Enabling End-User Specification and Debugging of Complex Events for Location Systems

Evan Welbourne

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington

2010

Program Authorized to Offer Degree: Computer Science & Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Evan Welbourne

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Co-Chairs of the Supervisory Committee:

_____

Magdalena Balazinska

_____

Gaetano Borriello

Reading Committee:

_____

Magdalena Balazinska

_____

Gaetano Borriello

_____

James Fogarty

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Enabling End-User Specification and Debugging of Complex Events for Location Systems

Evan Welbourne

Co-Chairs of the Supervisory Committee:
Professor Magdalena Balazinska
Computer Science & Engineering

Professor Gaetano Borriello
Computer Science & Engineering

While real-time location systems are gaining popularity as a platform for a variety of applications (e.g., asset tracking, safety and security, workflow monitoring and optimization), recent studies have shown that two critical barriers to adoption remain: 1) cost of location sensors, and 2) cost of tuning events in location-aware applications. In this dissertation, we present the Cascadia middleware for specification, detection, and management of location events. We also present a suite of end-user tools which, together with Cascadia, significantly reduce barriers to adoption of real-time location systems.

The Cascadia system leverages recent advances in probabilistic data management to facilitate specification, detection, and management of complex events over sporadic and incomplete location data. In particular, Cascadia extends the Lahar system [212, 273] to detect an important family of location events as patterns over inferred probabilistic database views representing entities' location traces. This approach enables use of less reliable location sensors (e.g., passive RFID) that are 10 times cheaper per tracked entity than state-of-the-art sensors. Clear semantics and a pattern-oriented query language also enable Cascadia to expose an intuitive API for querying and subscribing to location events.

In addition to tools that define metadata, we present the Panoramic tool, which supports end-users in customizing and tuning their applications by directly specifying and debugging location events. Panoramic does not require users to write code, understand complex models, perform elaborate demonstrations, generate test traces, or blindly trust deterministic events. Instead, it allows end-users to specify and edit complex events with a visual language that embodies natural concepts of space and time. It also takes a novel approach to verification in which events are extracted from historical sensor traces and presented with intelligible, hierarchical visualizations.

Our work with Cascadia and Panoramic is grounded in a comprehensive survey of events in location-aware computing as well as a detailed analysis of real location data from experiments with a building-scale RFID system called the RFID Ecosystem. We refine our designs through iterative laboratory studies and conduct a summative evaluation through additional laboratory studies and a longitudinal study with over 60 participants and 100s of location sensors in the RFID Ecosystem.

TABLE OF CONTENTS

Page

LIST OF FIGURES

GLOSSARY

DETECTION-RATE: The probability that an object with an attached RFID tag passing in the vicinity of an RFID antenna is detected at least once.

MAJOR DETECTION FIELD: An arc directly in front of an RFID antenna.

MINOR DETECTION FIELD: The area around an RFID antenna that is just outside the arc that is directly in front of the antenna.

NTP: Network time protocol, a protocol that is designed to syncrhonize clocks of computers over a network.

PVC: Polyvinyl chloride, a thermoplastic polymer often used to encase RFID tags.

READ-RATE: The rate at which an RFID reader can successfully read an RFID tag.

RTLS: Real-time locating system, a system that tracks and locates entities in real-time using some location technology.

TRE: Tag read event, tuples created by RFID Ecosystem RFID readers for later processing. They have the form `(tag ID, antenna ID, start time, number of reads)`.

VSWR: Voltage Standing Wave Ratio, a measure of how well load is impedance matched to a source.

ACKNOWLEDGMENTS

I would first like to thank my advisors Magdalena Balazinska and Gaetano Borriello for their positive outlook, constant encouragement, inspiring enthusiasm, and excellent advising. I would also like to thank Yang Li and James Fogarty for extremely helpful insight and guidance near the beginning and end of my doctoral work respectively.

I am also greatly indebted to my past collaborators for all their excellent work. I'm especially thankful to those I've worked with on projects related to the RFID Ecosystem, including Julie Letchner, Karl Koscher, Nodira Khoussainova, Vibhor Rastogi, Travis Kriplean, Chris Re, and Waylon Brunette.

Finally, I am very grateful to have advised or co-advised so many brilliant undergraduate students! You've given me a fantastic opportunity to explore another side of myself and my work. Specifically, I'd like to thank Sam Raymer, Victoria Kirst, Leilani Battle, Kayla Gould, Kyle Rector, Garret Cole, Julia Schwarz, Doug Copas, Caitlin Lustig, Angela Pai, Leaf Xu Ye, Paramjit Singh Sandhu, Jordan Walke, Andy Sun, Justin Vincent, Patricia Lee, and Robert Spies. Without you the large-scale vision of the RFID Ecosystem would never have become a reality.

DEDICATION

In memory of my old friend Ana Solajic (1978 - 2009).

You were the first to suggest that I attend graduate school,

and without the many long nights we spent studying I never would have made it here.

Chapter 1

INTRODUCTION

For two decades researchers have worked on context-aware computing systems that intelligently adapt their behavior by interpreting data from sensors embedded in the environment [288, 342]. Today, the commercial and cultural mainstream has its first taste of this technology in the hundreds of mobile and real-time web applications like FourSquare [109] and Google Latitude [128] that use primitive location context (e.g., "Fiona is at the park", "There's a Thai restaurant 2 blocks away"). Recent work has also led to research systems that infer complex, higher-level context (e.g. "Grandma is making coffee" [51], "You burned 40 calories by taking the stairs" [210]) using sensor deployments in everything from civil infrastructure to homes and mobile devices. In many ways context-awareness appears to be a looming disruption - indeed, Gartner Research predicts that it will comprise a $12 billion industry as early as 2012 [118]. However, before the next wave of systems leaves the laboratory they must overcome a critical barrier to adoption and effective use: users must be able to understand, trust and control applications that use higher-level context in spite of the growing diversity, uncertainty, scale and social impact of real-world deployments.

In principle, context-aware applications are life-changing because they precisely sense the context of a situation and use it to act on behalf of the user (e.g., send a message, turn off the lights, record an event). In practice, context is inferred over sporadic data streams from diverse and sub-optimally distributed sensors that may be calibrated incorrectly or fail frequently. Moreover, context as perceived by humans may contain elements that are imperceptible to physical sensors (e.g., psychological phenomena). Thus, context is highly uncertain - and for this reason users will not adopt a context-aware system until they can build trust by understanding, evaluating and controlling the context it produces [36, 217]. Unfortunately, existing infrastructures for context-aware computing do little to provide the rich, intelligible context users need. The majority provide no information on how context is

inferred or how it relates to the sensor data. The few that do [20] provide no quantification as to the degree of uncertainty in a particular context item. The result is that as context-aware applications become pervasive, end-users, application designers and even developers will be left frustratingly, and in some cases dangerously out of touch with the systems that mediate their relationship to the world.

This dissertation explores a solution to the problems above by unifying database technology with a middleware for event processing and a set of novel end-user tools. Specifically, we identify requirements through studies of real uncertain sensor streams and context usage in real applications. Based on these requirements we design and implement a middleware, *Cascadia*, for specifying, detecting, and managing context over uncertain sensor data. We also design and implement a suite of tools that allow end-users to interact with Cascadia. Included in this suite is *Panoramic*, a tool for specifying and debugging context for Cascadia to detect. We evaluate Cascadia and Panoramic through user studies with real sensor data from a representative sensor deployment and with non-expert end-users in both laboratory and longitudinal field studies. As a simplifying factor, this work addresses only location data. We argue that exclusion of other sensor data in justified by the fact that *location* is often regarded as the most fundamental and most important element of context. Moreover, location sensors and applications make up the most pervasive and rapidly growing segment of today's context-aware computing market.

Intelligent behavior in location-aware computing is driven by *location events*. Applications detect events by dynamically evaluating spatio-temporal relationships among people, places, and things. For example, a location-aware to-do list might detect simple events like "Alice is near the library" to trigger reminders. In contrast, many new applications for real-time location systems (RTLS) rely on *complex events* that contain sequences of interactions [11, 255]. For example, a hospital workflow tracker may log a "cardiology exam" whenever a patient is detected exiting the hospital after meeting with a cardiologist and then spending time with a nurse and an electrocardiogram machine. We explain the concept of location events, illustrate their utility, and highlight the challenges involved in their use through a specific example below.

1.1 Specific Example: Location-Events in Hospitals

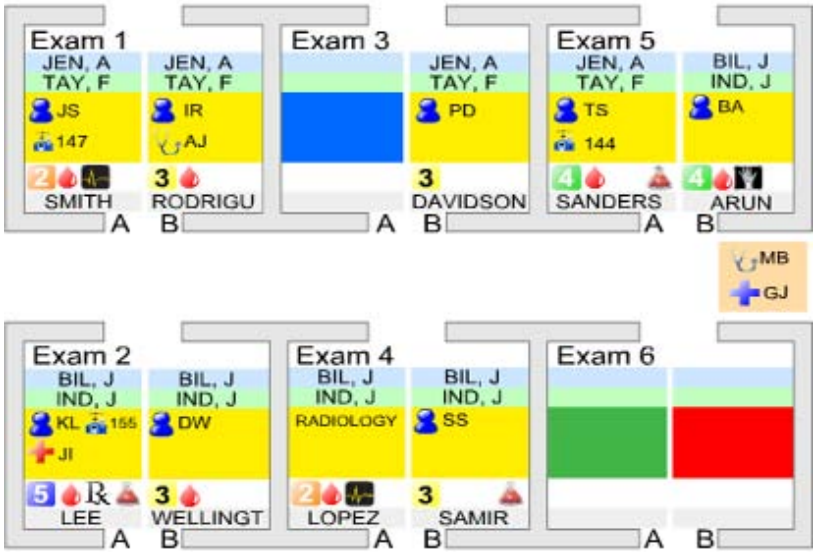
While hundreds of location-aware applications have been created in research projects as well as for consumer and enterprise markets, most share the same high-level information needs, basic functionality, and technology challenges. In this section, we illustrate the benefits and challenges of location events and technologies by examining a specific, state-of-the-art example: real-time location systems for hospitals. Hospital location systems represent a growing segment of the location-aware computing market and rely on a broad set of complex location events that raise critical challenges.

Hospital location systems track the real-time location of people and equipment to cut costs and improve quality of care. Many applications provide value by simply locating needed assets (see Figure 1.1) using raw location events (i.e., entity X is at location Y). In contrast, a variety of more sophisticated applications use complex events (i.e., sequences of interactions among people, places, and things) to answer queries, trigger updates, and log information on pertinent higher-level activities. For example, a business process management application may log *radiology workflow* events that are detected when a patient leaves the hospital after checking in, spending time with a nurse in an exam room, and then visiting the radiology lab with a technician (see Figure 1.2). Such applications have been shown to dramatically improve the efficiency and quality of care in hospitals. Indeed, following an in-depth longitudinal study of one hospital's RFID location system, lead researcher Barbara Christe noted:

"The level of data generated by these systems has the potential to profoundly change the management of technology and the delivery of patient care." [324]

However, additional studies have shown that two key barriers to adoption remain: 1) *the cost of location sensors*, and 2) *the cost of tuning vendor software to site requirements* [66, 328].

In a detailed survey of 325 case studies, articles, and peer-reviewed journals, Vilamoska et al. [328] found that the cost of location sensors, or "tags" (see Figure 1.3), was the single most frequently mentioned obstacle to adoption of RFID location systems in healthcare.

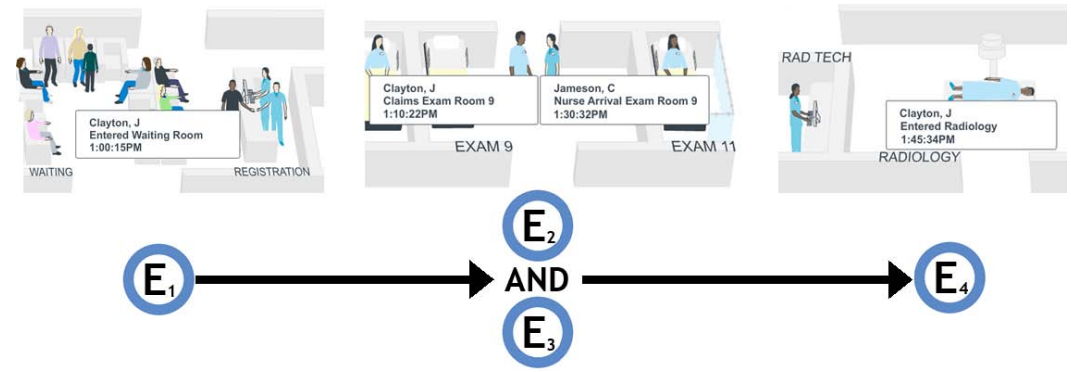


(a) Map display of asset locations

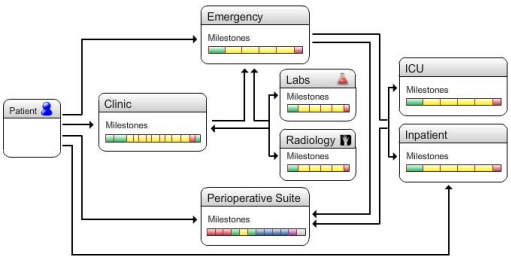
ID	Owner	Mfgr.	Model	Type	SubGrp.	PM Date	War. Exp.	Batt. Exp.	Current Loc.	Time	Hist.
115	ED	Alaris	PCA	IV Pump	IV All	12/24/10	03/24/10	05/24/10	Exam 2 A	2:52pm	...
116	ED	Alaris	PCA	IV Pump	IV All	01/18/10	02/18/10	04/18/10	Storage	4:01pm	...
255	CCU	Invacare	5000	Bed Simi	Bed Elec	03/01/10	10/01/10	01/01/10	Exam 7 B	11:02am	...
256	CCU	Invacare	5100	Bed Simi	Bed Elec	03/01/10	10/01/10	01/01/10	Exam 10 B	9:15am	...
270	ED	Verathon	6500	Blad Scan	BVM hh	02/15/10	11/15/10	05/15/10	Exam 9 A	4:35pm	...
315	CCU	Stryker	Epic II	CC Bed	CC Bed	12/05/10	01/05/10	12/05/10	Exam 3 A	2:10pm	...
360	ED	Shiller	AT101	ECG	ECG	10/20/10	03/20/10	05/20/10	Storage	8:15am	...
512	CCU	Stryker	2040	Stretcher	Stretcher	06/29/10	04/29/10	05/29/10	North Hall A	4:57pm	...
513	CCU	Stryker	2040	Stretcher	Stretcher	06/29/10	04/29/10	05/29/10	South Hall B	4:30pm	...
950	ED	Tamke	TMod	W-Chair	W-Chair	12/02/10	10/02/10	07/02/10	ISO WR 2	4:22pm	...

(b) Tabular display of asset locations

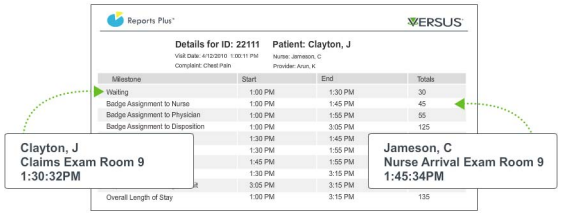
Figure 1.1: **Screenshots of applications that locate assets in a hospital using a location system from Versus Technology, Inc. [327].**



(a) A sequence of instantaneous events (E1 - E4) tracks a radiology workflow



(b) A diagram of complex workflows



(c) Event-level report of using (a)

Figure 1.2: **Examples illustrating complex location events in a hospital from Versus Technology, Inc. [327].**

The aforementioned longitudinal study by Christe et al. [66] provided deeper insight. That study of a RadarFind [261] location system in a 300-bed hospital found tags to be not only expensive at $50 each, but a burden to maintain. The tags were bulky and frequently dislodged from assets, damaged, or stolen. Moreover, new tags had to be purchased whenever batteries ran out because tag batteries were not rechargable or replaceable. For large hospitals with tens of thousands of assets, the periodic cost of buying and installing replacement tags easily exceeds $1 million (US).

Christe explained that significant costs are also incurred by addressing the second key barrier, that of *adapting vendor software to site requirements*. In that study, the 300-bed hospital needed to employ several full-time, on-site clinical engineers to install, tune, and maintain the location system. Another solution promoted by some vendors is for hospitals to adopt pre-established workflows that the vendor's software is designed to detect. Christe notes that process change is an even more costly alternative because hospitals often follow

(a)

(b)

Figure 1.3: **Key hardware components for real-time locating systems: (a) Active RFID tags. (b) Supporting reader infrastructure.**

well-established processes that are *"notoriously difficult to change"*. Vilamoska et al. also identified installation, tuning and process change as critical obstacles to adoption in most hospitals. In a follow-up report [326], Van Oranje et al. further explained:

"The organisational change required to make the application a success is often not considered properly. This leads to incompatibilities between the functionalities that the technology may offer and its use by hospital staff. Sometimes the interfaces do not display sufficient user-friendliness, or understanding of the context in which the technology is applied, but mostly the legacy processes and organisational culture are resistant to change, causing pilots to fail."

They concluded that *"RFID applications should be supportive of healthcare processes (not the other way around)."*

Thus, while hospital location systems show great potential, the high cost associated with deploying, tuning, and maintaining them are significant barriers to widespread adoption. These same problems are appearing in other domains and markets, especially those that use complex location events. We discuss how the contributions of this dissertation address these problems in the next Section.

1.2 Challenges and Research Contributions

This section generalizes the above example to articulate several key challenges faced by applications that use complex location events. We also describe the specific contributions made by this dissertation to address these challenges. Each contribution is numbered and later referred to as CK), where K is the number of the contribution.

1.2.1 Challenge: Cost of Location Sensors

As illustrated in the preceeding example, the cost of location sensing hardware can be prohibitive. This is especially true for applications that track a large number of entities. For example, some hospital location systems need to track tens of thousands of assets [93]. Similarly, a digital home may need to track hundreds of objects to intelligently reason about activities in the home [254]. Most state-of-the-art high-precision (e.g., room-level) location systems use battery-powered sensors that cost tens of dollars each [66]. Moreover, powered sensors inevitably require periodic battery changes, which again requires a large time investment for large deployments. Together, *these difficulties present a significant barrier to adoption of location systems in many application domains.*

Contribution 1) A Low-Cost Approach Using Unreliable Sensors and Bayesian Inference

In this dissertation, we present a low-cost approach to work with location events that uses Bayesian inference to compensate for uncertain data from cheap, unreliable location sensors. In particular, we enable use of sensors that provide only sporadic and approximate streams of location information on a tracked entity. Further, we present a Bayesian filtering technique that can probabilistically reason about the location of entities, even when those entities

Figure 1.4: **Passive RFID tags are a low-cost alternative to state-of-the-art location sensors at less than one-tenth the price per sensor.**

are in locations that are not covered by the location sensor deployment. For example, some hospital location systems do not cover every room due to budgetary or regulatory constraints (e.g., RF interference with critical medical equipment). Our solution could support probabilistic location tracking even in these rooms. Thus, in addition to enabling use of cheaper sensors, our approach enables cheaper deployments overall by reducing the amount of equipment needed to cover a given area. Chapter 4 provides more detail on the Bayesian inference technique used in this approach.

Contribution 2) Experimental Validation Using a Building-Wide Passive RFID System

Furthermore, we thoroughly explore and validate our Bayesian approach to work with location data through experiments with a real sensor deployment. In particular, we use passive EPC Gen 2 RFID sensors, or "tags" (see Figure 1.4) that cost an order of magnitude less than state-of-the-art sensors. We deploy a building-scale RFID system, called the *RFID Ecosystem*, with over 70 users and hundreds of tags that track people and personal objects. Extensive experimentation and long-term studies with the RFID Ecosystem validate the feasibility of using passive tags with our Bayesian approach. We also characterize the performance of RFID deployments in useful ways and identify a number of pitfalls and optimizations for sensor performance. The RFID Ecosystem and our experiments with it are introduced in Chapter 2.

1.2.2 Challenge: Flexible Support for Location-Events

Location-aware computing constitutes a broad and extremely diverse range of applications, from mobile friend finders, to GPS-based navigational aids, to efficiency and safety monitoring applications for enterprise use. These applications represent an equally diverse set of location-events, ranging from simple presence events to complex workflows. To effectively support even a cross-section of location-aware computing requires a deep understanding of this space and an extremely flexible architecture for specifying and detecting new events. Moreover, to support C1), any proposed infrastructure must also support event detection over the uncertain, probabilistic location data output by our Bayesian filter.

Contribution 3) A Detailed Survey of Events in Location-Aware Computing

To gain a comprehensive understanding of events in location-aware computing, we conduct an extensive survey of research, consumer, and enterprise location-aware applications over the last decade. We analyze the surveyed applications to distill a high-level understanding of application domains, common events, and event usage in applications. Indeed, a key result is a taxonomy that describes the most common events in all of location-aware computing as well as the most common techniques for composing these events into more complex events. We use these survey results to provide design guidance concerning all future systems and tools that support work with location events. Chapter 3 describes the survey and results.

Contribution 4) The Cascadia Middleware for Flexible Event Services

Using our low-cost, probabilistic approach to tracking in C1), the real sensor data from C2), and the in-depth understanding of events from C3), we design, implement, and evaluate a middleware for specification, detection, and management of location events. The middleware, *Cascadia*, leverages a new probabilistic event detection engine called Lahar [211, 273] to detect events over probabilistic location streams from low-cost sensors. Because Lahar is limited to support for a small set of events, Cascadia extends Lahar to support precisely the most common events identified by the taxonomy in C3). As a result, it supports over 90% of events used across the over 400 surveyed applications. We evaluate Cascadia's event detec-

tion performance using real RFID data from the RFID Ecosystem in C2), and demonstrate how it facilitates application building with a case study. Cascadia and its components are introduced and evaluated in Chapters 4 and 7.

1.2.3 Challenge: Cost of Deploying and Tuning Applications

As demonstrated by the example in Section 1.1, a significant effort is required to deploy and tune many location-aware applications to the requirements of an individual or organization. One fundamental part of support for this process is *event specification*; users must be able to specify new events to meet their evolving needs. Applications achieve this by leveraging event detection systems (e.g., context-aware computing infrastructures) that allow new events to be formally specified in some manner. However, while existing systems allow developers (i.e., engineers) to specify new events using low-level APIs [33, 286, 322] or a declarative query language [115, 148], dependence on developers is a costly inconvenience for both individuals and organizations. Indeed, recall that the survey of RFID location systems in hospitals by Vilamoska et al. [328] cited the cost of tuning vendor software to site requirements as a critical barrier to deployment.

A compelling alternative is to allow direct specification of complex events by end-users. This is challenging, however, because it requires translation of high-level concepts into conditions on low-level and uncertain sensor data about diverse entities and places. Unfortunately, existing systems for end-user event specification are either limited by inexpressive interfaces [203, 229] or require iterative and potentially unfeasible training demonstrations for machine learning models [89]. Thus, effective end-user specification of events remains a difficult challenge.

It is equally important that end-users be able to *verify* event specifications and debug those that do not work. Verification is difficult because it requires a system to produce high-level evidence of a specification's behavior over sensor traces that may be too complex for an end-user tool to generate. Moreover, because bugs can occur at the sensor level (e.g., calibration errors) or in the specification design, users must be able to understand detected events and evaluate their relationship to sensor data. This is impractical or impossible

when events are specified with inscrutable machine learning models or when they do not represent uncertainty. As such, existing systems for end-user event specification are limited by inadequate support for verification and debugging.

Contribution 5) The Scenic Tool for End-User Event Specification

We address the event specification problem with the Scenic tool, a tool that allows end-users to specify complex location events. Scenic is a web-based tool that uses an iconic visual language designed to support common location events and their composition through sequencing and conjunction. A storyboard layout describes how people and objects enact an event through a sequence of movements between places. The set of supported events again mirrors the taxonomy of events distilled from the survey in C3) and supported by Cascadia in C4). We design and evaluate Scenic through a series of laboratory studies and using real RFID data from the RFID Ecosystem in C2). We show that Scenic does indeed make complex event specification accessible to end-users, thereby empowering them and decoupling their work from expensive engineering resources. The design of Scenic is reviewed in Chapter 5.

Contribution 6) The Panoramic Tool for End-User Event Verification

Panoramic, an extension of Scenic, supports verification (or "debugging") of complex events as specified in Scenic. Panoramic does not require users to write code, understand complex models, perform elaborate demonstrations, generate test traces, or blindly trust deterministic events. Instead, it offers an intelligible verification interface using timeline and map-based interfaces for review and understanding of the hierarchical structure of events and the relationship between events and low-level sensor data. Panoramic also takes a novel approach to verification in which specifications are tested on historical sensor data. Panoramic is designed and evaluated through a series of laboratory studies using real RFID data on events that occurred in the RFID Ecosystem. We show that Panoramic effectively supports end-user verification of complex location events. Moreover, we identify a number of directions for future work that build on Panoramic. This work is presented in Chapter 6.

1.2.4 Recap of Contributions

In summary, this dissertation addresses several of the most critical barriers to adoption and effective use of context-aware applications. We cope with the cost and inherent uncertainty of sensor systems using a layer of Bayesian inference. We address the growing diversity and scale of context-aware applications by developing Cascadia, a middleware for specifying, detecting, and managing complex events on top of probabilistic, uncertain sensor data. Our Scenic and Panoramic tools allow end-users to understand, control, and hence trust context-aware applications by enabling them to directly specify and verify complex events. They also substantially reduce the difficulty of deploying and tuning a context-aware application. Finally, we evaluate all our contributions through experiments and studies with real sensor data from a representative sensor deployment, the RFID Ecosystem. While all our contributions are specific to location-aware computing, many of our results can be generalized to other categories of context-aware computing. Moreover, our contributions represent an important step toward supporting cheaper, more robust, flexible, and more understandable context-aware computing systems.

1.3 Dissertation Outline

The remainder of this dissertation is organized as follows. Chapter 2 provides background on RFID technology and the building-wide RFID Ecosystem in which this work is grounded. Chapter 3 presents a comprehensive survey of location events in research, consumer and enterprise applications over the last decade. A taxonomy and terminology for events is also introduced. Chapter 4 presents a substrate for detecting complex events over sproadic and uncertain location information and Chapter 5 presents the design and evaluation of Scenic and several associated tools. Chapter 6 discusses the crucial issue of verifying events that are specified using the said tools and presents the design and evaluation of a tool to do so. Chapter 7 discusses the design and evaluation of the Cascadia middleware that incorporates the event detection substrate in addition to the end-user tools to enable and facilitate computing with complex location events. Related work is reviewed in Chapter 8 before discussing future work and concluding in Chapter 9.

Chapter 2

PASSIVE RADIO FREQUENCY IDENTIFICATION

The design and evaluation of systems in this dissertation are grounded in *real* sensor data from experiments with a building-scale passive Radio Frequency Identification (RFID) technology deployment called the RFID Ecosystem. In this chapter we describe passive RFID and motivate its use as a location sensor for a complex location event detection system (see Figure 2.1). We also present the RFID Ecosystem and characterize its performance through laboratory experiments and a longitudinal field study. Finally, we argue that RFID is a suitable location technology for a range of applications and suggest methods for optimizing its performance.

2.1 Passive RFID Technology

As described in Chapter 1, there are different types of RFID technology. In particular, some systems use *active* (i.e., battery-powered) tags, while others use *passive* tags. Passive RFID, and particularly the Electronic Product Code Class-1 Generation-2 (EPC Gen 2) standard [22], has the potential to enable large scale location-aware applications at much better cost and convenience than current solutions based on active RFID. In this section, we present a detailed overview of passive RFID technology along with its associated benefits and challenges. We focus on EPC Gen 2 RFID because we use this standard in our experiments.

2.1.1 Readers and Tags

The two fundamental components of an RFID system are tags and readers (see Figure 2.2). RFID tags are devices that store and can wirelessly transmit a small amount of data using radio waves. As explained above, passive tags have no on-board source of power; instead, they harvest power from radio waves emitted by a nearby RFID reader. RFID readers are powered, networked devices with one or more attached antennas that continuously broadcast

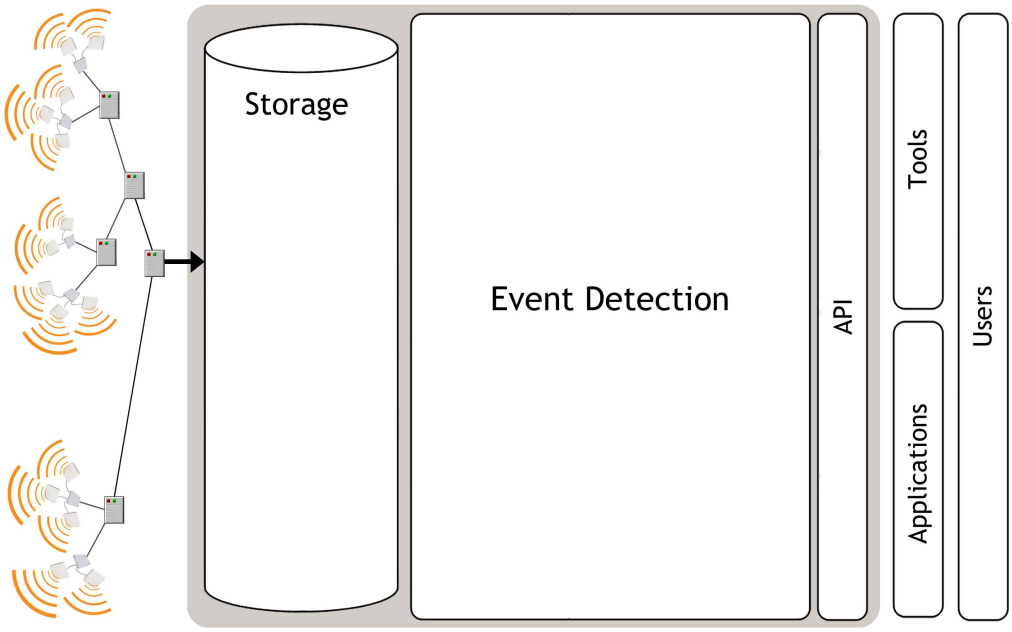


Figure 2.1: **In this chapter we focus on the sensor deployment portion of the system.**

a low-frequency (LF: 125 KHz), high-frequency (HF: 13-56 MHz), or ultra-high frequency (UHF: 860-960 MHz) interrogation signal in accordance with some standard; EPC Gen 2 is a UHF standard. When a compatible tag (i.e., one that implements the same standard) passes through an antenna's area of coverage, it can harvest the energy from the radio wave emitted by that antenna and use it to respond with its stored information. In most cases, the information stored on a tag is simply a numeric identifier. EPC Gen 2 tags store a 96-bit identifier.

2.1.2 Benefits and Constraints of Passive RFID

Its particular properties as an automatic identification technology have made passive RFID a good fit for a variety of applications while posing significant engineering challenges for system designers. Here we enumerate and categorize the key properties of passive RFID, and EPC Gen 2 RFID in particular, as specific benefits over other location and identification solutions or as technical constraints that impact the design of systems that consume RFID data streams.

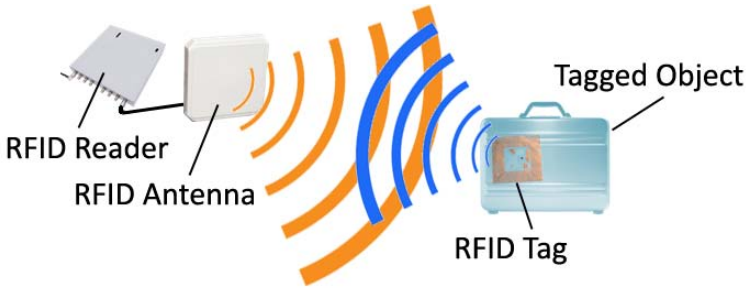


Figure 2.2: **RFID reader antennas read RFID tags carried by people and objects in an RFID deployment.**

There are seven key benefits of RFID technology:

- **Benefit:** *Wireless.* Passive RFID can be used to wirelessly identify an object at a distance of several meters, even in the presence of some occlusion. This is an advantage over line-of-sight technologies like barcodes and infrared.

- **Benefit:** *Efficient.* Most RFID standards allow tags to be read quickly. The EPC Gen 2 standard is designed to allow an RFID reader to detect over one thousand tags per second using an advanced anti-collision algorithm.

- **Benefit:** *Unique identification.* An EPC Gen 2 tag typically stores at least 96-bits of information. A numeric identifier this size is large enough to uniquely identify all the products in the world for many years to come. This is a distinct advantage over sensors that do not provide precise identification (e.g., barcodes, lasers, vision).

- **Benefit:** *Battery-less.* Passive RFID tags operate solely on power harvested from the RF signal emitted by a reader's antenna. No batteries or battery changes are ever needed. This is in direct constrast to all active RFID tags, some of which even use non-replaceable batteries, forcing periodic purchase of expensive new tags.

- **Benefit:** *Low-Cost.* Passive RFID tags typically cost anywhere from a few cents to a few dollars when purchased in bulk. This makes them more than an order of magnitude cheaper than most active RFID tags.

- **Benefit:** *Already pervasive.* EPC Gen 2 RFID is already used widely for tracking supply chain processes. By adopting EPC Gen 2 for additional location systems, companies can seamlessly integrate with existing systems to provide enterprise-wide location awareness.

- **Benefit:** *Evolving.* RFID technology is an active area of research with new capabilities such as increased storage, security and built-in sensors being added to standard technologies like EPC Gen 2 [179, 300]. While many location technologies are also evolving, RFID's shortcomings are rapidly being addressed by a large research community.

Passive RFID also presents five key technical constraints:

- **Constraint:** *Coarse-grained location.* The location information provided by RFID is only as fine-grained as the detection field of the reader antennas. This is typically between 2 and 5 meters.

- **Constraint:** *Sensitive to materials in the operating environment.* The presence of metal, water or other *RF-absorbent* material near a tag can prevent that tag from being read [200, 263]. The false negatives caused by such environmental conditions degrade the quality of RFID data by introducing intermittent *gaps* or *holes* into the data stream or by blocking the data stream altogether.

- **Constraint:** *Sensitive to tag orientation.* The physical orientation of a tag with respect to the reader antenna can affect a tag's readability. Non-optimal tag orientations can degrade an RFID data stream in a similar fashion.

- **Constraint:** *Sensitive to RF reflections and interference.* Without expert hardware installation, reflections and multipath effects can create RF fields with unexpected "dead zones" which are a further source of false negatives. RF interference from or with other nearby devices can also be a problem [325].

- **Constraint:** *Little or no security.* Most passive RFID standards currently implement limited or no security features like encryption and password protection. This means that most tags will reveal their data to any reader in range without authenticating first.

2.1.3 Elements of a Complete RFID System

Readers and tags are the definitive elements of an RFID system; however, from an end-to-end perspective, a system necessarily includes a variety of other important components. One fundamental component is a database for persistent storage of RFID data, metadata, and data on derived high-level events. Another component found in nearly all RFID systems is a software infrastructure or *middleware* for gathering, processing, storing and serving RFID data. Importantly, middlewares are also responsible for transforming raw RFID data into meaningful high-level events in spite of gaps and holes in the data stream. They expose event data to applications and existing enterprise infrastructure through carefully designed APIs. Chapter 7 discusses RFID middleware in greater detail. Finally, all systems also include networking infrastructure that links readers to middleware, databases and applications. In many cases Ethernet serves this function, but the Internet may also fulfill this role in part.

2.2 The RFID Ecosystem

The results of this dissertation depend critically on experimentation with a building-wide EPC Gen 2 RFID deployment called the RFID Ecosystem. The goal of this deployment is to simulate the conditions of an RFID-saturated future environment at scale for long periods of time and in a carefully controlled setting. We discuss the details of the RFID Ecosystem in this section. In particular, we describe the specific RFID equipment used as well as the deployment layout, parameters, and software. We also present results from laboratory experiments and microbenchmarks designed to characterize the behavior of our system in terms of data generated and success rates when operating under various conditions.

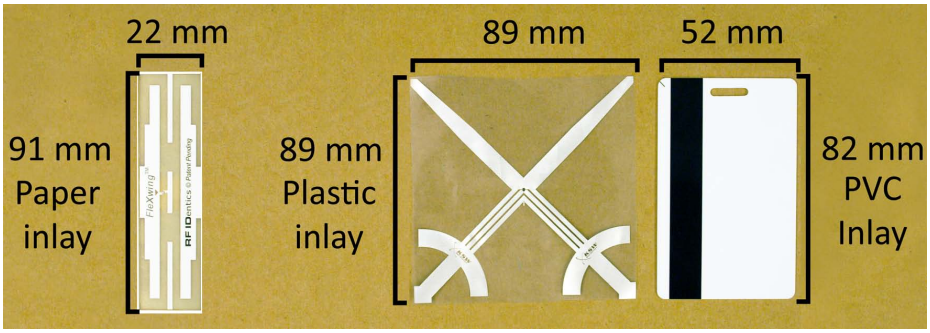


Figure 2.3: **The three tag designs used in our study: the FleXwing, the Excalibur, and the RFIDentics card.**

2.2.1 EPC Gen 2 Equipment

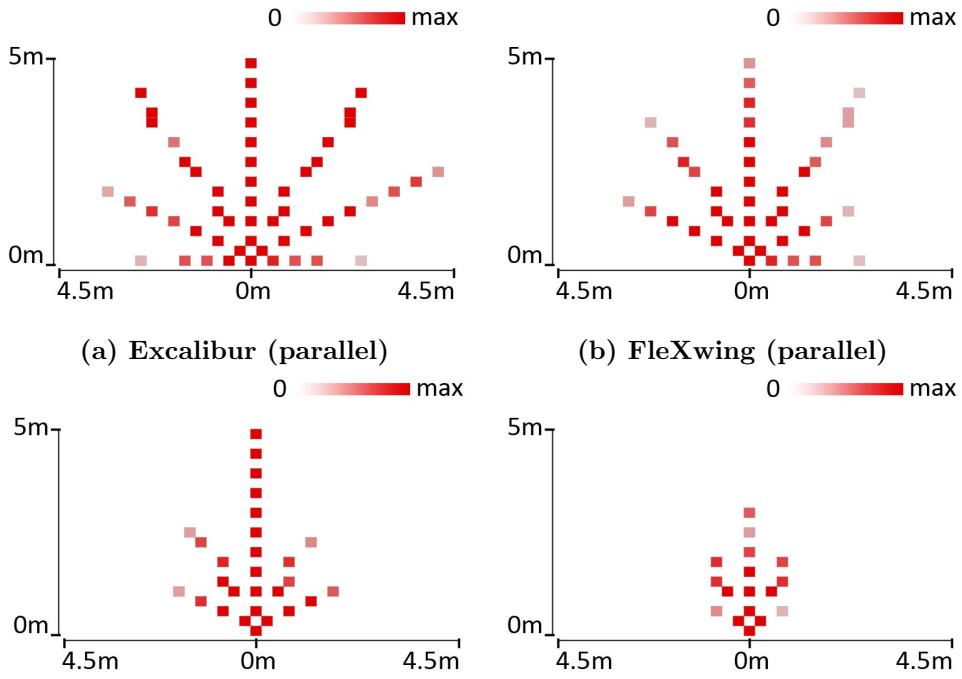
As explained above, we use the EPC Class-1 Generation-2 (EPC Gen 2) RFID standard [22] in our experiments. EPC Gen 2 RFID operates in the 860MHz-960MHz frequency range using the passive-backscatter, interrogator-talks-first approach and has an operating range that is typically measured in meters.

Specifically, we use Speedway readers from Impinj [168]. These readers are equipped with Ethernet and run Linux with proprietary firmware that exposes tag inventory results to user-authored code. To each reader, we attach between 1 and 4 Cushcraft S9028PC antennas. The S9028PC measures 25.4cm x 25.4 cm x 3.8cm, has a gain of 9 dBic, is circularly polarized (left or right), has a VSWR of 1.5:1 and a beamwidth of 60° by specification. We set the readers in *dense reader mode* (Miller-4 tag coding) [22]. This mode is designed to reduce RF interference from other nearby readers in our deployment and has been shown to provide a longer read range than other modes [50]. Finally, we set the reader transmit power to the maximum allowed 30 dBm to optimize for maximum operating range. In combination with the Speedway reader we use three tag designs (see Figure 2.3): the RFIDentics FleXwing [277] tag, the KSW Microtec Excalibur tag [196] and an RFIDentics tag embedded in a PVC magstripe card [277]. All three tag designs use the Impinj Monza chip and store a manufacturer-set 96-bit Electronic Product Code (EPC). In this paper, we refer to the EPC as the tag identifier or *tag ID* since we tag personal objects and people rather than products.

To characterize the relative performance of our three tag designs, we conducted a laboratory experiment in which we measured the *read-rate* for each design at various positions and orientations relative to a reader antenna. As Hodges et al. [155] point out, read-rate is a simple and widely accepted way to assess an RFID system's operating range that takes into account all factors affecting range in an environment. We formally define read-rate as:

Definition 2.2.1 *Read-rate: The rate at which a reader can successfully read a tag.*

We began the experiment by placing a single reader and antenna against one wall of a 6m x 8.5m room containing tables, desktops and wooden shelves (with metal braces) at its sides. Then, in three trials (one for each tag design), we moved a tag through a series of 245 points inside the half-sphere with radius 5m in front of the reader antenna. At each point the tag was oriented first parallel, and then perpendicular to the face of the reader antenna. We measured read-rate over a 10 second period for each position. A horizontal ($\phi = 90°$) slice of the half-sphere is shown for each tag design in Figure 2.4; darker shade indicates higher read rate. Figures 2.4(a),(b) and (c) show the results for parallel tag orientation. Notice first that in general, the read rates are highest in the *major detection field* [142], an arc directly in front of the antenna, and decrease as the tag moves out of this arc, through the antenna's *minor detection field*, to the periphery of the full operating range. The Excalibur design has the best read-rate throughout the full operating range while the FleXwing and PVC card offer good rates over a slightly smaller range. While the Excalibur and FleXwing tags were quite robust to change in orientation, the PVC card shows significant performance degradation when oriented perpendicular to the reader antenna (see Figure 2.4(d)). These results suggest that the Excalibur tag should offer the best performance in the variety of situations, followed by the FleXwing and then the PVC card. More rigorous laboratory studies on passive UHF RFID performance can be found in related work [50, 155, 263]. The goal of this experiment was simply to understand the basic performance properties of the tags used in our study.



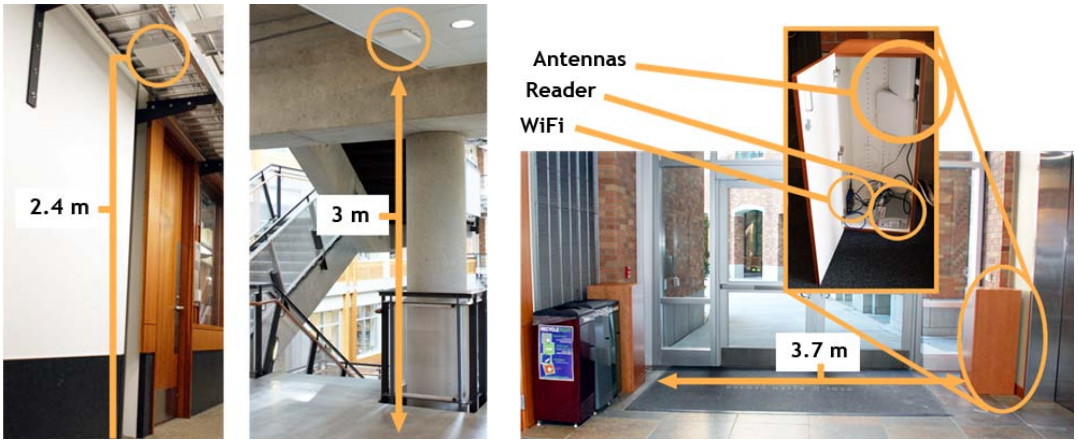
(a) Excalibur (parallel) **(b) FleXwing (parallel)**

(c) PVC card (parallel) (d) **PVC card (perpendicular)**

Figure 2.4: **The read rates for the three tag designs at points inside a horizontal slice of the half-sphere centered in front of the reader antenna.**

2.2.2 Building-Scale Deployment

We deployed 47 readers and 160 reader antennas throughout the 7 floors of our 8,000 square meter Computer Science and Engineering building with antennas located as shown in Figure 2.6. Figure 2.5 shows the two adaptations of industry standard configurations [123] that were used to install antennas: *hallway portal* (HP) and *gateway choke point* (CP). In the HP configuration, an antenna is mounted above a hallway pointing downward to read tags passing through that section of hallway. The CP configuration points antennas with opposite polarization inward (towards each other) from both sides of an entrance to a particular part of the building (e.g., atrium, laboratory). HPs afford a cheaper installation with wider coverage using less hardware while CPs are designed to provide higher read-rates at important transition points. The full deployment details are summarized in Table 2.1. These parameters were shaped not only by inevitable budgetary and regulatory considerations,



(a) Hallway portal **(b) Gateway choke point**

Figure 2.5: **Two adaptations of industry standard RFID antenna configurations.**

Count	Mounting	Config	Elevation
132	Under cable tray	HP	2.4 m
5	Under lounge ceiling	HP	3 m
5	Above stairwell	HP	3 m
10	Beside building entry	CP	45 cm
2	Beside conference rooms	CP	45 cm
6	Under lab cable trays	HP	3 m

Table 2.1: **RFID antenna deployment parameters.**

but by the aesthetic and ethical concerns of our community as well. For example, we needed to hide CP installations inside aesthetically appealing cherry wood boxes and avoid deploying antennas in offices altogether so as to avoid RF interference with telephones and other personal equipment. The latter constraint, which limits reader topology, is shared by important domains like hospitals, where minimizing interference from RFID equipment is a crucial requirement [325]. We address the problem of limited reader topology using Bayesian inference in Chapter 4.

All readers in our deployment run custom software (700 lines of C code) that processes new RFID data before streaming it over wired or wireless Ethernet it to a central server.

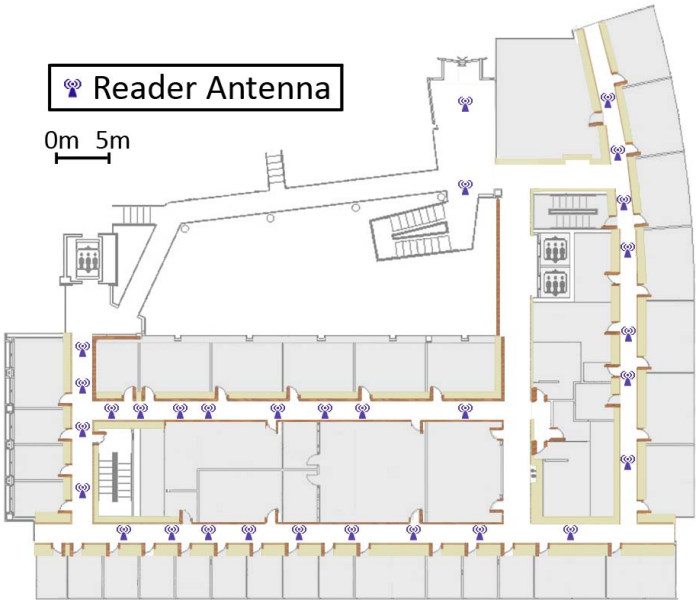


Figure 2.6: **Layout of the RFID Ecosystem on a representative floor.**

This software continuously polls the firmware for new tag reads and packs any data into one *tag read event* (TRE) per tag per antenna per second, a tuple with the schema: `(tag ID, antenna ID, start time, number of reads)`. For example, if within one second a tag `A` is read 32 times by antenna `1` starting δ_1 milliseconds after time `t` and 20 times by antenna `2` δ_2 milliseconds after time `t`, then our software will generate and send two TREs to the server: `(A, 1, t+`δ_1`,32)` and `(A, 2, t+`δ_2`, 20)`. Each reader also runs ntp [242] to synchronize its clock with the rest of the system.

The central server stores all TRE data in a SQL Server database [310] that also contains metadata on the deployment including the latitude and longitude of each antenna and a symbolic antenna name (e.g., "front entrance", "4th floor stairwell"). Custom Java code (56 classes) orchestrates data transmission between the readers and the database using JDBC [174] and Apache's MINA library [16] for efficient, secure networking. Finally, a set of administrative and diagnostic web applications are hosted with Apache [15] and Tomcat [17] on a separate web server.

Object	Material	Tag	HP	CP	MT
3-ring binder	paper	Excalibur	0.16	0.8	0.6
DVD	metal	FleXwing	0.09	0.2	0.12
Hardcover	paper	Excalibur	0.18	0.7	0.5
Magazine	paper	Excalibur	0.54	0.6	–
Mobile phone	metal	FleXwing	0.04	0.25	0.1
Paperback	paper	Excalibur	0.39	0.55	0.7
Power cable	metal	FleXwing	0.27	0.4	0.3
Water bottle	plastic	FleXwing	0.61	0.8	–
Backpack	cloth	Excalibur	0.58	0.9	–
Badge	plastic	Excalibur	0.77	1.0	–
Belt	cloth	PVC Card	0.02	0.15	0.02
Hat	cloth	Excalibur	0.82	1.0	–
Jacket	cloth	Excalibur	0.28	0.3	0.8
Shirt	cloth	Excalibur	0.07	0.4	0.1
Wallet	cloth	FleXwing	0.05	0.0	0.06

Table 2.2: **A table summarizing t he results of our microbenchmark experiments with the RFID Ecosystem. Tagged objects are in the first column, with the top 8 objects carried inside the backpack and the bottom 7 outside. The second column shows tag design, and the third column average detection rate for each microbenchmark (either HP for the hallway portal benchmark, CP for the gateway chokepoint benchmark, or MT for the multi-tagging benchmark.**

Deployment Benchmarks

Unlike the idealistic single antenna laboratory case, RFID performance in a large deployment is more complex and depends on a variety of other factors like velocity and changing orientation of tags, ambient RF interference and interference from materials composing tagged objects or people. To better understand performance in our deployment amidst these effects, we calculated some baseline measures using three microbenchmarks. In the first microbenchmark (HP), a researcher walked through a sequence of 145 HPs carrying 15 tags attached to various objects. In the second (CP), the same researcher walked through just 2 CPs carrying the same tagged objects. A third microbenchmark (MT) investigates the *multi-tagging* technique proposed by Ravindranath et al. [268], in which multiple tags are attached to an object at different orientations in an effort to increase the probability that the object is detected. In the MT microbenchmark, one additional tag was added to each object having poor results in the HP microbenchmark, after which the researcher walked through the same sequence of 145 HPs. For each microbenchmark, we calculate detection-rate, which we define as follows:

Definition 2.2.2 *Detection-rate: The probability that a tagged object passing in the vicinity of an HP or CP is detected at least once.*

We compute the detection-rate for each object as the fraction of HPs (or CPs) passed which detected at least one of the object's tags. The first three columns of Table 2.2 describe the tagged objects as well as the tag design used.

Results

Across 4 trials of the HP microbenchmark, 10 trials of CP and 4 trials of MT, we collected 155 minutes of trace data. Table 2.2 shows the average detection-rate across all trials and benchmarks. The results are consistent with the laboratory experiment in showing that tag design affects performance, with Excaliburs providing an average detection rate that is 0.28 greater than FleXwings and 0.47 greater than PVC cards across all objects and trials. The already well-known tendency [123, 343] for a tag's readability to depend strongly on the

materiality of the object it is attached to shows up here as well, with metal objects being more frequently missed than paper, cloth or plastic objects.

The way in which objects were carried and tags oriented also had a noticeable impact. Tags on objects kept close to the body (e.g., wallet, shirt, belt) are not read as well. Moreover, for all microbenchmarks there was variation in measured detection rate across trials, especially for objects inside the backpack which had detection rates with standard deviation greater than 0.1. This was likely due to shifts in the relative position of objects inside the backpack during and between trials (indeed, the backpack was unpacked and repacked between the second and third HP trial and between microbenchmarks). The results of the MT microbenchmark show that detection rate roughly doubled for objects not severely limited by the materiality problem or the way in which the object was carried. Dramatic increases in detection rate for MT were most likely due to a change in object orientation inside the backpack (e.g., for the 3-ring binder) or an additional tag being added in a much more readable orientation (e.g., a second tag clipped to the jacket's breast pocket in addition to a first tag in the side pocket). These results show promise for the multi-tag technique. However, we do not investigate the multi-tag technique further in this study as it complicates the study of end-user tag management as discussed in Section 2.3.2.

It is also clear from Table 2.2 that antenna configuration played a major role in determining whether or not a tag was detected, with CP providing an average detection rate across all tags and trials that is 0.21 better than the average HP rate. This is probably due to the fact that communication between Speedway readers and tags is *forward-link-limited* [167], meaning that detection depends on a tag's ability to harvest adequate power from the reader signal. In our case, the additional antennas in the CP configuration increase the probability that a tag is in an antenna's major detection field, making it more likely that the tag can harvest enough power to respond. Nevertheless, we chose to keep an HP-heavy deployment layout because the CP configuration was prohibitively intrusive and expensive to deploy widely throughout the building.

2.3 Field Study Setup

While laboratory and microbenchmark studies provide valuable insight into the performance of EPC Gen 2 RFID, they do not answer important questions regarding community use of a large-scale passive RFID system over a long period of time. Though many pervasive computing researchers have proposed passive RFID applications [103, 157, 32, 45, 220, 239, 268], to our knowledge, very few have investigated behavior and performance in such a scenario. Most previous studies were limited to controlled laboratory environments like those in Section 2.2.1 [50, 155, 263]. As such, we are left with a number of questions that laboratory studies cannot answer, such as how well do results from the laboratory map to an everyday-life setting? How much data does a community of RFID system users generate and what does the data look like? How well can tags be read when mounted on everyday objects by end-users? By experts? Are users eager to adopt this new technology? How do they manage their private location data? How do usage trends evolve over time? Are there any trends or correlations in the data that might inform system and application design?

To begin answering the above questions, we conducted a four-week study in the RFID Ecosystem with 67 participants and 324 tags. During the study, participants answered web-based surveys that collected ground truth location information and could review their RFID data with web-based tools. Table 2.3 summarizes the high-level parameters. This study contributes results in connection with three key aspects of an RFID deployment:

(1) *System utilization*: We measure utilization in terms of the amount and character of data generated. We also identify trends in these quantities over time. As part of these measurements, we record how often users interact with their RFID data or delete sensitive data. Overall, we find that an RFID infrastructure produces small data volumes (approximately 0.8 MB of data per participant over 4 weeks on average) and that this data can nicely be compressed (e.g., 20KB per person on average). We find more than an order of magnitude difference in the amount of RFID data generated by different users and objects and that at least three factors are responsible for this trend: differences in user mobility, differences in how people carry their tags, and differences in materiality of tagged objects. Interestingly, we also found that few people deleted any data and no one withdrew from the study.

Entity	Measure
Participants	67
Object types	19
Total tags	324
Duration (weeks)	4

Table 2.3: **Parameters of our four-week user study.**

(2) *System performance*: We measure the load distribution across readers and over time. As in previous mobility studies (e.g., [30]), we find that some locations are hot spots. In the case of an RFID deployment, however, such trends can amount to an order of magnitude difference in the quantity of data produced at different locations. More importantly, we estimate the reliability of the system: i.e., how often readers detect different tag designs in their vicinity when tags are managed and used by non-experts. We find that the detection rates are relatively low overall and highly variable. We find significant differences in detection rate among tag designs, types of objects, and individuals. Some individuals and objects are detected more than 80% of the time. The median detection rates, however, are below 40%.

(3) *Miscellaneous Challenges*: During the study and in preparation for the study, we encountered a variety of challenges. For example, deployment of antennas in a busy, shared public space; management and maintenance of a system with hundreds of distributed components; and ensuring user privacy when experimenting with a technology that is often considered to be insecure and invasive in nature. We report on the lessons that we learned.

We present the study methodology in detail throughout the remainder of this section.

2.3.1 Recruiting Participants

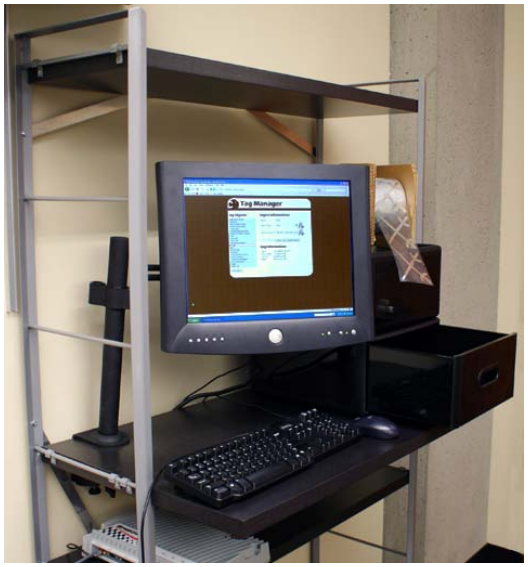
We recruited study participants from the community of faculty, staff, undergraduate and graduate students that occupy our building. During a 2-week recruiting period, we enrolled 67 participants (30 graduate students, 33 undergraduates, 2 faculty, 2 staff; 46 male, 21 female) who carried a total of 324 tags on 19 different types of objects. At the time of consent, all participants were educated on RFID, location-aware computing and the purpose

of the study. Prospective participants were also presented with a demonstration of the tools they would have access to if they chose to participate. As additional compensation for participating in our study, we offered participants $30 for up to four months of participation with a chance to earn up to $20 more for completing a series of short, web-based surveys (see Section 2.3.4). All participants were informed that they could withdraw from the study at any time and that they could also delete any subset of their data at any time using a web-based Data Browser tool which we describe in Section 2.3.2.

2.3.2 Tag and Data Management

Upon consenting, participants were asked to carry one RFID tag as a "personal badge" and as many additional tags as they liked for use with personal objects. In order to determine whether end-users could effectively mount and manage their own tags, we employed a two-phase strategy for tag management. In the first phase, participants chose, mounted and managed their own tags directly; in the second phase we assisted participants in remounting their tags in an optimal fashion. We describe each phase in detail below.

On entrance to the study, participants chose tags, created a digital link between each tag and the object to be tagged, and then physically attached the tags to those objects. To facilitate this process, we built an *RFID Kiosk* in our laboratory. The Kiosk (see Figure 2.7(a)) was built into a computer desk and includes a supply of RFID tags, a PC with an RFID reader and a single antenna configured for short-range reading. Participants accessed the *Tag Manager* tool (see Figure 2.7(b)) through the PC, which allowed them to create metadata for a particular object as well as to poll the Kiosk's reader for a tag to associate with that metadata. We did not encourage participants to associate multiple tags with a single object because this would complicate our evaluation of how expert optimization of tag mounting increases readability for an object. Later on, participants could access the Tag Manager from any web browser if they decided to edit their object metadata; however, creation or alteration of tag-object associations was only possible when using the Tag Manager at the RFID Kiosk.



(a) The RFID Kiosk.

(b) The Tag Manager.

Figure 2.7: **Tools used by longitudinal study participants to register and manage RFID tags: (a) The RFID Kiosk: PC and RFID reader with one antenna installed above the plastic drawer to the right of the display, and (b) the Tag Manager for end-user management of tags and data, integrates with the RFID Kiosk to allow new tags to be registered.**

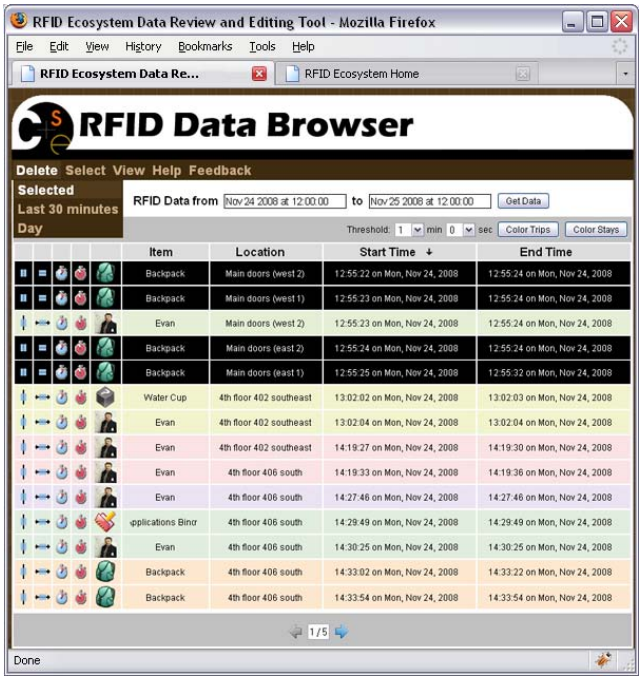


Figure 2.8: **Data Browser tool that allows study participants to review and optionally delete their data.**

Participants were given guidelines as to how RFID tags should be attached to objects and to themselves for the best performance (e.g., "a tag will be read infrequently or not at all if it is too near to your body or to a metallic object"). Throughout the first 3 weeks of study, participants were urged to follow the guidelines and reminded by email that they could directly see how well their tags were being read by checking the dynamic plot on the RFID Ecosystem Homepage or by reviewing the collected data with the Data Browser. In the last week of the study, we invited participants with less frequently detected tags back to our laboratory and helped them find a more optimal mounting technique. The results of this remounting experiment are covered in Section 2.4.3.

In addition to user-driven management of tags, we also wanted to investigate the durability of each tag design. To this end, we distributed laminated and non-laminated versions of the Excalibur and FleXwing tags and made note of all incidents in which participants reported a tag as broken.

A Data Browser tool (see Figure 2.8) facilitated management of personal privacy for participants. The Data Browser displays a table containing a dump of RFID data collected

about a user. The Data Browser can be used to survey what data has been collected as well as to delete any data at the granularity of a TRE, an hour, or a day. Participants had access to the Data Browser throughout the study.

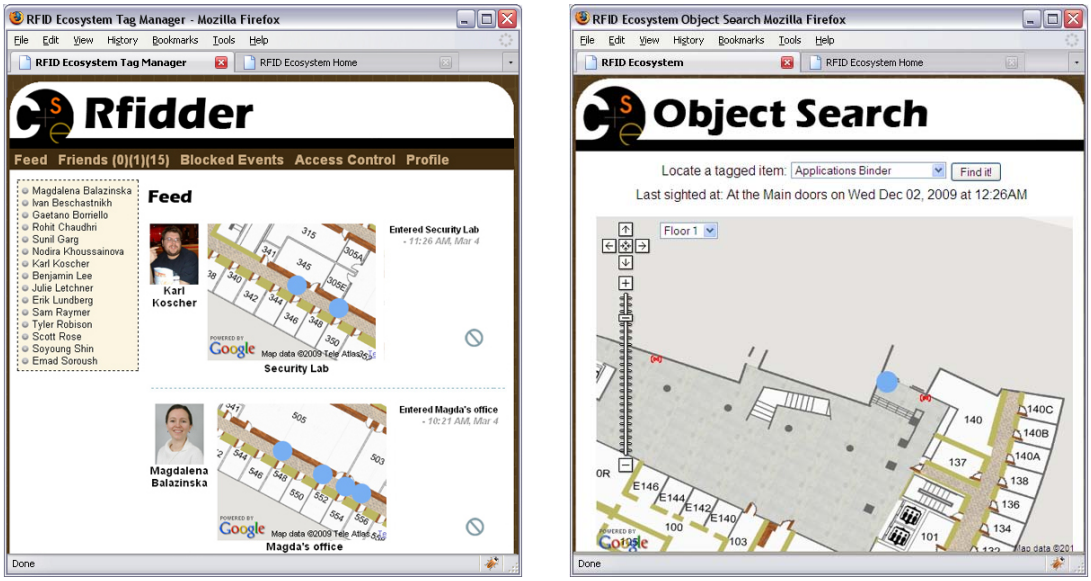
2.3.3 RFID Applications

As an incentive for building occupants to use our system, we created a set of simple location-aware computing applications. All applications used secure web-based interfaces and were implemented using the Google Web Toolkit [140]. As such, interfaces used JavaScript with AJAX calls to retrieve dynamic content from an Apache Tomcat Servlet [17]. The Servlet made simple queries over RFID data and metadata from the Tag Manager to provide information on the locations of people and objects and to detect very simple location events (e.g., "Alice passes under antenna 54").

All applications were accessible via links from the RFID Ecosystem Homepage which also displayed a dynamically generated plot showing how much data has been collected about the user over the past week. In order to better understand how users interact with RFID data, we logged all operations made through an application that resulted in a database transaction (e.g., add friend, query friend's location, delete range of data). We present a brief overview of the applications with screenshots below.

Search Applications

Three applications provide location-aware services that facilitate an awareness of the location of one's friends and one's personal objects. Two of the applications, Rfidder and a Facebook application, report near-real-time information on the location of a user's friends. Rfidder, shown in Figure 2.9(a), provides a stream of location updates for one's friends in the form of a microblog with entered-location and exited-location events. The Facebook plugin displays a user's friend list along with the most recently recorded location for each friend. A third application, Object Search, (see Figure 2.9(b)), allows users to view the location of the most recent TRE corresponding to an object they own.

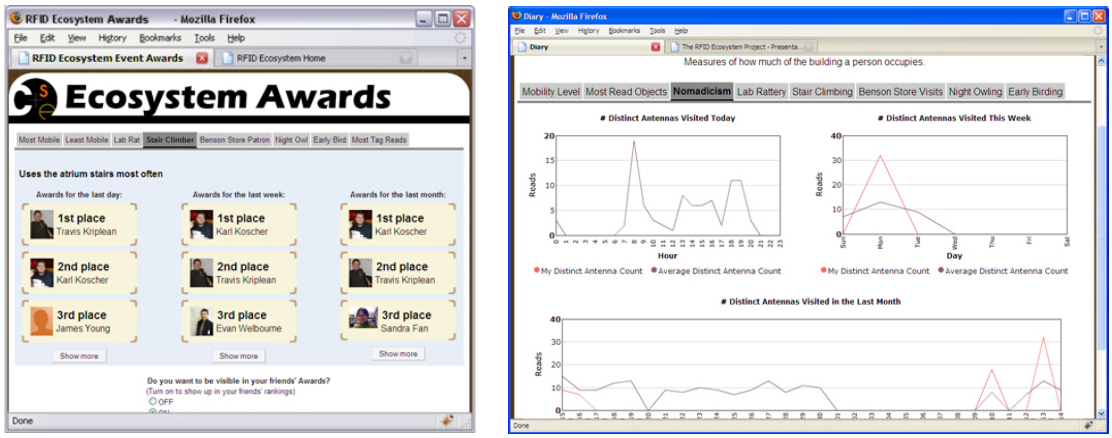


(a) Rfidder displays friend locations **(b) Object Search shows object locations**

Figure 2.9: **Screenshots of the Rfidder and Object Search applications. Both are applications that use RFID tags on people or objects to track current location (or last known location) using TREs.**

Event-Based Applications

We also implemented applications that provide services using simple location events detected over raw RFID data. The first such application, the Event Notifier, allows users to send emails and SMS messages when a person or object enters or leaves a particular location. For example, suppose Alice wants to meet with her advisor but he is currently away from his office. In this case, Alice could have the notifier send her an SMS message when her advisor enters the location (i.e., appears under the antenna) in front of his office. Two other applications track simple events and descriptive statistics over time. RFID Awards (see Figure 2.10(a)) is a game-like application that ranks users by how often they enact some event (e.g., walking up the stairs) or by some other aggregate measure (e.g., how many distinct antennas passed in a day). Similarly, the Diary application (see Figure 2.10(b)) tracks simple events and aggregate statistics over time and presents these measures for review in plots and charts.



(a) RFID Awards game **(b) The Diary application**

Figure 2.10: **Screenshots of the RFID Awards and Diary applications. Both are applications that use simple RFID events detected over time in combination with aggregate statistics to present the user with interesting, high-level information.**

2.3.4 Collecting Ground Truth

To evaluate the our system's ability to track people and objects, we needed some form of ground truth on the locations of tags at various times throughout the study. We collected two forms of ground truth information: survey results and card-key logs. Every weekday during work hours, a custom-built survey system prompted participants via email at random times to visit a survey web application at the soonest possible time. The web-based surveys asked where a participant currently was, what tagged objects he had with him and which antennas he passed on the way to his current location (if any); surveys were designed to take less than 20 seconds to complete. The timestamped survey responses allowed us to compute a sample-based representation of: 1) when and how often a participant was in the building or in a particular room, 2) when and how often a participant carried a particular RFID tag, and 3) approximately when a participant passed a particular antenna. We also administered a slightly longer exit survey to participants at the end of the study which asked a variety of questions on participants' experience including tag wearing, application usage, and privacy.

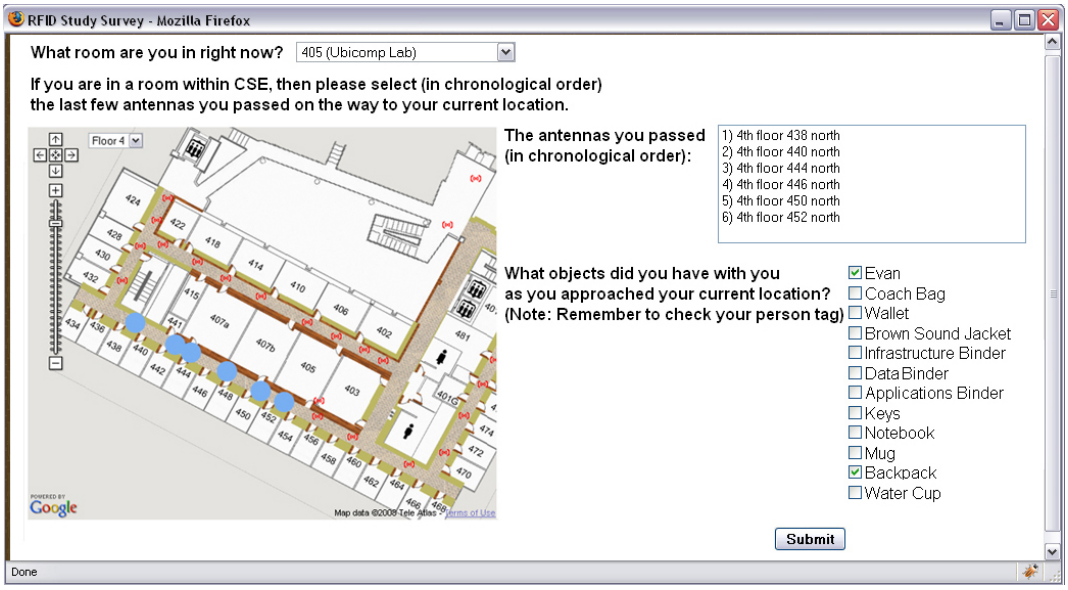


Figure 2.11: **A web-based survey uses a Google map mashup to collect participants' current location, a list of carried tags, and the sequence of antennas recently passed.**

2.3.5 Cleaning RFID Data

It is well known that streams of passive RFID data are noisey and inconsistent [176, 221], with many false negatives as a result of RF interference, limited read range, tag orientation and other intermittent environmental phenomena. As such, we apply some basic data cleaning and compression algorithms to the collected data before analysis. In particular, we use the adaptive window-based cleaning algorithm from the SMURF system by Jeffery et al. [176] to smooth over statistically insignificant gaps in the RFID data, obtaining a smoothed stream of TREs for each antenna. We then compress the smoothed data into STAYs. A STAY is a tuple that represents the length of time a tag stays at a particular antenna and has the following schema:

```
(tag ID, antenna ID, start time, end time, number of reads)
```

Smoothed TREs that occur consecutively are grouped into a single STAY record. Specifically, a tag must leave the read-range of an antenna for at least 1 second in order for a new STAY record to be generated. We found that this threshold works well in our environment but it could be easily changed.

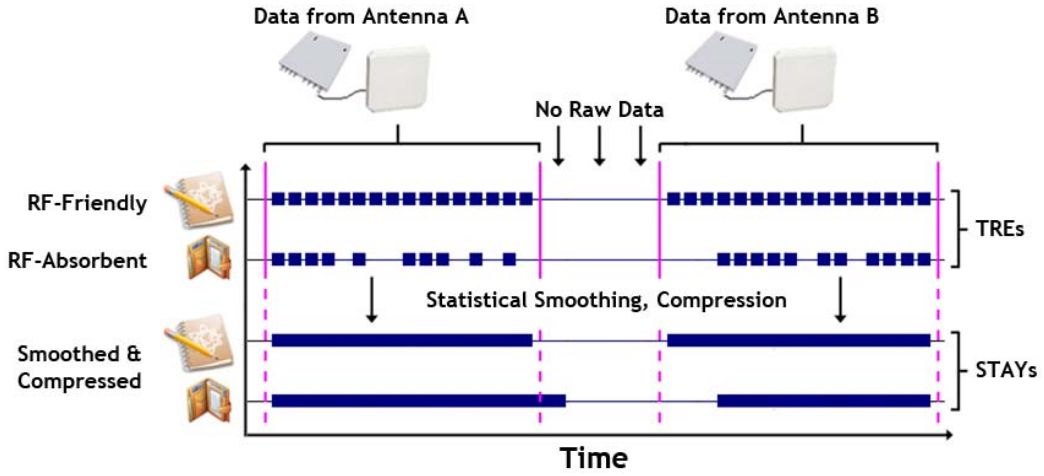


Figure 2.12: **The filtering algorithm transforms intermittent RFID data into smoothed data which we compress into STAY tuples. Note that the filtering algorithm is probabilistic so that the start and end times for the resulting STAYs are only estimates.**

2.4 Field Study Results

In this section, we present the results of our study in terms of survey responses, infrastructure and applications utilization, and system performance.

2.4.1 Survey Responses

Over the course of the study, we administered 2226 surveys and recorded 1358 responses. Though survey prompts were emailed during business hours, responses were submitted at all hours of the day. Figure 2.13 shows the distributions. On average, survey responses took 18 seconds to complete, this does not include the time taken to log in with username and password. Throughout the course of the study there were 7 incidents in which a participant self-reported an error in a survey response. In all cases these errors consisted in reporting that a particular tag was with the participant when in fact it was not. The effect of such errors on our results is that the reported detection-rate for that tag will be artificially lowered (see Section 2.4.3).

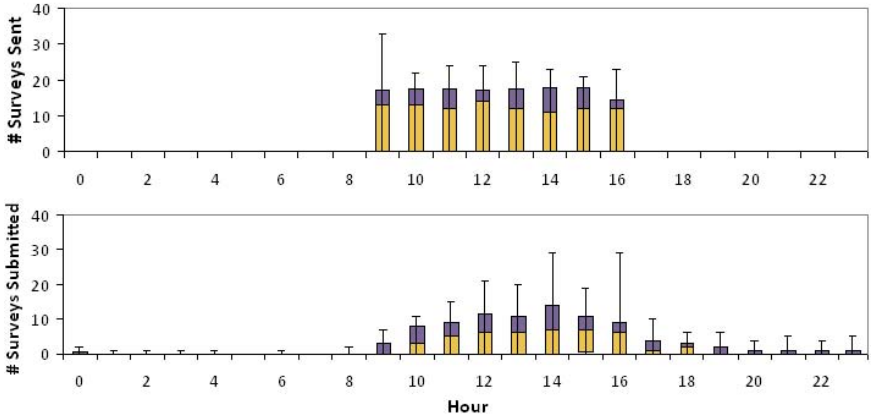


Figure 2.13: **Quartiles for the distribution of the number of survey prompts (Surveys Sent) and responses (Surveys Submitted) by hour of the day over all workdays.**

	Raw	**Compressed**
total data	1.5M TREs (52.95 MB)	38K STAYs (1.3 MB)
average rate	2295 TREs/hour	57 STAYs/hour
max rate	8408 TREs/hour	601 STAYs/hour
min rate	0 TREs/hour	0 STAYs/hour

Table 2.4: **Total RFID data generated during the study and average, minimum, and maximum hourly data rates.**

2.4.2 System Utilization

The first key questions that our study strives to answer are related to system utilization: How much data does a community of users generate and what does the data look like? What are the system utilization trends over time? Are users eager to adopt this new technology? Are privacy concerns a significant barrier to adoption? We answer these questions in this section.

Data Rates

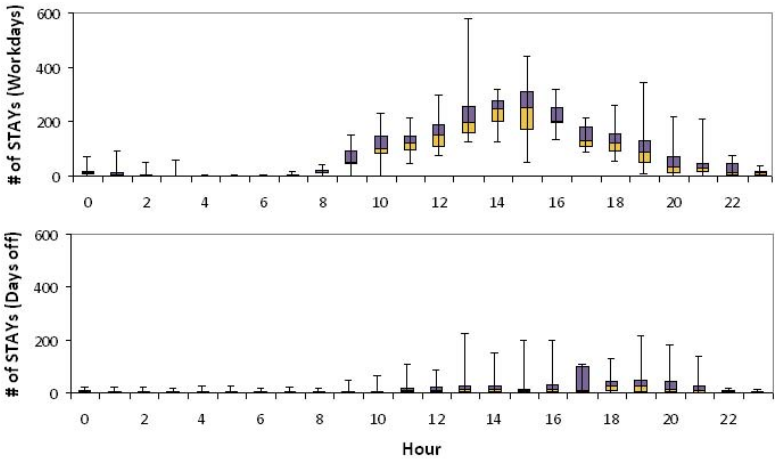
Table 2.4 summarizes the total and hourly data rates generated throughout the field study. Overall, the RFID infrastructure generates a small amount of data and that data can be

nicely compressed. On average, the system produced only 0.8MB of TRE data and a little over 20KB of STAY data per participant during the entire month. These data rates are much less than those generated by other sensor deployments (e.g., audio sensors may sample at 16000Hz, acceleration sensors at 550Hz [49]). As such, RFID data may be potentially easier to manage. However, as we discuss in Chapter 4, post-processing the raw RFID data to infer more detailed location information causes a significant data blow up (easily one order of magnitude).

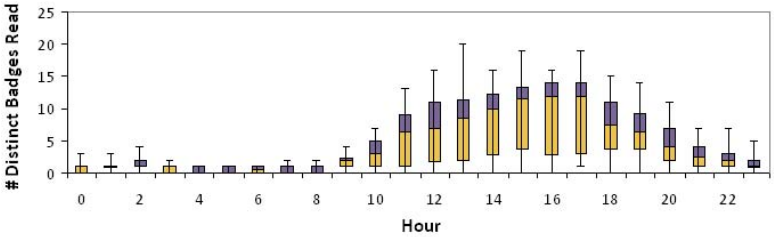
Figure 2.14 shows the overall system utilization in more detail. Figure 2.14(a) shows the quartiles for number of STAYs per hour across all working days and all days off (e.g., weekends and holidays). Figure 2.14(b) shows the number of distinct personal badges read per hour. As expected, the data generated by an RFID infrastructure approximately matches the level of activity in the building. The figures show clear diurnal patterns with activity occurring later in the afternoon on days off. The number of distinct tags and the total data produced during a day also parallel each other.

Figure 2.15 shows the system utilization trends by day throughout the study in terms of total number of STAYs and total number of distinct tags read. These data rates also closely match the normal building utilization. The first day corresponds to November 2nd, 2008. We can see the patterns for the four weeks of the study with dips on weekends and on the November 11th, 27th, and 28th holidays. It is notable that the total number of STAYs again closely mirrors the total number of distinct tags read, suggesting that each tag read may contribute equally to the total number of STAYs generated. However, as we see in Section 2.4.2, many tags are seldom read at all throughout the course of the study, so we can conclude that the smaller number of tags which *are* read each day are responsible for the majority of data generated.

Overall, the activity trends measured by our RFID infrastructure are consistent with earlier mobility studies using WiFi [30, 192].



(a)



(b)

Figure 2.14: **Quartiles over all days for (a) STAYs per hour, and (b) distinct badges read per hour.**

Data on People and Objects

To further characterize the data generated by a community of users we looked at the data for individual participants and for various object types. Among the 67 participants, each carried 1 personal badge tag and an average of 3 additional tags. The most tags carried by any participant was 21, while several participants carried only their personal badge.

Figure 2.16(a) shows a cumulative distribution function (CDF) of the number of STAYs generated by each personal badge, by each object, and by each participant (e.g., the sum of STAYs for that participant's tags). The plot shows that badges were read more frequently than objects, but that the most frequently read objects were detected as often as the most frequently read badges. The plot for participants' shows that the majority (65%) of partici-

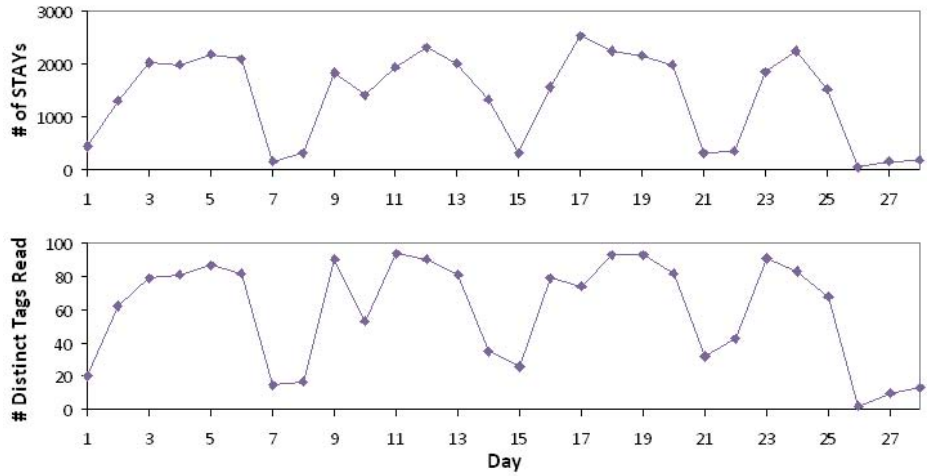


Figure 2.15: **For each day of the study: total STAYs recorded and number of distinct tags read.**

pants produced less than 1,000 STAYs, while 10% of participants produced upwards of 3,000 STAYs. This means that the most active participants were detected moving through the building approximately 3 times as much as the median participant and 30 times as much as the least active participants. The plots for badges and objects show the same trend, although a little less pronounced. There are several reasons for these differences in level of activity between different people and their objects which we discuss below.

Figure 2.16(b) shows a CDF of the average and max length of STAYs generated by badges and objects. As one would expect, we see that participants tend to have slightly shorter average STAY times than their objects. This is because participants often leave objects behind in a location while they move. We also see that the top 10% of both objects and badges in terms of maximum STAY lengths have approximately the same maximum STAY length of 2.5 to 3 days (about the length of a long weekend). Deeper analysis of the data in connection with location survey data showed that these STAYs were all cases where a participant left one or more tags out on his or her desk in range of an antenna for several days at a time. Such events also created the most raw TRE data overall but are easily compressed into STAYs.

The probability density functions (PDFs) in Figure 2.16(c) and (d) show how the type of participant or object affects the amount of data generated. Faculty, staff, and graduate

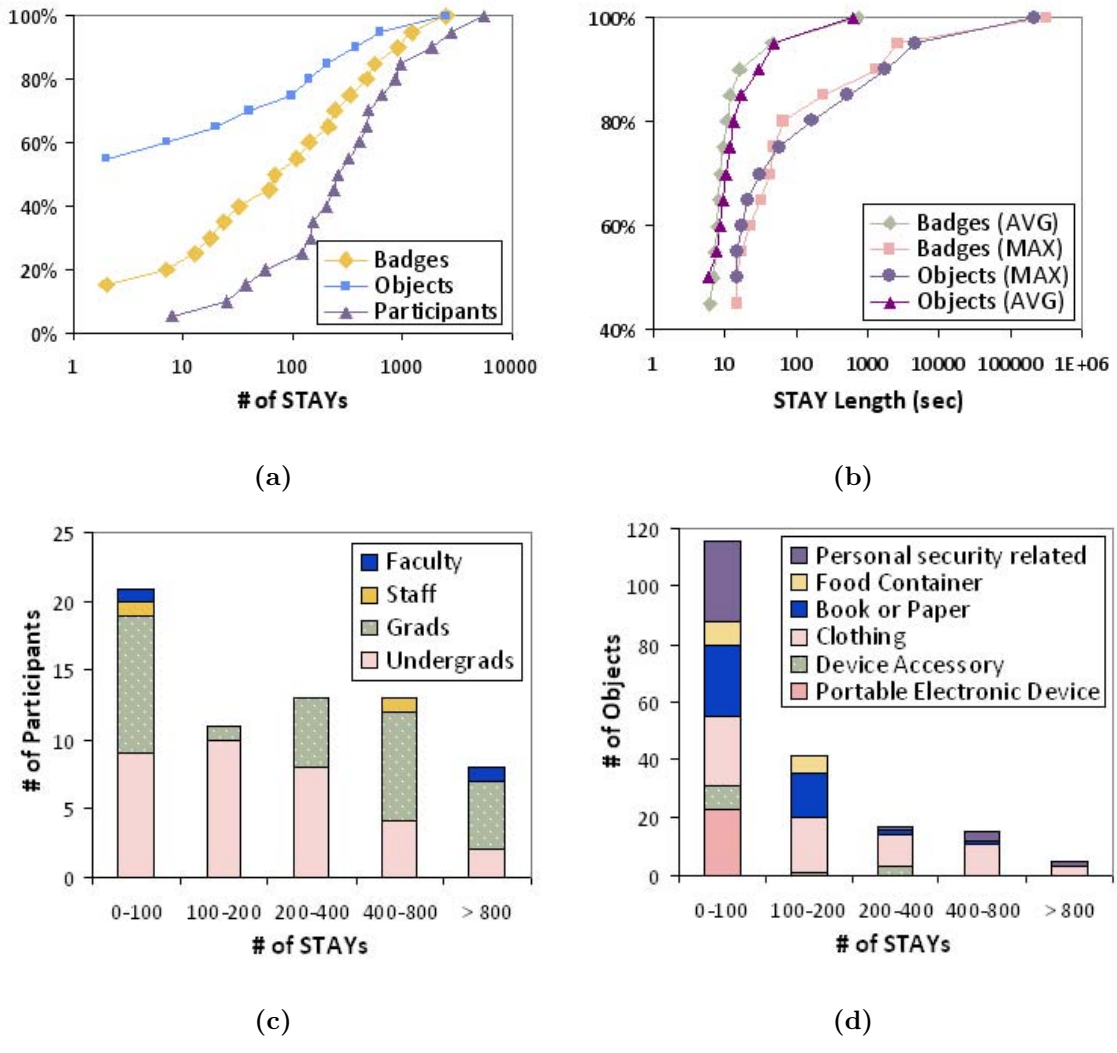
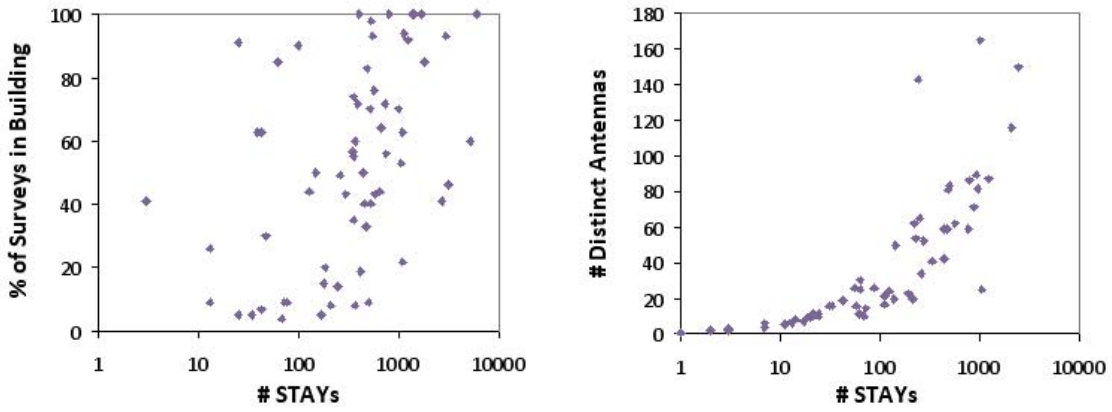


Figure 2.16: **CDFs showing (a) STAYs generated by badges, objects and participants, and (b) Average and max STAY lengths for badges and objects. PDFs showing the number of STAYs generated by (a) different types of people and (b) different types of objects. Note the x-axis has a log scale for plots (a) and (b).**

students tend to fall in one of two categories: they either produce significant amounts of data (more than 400 STAYs) or they produce almost no data (less than 100 STAYs). In contrast, undergraduate students tend to produce either almost no data or a small amount of data (200 to 400 STAYs). These groups tend to separate into large or small data producers because many participants simply wore their badges less often. In the exit survey, participants cited several reasons for this: 41 said they forgot to carry their tag on a weekly basis, 8 said they felt socially awkward carrying a tag, and 8 said they stopped wearing their tag because it didn't work well enough. Another reason for the split in data producers is that some participants just spent less time in the building. However, as we show below, there is a definite correlation between those who produced the most data and those who most actively used the applications. It is also likely that graduate students, faculty and staff produce more data than undergraduates because they occupy the upper floors of the building where there are many more antennas.

For objects, we see again a clear split between large and small data producers in all categories. These differences can partly be explained by how often participants move the objects. However, a second factor also affects the data generated by objects: the type of material. As the PDF shows, most portable electronic devices are metallic and so they are read quite infrequently even though we expect that participants carry them often. Meanwhile, non-metallic objects like clothing, books and papers are read most often. Thus, even with a variety of mounting strategies used across participants, the object material plays a very strong role in a tag's readability.

In summary, there are at least three factors that can affect the amount of RFID data produced by a person or object: mobility, how and when the tags are carried, and the material the tag is attached to. These results confirm our microbenchmarks in Sections 2.2.1 and 2.2.2 as well as the results of the next section. We now study how mobility correlates with system utilization.



(a) **(b)**

Figure 2.17: **The impact of user mobility on total data collected. (a) The percent of "in building" survey responses by a participant generating a particular number of STAYs. (b) The number distinct antennas visited by a participant generating a particular number of STAYs. Note the log scale on the x-axes.**

Person and Object Mobility

Figure 2.17 shows scatterplots that depict the number of STAYS generated by each participant against either (a) the percent of survey responses in which that participant indiciated he or she was in the building or (b) the number of distinct antennas visited by that participant. As expected, there is a clear correlation (with coefficient 0.38) between the approximate amount of time a participant spends in the building and the number of STAYs he generates. We also find that participants who occupy a larger portion of the building, as indicated by a larger distinct number of antennas visited, are likely to generate more STAYs (we computed a correlation coefficient of 0.71). We also examined the relationship between STAYs generated and average length of STAYs to find that there is neither a strong positive nor strong negative correlation (the coefficient is 0.16). The latter result is somewhat surprising and seems to indicate that independent of how much time some tags spend sitting in trackable areas, they are not actually more or less mobile.

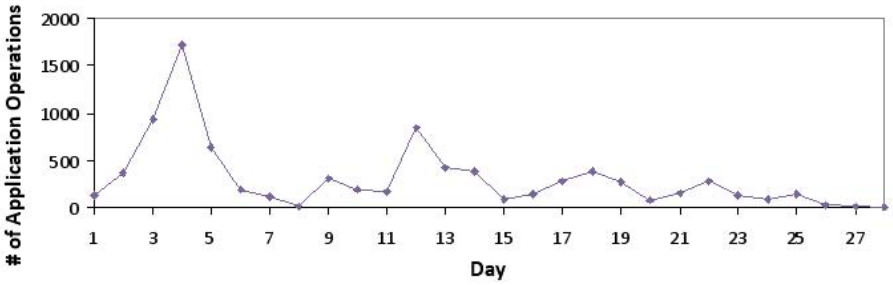
Application Usage

Figure 2.18(a) shows application usage measured in operations (as described in Section 2.3.3). The plot shows that application usage decreased over time. This may be explained in part by the fact that students and faculty had an increased workload near the end of the academic quarter during the second half of the study. In addition, the exit survey revealed that 18 participants who stopped using applications felt that it was too much effort to sign-in to a web page, while 15 others didn't have enough friends participating to keep Rfidder interesting. Although most participants explained that Rfidder and Object Search were appealing only as novelties (only 13 found these applications useful for coordinating activities), these were still the most popular applications. Another trend which emerged is that the amount of data generated by a participant is correlated with that participant's level of engagement with the applications. Figure 2.18(b) plots application usage in operations against amount of data generated by that participant. The most active application users definitely generate the most data, although the overall correlation coefficient is only .37 for these two variables.

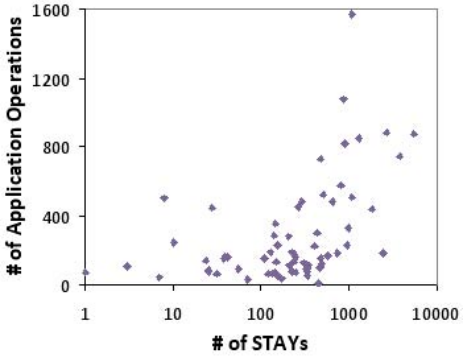
It is interesting to note that participants deleted very little of their data. Out of 67 participants, only 3 made any deletions. Moreover, the 4 deletions that were made correspond to only single STAYs (as opposed to an entire hour, day or trip through the building). Participants explained these operations as "a test to check that the system worked" in the exit survey. In addition, only 7 participants said they removed their tags for privacy reasons at least once throughout the study. This seems to indicate that for the first month of this study in a controlled, familiar setting, participants had little concern over the potential for a privacy breach. It may also be that participants did not understand the potential for privacy breach - even though the risks were carefully explained during the consent process. A more in-depth analysis of participants' reactions to the applications and tools is published elsewhere [345].

2.4.3 System Performance

In addition to basic system utilization, we are also interested in studying the performance of an RFID infrastructure: What is the load distribution across different readers? How



(a)



(b)

Figure 2.18: **Application utilization: (a) over the course of the study, and (b) per participant against data generated by that participant. Note the x-axis has a log-scale for (b).**

well are tags on various types of objects detected in practice? Are tags mounted by users detected as well as those mounted by experts? Are tags durable enough to last for long periods of time in an average user's possession? We address these questions in this section.

Load Distribution

Figure 2.19(a) shows a map of the 3rd floor of the Allen Center that highlights regions where the most STAYs occur. Consistent with previous location-tracking studies [30], certain locations and thus antennas are "hot spots" that generate significantly more data than others; the two hot spots in the map are outside the student store and outside a lab where several of the study participants spend time.

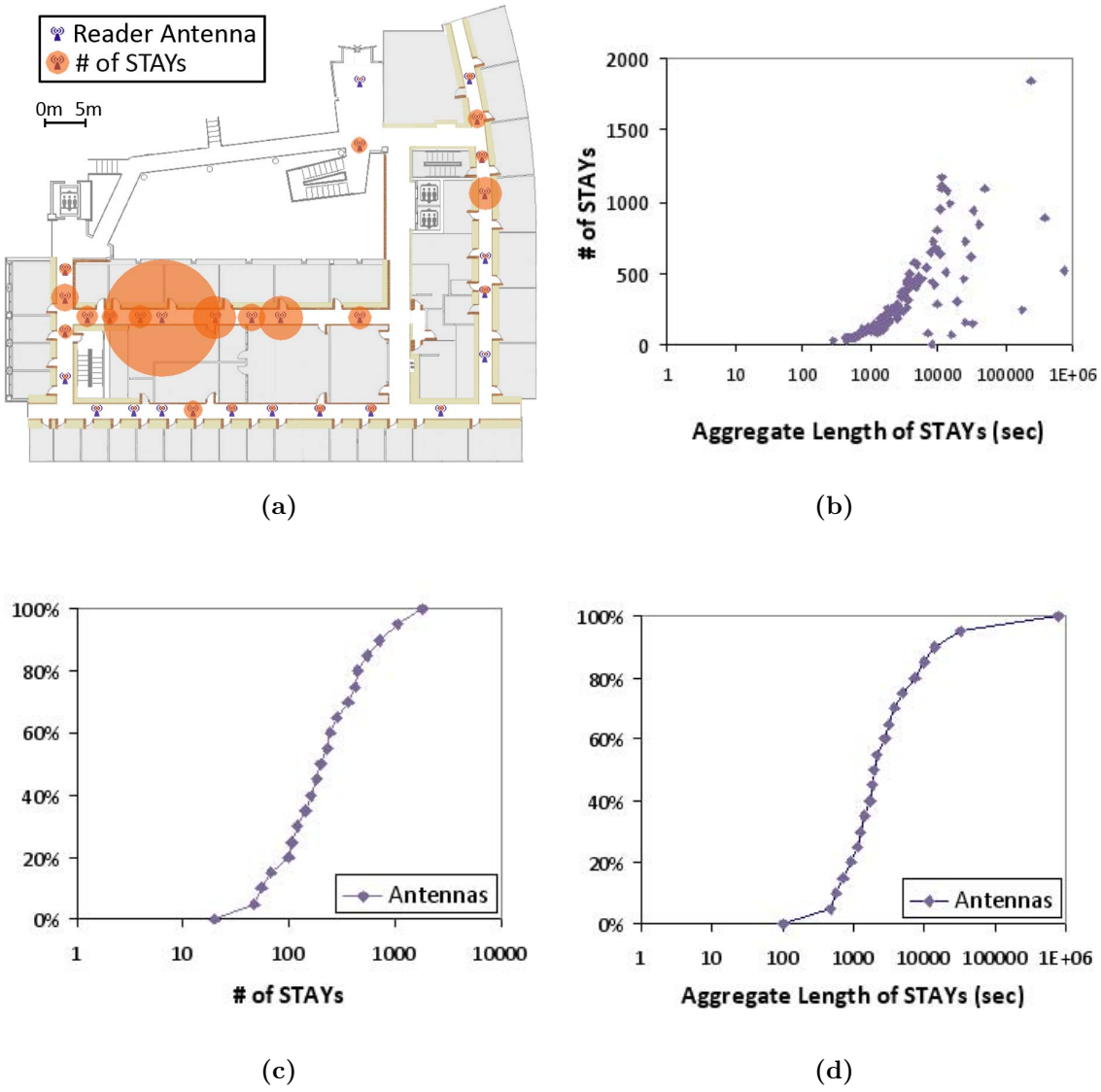


Figure 2.19: **(a) Map of RFID deployment highlighting antennas where the most STAYs occur. (b) Plot of aggregate length of all STAYs against total number STAYs for each antenna. The CDF over all antennas of (c) STAYs registered and (d) the aggregate length of all STAYs. Note the x-axis has a log-scale for plots (b), (c), and (d).**

Looking more closely, we see trends in Figures 2.19(c) and (d), the CDFs over all antennas of total STAYs occurring and total aggregate length of all STAYs. Both curves exhibit clear logarithmic trends. The first CDF shows that there are approximately two orders of magnitude difference between the number of STAY records at the least and most visited antennas. The second shows that there is an incredible four orders of magnitude difference between aggregate length of STAYs. These plots illustrate two different types of hot spot: one is a location that mobile tags pass frequently, and another is a location where tags dwell for very long periods of time. The data in Figure 2.19(b) is correlated with coefficient 0.29 and shows that while a single antenna can have both of these characteristics, many also have a much longer aggregate STAY length relative to the number of STAYs.

Detection Rate

We use the timestamped survey responses to compute approximate read probabilities for tags used in the study. Recall that the 1398 survey responses each include the participant's current location, antennas passed on the way to that location (if any) and the list of tags carried while passing those antennas.

An important challenge when processing the survey data is that participants were sometimes working for a long time at their desk when they got the survey. In those cases, the survey responses correspond to data collected much earlier than the survey timestamp. At other times, participants passed a series of antennas without being detected in which case there is no data corresponding to the survey response. For this reason we chose the following approach to compute the detection rates from the survey data which underestimates the detection rates.

We compute approximate detection rates by counting antenna "hits" and "misses" for each tag. To do so we take each survey response and search backwards from the timestamp to find the first STAY for each carried tag. If a tag's first STAY did not begin within 1 hour of the timestamp then we add N misses for that tag where N is the number of antennas passed. We basically assume that if no data is found within this time window, the participant was not detected at all during the reported trip. If a tag's first STAY did occur

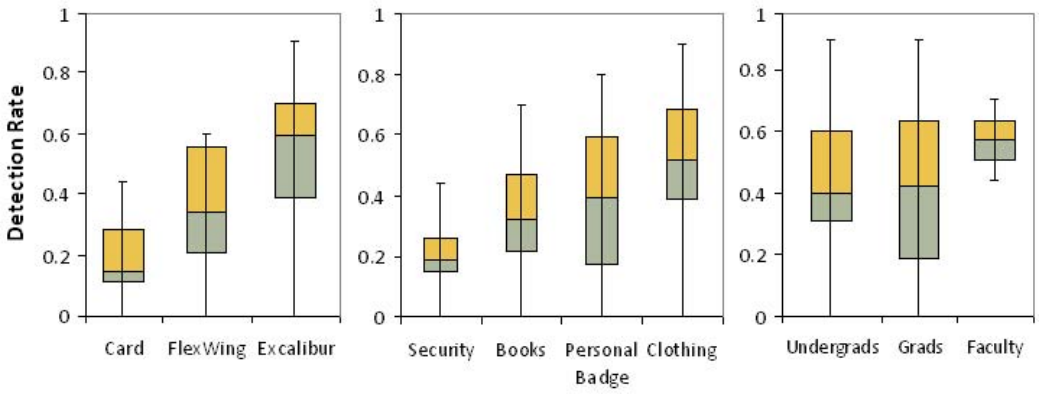


Figure 2.20: **Quartiles for detection rate of (a) different tag designs (b) different types of objects (c) different types of people.**

within 1 hour of the timestamp then for each antenna reported in the survey, we search for a STAY occurring at that antenna at most 15 minutes prior to the last STAY record. We add one hit for each such STAY found and one miss when no such STAY is found. The 15 minute window serves to limit the search for hits to the most recent trip by the participant. It is possible, though not common in our deployment, that a participant passes multiple times under the same antenna within such a short period of time. We compute an approximate detection rate as hits divided by the number of hits and misses. It should be noted that not all survey data is usable because some responses indicated that participants were not currently in the computer science building or that no tags were currently carried. We also threw out survey data for tags that had a total number of antenna hits and misses less then 10 across all survey responses. Overall, we use 956 surveys to compute detection rates. We present this approximate detection rate data below.

Figure 2.20 shows the detection rate aggregated across tag designs, object types, and class of participants. As expected given our micro-benchmarks (see Section 2.2.1), the Excalibur tags achieve the best performance among the three tag designs. These tags also have the largest antenna of the three models. The figure shows significant difference between types of objects. Clothing articles achieve the best performance while security related objects including keys, wallets and purses have the worst performance. Metallic objects do not appear on the graph as there were not enough hits in survey data for these objects. The

differences between objects are due to a combination of object material and tag mounting. We further explore the effect of tag mounting next. Finally, badges exhibit much larger variance than tags attached to objects indicating that different people carry their badges very differently and these choices significantly affect the detection performance. Interestingly, different groups of participants do not systematically get better read probabilities than others (the faculty group was simply smaller than the other ones).

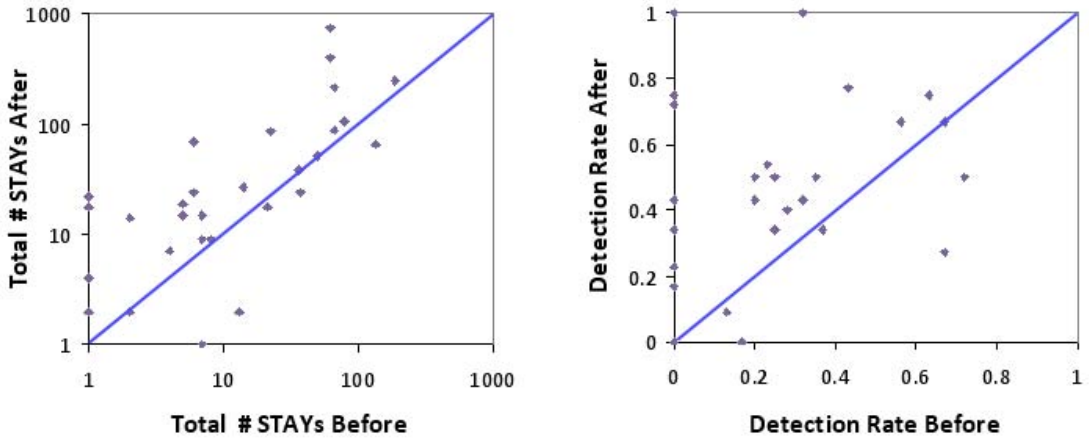
Tag Mounting

To explore the effect of tag mounting by users as opposed to mounting by experts we let participants mount and manage their own tags for the first 3 weeks of the study. In the last week of the study we requested that participants carrying tags with very low detection rates come to our lab to try an alternate tag mounting strategy for a week. Figure 2.21 shows the total STAYs collected and detection rate for each remounted tag before and after it was remounted. The plots show that although some tags remained consistently difficult to read, the readability improved for the majority of remounted tags. This suggests that in a passive RFID-based location system it is not enough to give basic tag mounting guidelines. Either experts must assist the tag mounting process or users must be sufficiently incentivized (possibly through compelling applications) to find an effective tag mounting strategy.

Regarding the durability of tags and the ability to endure use by a variety of participants, we observed a number of events which are summarized in Table 2.5. A total of 6 tags were broken, most of which were unlaminated tags that had been physically torn or creased. One laminated FleXwing was broken after being mangled inside a backpack. Three tags (one of each design) were lost, and 3 PVC cards were returned or thrown away during the study because the participant felt they didn't work well enough.

2.5 Other Observations

The process of deploying a large-scale RFID system in a busy, public building was difficult and time consuming. We learned a great deal about effective deployment practices and found several very helpful insights and strategies for working with key problems. In this section we present key findings on best practices for deployment and their implications



(a) (b)

Figure 2.21: **Plots showing the (a) total STAYs collected and (b) detection rate for each remounted tag before and after remounting.**

	Broken	**Lost**	**Returned**
EX	2	1	0
EX (laminated)	0	0	0
FW	3	1	0
FW (laminated)	1	0	0
PVC	0	1	2

Table 2.5: **Summarizing results for tag durability.**

for others who work with passive RFID deployments. In summary, however, users are willing to adopt passive RFID systems. Exploiting RFID infrastructures in location-aware computing environments, however, raises significant technical challenges primarily due to the low detection rates of tags, especially when these tags are managed directly by users.

2.5.1 Antenna Deployment Strategy

Deploying antennas in many locations was difficult due to health and safety concerns and regulations, building management and facilities issues, and even aesthetic issues. After experimenting with many alternatives for hiding the HP antennas (e.g., placing them above

ceiling tiles or inside plenum-rated enclosures, painting them to match the wall), we found the best strategy was to "hide the antennas in plain sight". The antennas are actually quite non-descript and are often assumed to be yet another piece of a modern building's complex technical infrastructure. Many building occupants didn't notice the antennas until we announced their presence months after they were installed.

2.5.2 Web-Based Diagnostic Tools

We implemented a set of web-based diagnostic tools to help in the early stages of system deployment and debugging. These tools included a Google map mashup of readers, antennas, and live tag readings; system monitors that send SMS updates and alerts to researchers regarding data collection and system status; and data visualization tools that automatically plot aggregate data in a web interface. These tools greatly facilitated our work and expedited both the deployment and debugging processes.

2.5.3 Building-in Privacy Assurances

From the beginning, we designed our study to allow users a large amount of direct control over their own data and over what information is disclosed about them. We also took special care to make sure the user's privacy control and data management interfaces were understandable and usable. This effort proved very helpful when we were recruiting users for the field study. Many participants were apprehensive about the study at the time of consent and immediately became much more comfortable and interested on seeing the privacy controls available to them.

2.6 Summary

In this chapter, we presented the results from a four week study of a building-wide EPC Gen 2 UHF RFID deployment with tens of participants and hundreds of tags. We analyzed the system from the perspective of the data generated as well as the performance and reliability of tags carried on various participants, objects and tags. We found that our RFID deployment, the RFID Ecosystem produces a very manageable amount of data overall,

but with orders of magnitude difference among various participants and objects. We also found that the tag detection rates were relatively low and with a large variance across tags. Many end-users also needed expert help in properly mounting their tags to objects. Overall, passive RFID deployments present a promising sensor infrastructure for a variety of location-oriented applications, but exploiting such deployments in a large-scale computing environment presents considerable technical challenges due to the highly uncertain nature of data generated.

Chapter 3

EVENTS IN LOCATION-AWARE COMPUTING

Location events are at the heart of this dissertation. This chapter establishes a solid framework for understanding and analyzing location-events in both research and commercial applications. In particular, we survey over 400 location-aware applications to identify key application domains, distill a taxonomy for location events, and understand trends in location-aware computing. The results of our survey provide not only a grounds for deeper understanding of location-aware computing, but a foundation for the design of the presented systems and tools in Chapters 4, 5, 6, and 7. The survey methodology, quantitative, and qualitative results are presented in detail below.

3.1 Survey Methodology

To better understand the type and stucture of events in location-aware applications, we performed a systematic review of events in research and commercial location-aware applications. While the survey is not completely exhaustive, it covers a representative sample of the location-aware computing space. For each of over 400 applications, we recorded the events used as well as various additional metadata (e.g., release date, application domain). We analyzed this data in aggregate to draw high-level conclusions about the relative importance of application domains and event types in terms of *frequency of appearance*. It should be noted that this measure favors applications domains that have a lower bar to entry, and as such it skews the results toward these domains and the events that they detect. For example, simple mobile phone applications that once detect when the user enters a place are more common (and hence carry more weight) than complex workflow tracking applications for hospitals. While "normalization" of the results is conceptually difficult, we balance the interpretation with information on market sizes and by splitting applications into smaller sub-categories that target similar users and have similar levels of complexity.

3.1.1 Survey of Location-Aware Applications in Research

We adopted a systematic approach to generating a representative sample of location-aware applications in research. We first identified the computer science conferences in which the most cited papers on location-aware computing appear. For each conference, we collected all available proceedings between January 2000 and January 2010. The conferences and corresponding proceedings are as follows:

- **CHI**: ACM Conference on Human Factors in Computing Systems;
  Proceedings from 2000 - 2009
- **LoCA**: International Symposium on Location and Context Awareness;
  Proceedings from 2005-2007, and 2009 (no symposium in 2008)
- **MobiSys**: The International Conference on Mobile Systems, Applications, and Services;
  Proceedings from 2003-2009
- **Pervasive**: International Conference on Pervasive Computing;
  Proceedings from 2002 - 2009
- **Ubicomp**: ACM International Conference on Ubiquitous Computing;
  Proceedings from 2001 - 2009

For each proceedings, we reviewed paper abstracts and retained papers that were likely to contain content on location-aware applications. We then reviewed all retained papers and for each location-aware application mentioned, we recorded an English description of every location event that application used as well as other metadata (e.g., paper title, conference, publication date). Overall, this process generated a total of 144 papers with applications that accounted for 299 instances of a location event being used. Figure 3.1 summarizes the paper collection process.

3.1.2 Survey of Commercial Location-Aware Applications

As commercial ventures are not aggregated in conference proceedings like research applications, we adopted a different approach to generating a representative sample of commercial location-aware applications. We first identified several freely available databases as primary sources of information on applications and companies:

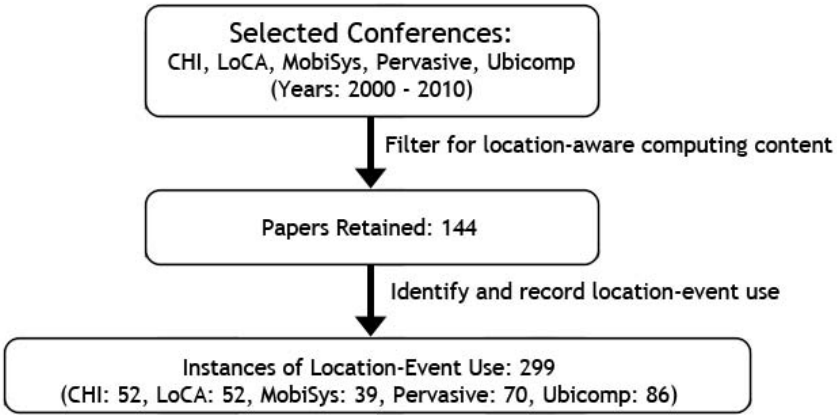


Figure 3.1: **A flowchart illustrating the process we used to generate a list of location events from research papers on location-aware computing.**

- **CrunchBase**: A free tech company database

  `http://www.crunchbase.com/`

- **GISuser Directory**: A web directory of GIS-related organizations

  `http://www.gisuser.com/component/option,com_mtree/Itemid,174/`

- **RFID Journal Search**: A search engine for the RFID Journal website

  `http://www.rfidjournal.com/`

- **RTLS Blog Vendors**: A web directory of RTLS vendors

  `http://www.thertlsblog.com/vendors`

The CrunchBase database was chosen to provide breadth as it contains information and links to over 38,000 companies. While CrunchBase is text-searchable, companies are also grouped by tags. We retained information on companies having any of the following tags: "location", "location-based", "location-based-services", "LBS", "geo", "GPS", "RTLS", "RFID", "NFC", and "near-field-communication". In contrast, the GISuser and RTLS Blog directories provided domain-specific links to companies specializing in location-based applications and services. We retained information on all companies listed in these directories. Searches of the RFID Journal database were intended to retrieve information on applications and companies that use passive and active RFID to obtain location information. The search terms "RTLS" and "location-based" were used to retrieve articles from

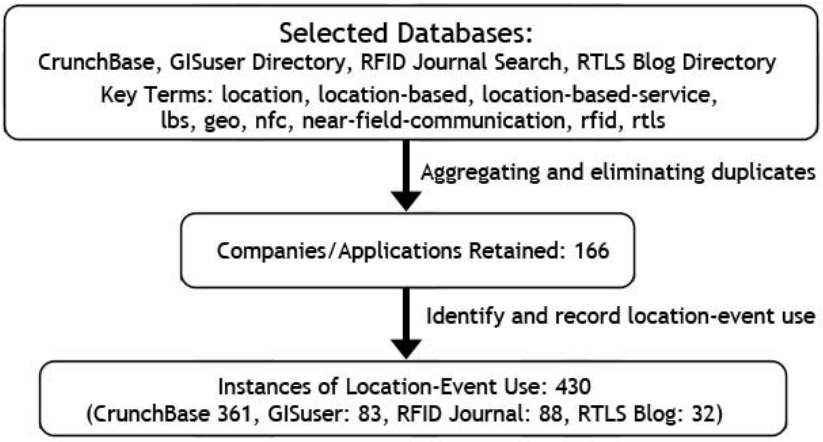


Figure 3.2: **A flowchart illustrating the process we used to generate a list of location events from commercial companies and applications on location-aware computing.**

RFID Journal. Information on all companies and applications in the generated articles was retained. Any overlapping records from the four data sources were merged into a single record.

For each company or application retained, we reviewed the corresponding website and recorded an English description of the application or company as well as associated metadata (e.g., company name, website, application release date). Overall, this process generated a total of 166 companies and applications that accounted for 430 instances of a location event being used. Figure 3.2 summarizes the process we used to collect instances of location event usage in commercial applications.

3.2 Aggregate Survey Results

By carefully reviewing the records collected in the survey, we were able to refine our criterion for classification and comparison of survey data. In particular, we derived classifications for application domains, users, location events, and even basic models for event detection APIs. We then used these classifications to determine the relative importance of events (in terms of frequency of use) for various application domains and target users. This work is described and presented in greater detail below.

3.2.1 Surveyed Application Domains

Across all surveyed applications, we identified 15 rough application domains. We present these domains in this section to highlight the most active areas of development and to provide background information that will support deeper interpretation of the survey results. The domains, with accompanying examples are as follows:

- **Advertising**: Targeted advertising for users
  Example: *VouChaCha*, an application that sends personalized vouchers to users within a 2 minute walk of a music venue [333].
- **Person-to-Person Communication**: Communication between two users
  Example: *IMBuddy*, a location-aware chat client that automatically updates a user's status with their current location [158].
- **Community Communication**: Communication with a community of users
  Example: *BlockChalk*, a mobile application that lets users leave geo-tagged "chalk" messages for others in their neighborhood [41].
- **Design**: Tools for assisting application designers
  Example: *MyExperience*, a mobile experience sampling method application that supports location-triggered surveys [113].
- **Entertainment**: For entertainment purposes
  Example: *FourSquare*, a mobile game that allows users to earn badges by "checking-in" at various locations [109].
- **Health and Fitness**: Support for health, healthcare, and/or fitness
  Example: *UbiFit Garden*, a fitness application that monitors the user's activities and promotes fitness with a garden display [70].
  Note: this category also includes RTLS systems for hospitals.
- **Industrial Tracking**: For tracking in an industrial setting
  Example: *AssetPulse*, an application that helps large businesses track assets throughout their site [21].

- **Local Search**: Search for nearby people, objects or places

  Example: *Outalot*, a mobile application that allows users to find restaurants and other places nearby them [247].

- **Military**: Support for military operations

  Example: *Shooter localization*, an application that identifies the location of a shooter using a distributed network of microphones [332].

- **Navigation and Transit**: Aids for indoor and outdoor navigation

  Example: *Landmark-Based Pedestrian Navigation*, a project that allows mobile phone users to receive directions via arrows superimposed on photographs [151].

- **News**: News and weather

  Example: *BlipSocial*, a mobile application that provides users with locally relevant news items [40].

- **Productivity**: Support for efficiency or productivity

  Example: *Place-Its*, a mobile application that sends users location-triggered reminders [305].

- **Power Management**: Intelligent power management for devices and buildings

  Example: *GPS-controlled Thermostat*, an application that conserves energy keeping by the thermostat setting low as long as the user is far from home [138].

- **Security and Privacy**: Support for a user's or community's security and privacy

  Example: *Visonic*, an RFID-based anti-theft system for warehouses [331].

- **Tourism**: Assistance for tourists

  Example: *TriOut*, a mobile application that suggests nearby places and activities for visitors to North Carolina [320].

Figure 3.3 shows the percent of all applications covered by each of these application domains. Figure 3.4 shows the breakdown by research and commercial applications. Notice that while communication-oriented applications are most common overall, they appear much more frequently in the commercial sector than in research. This is largely due to the recent wave in mobile location-based messaging and social networking applications for high-end mobile phones like Apple's iPhone [170]. It is also interesting to note that while productivity

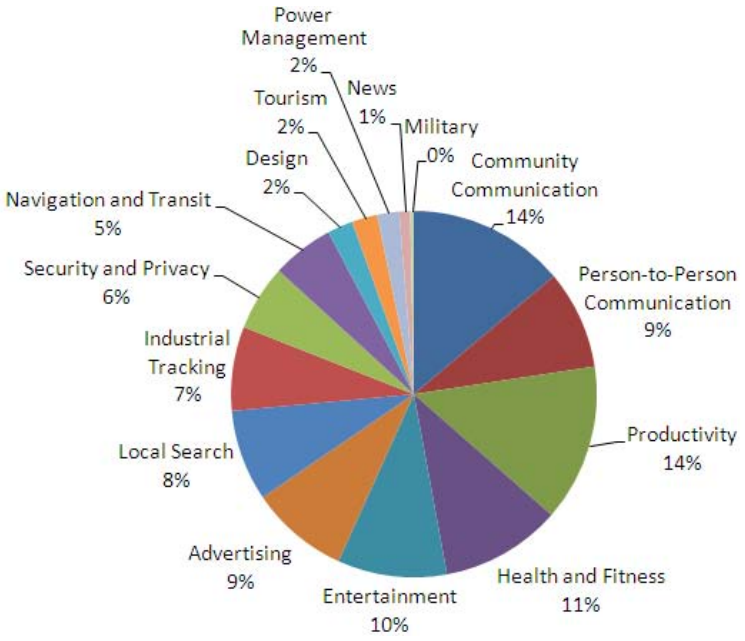


Figure 3.3: **A pie chart illustrating distribution of application domains among the surveyed research applications.**

applications (e.g., location-based TODO lists and reminders) account for 25% of research applications, this category currently only covers 4% of commercial applications. While this difference may suggest a lack of commercial interest in productivity applications, it could also reflect a high level of sophistication and resources required to produce applications in this domain. For example, while Gartner Research cites navigation as the largest consumer location-aware application market in the US [117], this application domain only accounts for a small portion of surveyed events because the detailed map resources required to build such an application is so expensive. Another reason for the discrepancy could be a lag in research-to-market time for productivity applications, such trends are examined further in Section 3.3.

We also separated applications according to the type of user targeted. The user can be a *consumer*, in which case the application is more likely to have personal relevance (e.g., a location-enhanced friend finder). Applications that target *enterprise* are more likely to benefit an organization as a whole (e.g., a location-aware fleet management application for taxis). Overall, we found that 264 were consumer applications while only 46 were intended

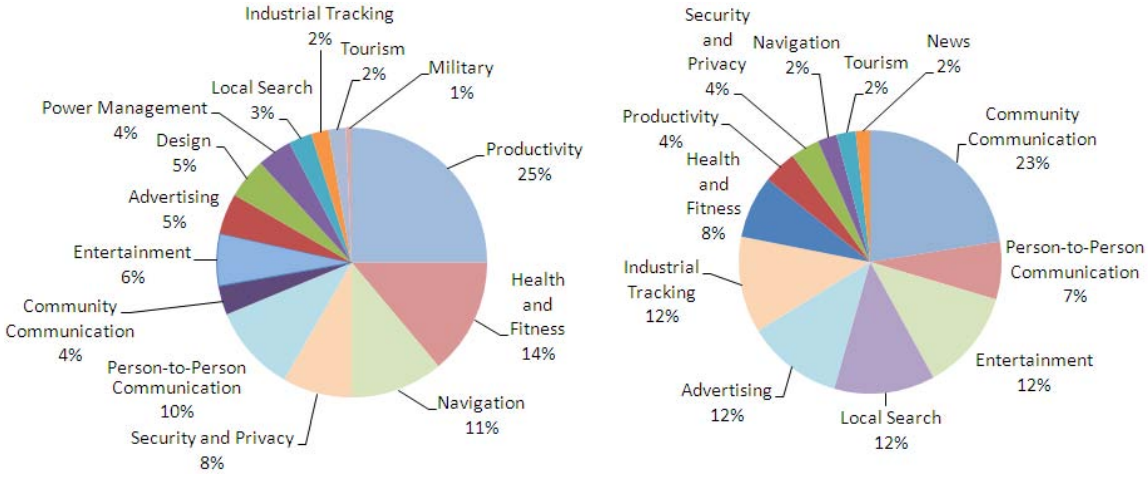


Figure 3.4: **The breakdown of applications by domain (a) For research applications (b) For commercial applications**

for enterprise. This again is likely to reflect the higher level of sophistication and resources required to build enterprise applications. In an effort to present a more balanced view of the application domains, Table 3.1 summarizes some market size information. Lacking precise market information in our survey, we cite some recent statistics from market research firms in the table.

Note that the market for enterprise RFID applications that support tracking and workflow analysis (including supply chain) is significantly larger than the current market for consumer location-based services for mobile phones. This is in direct contrast to the relatively small number of recorded enterprise applications in tracking and RTLS compared to the number of location-aware mobile phone applications. As such, *we emphasize that applications and events that support tracking and workflow analysis are and will become increasingly important.* It is also clear from Table 3.1, however, that the market for location-based mobile phone services is growing quickly, and has indeed more than doubled in 2009 [117]. Thus, while consumer applications may currently be quite simple, the market is rapidly expanding and we may soon see categories of more complex consumer applications as well.

Application Domain	Estimated Size	Predicted Size	Source
Location-based mobile phone services	$1.3B in 2008	$8B in 2011	Gartner, 2008 [116]
Tracking, workflow w/RFID	$5.56B in 2009	Over $25B in 2019	IDTechEx, 2009 [166]
Tracking, workflow w/active RFID	Over $600M in 2009	$6.74B in 2019	IDTechEx, 2009 [163]
RTLS	Over $153M in 2009	$2.6B in 2019	IDTechEx, 2009 [164]
Auto and mobile navigation	N/A in 2009	$26B in 2015	Strategy, 2009 [313]
All Context-Aware Computing	N/A in 2009	$12B in 2012	Gartner, 2009 [118]

Table 3.1: **Estimated current and predicted market sizes for various application domains as determined by various market research firms.**

3.2.2 A Taxonomy for Events

Through a preliminary review of the surveyed location events, we sought to derive a location event taxonomy that classifies events according to primitive spatio-temporal relationships and common compositional techniques. Such a taxonomy is needed in order to identify shared events among applications and to allow comparisons of applications across domains. In our review we found it useful to consider only events that are *meaningful* to end-users. That is, events that could be understood and directly valued by average users. For example, a person entering a room is meaningful, but the recent Fourier transform of a sensor signal is not. We also categorized meaningful events as either *complex* or *simple*. Complex events are relationships among two or more subjects (e.g., "Alice takes a coffee break") and can be decomposed into simpler, meaningful *sub-events* (e.g., "Alice left her office, is in the kitchen, and has her coffee mug"). Simple events have two subjects in some basic and instantaneous relationship that cannot be further decomposed (e.g., "Alice is in the kitchen").

We noted all uses of complex and simple events, and then recursively decomposed all complex events into simple events, making a separate note on how the sub-events were composed (e.g., by sequencing). Arguably the most fundamental simple event is a person or object being **at** particular location coordinates. We clustered all other simple events into into groups having a similar description to derive three more *primitive events*: **inside**, **with**, and **near**. Together with their negations, this makes a total of seven primitive events, which we present in the list below. Note that in the primitive event descriptions, we use the term "mobile entity" to refer to a person or object, and "entity" to refer to a person, object, or place.

- **at**: a mobile entity is at some location coordinates
- **with**: some mobile entities are next to or touching each other
- **without**: some mobile entities are not next to or touching each other
- **inside**: a mobile entity is inside a place
- **outside**: a mobile entity is outside a place
- **near**: some entities are within a distance k of each other
- **far**: some entities are beyond a distance k of each other

Review of the complex event decomposition process revealed that events are typically composed in three ways: conjunction, duration, and sequencing. Conjunction combines events that occur nearly simultaneously into a more complex event. For example "Alice is in the lounge with Bob". Duration can be used to extend a primitive event in time, for example: "Alice stays in the lab for 10 minutes". Finally, sequencing composes events as a sequence in time. As an alternative to these methods, machine learning techniques have been used to derive higher-level information (e.g., current activity) [252, 254] from low-level events. However, as we discuss in Chapter 8, these techniques are typically only for identifying events that are pre-defined by a model or by labeled data.

To estimate the relative importance of each type of event, we counted every occurrence of that event type being used directly by an application or user. For example, if the event was "user exits workplace" then we incremented the count for "person exits place", but not for either of the composing sub-events "person in place" or "person outside place". Table 3.2 shows the most commonly used events over all surveyed applications by percent of total event usage. We also collapsed people and objects into tracked entities X and Y to obtain a "compressed" list of events as shown in Table 3.3.

The tables show that by far the most common events concern a person or object being inside or near a particular place. This makes sense because a majority of communication-oriented consumer applications are focused on sharing one's current location. Local search applications like Yelp for Mobile [354] also drive up the frequency of the "places near person" event in commercial applications. The event in which a person or object moves through a

Event	% of Occurrences	% of Research	% of Commercial
1) person inside place	19%	31%	10%
2) person at coordinates	14%	16%	13%
3) person moves on path	6%	5%	7%
4) object at coordinates	6%	5%	7%
5) person with object	4%	3%	5%
6) places near person	6%	2%	10%
7) person exits place	4%	4%	4%
8) person near place	4%	4%	4%
9) object inside place	4%	2%	5%
10) person near person	3%	2%	5%
11) person enters place	3%	6%	2%
12) person outside place	3%	5%	2%
13) object exits place	3%	0%	5%
14) person with person	2%	2%	2%
15) object enters place	2%	0%	3%
16) object outside place	2%	0%	3%
17) velocity	1%	2%	1%
18) trajectory	1%	1%	0%
19) object near person	1%	1%	1%
20) person orientation	1%	1%	1%

Table 3.2: **The most common location events ranked by frequency of use in all surveyed applications, in research applications only, and in commercial applications only. See Table 3.3 for how these events can be abstracted and distilled into the 10 most common primitive events.**

sequence of locations is also very common. These path-oriented events appear frequently in enterprise applications that track people and assets through some process (e.g., patients moving through a hospital) as well as in consumer entertainment (e.g., a treasure hunt style games). Enterprise applications also frequently detect sequences of conjunctions like "patient is in the waiting room, then patient is with a nurse in the lab".

Overall, the tables show that the seven primitive events are the most frequently used events across all applications. They also show that these events are frequently composed using sequencing, this trend is even more pronounced when looking at enterprise commercial applications only. Conjunction does not show-up in the top events, but also appears frequently as a composition technique in the small set of high-value enterprise applications such as the hospital RTLS described in the introduction.

Event	% of Occurrences	% of Research	% of Commercial
1) X inside place	24%	34%	16%
2) X coordinates	22%	22%	22%
3) X near Y	18%	11%	25%
4) X exits place	9%	6%	11%
5) X with Y	8%	7%	9%
6) X moves on path	7%	6%	8%
7) X enters place	6%	7%	6%
8) X outside place	6%	6%	7%
9) X velocity	2%	3%	1%
10) X trajectory	2%	2%	1%

Table 3.3: **A compressed version of Table 3.2 in which people and objects have been collapsed into a single variable X or Y. This emphasizes the most important core spatial relationships (e.g., inside, near).**

3.2.3 Complexity of Applications

Tables 3.2 and 3.3 show overall event use, but they don't show how different types of applications make use of different sets of events. For example, co-location events (e.g., person with object) are ranked lower overall, but are the most frequently used events in home and hospital healthcare applications that track activities (e.g., "Alice is taking her medicine") and workflows (e.g., "the patient met with the nurse and then the radiologist"). Similarly, some more complex applications use a much larger set of events while others use only one event repeatedly.

Figure 3.5 shows the complexity of each application domain in terms of the minimum, maximum, and average number of events used by applications in that domain. While there is substantial variation, the heavily enterprise-oriented domains of Industrial Tracking and Health and Fitness (which includes numerous hospital RTLS applications) are most complex overall. In addition to using the most events, applications in these domains use events that are more complex on average, often including sequences and multiple conjunctions. Figures 3.6(a) and (b) breakdown complexity of commercial applications by consumer and enterprise applications respectively. Here, it is even more pronounced that enterprise applications are more complex, with enterprise Security and Privacy applications also presenting

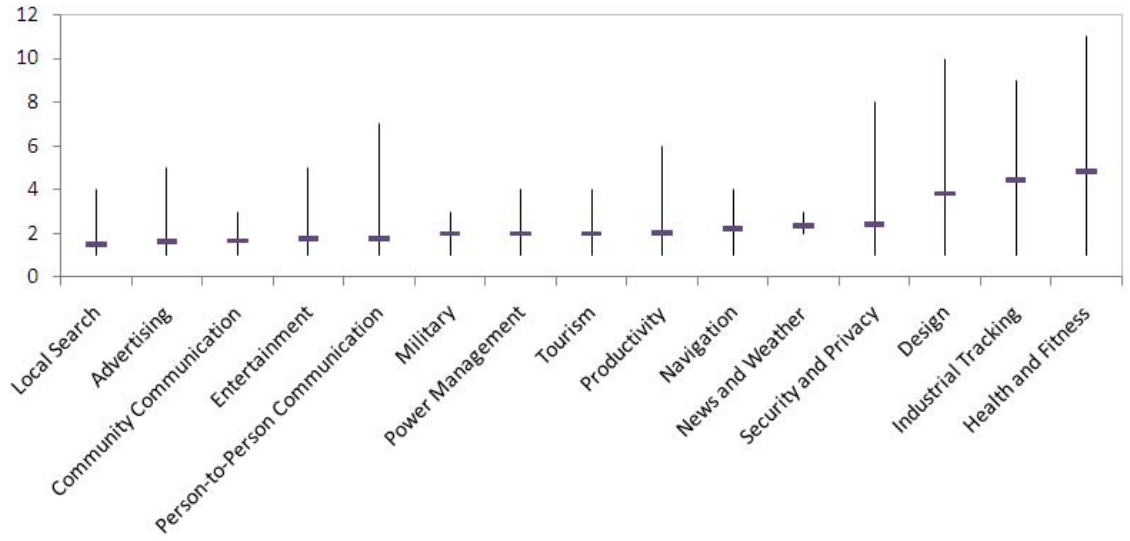
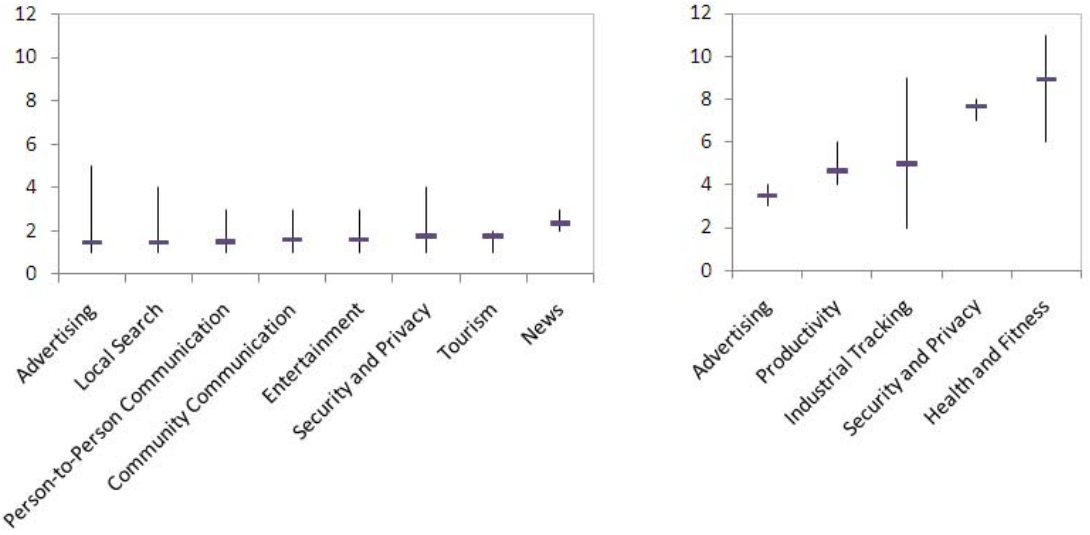


Figure 3.5: **A chart showing the complexity of application domains measured in minimum, maximum, and average number of events used by applications in that domain.**

a high complexity from events that detect theft and monitor the safety of detailed industrial processes. These plots also show that the complexity of most consumer applications is currently relatively low, though it remains to be seen how the consumer location-aware computing market will evolve as discussed in Section 3.3.

3.2.4 Event API Models

In most cases, applications used either a *push* or a *pull* model for obtaining updates on new event occurrences. We call APIs that support the push technique *subscription* based APIs, because the application subscribes to receive updates on new events whenever they occur. For example, a reminder application that automatically alerts the user when she leaves home without her purse would use the subscription model. APIs that support the pull model are called *query-based* because the application polls to receive new information on events. Applications that let users search for nearby restaurants matching some criterion are most often query-based. A third model combines subscription and querying. Hospital RTLS applications may use this mixed type of API to support alerts as well as offline analysis of patient flow.



(a) (b)

Figure 3.6: **The complexity of application domains measured in minimum, maximum, and average number of events used by applications in that domain (a) For consumer commercial applications (b) For enterprise commercial applications**

There has been a significant amount of discussion in the literature concerning the benefits and drawbacks of these models. Many argue that the query-based model is preferable for privacy reasons. For example, the Reno location-based messaging application made users explicitly query for location in an effort to provide the subject with greater control over disclosure of his or her location [299]. Alternatively, some commercial applications like Google Latitude [128] rely on the simplicity of the subscription model to provide a seamless user experience. Other applications like Hospital RTLS systems need event subscriptions to provide the most up-to-date information and alerts when they are needed. As such, both models have their merits and the event detection middleware we present in Chapter 7 supports both.

3.3 An Historical Perspective on the Evolution of Location Events

Although military location systems (e.g., Radar, Sonar, IFF, GPS) have been used for navigation and vehicle tracking since the 1940s, the first experiments in civilian location-aware computing didn't begin until the late 1980s (see Figure 3.7). At that time, research systems

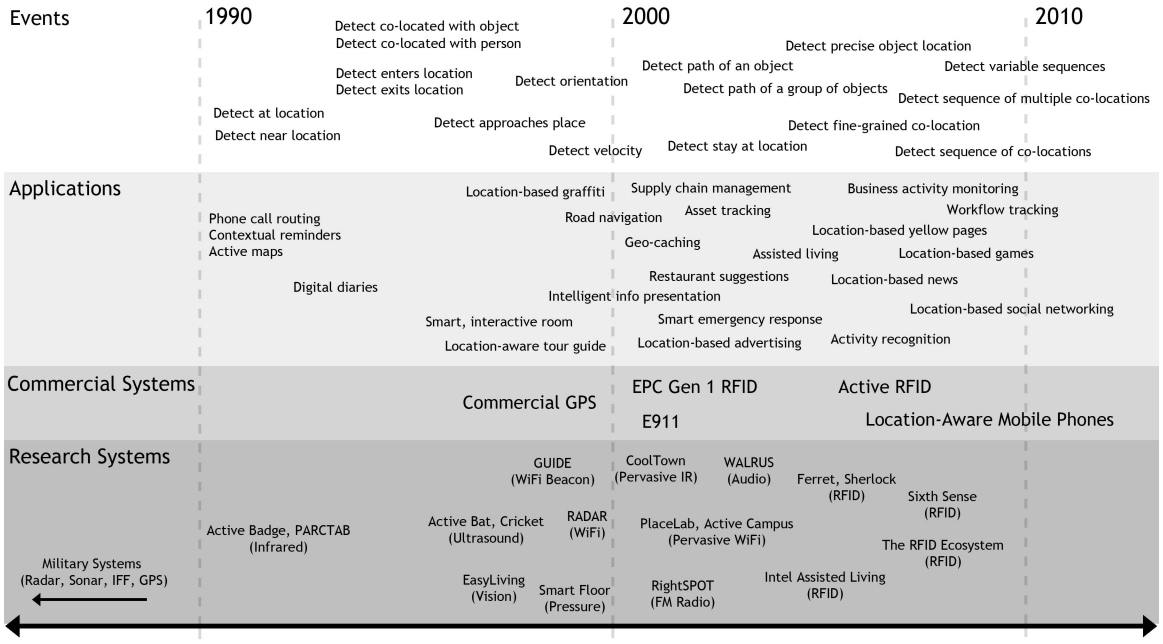


Figure 3.7: **A timeline showing the evolution of location events in parallel with the progression of location-aware computing. The displayed technologies and applications represent only a small sample of the entire location-aware computing space.**

like the Active Badge [339] and Xerox's PARCTAB [340] were using infrared beacons and badges to detect when a person was **at** or **near** a particular indoor location. These simple location events were used to support a variety of novel consumer and enterprise applications (e.g., location-based reminders, location-based routing of phone calls, digital diaries) that remained popular throughout the 1990s. However, research during this formative decade was primarily focused on refining the capabilities and properties of location sensors. While some projects aimed to increase precision [146], others sought to lower costs [26] or reduce obtrusiveness [246], some even explored the alternate model of proximity-based location systems [34, 64]. These projects laid the groundwork for the rapid proliferation of location-based services in the next decade.

The commercialization of GPS in 2000 led to a wave of location-aware navigational aids for consumers and enterprise. These applications continued to use only simple events concerning a user's position at or near a particular location. Mass adoption drove dramatic

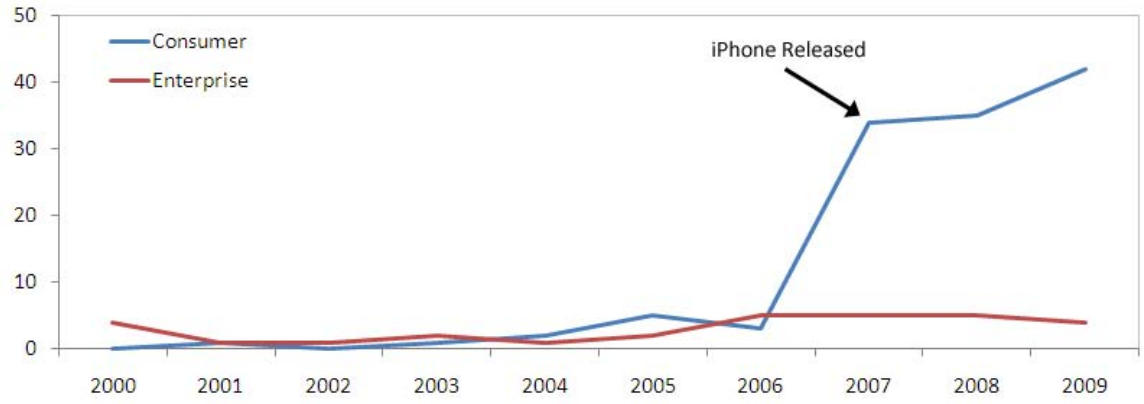


Figure 3.8: **A chart showing our survey data on the number of new commercial location-aware applications released each year targeting either consumer or enterprise users.**

improvements in GPS technology and inspired research into increasingly pervasive location services for mobile devices [134, 289]; the E911 initiative also contributed to the proliferation of location-enabled devices. At the same time, a new low-cost RFID standard (EPC Class-1 Gen 1) was driving innovation in the supply chain management domain. Here, systems were beginning to use RFID to track tagged assets as they move throughout a supply chain. While most work on these systems focused on compressing and warehousing massive RFID data sets, the events of interest were the first to include sequences of locations (e.g., paths through the supply chain). Moreover, these vast RFID deployments were the first to consider large-scale tracking of objects rather than humans.

The advent of powerful, user-friendly mobile phones (e.g.iPhones, smart phones) finally brought location-aware computing to the mainstream consumer market. Figure 3.8 shows the number of commercial location-aware applications released each year for consumer and enterprise users. Notice that the arrival of the iPhone and its app store in 2007 marks a dramatic increase (more than 4x) in consumer application releases per year. In the last three years, mobile location-aware services usage has also seen exponential growth - with the number of subscriptions doubling, to over 18 million in 2008 alone [116]. A majority of mobile location-aware applications use the at or near events in conjunction with existing databases to automatically filter information on restaurants, entertainment, news, weather and ad-

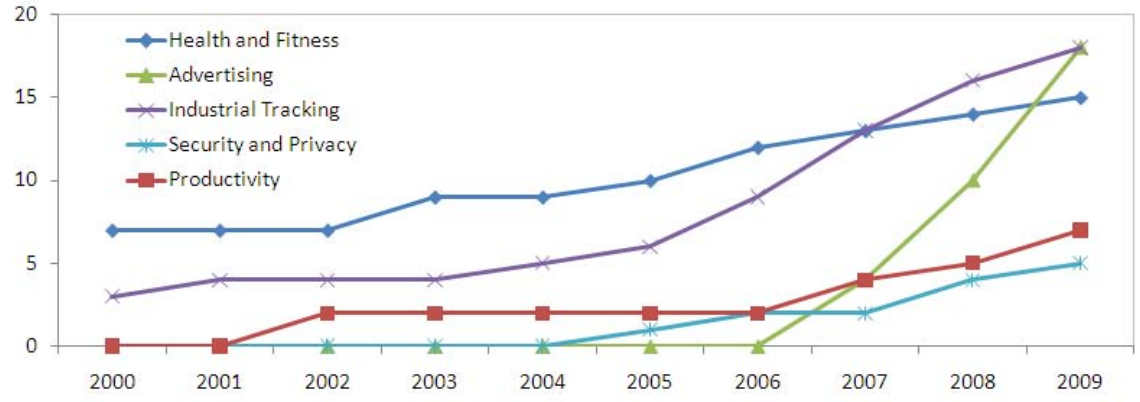


Figure 3.9: **A chart showing our survey data on the total number of companies actively producing commercial location-aware applications for enterprise users in high complexity application domains.**

vertising. In addition, mobile social networking applications are now detecting co-location events (e.g., "I'm at lunch with my friend Alice"). As this market rapidly evolves, applications with more complex events are beginning to emerge. For example, SCVNGR [291] uses sequence events to detect a person's progress on a treasure hunt.

The enterprise market for location-aware computing has shifted heavily toward real-time location systems (RTLS), many of which use active RFID chips. Figure 3.9 shows the growth of the enterprise market in terms of the total number of companies producing location-aware applications for enterprise each year. These systems are rapidly being deployed at sites like hospitals and law offices that need to track assets and monitor complex business workflows. Like earlier supply chain applications, these applications detect events consisting of sequences of locations. However, RTLS applications are beginning to use even more complex events such as sequences with multiple paths, and which contain multiple instantaneous events at each step in the sequence (e.g., "a patient leaves the hospital after meeting with a nurse and an electrocardiogram machine, and then a cardiologist"). We can see the significant complexity of these enterprise applications in the Figures from Section 3.2.3. Support for applications with such complex events is likely to become increasingly important as RTLS becomes more widely adopted and integrated.

Future trends are evident in recent research on location-aware computing. As commercial applications have established a pervasive location infrastructure for the future (e.g., mobile phones, GPS, GSM, WiFi), researchers have shifted in focus from sensors to algorithms that can extract higher-level information from location traces. For example, there has been a substantial amount of work on algorithms that extract a user's significant places from a collection of location traces [150]. Other work has applied machine learning to location data in order to learn frequent transit routes and predict destinations [195]. More recently, some researchers have used co-location information between a person and objects to detect high-level activities (e.g., "Making coffee", "Brushing teeth") [51]. While such algorithms and applications are not yet commercial, it's likely that they will be. As such, we are likely to see a continuing trend toward increased complexity in location events.

3.4 Summary

In this chapter, we have presented the methodology and results of an extensive survey of events in both research and commercial location-aware computing applications. The survey enabled us to identify key application domains as well as to derive a taxonomy for location events. This framework in turn led us to a deeper analysis and comparison of location-aware applications in terms of complexity and popularity. We found that while all application domains are growing and changing, commercial applications for enterprise users remain the most complex and demand support for a variety of primitive events and their composition using sequencing and conjunction. Consumer application domains are also rapidly expanding and research trends suggest that while current applications are simplistic, future consumer applications may be more complex as well. Throughout the remainder of this dissertation, we will refer to the application domains, event taxonomy, and trends identified in this chapter.

Chapter 4

EVENT DETECTION SUBSTRATE

While Chapter 2 showed that EPC Gen 2 RFID could be successfully applied in an everyday setting, many questions remain as to the practicality and usefulness of passive RFID technology for sophisticated location-aware applications. Indeed, non-trivial applications need support for a diversity of complex events such as those presented in Chapter 3. Detecting such events is challenging because the detection system must be extremely flexible and capable of coping with sporadic and imprecise data from an unreliable, non-optimally distributed RFID deployment. In this chapter we present and evaluate an event detection substrate that addresses these challenges by creating a layer of Bayesian inference, modeling, and probabilistic data management between sensors and applications. At the heart of this substrate is a new probabilistic complex event detection engine called Lahar [211, 273]. While Lahar itself is an evolving research prototype that supports precise detection of a small set of events, it is not a contribution of this dissertation. Instead, we show how Lahar can be adapted and extended for use in detecting a larger set of location events, including the most common events from the taxonomy presented in Chapter 3.

4.1 Probabilistic Reasoning About Location

Tag read events from a passive RFID system create sporadic streams of coarse-grained location information. As such, a system can at best maintain an approximation of an RFID tag's true location. However, by reasoning probabilistically on RFID readings given additional information, a system can produce a continuous and often more precise stream of location estimates. For example, better estimates can be achieved if a system knows how a tagged person or object is likely to move (e.g., average velocity) and how an environment constrains that movement (e.g., the location of walls, doors). Similarly, knowledge of an RFID tag's behavioral characteristics (e.g., average read range and rate) can be used to

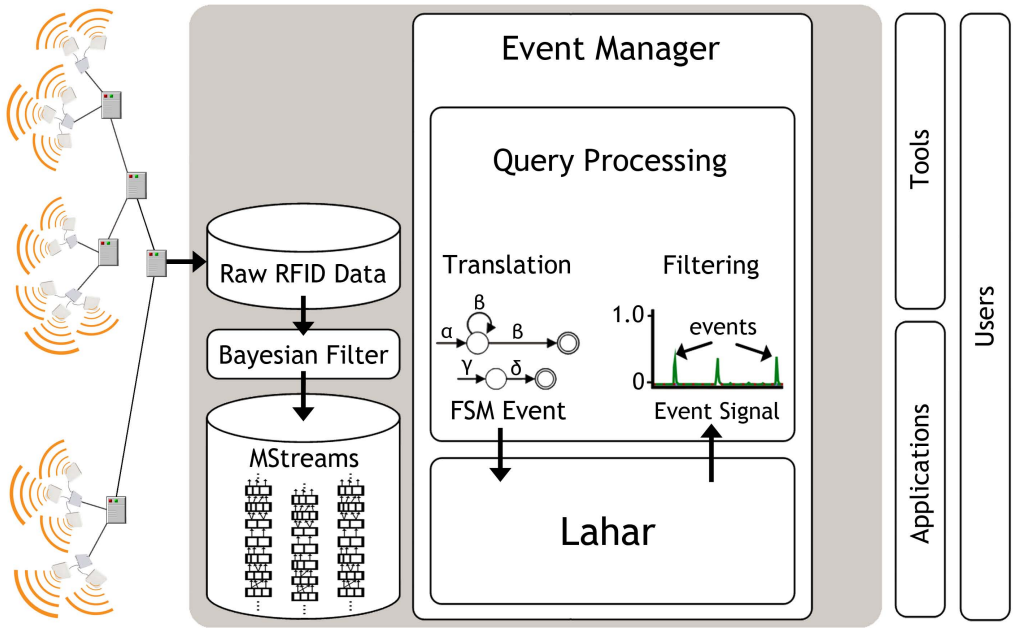


Figure 4.1: **In this chapter we focus on the event detection portion of the system.**

refine location estimates. Even correlations between sequential estimates provide useful information about an entity's current trajectory or velocity. There are a variety of temporal models and probabilistic inference techniques that provide a principled framework for this type of reasoning [282].

In this section, we present a high-level overview of temporal modeling and show how several probabilistic inference techniques can be used to produce a continuous, more precise stream of location estimates from a sporadic stream of RFID tag read events. These models and techniques do not represent a contribution of this dissertation, they are established elsewhere [111, 198, 211, 92, 260, 282] and many tools exist for working with them. However, their role in the event detection substrate is fundamental, and it is instructive to describe how we apply them to RFID data to facilitate event detection.

4.1.1 Temporal Models for Location

Many temporal models have been developed in the field of Artifical Intelligence [282]. Most have the same basic structure and support the same types of inference through similar algorithms. A generic temporal model consists of a sequence of *time slices*, each containing

a set of random variables that describe the current state, some of which are *observable* and some of which are *hidden*. In a model of location, a single hidden variable L_t represents the true location of some person or object at time t, while the observable variables are the location sensor readings (i.e., RFID tag reads) S_t. The time interval between slices can vary, but a model that supports a more continuous stream of location estimates will have a fixed interval of at most a few seconds. Most temporal models are also first-order, *stationary Markov processes*, meaning that the state at time t is determined entirely by the state at time $t-1$, and that the laws governing state transitions do not change. As such, these models may describe state transition (e.g., location transition) with a conditional probability distribution $P(L_t|L_{t-1})$. Sensor readings are incorporated by way of a sensor model, another conditional probability distribution that describes the probability that a particular sensor reading S_t is observed given some state of the world L_t. Finally, an initial prior probability over all possible states must be specified to initialize the process.

Models with this structure lend themselves to several forms of inference. The first is *filtering*, which computes a posterior distribution over the current state given the sequence of all sensor readings. Filtering provides us with a probability distribution over all possible locations at each time slice. The second form of inference is *smoothing*, which moves backward from the present state to update past state estimates using all sensor data available up to the present. Not only does smoothing refine the location estimate at each time slice, but it can extract correlations between one time slice and the next that capture the likelihood of transitions from one location to another. Another common form of inference is to *find the most likely explanation* for a sequence of sensor readings. This type of inference computes the most likely path a person or object has taken given the sequence of available sensor readings.

Each of these inference tasks is supported by one or more algorithms for a given model. Filtering is performed using some variant of the *forward* algorithm, while smoothing is performed with some variant of the *backward* algorithm. When filtering and smoothing are performed together, a hybrid algorithm called the *forward-backward* algorithm may be used. It should be noted that true smoothing cannot be used in real-time on incoming tag read event streams because it would need sensor readings from the future to refine the present

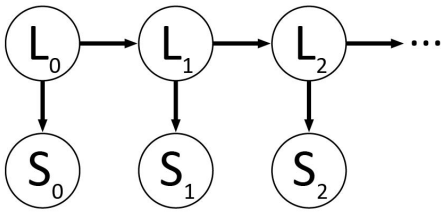


Figure 4.2: **A Hidden Markov Model for location includes a sequence of time slices, each having one hidden variable (L) for a person or object's true location, and an observable variable (S) for sensor readings.**

state. While smoothing can still be applied to archived streams, there is an online analog called smoothing with lag. In this potentially less precise algorithm, smoothing is performed over a sliding window, with smoothed estimates becoming available after a certain lag period (e.g., the window length). Both forward and backward algorithms have linear complexity in the length of a stream. The most likely explanation can be found using the Viterbi algorithm which also has linear complexity.

4.1.2 Inference with RFID Data

We use the most popular temporal model for representing the changing location of a tagged person or object, a Hidden Markov Model (HMM) [260]. An HMM has one hidden state, a discrete random variable representing the entity's true location, and a state-dependent observable variable that represents a tag reading (or lack thereof) (see Figure 4.2). We set the time interval between consecutive time slices to be one second. This will generate continuous but not overly dense streams and should be sufficient granularity for tracking most activities that occur on a human-observable time scale (e.g., entering a room, walking down a hallway).

Filtering

We perform filtering on our model using a technique called *particle filtering* which has been used widely for location estimation in pervasive computing [111]. Particle filters are favored for their ability to represent arbitrary probability densities, including the asymmetric, multi-modal distributions that are typical of location estimates. They represent the distribution

over an entity's possible location with a set of samples, or *particles*, such that more likely places are associated with more samples. When the set of locations in an environment is large, this sample-based approach allows particle filters to perform more efficiently than other techniques (e.g., grid-based techniques [111, 110]) that consider every location. Our particle filter updates its stream of location estimates by reasoning about location transitions and tag read events using a set of models that are specific to our deployment environment (e.g., our computer science building), tagged entities, and the RFID Ecosystem.

Locations defined by antenna deployment points can be coarse and there may be a granularity mismatch between antenna topology and the location information needed by an application. We use a connectivity graph as a topological model to discretize the space in our computer science department building and to represent the locations that are interesting to applications (see Figure 4.3). In our graph, each room is a place (e.g., a node) and hallways are sliced into non-uniform segments based upon the adjoining doorways. Places can have varying size, ours are all at least several square feet in size. We defined our graph by hand, and in non-research settings such a graph could easily be defined by a system administrator. During a location update, particles may move along the edges of the graph. However, to extract a location estimate that references meaningful high-level places particles can be quickly snapped to a node. In addition to providing location information in terms of meaningful places (instead of antennas), our topological approach also increases efficiency by constraining the set of locations that need to be considered during each update. It should also be emphasized that tracking on a graph allows us to probabilistically follow a person or object *even when it is at a location that does not have an RFID antenna.* This is a crucial support for higher-level programming tasks and applications that need to reference locations which may not be covered by an RFID deployment. This is often the case when regulatory or budgetary constraints prevent antenna deployment.

Our particle filter also uses a *motion model* that describes how a particle should move when it is updated. The motion model captures a notion of how the tagged entity is likely to move. We assume that people and objects move straight down hallways at roughly 1.0 meter/second, and choose a new direction with uniform probability at intersections (doorways are considered intersections). This simple model captures several aspects of a

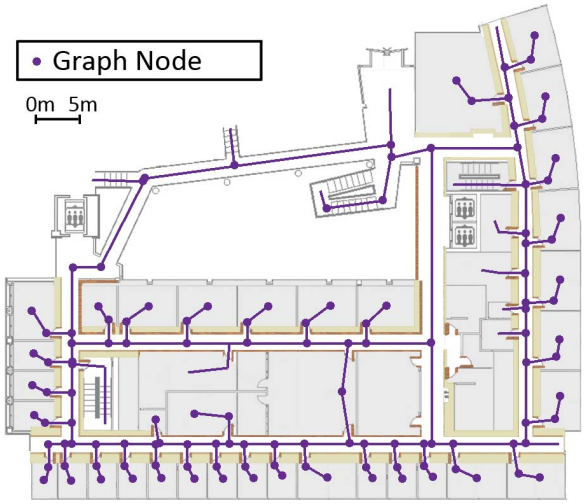


Figure 4.3: **A connectivity graph that discretizes the space in our deployment. The graph also matches locations to meaningful high-level places (e.g., rooms).**

real situation. First, that an entity in an indoor RFID system is not likely to move any faster than a human. Second, that entities cannot "teleport" through walls, they must move through doorways to enter an enclosed space. Finally, that we cannot reliably predict what door a person or object will enter without additional information. When the set of particles is moved by the motion model, they represent the updated distribution over an entity's current true place.

When a new TRE is available, the particle filter re-weights particles using a *sensor model*: particles having coordinates consistent with the TRE are given higher weights than those with inconsistent coordinates. Our default sensor model assigns a fixed, high weight (e.g., 0.8) to particles within the read range of the detecting antenna, and a low weight (e.g., 0.01) to particles beyond this range. After re-weighting, a new set of uniformly-weighted particles is produced from the weighted set using importance sampling with replacement, and the process repeats. This simple default model is parameterized with deployment specific details (e.g., antenna locations) and empirical measurements (e.g., average read range) from the RFID Ecosystem (see Chapter 2). However, suitable default parameters could be used or tweaked by an administrator for use in another deployment.

This update process itself is concretely depicted in Figure 4.4. In the first timestep, (a), antenna A detects the tag. Particles close to A get high weights and are more likely to be

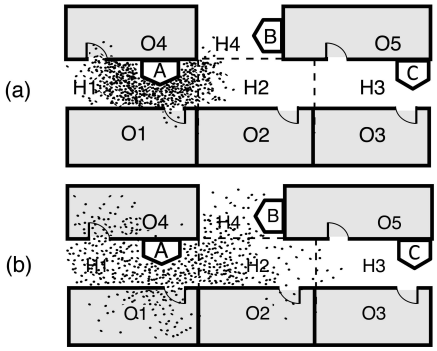


Figure 4.4: **A particle filter's sample-based representation of the distribution over a single tag's place at two timesteps. (a) Antenna A detects the tag, creating a focused distribution. (b) No antennas detect the tag, creating a distribution with more uncertainty.**

resampled, producing a distribution that is fairly concentrated around A: that is, the tag's place is fairly certain at that time (in the figure, hallway H1 has roughly 0.85 probability). This resampling procedure further increases efficiency by focusing resources on the most likely locations. In the next timestep, (b), no antennas detect the tag. Without any sensor input, the particle filter cannot determine whether the tag entered an office or whether it perhaps remained in the hallway and was simply missed by the readers. The particle distribution reflects this ambiguity; it is more diffuse and covers a larger set of places than the distribution in (a). These particles will continue to disperse with each timestep until another tag read re-focuses the distribution. It is important to note that when another tag read finally occurs, particles within range of the antenna will be more heavily weighted and hence probably resampled, causing many particles with non-viable location coordinates to disappear. Again however, the particle filter does not "teleport" to the coordinates of the tag read any particles that were not already there.

A probability distribution over an entity's possible location is materialized by counting particles in places. Each place P containing at least one particle produces a new possible place in the distribution. The place P will have a probability equal to the sum of the particle weights for particles inside that place (i.e., nearest to that node in the connectivity graph). An important consequence of this construction is that when a particle distribution is diffuse, as in Figure 4.4(b), the probabilities of *all* places at the current timestep, including

the correct one, are low. This is because the probability mass is spread across many places. Thus, *low absolute probabilities can still identify meaningful locations and events.* As an example consider a distribution in which place P_1 has probability 0.2 and all other places have probability 0.01. This distribution suggests that it is twenty times more likely that the tag is in P_1 than anywhere else. In contrast, a distribution where P_1 and P_2 both have probability 0.5 actually has *less* certainty about the tag's place, despite the high absolute probability values. Finally, we note that our particle filter generally produces distributions in which a small number (1-3) of places have significant probability (e.g., > 0.2), while all remaining places have diminutive probabilities (e.g., < 0.01).

Smoothing

A particle filter can be run in near real-time on an incoming stream of RFID tag read events to produce a continuous stream of location estimates that refer to locations of interest to applications and users. Once this stream has been archived, or after some lag has passed, a smoothing algorithm can be run to refine the estimates using additional TREs. For example, suppose in Figure 4.4 that antenna C detects a tag at a third timestep immediately after (b). Given this reading, it is highly unlikely that the tag was in place O1 at the timestep in (b). As such, smoothing will refine the past distribution at timestep (b) by descreasing the probability that the tag was in O1.

The smoothing algorithm also extracts a set of correlations between locations at consecutive timesteps, representing the probability that an entity moved from one place to another. These transition probabilities are derived from the connectivity graph, motion and sensor models. For example, the probability that an entity moves through a wall from one place to another will be 0. Similarly, the probability that an entity moves from inside O1 at (b), to O3 down the hallway in a single timestep will also be 0. Further, an entity moving down the hallway from H1 to H2 is less likely to switch direction and return to H1 in the next timestep. Thus, the smoothing algorithm produces a stream of refined location estimates with additional information provided by the correlations (i.e., transition probabilities) between each estimate.

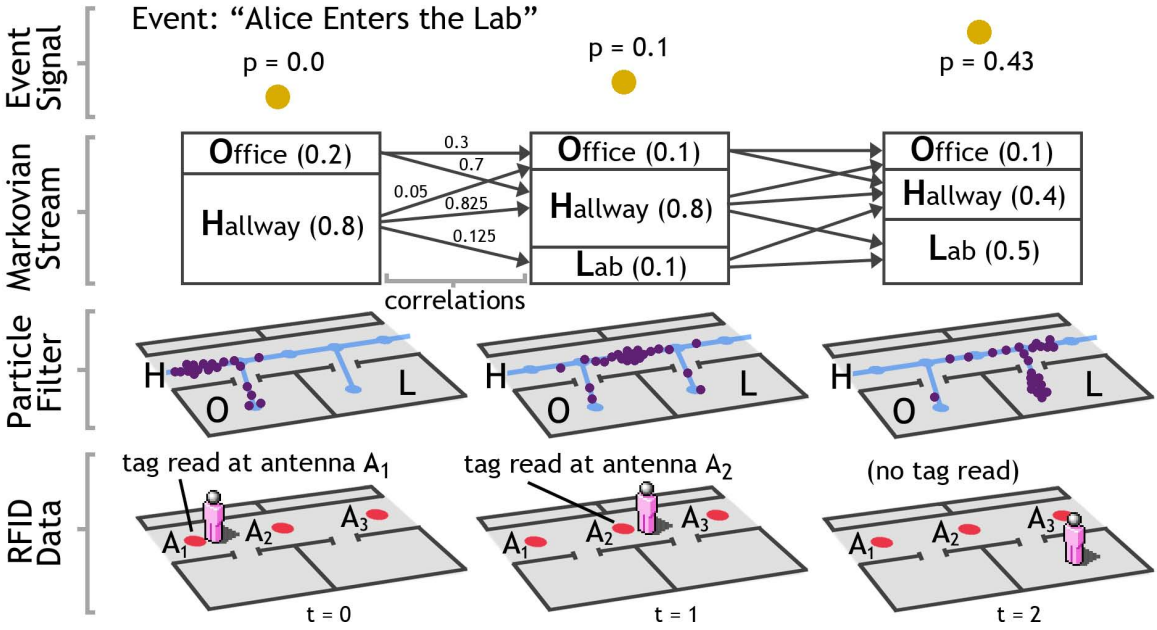


Figure 4.5: **The substrate transforms raw, uncertain location data into smoothed, probabilistic Markovian streams over which Lahar detects complex events.**

4.2 A Model-Based View to Facilitate Queries

The particle filter and smoothing algorithm produce continuous, probabilistic, and correlated sequences of meaningful location estimates from sporadic, low-level RFID data. To make these sequences accessible to higher-level processes and applications, we materialize them in a *model-based database view*. Model-based views are a recently proposed technique for creating a view from sparse or imprecise sensor data in a way that is consistent with some probabilistic model [86, 84]. The model often embodies a higher level concept that is easily understood by programmers and end-users. In our case, the model is our HMM, and embodies the intuitive concept of an entity's true location over the space of possible rooms and hallways in a building. This type of view is called a *Markovian Stream* (MStream) (see Figure 4.5), and was introduced by Letchner et al. [211]. Using an MStream considerably simplifies the task of specifying a location event over RFID location streams by eliminating the need to consider antenna topology and missed readings.

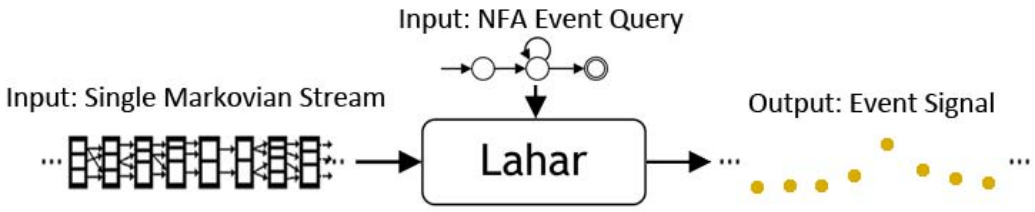


Figure 4.6: **The Lahar engine takes an NFA event query and a single MStream as input and outputs a probabilistic event signal.**

4.3 A Probabilistic Complex Event Detection Engine

As explained at the beginning of the chapter, we use a probabilistic complex event detection engine called Lahar [211, 273] to extract location events from MStreams. Lahar's query language is based on regular expressions that are represented internally with finite state machines (FSMs). A key advantage to using Lahar is that, together with standard query predicates, its language reduces the problem of specifying an event to that of specifying a pattern. Indeed, Lahar essentially works by performing probabilistic pattern matching on an MStream. This greatly facilitates complex event specification and detection.

Given an event pattern and an MStream, Lahar outputs a single probabilistic query signal, or *event signal*. The event signal consists of timestep-probability pairs <`t,p`> which indicate that the event occurred at timestep `t` with probability `p` (see Figure 4.6). Each probability `p` is derived from an MStream using well-established query answering techniques for probabilistic databases [273]. Thus, a second key advantage to using Lahar is that it extracts events over uncertain data without requiring users to train complex models or create custom-coded event-detection modules.

Lahar is an evolving research prototype, and as such it is currently quite limited in the set of events is can support. At present, it supports instantaneous and sequence events on a single MStream (see Figure 4.7). Moreover, it only supports evaluation of the most basic predicates on each edge of a FSM event query. In terms of the events from the taxonomy established in Chapter 3, Lahar can detect primitive events that refer only to a single tracked person or object and require only a simple comparison of location values (e.g., "Alice is in

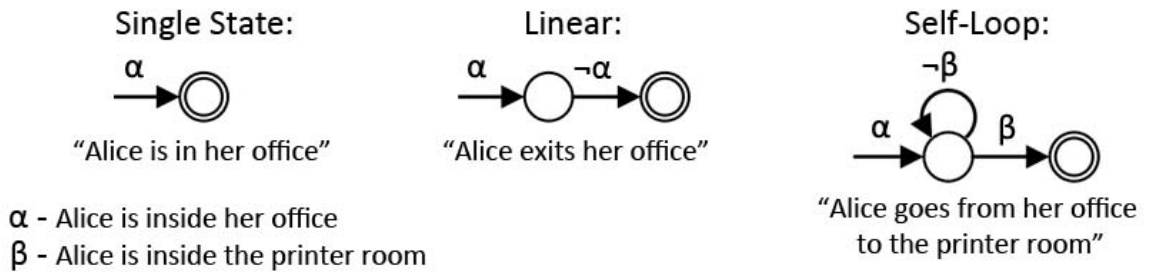


Figure 4.7: **Representative examples of the three basic classes of event supported by Lahar: single state, linear, and self-loop. Event queries supported directly by Lahar are also constrained to only the simplest predicates (e.g., "node ID == 1254").**

her office"). It can also detect sequences of such primitive events in which each primitive event occurs either immediately after or some time after the previous event (e.g., "Alice exits her office", or "Alice walks from her office to the printer room"). However, Lahar *cannot* detect a large set of events that are common to many sophisticated location-aware applications. For example, it does not support events that refer to multiple MStreams (e.g., "Alice is meeting with Bob in the lounge"). Neither does it support FSM event queries that require slightly more complex predicates on location values (e.g., "Alice is within 5 meters of her office"). In the next Section we present extensions that enable detection of this larger and more critical set of events.

4.4 Extensions to Support a Larger Family of Complex Events

We extend Lahar to support a larger portion of the location events (and application domains) discussed in Chapter 3 by introducing a new component called the Event Manager. The Event Manager extends Lahar by caching intermediate results, evaluating more complex predicates on location (e.g., "Alice is less than 5 meters from her office") and by translating queries over multiple MStreams into sets of single MStream queries that Lahar can answer. In this section we describe the services the Event Manager provides and present a taxonomy for the types of events it supports.

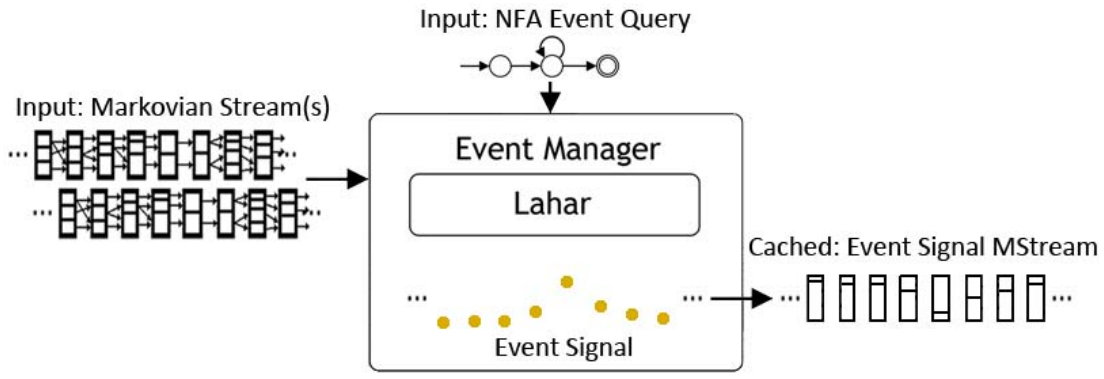


Figure 4.8: **The Event Manager caches output event signals as MStreams for reuse in answering later queries.**

4.4.1 Caching Event Signals

An event may be detected many times over the same data set, this is especially true when the event is a *sub-event*, or component, of a larger event or sequence. To maximize efficiency when repeatedly detecting a popular event, the Event Manager transforms the output event signal into an *independent* MStream (i.e., an MStream without correlations) and stores it for future reuse (see Figure 4.8) along with metadata on the query, MStream, and time range. Because many event queries share the same sub-queries (e.g., "Alice is in her office" and "Alice exits her office"), reuse of cached results can greatly optimize the query answering process. This substantially reduces event detection latency and supports efficient detection of events over multiple MStreams as discussed below.

4.4.2 Evaluating Complex Predicates

Currently, Lahar can only evaluate inside and outside events over a single MStream (i.e., for a single person or object). Chapter 3 showed that many primitive events require evaluation of more complex predicates on location, such as whether a person is far from a place, an object is near a person, or a *group* of people are all inside the same place. The Event Manager completes Cascadia's support for *all* primitive event predicates over one *or more* MStreams. That is, it supports inside and outside for multiple MStreams, and near, far, with, and without for both single and multiple MStreams. Note that here the Event Manager operates directly on the MStream(s) to produce an event signal and does not use Lahar.

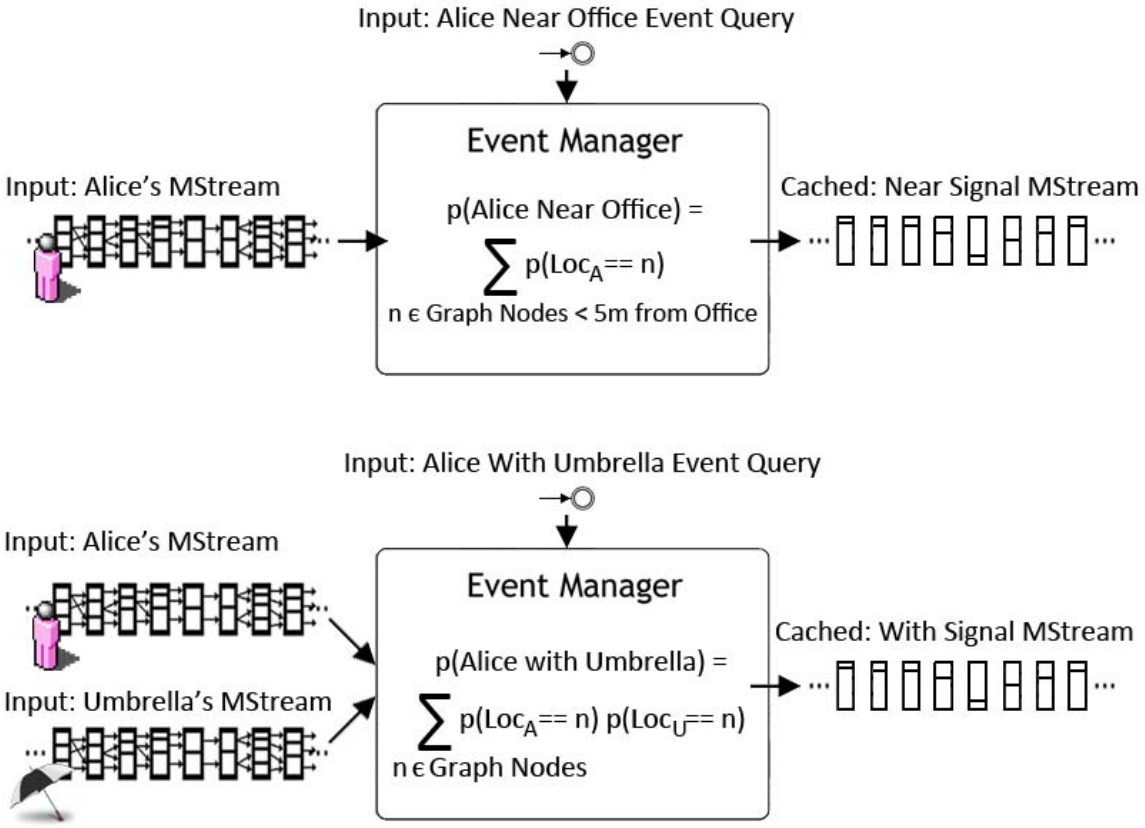


Figure 4.9: **The Event Manager is able to evaluate more complex predicates, such as "Alice is near her Office", or "Alice is with her umbrella", that may require non-trivial computations or comparisons involving more than one MStream.**

To evaluate near and far predicates, the Event Manager first precomputes the shortest path between each pair of nodes in the connectivity graph and caches the result as a hashtable in memory. Near and far predicates are then evaluated by computing the probability that the entities in question are at graph nodes with a shortest path between them that is shorter or longer than some distance. As shown in Figure 4.9, with and without predicates are evaluated by computing the probability that two people or objects are either at the same or a different location at a given time step. Inside and outside predicates are evaluated over multiple MStreams in a similar fashion. The Event Manager evaluates predicates at every timestep of the input MStream(s) to output an event signal; this event signal is cached as described above.

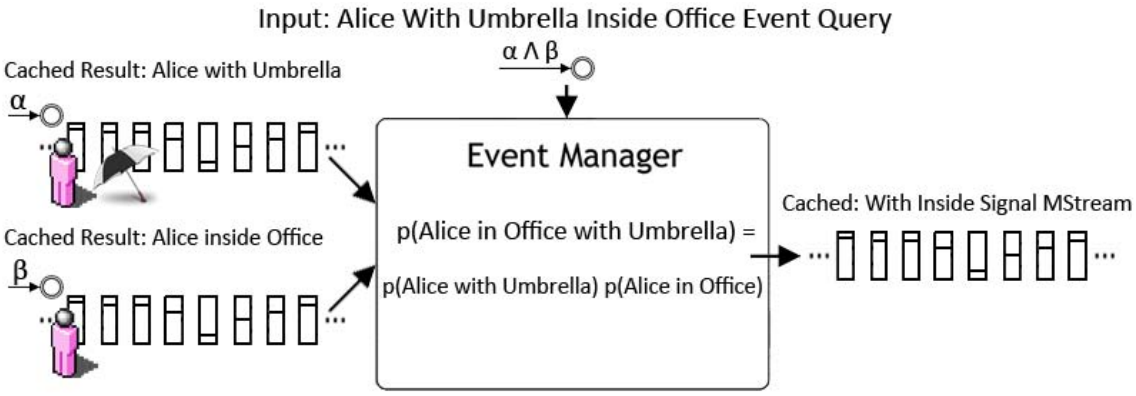


Figure 4.10: **The Event Manager answers single-state FSM event queries over multiple streams by computing the event signal for each conjunct, and then merging all such event signals by assuming independence and taking their product at each timestep.**

4.4.3 Detecting Single State Conjunctions Over Multiple MStreams

Lahar's most limiting constraint is that it can only answer queries over a single MStream at a time. Thus, events involving multiple people or objects cannot be detected directly by Lahar. For multi-stream events having FSMs with a single state, the Event Manager overcomes this limitation by breaking the query into a set of queries that either it or Lahar can answer, one for each conjunct in the multi-stream query. After the breakdown, each *sub-event* is detected either directly by Lahar (i.e., single MStream inside and outside events) or by the Event Manager (i.e., all other events). The resulting event signals are merged into a single event signal for the multi-stream query by assuming independence and taking their product at each timestep (see Figure 4.10).

4.4.4 Detecting Multiple State Events Over Multiple MStreams

Like the single-state FSM case, the Event Manager detects event queries having multiple FSM states by breaking them down into sub-events and then merging sub-event result signals. However, in the multi-state case, once the query for each state has been answered as above, the resulting event signals are not merged by taking their product. Instead, the merged stream is formed as the joint distribution over the event signals. The distribution over the hidden variable for this merged stream does not have only 2 possibilities (e.g., true

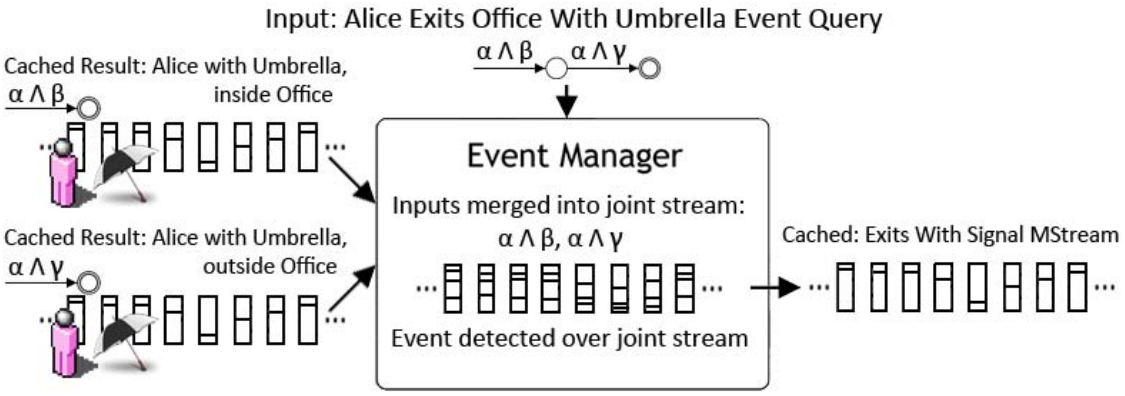


Figure 4.11: **The Event Manager answers multi-state NFA event queries over multiple streams by computing the event signal for each state, merging all signals for states into a single stream by forming their joint distribution, and then computing the event signal for the multi-state query over that joint MStream, assuming independence.**

and false), but 2^n where n is the number of event signals merged. Finally, the multiple state FSM query is then run directly by Lahar over this merged stream to produce the event signal as shown in Figure 4.11.

4.4.5 A Taxonomy of Supported Events

The Event Manager combines the techniques described above to support detection of a large and important family of events. Here we precisely articulate *a taxonomy for supported events*. For each type of event we provide an example and present the frequency with which that event occurs in all location-aware applications, research applictions, and commercial applications according to the survey in Chapter 3.

The supported events are distinguished along several dimensions. The first dimension is *number of MStreams* referenced which corresponds to the number of people or objects involved in the event. The second dimension is the *type and number of primitive events* used as predicates on the edges of the FSM query. The third dimension is the *structure of the FSM query* itself, which corresponds to the way primitive events are composed as a sequence. The taxonomy can be enumerated inductively as shown below.

Base Case. The simplest events refer to a single MStream, use a single primitive event, and have a FSM with a single state. An example of such an event is "Alice is in her office". This very simple type of event accounts for 62% of events in all applications, 68% of events in research applications, and 60% of events in commercial applications.

Conjunctions. Events may also use a conjunction of primitive events as a predicate for an edge of the FSM. An example of such an event is "Alice is in her workplace and near her office". This type of event, when extended to cover multiple MStreams as described below, covers 5% of events in all applications, 4% of events in research applications, and 5% of all events in commercial applications.

Multiple FSM States. Many events include a sequence of sub-events that are represented as a linear sequence of states in the FSM (i.e., state machines without loops). An event with a linear FSM is "Alice exits her office and walks down the hall to the printer room".

FSM States with Self-Loop Edges. Not all sequence-oriented events consist of strictly consecutive sub-events. Some events may have sub-events that are separated by a longer period of time. For example, "Alice is in her office and then (sometime later) exits the building". The advantage to such events is their generality (e.g., one need not specify the exact path Alice took to leave the building). Sequence events that potentially include self-loops account for 20% of events in all applications, 19% of events in research, and 20% of events in commercial applications.

Multiple MStreams. All combinations of the above event types can be extended by incorporating additional MStreams. By incorporating multiple MStreams, events that refer to multiple people and objects can be supported. For example, "Alice leaves her office with her mug to meet Bob in the coffee room". By adding support for multiple MStreams, the Event Manager supports an additional 15% of events in all applications, 9% from research, and 17% from commercial applications.

This taxonomy of supported events cover a large portion of the events in Chapter 3. As such, we argue that this event detection substrate is sufficient to support most location-aware applications, including those that use more sophisticated complex location events.

4.5 Evaluation with Real RFID Data

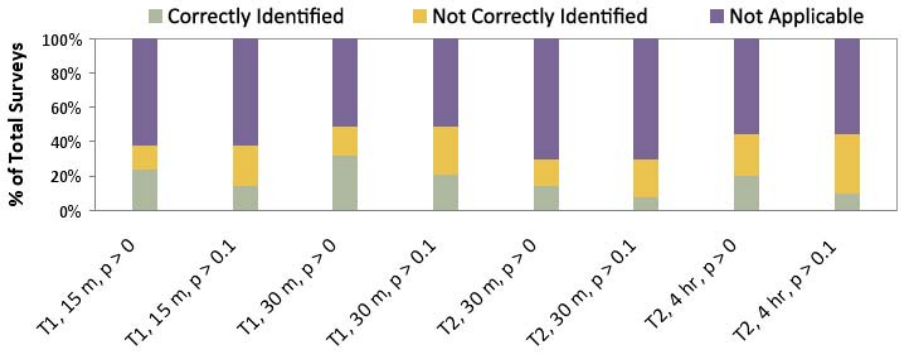
We evaluated our event detection substrate using real RFID data from the RFID Ecosystem. We discuss evaluation of each part of the substrate below. All experiments were conducted on a Dual Xeon 3GHz server with 8GB of RAM and 700GB of disk.

4.5.1 Evaluating Location Inference Techniques

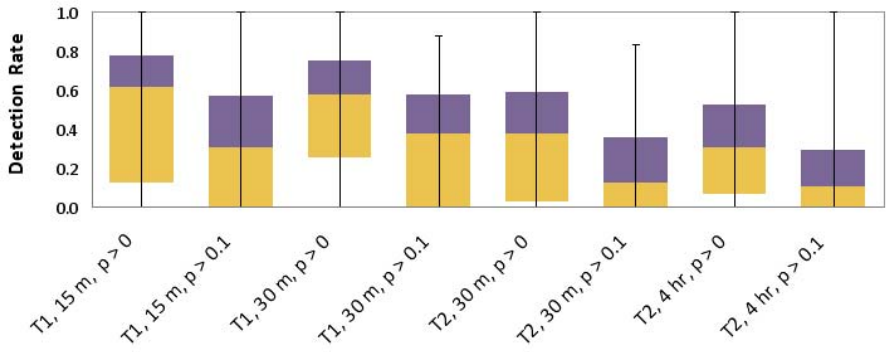
The location inference techniques can be characterized both by their execution time and resulting precision. A first evaluation examined performance over a single one-hour trace with less than 10,000 TREs collected by researchers who carried several Excalibur RFID tags. While the particle filter's execution time (per timestep) depends on the number of tags being tracked, we found that the particle filter produced a stable stream of estimates with near real-time performance for at least 175 tags if 500 particles were used for each tag. In total, over 2.3 million particles were produced for the trace, resulting in a storage blowup of more than 2 orders of magnitude.

In a second experiment, we materialized MStreams for all 324 tags over all four weeks of the longitudinal study from Chapter 2. Initially, we tried to run the particle filter and smoothing algorithms non-stop over the 4 weeks. This approach, however, lead to a significant amount of low-quality data due to long periods of tag immobility: in the absence of RFID data, the view is highly uncertain of a tag's location, producing unusable results. In a second approach, which we report on here, we only ran the algorithms over periods of time in which tag read events occurred. We call these periods *slots*.

We evaluated the quality of the probabilistic view by comparing the study's survey results (see Section 2.4.1 1358 points of ground truth on where each person and object is at randomly chosen times) to the materialized view for the appropriate tag. We use two comparison techniques T1 and T2. In T1, for each survey, we look at the tag's probabilistic



(a)



(b)

Figure 4.12: **Detection rate results for probabilistic view methods in terms of (a) Surveys correctly identified, and (b) quartiles. In both cases, the x-axis shows technique, time window, and probabilisty threshold**

view in a time window centered on (i.e., preceding and following) the survey response time. If the probability that the tag is in the correct location (according to the survey) is greater than a threshold, we consider the view to be correct. In T2, we only consider the last time slot before or containing the survey response time. T2 assumes that a tag does not move in between time slots.

For each technique, Figure 4.12 shows (1) the fraction of surveys where the view correctly identified a tag's location, (2) the fraction of surveys when the view was wrong, and (3) the fraction of surveys for which no RFID data for the correct floor was registered within the given window (these surveys are indicated as "not applicable"). For clarity, Table 4.1 shows

the corresponding detection rate values. The plots for probability threshold 0.1 correspond to instances where the view is confident that a tag is located in the given room (probabilites are low in general due to location ambiguity in cases where a tag could be in one of three adjacent rooms or in the corridor; spreading the probability values across several locations). The plots for threshold of 0 show instances where the view only localizes the tag to the correct vicinity (e.g., corridor or floor).

For T1, on average, the probabilistic view places the tag in the correct vicinity most of the time, but in the correct room less than half the time. Looking only at the most recent slot before a survey (T2) instead of the time around a survey (T1) yields worse results simply because fewer slots are considered. For example, if the system does not notice when a tag enters a room but correctly identifies that the tag left, T1 will report the correct location but T2 will not. Overall, *continuous*, fine-grained tracking in an RFID deployment is thus difficult. There are times when the system fails to capture the correct tag movements even after smoothing the sensor data.

The overall low detection rates are partly due to the fact that performance varies considerably among tags depending on the properties of the tag and tagged object (recall Figure 2.20 from Chapter 2). Figure 4.12(b) shows the quartiles for detection rate of different types of tags. For the best 25% of tags, the probabilistic view places the tag in the correct vicinity upwards of 80% of the time and in the correct room upwards of 60% of the time. These results are thus encouraging: a probabilistic view can smooth away significant errors and ambiguity. This approach, however, clearly fails with large error rates, suggesting that a practical deployment must take further measures to ensure tags have sufficiently high read rates.

4.5.2 Evaluating Probabilistic Complex Event Detection

In a second experiment, we studied the utility of the probabilistic view for inferring simple, higher-level events with Lahar and the Event Manager. Specifically, we defined a simple meeting event: two people meet if they appear in the same physical location within at most five minutes of each other. We then used the card-key logs for 4 researchers to

Technique	Win	Threshold	Detection rate
T1	15 m	p > 0.0	0.62
T1	15 m	p > 0.1	0.37
T1	30 m	p > 0.0	0.65
T1	30 m	p > 0.1	0.43
T2	30 m	p > 0.0	0.48
T2	30 m	p > 0.1	0.24
T2	240 m	p > 0.0	0.45
T2	240 m	p > 0.1	0.21

Table 4.1: **Average detection rate for each comparison technique, time window, and probability threshold.**

Prob. threshold	0	0.05	0.1	0.2	0.3	0.6
Data ratio	13.54	2.51	1.69	1.05	0.74	0.25

Table 4.2: **Probabilistic view data blow-up. The ratio is computed as the number of records in the materialized probabilistic view over the number of TREs when counting only data above a given probability threshold.**

pinpoint 10 meetings that occurred within a one week period. Of these 10, 6 were correctly identified using the probabilistic views, thereby echoing the results of the room-level location experiment.

Using probabilistic views over RFID data in large-scale, continuous tracking scenarios is thus necessary and promising but it raises important challenges in practice. As we showed above, probabilistic views help only when the data is already of sufficiently high quality. For alerting applications, the process must also be sufficiently fast to keep up with the input data (although techniques to achieve this goal exist [198, 241]). Finally, the per-timestep location distributions in probabilistic views can cause a data blow-up. Table 4.2 shows the data blow-up for the month long view over all participants. The entire view uses more than 13 times the space of the raw data. Eliminating data with low probability (less than 0.05), however, quickly reduces the overhead to just a small factor. A study of the impact of such compression techniques on performance is an interesting area of future work.

4.6 Summary

Overall, our event detection substrate presents a novel combination of probabilistic modeling, inference, and data management that enables detection of important events over uncertain location data. The filtering and smoothing used to materialize the MStream-based view of location data relieves programmers and end-users of the need to account for missed tag reads and incomplete sensor deployments. In combination with the MStream view, the Lahar probabilistic complex event detection engine reduces the problem of event detection to that of probabilistic pattern matching. This further simplifies the task of specifying events because the user has only to create an intuitive FSM, no complex models need to be trained or understood. Finally, the Event Manager extends the benefits of Lahar to cover a much larger set of important events from Chapter 3. Using real RFID data from the RFID Ecosystem, we have shown that even in situations with *very high uncertainty*, our event detection substrate produces useful results. Smaller, more targeted studies of RFID-based event detection with Lahar corroborate these results [212, 213].

Chapter 5

EVENT SPECIFICATION TOOLS FOR END-USERS

The probabilistic inference techniques and MStream database view presented in the previous Chapter eliminate the need to work directly with sporadic and incomplete sensor streams. This allows us to reduce costs by using sparser deployments of cheaper, less reliable location sensors (e.g., passive RFID), and hence address the first key challenge from the introduction: cost of location sensors. The substrate's versatile event detection support using the Lahar engine further facilitates work with sensor data by reducing the problem of event detection to probabilistic pattern matching over MStreams. This addresses the second key challenge: flexible support for events and uncertainty. However, the FSM-based query language used by our substrate is unlikely to be accessible to end-users in many scenarios (e.g., nurses in a hospital). Thus, additional work is required to address the third challenge, *the cost of deploying and tuning applications.*

In this chapter, we present the design and evaluation of a suite of web-based user-level tools that allow non-expert end-users to specify events for our event detection substrate. Two tools, the Tag Manager (from Chapter 2) and the Place Manager, simply provide a mechanism for specifying metadata about objects and places. A third tool, Scenic, allows end-users to generate FSM event specifications using an intuitive storyboard metaphor. Scenic was designed and refined using the results of the survey of events in Chapter 3, and through several formative studies which we describe in the following Sections. We conclude by motivating the work in Chapter 6 with an additional study that identifies common difficulties end-users face when using Scenic to specify events.

5.1 Example Usage Scenario

Recall from Section 1.1 that the problem of deploying and tuning applications is particularly acute in the hospital domain. Here, location systems are used by nursing staff and

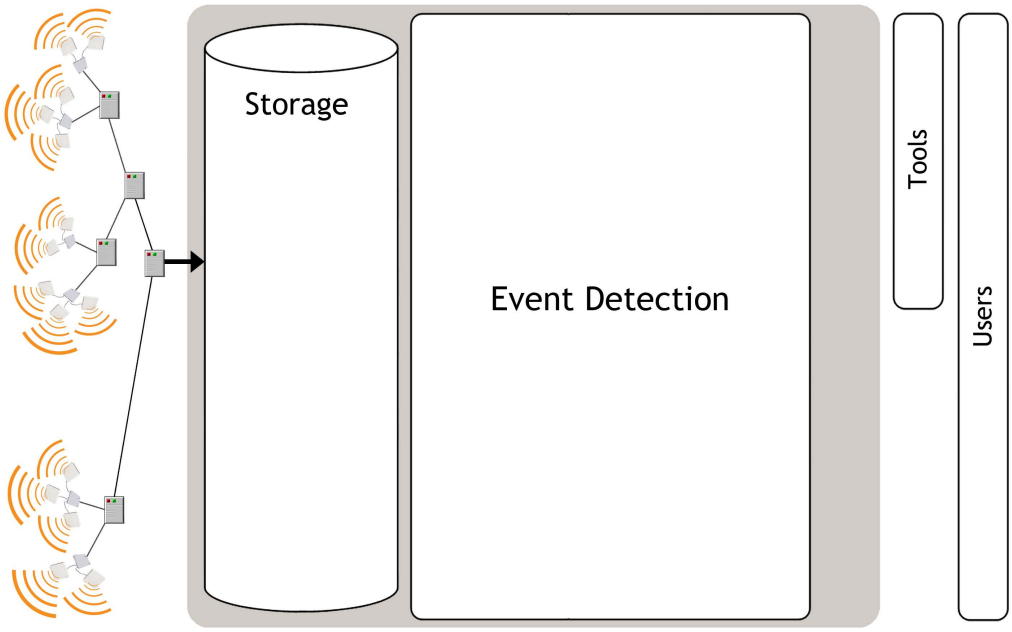


Figure 5.1: **In this chapter and the next we focus on user-level tools.**

administration officials to track equipment and healthcare workflows that include complex interactions among staff, patients, and equipment. In a longitudinal study of such a location system in a 300-bed hospital, Christe et al. [66] observed that the hospital needed to employ several full-time, on-site clinical engineers to install, tune, and maintain the location system. In addition to this expense, they also found the software was difficult for the end-users to work with - clinical engineers frequently assisted nursing staff in obtaining needed information from the location system software. While some vendors aim to reduce these problems by requiring hospitals to change workflow processes to match those supported by their software, experts find this solution even less desirable. Indeed, Christe argues that hospitals often follow well-established processes that are *"notoriously difficult to change"*, and a larger survey of RFID systems in hospitals concluded that *"RFID applications should be supportive of healthcare processes (not the other way around)."* [326].

While the suite of end-user tools that we present in this chapter may not entirely eliminate the need for support from clinical engineers, it significantly lowers the barrier for end-users to specify and detect events. Thus, the effort required to specify events could be shared across a larger number of non-expert staff members - not just clinical engineers.

Moreover, even expert clinical engineers could benefit from the proposed tools because the time and expense of specifying new events would be significantly reduced.

In an example scenario, a hospital's CIO may wish to analyze and improve on the cardiology department's patient flow. With our Tag and Place Manager tools, the CIO could directly specify the relevant people and equipment from her web browser. Then, using the Scenic tool, she could specify one or more workflows that the department currently implements. If the CIO lacks expertise in the department's current workflows, she could ask a member of the nursing staff with direct knowledge of workflows to specify the event. Once the event is specified, the event detection substrate will detect all occurrences of the event in addition to any occurrences of sub-events. The CIO will later be able to review and analyze the accumulated event data to extract valuable performance data like patient flow, patient throughput, and equipment availability. Using these metrics, the CIO can make carefully informed decisions on whether and how to change current processes and on whether to purchase new equipment. By using the suite of event specification tools, a hospital staff could eliminate the cost of a full-time team of clinical engineers and enable end-users to *directly* specify and extract the information they are interested in.

5.2 Tools for Specifying Metadata

The data produced by most location systems is low-level and specified with coordinates or in terms of specific physical sensors. For example, the RFID Ecosystem produces TREs that refer to tags and antennas. Such data are only useful to end users if they can be transformed into more meaningful high-level information about people, objects, and places (e.g., *Alice's purse is in her office*). As such, we developed tools that allow end-users to attach meaningful high-level metadata to locations (e.g., Alice's office) and location sensors (e.g., the tag in Alice's purse). Both tools are web-based and we discuss them in more detail below.

The first tool, the *Tag Manager* (see Figure 5.2), was briefly introduced in Chapter 2. The Tag Manager presents a set of menus, tables and web forms for creation and management of metadata on a user's tags and personal objects. It interfaces with the RFID kiosk (see Chapter 2) so that when users are at the kiosk they can associate one or more physical

Figure 5.2: **The Tag Manager tool creates and manages metadata on objects to which tags can be bound. The list at the left displays a user's objects while the display to the right provides metadata on the currently selected object.**

tags with an object. For example, a new user can use the Tag Manager at the kiosk to register several personal tags. Later, the same user can access the Tag Manager from his laptop to delete objects and to review or edit object metadata (e.g., name, type, image URL, where the object's tags were last seen).

A second tool, the *Place Manager*, supports the creation and editing of high-level location information items called *places* [150]. We define a *place* as a geometric region with a label. For example, the area corresponding to a user's office might be grouped and labeled "my office". The *Place Manager* displays the locations in a place as a set of circular icons on a Google Map mash-up of the location sensor deployment. The icons may correspond to RFID antennas or to nodes in the connectivity graph described in Chapter 4. When a person or object is at an antenna or node that is part of a place, then that person or object is considered to be in that place. Users can create or edit a place by clicking on the map to add or remove additional circular icons and by entering the place label in a text box (see Figure 5.3).

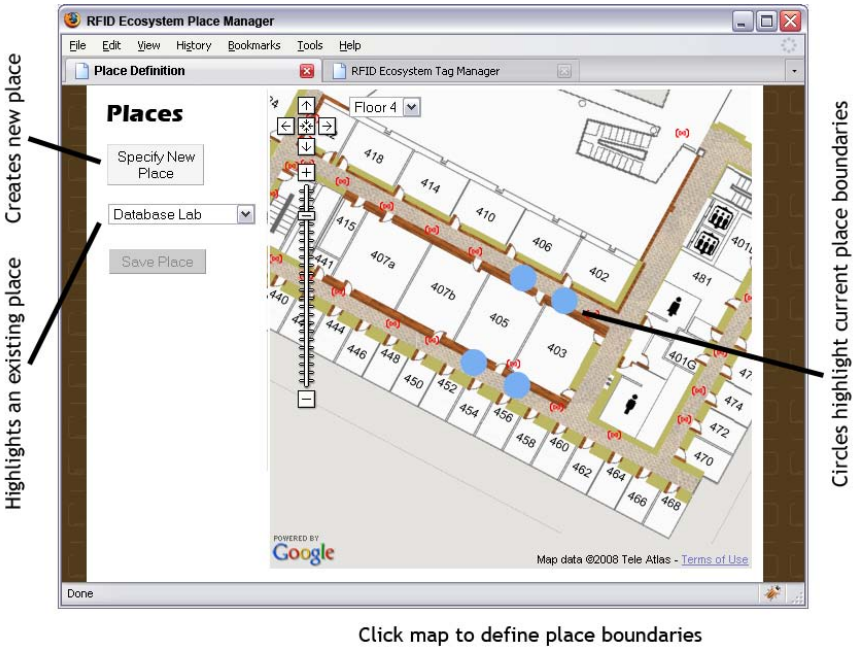


Figure 5.3: **The Place Manager tool creates and manages metadata on places covered by the location system. Users can create a new place by clicking the "Specify New Place" button and then clicking on the map to indicate place boundaries. Users can review and edit an existing place by selecting it from the drop-down list to the left of the map.**

Once defined with the *Tag Manager* and *Place Manager*, metadata that binds tags to objects and antennas to places can be used by applications and other system components to generate higher-level information that is personalized and more directly meaningful to users. The third tool, *Scenic*, allows a user to specify *what* higher-level events are to be extracted from his or her TREs (see Figure 5.4). We discuss Scenic further in the next Section.

5.3 Initial Scenic Prototype

Authoring event specifications using our substrate's FSM-based query language may be difficult for developers and impossible for end-users. As such, we designed a visual language for event specification. The language is embodied in the Scenic tool, a web-based tool that allows users to create or edit event specifications at run-time. Scenic can be used as a

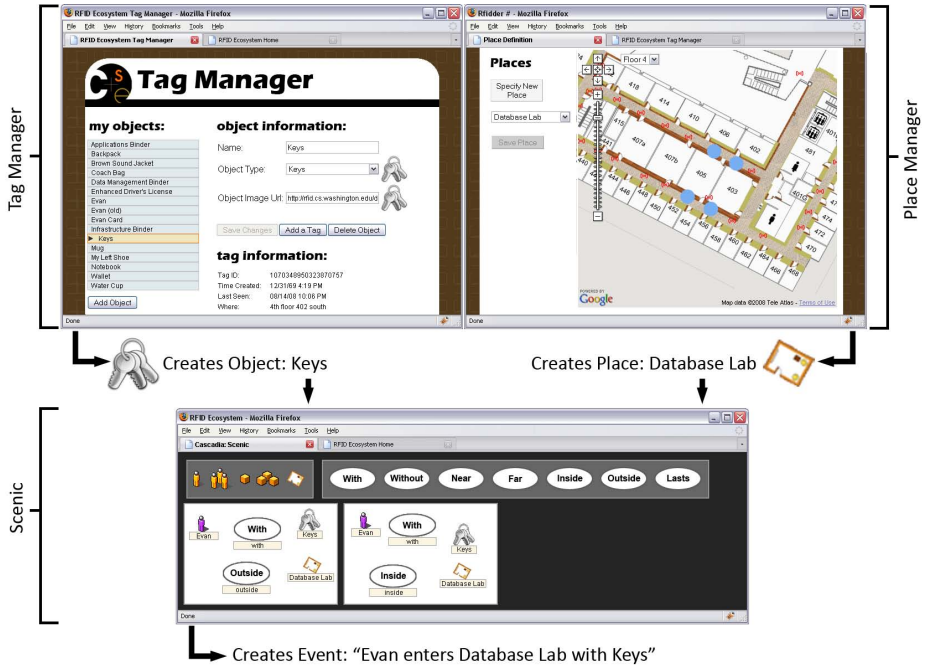


Figure 5.4: **Metadata management tools: Tag Manager creates and manages tagged objects; Place Manager groups antennas or nodes into places; Scenic uses metadata on objects, people and places to specify how a user's TREs are transformed into higher-level events.**

standalone web tool or embedded as a module within another application. For example, a digital diary application (see Chapter 7) could allow users to launch Scenic as a toolbar for specifying and customizing events that will be automatically detected and inserted into their calendars. Developers can also use Scenic to specify "template events" that can be included with applications and later customized by end users.

The inital Scenic prototype was guided by the results of the survey in Chapter 3. Entities (e.g., people, places, and things), primitive events (e.g., with, without, near, far, inside, outside), and composition techniques (e.g., sequencing, conjunction) are all represented directly in the interface. To facilitate interaction, Scenic uses an iconic visual language that represents primitive events and entities as icons that can be dragged and dropped onto a storyboard to specify instantaneous events called *Scenes*. A *Sequence* of Scenes specifies a complex event as a sequence of spatio-temporal *sub-events*. Thus, to specify an event users just "tell the story" of the event, Scene by Scene. Figure 5.5 shows the Scenic interface,

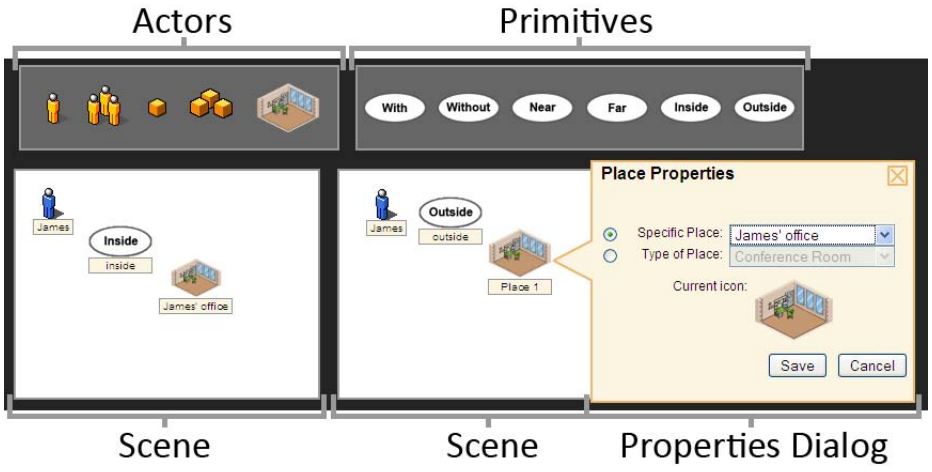


Figure 5.5: **A screenshot showing the initial Scenic prototype with Scenes, Actors, Primitives and a properties dialog for an Actor.**

which consists of a toolbar, below which is a working area called the sequence panel. We discuss each interface component in greater detail below.

Scenes. Scenes represent instantaneous events which may contain a conjunction of primitive events. They are displayed as white panels over the grey sequence panel. Users can compose Scenes as sequences by arranging them horizontally on the sequence panel; time is assumed to flow from left to right, with each scene strictly following the previous one (i.e., not overlapping). Scenes can be inserted and deleted with a few clicks, or reordered by dragging.

Actors. Actors represent people, places and things in an instantaneous event and map to metadata defined by the Tag and Place Manager tools. Scenic also supports groups of people or things that have a particular type (e.g., a group of four "students", a group of "books"). Each type of Actor is displayed in the toolbar as a separate icon; by clicking on an icon, users can create and drag a new Actor into a Scene. Each successively created Actor is represented by a different color icon and a distinct, anonymous unique identifier (i.e., "Person #4"). After dropping an Actor into a Scene, a user can right-click to pop-up a properties menu that sets the Actor's identity as a specific person or thing (e.g., "Ana"), or as a type of person or thing (e.g., "student"). Bounds for the size of a group must also

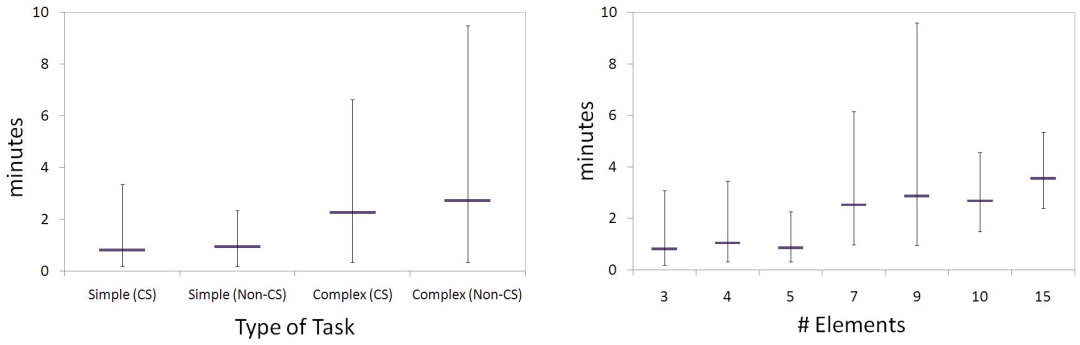
be set this way. The list of available identities for an Actor is retrieved from a database of metadata tables when Scenic launches.

Primitives. Scenic also provides icons for the event primitives. Conjunctions (`AND`) are indicated by dragging multiple primitives onto the same Scene. In this initial design, each Primitive in a given Scene applies to all Actors in that Scene. Primitives are created and used in a similar way to Actors. By dragging a Primitive onto a Scene, a user can specify a relationship between Actors in that Scene. The *near* and *far* Primitives also have properties that can be set to indicate how near or far Actors must be from each other.

5.3.1 Preliminary Evaluation

We evaluated our initial Scenic design with a laboratory study that included 11 participants: 6 with computer science (CS) backgrounds and 5 with non-technical or non-programming (non-CS) backgrounds. Participants were offered $20 to participate in a 90 minute study session. After a 10 minute tutorial and practice period, participants were given a series of 22 timed event specification tasks. Each task included a general, high-level English description of an event from Chapter 3 that participants were asked to specify as precisely as possible. Of the tasks, 12 were *simple tasks*, which presented descriptions of simple events, likely to be specified in one Scene with one Primitive. Another 10 were *complex tasks* which presented descriptions of complex events, likely to be specified with multiple Scenes and Primitives. The study session concluded with a questionnaire that asked participants to rate various aspects of Scenic on expressiveness, precision, and ease-of-use. Finally, we hired 2 coders (both CS) to rate how well each event specification from each participant captured the original English event description. The coders were trained as experts with Scenic and the basic workings of RFID systems during the first hour of a 2 hour session in which they rated every event specification.

Overall, CS participants spent 93 seconds per event specification task on average, with a standard deviation of 21 seconds, while non-CS participants spent an average of 111 seconds with a standard deviation of 42 seconds. Figure 5.6 describes the timing results in detail. The first plot shows that on average, participants could complete simple tasks within 1



(a) (b)

Figure 5.6: **Min, avg, max specification times. (a) For CS and Non-CS. (b) For increasing number of elements (i.e., the number of Actors + the number of relations between Actors).**

minute and complex tasks within 3 minutes. A few participants took significantly longer on complex tasks. In the second plot we clustered tasks by the number of elements in the event (i.e., number of Actors + number of relations between Actors). This metric directly correlates with the number of operations (e.g., click and drag, dialog interaction) that must be performed to complete the specification. As the number of elements increases, the average time taken per event remains flat for small numbers of elements (< 5) and afterwards increases approximately linearly with the number of elements in the event. The maximum time, however, increased much more rapidly, showing again that some participants needed to think longer about more complex events. The slight drop in time taken for the events with 10 and 15 elements is due to the fact that these events were logically quite simple but involved many entities - so they required less thought and the time was dominated by manual operations.

Scenic received an overall average rating of 3.9 for expressiveness, 3.4 for precision, and 4.1 for ease of use on a 5 point scale where 5 is the most and 1 is the least (detailed ratings are shown in Figure 5.1). The lowest ratings received were for precision; participants explained that they were unsure whether the system would correctly interpret their intended meaning for some complex events. For example, in specifying the event "Bill prints his meeting

Dimension	EXP	PR	EASE
Choice of words for primitives	3.8	3.6	3.7
Properties of primitives	4.3	4.2	3.9
Overall use of primitives	3.6	3.4	3.8
Choice of icons for actors	4.5	4.3	4.4
Choice of types for actors	4.3	4.4	4.5
Overall use of actors	4.4	4.2	4.4
Overall use of Scenic	3.9	3.4	4.1

Table 5.1: **Table showing the average questionnaire ratings for expressiveness (EXP), precision (PR), and ease of use (EASE) on a 5 point scale where 5 is the best.**

notes, picks them up from the printer room, and returns to his office", one participant did not specify that Bill picked up the meeting notes in the printer room, only that he had them after he returned. The user later expressed concern about whether the system would be able to infer what was intended. We further investigate this type of problem in Section 5.6. The highest ratings were for Actors, showing that participants appreciated the direct representation and manipulation of people, places, and things. Some participants (both CS and non-CS) really enjoyed using the prototype and even spent extra time to neatly arrange their icons before submitting an event.

On average, the coders gave a rating of 6 to specifications for simple tasks and 5.5 to specifications for complex tasks, with standard deviation under 1.5 for both on a 7 point scale (where 7 is the most similar and 1 is the least similar). The mode rating for specifications from each simple task was 7. Events specifications for three of the complex tasks had a mode rating of 6, while specifications for the other complex tasks had mode rating of 7. Coders explained that ratings less than 7 were assigned most often due to missing details in the specification (e.g., a person or object is left out of a Scene when it needed to be included). These results show that not only could users quickly create event specifications, but they could create specifications which were arguably correct.

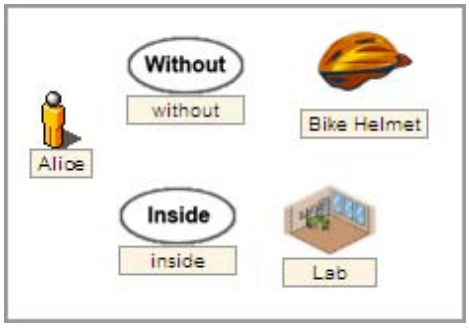


Figure 5.7: **A screenshot of the initial Scenic prototype illustrates how the lack of explicit conjunctions can lead to awkward and ambiguous event specifications.**

5.4 Limitations of Initial Scenic Design

Our preliminary evaluation showed that even non-technical end-users could indeed understand and use Scenic to produce meaningful event specifications. However, a number of usability and expressivity problems were identified as well. First, a number of participants requested time-saving features such as copy-and-paste for Actors and Scenes. Many also felt that the implicit nature of conjunctions within a Scene (e.g., all Primitives in a Scene apply to all Actors in that Scene) imposed awkward constraints. For example, in Figure 5.7 it is unclear whether Alice is inside the lab without her bike helmet, or if the bike helmet is in the lab without Alice.

Additional limitations involving timing also became apparent. Some participants wanted to specify constraints on absolute time in order to indicate that a particular event could only occur in the morning. Another problem was that the transition between each Scene was assumed to be instantaneous. As such, events that occur over a longer period of time (e.g., Bob leaves his office and then leaves the building a few minutes later) were awkward if not impossible to specify.

Overall, and most importantly, the initial Scenic design was limited in that it supported a smaller set of events than the substrate described in Chapter 4. The event detection substrate is equipped to detect complex conjunctions and events that occur over time. Indeed, the resulting detected events could also be easily filtered to meet absolute timing

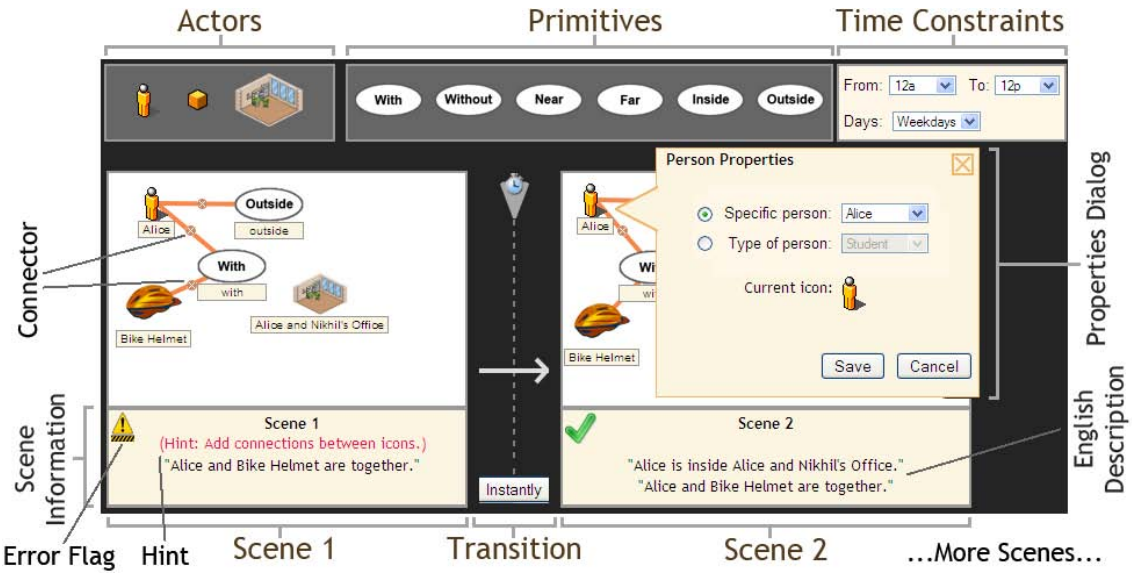


Figure 5.8: **The revised Scenic interface employed by the Panoramic tool. Includes explicit Connectors between Actors and Primitives and Transitions between Scenes. Also includes proactive syntax checking and debugging support as discussed in Section 5.6.**

constraints. In the next Section we describe the event specification interface for Panoramic, a redesign of Scenic that is intended to address these limitations to better support a wider range of events.

5.5 Specifying Events in Panoramic

We addressed Scenic's limitations in the design of the Panoramic event specification interface (see Figure 5.8). Panoramic adopts all of Scenic's fundamental components (e.g., Actors, Primitives, Scenes, Sequences) but adds mechanisms to increase expressiveness and eliminate ambiguity. For example, Panoramic includes explicit *Connectors* between Actors and Primitives, enabling any combination of Primitives to appear within a single Scene. Explicit *Transitions* between Scenes are also included. Transitions are explicitly set by the user as occurring either *instantly* (i.e., in one timestep) or *over time* (i.e., some time may pass before the next scene occurs). This corresponds directly to linear edges and self-loops in the FSM query language used by our event detection substrate. Finally, we included simple constraints on absolute time (e.g., "before 12pm on Weekdays") as a convenience.

Figure 5.9: **A Single Scene event: "***I'm in my office* **", translates into a single state FSM that enters the accept state whenever the user is inside his office.**

Recall from Chapter 3 that six instantaneous events are most often used in location-aware applications: *with*, *without*, *inside*, *outside*, *near*, and *far*. Our event detection substrate supports these primitive events and their composition using conjunction and sequencing with self-loops to detect events over one or more MStreams. Panoramic is capable of expressing a large subset of the events our substrate can detect. We now describe and illustrate that set of location events.

Single Scene Events. Single Scene events use connected Primitves and Actors to describe a set of instantaneous primitive events that occur simultaneously. Scenes including one person or object (i.e., one MStream) can be translated into a FSM query having a single accept state and a single incoming edge that is satisfied when the Scene's Primitives are true (see Figure 5.9). When the Scene includes multiple people or objects, the Event Manager breaks down the query and detects the event as described in Chapter 4. The set of events that can be represented by a single Scene is greatly extended by Connectors. Single Scene events form the basis of all location-aware computing applications and are sufficient to account for approximately 70% of events in all surveyed applications from Chapter 3. They cover over 90% of events in consumer-oriented mobile location aware applications like friend finders, local search, and location-based social networking.

Consecutive Sequences Consecutive Sequences contain Scenes separated by instantaneous Transitions. They are translated into linear FSM queries that have a single accept state preceded by a sequence of states and edges as shown in Figure 5.10. Multiple MStreams are handled as discussed in Chapter 4. Consecutive Sequences are useful for detecting state transitions like entering or exiting a place, approaching an object, or beginning a trajectory;

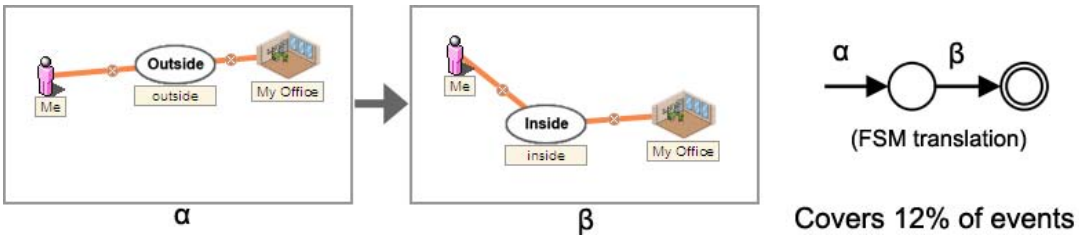


Figure 5.10: **A Consecutive Sequence event: "***I enter my office***", translates into a linear FSM that enters accept state when Scenes are satisfied consecutively.**

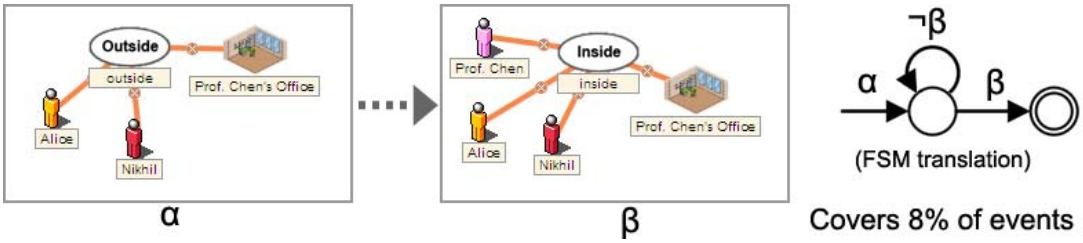


Figure 5.11: **A Sequence event with a gap: "***Alice, Nikhil and Prof. Chen begin a meeting in Prof. Chen's office***". This event occurs as soon as the last participant enters the office. A FSM with a self-loop is used to wait for the last participant.**

such events account for another 12% of events from the survey in Chapter 3. The majority of events that use consecutive sequences are very simple entered and exited events.

Sequences with Gaps Sequences may also contain Transitions that allow time (and potentially other events) to pass between one Scene and the next. Panoramic translates these Sequences into linear FSMs having a self-loop edge that is satisfied by the *negation* of the condition on the edge which leads to the next state. For example, the self-loop edge in Figure 5.11 ensures that the FSM remains in "pre-meeting" state until Alice, Nikhil, and Prof. Chen are all in Prof. Chen's office. Sequences with gaps represent a class of events not previously supported by Scenic, thus increasing the flexibility and expressiveness of specifications. They also account for another 8% of the events in Chapter 3, mostly those that detect a path or a complex sequential workflow.

Disjunctions of Sequences A disjunction of Sequences may represent different orderings of instantaneous events or alternate paths in a workflow. While such events comprise less than 1% of events in the surveyed literature, they are of growing importance in emerging

domains like workflow monitoring in hospitals [255]. Indeed, they account for more than 20% of events in enterprise applications for hospitals. Panoramic does not directly support disjunctions, but users may specify multiple Sequences and compose their disjunction within an application. Here we stress that our substrate has the capability to support disjunctions of Sequences and that we plan to extend Panoramic to directly specify them in future work.

Actor variables also create disjunctions of Sequences. Most surveyed events can be usefully modified with variables. For example, instead of "*Alice meets Nikhil in her office*", one could express "*Two researchers meet in any room*". These events are translated into a disjunction of Sequences where the variables in each Sequence are replaced with a different combination of possible values. Our substrate can efficiently support such events when there are few variables that range over a limited domain.

Unsupported Events There are another 9% of surveyed events for which Panoramic provides limited or no support. This includes repeating sequences, which can be translated into FSMs with cycles - but which Lahar (even with the Event Manager) cannot detect. While these queries are computationally difficult to answer over probabilistic data, they can be approximated by "unrolling" the cycle. Other common and currently unsupported location events involve precise 3D location, orientation, and events involving speed of state transitions (i.e., velocity).

5.5.1 Formative Evaluation

In addition to extending the set of supported events to better match the capabilities of our event detection substrate, the design of Panoramic's event specification interface was driven by a formative user study. In this study, 12 non-programmers were given a brief tutorial on Panoramic and asked to complete a series of event specification tasks while talking aloud. Each task presented participants with an example application and usage scenario for which they were asked to specify an appropriate event. After participants declared a task complete, a researcher would then review the produced specification and explain exactly how Panoramic would interpret it. Participants could then revise their specification if the researcher's explanation did not match their intention.

As a result of the study, we identified a variety of usability and expressivity problems that we addressed with the design features presented above (e.g., Connectors, Transitions). However, we also observed several more fundamental problems regarding user ability to understand and verify the behavior of a created specification. These problems echo those discovered in the Scenic evaluation from Section 5.4. We discuss these problems in the next section.

5.6 Problems in Specifying an Event

Even with the enhancements to the Scenic interface, users encountered a variety of difficulties while authoring event specifications in Panoramic. Here, we summarize and discuss those most frequently observed in our formative study.

5.6.1 Syntax Errors

Nearly all participants produced one or more syntactically illegal specification. Most syntax errors were the result of forgotten connections or targeting errors while rapidly creating a Scene. In a few cases, participants made mistakes because they did not clearly understand how Actors and Primitives could be connected. We addressed these problems with three new features (see Figure 5.8). First, we constrained the interface to allow only legal connections between Actors and Primitives, thereby eliminating the possibility of syntax errors. We also added an *information panel* below each Scene that displays English explanations for fully connected Primitives in that Scene. Finally, we included a *status flag* in the information panel that shows a green checkmark when a Scene is legally specified or an under construction symbol when more work needs to be done. If more work is needed, the information panel provides a hint as to what elements (e.g., Actors, Primitives, Connectors) are needed to complete the Scene.

5.6.2 Design Problems

A problem encountered by 3 of the first 5 participants was in deciding on a specification design that would meet the task's requirements. While 1 participant had difficulty rea-

soning about what needed to be specified, others knew what they wanted to specify but were not sure how the available widgets could be composed to do it. To address both of these problems, we introduced *event templates*, stock specifications that provide examples of common events with their usage scenarios - at least one for every type of event in the taxonomy. The last 7 participants in our formative study were provided with a library of templates during the study session; those that encountered design challenges were able to use the template library to decide on a design.

5.6.3 Problems with Timing

Half of the participants chose to revise a completed specification due to problems with timing. These problems consisted in use of the wrong Transition type (e.g., *instantly* instead of *over time*) or in using one Scene when two Scenes were needed. For example, a participant intended to specify "the custodian leaves the lab and goes to the closet" as two Scenes separated by an instantaneous Transition. This specification was flawed because a custodian cannot move instantly between rooms. Another participant used a single Scene to program a context-aware notifier to send her an email when a meeting occurs. While the single Scene would effectively detect a meeting and send an email, it would also occur repeatedly throughout the meeting, causing many emails to be sent. Event templates helped to reduce problems with timing, but some more subtle problems remained, forcing participants to revise their specifications. We therefore developed additional solutions to timing problems which we present in Chapter 6.

5.6.4 Tuning the Level of Specificity

A critical challenge faced by all participants while designing and revising their specifications was to determine the appropriate level of specificity. Some participants routinely *under-specified* events by leaving out Actors or Primitives. One explained her under-specification by saying "I wanted it to cover every case so I only need one event," another simply explained that it took two revisions before he realized that another object was needed. *Over-specification* was also common. In such cases, participants often explained that they added

non-essential Actors, Primitives, or Scenes because they weren't confident that Panoramic would detect the intended event without them. For example, one participant constrained a "group meeting" event to occur only when all group members were together in a room with laptops and coffee mugs. He explained that "they could be in the room for some other reason, but if they all have laptops and coffee then they're probably in a meeting." Whether an event is over or under specified, the result is a specification that does not fit the user's intention.

While mismatches in specificity may be less of a problem when users are reasoning about events in their own life with which they are intimately familiar, some tuning is likely to be required whenever a new event is specified. Many participants adopted a trial and error methodology when specifying an event, using the researcher's explanation to "test" an event's behavior over multiple design iterations. This motivated the creation of additional interface components that support end-users in understanding and verifying the behavior of a specification. We discuss these components in the next Chapter.

5.7 Summary

In this section, we have presented a tool that addresses the third key challenge facing many location systems: that of deploying and tuning applications. The Scenic tool and its successor, Panaoramic allow a significant portion of important location events to be specified using an iconic visual language that features direct manipulation and an intuitive storyboard metaphor. We have shown through several user studies with non-expertss that this design is both understandable and attractive to end-users. We have also uncovered several usability problems regarding the correctness of specified events. While we address some of these problems through minor interface changes, others must be addressed through more substantial mechanisms discussed in the next Chapter.

Chapter 6

EVENT VERIFICATION TOOLS FOR END-USERS

The event detection substrate and end-user tools presented in Chapters 4 and 5 allow end-users to specify an important family of events for detection over even sparse and uncertain location data. However, as identified in Section 5.6, it is also critically important that end-users be able to *verify* event specifications and debug those that do not work. Verification is difficult because it requires a system to produce high-level evidence of a specification's behavior over sensor traces that may be too complex for an end-user tool to generate. Moreover, because bugs can occur at the sensor level (e.g., calibration errors) or in the specification design, users must be able to understand detected events and evaluate their relationship to sensor data. It is impractical or impossible to achieve such an understanding when events are specified with inscrutable machine learning models or when they do not represent uncertainty. As such, existing systems for end-user event specification are limited by inadequate support for verification and debugging [60, 89, 203, 206, 229, 304, 322].

In this chapter, we present extensions to the Panoramic tool that support event verification by leveraging our substrate's support for intuitive, pattern-style event specifications that account for uncertainty with probabilities (see Figure 6.1). These extensions do not require users to write code, understand complex models, perform elaborate demonstrations, generate test traces, or blindly trust deterministic events. Instead, they present an intelligible verification interface based on pattern matching, and which uses readily available historical sensor data. Specifically, we contribute a mechanism for intelligible, hierarchical visualization of context using widgets that even allow users to distinguish sensor errors from design errors. We also contribute a novel approach to verification that leverages both historical location data and a user's knowledge of past events. Finally, we present a qualitative evaluation of Panoramic's event verification interface with 10 non-programmers and real RFID data from the RFID Ecosystem. We discuss these contributions in detail below.

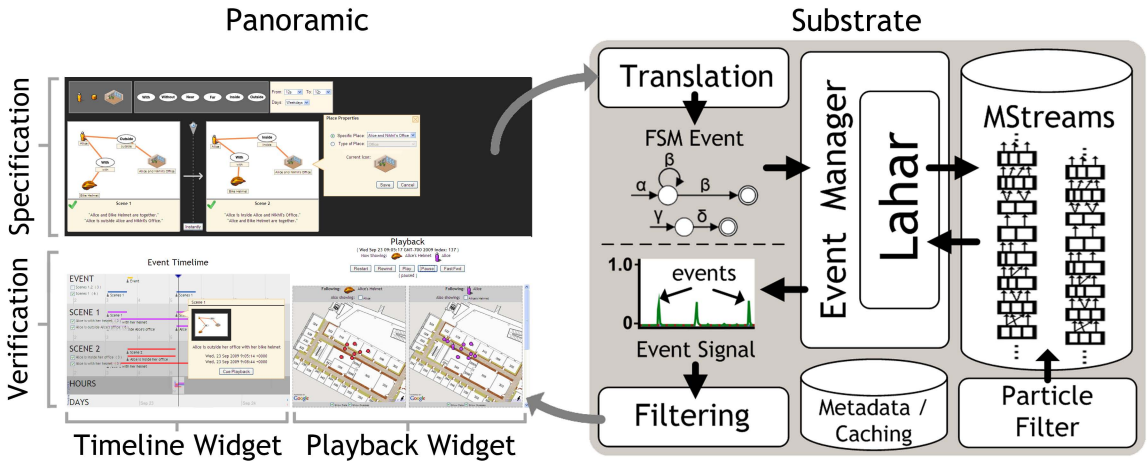


Figure 6.1: **The Panoramic system architecture. Events are iteratively specified in Panoramic, detected over historical location traces by our event detection substrate, and then displayed in Panoramic's verification interface.**

6.1 Understanding and Verifying Events

In response to the previously discussed event specification problems, we extended Panoramic to support end-users in understanding and verifying their event specifications. Here we face the hard problem of generating test data for specifications. Our approach is to allow users to assess the correctness of a specification by running it on *historical data*. We use our substrate to detect the event together with the timeline-based and map-based widgets presented below to visualize the results. The advantage of this approach is that it does not require complex simulations or synthetic data which may not be truly representative. Instead, it reveals the behavior of the event on real, readily available sensor data. The obvious constraint is that the tested event must have already been recorded by the user's RFID deployment. This is a reasonable sacrifice for enterprise environments (e.g., hospitals, office buildings) where both simple events and complex workflows occur repeatedly.

6.1.1 Timeline Overview

We developed a timeline widget (see Figure 6.2) that provides a rapid overview of detected event results as horizontal *bars* in a timeline where a bar's start and end points correspond

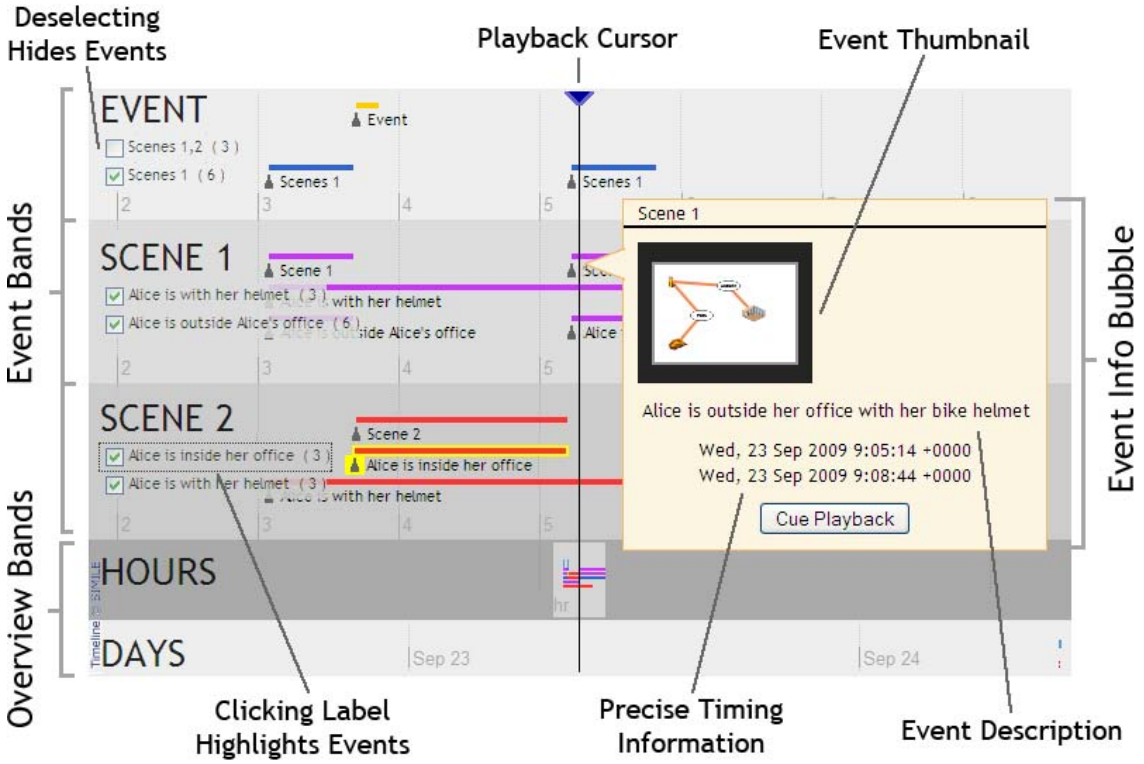


Figure 6.2: **The timeline reviews the event "Alice enters her office with her helmet".**

to a detected event. Events are organized in groups of sub-events (e.g., sub-sequences, Scenes, Primitives) in order to provide an in-depth explanation as to why an event was or was not detected. Each group of events is displayed in its own *band*, with one band for the event itself along with its sub-sequences, and one band for each Scene with its Primitives. By correlating the sub-event bars with the presence or absence of an event bar, users can gain an understanding of how each sub-event contributes to or detracts from the detection of the event. For under-specified events, this process can reveal that frequently occurring sub-events result in a larger than intended number of detections. In the case of over-specification, it can show a user that absent sub-events are preventing the event from being detected. Timing problems are also visible as unexpectedly long or short event durations.

The timeline further facilitates the verification process with exploratory browsing functions. The timeline can be dragged left or right to move through time, and two zoomed-out

bands (one for hours and one for days) provide additional context for large datasets. By default, event bands are rendered with minute-level granularity but may be zoomed in or out using the mouse scroll wheel. Checkbox labels on the left side of each band describe the contents of each sub-event group in that band. The bars representing a sub-event can be shown or hidden from the timeline view by checking or unchecking that sub-event's checkbox. Selecting a checkbox label will highlight all occurrences of an event and its sub-events in the timeline. Finally, clicking a bar in the timeline brings up a bubble containing a thumbnail image of the corresponding event or sub-event along with an English description and precise timing information.

The timeline-based design is particularly well-suited for display of sequential data and allows users to leverage their natural ability for reasoning about temporal events [188]. With a suitably large dataset, we anticipate that the timeline view can help users to quickly gather evidence of a specification's behavior.

Filtering Event Signals

The timeline must present a simple, discrete view of the complex probabilistic data it displays. As such, we take several steps to transform raw event signals from Lahar into discrete event streams that are amenable to visualization. First, because the probability of a true event occurrence may vary widely, we identify and flag all local maxima in a signal as potential event occurrences. We then filter out all spikes (i.e., peaks lasting less than 2 seconds) from this set. Spikes are unlikely to correspond to a true event because they are faster than any action humans commonly perform. They may occur in an entered-room event signal, for example, when a user passes but does not enter the room. After removing spikes, we transform the remaining "humps" into discrete events having the same duration. These discrete events can then be displayed on the timeline.

6.1.2 Detailed Playback

Though it provides a useful overview and a direct look at the cause for Sequence and Scene events, the timeline does not explain why a given *primitive event* does or does not

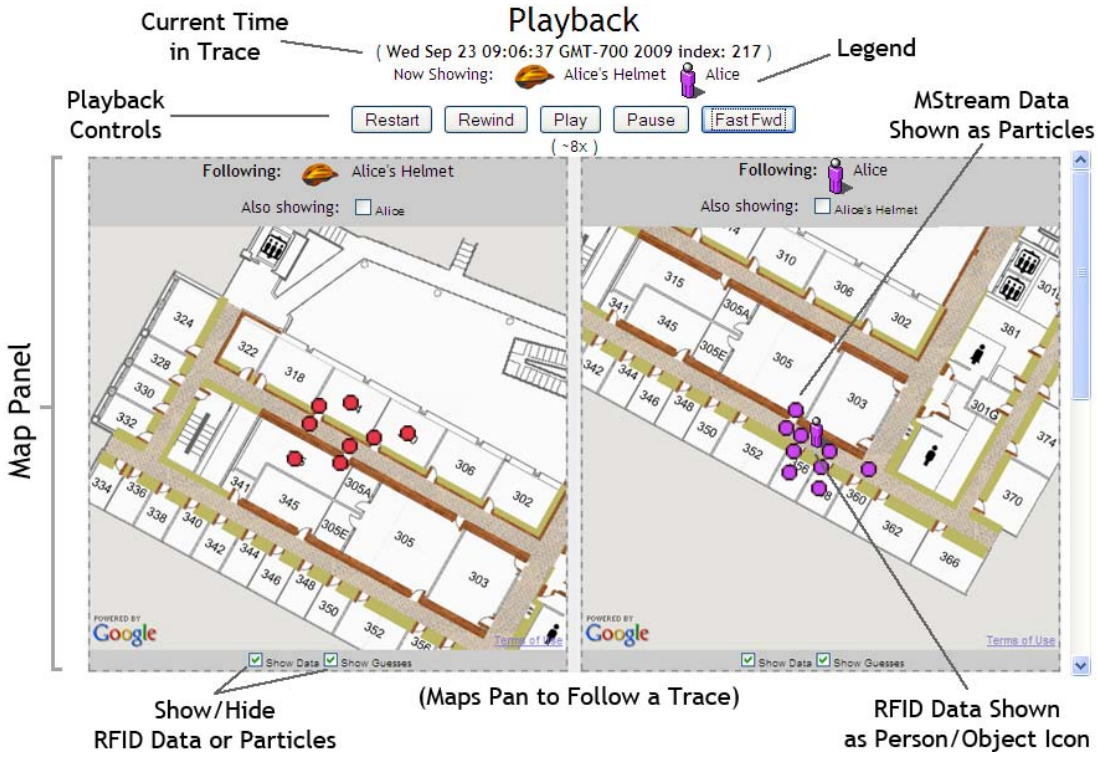


Figure 6.3: **The playback widget reviews traces for Alice and her helmet. Person and helmet icons represent raw location data. Particle icons display a probability distribution over a tracked entity's possible locations. Note that in the displayed timestep there is no raw sensor reading for the helmet, so no helmet icon is shown.**

occur. This is a crucial question when working with real historical sensor traces because missed RFID readings can lead to false positives or negatives that are indistinguishable from unexpected behavior in the timeline. For example, a "group meeting" specification may match a user's intention but fail to work in practice because one group member's RFID tag is routinely missed. Using only the timeline, it would be impossible to distinguish an absent group member from a tag that needs to be replaced. To help users identify problems that are rooted in sensor errors rather than in a specification, we developed a map-based trace playback widget. The widget allows a user to semantically drill-down into any point in the timeline and review both the sensor trace and the MStream starting at that time.

The playback widget renders at the right side of the timeline (see Figures 6.1 and 6.3) whenever a user clicks the "Cue Playback" button in the pop-up bubble for a timeline bar.

At the same time, a playback cursor appears over the timeline to designate the current time in the trace to be replayed. Users may also be interested in portions of the timeline that contain no detected events, as such, they can drag the timeline to any location to cue playback there. Standard video controls (e.g., pause, play, stop, rewind and fast-forward) provide a familiar interface for reviewing segments of the trace. Playback occurs in a collection of *map panels* that show RFID readings and MStream data (i.e., particles from the particle filter) overlaid on a map of the RFID deployment. The timeline scrolls synchronously with the map-based playback. Each such panel follows a particular person or object from the trace, automatically panning and switching floors as needed. The user may also choose to show multiple traces in a single map panel by selecting checkboxes above the map that correspond to additional traces. RFID readings and MStream data may be switched on and off in a given map panel using the "Show Data" and "Show Model" checkboxes. Unneeded map panels can be collapsed to facilitate side-by-side comparison of traces. Labels at the top of the playback widget show the current time in the trace and summarize the collection of people and objects being viewed in the trace.

6.2 Implementation

Panoramic's event verification interfaces are entirely web-based and built using the Google Web Toolkit [140]. The playback widget was built using Google Maps [127], and a customized version of MIT's Simile Timeline [298] was used to implement the timeline widget. In total, Panoramic contains 152 Java classes that control the interface and orchestrate AJAX communication with a server. An additional 42 classes were used to support translation and execution of Panoramic event specification with our event detection substrate.

6.3 Evaluation

We ran a qualitative study of Panoramic's event verification capabilities with 10 non-programmers. Participants were offered $20 to perform 60 minutes of event verification tasks. In addition to a 10 minute tutorial, participants were prepared with a background story that described a week in the life of Alice, a fictional student. The story included precise information on events that occurred (e.g., "group meeting on Monday at 11") as

well as information on events that often occur (e.g., "Nikhil often stops by Alice's office to talk"). Each task included a specification, a description of the application and usage scenario for which Alice created the specification, and the corresponding detected events from the week of Alice's data. Participants were asked to talk aloud as they used Panoramic in combination with what they knew about Alice's week to decide how each specification worked, whether it met her needs, and how it could be fixed.

The week of historical data was spliced together from traces collected in the RFID Ecosystem and for which we have ground truth information on when and where events occurred. We combined a set of high fidelity traces with a small number of highly ambiguous traces (e.g., traces with a large number of missed tag reads). The first four tasks were presented in random order with one task having a specification that clearly succeeded over the week of data and three that failed because they were over-specified, under-specified, or contained a timing error. A fifth task contained a specification that should have met Alice's needs but was not detected over the week's data as a result of ambiguous sensor traces.

6.3.1 Observations and Enhancements

Overall, participants were able to complete the tasks, averaging 15-20 minutes for the task involving ambiguous traces and 10-15 minutes for all other tasks. All participants understood the behavior of specifications and could distinguish sensor errors from specification errors. Participants were also able to grasp the intended behavior of a specification both from the usage scenario and from the specification itself. As such, they used the timeline and playback widgets as a means for verifying intuitions about a specification's behavior rather than for exploring its overall behavior. Moreover, though overconfident participants initially declared a flawed specification to be correct in 6 of 50 tasks, they quickly changed their minds after comparing the timeline to their knowledge and intuitions about events. All participants were comfortable using Panoramic and several remarked that it was fun or "like a game". However, while they did not encounter any critical barriers to task completion, many participants faced recurring difficulties for which we have developed preliminary solutions (see Figure 6.4).

First, participants often checked the timeline for consistency with the events they knew occurred during Alice's week. In many cases the first question they tried to answer was "how many times was the event detected?". The timeline does not directly answer this question, so participants had to scroll through the week to count event occurrences. We addressed this problem by adding a count for each event beside that event's label at the left side of the timeline.

The task involving an under-specified event required participants to further constrain the specification by adding an object. This was difficult because the set of available objects was buried in the specification interface, leaving participants unsure of what objects were available. Moreover, without the ability to review traces for other Actors alongside the currently loaded trace, it was difficult to decide whether or not another Actor was relevant to an event. While this problem may be less critical when users reason about people and objects they know, we did introduce a new section to the legend at the top of the playback panel that shows other Actors which may be relevant to the event. By clicking the checkbox next to an Actor, users can load a new map panel that plays the trace for that Actor. The set of displayed Actors is currently chosen as those that are proximate to, or move in the same time window as the currently loaded trace. This is a reasonable compromise to the unscalable alternative of displaying every possible Actor because proximate or moving Actors are likely to be more relevant.

Two additional recurring frustrations were voiced by participants when using the playback widget. First, they had difficulty understanding the visualization of the MStream as a set of particles. After explaining the particles as "Panoramic's guess at where a person or object is," participants were better able to understand but still had difficulty reasoning about sensor errors and missed detections as a result of ambiguity. As such, we changed our rendering of particles to include an opacity level that corresponds to a particle's probability. This helped the last 8 of 10 participants to identify sensor errors in the ambiguous trace 3-5 minutes faster than the 2 who did not have this feature available. A second difficulty was that participants felt it was difficult to correlate the trace playback with the events in the timeline. Although the timeline was animated to correspond to the trace, participants were uncomfortable looking back and forth between the trace and the timeline. One participant

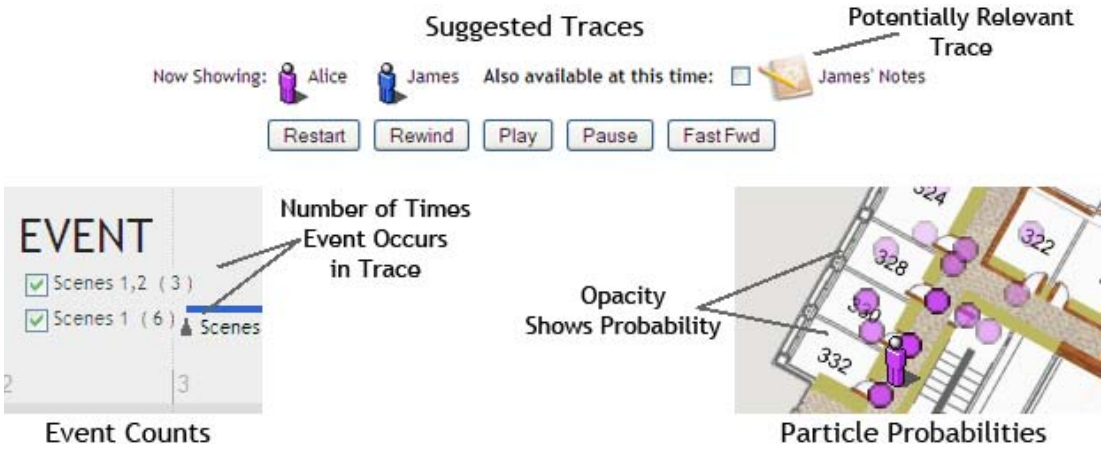


Figure 6.4: **Enhancements to the timeline and playback widgets.**

explained her difficulty with the playback widget by saying "it shows where Panoramic thinks the people are, but not what it thinks about the events, you have to keep looking back at the timeline to see the events, that's hard". This problem could be addressed in future work by arranging the timeline below the playback widget, or by embedding pop-ups in the playback maps that mark when and where events occur.

6.3.2 Limitations

Panoramic currently has several key limitations. First, while it supports specifications with Actor variables (i.e., icons that represent a *type* of person, place, or thing rather than a specific person, place, or thing), verification of such events is tedious because users may need to review multiple sets of historical traces - one for each possible parameterization of the variables. Panoramic is also limited by its minimal support for suggestions on *how* to debug a specification that is incorrect (see Chapter 9 for future work in this area. While a list of potentially relevant Actors is useful, more intelligent suggestions could be made by assessing the sensitivity of event detection results to slight changes in a specification. Finally, Panoramic's reliance on historical data may be problematic for events that seldom occur. Here it may be possible to automatically generate synthetic test data for a particular specification using techniques similar in spirit to recent work by Olston *et al.* [243]. Future work will address these and other limitations.

6.4 Summary

In this chapter we presented the design and evaluation of extensions to Panoramic that allow end-users to *verify* the location events they specify. Our design leverages the event detection substrate from Chapter 4 to contribute an intelligible, hierarchical visualization of location events that accounts for uncertainty. The design even allows users to distinguish sensor errors from design errors by reaching all the way down to the sensor data and its correponding MStream to replay the relevant portions of a sensor trace. We evaluated the presented verification interface with 10 non-programmer study participants and found that in spite of minor difficulties with the interface, all users were able to complete five representative verification tasks. As such, we have made a substantial contribution toward meeting the three key challenges articulated in the introduction through our work on the event detection substrate in Chapter 4 and on the design of event specification and verification with Panoramic in Chapter 5 and in this chapter.

Chapter 7

MIDDLEWARE FOR SPECIFYING, DETECTING, AND MANAGING EVENTS

The previous chapters articulated the requirements for location-event services and presented a substrate with tools for event specification and verification that support a low barrier-to-entry solution. In this chapter, we assemble these core contributions into one middleware for location event specification, detection, and management called Cascadia. We describe Cascadia's overall architecture and how it extends the substrate from Chapter 4 with additional support for metadata management and a straightforward API that is used by Scenic and Panoramic, and is available to all event-based applications. We also demonstrate Cascadia's utility with an application building case study.

7.1 Architecture

Given the event substrate from Chapter 4, the Cascadia middleware simply adds an intutive API along with services to support it. While we discuss the API in the next Section, we present the supporting metadata management, event filtering and aggregation, and event template services here. All these services are implemented within the Event Manager component of the substrate.

7.1.1 Managing Metadata

Every event specification is associated with a large amount of metadata. This includes the names and IDs of the entities (e.g., people, places, things) involved as well as the name, ID, and preferences of the user that owns them and the name and specification of the event itself. Cascadia manages a persistent relational store of this metadata and exposes services for accessing it. In particular, there is one table, `User`, that stores user names and unique IDs with the schema `(user_id, user_name)`. The user's ID is then used as a foreign key

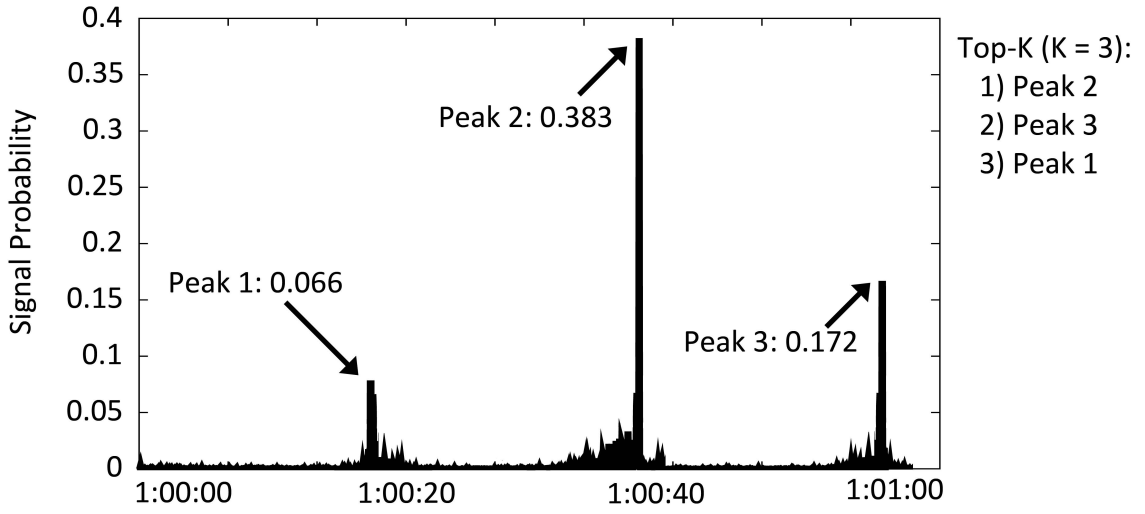


Figure 7.1: **An illustration showing the peaks extracted from an event signal by Cascadia. The three peaks are extracted from the top of spikes in the signal and aggregated in a top-K list.**

to associate entities with a particular user in the tables `Person`, `Place`, and `Thing`, all of which have a schema that includes an entity ID, a name, and a type. Entity types (e.g., "student", "faculty", "book") are further defined in an `EntityType` table. Events are stored and retrieved from an `Event` table that has the schema `(event_id, event_specification, event_flags)`. Additional helper tables store preferences and other auxiliary information used by the API. In general, it is easy to extend metadata management in Cascadia by adding additional tables to the database along with a small amount of corresponding code.

7.1.2 Event Filtering and Aggregation

Recall from Chapter 4 that the event detection results produced by our substrate are in the form of *event signals* that indicate the probability that a given event has occurred at each time step. Like the Panoramic tool (see Section 6.1.1, Cascadia filters event signals to extract "peaks" at which events are likely to have occurred. This process is illustrated in Figure 7.1. Filtering is a common and valuable task that supports many applications and users with higher-level, more easily understood events.

While all peaks can be provided in a stream, peak-based event filtering is also used to support top-K style aggregation of events over a particular time window. Here, each extracted "peak" has a probability corresponding to the top of the peak. Thus, a top-K result can be constructed by sorting the peaks within a time window by probabilty and returning the top-K.

7.1.3 Event Templates

Though users can define and store their own events using Panoramic, many applications are developed with particular types of events in mind. As such, and because developers cannot anticipate what entities will be relevant to a user, we included support for *event templates* in which some or all entities are variables. Developers define templates directly using XML generated by Panoramic. Users can load event templates in Panoramic to customize their specification. For example, a developer might define an event template, `MEETING` that specifies a meeting between two people in some room. Alice may edit `MEETING` with Panoramic to refer to she and Bob in the lab. Cascadia stores event templates in the `Event` table with a special template flag, and Panoramic can be invoked with a query string which indicates that it should load a particular event template.

7.2 API

Cascadia services are provided to programmers through the Cascadia client API. This Java API exposes entities and events as first class objects and supports both event-driven and query-based programming models. The central class is `CascadiaClient`, which maintains a connection to the Cascadia middleware and has interfaces for working with entities and events. Figure 7.2 shows how a `CascadiaClient` object is created with information on a particular user and connection details for the Cascadia middleware. We discuss the central classes and methods of `CascadiaClient` below.

```
// Client objects are created using CascadiaFactory
// user: username for the user making the connection
// password: the user's password
// host: hostname for the machine hosting the Cascadia middlware
// port: port number on which to connect to the middleware (default is 1234)

CascadiaClient c = CascadiaFactory.getClient(user, password, host, port);
```

Figure 7.2: **Code that creates a new instance of `CascadiaClient` for a particular user and connection to the Cascadia middleware.**

```
// Entity methods (similar for places and things)
boolean      addPerson(Person p);
boolean      addPersonGroup(PersonGroup pg);
boolean      removePerson(Person p);
boolean      removePersonGroup(PersonGroup pg);
List<Person> listPeople();
Person       getPerson(String name);
PersonGroup  getPersonGroup(String name);
```

Figure 7.3: **List of methods for working with entities in the Cascadia client API.**

7.2.1 Entity Interface

Applications need to reason about meaningful people, places, and things, not location sensors. As such, the entity interface represents entities directly using classes `Person`, `Place`, and Thing which can be users to create, update, and delete persistent entities. Each has a set of attributes representing an entity's name, ID, type, and owner, as well as *dynamic attributes* (e.g., `location` for `Person` and `occupants` for `Place`) that are continuously and automatically updated with a list of K most likely values. The API provides `addPersonGroup` and similar methods to define entity types. For applications with admin privileges, methods called `addLocNode` and `addLocEdge` can be used to define the connectivity graph that discretizes the space in a deployment. Figure 7.3 lists methods in `CascadiaClient` that are used to create, delete, retrieve, and edit entities.

```
// Creation and retrieval of event specifications
RegEvent       addEvent(EventDef definition);
boolean        removeEvent(EventDef definition);
List<EventDef> getEvent(String name);
List<EventDef> listEvents();

// Event subscription management
EventStream subscribe(Subscription subscription);
boolean     unsubscribe(EventStream stream);

// Event queries
List<Event> queryEvent(RegEvent re, long startTime,
                       long stopTime, boolean filtered);
List<Event> queryTopKEvents(RegEvent re, long startTime,
                            long stopTime, int k);
```

Figure 7.4: **List of methods for working with events in the Cascadia client API.**

7.2.2 Event Interface

Events are also first class objects in Cascadia, encapsulated by the `EventDef`, `Event`, and `EventStream` classes. The `EventDef` class includes an event name and specification in XML (generated by Panoramic) and represents an event specification. The `Event` class contains information on a particular occurrence of an event (e.g., event ID, timestamp, probability). Similarly, an `EventStream` provides a connection to Cascadia from which newly occurring `Event` objects are continually issued. We recognize from the survey in Chapter 3 that it is important to support both subscription (i.e., "push") and query-based (i.e., "pull") APIs for working with events. As such, in addition to supporting subscriptions through `EventStream` objects, `CascadiaClient` has methods for making one-time event queries. Figure 7.4 shows the event-related methods of `CascadiaClient`.

As an example, Figure 7.5 shows a code snippet for a digital diary application in which a STUDY-SESSION event is being defined, registered, subscribed to, and assigned an event handler. In the first statement the application creates a new client for a particular user which will connect to Cascadia event services on the specified host and port. In lines 4-7, the appli-

```
1  CascadiaClient c =
2           new CascadiaClient(user, password, host, port);
3
4  String   panL = <some PanL code>
5
6  EventDef def  = new EventDef(panL, schema,
7                               "Study Session", "Example Event");
8
9  RegisteredEvent re     = c.registerEvent(ed);
10 Subscription    sub    = new Subscription(re,true);
11 EventStream     stream = c.subscribe(sub);
12
13 stream.addEventHandler(this);
```

Figure 7.5: **A code snippet that shows event definition, registration, and subscription to an event stream as well as addition of an event handler.**

cation creates an `EventDefinition` object that defines, names, and describes the STUDY-SESSION event. The event is then registered with Cascadia using the `registerEvent` method; this method throws an exception if the specification XML is invalid, and returns a `RegisteredEvent` object if Cascadia successfully parses and adds the event. The application requests a new subscription to the STUDY-SESSION event stream on line 10 and specifies that it wants to receive filtered events rather than the raw event signal. Finally, the application adds itself as an event handler for the new event stream. Applications can implement the `EventHandler` interface in order to provide custom handlers for events coming from one or more `EventStream` objects.

7.3 Implementation

Cascadia comprises 122 classes and over 14,000 lines of Java code, not including comments or other boiler plate. We use Microsoft's SQL Server [310] RDBMS with the Java JDBC API [174]. Secure network communications are implemented using Apache's MINA framework [16]. Panoramic and other tools are hosted as Java Servlets on a Tomcat web server [17] from which they connect to a Cascadia middleware using the Cascadia API.

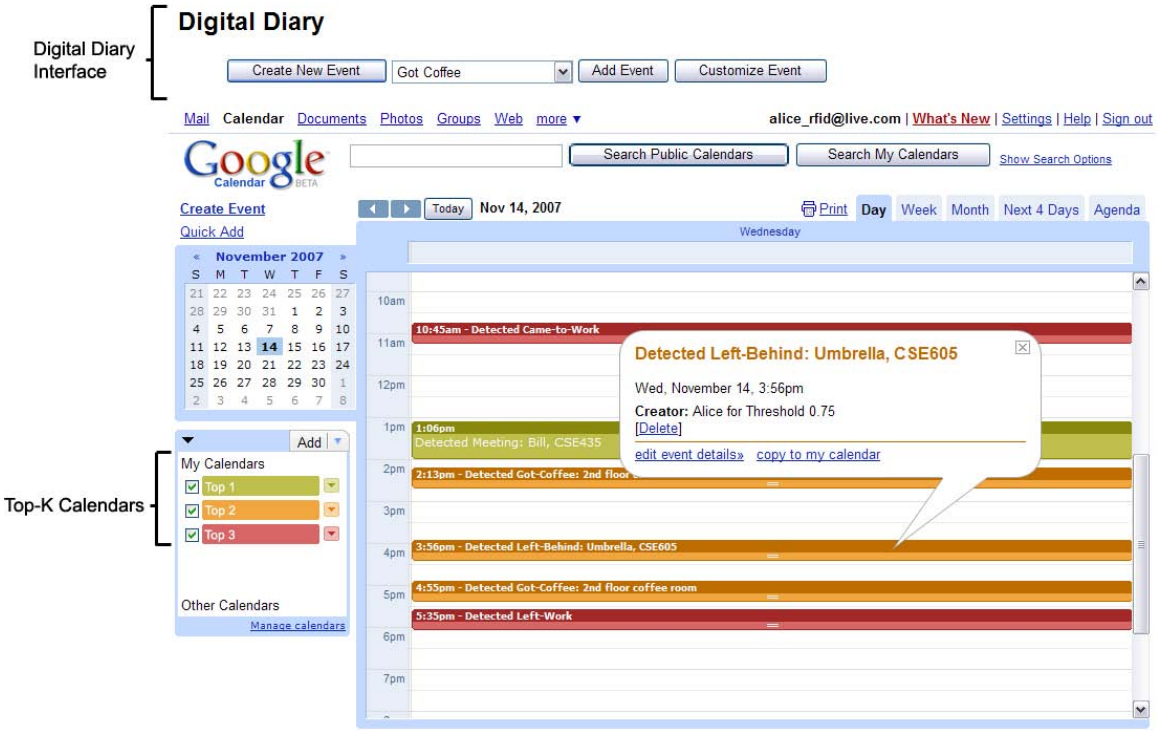


Figure 7.6: **A screenshot of the digital diary application.**

7.4 Evaluation with an Example Application

To demonstrate how Cascadia can facilitate application development, we implemented a digital diary application which records occurrences of user and developer defined events in a Google calendar [126]. This application could be a useful tool for analyzing how, where, and with whom one has spent one's time. Figure 7.6 shows a screenshot of the diary application.

The digital diary has two components: a daemon that continuously receives and posts newly extracted events, and a web-based calendar that the user loads to review her diary and edit its settings. The calendar supports three views in which the top-1, top-2, or top-3 detected events for each time period are displayed. The calendar also offers controls for adding, specifying, and customizing the events for the diary. Clicking "create new event" or "customize event" loads Panoramic in another window or tab of the browser.

Diary Component	Classes	Lines of code
Parsing config files and startup	8	400
Creating events and event streams	1	50
Incorporating user-defined events	1	50
Event handlers	1	25
Entity management	1	200
Event management and processing	3	300
Populating Diary with Google APIs	1	150

Table 7.1: **Table showing the approximate break down in lines of code for various parts of the digital diary application.**

The core logic of the diary daemon uses Cascadia's event-driven programming interface to receive and post newly extracted events. The daemon also implements event-specific logic for advanced handling of developer-defined events. For example, the diary comes with the built-in event `ENCOUNTER`, for recording encounters between people, as well as pairs of `MEETING-START` and `MEETING-END` events which record long duration meetings between people. However, the diary filters out redundant `ENCOUNTER` events that occur close in time to `MEETING-START` or `MEETING-END` events as people gather or disperse. An additional filtering thread that processes a queue of recently received events is used to implement this logic.

The diary implementation shows how Cascadia can greatly simplify application logic in location-aware applications. The diary took only 3 days to develop and consists of about 1,200 lines of Java code in 12 classes. The code break down is presented in Table 7.1. As shown in the table, the bulk of the application logic dealt with the filtering of Cascadia events (i.e., event management).

7.5 Summary

The Cascadia middleware embodies our probabilistic approach to event specification, detection, and management using uncertain sensor streams. It abstracts away burdensome details of intermittent and incomplete low-level sensor straems and reduces event detection to a pattern matching problem. Cascadia also supports a large and important family of location events and exposes this support to tools and applications through an intuitive API.

Indeed, the Scenic, Panoramic, Tag Manager, and Place Manager tools all use Cascadia to interact with a user's sensor, context, and metadata. We have demonstrated through previous evaluations in Chapter 4 and through a simple case study that Cascadia provides effective event detection services and can significantly reduce the time and effort required to write applications that user location events.

Chapter 8

RELATED WORK

The research presented in this dissertation has connections to work in a variety of areas. In this Chapter we review this related work to emphasize how our work differs and builds on it. We discuss efforts in several key areas corresponding to the key contributions of this dissertation: RFID systems, measurement, and optimization, RFID data cleaning and event management, pervasive computing infrastructures, systems for event detection and management, interfaces for event specification and verification, and support for intelligible context.

8.1 RFID Systems, Measurement, and Optimization

Passive RFID has been used in a variety of pervasive computing experiments over the last decade. An early paper on the subject by Want et al. [341] spurred a variety of projects that use RFID to detect interactions with objects [103, 104, 157]. This direction has culminated in a substantial amount of work on RFID-based human activity recognition [221, 254, 301]. Other work from the pervasive and mobile computing community has investigated the design and implementation of fine-grained location systems that use RFID [220, 239, 240]. Our work does not currently include detection of interaction with objects or precise tag localization. Instead we focus on systems, techniques and applications that use coarse-grained (e.g., room-level) localization. A variety of recent work has also focused on RFID in this capacity [32, 45, 187]. Most similar to our vision is MSR India's SixthSense system for RFID-based Enterprise Intelligence [268]. However, recent work on SixthSense has focused on automatic inference of object type and ownership information while we explore more explicit, user-centered mechanisms for tagging and management of objects in an RFID system. To our knowledge, the RFID Ecosystem is a much larger scale RFID deployment than any past pervasive or mobile computing research project has used.

Other work from the pervasive computing and networking communities has evaluated the performance of passive UHF RFID in carefully controlled laboratory studies. Ramakrishnan and Deavours [263] present a set of benchmarks for evaluating RFID performance in a variety of environmental conditions (including near metal and water). Hodges et al. [155] present a sophisticated, robotically automated technique for assessing the operating range of an RFID system. Buettner and Wetherall [50] measured EPC Gen 2 performance with a software defined radio and found opportunities for improvement in that standard's physical and MAC layers. All these studies contribute a great deal to our understanding of RFID performance and its influencing factors. However, they are all constrained to a laboratory setting and thus omit many of the conditions affecting pervasive RFID deployments such as ambient RF interference, tag mounting, movement and user behavior patterns. In a previous study [343] we looked at the performance of EPC Gen 1 RFID technology in a shorter, smaller scale pervasive deployment but did not recruit non-researcher participants nor did we collect as much ground truth information.

A variety of other projects have studied large-scale technology deployments like ours. Early projects like ParcTab [340] and Sentient Computing [6] were quite similar in that they offered indoor location aware services to a community of users who each carried a tracking device. Our work differs chiefly in that these projects used active, battery-powered tracking devices and were most appropriate for ubiquitous tracking of people and not objects. More recent work has measured sensor performance and mobility in both home and public spaces. Logan et al. [221] evaluated the effectiveness of various sensors in a long-term human activity recognition study and found that RFID was among the worst performing sensors. However, in this case the researchers were using a different type of RFID at short range to detect interaction with objects; this differs substantially from our target scenario. Measurement studies on wireless networks have found similar patterns in daily human mobility as evident in network utilization [30, 192]. Wireless network data is different in character than RFID data though because RFID data has more data sources (e.g.objects as well as people, and many more readers than access points) and because each RFID data source is typically less reliable than an access point.

Numerous consultancies, corporations and other stakeholders have produced case studies on the performance of real-world RFID deployments. Much of this work is focused on the supply chain and contactless payment domains which pose problems that are only peripherally related to our work. However, there has also been increased interest in passive UHF RFID for tagging and tracking patients, personnel and equipment in hospitals [66, 275, 324, 325, 326, 328]. Though work on RFID in hospitals typically emphasizes the requirements for cost effective deployment and business integration, it also raises many challenges we share: an indoor environment, tags on highly mobile people and objects, and increased chance of RF interference. As such, the hospital domain is an interesting area to watch and one to which some of our results may transfer well.

8.2 RFID Data Cleaning and Event Management

The large scale and low quality of RFID data has made RFID data management an attractive topic for database research. While *data cleaning* does not necessarily involve events, it is often performed in preparation for event detection and may indeed rely on some basic event detection mechanisms itself. As such, several techniques for cleaning RFID data have been proposed. In these techniques, users declaratively specify either the data cleaning algorithm [112, 175] or a pattern over the data with matching cleaning actions [264, 336]. In contrast, Cascadia performs probabilistic inference with a particle filter directly on the dirty data with no requirement for user-specified cleaning mechanisms. However, in Chapter 2, we did use a low-level cleaning technique to prepare TRE streams for further analysis. This technique was originally proposed by Jeffery et al. [176] and leverages simple low-level cleaning techniques that average measurements within a short time-window [176] and across a group of sensors covering the same area [175]. Such techniques could be integrated more deeply with Cascadia to provide other services (e.g., support for tag data analysis).

Other work on managing RFID data has focused on techniques for compactly representing, summarizing, and efficiently accessing RFID data. While most of this work has taken the form of RFID data warehousing [125, 159], the project that is most similar to Cascadia is the Siemens RFID middleware [125, 159]. The Siemens middleware is an integrated data management system with a rule-based framework that transforms RFID readings into

business logic. It is intended for supply-chain management, however, and unlike Panoramic its data model strongly emphasizes containment relations (e.g., tagged cases inside pallets). Furthermore, the rules system is deterministic and operates on raw RFID data rather than a probabilistic view.

8.3 Pervasive Computing Infrastructure

Many systems have been built to provide event services for pervasive computing. Several of these systems, such as the Context Toolkit [286], Gaia [281], Aura [307], Solar [61], and ConFab [156] have sought to address issues in addition to event services such as discovery, allocation, and management of resources, potentially heterogeneous and multimodal sensors, and distributed computing. In contrast, Cascadia focuses only on location data and assumes that an infrastructure for aggregating and centrally storing that data is readily available (as would be the case in a hospital, corporate campus, or smart home). Furthermore, none of these systems has focused on providing Cascadia's combination of user and developer-level support for expressive, declarative event specification, subscription, notification, and management on top of a probabilistic data model.

The ParcTab [340], Sentient Computing [6], Event Heap [178], and ConFab [156] infrastructures all propose data models similar to Cascadia's, yet they do not support uncertain base data and streams of probabilistic events. JCAF [33] presents an event-driven API with event subscriptions that are similar Cascadia's, but requires developers to write custom modules that perform event detection. ParcTab, Stick-e Notes [250], and ConFab provide declarative event specification languages, but these languages are substantially less expressive and do not leverage efficient query processing technology. In contrast, Liquid [148] and the Data Furnace project [115] do provide declarative, expressive event specification languages. Moreover, the Data Furnace project seeks to manage imprecise sensor data and probabilistic events. However, to the best of our knowledge, no algorithmic or system implementation details are published for either system.

8.4 Database Systems for Event Detection and Management

Event detection and processing has previously been addressed in three main areas of database research: active databases, streaming and complex event processing systems, and probabilistic data management.

8.4.1 Active Databases

Active databases [251] work by automatically detecting and responding to events that are taking place either inside or outside the database. At any time, an active database maintains a set of event specifications and associated *trigger* procedures that enact a programmed response to an event. While a variety of event specification languages, execution models, and architectures have been proposed [4, 25, 55, 119, 256], all such systems are deterministic and ignore the potential ambiguity and incompleteness of input data.

8.4.2 Streaming and Complex Event Processing

Streaming database systems have aimed to support continuous queries over incoming streams of sensor data. Early systems like Aurora [1] and Borealis [2] supported filtering and relational sliding-window queries but did not explicitly support events. More recent systems like SASE, SASE+ [7, 141, 351], Cayuga [48, 80, 82], and ZStream [232] have provided explicit support for event queries, but again without uncertain or probabilistic input streams. The Lahar system is the first such event processing engine to support probabilistic, correlated streams. It is notable, however that the FSM-based query language used by Lahar (and hence, Cascadia) is in fact based on Cayuga's query language.

8.4.3 Probabilistic Data Management

A set of recent probabilistic data management systems including MayBMS [190, 14], MystiQ [74, 270, 271, 272], and Trio [8, 37, 348] each support answers to a constrained set of queries over probabilistic data. Query complexity on such databases has been studied in [75, 76]. Probabilistic temporal databases have also been introduced [79], but they use a semantics based on probability intervals, which is different from ours. Moreover, none

of these systems supports event queries over probabilistic streams as in the Lahar system. Other systems such as BayesStore [334, 335], the Data Furnace [115], and those proposed by Kanagal and Deshpande [181, 182] have sought to embed graphical models in database management systems and thus to answer queries on uncertain data through *inference on these models*. While this innovative approach can accelerate inference and relieve system designers of the need to perform probabilistic inference, they are unlikely to scale effectively to large datasets the way Lahar can. In addition, none of these systems are designed specifically for use in detecting location events that concern people, objects, and places the way Cascadia is.

8.5 Interfaces for Event Specification and Verification

Panoramic can be viewed as an end-user software engineering tool for sensor systems. In this connection, a variety of similar systems have been investigated. We discuss them in this Section, focusing on specification and verification of events, and omitting discussion of techniques that specify event-triggered behaviors.

8.5.1 Specification Languages

Event specification systems for end-users are difficult to design because they must lower the barrier to entry without compromising expressive power. One approach is to create a specification language with abstractions that represent high-level concepts in the target application domain. For example, early systems like ParcTab [340], Stick-e Notes [250], and SPECs [203] used scripts to describe primitive location events. More recent work with Semantic Streams [347] and probabilistic context-free grammars [222] can detect some complex and even uncertain events in sensor networks. While these systems use data processing techniques similar to those in Cascadia, they expose languages that are not well-suited to end-users.

8.5.2 Specification Interfaces

Several systems have made specification more accessible with graphical interfaces that declaratively specify events. EventManager [229] used four drop-down boxes to specify a small set of primitive location events. CAMP [322] specifies non-sequence (i.e., instantaneous) events with a magnetic poetry interface that answers the questions: who, what, where and when. The Topiary [215] design tool also specifies instantaneous events, but uses an interface with an active map and a storyboard. Panoramic is quite similar to iCAP [304], a visual interface for specifying spatio-temporal sequence events. However, CAMP, iCAP, and Topiary are all less expressive than Panoramic because they rely on custom-coded event detection modules instead of a flexible event detection engine like Lahar. Moreover, they do not explicitly support event detection over uncertain sensor data like Panoramic.

8.5.3 Programming by Demonstration

Another approach is programming by demonstration (PBD), in which users supply example sensor traces to train an event detector. a CAPpella [89] is a PBD system that allows users to train a Dynamic Bayesian Network with labeled sensor traces. Apart from low detection rates, a CAPpella is difficult to use in our motivating scenarios because it requires 5 or more traces for training. Other systems focus on detecting simpler events with fewer examples and rapid feedback. For example, Crayons [99] and Eyepatch [228] enabled users to rapidly train visual classifiers using a demonstration-feedback loop. The Exemplar system [147] employs a similar loop featuring an algorithm that requires only one demonstration. Exemplar focuses on exposing an intelligible and editable visualization of its model, addressing the fact that automatically learned models are often inscrutable [60, 206]. These prior PBD methods become impractical for complex events that involve the simultaneous movement of people and objects.

8.5.4 Verifying Specifications

Past work has shown that end-users must be able to verify that a specification works as intended [60, 206]. Verification identifies three broad categories of error: (1) syntactic

errors that make specifications illegal or ambiguous, (2) semantic errors that make valid specifications behave in unexpected ways, and (3) sensor errors that cause event detectors to erroneously detect or miss events. Most languages and declarative interfaces use interactive visual feedback (e.g., error flags, prompts for disambiguation) to cope with syntactic errors [215, 304]. Semantic errors are often identified by testing with sensor traces (as discussed below). However, both CAMP and Panoramic can reveal semantic errors in non-sequence events by generating high-level English descriptions - in Panoramic these are descriptions of Primitives. Most systems provide no support for identifying sensor problems beyond what may be inferred from detection errors. In contrast, Panoramic directly supports discovery of sensor errors by providing a visualization that correlates sensor data with detected (and missed) events.

8.5.5 Trace-Driven Debugging

Test traces are commonly generated using either Wizard of Oz [215, 304], in which the user simulates sensor traces with a special interface, or demonstration [89, 147], in which the user enacts an event while recording it with sensors. Both of these techniques become prohibitively demanding for complex events. The Wizard of Oz approach also fails to capture the impact of uncertainty in real sensor data. Panoramic avoids these problems by using pre-recorded traces. Moreover, though other systems could adopt Panoramic's approach, they do not provide the support for archiving, exploration, and visualization of uncertain events and sensor data that Panoramic does. Debugging also requires that users correct erroneous specifications. This is a simple matter of modifying the specification in declarative interfaces like iCAP and Panoramic. It is much less straightforward in PBD systems [60], and may require that entirely new sets of demonstrations be recorded.

8.6 Support for Intelligibile Context

A key contribution of the Panoramic tool is that it presents high-level context information (i.e., complex location events) in a way that can be understood by end-users. Several papers have established and articulated the need for this [36, 65, 217]. In this Section, we discuss several projects that have also explored explicit support for intelligibility.

8.6.1 System Infrastructure for Intelligibility

There is a recent thread of research that explores support for intelligibility at the level of system infrastructure. Cheverst et al. [65] supported end-users with scrutable decision tree rules and context histories. Alternatively, the PersonisAD [20] framework allowed developers to programmatically access supporting evidence for context items. Dey and Newberger [87] support intelligibility in the Context Toolkit using Situation components that expose application logic to developers and designers. Panoramic adds to this body of work by providing an intelligible, scrutable context model for complex location events. Moreover, Panoramic contributes a context management system that directly copes with uncertainty using probabilities while not requiring users to explicitly specify probability thresholds.

8.6.2 Visualizations for Intelligibility

Other work has focused on generating visualizations of context data that help end-users understand that context. Early work on activity rhythms [35, 152] sought to detect, model, and visualize recurrent temporal patterns in user activity based on instant messaging status, calendar entries, and other digital information. The activity rhythms project used a timeline-based end-user visualization quite similar to Panoramic's but was less rich in meaning and did not allow semantic drill-down to further explain activities. Andrienko et al. [12] built a system that supports interactive visual analysis of GPS traces. While this system supports end-user understanding of location context, it is limited primarily to identification of significant places and common paths and does not support the complex conjunctions and sequences like Panoramic and Cascadia. The IMPACT system [214] supports awareness and motivation of physical activity through a web-based timeline interface that presents historical information on step counts. The interface also features semantic widgets that correlate step count with user-provided activity labels, thus helping users identify patterns in user activity. While similar in spirit to Panoramic, IMPACT does not currently support display of complex context information that is automatically extracted from sensor data. Instead, it uses deterministic user-provided labels to circumvent this challenge. The Situvis

tool [67] is very similar to Panoramic in that is uses historical context data and interactive visualization to test and refine event specifications. However, Situvis presents users with a substantially more complex Situation model and Parallel Coordinate visualization which limits access to end-users. Situvis also has no support for uncertain or probabilistic context data.

8.7 Summary

In summary, there have been many sensor deployment and measurement studies, and many systems for sensor data processing, event detection, probabilistic data management, or context-aware computing. There have also been many tools that support end-user interaction with context-aware computing systems. However, in this dissertation we focus specifically on support for complex location events. Moreover, no other research has combined probabilistic event detection with declarative event specifications, an intelligible end-user interface for interacting with context data, and evaluation using real sensor data from a building-scale sensor deployment. This unique combination allows Cascadia and Panoramic to leverage low-cost, unreliable sensors and provide flexible support for events while remaining accessibile to end-users.

Chapter 9

CONCLUSION AND FUTURE WORK

The key problem this dissertation addresses is that wide-spread adoption of context-aware computing systems is hampered by the cost of sensors and by the cost of the expert assistance needed to install and tune applications. To make this problem tractable, we constrain ourselves to one fundamental type of context, *location*, and one key abstraction for context-aware computing, *events*. We thus address the barriers to adoption of location-aware computing in two ways. First, we present the design, implementation, and evaluation of Cascadia, a middleware for specifying, detecting, and managing location events over uncertain data from low-cost sensors. Second, we design, implement, and evaluate Panoramic, a tool that reduces the cost of installing and tuning applications by allowing non-expert end-users to directly specify and verify the events they want to use in an application.

This dissertation makes six research contributions, C1 through C6, that address three major challenges. The first challenge is to lower the cost of location sensor deployments. This is difficult because most low-cost sensors provide only sporadic, imprecise streams of location data from which events cannot easily be extracted. To this end, in C1 we show how a layer of Bayesian inference can transform low fidelity location streams into smooth, probabilistic streams that are more amenable to event extraction. This allows us to replace state-of-the-art location sensors (e.g., active RFID) with less reliable sensors (e.g., passive RFID) that are more than an order of magnitude cheaper. We evaluate this approach and all other contributions with C2, a building-scale passive RFID location system called the RFID Ecosystem. We show that Bayesian inference can smooth away significant holes and ambiguity in sensor data streams, but only when sensors have sufficiently high success rates (i.e., read rates) to begin with. To this end, we conduct a series of benchmarking and longitudinal studies in the RFID Ecosystem that measure the fidelity and character of passive RFID data; we also identify several techniques that improve sensor performance.

Chief among our findings is that RFID performance in the field is substantially lower and more varied than in laboratory studies, but overall, RFID data is similar in character to data from other wireless deployments (e.g., WiFi), it compresses dramatically, and can be managed easily. We further identified the key factors influencing RFID tag performance (i.e., tag design, tag mounting, object material) and found that a novel multi-tagging technique could be used to double the success of tracking objects in field studies.

The second major challenge is to provide flexible support for the diversity of location events in applications that exist or have been proposed. Our first contribution here, C3, is to describe and analyze the space of location events by conducting an extensive survey of events in location-aware applications between the years 2000 and 2010. Through this survey, we distilled a taxonomy for location events that identifies six core, *primitive events* (i.e., inside, outside, near, far, with, and without) that are composed into *complex events* through sequencing and conjunction. While this taxonomy does not account for all location events, it does cover over 90% of those surveyed. Moreover, the taxonomy informed the design of the Cascadia middleware, C4. Cascadia leverages probabilistic data management techniques to support the specification, detection, and management of all events in the taxonomy over the layer of probabilistic data created by Bayesian inference in C1. We show through experiments with the RFID Ecosystem that Cascadia facilitates application development and that it can detect important location events even when uncertainty is very high.

The third challenge addressed by this dissertation is the need to reduce the cost of deploying and tuning location-aware applications. We address this problem with the design, implementation, and evaluation of the Scenic and Panoramic tools in contributions C5 and C6 respectively. The Scenic tool allows end-users to directly specify location events for Cascadia with an iconic visual language that embodies natural concepts of space and time. Through a series of laboratory studies we showed that non-expert end-users could indeed use Scenic to specify complex location events in 10-20 minutes per event. The Panoramic tool addresses the complementary problem of verifying events that a user specifies. By revealing the intelligible, hierarchical structure of detected events through timeline-based and map-based interface widgets, Panoramic allows end-users to evaluate, understand, and potentially

correct the behavior of their event specifications. We showed that 10 non-programmer study participants could use Panoramic to identify common specification errors and to distinguish sensor errors from specification errors.

9.1 Future Work

In this section we outline several areas of future work that build on the contributions of this dissertation. The proposed work is divided in two sub-sections, presenting near-term future work and long-term future work respectively.

9.1.1 Near-Term Future Work

Here we discuss areas for future work in the coming months. The proposed work focuses on techniques that augment Cascadia and Panoramic either to extend their capabilities or for improved performance.

Speculative Caching of Event Signals in Cascadia

Complex location events are composed of one or more component events, or *sub-events*. For example, the event "Alice is in the lounge with her coffee mug" is composed as the conjunction of events "Alice is in the lounge" and "Alice is with her coffee mug". These sub-events may be shared by other complex events, for example, the sub-events in the latter example are also used by the event "Alice and Bob are in the lounge with their coffee mugs". We saw in Chapter 4 that Cascadia detects complex events by breaking them down into sub-events, detecting those sub-events by breaking them down further or by evaluating them as primitives, and then merging the resulting event signals. Thus, significant time savings can be achieved when detecting a complex event over historical data if that event's sub-events have already been detected, with the corresponding sub-event signals stored and ready for reuse. Indeed, the more complex the sub-event, the more time is saved by reusing a cached result. Broad support for caching and reuse of event signals could lead to substantial performance gains.

As described in Chapter 4, Cascadia already supports caching and reuse of event signals when detecting events that have been specified by users. However, it may also be possible to speculatively detect events that *have not yet been specified but are likely to be used*. For example, a user Alice may spend a significant amount of time in the lounge, but it could be that no events have yet been specified that include the event "Alice is in the lounge". In this case, Cascadia could examine the data to learn that Alice is frequently in the lounge (e.g., because this place occurs repeatedly in her location stream). Cascadia could then speculatively detect the event "Alice is in the lounge", caching the resulting event signal as an MStream for later reuse. This extension would require the addition of a caching module to Cascadia that would periodically scan the historical Markovian stream data looking for events to speculatively detect and cache. The search for events could be based on simple heuristics concerning correlations among a group of people, people and places, or people and objects. However, it is likely that more intelligent heuristics and pattern recognition algorithms will also be useful for identifying which events to detect and cache event signals for.

Debugging Suggestions in Panoramic

As described in Chapter 6, Panoramic supports end-users in understanding and verifying the behavior of an event specification by visualizing detection results over historical data. When the detection results do not match what the user intended, it may be clear from the presence or absence of certain sub-events in the timeline widget that the specification must be modified in some way. However, it is not always be clear *how* a specification must be modified to correct its behavior. In certain cases, it may be possible for Panoramic to suggest modifications to the user by examining the detection results. For example, when an event is over-specified, it is likely that the event is seldom if ever detected because its sub-events do not all occur in the required way. In this case, Panoramic could identify the most infrequent sub-events and suggest that they be removed or changed to loosen the constraints. This would also require addition of a module to Cascadia that could support this type of data analysis, and new widgets for debug suggestions in Panoramic.

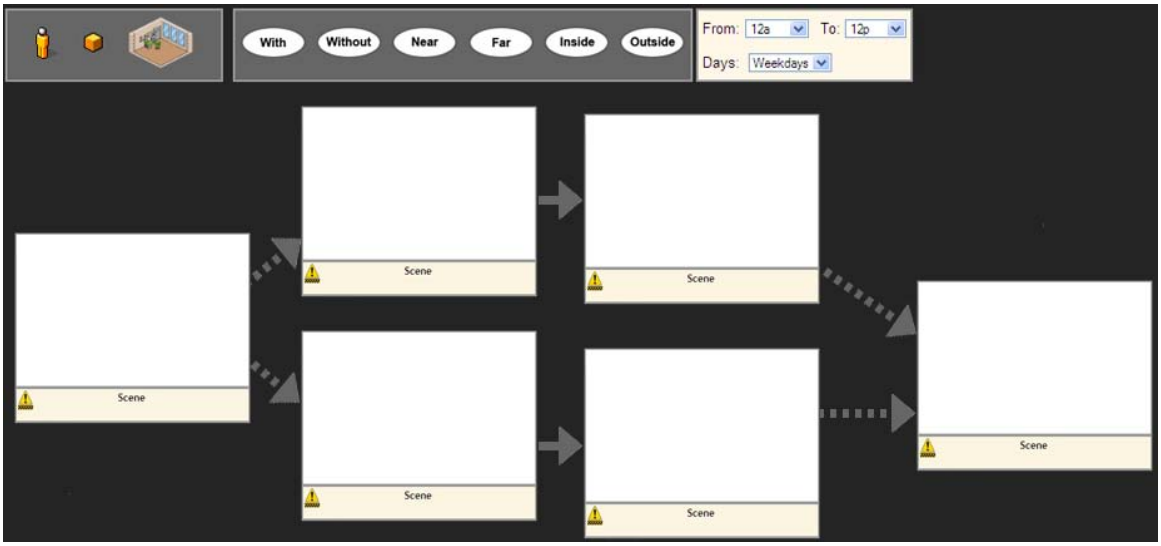


Figure 9.1: **A mock-up screenshot of the Panoramic tool adapted to specify events that include parallel sequences.**

Support for Parallel Sequences in Panoramic

Panoramic supports the specification of complex events as sequences. However, workflows in healthcare and other domains typically contan parallel sequences of events. While such events can currently be specified and detected as the disjunction of several sequence specifications, Panoramic should also support parallel workflows as a single specification. This would require extensions to the Panoramic interface that support specification of events with more complex structure. Specifications would consist of more free-form storyboards (with connecting transition arrows) than rigid sequences. Panoramic would also have to be modified to support translation of such specifications into disjunctions of sequences for Cascadia.

9.1.2 Long-Term Future Work

Here we discuss areas for future work that may span the next few years. The proposed work builds on Cascadia and Panoramic to provide support for other forms of context (i.e., non-location context) and modes of interaction with context data.

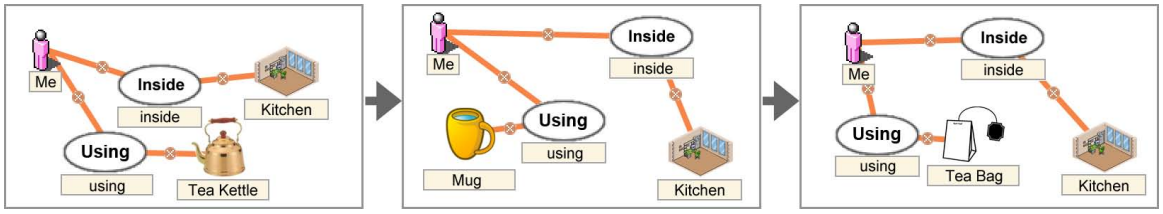


Figure 9.2: **A mock-up screenshot of the Panoramic tool adapted for specifying events that include non-location context.**

Support for Non-Location Context

Location context is fundamental to context-aware computing, but a substantial number applications use context that is derived from other sensors (e.g., accelerometers, microphones) in addition to location. It should be possible to extend Cascadia and Panoramic to detect a broad set of high-level activities using some additional sensor modalities. This would require that activities of interest be analyzed and appropriate activity primitives identified. Then an new inference technique on a graphical model (e.g., an HMM) would need to be developed to produce MStreams for these primitives. Panoramic would be augmented with interface primitives (e.g., "Using", "Speaking To") and widgets for specifying and verifying activities as shown in Figure 9.2, while Cascadia may need to be optimized to support more diverse "activity events".

Interactive Visual Analysis of Sensor Repositories

In addition to specifying and verifying context events as discussed above, a tool like Panoramic could be used to explore context data. Through their everyday activities, future consumers and businesses will leave a "context shadow" containing terabytes or petabytes of sensor data. This sensor data may contain extremely useful information on personal and business activities, but it is noisy, unstructured, and extremely difficult to transform into meaningful high-level context for end-users. By building Panoramic into an interactive visual analysis interface for a warehouse of context data (e.g., Cascadia), end-users could visually explore their context shadow. A variety of speculative caching, indexing and approximation techniques could be used to support the incremental queries and rapid visualizations required by such a system.

BIBLIOGRAPHY

[1] Abadi, D. J., et al. Aurora: A New Model and Architecture for Data Stream Management. *VLDB Journal*, 12(2), September 2003.

[2] Abadi, D. J., et al. The Design of the Borealis Stream Processing Engine. In *2nd Biennial Conference on Innovative Data Systems Research (CIDR'05)*, pages 277–289, 2005.

[3] Abadi et. al. The design of the Borealis stream processing engine. In *Proc. of the 2nd CIDR Conf.*, January 2005.

[4] Adaikkalavan, R. and Chakravarthy, S. SnoopIB: Interval-based event specification and detection for active databases. In *Proc. of ADBIS 2003 Conf.*, September 2003.

[5] Adaikkalavan, R. and Chakravarthy, S. SnoopIB: interval-based event specification and detection for active databases. *Data Knowl. Eng.*, 59(1):139–165, 2006.

[6] Addlesee, M. et al. Implementing a Sentient Computing System. *IEEE Computer*, 34(8):50–56, 2001.

[7] Agrawal, J. et al. Efficient pattern matching over event streams. In *SIGMOD '08*, pages 147–160, 2008.

[8] Agrawal, P. et al. Trio: A System for Data, Uncertainty, and Lineage. In *VLDB*, pages 1151–1154, 2006.

[9] Marcos K. Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar D. Chandra. Matching events in a content-based subscription system. In *Proc. of the 18th PODC Conf.*, 1999.

[10] Aguilera, M. K. et al. Matching Events in a Content-Based Subscription System. In *Symposium on Principles of Distributed Computing*, pages 53–61, 1999.

[11] Amelior ORTracker: Orchestrate Patient Flow. `http://www.pcts.com/unified/ortracker.php`, 2009.

[12] Andrienko, G. et al. Visual analytics tools for analysis of movement data. *SIGKDD Explor. Newsl.*, 9(2):38–46, 2007.

[13] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases. In *Proc. of the 22nd ICDE Conf.*, April 2006.

[14] Antova, L. et al. MayBMS: Managing Incomplete Information with Probabilistic World-Set Decompositions (Demonstration). In *ICDE*, 2007.

[15] Apache HTTP Server Project.
`http://httpd.apache.org`.

[16] Apache MINA.
`http://mina.apache.org`.

[17] Apache Tomcat.
`http://tomcat.apache.org`.

[18] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. STREAM: The stanford data stream management system. To appear in a book on data stream management edited by Garofalakis, Gehrke, and Rastog i.

[19] L. Arnstein, G. Borriello, S. Consolvo, C.Y. Hung, and J. Su. Labscape: A smart environment for the cell biology laboratory. *Proc. of the 1st Pervasive Conf.*, 1(3), September 2002.

[20] Assad, M. et al. PersonisAD: Distributed, Active, Scrutable Model Framework for Context-Aware Services. In *Pervasive 2007*, pages 55–72, 2007.

[21] AssetPulse. AssetPulse.
`http://assetpulse.com`, 2010.

[22] EPCglobal UHF Class 1 Gen 2.
`http://www.epcglobalinc.org/standards/uhfc1g2`, 2004.

[23] I. Avila-Campillo, T. J. Green, A. Gupta, M. Onizuka, D. Raven, and D. Suciu. XMLTK: An XML toolkit for scalable XML stream processing. In *Proceedings of PLANX*, October 2002.

[24] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proc. of the 21th PODS Conf.*, June 2002.

[25] Bacon, J. et al. Event Storage and Federation using ODMG. In *In Proc. of POS9*, pages 265–281, 2000.

[26] Bahl, P. and Padmanabhan, V. RADAR: An In-Building RF-based User Location and Tracking System. In *INFOCOM 2000*, pages 775–784, 2000.

[27] Balakrishnan, H. et al. Retrospective on Aurora. *VLDB Journal*, 13(4), December 2004.

[28] Magdalena Balazinska, Hari Balakrishnan, Samuel Madden, and Michael Stonebraker. Fault-tolerance in the Borealis distributed stream processing system. In *Proc. of the 2005 SIGMOD Conf.*, June 2005.

[29] Magdalena Balazinska, Hari Balakrishnan, and Michael Stonebraker. Load management and high availability in the Medusa distributed stream processing system. In *Proc. of the 2004 SIGMOD Conf.*, 2004. (System demonstration).

[30] Balazinska, M. and Castro, P. Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. In *MobiSys 2003*, pages 303–316, May 2003.

[31] Barbara, D. et al. The management of probabilistic data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, October 1992.

[32] J. E. Bardram. Applications of context-aware computing in hospital work: examples and design principles. In *SAC '04*, pages 1574–1579, 2004.

[33] Bardram, J. E. The Java Context Awareness Framework (JCAF) – A service infrastructure and programming framework for context-aware applications. In *Pervasive 2005*, pages 98–115, 2005.

[34] Barton, J. and Kindberg, T. The CoolTown User Experience. Technical report, HP Laboratories, 2001.

[35] Begole, J. et al. Rhythm modeling, visualizations and applications. In *UIST 2003*, pages 11–20, 2003.

[36] Bellotti, V. and Edwards, K. Intelligibility and Accountability: Human Considerations in Context Aware Systems. *HCI*, 16:193–212, 2001.

[37] Benjelloun, O. et al. Databases with Uncertainty and Lineage. Technical Report 2007-26, Stanford InfoLab, 2007.

[38] L. Bertossi and J. Chomicki. Query answering in inconsistent databases. In G. Saake J. Chomicki and R. van der Meyden, editors, *Logics for Emerging Applications of Databases*. Springer, 2003.

[39] U. Blien and F. Grael. Entropy optimizing methods for the estimation of tables. In *I. Balderjahn, R. Mathar, and M. Schader, eds.: Classification, Data Analysis, and Data Highways (Springer Verlag, Berlin)*, 1997.

[40] BlipSocial. BlipSocial. `http://blipsocial.com`, 2010.

[41] BlockChalk. BlockChalk. `http://blockchalk.com`, 2010.

[42] G. Borriello, M. Chalmers, A LaMarca, and P. Nixon. Delivering real-world ubiquitous location systems. *Communications of the ACM*, 48(3), March 2005.

[43] Borriello, G. Rfid: Tagging the world. guest editor's introduction. *Communications of the ACM*, 48(9), September 2005.

[44] Borriello, G. A Data Architecture for Consumer RFID Applications. In *MobiDE06*, June 2006.

[45] Borriello, G. et. al. Reminding about Tagged Objects using Passive RFIDs. In *Proc. of the 6th Ubicomp Conf.*, pages 36–53, September 2006.

[46] M. Boss. Matlab central file exchange - entrop. `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=5566&objectType=file`, 2004.

[47] J. Boulos, N .Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. MYSTIQ: A system for finding more answers by using probabilities. In *SIGMOD*, 2005. (System demonstration).

[48] Brenna, L. et al. Cayuga: a High-Performance Event Processing Engine. In *SIGMOD '07*, pages 1100–1102, 2007.

[49] Brunette, W. et al. Some sensor network elements for ubiquitous computing. In *IPSN '05*, 2005.

[50] Buettner, M. and Wetherall, M. An Empirical Study of UHF RFID Performance. In *MobiCom 08*, pages 223–234, September 2008.

[51] Buettner, M., et al. Recognizing Daily Activities with RFID-Based Sensors. In *Ubicomp 2009*, pages 51–60, 2009.

[52] Cardenas, T. et al. Testing Physical Computing Prototypes Through Time-Shifted & Simulated Input Traces. In *UIST 2008*, 2008.

[53] Castelli, V. et al. Augmentation-Based Learning Combining Observations and User Edits for Programming-by-Demonstration. *Know.-Based Syst.*, 20(6):575–591, 2007.

[54] Roger Cavallo and Michael Pittarelli. The theory of probabilistic databases. In *Proc. of the 13th VLDB Conf.*, pages 71–81, September 1987.

[55] Chakravarthy, S. et al. Composite events for active databases: Semantics, contexts and detection. In *Proc. of the 20th VLDB Conf.*, September 1994.

[56] C.-Y. Chan, P. Felber, M. Garofalakis, and R. Rastogi. Efficient filtering of XML documents with XPath expressions. *VLDB Journal*, 11(4), 2002.

[57] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Sam Madden, Vijayshankar Raman, Fred Reiss, and Mehul Shah. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *Proc. of the CIDR Conf.*, January 2003.

[58] Chang, J. and Kolobov, A. CSE 544 Class Project. `http://www.cs.washington.edu/544`, 2009.

[59] Sudarshan S. Chawathe, Venkat Krishnamurthy, Sridhar Ramachandran, , and Sanjay Sarma. Managing RFID data. In *Proc. of the 30th VLDB Conf.*, September 2004.

[60] J. Chen and D. S. Weld. Recovering from Errors During Programming by Demonstration. In *IUI 2008*, pages 159–168, 2008.

[61] Chen, G. and Kotz, D. Context Aggregation and Dissemination in Ubiquitous Computing Systems. In *Proc. of WMCSA 2002*, 2002.

[62] R Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proc. of the 2003 SIGMOD Conf.*, pages 551–562, 2003.

[63] Cherniack, M., et al. Scalable Distributed Stream Processing. In *Proc. of the CIDR Conf.*, January 2003.

[64] Cheverst, K. et. al. Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences. In *CHI 2000*, pages 17–24, 2000.

[65] Cheverst, K. et al. Exploring Issues of User Model Transparency and Proactive Behaviour in an Office Environment Control System. *User Modeling and User-Adapted Interaction*, 15(3-4):235–273, 2005.

[66] Christe, B. et al. Analysis of the Impact of a Radiofrequency Identification Asset-Tracking System in the Healthcare Setting. *Journal of Clinical Engineering*, pages 49–55, 2010.

[67] Clear, A. et al. Situvis: A Visual Tool for Modeling a User's Behaviour Patterns in a Pervasive Environment. In *Proc. of the 7th Pervasive Conf.*, pages 327–341, 2009.

[68] Computerworld. Procter & Gamble: Wal-Mart RFID effort effective. `http://www.computerworld.com/action/article.do? command=viewArticleBasic&articleId=284160`, February 2007.

[69] Consolvo, S. and Walker, M. Using the experience sampling method to evaluate ubicomp applications. *IEEE Pervasive Computing*, 2(2):24–31, 2003.

[70] Consolvo, S., et al. Activity Sensing in the Wild: A Field Trial of UbiFit Garden. In *Proc. of CHI 2008 Conf.*, 2008.

[71] Cormode, G. and Garofalakis, M. Sketching probabilistic data streams. In *Proc. of the 2007 SIGMOD Conf.*, June 2007.

[72] R. G. et al. Cowell. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[73] Chuck Cranor, Theodore Johnson, Oliver Spatscheck, and Vladislav Shkapenyuk. Gigascope: A stream database for network applications. In *Proc. of the 2003 SIGMOD Conf.*, June 2003.

[74] Dalvi, N. and Suciu, D. Efficient Query Evaluation on Probabilistic Databases. In *VLDB*, pages 864–875, 2004.

[75] Dalvi, N. and Suciu, D. Efficient query evaluation on probabilistic databases. In *Proc. of the 30th VLDB Conf.*, September 2004.

[76] Dalvi, N. et al. Query evaluation on probabilistic databases. *IEEE Data Engineering Bulletin*, 29(1):25–31, 2006.

[77] A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. In *Proc. of the 22nd ICDE Conf.*, April 2006.

[78] Debaty, P. and Caswell, D. Uniform Web Presence Architecture for People, Places, and Things. Technical Report HPL-2000-67, HP Labs, June 2000.

[79] Dekhtyar, A. et al. Probabilistic temporal databases, I: algebra. *ACM TODS*, 26(1):41–95, March 2001.

[80] Demers, A. et al. Towards Expressive Publish/Subscribe Systems. In *In Proc. EDBT*, pages 627–644, 2006.

[81] Demers, A. et al. Towards expressive publish/subscribe systems. In *Proc. of the 10th EDBT Conf.*, 2006.

[82] Demers, A.J. et al. Cayuga: A General Purpose Event Monitoring System. In *CIDR*, pages 412–422, 2007.

[83] Amol Deshpande, Carlos Guestrin, Wei Hong, and Samuel Madden. Exploiting correlated attributes in acquisitional query processing. In *Proc. of the 21st ICDE Conf.*, 2005.

[84] Amol Deshpande, Carlos Guestrin, Samuel Madden, Joseph Hellerstein, and Wei Hong. Model-based approximate querying in sensor networks. *VLDB Journal*, 14(4), 2005.

[85] Deshpande, A. and Madden, S. MauveDB: Supporting Model-based User Views in Database Systems. In *Proc. of the 2006 SIGMOD Conf.*, pages 73–84, 2006.

[86] Deshpande, A. et al. Using Probabilistic Models for Data Management in Acquisitional Environments. In *Proc. of the 2nd CIDR Conf.*, pages 317–328, January 2005.

[87] Dey, A. and Newberger, A. Support for Context-Aware Intelligibility and Control. In *CHI 2009*, pages 859–868, 2009.

[88] Dey, A. et al. Distributed mediation of ambiguous context in aware environments. In *Proc. of the UIST 2002 Conf*, pages 121–130, 2002.

[89] Dey A. K., et al. a CAPpella: Programming by Demonstration of Context-Aware Applications. In *CHI 2004*, volume 1, pages 33–40, 2004.

[90] Yanlei Diao, Mehmet Altinel, Michael J. Franklin, Hao Zhang, and Peter Fischer. Path sharing and predicate evaluation for high-performance XML filtering. *ACM TODS*, 28(4), 2003.

[91] Dinur, I. and Nissim K. Revealing information while preserving privacy. In *Proc. of the 22nd PODS Conf.*, pages 202–210, June 2003.

[92] Doucet, A. et al., editor. *Sequential Monte Carlo methods in practice*. Springer, 2001.

[93] The Ohio State University Medical Center Leverages Expansive Wi-Fi Network to Track and Manage Medical Equipment, Improve Workflow and Increase Staff and Patient Safety Using Ekahau RTLS.
`http://www.ekahau.com/news/readallnews/press-releases/`, January 2010.

[94] EPCGlobal application level events specification.
`http://www.epcglobalinc.org/standards/ale`, 2005.

[95] EPCglobal ALE.
`http://www.epcglobalinc.org/standards/ale`, 2009.

[96] EZ-Pass.
`http://www.ezpass.com/`, June 2009.

[97] Francoise Fabret, H. Arno Jacobsen, Francois Llirbat, Joao Pereira, Kenneth A. Ross, and Dennis Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *Proc. of the 2001 SIGMOD Conf.*, 2001.

[98] R. Fagin, P.G. Kolaitis, R.J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224. Springer, 2003.

[99] Fails, J. and Olsen, D. A Design Tool for Camera-Based Interaction. In *CHI 2003*, pages 449–456, 2003.

[100] Fails, J., et al. A Visual Interface for Multivariate Temporal Data: Finding Patterns of Events over Time. In *VAST 2006*, pages 167–174, 2006.

[101] Falk, J. et al. Pirates: proximity-triggered interaction in a multi-player game. In *Proc. of CHI 2001 Conf.*, volume 2, pages 119–120, 2001.

[102] M. Fernandez, Y. Kadiyska, A. Morishima, D. Suciu, and W. Tan. SilkRoute : a framework for publishing relational data in XML. *ACM Transactions on Database Technology*, 27(4), December 2002.

[103] Fishkin, K. et al. I Sense a Disturbance in the Force: Unobtrusive Detection of Interactions with RFID-tagged Objects. In *Ubicomp 2004*, pages 268–282, October 2004.

[104] C. Floerkemeier and F. Mattern. Smart Playing Cards - Enhancing the Gaming Experience with RFID. In *PerGames 2006*, pages 79–88, May 2006.

[105] Floerkemeier, C. and Lampe, M. Issues with RFID usage in ubiquitous computing applications. In *Proc. of the 2nd Pervasive Conf.*, April 2004.

[106] Floerkemeier, C. and Lampe, M. RFID middleware design - addressing application requirements and RFID constraints. In *sOc-EUSAI 05*, October 2005.

[107] Floerkemeier, C. et al. Facilitating RFID Development with the Accada Prototyping Platform. In *PerCom Workshops*, pages 495–500, 2007.

[108] Florida Court Ups Its RFID System.
http://www.rfidjournal.com/article/view/4914/1, May 2009.

[109] FourSquare. FourSquare.
http://foursquare.com, 2010.

[110] Fox, D. et al. Adapting the sample size in particle filters through kld sampling. *Int'l Journal of Robotics Research*, 22, 2003.

[111] Fox, D. et al. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, July-September 2003.

[112] Franklin, M. J. et. al. Design considerations for high fan-in systems: The hifi approach. In *Proc. of the 2nd CIDR Conf.*, January 2005.

[113] Froehlich, J., et al. MyExperience: A System for In Situ Tracing and Capturing of User Feedback on Mobile Phones. In *Proc. of the 5th MobiSys Conf.*, 2007.

[114] Norbert Fuhr and Thomas Roelleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[115] Garofalakis, M. N. et. al. Probabilistic Data Management for Pervasive Computing: The Data Furnace Project. *IEEE Data Eng. Bull*, 29(1):57–63, 2006.

[116] Gartner Research. Gartner Forecasts Worldwide Location-Based Services to Grow Nearly 170 Per Cent in 2008.
http://www.gartner.com/it/page.jsp?id=600011, February 2008.

[117] Gartner Research. Gartner Says Consumer Location-Based Services Market Will More Than Double in 2009.
http://www.gartner.com/it/page.jsp?id=1059812, 2009.

[118] Gartner Research. Gartner Says Context-Aware Computing Will be a $12 Billion Market by 2012.
http://www.gartner.com/it/page.jsp?id=1229413, November 2009.

[119] Gehani, N. H. et al. Composite event specification in active databases: Model & implementation. In *Proc. of the 18th VLDB Conf.*, August 1992.

[120] Johannes Gehrke and Samuel Madden. Query processing in sensor networks. *IEEE Pervasive*, 3(1), 2004.

[121] Gelman, A. et al. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2003.

[122] Gershenfeld, N. et al. The Internet of Things. *Scientific American*, October 2004.

[123] Glover, B. and Bhatt, H. *RFID Essentials*. O'Reilly, 2006.

[124] H. et al. Gonzalez. Mining Compressed Commodity Workflows from Massive RFID Data Sets. In *CIKM 06*, 2006.

[125] Gonzalez, H. et al. Warehousing and Analyzing Massive RFID Data Sets. In *Proc. of the 22nd ICDE Conf.*, April 2006.

[126] Google. Google Calendar APIs and Tools. `http://code.google.com/apis/calendar`, 2007.

[127] Google Maps API - Google Code. `http://code.google.com/apis/maps/`, 2009.

[128] Google Latitude. `http://www.google.com/latitude/`, February 2010.

[129] T. J. Green, A. Gupta, G. Miklau, M. Onizuka, and D. Suciu. Processing XML streams with deterministic automata and stream indexes. *ACM TODS*, 29(4):752–788, December 2004.

[130] Todd J. Green, Gerome Miklau, Makoto Onizuka, and Dan Suciu. Processing XML streams with deterministic automata. In *Proc. of the 9th ICDT Conf.*, 2002.

[131] Green, T. J. and Tannen, V. Models for incomplete and probabilistic information. *IEEE Data Engineering Bulletin*, 29(1):17–24, March 2006.

[132] Griffin, J.D. et al. RF Tag Antenna Performance on Various Materials Using Radio Link Budgets. *IEEE Antennas and Wireless Propagation Letters*, 5(1), December 2006.

[133] Grimm, R. et al. System support for pervasive applications. *ACM Trans. Comput. Syst*, 22(4):421–486, 2004.

[134] Griswold, W.G., et al. ActiveCampus: Experiments in Community-Oriented Ubiquitous Computing. *Computer*, 37(10):73–81, 2004.

[135] S. Guiasu and A. Shenitzer. The principle of maximum entropy. *The Mathematical Intelligencer*, 7(1), 1985.

[136] Ashish Gupta and Dan Suciu. Stream processing of XPath queries with predicates. In *Proceeding of ACM SIGMOD Conference on Management of Data*, 2003.

[137] Ashish Gupta, Dan Suciu, and Alon Halevy. The view selection problem for XML content based routing. In *Proceeding of PODS*, 2003.

[138] Gupta, M., et al. Adding GPS-Control to Traditional Thermostats: An Exploration of Potential Energy Savings and Design Challenges. In *Proc. of the 7th Pervasive Conf.*, pages 95–114, 2009.

[139] Ralf Hartmut Gŭting and Markus Schneider. *Moving Objects Databases*. Morgan Kaufmann Publishers, 2005.

[140] Google Web Toolkit - Google Code. `http://code.google.com/webtoolkit/`, 2009.

[141] Gyllstrom, D. et al. On Supporting Kleene Closure over Event Streams. In *ICDE*, pages 1391–1393, 2008.

[142] Hahnel, D et al. Mapping and Localization with RFID Technology. In *ICRA*, pages 1015–1020, 2004.

[143] Eric N. Hanson. Rule condition testing and action execution in Ariel. In *Proc. of the 1992 SIGMOD Conf.*, June 1992.

[144] Eric N. Hanson, Chris Carnes, Lan Huang, Mohan Konyala, Lloyd Noronha, Sashi P arthasarathy, J. B. Park, and Albert Vernon. Scalable trigger processing. In *Proc. of the 15th ICDE Conf.*, March 1999.

[145] Harper, R. *Inside the Smart Home*. Springer, 2006.

[146] Harter, A. et. al. The Anatomy of a Context-Aware Application. In *Mobicom 1999*, pages 59–68, 1999.

[147] Hartmann, B. et al. Authoring Sensor-Based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. In *CHI 2007*, pages 145–154, 2007.

[148] Heer, J. et al. liquid: Context-Aware Distributed Queries. In *Ubicomp 2003*, volume 2864, pages 140–148, 2003.

[149] Hightower, J. and Borriello, G. Location Systems for Ubiquitous Computing. *IEEE Computer*, 34(8):57–66, 2001.

[150] Hightower, J. et al. Learning and Recognizing the Places We Go. In *Ubicomp 2005*, volume 3660, pages 159–176, 2005.

[151] Hile, H., et al. Landmark-Based Pedestrian Navigation with Enhanced Spatial Reasoning. In *Proc. of the 7th Pervasive Conf.*, pages 56–76, 2009.

[152] Hill, R. and Begole, J. Activity rhythm detection and modeling. In *CHI 2003*, pages 782–783, 2003.

[153] S. Hinske and M. Langheinrich. *An RFID-based Infrastructure for Automatically Determining the Position and Orientation of Game Objects in Tabletop Games*, volume 1, pages 311–336. Shaker Verlag, December 2007.

[154] Hochheiser, H., et al. Dynamic Query Tools for Time Series Data Sets, Timebox Widgets for Interactive Exploration. In *Info Viz 2004*, 2004.

[155] Hodges, S. et al. Assessing and Optimizing the Range of UHF RFID to Enable Real-World Pervasive Computing Applications. In *Proc. of the 5th Pervasive Conf.*, pages 280–297, 2007.

[156] Hong, J. and Landay, J. A. An Architecture for Privacy-Sensitive Ubiquitous Computing. In *Proc. of the 2nd MobiSys Conf.*, 2004.

[157] S. Hsi and H. Fait. RFID enhances visitors' museum experience at the Exploratorium. *Commun. ACM*, 48(9):60–65, 2005.

[158] Hsieh, G., et al. Field deployment of IMBuddy: a Study of Privacy Control and Feedback Mechanisms for Contextual IM. In *Proc. of the 9th Ubicomp Conf.*, pages 91–108, 2007.

[159] Hu, Y. et al. Supporting RFID-based item tracking applications in Oracle DBMS using a bitmap datatype. In *Proc. of the 31st VLDB Conf.*, September 2005.

[160] Bret Hull, Vladimir Bychkovsky, Kevin Chen, Michel Goraczko, Eugene Shih, Yang Zhang, Hari Balakrishnan, and Samuel Madden. CarTel: A distributed mobile sensor computing system. In *Proc of the 4th SenSys Conf.*, 2006.

[161] Jeong-Hyon Hwang, Magdalena Balazinska, Alexander Rasin, Uğur Çetintemel, Michael Stonebraker, and Stan Zdonik. High-availability algorithms for distributed stream processing. In *Proc. of the 21st ICDE Conf.*, April 2005.

[162] IDTechEx. Complete RFID analysis and forecasts 2008-2018. `http://www.idtechex.com/research/reports/`, 2008.

[163] IDTechEx. Active RFID and Sensor Networks 2009-2019. `http://www.idtechex.com/research/reports/`, 2009.

[164] IDTechEx. Real Time Locating Systems 2009-2019.
http://www.idtechex.com/research/reports/, 2009.

[165] IDTechEx. Real-Time Locating Systems 2009-2019.
http://www.idtechex.com/research/reports/, 2009.

[166] IDTechEx. RFID Forecasts, Players and Opportunities 2009-2019.
http://www.idtechex.com/research/reports/, 2009.

[167] Impinj. RFID Communication and Interference.
http://www.impinj.com/WorkArea/, 2007.

[168] Speedway Product Brief.
http://www.impinj.com/files/ MR_SP_PB_00001_SpeedwayProductBrief.pdf.

[169] Intille, S. S. et. al. A living laboratory for the design and evaluation of ubiquitous computing interfaces. In *CHI 2005 extended abstracts*, April 2005.

[170] iPhone. Apple – iPhone – Mobile phone, iPod, and Internet device.
http://www.apple.com/iphone/, 2010.

[171] H. V. Jagadish, Inderpal Singh Mumick, and Abraham Silberschatz. View maintenance issues for the chronicle data model. In *Proc. of the 14th PODS Conf.*, May 1995.

[172] Jayram, T. S. et al. Efficient aggregation algorithms for probabilistic data. In *ACM-SIAM Symposium on Discrete Algorithms*, January 2007.

[173] Jayram, T. S. et al. Estimating statistical aggregates on probabilistic data streams. In *Proc. of the 26th PODS Conf.*, June 2007.

[174] JDBC.
http://java.sun.com/javase/technologies/database/.

[175] Jeffery, S. et. al. Declarative support for sensor data cleaning. In *Proc. of the 4th Pervasive Conf.*, March 2006.

[176] Jeffery, S. R. et al. Adaptive Cleaning for RFID Data Streams. In *Proc. of the 32nd VLDB Conf.*, pages 163–174, September 2006.

[177] Jin, Y. and Strom. Relational subscription middleware for Internet-scale publish-subscribe. In *Proc of 2nd DEBS Workshop*, June 2003.

[178] Johanson, B. and Fox, A. The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In *WMCSA*, pages 83–93, 2002.

[179] Juels, A. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications*, 24:381–394, February 2006.

[180] Kanagal, B. and Deshpande, A. Online filtering, smoothing and probabilistic modeling of streaming data. Technical Report CS-TR-4867, University of Maryland, May 2007.

[181] Kanagal, B. and Deshpande, A. Efficient Query Evaluation over Temporally Correlated Probabilistic Streams. In *ICDE*, pages 1315–1318, 2009.

[182] Kanagal, B. and Deshpande, A. Indexing Correlated Probabilistic Databases. In *SIGMOD 09*, 2009.

[183] Richard Karp and Michael Luby. Monte-carlo algorithms for enumeration and reliability problems. In *Proceedings of the annual ACM symposium on Theory of computing*, 1983.

[184] Khoussainova, N. et al. Towards correcting input data errors probabilistically using integrity constraints. In *Proc. of Fifth MobiDE Workshop*, June 2006.

[185] Khoussainova, N. et. al. Probabilistic RFID Data Management. Technical Report UW-CSE-07-03-01, University of Washington, CSE, March 2007.

[186] Khoussainova, N. et al. Probabilistic Event Extraction from RFID Data (poster). In *Proc. of the 24th ICDE Conf.*, April 2008.

[187] Kittredge, L. and Borriello, G. An RFID-based Game to Encourage Social Interaction. *PerGames 2006*, May 2006.

[188] Knoll, S. et al. Viewing Personal Data Over Time. In *CHI 2009 Workshop on Interacting with Temporal Data*, April 2009.

[189] Ko, A. J. and Myers, B. A. Designing the Whyline: a Debugging Interface for Asking Questions About Program Behavior. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 151–158, 2004.

[190] Koch, C. MayBMS: A system for managing large uncertain and probabilistic databases. In *Managing and Mining Uncertain Data*. Springer-Verlag, 2009.

[191] Konomi, S. and Roussos, G. Ubiquitous computing in the real world: Lessons learnt from large scale RFID deployments. *Personal and Ubiquitous Computing*, 12(2), September 2003.

[192] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. *Wirel. Netw.*, 11(1-2):115–133, 2005.

[193] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. *Wirel. Netw.*, 11(1-2):115–133, 2005.

[194] Kriplean, T. et. al. Physical access control for captured RFID data. *IEEE Pervasive Computing*, 6(4), November 2007.

[195] Krumm, J. and Horvitz, E. Predestination: Inferring destinations from partial trajectories. In *Proc. of the 8th Ubicomp Conf.*, pages 243–260, October 2006.

[196] KSW Microtec.
`http://www.ksw-microtec.de/`.

[197] Krishna G. Kulkarni, Nelson Mendonça Mattos, and Roberta Cochrane. Active database features in SQL-3. In Norman W. Paton, editor, *Active Rules in Database Systems*. Springer-Verlag, 1999.

[198] Kwok C. et. al. Real-time Particle Filters. *Proc. of the IEEE*, 2(92):469–484, 2004.

[199] Lafferty, J. D. et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[200] Lahiri, S. *RFID Sourcebook*. IBM Press, 2006.

[201] L. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. Probview: A flexible probabilistic database system. *ACM Trans. Database Syst.*, 22(3), 1997.

[202] A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello. PlantCare: An investigation in practical ubiquitous systems. In *Proc. of the 4th Ubicomp Conf.*, September 2002.

[203] Lamming, M. and Bohm, D. SPECs: Another Approach to Human Context and Activity Sensing Research, Using Tiny Peer-to-Peer Wireless Computers. In *Ubicomp 2003*, pages 192–199, 2003.

[204] Lampe, M. and Floerkemeier, C. The Smart Box Application Model. In *Advances in Pervasive Computing, Austrian Computer Society (OCG)*, April 2004.

[205] Landwehr, N. et al. Relational Transformation-Based Tagging for Human Activity Recognition. In *MRDM 07*, September 2007.

[206] Lau, T. et al. Why PBD Systems Fail: Lessons Learned for Usable AI. In *CHI 2008*, 2008.

[207] Alberto Lerner and Dennis Shasha. AQuery: query language for ordered data, optimization techniques, and experiments. In *Proc. of the 17th VLDB Conf.*, September 2003.

[208] Leshed, G. et al. CoScripter: Automating & Sharing How-To Knowledge in the Enterprise. In *CHI 2008*, pages 1719–1728, 2008.

[209] Lester, J. et al. A Practical Approach to Recognizing Physical Activities. In *Pervasive 2004*, volume 3968, pages 1–16, 2006.

[210] Lester, J., et al. Validated Caloric Expenditure Estimation Using a Single Body-Worn Sensor. In *Ubicomp 2009*, pages 225–234, 2009.

[211] Letchner, J. et al. Challenges for Event Queries over Markovian Streams. *IEEE Internet Computing*, 12(6):30–36, 2008.

[212] Letchner, J. et al. Access Methods for Markovian Streams. In *Proc. of the 25th ICDE Conf.*, April 2009.

[213] Letchner, J. et al. Approximation Trade-Offs in Markovian Stream Processing: An Empirical Study. In *Proc. of the 26th ICDE Conf.*, April 2010.

[214] Li, I. et al. Using Contextual Information to Improve Awareness of Physical Activity. In *Engaging Data 2009*, 2009.

[215] Li, Y. et al. Topiary: A Tool for Prototyping Location-Enhanced Applications. In *UIST 2004*, pages 217–226, 2004.

[216] Lifton, J. at al. Tricorder: A mobile sensor network browser. In *CHI 2007*, 2007.

[217] Lim, B. and Dey, A. Assessing Demand for Intelligibility in Context-Aware Applications. In *Ubicomp 09*, pages 195–204, 2009.

[218] A. Liu, H. Hile, H. Kautz, G. Borriello, P. Brown, M. Harniss, and K. Johnson. Indoor wayfinding: Developing a functional interface for individuals with cognitive impairments. In *8th International ACM SIGACCESS Conferene on Computers and Accessibility*, October 2006.

[219] Jie Liu and Feng Zhao. Towards semantic services for sensor-rich information systems. In *Proc. of the Basenets Workshop*, October 2005.

[220] Liu, X. et. al. Ferret: RFID Localization for Pervasive Multimedia. In *Proc. of the 8th Ubicomp Conf.*, pages 422–440, September 2006.

[221] Logan, B. et al. A Long-Term Evaluation of Sensing Modalities for Activity Recognition. In *Ubicomp 2007*, pages 483–500, September 2007.

[222] Lymberopoulos, D., et al. A Sensory Grammar for Inferring Behaviors in Sensor Networks. In *IPSN 2006*, pages 251–259, 2006.

[223] MacIntyre, B. et al. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In *SIGGRAPH 2005*, pages 932–932, 2005.

[224] Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong. TinyDB: An acquisitional query processing system for sensor networks. *ACM TODS*, 30(1), 2005.

[225] V. Markl, N. Megiddo, M. Kutsch, T.M. Tran, P. Haas, and U. Srivastava. Consistently estimating the selectivity of conjuncts of predicates. In *VLDB*, pages 373–384, 2005.

[226] MasterCard PayPass. `http://www.paypass.com/`, June 2009.

[227] The MathWorks. Matlab. `http://www.mathworks.com/products/matlab/`, 2006.

[228] Maynes-Aminzade, D. et al. Eyepatch: Prototyping Camera-Based Interaction Through Examples. In *UIST 2007*, pages 33–42, 2007.

[229] McCarthy, J. F. and Anagnost, T. D. EVENTMANAGER: Support for the Peripheral Awareness of Events. In *HUC*, volume 1927, pages 227–235, 2000.

[230] McCullagh, D. Perspective: RFID tags: Big brother in small packages. C—NET News.com., January 2003.

[231] McGinity, M. RFID: Is this game of tag fair play? *Communications of the ACM*, 47(1), January 2004.

[232] Mei, Y. and Madden, S. ZStream: A cost-based query processor for adaptively detecting composite events. In *SIGMOD '09*, 2009.

[233] Jim Melton. *Advanced SQL:1999 – Understanding Object-Relational and Other Advanced Features.* Morgan Kaufmann Publishers, September 2002.

[234] Microsoft Corporation. Microsoft SQL server. `http://www.microsoft.com/sql/default.mspx`, 2007.

[235] Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Manku, Chris Olston, Justin Rosenstein, and Rohit Varma. Query processing, approximation, and resource management in a data stream management system. In *Proc. of the CIDR Conf.*, January 2003.

[236] Razvan Musaloiu, Andreas Terzis, Katalin Szlavecz, Alex Szalay, Joshua Cogan, and Jim Gray. Life under your feet: A wireless soil ecology sensor network. In *EmNets 2006*, 2006.

[237] B. A. Nardi. *A Small Matter of Programming: Perspectives on End User Computing.* MIT Press, 1993.

[238] Nath, S. et al. IrisNet: An Architecture for Internet-scale Sensing Services. In *VLDB*, pages 1137–1140, 2003.

[239] Nemmaluri, A., et al. Sherlock: Automatically Locating Objects for Humans. In *MobiSys 2008*, pages 187–198, 2008.

[240] Ni, L. M. et al. LANDMARC: Indoor Location Sensing using Active RFID. *Wirel. Netw.*, 10(6):701–710, 2004.

[241] Nie, Y. et al. Probabilistic Inference over RFID Streams in Mobile Environments. In *Proc. of the 25th ICDE Conf.*, pages 1096–1107, April 2009.

[242] ntp.org: Home of the Network Time Protocol. `http://www.ntp.org`.

[243] Olston, C., et al. Generating Example Data for Dataflow Programs. In *SIGMOD 2009*, pages 245–256, 2009.

[244] Oracle sensor edge server. `http://www.oracle.com/technology/products/`.

[245] New Oregon Hospital Adopts IR-RFID Hybrid System. `http://www.rfidjournal.com/article/view/4846/1`, May 2009.

[246] Orr, R.J. and Abowed, G.D. The Smart Floor: A Mechanism for Natural User Identification and Tracking. In *CHI 2000*, 2000.

[247] Outalout. Outalot. `http://outalot.com`, 2010.

[248] Gultekin Ozsoyoglu and Richard T. Snodgrass. Temporal and real-time databases: A survey. *IEEE TKDE*, 7(4):513–532, 1995.

[249] Pane, J.F. and Myers, B. A. Tabular and Textual Methods for Selecting Objects from a Group. In *VL 00*, page 157, 2000.

[250] Pascoe, J. The Stick-e Note Architecture: Extending the Interface Beyond the User. In *IUI 1997*, pages 261–264, 1997.

[251] N. W. Paton and Oscar Díaz. Active database systems. *ACM Comput. Surv.*, 31(1):63–103, 1999.

[252] Patterson, D. J. et al. Inferring high-level behavior from low-level sensors. In *Proc. of the 5th Ubicomp Conf.*, October 2003.

[253] Petney, W. et al. Learning Large Scale Common Sense Models of Everyday Life. In *AAAI 07*, July 2007.

[254] Philipose, M. et al. Inferring Activities from Interactions with Objects. *IEEE Pervasive Computing*, 3(4):50–57, 2004.

[255] Philly Hospital Uses RTLS to Track Patient Flow, Care and Training. `http://www.rfidjournal.com/article/view/4934/1`, May 2009.

[256] Pietzuch, P. R. and Bacon, J. Hermes: A Distributed Event-Based Middleware Architecture. In *ICDCS Workshops*, pages 611–618. IEEE Computer Society, 2002.

[257] Prabhu, B. S. et. al. *WinRFID - A Middleware for the enablement of Radio Frequency Identification (RFID) based Applications, In Mobile, Wireless and Sensor Networks: Technology, Applications, and Future.* Wiley 2005, 2005.

[258] Priyantha, N. B., et al. The Cricket Location-Support System. In *Proc. of the 6th MOBICOM Conf.*, August 2000.

[259] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[260] Rabiner, L. R. A tutorial on hidden Markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296, 1990.

[261] RadarFind: A TeleTracking Technologies Company. `http://www.radarfind.com/`, March 2010.

[262] Rahmati, A. et al. Reliability Techniques for RFID-Based Object Tracking Applications. In *DSN 2007*, pages 113–118, 2007.

[263] Ramakrishnan, K. M. and Deavours, D. D. Performance Benchmarks for Passive UHF RFID Tags. In *Proc. of the 13th GI/ITG Conf. on Measurement, Modeling, and Evaluation of Computer and Communication Systems*, pages 137–154, March 2006.

[264] Rao, J. et al. A deferred cleansing method for RFID data analytics. In *Proc. of the 32nd VLDB Conf.*, September 2006.

[265] Rao, K.V.S. et al. Antenna Design for UHF RFID Tags: a Review and a Practical Application. *IEEE Trans. on Antennas and Propagation*, 53(12), December 2005.

[266] Rapid Edge Development & Deployment - Event Workflow Editor (EWE). `http://www.globeranger.com/rapidedgedeployment/ewe.asp`, 2009.

[267] Rastogi, V. et. al. Authorization Views for Pervasive Sensor Networks. In *UbiPriv 2007*, September 2007.

[268] Ravindranath, L. et al. SixthSense: RFID-based Enterprise Intelligence. In *MobiSys 2008*, pages 253–266, 2008.

[269] C. Re, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *Proc. of the 23rd ICDE Conf.*, 2007. to appear.

[270] C. Ré and D. Suciu. Management of data with uncertainties. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 3–8, 2007.

[271] C. Ré and D. Suciu. Managing Probabilistic Data with MystiQ: The Can-Do, the Could-Do, and the Can't-Do. In *SUM '08: Proceedings of the 2nd international conference on Scalable Uncertainty Management*, pages 5–18, 2008.

[272] C. et al. Ré. Query Evaluation on Probabilistic Databases. In *IEEE Data Engineering Bulletin*, volume 29, pages 25–31, 2006.

[273] Ré, C. et al. Event Queries on Correlated Probabilistic Streams. In *SIGMOD 2008*, pages 715–728, June 2008.

[274] RTLS Providers Cite Strong Demand From Hospitals. `http://www.rfidjournal.com/article/print/4981`, June 2009.

[275] RFID Update. Reva Launches RTLS Solution Based on Passive RFID. `http://www.rfidupdate.com/articles/index.php?id=1774`, April 2009.

[276] The RFID Ecosystem. `http://rfid.cs.washington.edu/`, 2006.

[277] RFIDentics Products.
http://www.rfidentics.com/products.html.

[278] RFID journal.
http://www.rfidjournal.com, 2006.

[279] Rieback, M. R. et. al. Is your cat infected with a computer virus? In *Proc. of the PerCom Conf.*, March 2006.

[280] Rizvi et. al. Events on the edge. In *Proc. of the 2005 SIGMOD Conf.*, June 2005. (System demonstration).

[281] Roman, M. et al. Gaia: a middleware platform for active spaces. *Mobile Computing and Communications Review*, 6(4):65–67, 2002.

[282] Russell, S. and Norvig, P. *Artificial Intelligence A Modern Approach*. Prentice Hall, 2003.

[283] Esther Ryvkina, Anurag Maskey, Mitch Cherniack, and Stan Zdonik. Revision processing in a stream processing engine: A high-level design. In *Proc. of the 22nd ICDE Conf.*, April 2006.

[284] Reza Sadri, Carlo Zaniolo, Amir M. Zarkesh, and Jafar Adibi. Optimization of sequence queries in database systems. In *Proc. of the 20th PODS Conf.*, 2001.

[285] Sadri, R. et al. Expressing and Optimizing Sequence Queries in Database Systems. *ACM Trans. Database Syst.*, 29(2):282–318, 2004.

[286] Salber, D. et al. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *CHI 1999*, pages 434–441, 1999.

[287] A. D. et al. Sarma. Exploiting Lineage for Confidence Computation in Uncertain and Probabilistic Databases. In *ICDE*, pages 1023–1032, 2008.

[288] Schilit, B. et al. Context-Aware Computing Applications. In *WMCSA 94*, pages 85–90, 1994.

[289] Schilit, B. et. al. Challenge: Ubiquitous Location-Aware Computing and the Place Lab Initiative. In *WMASH 2003*, September 2003.

[290] Ulf Schreier, Hamid Pirahesh, and Rakesh Agrawal an C. Mohan. Alert: An architecture for transforming a passive DBMS into an active DBMS. In *Proc. of the 17th VLDB Conf.*, September 1991.

[291] scvngr.
`http://www.scvngr.com/`, February 2010.

[292] Sen, P. and Deshpande, A. Representing and Querying Correlated Tuples in Probabilistic Databases. In *ICDE*, pages 596–605, 2007.

[293] Sen, P. et al. Exploiting shared correlations in probabilistic databases. *PVLDB*, 1(1):809–820, 2008.

[294] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. Sequence query processing. In *Proc. of the 1994 SIGMOD Conf.*, 1994.

[295] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. The design and implementation of a sequence database system. In *Proc. of the 22nd VLDB Conf.*, September 1996.

[296] Mehul Shah, Joseph Hellerstein, and Eric Brewer. Highly-available, fault-tolerant, parallel dataflows. In *Proc. of the 2004 SIGMOD Conf.*, June 2004.

[297] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 1948.

[298] SIMILE Timeline.
`https://simile.mit.edu/timeline/`, 2009.

[299] Smith, I., et al. Social Disclosure of Place: From Location Technology to Communication Practices. In *Proc. of the 3rd Pervasive Conf.*, pages 134–151, 2005.

[300] Smith, j. et. al. A Wirelessly Powered Platform for Sensing and Computation. In *Ubicomp 2006*, September 2007.

[301] Smith, J. R. et al. RFID-Based Techniques for Human-Activity Detection. *Communications of the ACM*, 48(9):39–44, September 2005.

[302] Richard Snodgrass and Ilsoo Ahn. A taxonomy of time in databases. In *Proc. of the 1985 SIGMOD Conf.*, May 1985.

[303] Richard T. Snodgrass, editor. *The TSQL2 Temporal Query Language*. Kluwer, 1995.

[304] Sohn, T. and Dey, A. iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications. In *CHI 2003*, pages 974–975, 2003.

[305] Sohn, T., et al. Place-Its: A Study of Location-Based Reminders on Mobile Phones. In *Proc. of the 7th Ubicomp Conf.*, pages 232–250, 2005.

[306] Songini, M. L. Wal-Mart details its RFID journey. ComputerWorld, March 2006.

[307] Sousa, J. P. and Garlan, D. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *WICSA*, volume 224, pages 29–43, 2002.

[308] Speedpass.
http://www.speedpass.com/, June 2009.

[309] Spyglass Consulting. Trends in RFID 2008.
http://www.spyglass-consulting.com/ spyglass_whitepaper_RFID.html, July 2008.

[310] SQL Server 2008 Overview.
http://www.microsoft.com/SQL.

[311] Stanford, V. Pervasive computing goes the last hundred feet with RFID systems. *IEEE Pervasive Computing*, 2(2), April 2003.

[312] Michael Stonebraker and Greg Kemnitz. The POSTGRES next generation database management system. *Communications of the ACM*, 34(10):78–92, 1991.

[313] Strategy Analytics. Automotive and Portable Navigation Market FORECAST UPDATE 2007-2015.
http://www.strategyanalytics.com/, June 2009.

[314] Robert E. Strom. Fault-tolerance in the SMILE stateful publish-subscribe system. In *Proc of 3rd DEBS Workshop*, May 2004.

[315] Tekrati. Portable navigation market on track for connectivity and new leaders, says TRG.
http://ce.tekrati.com/research/9811/, January 2008.

[316] The Consumerist. German department store launches RFID-enhanced men's department.
http://consumerist.com/consumer/the-store-of-tomorrow/german-department-store-launches-rfid+enhanc
September 2007.

[317] The PostgreSQL Global Development Group. Postgresql database management system. http://www.postgresql.org, 2006.

[318] The U.S. Electronic Passport.
http://travel.state.gov/passport/eppt/eppt_2498.html, June 2009.

[319] WA State Licensing: Projects and Priorities - Enhanced Driver License.
http://www.dol.wa.gov/about/news/priorities/edl.html, June 2009.

[320] TriOut. TriOut.
http://trioutnc.com, 2010.

[321] Truong, K. et al. INCA: Architectural Support for Building Automated Capture & Access Applications. In *UIST 02*, 2002.

[322] Truong, K. N. et al. CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home. In *Ubicomp 2004*, October 2004.

[323] University of Washington. RFID Ecosystem.
http://rfid.cs.washington.edu/.

[324] University Research Concludes Real Time Location Systems Can Transform Hospitals.
http://www.radarfind.com/pages/69/, November 2009.

[325] van der Togt, R. et al. Electromagnetic Interference From Radio Frequency Identification Inducing Potentially Hazardous Incidents in Critical Care Medical Equipment. *Journal of the American Medical Association*, 299(24), June 2008.

[326] Van Oranje, C. et al. Policy options for Radio Frequency Identification (RFID) in healthcare; a prospective view. Technical report, RAND Corporation, August 2009.

[327] VersusTech. Versus Technology.
http://www.versustech.com/, 2010.

[328] Vilamoska, A., et al. Study on the requirements and options for RFID application in healthcare. Technical Report TR-608-EC, RAND EUROPE, July 2008.

[329] Vilamovska, A. et al. Study on the requirements and options for RFID application in healthcare. Technical report, RAND Corporation, July 2009.

[330] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, R. Stoleru, S. Lin, and J.A. Stankovic. An assisted living oriented information system based on a residential wireless sensor network. In *In Proc. of the 1st Distributed Diagnosis and Home Healthcare (D2H2) Conference.*, 2006.

[331] Visonic. Visonic.
http://visonic.com, 2010.

[332] Volgyesi, P., et al. Shooter Localization and Weapon Classification with Soldier-Wearable Networked Sensors. In *Proc. of the 5th MobiSys Conf.*, pages 113–126, 2007.

[333] VouChaCha. VouChaCha: Great deals on your mobile phone. `http://vouchacha.com`, 2010.

[334] Wang, D. Z. et al. BayesStore: managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1):340–351, 2008.

[335] Wang, D.Z. et al. Declarative information extraction in a probabilistic database system. In *Not yet published.*, 2009.

[336] Wang, F. and Liu, P. Temporal Management of RFID Data. In *Proc. of the 31st VLDB Conf.*, pages 1128–1139, September 2005.

[337] Want, R. The magic of RFID. *ACM Queue*, 2(7), October 2004.

[338] Want, R. RFID: A key to automaing everything. *Scientific American*, January 2004.

[339] Want, R. et al. The Active Badge Location System. *ACM TOIS*, 10(1), 1992.

[340] Want, R. et. al. An Overview of the PARCTAB Ubiquitous Computing Experiment. *IEEE Personal Communications*, 2(6):28–33, Dec 1995.

[341] Want, R. et. al. Bridging Physical and Virtual Worlds with Electronic Tags. In *CHI*, pages 370–377, 1999.

[342] Weiser, M. The Computer for the 21st Century. *SIGMOBILE MC2R*, 3(3):3–11, 1999.

[343] Welbourne, E. et. al. Challenges for Pervasive RFID-based Infrastructures. In *PERTEC 2007*, pages 388–394, March 2007.

[344] Welbourne, E. et. al. Cascadia: A System for Specifying, Detecting, and Managing RFID Events. In *MobiSys 2008*, pages 281–294, June 2008.

[345] Welbourne, E. et. al. Building the Internet of Things Using RFID: The RFID Ecosystem Experience. *IEEE Internet Computing*, May 2009.

[346] Welbourne, E. et. al. Longitudinal Study of a Building-Scale RFID Ecosystem. In *MobiSys 2009*, June 2009.

[347] Whitehouse, K., et al. Automatic Programming with Semantic Streams. In *SenSys 2005*, pages 290–291, November 2005.

[348] J. Widom. Trio: A System for Data, Uncertainty, and Lineage. In *Managing and Mining Uncertain Data*. Springer, 2008.

[349] Jennifer Widom and Sheldon J. Finkelstein. Set-oriented production rules in relational database systems. In *Proc. of the 1990 SIGMOD Conf.*, June 1990.

[350] Widom, J. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. of the 2nd CIDR Conf.*, pages 262–276, January 2005.

[351] E. et al. Wu. High-performance complex event processing over streams. In *SIGMOD 06*, pages 407–418, 2006.

[352] Wu, E. et al. High-performance complex event processing over streams. In *Proc. of the 2006 SIGMOD Conf.*, June 2006.

[353] Yong Yao and Johannes Gehrke. Query processing in sensor networks. In *Proc. of the CIDR Conf.*, 2003.

[354] Yelp. Yelp for Mobile. `http://www.yelp.com/yelpmobile`, 2010.