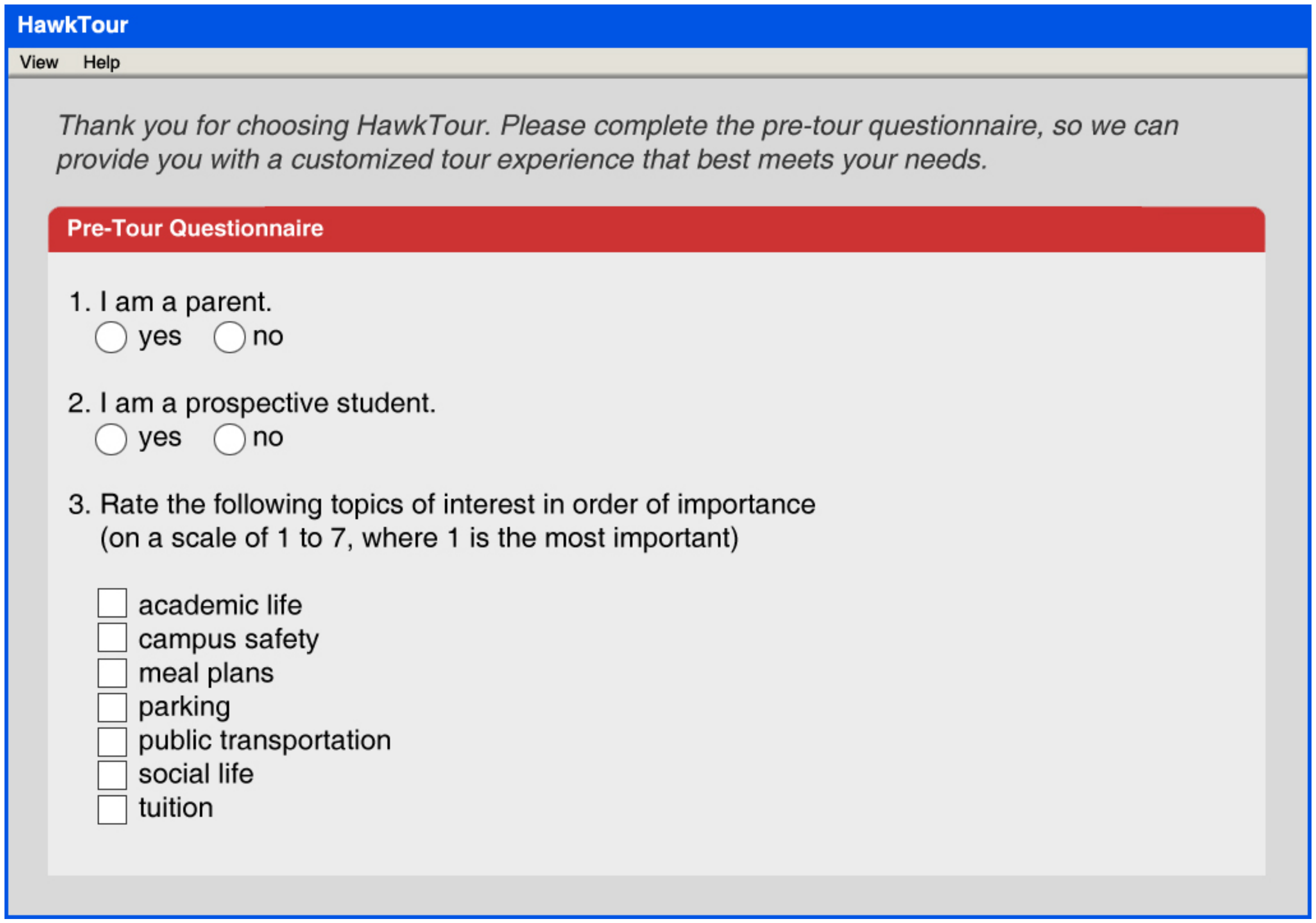
IPRO 305 Applications of Pervasive Computing
Dr. Xian-He Sun, Tyler Butler, Nehal Mehta

Motorola & Ekahau
Office of Communications & Marketing at IIT

ILLINOIS INSTITUTE OF TECHNOLOGY

Prototype

Develop a pre-tour questionnaire for tour customization



Thank you for choosing HawkTour. Please complete the pre-tour questionnaire, so we can provide you with a customized tour experience that best meets your needs.

Pre-Tour Questionnaire

1. I am a parent.
   ◯ yes ◯ no

2. I am a prospective student.
   ◯ yes ◯ no

3. Rate the following topics of interest in order of importance (on a scale of 1 to 7, where 1 is the most important)

   ☐ academic life
   ☐ campus safety
   ☐ meal plans
   ☐ parking
   ☐ public transportation
   ☐ social life
   ☐ tuition

Prototype

Develop a pre-tour questionnaire for tour customization

Green icon indicates user's location followed by user trail

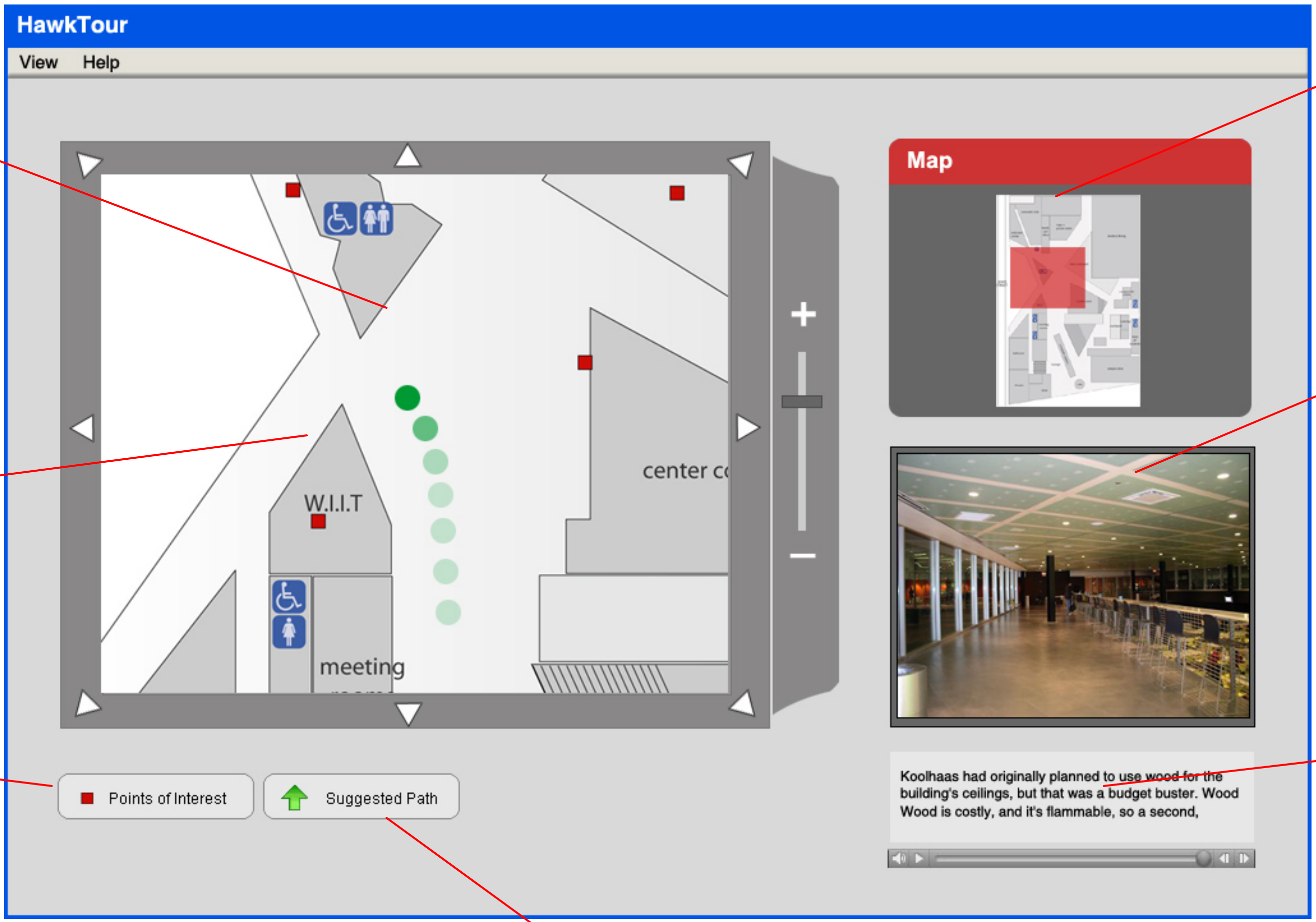
Point of interest

User can turn on and off points of interest and suggested path

Red box shows user's map screen

360 degree photo

Audio bar



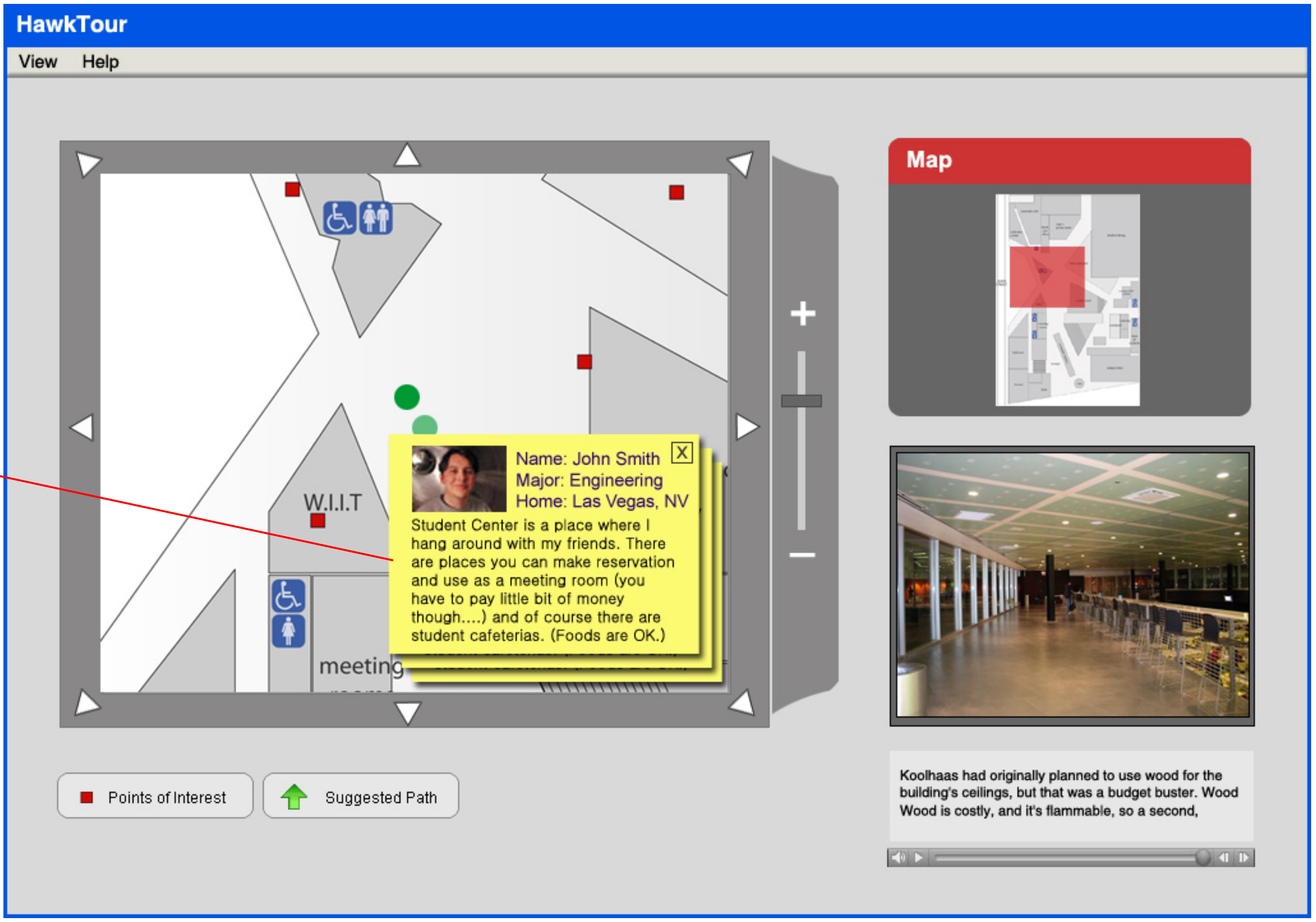
Koolhaas had originally planned to use wood for the building's ceilings, but that was a budget buster. Wood Wood is costly, and it's flammable, so a second,

Map provides a suggested path for touring the building, i.e. walking tour

Prototype

Users can view contextual data added by students and/or tour participants

Contextual comments by students



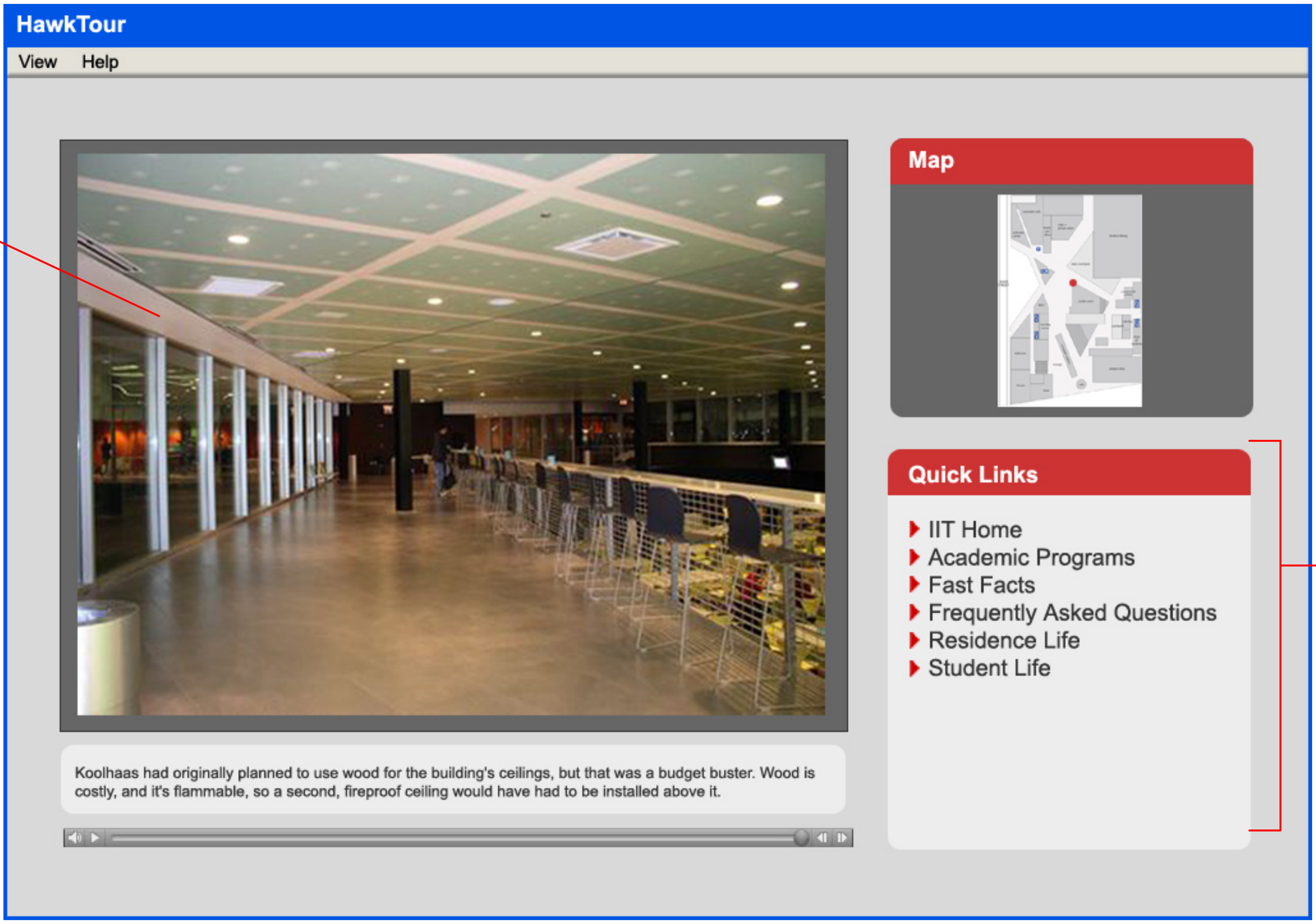
Name: John Smith
Major: Engineering
Home: Las Vegas, NV

Student Center is a place where I hang around with my friends. There are places you can make reservation and use as a meeting room (you have to pay little bit of money though....) and of course there are student cafeterias. (Foods are OK.)

Koolhaas had originally planned to use wood for the building's ceilings, but that was a budget buster. Wood Wood is costly, and it's flammable, so a second,

Prototype

Users have the option of switching to a content view (View → Image View)



Koolhaas had originally planned to use wood for the building's ceilings, but that was a budget buster. Wood is costly, and it's flammable, so a second, fireproof ceiling would have had to be installed above it.

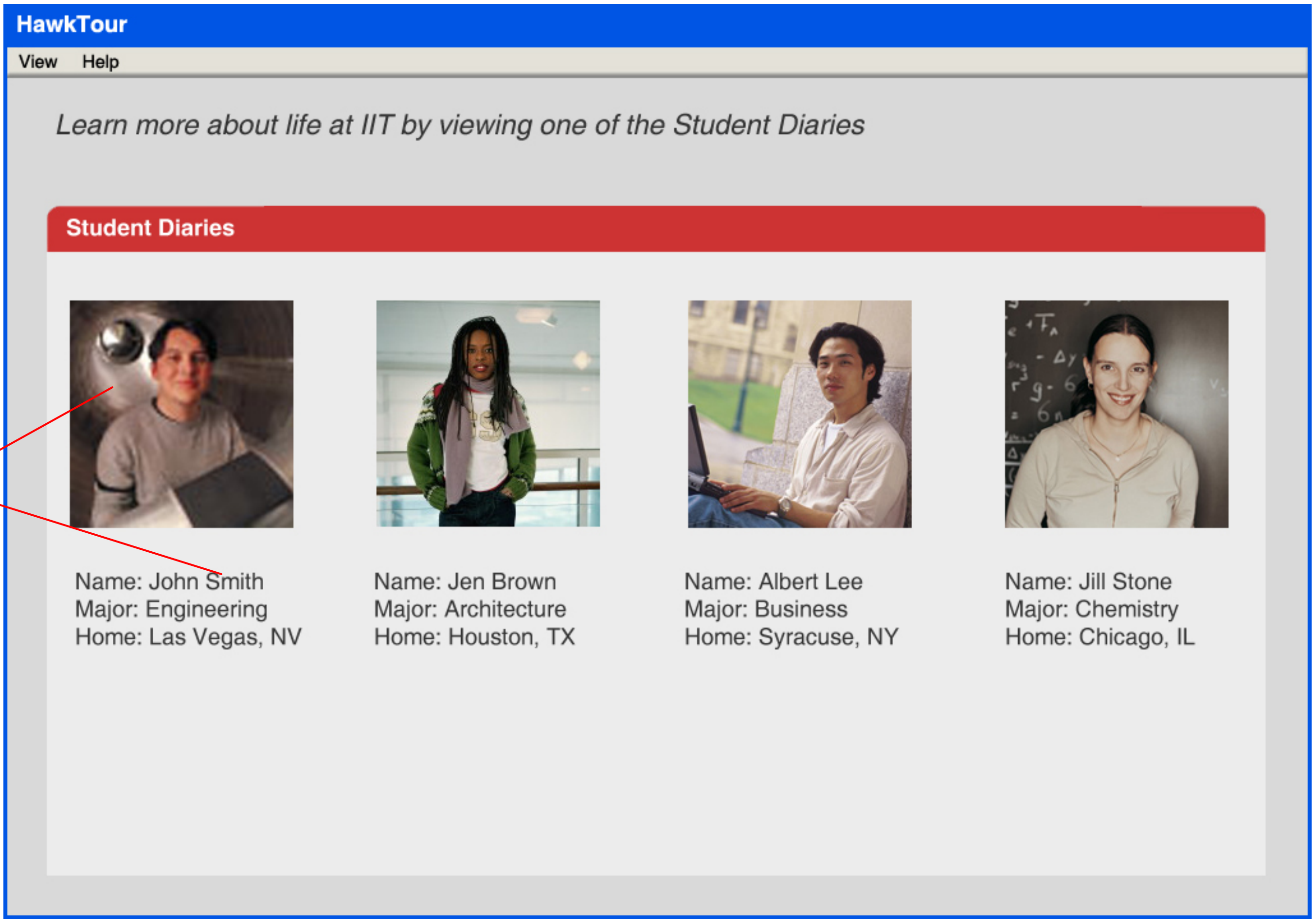
Contextual Image

Hyperlinks to IIT site

Prototype

Users can access student diaries

Click name or photo to view student diary



Learn more about life at IIT by viewing one of the Student Diaries

Name: John Smith
Major: Engineering
Home: Las Vegas, NV

Name: Jen Brown
Major: Architecture
Home: Houston, TX

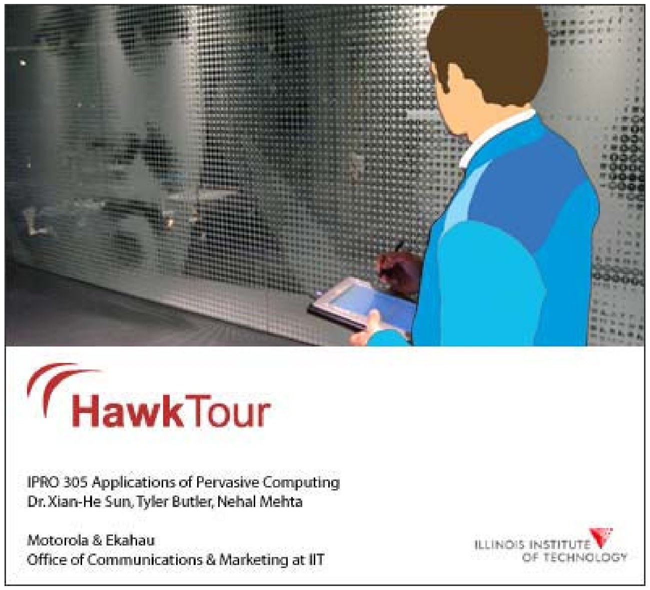
Name: Albert Lee
Major: Business
Home: Syracuse, NY

Name: Jill Stone
Major: Chemistry
Home: Chicago, IL

Prototype

About HawkTour Screen Options

Users can view the About Screen by going to Help → About HawkTour

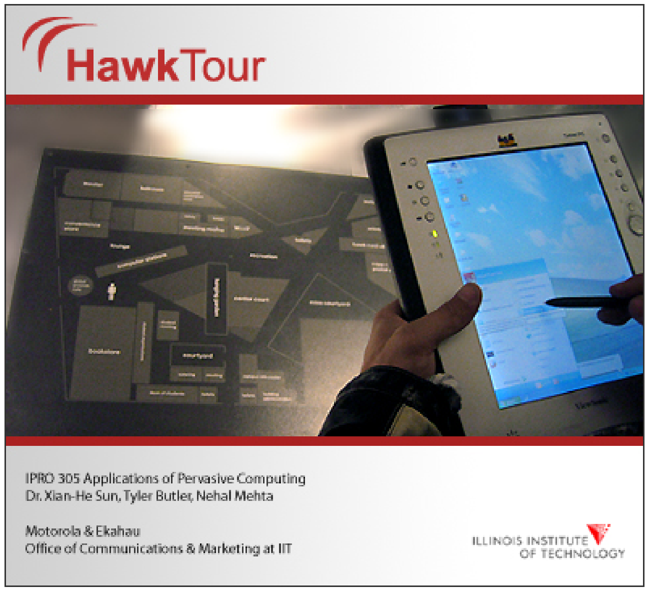


HawkTour

IPRO 305 Applications of Pervasive Computing
Dr. Xian-He Sun, Tyler Butler, Nehal Mehta

Motorola & Ekahau
Office of Communications & Marketing at IIT

ILLINOIS INSTITUTE OF TECHNOLOGY



HawkTour

IPRO 305 Applications of Pervasive Computing
Dr. Xian-He Sun, Tyler Butler, Nehal Mehta

Motorola & Ekahau
Office of Communications & Marketing at IIT

ILLINOIS INSTITUTE OF TECHNOLOGY

Appendix

Open House Tour Questions

Parent asked

- Is it easy to get EI access?
- Can we see the school swimming pool?
- Is there a swim team?
- Where’s the dining room?
- What line is next to State Street dorm?
- What percentage of students lives on campus?
- Who is the parking lot for?
- Can freshman have cars?
- Do most students have cars?
- Does every floor have a lounge?
- Is it difficult to get housing?
- There are two desks? Is the TV included?
- Can freshman be here [in this dorm]?
- Are there any single rooms?
- Is there a shuttle to the train station?
- Do many public people [non-students] come into the MTCC?

Appendix

Open House Tour Questions Continued

Student asked

- What is the difference between dorms?
- What is the price of rent?

IPRO 305

The Applications of Pervasive Computing

HawkTour Developer's Guide

by

Daniel Wolkenfeld

November 11, 2004

Table of Contents

[1] Overview

SCOPE OF THIS DOCUMENT

This document serves as a guide for new team members, and a reference for returning team members to the design and implementation of HawkTour. The application has gone through many iterations of redesign and evolution by the successive IPRO 305 teams, but this document is the first attempt to capture the architecture of the functional application as a whole. This document explains how each component and service of HawkTour is designed and why things were implemented the way they were. Undoubtedly, this document will need to be updated at the end of each semester to reflect the new design and features of HawkTour. In the future it would be best to make this document into a web site with links of each class name that occurs linking to its associated JavaDoc page. For more specific details on how the code works, refer to the JavaDocs, and the code itself.

Conventions in This Document

When referring to a class file, as in "class.java" for example, the name of the class will be in blue (RGB: 0,0,128) without the .java extension.

INTRODUCTION TO PERVASIVE COMPUTING

Why Pervasive Computing?

As hardware integration increases to amazing new levels with each generation of design, devices increase in their functionality and versatility. They also become cheaper, smaller, and handier. In tech-savvy societies, people are becoming more and more dependent upon the technology that they carry with them – mobile phones, portable computers, calculators, PDAs, music players, and navigation devices – to the point where an increasing majority of information that people consume and produce daily is in electronic format. With the technology abounding, and with networks connecting them, we can integrate the functionality into a networked application that utilizes this information and makes the technology more useful to us in a whole new way.

***What is Pervasive Computing*?**

When a technology becomes widely accepted by society, it tends to disappear. This means that it is there, is easily affordable, and people may use it heavily enough to be dependent upon it, but the technology itself does not require a significant amount of effort or skill to be able to use it so as to get in the way or stand out as an "experience" for the user. A good example of this is electricity. It is available virtually everywhere in the world and people simply plug in whenever they need to, or go equipped with batteries. This has happened with a number of technologies in the 20th century in the Western world, for example transportation, telephone service, and junk food. The user simply benefits from the service of the technology without regard to particular maintenance or ownership of the attendant equipment. The same thing has begun to happen with computer technology. We interact with computers many times on a day-to-day basis, but traditional uses of computer technology, such as controlling the elevator, the traffic light, the ATM, or the telephone auto-attendant at the customer service department, are not *invisible*. In most cases they are a barrier, a substitute for human interaction. Pervasive computing, on the other hand, is *ubiquitous* technology, used as a tool for providing just-in-time information, or to assist users in some way.

Why is this computer system different from all other computer systems?

The thing that distinguishes pervasive computing from "a bunch of computers all around us that we use all the time" is that a pervasive application is *context aware*. It takes into account one or more environmental data such as your location, the weather, the time, your personal information, your biometrics, etc., in determining its behavior. A pervasive application also uses shared data from other systems, applications, or services.

WEB SERVICES AND HAWKTOUR

To facilitate the needs of a mobile, pervasive application, it is very convenient to transfer data across components of the application using Web Services. This is not only convenient, but it allows HawkTour modules to be virtually hot swappable, so that the application does not have to rely on any particular technology. In terms of modules, right now the major components of HawkTour are the Location Service, the Content Service and the local HawkTour client, of which the Location Service and Content Service are web services. As a side-note, a middleware web service called Scarlet can be used to broker these and other web services with HawkTour.

However, Scarlet is not important to this version of HawkTour. So in the future, this application may interact with more web services for retrieving other contextual data.

MODEL VIEW CONTROLLER PARADIGM

Referred to as MVC, this software architecture concept was first developed at the Xerox Palo Alto Research Center for their Smalltalk-80 system for the multi-windowed graphical interface – the first in the world. Succinctly, the Model section of the program is the function, algorithm, or state-machine; the View is the output, usually a rectangular portion of the screen; and the Controller is the input handler / interpreter. Within this triad there is heavy communication between the View and the Controller, and communication between the model and each of the view and the controller. HawkTour is built on the MVC design pattern, with the HawkTourClient (GUI) corresponding to the View, the MainManager corresponding to the Controller, and each of the back end services corresponding to the Model.

[2] Application Architecture

HIERARCHY

HawkTour went through a design overhaul over the summer of 2004 to make it more organized and hierarchical, and to prevent headaches from reading the code. The main purpose, besides code readability, however, was to implement a consistent interface for the application to communicate with the different services. The motivation for this was to make the different services completely independent of each other, so that the application would not be inherently tied to a technology used to implement one of the services. For example, right now Quicktime is used for content delivery, and Ekahau is used for WiFi positioning. Either of these components can and may be upgraded to newer or more favorable technologies in the future, or another development group may want to substitute a technology more appropriate for their purpose for their version of the HawkTour application. The graphs on the following pages illustrate how each of the component services works.

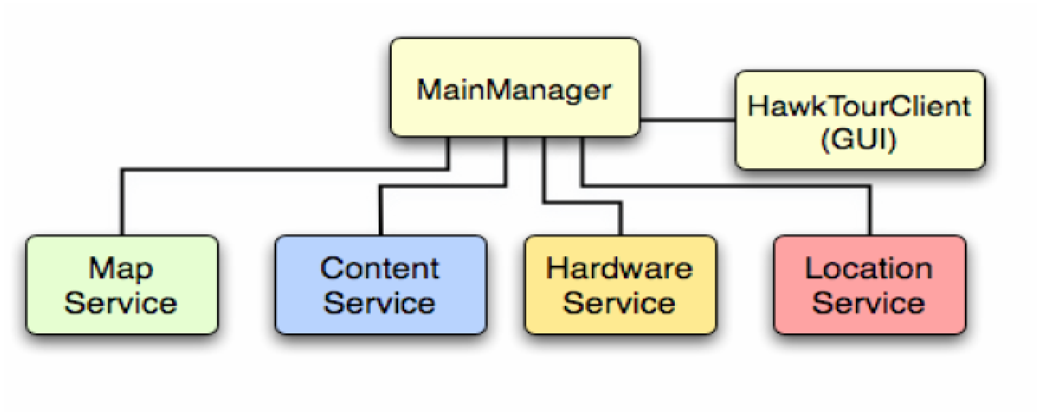


Fig.1 HawkTour Hierarchy

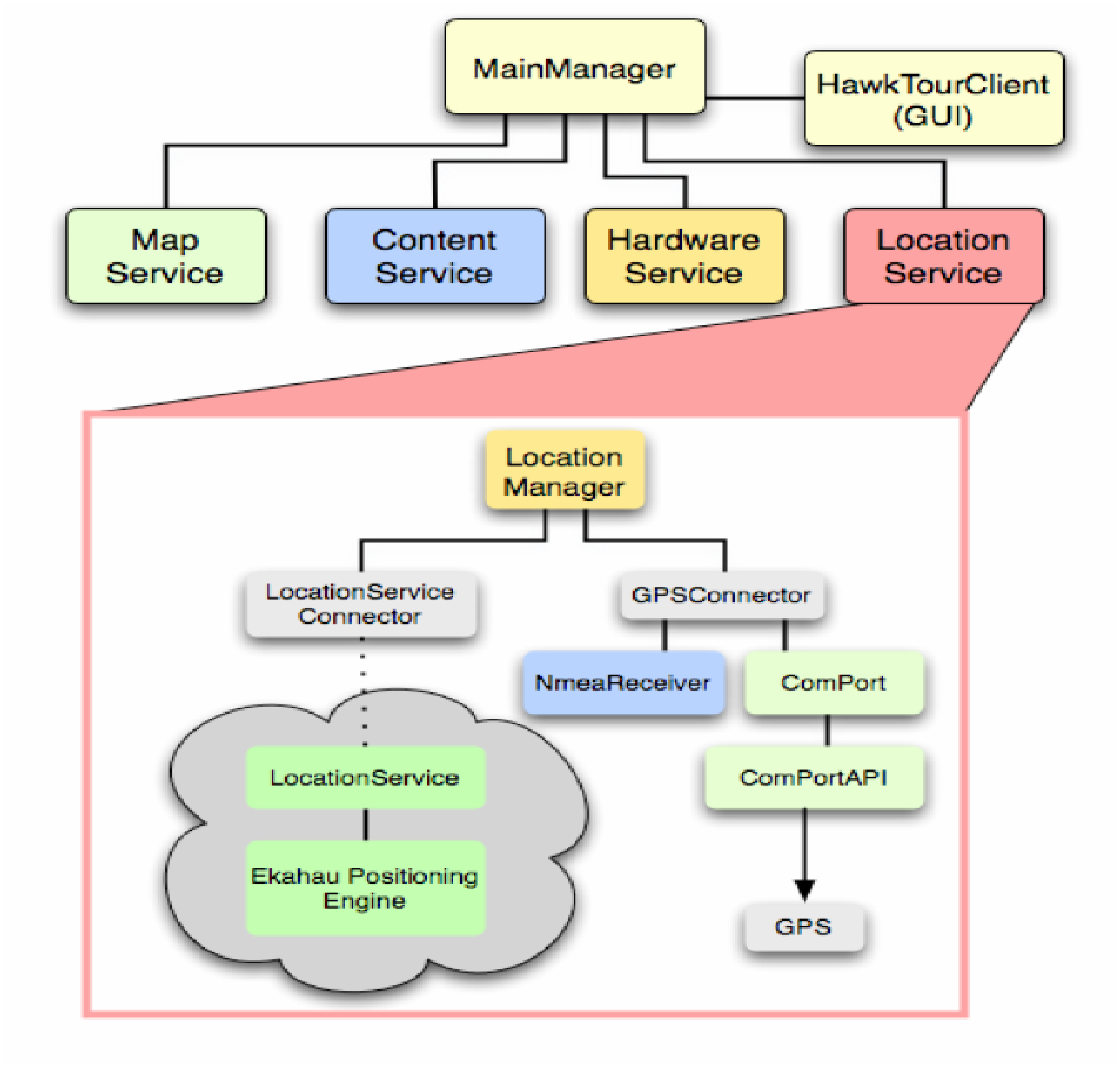


Fig.2 HawkTour Location Service

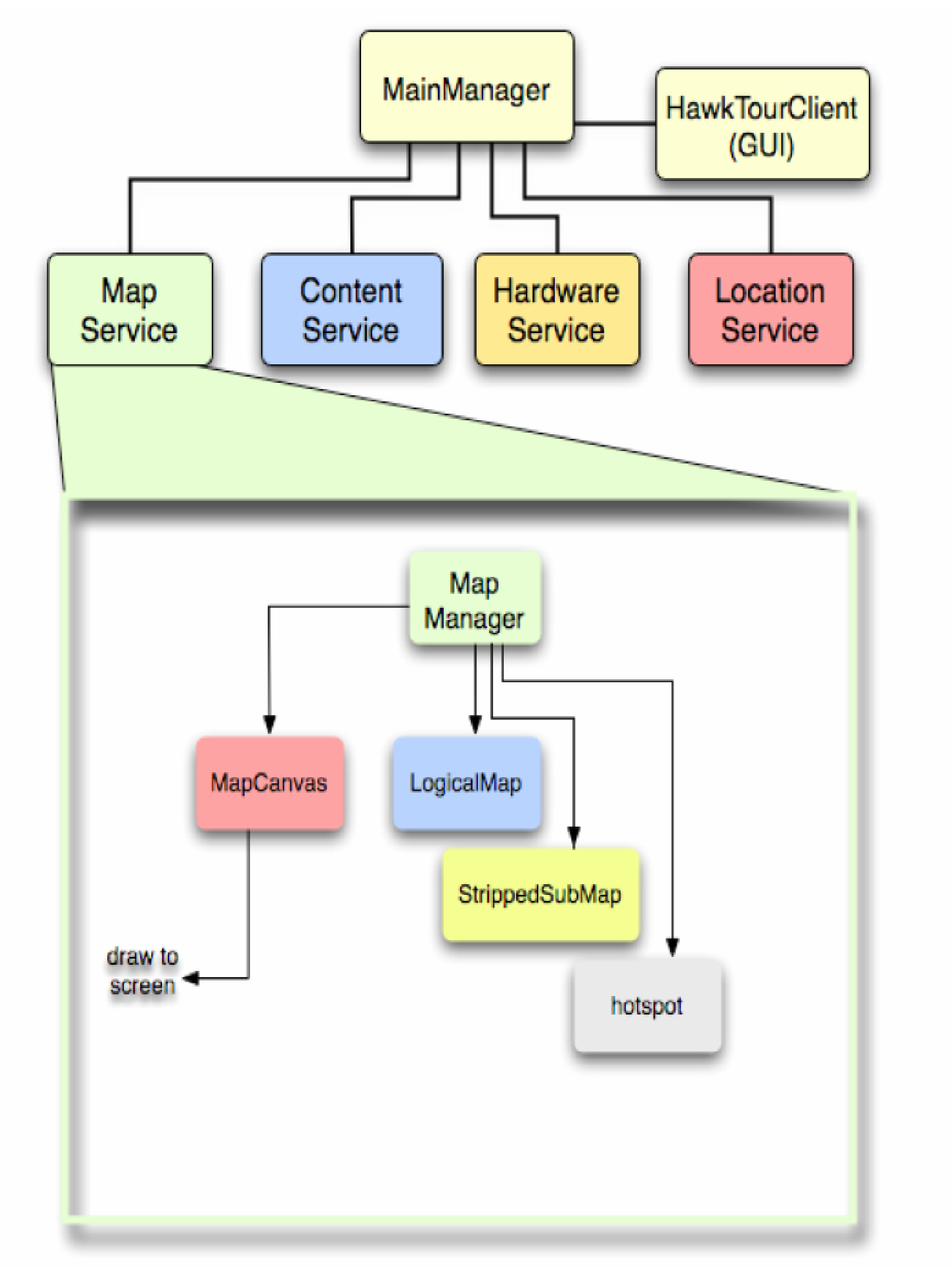


Fig.3 HawkTour Mapping Service

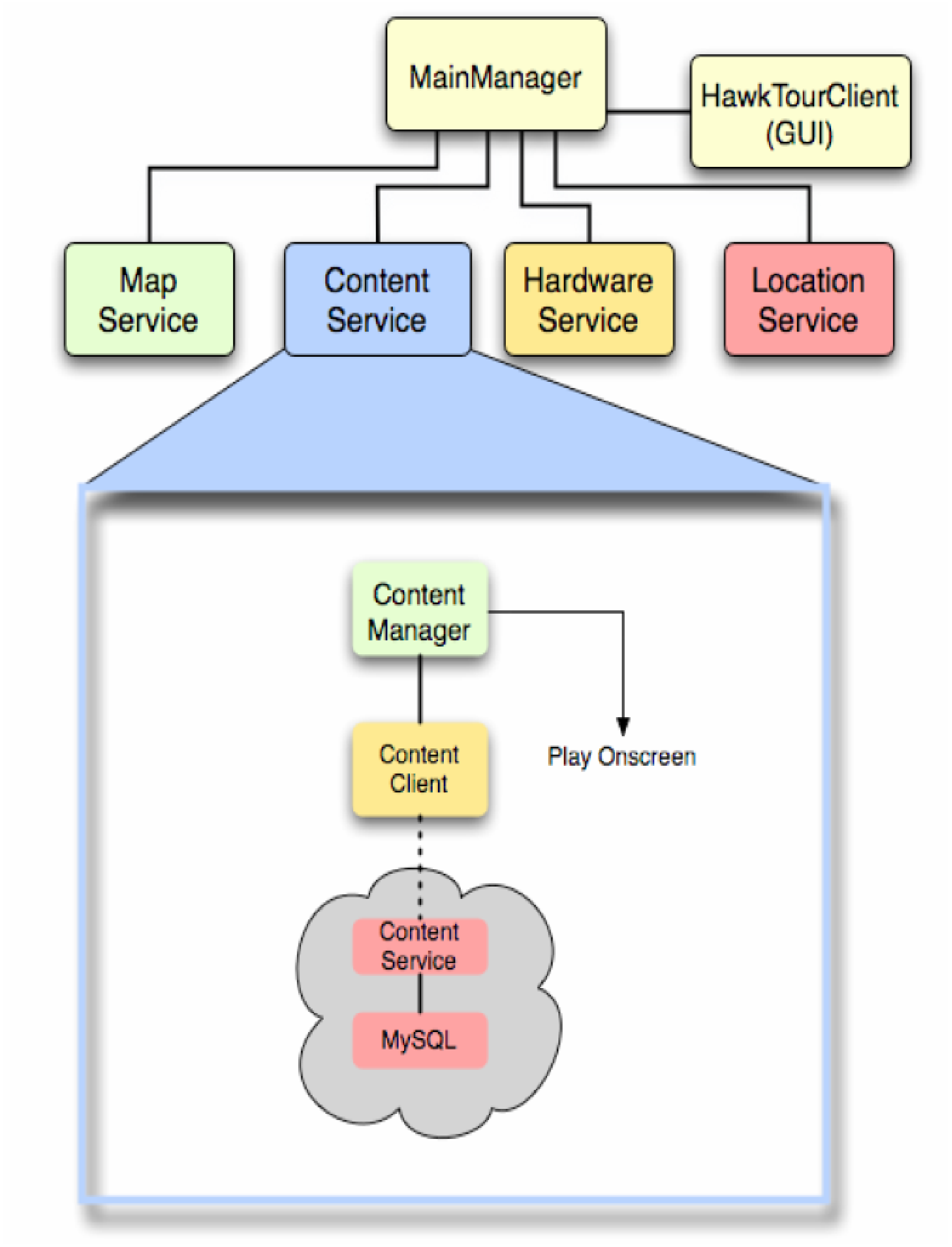


Fig.4 HawkTour Content Service

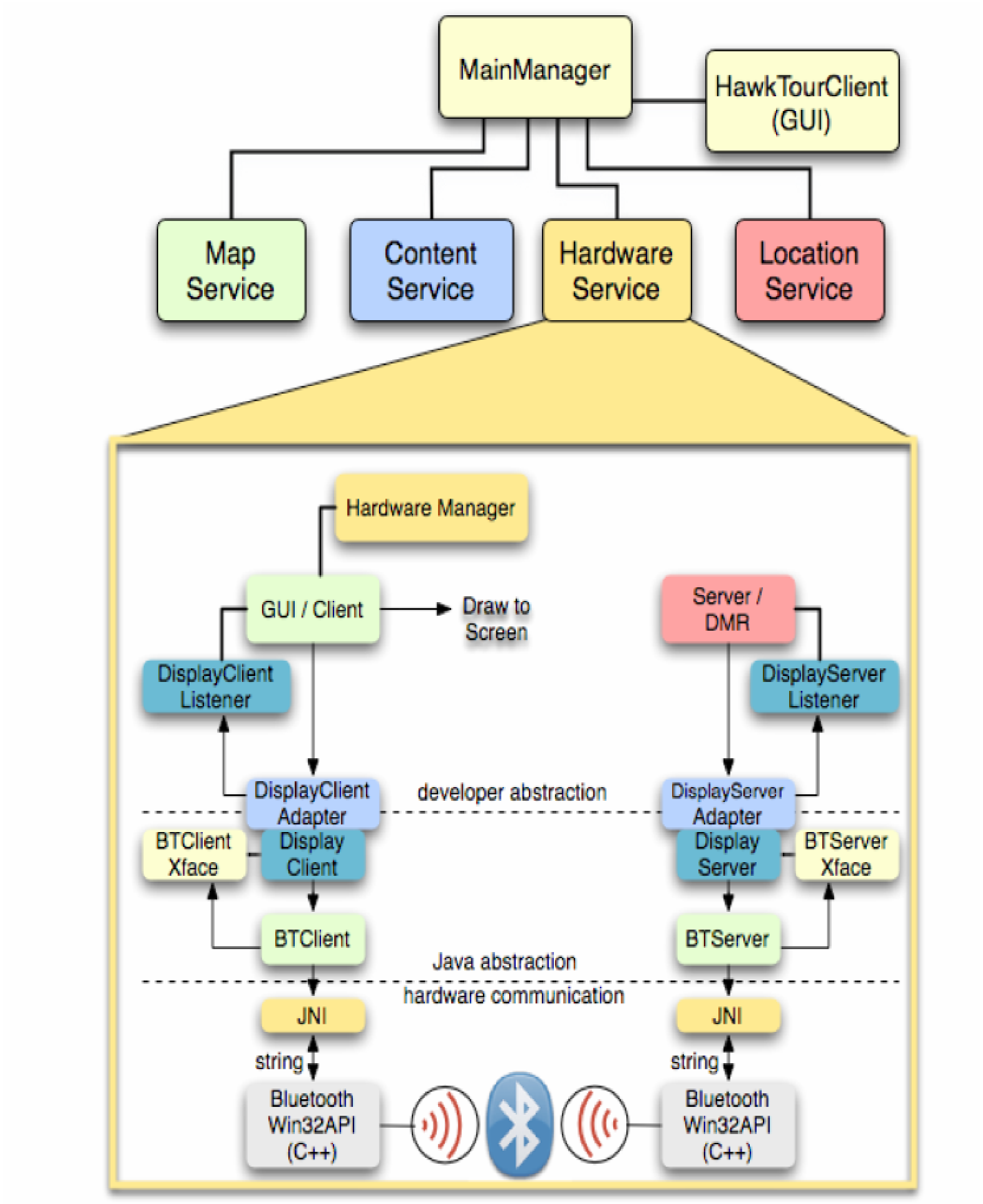


Fig.5 HawkTour Hardware Interface Service

[3] Application Implementation

MAIN MANAGER

MainManager is the mother of all classes of HawkTour. This class was written in Fall 2004 as part of the move to de-spaghettify the code and make it more hierarchical. It is a Singleton class, and creates instances of the "middle managers" {LocationManager, MapManager, ContentManager, and HawkTourClient}. It acts as the middleman between them. If, for example, the application needs to find the current location, you just call the name of the MainManager (mainMan for the sake of argument), as in mainMan.Location().

LOCATION MANAGER

The LocationManager class is another result of the move towards de-spaghettification. It is in charge of gathering location data from the LocationService. The LocationService, in turn, connects to the locator servers and puts the information in a plain old Vector, called LocationData. The format of LocationData is (in Global coordinates) X (longitude), Y(latitude), speed, heading. There are currently two location services: WiFi and GPS. The LocationServiceConnector (package edu.iit.cs.scs.HawkTour.ServiceCommunication) retrieves the location data from the WiFi location server, which right now uses the Ekahau Positioning Engine. If a different technology becomes more favorable in the future, a separate Connector class would be written for it, and the LocationServiceConnector would be deprecated. GPSConnector, of course, gets the information from the GPS device, which "happens to be" on the local host, through the ComPortAPI class (package edu.iit.cs.scs.HawkTour.LocalServices). But it is cleaner to have a separate Connector for each locator service, and they all return values to LocationService.

In addition to gathering data, LocationManager also arbitrates which locator service the application is using, so that the source of information is hidden from the application. It simply asks LocationManager for the current location, and is returned an accurate value, from whichever service the LocationManager deems most suitable for the present conditions. Currently, automatic arbitration is not workable, so a manual mode switcher is being used.

MAP MANAGER

MapManager controls the classes that are related to the mapping service in HawkTour. Right now we only have one MapManager constructor and it accepts a MainManager object, 2 floats, 1 string, and a dimension object as its starting values. Using the setLocation function, which will be called from the MainManager every 10 seconds or so (a fixed time), the current PixelPoint and currentGpsPoint is set together with the currentPixelView, which will be later sent to MapCanvas to draw. The gpsToPixel class is used to convert GPS coordinates which are obtained to Pixel coordinates, which is used to display graphics on the screen. The MapManager keeps all the submaps for the current location in a vector called mapStack. When the current location changes, MapManager uses is_In functions to determine whether the current map is still valid. If it is not, and an adjacent map needs to be displayed, it traverses up the stack until the is_In function returns true, then uses the findSubmap function to find the smallest available map for the current location.

MapCanvas handles the display of the map. MapCanvas takes in an image of a map, draws a portion of it and displays it as our current viewable map. It also handles the size of the user location dot.

LogicalMap's main function is to take in a map file, parse it, and load up the appropriate Submap (*.sub) and Hotspot (*.hot) files.

The HotSpot class' job is to just return the hotspot ID and the coordinates of the hotspot to LogicalMap.

The StrippedSubMap class is similar to the HotSpot class, but instead returns the coordinates of the Submap and the name of the Submap found within the coordinates of the current map. The coordinates mentioned are all global (GPS). In addition, LogicalMap provides is_In and findSubMap functions for MapManager to call when searching for the smallest available map to display.

CONTENT MANAGER

ContentManager receives a list of URLs from ContentClient, which could be better named "ContentServiceConnector." ContentClient uses SOAP to communicate with the ContentService. ContentService is basically a Java database accessor with a bunch of getter functions that are called by the getter functions of ContentClient.

HAWKTOUR CLIENT

The HawkTourClient handles the rendering of the main window, the user interface which we see when we run the application. It makes calls to MainManager and MapCanvas. It generates the map window (zooming/scrolling buttons and main map display), and menu options (buttons).

HARDWARE SERVICE

The hardware section of HawkTour is new as of this semester (Fall 2004), and so far it supports Bluetooth functionality. With Bluetooth, the application is able to communicate with DMRs (Digital Media Receiver). A DMR is a Bluetooth enabled PC whose video output goes to a large-screen TV or HDTV and digital media is stored on it. The DMR is called upon by the Bluetooth client (i.e. HawkTour running on a tablet PC) to start displaying pictures or video. This is done when both devices, the tablet and the DMR, discover each other through Bluetooth when in near proximity.

In the HardwareService diagram above, the Bluetooth functionality is divided in two: the client and the server side. On the client side, the GUI / Client is implemented in a "SampleClient" (a generic implementation written by a generic developer who wishes to use the Bluetooth function), which extends the DisplayClientAdapter. This setup allows the developer to implement just a subset of the methods in DisplayClient.

The SampleClient receives events to display to the screen through the DisplayClientListener, which defines the events that are received from the DisplayClient, and sends messages over to the DMR through the DisplayClientAdapter. This way, SampleClient and DisplayClientAdapter do not need to make constructors for each other, and message passing is simpler. The user (generic developer) never needs to know what is behind the DisplayClient. Similarly, the DisplayClient receives messages through the BTClientXface from BTClient, and sends messages to BTClient.

The BTClient, responsible for device detection, device info, and device connection parameters, is the lowest level that the application transmits the messages before they actually enter the Bluetooth hardware. The Bluetooth Tx / Rx hardware is dealt with by the Java Native Interface and the Win32 API.

On the right side is the DisplayServer class, which is called by SampleServer, or any other class that needs to use it, and DisplayServer will return data from the BTServer class. Again, the user will never need to know what is behind the DisplayServer class

when writing an application like SampleServer. DisplayServer provides the functionality needed to create a server by encapsulating Bluetooth functionality and message passing. DisplayServer provides events through the interface DisplayServerListener. DisplayServerListener defines the events that are received from the DisplayServer.