ActiveCampus - Sustaining Educational Communities through Mobile Technology*

William G. Griswold, Robert Boyer, Steven W. Brown, Tan Minh Truong,
Ezekiel Bhasker, Gregory R. Jay, R. Benjamin Shapiro

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114
{wgg,rboyer,sbrown,mtruong,ebhasker,gjay,rshapiro}@cs.ucsd.edu

Abstract. The traditional university campus is designed to foster a thriving community of learners, but modernity has introduced many stresses. Mobile computing holds the potential to strengthen a campus's traditional institutions of community through a process of indirect mediation. This paper introduces ActiveCampus, a suite of personal services for sustaining an educational community. The design and implementation of ActiveCampus is described, which is now in deployment on the UCSD campus.

1 Introduction

With the arrival of the baby boomers' children, the University of California, San Diego is quickly growing from an intimate small town into a bustling city full of unfamiliar faces. Building proceeds apace, with dozens of departments and hundreds of labs and institutes finding homes in odd corners of undistinguished buildings, old and new. This rapid growth has brought numerous "big city" stresses. It is hard to keep up with the building on campus and who occupies what building. Unfamiliar faces are everywhere, even obscuring those that you know. With growing diversity and the inability to build as fast as people arrive, more students work and live off campus, making them visitors to their own campus and education. One third of undergraduates transfer to UCSD after two years at another college, abbreviating their campus experience.

These changes strike at the heart the campus's mission of learning—research and education. With this mission comes a culture that believes in the power of knowledge to transform people and the world in the most positive way. The university campus, as originally conceived, was a place to nurture those values and pass them on to the next generation, a kind of perpetually rejuvenating cocoon. In becoming a city of towns, and perhaps a city of visitors, UCSD could lose its transformative powers—or it could magnify them. Magnification will require new ways for people to stay in touch with old colleagues, meet new ones, and stay aware of the exciting opportunities around them.

While the campus administration pursues new policies and institutions to keep our community strong, the ActiveCampus project is exploring the use of technology to meet this challenge. With assistance from Hewlett-Packard, we have given HP Jornada 548

* This work is supported in part by a gift from HP, support from the California Institute for Telecommunications and Information Technology (Cal-(IT)^2), NSF Grant IIS-9873156, and the ActiveWeb project, funded by NSF Research Infrastructure Grant 9802219.

PDA's with 802.11b wireless to 500 undergraduates studying computing at UCSD. With UCSD's wide deployment of 802.11b access, we are able to explore research questions in sustaining educational communities through mobile computing.

Sustaining dispersed communities through virtual spaces is well known [Rhe00]. Explicit support of physical communities is seen only in the discourse enabled by E-Graffiti [BG01,BGar] and GeoNotes [EPS+01], where users can leave their electronic thoughts in physical space for those who follow (See Section 3.2). These projects provide a compelling application and warn of the need for a large community and sufficient content to be successful.

Our approach is a variant on a familiar theme[NIH01,OS00,LKAA96,PBC+01,DCME01]: if you and every person on campus carried a mobile, wirelessly connected device, then it could be used as a kind of "x-ray glasses" onto your immediate vicinity that would let you see through the crowds and undistinguished buildings to reveal nearby friends, potential colleagues, departments, labs, and interesting events. By making the clutter transparent and highlighting otherwise invisible things, the confusing bustle of the campus becomes more sensible and within reach.

A simple realization of this idea is shown in Figure 1. The large area is a map of a person's immediate vicinity, as detected through some geolocation method. Overlaid are links showing the location of nearby departments and friends. Department links and the like can be followed to bring up a web page. A nearby colleague, formerly no more available for lunch than a hundred others, is seen to be in the vicinity and can be instantly messaged or found on foot. Any place or entity can be tagged with digital graffiti, supporting contextual, asynchronous discourse.

ActiveCampus is a working system available for use by everyone at UCSD. It is in active use by our group and is being deployed in stages to our existing base of Jornada users for beta testing. All the functionalities described in this paper are operational unless otherwise stated. As part of a broader project using the campus as a living laboratory, researchers in the department of Communication are conducting ethnographies to understand ActiveCampus's impact on campus life.

Several challenges had to be overcome to implement ActiveCampus, and more yet must be overcome to realize our goal of sustaining our educational community. Because ActiveCampus is used in dynamic settings, the information it displays must be easily grasped and acted upon, despite limited display space and input options on PDA devices. Likewise, ActiveCampus must make minimal use of the client's computing resources, which should remain dedicated to the user's primary tasks. ActiveCampus must be readily adaptable and extensible, as the campus setting is continually evolving. Moreover, with UCSD being a research setting, ActiveCampus must enable experiments in technology-sustained community. ActiveCampus gains its value from the number of active users, thus sustainability (e.g., affordability, political viability) is essential to success. In the same vein, scalability of the technology is vital to supporting UCSD's large and growing community.

This paper makes four contributions. First, it identifies a set of sociological issues and places them in a conceptual framework that clarifies how technology can contribute. Second, it defines a base set of services necessary to sustain a community through mobile computing. Third, this definition is complemented by a particular design and imple-

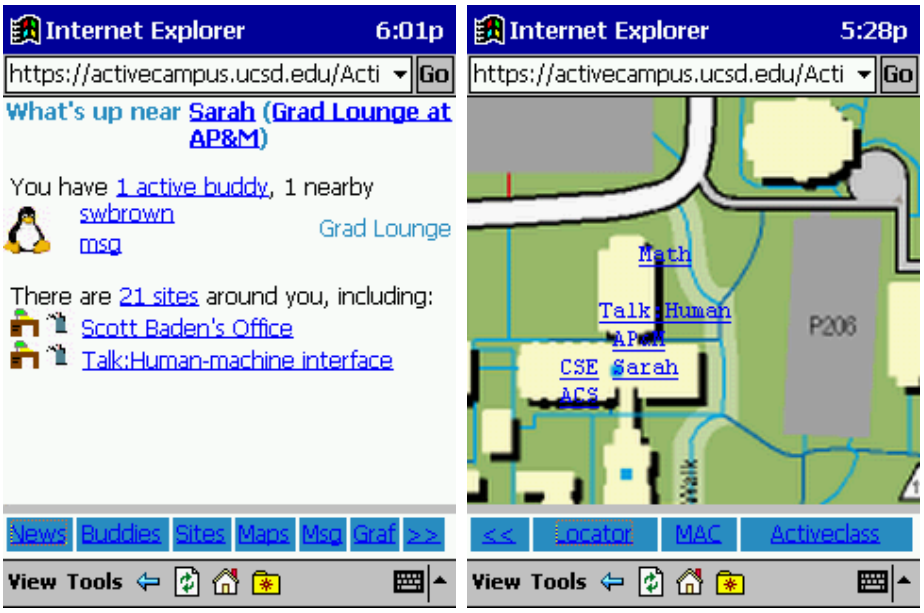

Fig. 1. The What's Up and Map pages of ActiveCampus, as shown on an HP 548 Jornada. What's Up displays key information like the arrival of new messages and the user's location, as well as highlighting randomly selected buddies and sites in the user's vicinity. Map shows an outdoor or indoor map of the user's vicinity, with buddies, sites and activies overlaid as links at their location. Separate pages, reached by the navigation bar or clicking embedded links, show lists of sites, graffiti, and buddies, and performing operations on them.

mentation, and thus provides preliminary solutions to a number of technical challenges we encountered. These challenges include balancing our community's needs for openness and privacy, enhancement of services, scalable spatial querying, and inexpensive geolocation. Last, we lay out a number of open problems that must be resolved, along with our current thinking on these issues.

2 Theory and Requirements

Learning activities of all kinds are heavily mediated (assisted) by a university campus through its structural configuration and its institutions.[1] First, the campus organization itself brings people with complementary interests into close proximity, easing communication. The campus not only brings learners and teachers together, but also concentrates area specialists by organizing the campus into schools and departments of expertise (such as schools of Engineering and departments of Computer Science). A department is not just an aggregation of interest, but is a full-blown institution providing services for its aggregate of people, including working spaces, meeting spaces,

[1] Here, we interpret institution broadly, including entities like departments, libraries, seminar series, and even people. The notions of mediated learning described herein are informed by the work of Michael Cole [Col96].

seminars, opportunities for chance interaction, equipment, curricula, degree programs, funding, etc., to enable and encourage the processes of learning.

Because these institutions operate through proximity, they function less well when people are not "there" on a full-time or full-attention basis. Moreover, it can take considerable time for someone to internalize the workings—the culture—of an institution. If someone does not know the internal workings of an institution (for example, how talks are scheduled and where they normally occur) its mediating power is lost on them, and indeed possibilities are disguised (when it is possible to drop in for a talk). When such obfuscation is combined with a busy schedule, conflicting priorities, distractions, interruptions (half of UCSD undergraduates possess cell phones), it is not surprising that many opportunities are missed. Further complicating matters is that many campus institutional structures crosscut each other, creating ambiguity, but also richness. For example, UCSD is divided into residential college neighborhoods. Each department sits in a college neighborhood and is nuanced by it, but does not belong to that college; it belongs to a school. Each faculty member belongs to a college, however, and of course a department.

We hypothesize that mobile computing applications, by mediating the institutional mediation of learning, can accelerate one's on-going acclimation process, thereby mitigating time and attention deficit. In such a role, ActiveCampus is not a replacement or proxy for extant institutions, but rather a facilitator. Such a role fits mobile devices well, given (on the negative side) their limited form factor, interface, and computing power, as well as (on the positive side) their mobility and relative unobtrusiveness.

Building on the concept that a campus is an organization of institutions for mediating learning, it is natural to consider reifying (displaying) contextual information about (a) you (the learner), (b) mediating institutions, and (c) the sources of learning enabled by those institutions such as a professor, friend, book, event, or another institution like a lab. Since a campus institution is typically a physically aggregated entity, displaying an institution in a *transparent* form and showing its mediated sources of learning "inside" it (or even next to it) is a natural way to convey mediating relationships. Depending on the possible relationships between the learner and the learning source (including role reversal), participants may need the ability to talk through walls as well as see through them. Gradually, then, through experience, a participant learns to parametrically associate the institution with learning sources, imbuing the institution with its full power.

There are many research efforts on augmenting the physical world with information from virtual spaces, albeit without an explicit focus on communities, culture, and learning. At ATT Research Cambridge, users wear goggles which overlay information to enhance their knowledge of what they are already seeing [NIH01]. Hippie [OS00], CyberGuide [LKAA96], GUIDE [DCME01], and a host of other electronic tour guides provide information for the user about the local surroundings using a mapping metaphor to abstract the world, making physical boundaries transparent, and thereby expanding the horizons of the user. These interfaces typically include links to allow the user to drill down for more detailed information. HP's Cooltown [PBC+01] creates a web presence for people, places, and things to support users as they go about their everyday tasks. IR Beacons, RF ID tags, and bar codes are used to identify elements in the environment.

3 ActiveCampus Scenario

3.1 A Day with Sarah

Sarah, a UCSD computer engineering sophomore who transferred from Mesa Community College last quarter, walks out of her morning Engineering 53 lecture, introduction to electrical engineering. This isn't what I signed up for, she's thinking, wondering where was the engineering her Dad had told her about—building things that improved people's lives? Flipping open her PDA, ActiveCampus shows a map of her vicinity, and she sees a link to a talk with "human" in the title (Figure 1). Clicking through, she sees there's a talk just starting in the engineering building on the human-machine interface. Curious, she decides to go. Although the talk gets technical quickly, the introduction has shown her a link between people and computer engineering. Seeing her teaching assistant Mark from class there, but too shy to speak to him, she adds him to her ActiveCampus buddy list by typing in his UCSD e-mail name (found in her class syllabus), with the note "53 student wants to talk about human machine interface." Later, Mark will notice that Sarah is pending on his buddy list and decide, what the heck, add her.

Sarah realizes she's hungry, so she heads to the Price Center for some lunch. Her usual table of friends is probably gone by now. Really wanting to talk to someone about adjusting to UCSD and her major, she checks ActiveCampus and sees that her "buddy" Brad is nearby, clicks on him and sends him a "Wanna eat?" with a couple of clicks (Figure 2). Brad notices the "dome" on his PDA flashing,[2] and flips it open to see that Sarah has sent him a message and is nearby. Now both looking for each other, they see each other through the lines of people and sit down to talk about their day.

After lunch, Sarah decides to go to the library to get a head start on her Engineering 53 homework. Looking at ActiveCampus, she sees a "concert" link on the library, which seems odd. She clicks on the link and finds that there is a "short attention span" concert going on in the basement. It might already be over, but it's right there, so she decides to peek in. It turns out to be pretty cool, a group of graduate students performing Tchaikovsky's 1812 Overture on toy pianos and drums, now moving toward the finale. This isn't what she thought UCSD would be like, but she isn't complaining.

At the end, with everyone laughing and applauding, she heads over to the engineering library wing, also in the basement, to study. After struggling through a few problems, she notices the dome on her PDA flashing; Mark has added her as a buddy. Maybe the next time she goes to the Price Center for lunch she'll see him on her PDA and message him to meet and talk.

Later, leaving the library, she notices that the tree outside the library is not dead, as she'd thought—it's made out of metal and talking quietly. That's so weird. Flipping open her PDA, she clicks over to the digital graffiti page of ActiveCampus, since a friend told her there was lots of arts stuff in there (by default graffiti is not shown on the map since it can clutter). There is a list of graffiti that's been "tagged" in the area, including a "living dead tree" link near the top (Figure 2). Clicking on different parts of the tree leads to different parts of an interactive artwork. Clicking on the tree's roots leads to a story about the tree, pointing her to other talking trees on campus, and gives the lowdown on UCSD's Stuart art collection. Now she begins to understand all the weird stuff she'd been seeing on campus! Clicking on the spray can to the left

[2] The flashing dome feature has been prototyped but is not yet deployed.

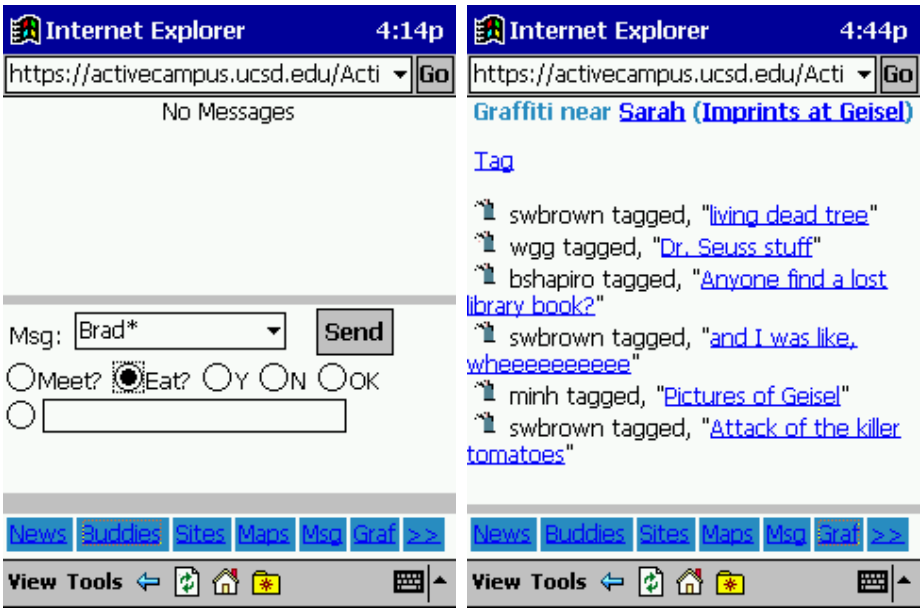

Fig. 2. The Messages and Graffiti pages of ActiveCampus.

of the graffiti's subject line, she is taken to a page where she "tags" the interactive tree with a "Thanks tree!" note to be seen by others who view the living dead tree via ActiveCampus. Walking off, she thinks, huh, I wonder if there is a role for art in engineering? She'd have to ask Mark about that.

3.2 Discussion

Sarah's day reveals several crucial properties of ActiveCampus. The most notable is that it helps campus denizens see through the unintentional barriers created by institutions. Sarah can see that there is a talk starting nearby, even though it was only officially disseminated to the campus via posters in engineering building hallways. Even if she had seen these posters earlier, it would not have been in the context of her frustrating day and probably long forgotten. Seeing a talk with "human" in the title, and in an engineering building, was her cue that this talk might be especially relevant to her. This is a function of mediation—the particulars in the scope of the general establish a context for meaningful interpretation.

ActiveCampus has similar, but not identical, benefits at the Price Center food court and library. In the Price Center, the mere concentration of people is the barrier created by the institution, but the context is eating, which implies relatively unstructured time—a friend of hers at the Price Center is probably free to chat. At the library, the barrier is both physical and conceptual. It is a huge structure with many things going on, and it is counterintuitive to look for music here. On the positive side, the study spaces of the library are what brought her there in the first place. ActiveCampus merely provided the "final mile" solution, timely and contextualized information about the concert.

Colleague Interactions. Sarah's use of the buddy features are indicative of ActiveCampus's facilitator role. After helping her notice that her friend Brad was nearby, she used messaging and his displayed location to purposefully track him down to share lunch. If many friends were nearby, she could have messaged all nearby or active buddies in one action to speed the process. In this way, Sarah is using ActiveCampus to maintain and even develop her social network in a chaotic context. ActiveCampus's messaging does not replace e-mail or instant chat, but incorporates ideas from each to create a service that can unobtrusively help arrange meetings in the physical world. There are no pop-ups (they are disruptive) or conversational threads (if you can't remember the context for a message in an on-going conversation, the opportunity for meeting is likely passed), and simple meeting-directed messages can be sent with a couple of clicks.

Revealing one's location on ActiveCampus could lead to unwanted interactions. Thus, before Sarah could see Mark on her PDA (or vice versa), both she and Mark had to add each other as buddies. We call this the *mutual acceptance policy*. This policy is simultaneously effortless and effective for participants. Neither person has to explicitly handle a subscription request or block such requests (e.g., as spam). If a buddy becomes unwanted at a later time, they are removed by dropping their name from the buddy list. In an ad hoc community, it might be hard to buddy-up with such a method. At UCSD Sarah can use Mark's campus e-mail name to add him to her list. She didn't have to ask him for his ActiveCampus ID or exchange contacts with someone else. In fact, UCSD has a "finger" service that maps names to e-mail names.

Digital Graffiti. Sarah used digital graffiti to answer the question "What is this tree?" because there was no official link for the tree. Consequently, she found out not only what the tree was, but what other people thought about it. This is beneficial to Sarah because she is discovering that this is not just a campus of busy, stuffy professors lecturing to quietly listening undergraduates, but a place where people just a bit "ahead" of her are participating in the campus's academic life. Thus, as with discovering the talk, Sarah has—conceptually—seen through the walls of an art studio to see the campus in action. In actually posting her own graffiti, Sarah has taken an important step from being a passive visitor to a campus citizen. And perhaps the visual arts graduate student who created the "living dead tree" will see Sarah's posting and know that her work is having an impact. (A planned feature of digital graffiti is to get a notification when someone tags graffiti that interests you.) A conversation has been started, perhaps growing into a discursive subcommunity of campus life where new friends will be made through their affinity for this topic.

Many of digital graffiti's possibilities are not revealed in Sarah's day. Any ActiveCampus entity can be tagged: a static object such as a restaurant (e.g., "Get the ham sandwich, it's great!"), physical location (e.g., someone's favorite spot from which to watch a sunset), transient object (e.g., a buddy), or other graffiti. Through artistic expressions, political debates, and the like, graffiti can become a record of campus life. For example, a student might learn what others thought about recent concerts held at a campus venue, find links to band web sites, etc.

With digital graffiti's potential comes debate about whether participants should be able to tag whatever they want. Consistent with the values of an academic commu-

nity, we have favored free speech, including anonymous tagging. However, we must be sensitive to the privacy and safety of members of our community; this could require removing some graffiti. A related concern is how participants might avoid expressions that they find offensive.

4 Software Design and Implementation

Several hurdles had to be overcome to achieve a sustainable, scalable implementation that would not burden client devices with computations that would crowd out the user's primary PDA applications. An additional software requirement is adaptability, which is essential to permit the addition of improved services to ActiveCampus as we learn more about our community's evolving needs. The challenge is that each contributor (e.g., an HCI researcher) has specific goals and skills, and wishes to leverage the ActiveCampus infrastructure without having to understand all of its details (e.g., database programming).

Several high-level design decisions put us on the path to a sustainable, scalable, and adaptable design. First, we chose a client-server model because of the maturity of the technology. Also, having little control over what client devices might appear in our environment, a server-based strategy permits hosting as much computation on the server as necessitated by the client. Along these lines, we decided on a simple web-server-based architecture. For purposes of sustainability, we chose the Apache web server, MySQL database, and PHP for the technologies. Because these are free, portable, and widely known components, ActiveCampus should be easily movable to another environment. The default client interface is rendering to HTML, permitting essentially any device to connect to our server.

More generally, we've made choices that ensure the burden—whether computational or technological—can be shifted to the server, but still allow the client to take control if desired. At this time, the client need only run a web browser and a lightweight application for communicating received wireless radio signal strengths (See Section 4.3). Except for the browser, such client applications are optional, only increasing convenience.

The software design is comprised of two principal kinds of components. *Services* are datatypes that provide unique kinds of function. The root datatype in ActiveCampus is the *site*, which is an entity with a name, location in space, and a default action-on-select. A service is typically represented by a database table of entities (e.g., sites). Many of the tables in ActiveCampus are subtypes of site with more elaborated behaviors, for example users and digital graffiti. *Clients* are viewers on (sets of) services, typically providing a GUI interface to the service. Clients are also frequently subtyped according to the display protocol they satisfy. For example all the HTML client panes are a subtype of HTML client, ensuring that they fit a certain form factor, eschew certain HTML features, and adhere to a particular formatting and color scheme.

Services and clients are decoupled using a mediator pattern [GHVJ95]. The bulk of a client—the nuts and bolts of interaction and rendering—are prohibited from directly calling services. Instead a third component, which sits on top of the two, makes calls to one or more services to retrieve data, and then makes calls to the client nuts-and-bolts to render it. This permits, for example, extensive modifications to a service without

impacting the details of the client implementation. It also permits dramatic changes to the client implementation without having knowledge of the service. Such decoupling is especially advantageous when specialists desire to make modifications and don't have the time or inclination to master details from another domain.

Separation could be further increased by enabling clients to "discover" new site services and clients to discover new subclients, thereby auto-configuring to what is available without intervention from a programmer. These behaviors could be achieved by a simple reflective mechanism, essentially a broker [MZ95,Arn01], in which site subtype services and sub-clients would be registered. In our current centralized implementation, reflection could also be provided by adding database tables of registered site and client subtypes. Although discovery is not yet implemented, we have refined our architecture (described in the next subsection) so that it can make maximum use of such features.

Within this framework, there are two principal ways in which functionality can be added to ActiveCampus. First is adding a new service, for example digital graffiti. Implementation of the graffiti service is simplified by being a subtype of the existing site service, as much of its behavior is inherited from generic sites. Second is adding a new client, like a graffiti viewer. Since graffiti is a subtype of site (e.g., it adds authorship properties), any existing generic site viewer can can display it. Consequently, a new service does not require a new client. However, there may be much graffiti, so users may dislike seeing it in existing clients and demand a specialized client. Even if such a client proves necessary, existing clients can stand in for the interim, permitting recruitment of specialists and development to be staged over time.

Once these global parameters were set, several additional problems had to be solved in different parts of the system. These include enabling services to evolved over time, achieving fast spatial queries, and simple-but-accurate geolocation. Each of these is discussed in turn.

4.1 Service Interface Design

When services are added or evolved, service and client designers wish to see those enhancements automatically manifested in clients. If changes to clients are required, the improvements are at best delayed until client designers can be recruited to make the change. The question, then, is how to design the client–service interface so that most service improvements can be automatically manifested in the client. Our solution is to effectively hide the current limitations of the service by the addition of a small but carefully designed middleware layer to the simple mediator architecture described above [HGM00].

Consider the initial design of our software interface to the digital graffiti service (simplified for presentation), derived by identifying "generic" operations that the graffiti client would require (Figure 3a).

This interface is computationally serviceable, yet there are problems with the client calling it directly. First, it assumes that the only way to retrieve sites on to possibly tag is by current location. This is the current way of identifying sites in ActiveCampus, but will not remain the only one. Second, it explicitly identifies the two ways that a new graffito could be posted, by naming a location or a site. However, ActiveCampus may later provide other modes of posting, such as "roving graffiti". Finally, by using the site notion, this interface is specific to ActiveCampus, complicating the porting of any client

get-current-location()	get-annotation-modes()
get-nearby-sites()	get-annotatables(annotation-mode)
post-graffiti-on(title, text, site, authorship)	get-authorship-modes()
post-graffiti-at(title, text, loc, authorship)	create-graffiti(title, text, authorship)
	get-posting-modes()
	post-graffiti(graffiti, annotatable, post-mode)

Fig. 3. (a) Initial digital graffiti service interface. (b) The adapter abstraction.

that uses it to another application. Graffiti do not intrinsically require being attached to a site. They could attach to any "annotatable" entity, like a page in an on-line book.

Rather than reimplement this interface, we abstract it away, in effect by applying the adapter pattern, having any client call the adapter rather than the original, underlying interface [GHVJ95]. Treating the graffiti service interface as a strawman for the adapter, we can appreciate what the adapter abstraction achieves in its place.

The first step in resolving the above problems is to make a simple (if not previously obvious) naming change in the adapter interface: rather than saying "site", say "annotatable". Next is to identify the range of behaviors of an annotatable entity, rather than a site. For cases where there are many possible variants (e.g., many modes of posting), it is likely that the current version of the underlying service provides only a subset of these variants. Consequently, the client needs to *query* the adapter about what modes are present. Of course, the ActiveCampus designers had no notion of "graffiti posting modes", much less that it should publish them as such. Thus, the adapter takes on the responsibility to provide the query capability. Since the adapter is written, logically speaking, *after* the service to glue the client and service together, the query capability is not difficult to write. The adapter implementer inspects the interface of the service, notes the range of "annotatable" subservices, and then writes a query function that reports these subservices when called.[3] Our convention has been to return strings that suggestively name the modes. These strings can be used as items in client pull-down menus and passed back to the adapter to drive logic that invokes the appropriate operations in the service interface. This approach resulted in the adapter interface shown in Figure 3b.

In the above design, the client designer (an HCI student familiar with GUI technologies) can now freely code to the adapter interface, while his partner (a CS student) works out the details of implementing the adapter and extending the service to provide more of the desired features. Moreover, development below the client can be incrementalized: First basic modes can be implemented, then advanced modes attempted, all without requiring further client changes.

4.2 Database Performance

Spatial Data Structures in SQL. ActiveCampus needs to locate nearby entities and determine if one entity (e.g., the user) is contained in another (a building with a map). Recently, PostgreSQL added efficient support for spatial queries over geometric data using R-trees [Gut84]. For reasons of sustainability—both cost and compatibility with ex-

[3] When our core architecture is extended to support discovery of site subtypes, the adapter could use discovery for some reflective queries.

isting computing environments—we sought efficient implementations of these queries that would be portable to any SQL-compatible database.

Tree traversals are not efficient on a standard relational database because each traversal step in the tree requires a unique communication with the database. Our goal, then, was to convert each of our relatively simple spatial queries into a simple relational query. Our approach is to partition the coordinate space into quadrants, giving each a name that can be used in a query for fast retrieval. The name is constructed by concatenating the X and Y coordinates each divided by the grid resolution that forms our conceptual grid in 2 dimensions. Every table containing spatial data is expanded with columns to store these indices.

With this grid system, a 'nearby' query can be performed by the PHP layer by (a) calculating a set of grid squares that intersect the region defining 'near' (simple math can generate grid square names adjacent to a central square), (b) requesting their data from the database (e.g., sites in each queried grid square), and (c) trimming data from the peripheral grid squares that are outside the query region. Step (c) is optional when the data is intended to give just a sense of the user's context. The list of grid square names form a disjunctive query predicate of the form index = '2660x4456' or index = '2760x4456' or ..., so all potentially contained data can be returned with a single request to the database. Because the radius of a query may vary widely depending on whether the user is indoors or outdoors, etc., we created indices at multiple resolutions to avoid inefficient queries that included either hundreds of squares or just one square with excessive coverage.

Using this same method, we also implemented a simple, fast "conceptual location" query by building a fine-grained 3-D grid in which each cube maps to a single campus site name. Cubes that have no site are filled by a nearest adjacent site.

Progressive Query API. There are many steps to choosing the best map for a given location in ActiveCampus. First, an intersection test with a two-dimensional point gives all the maps that contain the point of interest. Then, the set of maps are filtered to pick the ideal pixel resolution, zoom level, height, and many other characteristics. Because the map display application only needs the best map for a point, originally all the filtering steps were performed in one function. However, more needs began to arise, such as applications that check what zooms are available for a point or to filter out everything but floorplan maps early in the selection process. There are naive solutions that are either slow (perform a sequence of queries), or complex and unstable (writing a slightly different function for each need). We designed a progressive API in which each selection or querying step appears as a single function call, but in fact each call is merely folding its step into a large as-yet-unperformed query. The query is performed only when the contents of the result are referenced outside the API.

4.3 Location Detection

For users to see appropriate contextual information, their location as well as their colleagues' needs to be computed with some accuracy. With a query radius of a hundred feet or more, errors of thirty feet in the horizontal plane will still retrieve the appropriate map and display the appropriate sites and colleagues. More problematic are vertical errors indoors, where an error of just eight feet can result in displaying the wrong map and sites. As discussed below, there are a surprising array of barriers to achieving this goal,

both theoretical and pragmatic, both essential and accidental. We use a combination of methods to achieve a usable solution, including user adjustments to the computed floor via simple clicking.

There has been much recent research on location detection. Indoor location positioning research includes systems like Active Badge [WHFG92], Active Bat [ACH+01], and PinPoint [Tec02], which require the user to wear a transmitter that periodically emits a pulse picked up by a grid of receivers whose positions are known and computes the RF time of flight to determine position. SCADDS also uses acoustic ranging [HSI+01]. RADAR uses IEEE 802.11b to develop a signal strength map of a building and uses a nearest neighbor algorithm on actual signal strengths to determine a user's position [BP00]. The Ad Hoc Positioning System makes use of the number of hops for a message to get to receiving stations whose positions are known [NN01]. The SmartHome uses multi-modal sensing (optical, audio, mobile, embedded, and other sensors) and data fusion to detect a persons location [San00,Ess00]. Hightower and Boriello provide an in-depth survey in provides a taxonomy of location systems for mobile-computing applications [HB01].

Our sustainability goal requires a geolocation method that exploits properties of our environment that are readily available to us (i.e., cost nothing), precluding the use of beacons and extensive manual mapping of actual signal strengths, for example. Indeed, our goal to support any device requires us to provide manual geolocation as a fallback.

In an 802.11b environment, it is natural to exploit access point (AP) signal strengths as perceived by a device. Using the received signal strengths and the known locations of the perceived AP's, it is possible to estimate a device's location through a triangulation process.

Although the software interfaces to acquire signal strengths are not always published, so far we've been able to acquire the required access through vendor agreements or use of logging services provided with the wireless card driver. More recently, Windows XP has published its WMI interface, making received signal strength available as an application-level service. However, we typically run into strange problems with these interfaces—values are clipped or cached, clearly visible AP's are not reported, etc. Our experience with the Symbol Wireless Networker for CompactFlash, a well-built device, is not unusual. It reports abstract received signal strength values in the range 1 to 31; in mapping these to decibels, we discovered that the value 31 stands in for all values above -44 dbm, clipping some 25dbm of dynamic range.

Estimating Distance from an Access Point. With triangulation, the received signal strengths are converted into distances via a formula. 802.11b operates in the 2.5GHz radio band, which are readily attenuated by line-of-site obstructions, and sometimes reflected [Beu00]. Theoretical and empirical models derive very similar results in a well-behaved environment, but behave badly under attenuation. This empirically derived one is typical:

$$d = m * e^{\frac{\bar{S}+17.027}{c*-11.901}} \quad (1)$$

where $\bar{S}$ is signal strength in dbm,[4] and m and c are both 1. However, even a small negative error in a weak dbm value (a large negative number) generates a large absolute change in estimated distance, resulting in a poor location estimate. However, increasing c to 1.5 and m to 2 results in a much better behaved formula. Although it could chronically underestimate distance, it tends not to due to obstructions, and the absolute errors are small compared to the base formula. Even linear formulae have provided a good statistical fit to empirical data, but work less well in some outdoor settings where there are fewer obstructions.

Triangulation. Our first attempts at triangulation applied direct methods of computation. These proved unstable (distance errors due to extreme signal attenuation would yield poor estimates) and potentially computationally expensive (quadratic time) in the presence of a large number of AP's. Coping with the latter by picking a subset of AP's only diminishes stability.

In looking at the details of our results, two things became clear. First, the placement of the AP's in our environment are optimized for coverage and simplicity of administration. In particular, most AP's were arranged in a plane, meaning that accurate three-dimensional estimates were essentially impossible and the power of the direct methods was at least going to waste. Second, as observed above, the weaker a signal was, the less we could depend on the accuracy of the computed distance. This meant that methods that worked directly with distances and treated them all as equally valid would be prone to error.

These problems led us to consider alternate methods that would (a) scale gracefully to a large number of AP's and (b) let us readily incorporate physical and logical properties of the environment in the geolocation computation. A hillclimbing method of geolocation [HWB00], which generates a sequence or set of guesses that are checked against a measure of quality (e.g., minimal error), seemed natural.

Against our criterion of scaling, hillclimbing is advantageous because computing the location is fast (a guess), and computing its error is linear in the number of AP's, a computation of the "distance" between the observed signal strengths and the expected signal strengths computed from the guessed location.

Against our second criterion, hillclimbing allows for the incorporation of properties of the environment by permitting us to bias our guesses towards a likely answer or preemptively eliminating some guesses altogether. Constraining the search space has the additional benefit of reducing the number of guesses, further improving scalability. We currently employ two constraints. First, we restrict guesses to the sphere defined by the expected distance from the AP showing the strongest received signal strength. Second, we further restrict the guesses to the horizontal circles on the sphere that are defined by the expectation that a mobile device is generally a few feet off the floor–neither on the floor or near the ceiling. In particular, the algorithm searches the circles defined by the floor on which the strong AP resides, as well as the floors above and below if the computed distance is great enough to reach them. With these constraints, only a few guesses on each circle are sufficient to produce a plausible set of locations, permitting us to do a simple blind search, rather than a directed one. If there are multiple

[4] dbm is a logarithmic measure of signal strength, with 0 dbm $\equiv 1mW$. For 802.11b, values between -90 and -20 dbm are typical.

guesses on a floor with nearly the same minimal error, their average is taken to compute a "best" location for the floor.

Results and Remediation. This algorithm was tested at a large number of locations inside and outside our computer science building. Using an unweighted average, error was about thirty five feet. This is not surprising, for one, given our inability to detect signal variations in the -20 to -44dbm range on the Symbol wireless card. Yet, accuracy for our typical usage is often high because most of us work or meet near an AP. The strong signal from that AP virtually guarantees accuracy. Accuracy drops substantially for the brief periods that we're moving about, as it is less likely that an AP is close by, thus there is no "good" AP to reliably hillclimb on. Perhaps in cases where no AP is close by, our earlier direct triangulation method would be better, as it does not favor any particular access point.

More worrisome than the raw error in distance is the computation of the correct floor. Even the most modest error in the vertical can locate you on the wrong floor, and the clipping at the top of the Symbol wireless card's range can generate just such a modest error. Worse, even when the PDA or laptop is stationary and there is no apparent motion in the vicinity, the received signal strengths can vary widely over time, causing "floor jumping". (We see this on all card types. It appears to be a property of the environment and not the cards.)

We have devised two techniques to minimize the ill effects of this behavior. One, we average signal strengths over time to damp the apparently random variations. Two, we provide one-click options via our manual geolocator to adjust one's location up or down a floor, and have it stick. Neither of these solutions are perfect. Averaging, for example, slows real changes as well as random variations, and besides the Symbol card driver can provide stale signal strength values in our current implementation. Yet, they are acceptable compromises in our sustainability-oriented environment. Other forms of assisted geolocation might be a promising avenue for future research. Multi-source geolocation in general is also a promising direction [HBB02].

5 Open Problems

In the foregoing we have presented our ideas through a working prototype that is in use by about a dozen members of our group, and is currently in staged deployment for beta test to our 500 undergraduate Jornada users. Many issues remain to be explored and evaluated. This section lays out some open problems and our current thinking on them.

5.1 Fortuitous Discovery

A university campus provides a seemingingly infinite array of new opportunities, and a goal of ActiveCampus is to help students stumble across them. Given the small size of mobile devices and one's limited time, the information glut must be managed, but at the risk of feeling like censorship. One small step we've taken is the What's Up? page, which uses random selection and abstraction to condense without losing the chance for discovery (Figure 1).

Conversely, given participants' needs for retaining control of personal information, how can ActiveCampus help, for example, bring together two "birds of a feather" who are unknown to each other? We are investigating the use of *asynchronous negotiated*

access, which can ease the sharing of information without excessive information exposure [HS00]. For example, consider an extended buddy system that permits you to buddy with *groups* such as "scuba diving". Unlike with personal buddies, information sharing could staged. Another nearby scuba diving person might first see that someone is "nearby" without revealing your location or identity. An anonymous message exchange might lead the originator or you to lift the cloak on location and identity to facilitate a face-to-face meeting.

5.2 Bootstrapping Content

The success of a system like ActiveCampus depends on a rich base of information to keep people coming back for more. Although some features are in place for bootstrapping buddies, what about other content? With the help of UCSD's Facilities Office, which maintains CAD drawings and GIS information for the campus, we have painstakingly entered a base level of campus data for ActiveCampus—all buildings, departments, programs, and 802.11b access points on campus. Just adding the office and laboratory information for our building doubled the number of entries, and there are about 300 other buildings on campus containing thousands of offices and labs. Even UCSD Facilities depends on unit administrators to maintain basic occupancy information, which is evolving all the time. UCSD Libraries has plans for elaborating the Facilities data, but only modestly and the work is just beginning. A related problem is that the natural "owners" of information may wish to suppress its dissemination; asking for permission to publish information adds significant cost.

A natural solution for a community-oriented system is to enroll the community in publishing its content—in other words, let self-interest drive bootstrapping by allowing for a sort of official graffiti. The "official" part is that there is limited space in site lists and the map view to display content, and it is relatively permanent. It needs to be rationed fairly and reliably correct.

To this end, we have prototyped a site-entry client for ActiveCampus users. Location can be specified by navigating on a displayed map, like our manual geolocator. Other data are entered via form input. We envision three complementary methods of ensuring the quality of such content. First is to require administrator approval before posting submitted information. This approach is unappealing because it puts us in an enforcer role and lacks immediate gratification for participants. We could publish unvalidated content as "pending approval", a mechanism that we've used successfully in a room reservation system in our department (admittedly a more tightly knit community). A second approach is to give certain users (e.g., the same people that have "Facilities" database access) privileged submission rights. Our role then would be validation rather than enforcement. We anticipate, however, that many such people will not be motivated to get involved. Third, we've considered letting every active ActiveCampus user post one site—perhaps a web page to a dorm room, laboratory, or office. Rationing both bounds the amount of abuse and encourages wise use. The last two approaches can be combined, giving ActiveCampus users privileged posting rights until they abuse those rights. Since ActiveCampus is an experiment in community and universities promote freedom of expression, we're starting with this last proposal to see what happens. Whether it succeeds, fails, or succeeds after some improvements, we will learn a lot about bootstrapping communities supported by mobile computing.

An orthogonal issue is that the preparation of building maps for entry requires specialized software and skills (e.g., a CAD program and comfort with planar coordinate systems). Fortunately there is at least one order of magnitude fewer maps than other content. Also, one objective of the UCSD Libraries' GIS program is to normalize such content, which would eliminate much of the manual processing.

5.3 Componentization of Crosscutting Aspects

The problems driving software researchers are often necessarily *crosscutting*, touching on portions of several other endeavors. Such an issue is called an *aspect*, denoting both its conceptual cohesiveness and likely physical dispersion. Aspects can compromise componentization, requiring contributors to cope with issues outside their expertise.

An example aspect problem in ActiveCampus was improving the performance of spatial queries, described in Section 4.2. First, the service designers are not the designers of the fast lookup method. The latter team (a) added a quadrant index field to the main sites table, (b) added code and queries to compute the quadrant of each site, and (c) modified existing location-based queries and updates to use their query and update routines. Several modules and layers of the system were affected, creating maintenance problems, as future changes to the indexing mechanism (e.g., addition of caching) will be non-local. Also, service maintainers must now cope with the somewhat tricky, alien, and non-modular performance code. Second, the indexing mechanism should be applied to all site services, including ones added in the future. Repeatedly applying such non-modular changes not only further disperses the indexing aspect, but also requires the implementors of new services to know how to add this crosscutting feature to their service, since the performance developers may be long gone.

Language solutions are appealing [KHH+01,OT00], but our system is multi-language and multi-technology. Analysis and visualization tools can track crosscutting [GYK01,HK01], but they lack the advantage of localizing aspects in the code. We've taken an ad hoc approach for the time-being, using PHP's new object-oriented features to increase the modularity of the spatial queries, and writing scripts to automatically "quad" the site tables. However, there is still considerable aspect distribution in the system. One possibility in the future is to construct a comprehensive "build script" that weaves aspects into the code at the time of deployment, which would add fields to tables, redirects to the web server, programming triggers into the database, and overrides to function/method definitions. The parser-generator tools YACC and CUP are well-known examples of such tools [Joh75].

5.4 Decentralization of the ActiveCampus System

The current ActiveCampus system architecture's strength, and weakness, is its simplicity, easing development and promoting sustainability, but limiting in scalability and local control. For one, units on campus might wish to run their own ActiveCampus servers, providing their own unique services or "coverage" of a particular area of the campus. Such changes would require spinning out or creating discovery services to keep the federation intact.

A less radical change to aid scalability and local control would be to move more computation and state to clients. Moving geolocation to the client is trivial; the client would send back a location rather than signal strengths. It was originally implemented that way, but was moved to the server to enable geolocator improvements without client

software updates. We have also prototyped "thick clients" implemented in VisualBasic and Java that could connect to the ActiveCampus server via XML/RPC. These provide superior look-and-feel and can offload the server by caching data and retrieving incremental changes. Combined with geolocation on the client, it would be possible for the client to recenter the map without any communication with the server. Some updates of the content of the display could also be moved off the server if clients communicated locations amongst each other in a peer-to-peer configuration.

6 Conclusion

UCSD, like many campuses, is adjusting to a growing, changing student body that may feel more connected to off-campus life than on-campus life. Although mobile technology is part of the problem, it can also be part of the solution. Employing the contextual display of campus participants, their institutions, and the learning opportunities enabled by those institutions through the metaphor of transparency, it is possible to make campus life as accessible and compelling as off-campus life.

We have identified a set of complementary services that we believe can achieve this goal. By using the context of location, time, and one's stated colleagues, a display in the form of a map or labelled list helps a participant see opportunities that are within reach and act upon them by physically moving to them or clicking on their links to learn more. When that opportunity is talking to a colleague, click-driven messaging facilitates a meeting. Our colleague subscription method is simple, unobtrusive, and respects privacy. ActiveCampus augments campus discourse through digital graffiti, the ability to virtually "tag" locations, entities, and other graffiti with commentary or artistic expressions.

The relatively seamless integration of different services in a small form factor presents challenges. Establishing the *site*—a campus entity with a name, location, and default action-on-click as a system supertype permits new services to define themselves as subtypes of site. Then, generic displays like our map view can display all (desired) site types in a single display in a natural fashion. The unique features of a service are accessed through a custom display that is made available through a simple navigation bar, or perhaps also through the default action-on-click.

Making such a system viable requires addressing the conflicting requirements of sustainability, adaptability, and scalability. Through careful choices of technology, software architecture, and algorithms, we have succeeded in building a system that is easy to deploy, is easy to adapt, accommodates lowest common denominator devices, and performs well.

We are now deploying ActiveCampus, in stages, to our population of 500 Jornada users, and preparing for ethnographic investigation of its impact on campus life. Many open problems remain, both social and technical. Issues such as bootstrapping community and content, and balancing the needs of chance discovery and information management are just two. We hope that our approach of augmenting existing institutions, rather than replacing them, will help, but it remains to be seen whether our tools will add sufficient value.

Acknowledgements. We thank Jim Hollan for his spiritual and technical guidance on technology-sustained communities. We thank UCSD's Facilities office, in particular

Roger Andersen, Robert Clossin, and Kirk Belles, for their their time, expertise, and resources. Thanks to Ed Lazowska for reading a draft of this paper. We thank David Hutches, Jadine Yee, Justin Lee, Daniel Wittmer, Antje Petzold, Jean Aw, Linchi Tang, Adriene Jenik, and Jason Chen for their assistance on the project. Finally, we thank Intel's Network Equipment Division for donating network processors and Symbol Technologies for their software technical support.

References

[ACH+01] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8):50–56, 2001.

[Arn01] K. Arnold, editor. *The Jini Specifications*. Addison-Wesley, second edition, 2001.

[Beu00] J. Beutel. Geolocation in a picoradio environment. Masters Thesis, 2000.

[BG01] J. Burrell and G. K. Gay. Collectively defining context in a mobile, networked computing environment. In *CHI 2001 Extended Abstracts*, May 2001.

[BGar] J. Burrell and G. K. Gay. E-graffiti: Evaluating real-world use of a context-aware system. *Interacting with Computers*, To appear.

[BP00] P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of IEEE INFOCOM*, volume 2, pages 775–784, March 2000.

[Col96] M. Cole. *Cultural Psychology: A Once and Future Discipline*. Harvard University Press, Cambridge, MA, 1996.

[DCME01] N. Davies, H. Cheverst, K. Mitchell, and A. Efrat. Using and determining location in a context-sensitive tour guide. *IEEE Computer*, 34(8):35–41, 2001.

[EPS+01] F. Espinoza, P. Persson, A. Sandin, H. Nystrom, E. Cacciatore, and M. Bylund. Geonotes: Social and navigational aspects of location-based information systems. In *Ubicomp 2001*, pages 2–17, Berlin, 2001. Springer.

[Ess00] I. A. Essa. Ubiquitous sensing for smart and aware environments. *IEEE Personal Communications*, 2000.

[GHVJ95] E. Gamma, R. Helm, J. Vlissides, and R. E. Johnson. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1995.

[Gut84] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD*, pages 47–57, 1984.

[GYK01] W. G. Griswold, J. J. Yuan, and Y. Kato. Exploiting the map metaphor in a tool for software evolution. In *Proceedings of the 2001 International Conference on Software Engineering*, pages 265–274, May 2001.

[HB01] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[HBB02] Jeffrey Hightower, Barry Brumitt, and Gaetano Borriello. The location stack: A layered model for location in ubiquitous computing. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002)*. IEEE Computer Society Press, June 2002. To appear.

[HGM00] J. Hayes, W. G. Griswold, and S. Moskovics. Component design of retargetable program analysis tools that reuse intermediate representations. In *Proceedings of the 2000 International Conference on Software Engineering (ICSE 2000)*, pages 356–365, Limerick, Ireland, June 2000.

[HK01] J. Hannemann and G. Kiczales. Overcoming the prevalent decomposition in legacy code. In *ICSE 2001 Workshop on Advanced Separation of Concerns*, May 2001.

[HS00] Jim Hollan and Scott Stornetta. Asynchronous negotiated access. In *Proceedings of Human Computer Interaction 2000*, pages 17–26, 2000.

[HSI+01] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the Symposium on Operating Systems Principles*, pages 146–159, Chateau Lake Louise, Banff, Alberta, Canada, October 2001. ACM.

[HWB00] J. Hightower, R. Want, and G. Borriello. SpotON: An indoor 3D location sensing technology based on RF signal strength. Technical Report UW CSE 00-02-02, University of Washington, Department of Computer Science and Engineering, Seattle, WA, February 2000.

[Joh75] S. C. Johnson. Yacc—yet another compiler compiler. Computing Science Technical Report 32, AT&T Bell Laboratories, 1975.

[KHH+01] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An overview of AspectJ. In *15th European Conference on Object-Oriented Programming (ECOOP 2001)*, pages 327–353, June 2001.

[LKAA96] S. Long, R. Kooper, G. D. Abowd, and C. G. Atkeson. Rapid prototyping of mobile context-aware applications: The cyberguide case study. In *Proceedings of the 2nd ACM International Conference on Mobile Computing and Networking (MobiCom'96)*, November 1996.

[MZ95] J. Mowbray and R. Zahavi. *The Essential CORBA*. John Wiley and Sons, 1995.

[NIH01] J. Newman, D. Ingram, and A. Hopper. Augmented reality in a wide area sentient environment. In *Proceedings of the 2nd IEEE and ACM International Symposium on Augmented Reality (ISAR 2001)*, New York, 2001.

[NN01] D. Niculescu and B. Nath. Ad-hoc positioning system. In *Proceedings of IEEE GLOBECOM*, November 2001.

[OS00] R. Oppermann and M. Specht. Context-sensitive nomadic exhibition guide. In *Ubicomp 2000*, pages 127–142, Berlin, 2000. Springer.

[OT00] H. Ossher and P. Tarr. Multi-dimensional separation of concerns and the hyperspace approach. In *Proceedings of the Symposium on Software Architectures and Component Technology: The State of the Art in Software Development*. Kluwer, 2000.

[PBC+01] S. Pradhan, C. Brignone, J. H. Cui, A. McReynolds, and M. T. Smith. Websigns: Hyperlinking physical locations to the web. *IEEE Computer*, 34(8):42–48, 2001.

[Rhe00] H. Rheinhold. *The Virtual Community*. MIT Press, Cambridge, revised edition, 2000.

[San00] J. M. Sanders. Sensing the subtleties of everyday life. *Research Horizons*, Winter, 2000.

[Tec02] RF Technologies. Pinpoint local positioning systems. URL, 2002. See `http://www.rftechnologies.com/pinpoint/`.

[WHFG92] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.