# Deriving Long-Term Value from Context-Aware Computing

**Guruduth Banavar[1], Jay Black[2], Ramón Cáceres[1], Maria Ebling[1], Edie Stern[1], Joseph Kannry[3]**

[1]{banavar, caceres, ebling, estern}@us.ibm.com
IBM T. J. Watson Research Center
Hawthorne, NY, USA

[2]jpblack@uwaterloo.ca
Associate Professor, School of Computer Science,
University of Waterloo
Waterloo, Ontario, Canada[**]

[3]joseph.kannry@mountsinai.org
Chief, Division of Clinical Informatics
Mount Sinai Medical Center
New York, NY, USA

*Modern businesses are increasingly dynamic in nature, which creates a need for computer systems that can sense and respond to rapid changes in the environment, or "context," of the enterprise. Custom solutions have emerged to address some of these issues, but the time has come for more systematic approaches. In this paper, we present our vision of a context "ecosystem" that helps enterprises, applications, and developers respond to these dynamic changes and derive long-term value from context information. The ecosystem includes providers of raw context information, components that derive more abstract context information from lower-level sources, middleware that provides systematic context services to applications, development tools, and context-aware applications.*

Modern businesses are dynamic in nature. To stay competitive, they need to optimize their business processes by understanding and reacting to the rapid changes in their environment. For example, changes in market demand, inventory levels, transportation constraints, and scheduling should ideally influence a supply chain. Similarly, current workload, availability, proximity, and expected travel time should affect insurance claim handling. These and other examples in domains as diverse as financial services, distribution, fleet management, and healthcare suggest that business processes could be

[**] This work was done while the author was on sabbatical at the IBM TJ Watson Research Center.

optimized and enhanced with sufficient information about the current and projected states of the participating entities, combined with the ability to respond to those changes. This ability to adapt business processes to changes in the environmental context is of growing interest to enterprises.

Custom solutions address some of these problems, but at a high cost. For example, companies have deployed fleet management applications that track mobile resources and optimize scheduling. The cost of building custom solutions that exploit highly dynamic data is prohibitive, and assets developed for one solution cannot easily be reused or extended for others—even within the same domain. In addition, when the data sources change, such as when an enterprise subscribes to a different cellular provider for location information, the application must often change as a consequence.

Context-aware computing has been attempting to solve these kinds of problems in a more general and systematic manner for some time. However, most of the attempts have been small-scale solutions within the confines of a room or a building, where the data sources, points of data aggregation, and applications are small in number, limited in variety, and under the control of a single organization. Although this is a necessary first step, our experience suggests that the real value of context-aware computing can only be realized when these solutions scale to support a large and diverse set of sources and aggregation points that can be built and managed by multiple stakeholders, but that remain open and reusable by third parties. For example, the supply chain example mentioned above requires context data from retailers, warehouses, transportation companies, and manufacturers, routed through different service providers, each of which can be owned and operated by a different business. Furthermore, this data can be aggregated to provide useful services by third parties, such as “current demand for electronic toys” in our supply-chain example. We envision that an open market will eventually emerge, comprised of many independent application developers and data providers. This will enable a variety of new business models. We refer to this kind of large-scale, multi-party, open, extensible system as a *context ecosystem*.

Context-aware computing has the potential to become broadly established in a variety of domains. A context ecosystem is not specific to any particular domain, and so context sources common to many domains (e.g., location sources) can be shared and reused. The ecosystem will cover data adaptation from external context sources, data analysis and composition into higher abstractions, and applications taking context-influenced actions, with the system respecting the security of sensitive data and the privacy rights of individuals. With such an ecosystem in place, enterprises stand to derive value over the long term because applications become easier to build, and evolve gracefully as new components are added and as technology improves.

It is important to note, however, that even before context-aware computing develops to this level, enterprises stand to derive value from the context available within their own domains. Just as intranets provided business value before the Internet matured, so too can intra-enterprise context provide business value before the full context ecosystem emerges. In fact, this intra-enterprise use represents the first step toward realizing a context ecosystem.

The context ecosystem described above will need to be supported by a comprehensive technological infrastructure. This paper presents an overview of this infrastructure needed for a context ecosystem. We describe a prototype of a comprehensive software infrastructure to build context-aware solutions in an extensible, reusable, and scalable way. This infrastructure facilitates collecting, processing, and analyzing environmental context data, and ultimately enables dynamic changes to the real world in response. We argue that the technological infrastructure presented here is capable of supporting the requirements of a context ecosystem.

The next section presents some healthcare scenarios, used as running examples in the rest of the paper. They motivate the requirements for a context ecology discussed in the following section. The requirements, in turn, are used as background for the section that surveys previous work in context-aware computing, leading to a presentation of our architecture and prototype. Finally, we describe our experience with the prototype and present our conclusions and areas for further work.

## <A>Scenarios

We present scenarios in the healthcare domain in which we compare and contrast the activities as they occur today and as they might occur in the future within a context ecosystem. The present day presumes a paper-based hospital with clinical information only available on a limited number of workstations, and patient status monitored by lights and sounds at the front desk, exploiting the patient's proximity to the nursing station. The future state demonstrates the use of an intra-hospital context ecosystem that improves patient safety and increases operational efficiency. Patients are instrumented with vital-sign monitors and positioning devices (e.g., active RFID tags). Physicians and nurses have wireless personal digital assistants (PDAs), also instrumented with positioning technology (e.g., triangulation on 802.11 base stations). Context applications optimize physician rounds, support nurse triage, simplify the user interface to pervasive devices, provide additional data for billing reconciliation, support compliance with government requirements, and provide clinical communications (the right information, at the right time, in the right way, all in accordance with legislation such as the U. S. Health Insurance Portability and Accountability Act (HIPAA, 2005)). As technology, particularly pervasive and sensor technology, undergoes rapid evolution, middleware can play a crucial role in providing a stable base for applications and bringing cohesion to this type of complex environment.

### *<B>Example 1*

**Present State:** Staff physician Dr. Ready arrives on the second floor of the hospital. He looks at a sheet of paper with patient location and some clinical information. This sheet of paper, called a SignOut, is generated by hand or printed from the SignOut system (Kannry and Moore, 1999; Kushniruk et al., 2003). He makes a quick decision whether any of these patients need to be seen urgently and, if not, simply moves from room to room following the room number order. He has no idea whether a patient is in his/her room or is away because of a scheduled test or procedure. Dr. Ready enters the first room. After checking the bathroom, he determines that John Doe is off the floor. Dr. Ready does not have time to track down Mr. Doe or to determine when Mr. Doe will return. Finding out when Mr. Doe will be back on the floor would be a multi-step process of looking for the nurse, walking to the front of the floor, checking a transportation schedule, etc. When asked by his attending (supervisor) physician if Mr. Doe went for the echocardiogram this morning, Dr. Ready responds that he thinks that’s where Mr. Doe is as the test was ordered yesterday. Dr. Ready pauses to record the necessary billing information for the time required to visit Mr. Doe.

Dr. Ready walks into a room to see his next patient, April Dancer, and finds he has very little information on Ms. Dancer from the doctor who admitted her. Valuable minutes tick by as he has to leave the room to get more information. He returns as Ms. Dancer is whisked off to diagnostic testing.

**Future State:** A physician, Dr. Able, arrives on the second floor of the hospital. A graphic appears on his PDA showing the rooms assigned to his patients. Rooms are marked differently depending on whether the patient is currently in the room. Because Mr. John Doe is not currently in his room, the PDA also shows his current location and the time at which he is expected to return. Ambulating patients are notified that the doctor is making rounds. As Dr. Able enters his next patient’s (April Dancer) room, a subset of her medical record, together with her current vital signs, appears on the doctor’s PDA. Dr. Able’s proximity to the patient is logged, for input to (or reconciliation with) billing applications. Dr. Able leaves the room to see his next patient, while Ms. Dancer is whisked off to diagnostic testing.

### <B>Example 2

**Present State:** Sydney Bristow experiences difficulty breathing. Dr. Ready hears one of nurses shouting “Is there a doctor? …I need help!” Dr. Ready pulls out his SignOut sheet and fortunately finds Ms. Bristow on it. He realizes there’s a lot he doesn’t know about Ms. Bristow such as a complete list of medications, allergies, and the like. Neither the nurse or doctor knows how long she has been short of breath, but it looks like she may need ventilation.

**Future State:** Sydney Bristow experiences difficulty breathing. The patient monitoring application, given her medical history, triggers an alert that this is a serious condition. Notification is sent to the assigned nurse (Nurse Baker), to the nearest nurse (Nurse Charles), and, because he is in the hospital, to her primary physician (Dr. Able) identified by the SignOut system (physician coverage application). Nurse Charles enters Ms. Brisow's room. Nurse Baker and Dr. Able are notified that someone has responded. If no one had responded within a reasonable time, the next care provider on the escalation path would have been notified (e.g., the nursing supervisor on duty and Dr. Able's coverage) (Kuperman et al., 1996; Wagner et al., 1999). The escalation path is, itself, context-sensitive.

***<B>Example 3***

**Present State:** Nurse Baker views her triage list on the crumpled piece of paper on which she rapidly scribbled notes before the previous nursing shift left work. She looks over the list and makes decisions on priorities as best she can. She was told all patients are stable and simply decides to visit the patients room by room.

**Future State:** Nurse Baker views her triage list on her PDA, where her assigned patients are ordered according to an assessment of their current need for her care (e.g., from sensors, call buttons, and medical history, etc.). The ordering is determined by a combination of medical practice and hospital policy. Nurse Baker does not see an electronic chart, but rather a red-yellow-green summary of vital signs along with the triage list. Jane Smith is clearly most in need of her help, but along with the vital-sign summary, a notation shows that another nurse is with her. Nurse Baker proceeds to respond to the next most-critical patient.

***<B>Example 4***

**Present State:** Dr. Ready, an attending physician, finishes patient rounds, and now answers his pager, which has five pages he has not answered. All he knows is the phone numbers. He has no context in terms of who paged him, why they paged him, or how urgent the page is. A few hours later, while routinely checking laboratory results, he discovers an abnormal result and then goes to look for the chart. He then looks at the X-Rays on a workstation. Treatment is delayed by several hours (Tate et al., 1995). A few weeks later he finds out that one of the residents has violated governmental regulations on hours worked (Conigliaro et al., 1993; Kelly et al., 1991; Thorpe, 1990).

**Future State:** Dr. Able, an attending finishes patient rounds. Now that he is no longer “busy,” all pending non-emergency notifications are presented on his PDA. There are two. One is a notification that one of the residents under Dr. Able’s supervision is about to violate governmental regulations on hours worked per four-week period. The notification is based on the resident’s schedule, and on her location as logged within the hospital in this four-week period. Because the resident has also been notified, Dr. Able takes no action at this time. The other notification is an important (but not urgent) lab result for Jane Smith. Dr. Able taps the lab result item, and it appears on the PDA, along with a subset of her electronic chart needed to make sense of the result. He walks to the nursing station to view Ms. Smith’s X-ray on a widescreen display. Based on his proximity to the display, he is presented with a pick-list containing the last three patients seen as well as three other nearby patients. Dr. Able reviews Ms. Smith’s data, and decides on a course of treatment.

## *<B>Discussion*

The above scenarios provide qualitative demonstrations of the value of context information to physicians in their day-to-day tasks. Preliminary studies have shown that availability of information is a key determinant of utility to physicians (Cohen et al., 1982; Covell et al., 1985; Curley et al., 1990). Handhelds provide immediate availability. A workstation-based display showing patient status, availability, and alerts by room number was tested and found to be effective (Geissbuhler et al., 1997). Alerting studies have found that time to respond decreased significantly (Shabot et al., 2000; Kuperman et al., 1999; Tate et al., 1995) when alerting was automated and sent to a pager or two-way messaging device. A randomized control trial by Kuperman and his colleagues (1999) demonstrated a reduction of 38% in response time. Key features of the alerting system are coverage schedules and escalation algorithms (Wagner et al., 1998; Wagner et al., 1999).

A context ecosystem can support future states as described in these scenarios. Adapters gather raw context information like vital signs, the location of individuals, or their shift schedule. Data aggregators or "composers" process information from the adapters to yield higher-level information (in this case, conditions constituting medical alerts and red-yellow-green summaries of patient information for a nurse). Context also feeds into back-end systems (like accounting applications), and causes notifications to be sent to people based on a rich representation of communication semantics. In a mature ecosystem, manufacturers of medical devices provide context information in standardized form, and composer specifications reduce the cost of producing context-aware applications. New applications are built and deployed quickly; in this scenario, they improve patient care, improve hospital efficiency and auditability, and reduce nurse and physician workloads for routine situations. Based on the ability of a context ecosystem to support future scenarios like the ones above, we assert that a context ecosystem can bring long-term value to the healthcare domain.

## <A>Requirements

As described above, the healthcare domain provides a number of interesting and high-value scenarios for context-aware applications. It is also a demanding domain, with highly sensitive, rapidly changing biometric data requiring timely evaluation and analysis, and for which the handling is constrained by national privacy laws. Broad and rapid adoption of these scenarios is inhibited by the lack of supporting middleware and appropriate data sources, that is, by the lack of a healthcare context ecosystem.

Biometric monitoring provides a good example. Such telemetry data today is analyzed by dedicated software, generally supplied by the same company as supplies the sensor device. A new source device requires the hospital to purchase a new system for analysis. A patient may be required to use the hospital sensors even though a superior data source (e.g., an implanted monitor) is available simply because the implanted device is incompatible with the hospital's monitoring system. What is needed is the ability to transform this data so it is usable and the ability to preserve both the privacy of the patient and the ability to protect the application investment of the hospital. As the population ages, and makes more use of remote biometric monitoring, scalability and manageability will become critical.

In addition to the healthcare industry, a number of other industries have deployed custom solutions using context data over the last decade. Location-based services (e.g., enhanced 911) and sensor-based applications (e.g. utility-consumption monitoring) have used context data in a variety of consumer services. Examination of the successes and failures of such custom solutions suggests broader requirements necessary to take full advantage of what are, largely, a set of newly available data sources.

A context ecosystem will include producers of context data, providers of context-awareness middleware, developers of context-aware applications, system administrators, and end users. Based on our observations on existing custom solutions, and the combined needs of these participants, we articulate the following requirements on the technology infrastructure for a context ecosystem.

**Valuable Applications:** First and foremost, the ecosystem must provide useful applications that demonstrate a clear return on investment to all participants in the ecosystem, including the enterprises and the consumers.

**Low Barriers to Entry:** Initial intra-enterprise applications will be developed using only one or two context sources. As the ecosystem matures, additional sources will be available. To mature fully, the ecosystem needs a critical mass of context sources from a diverse set of data providers.

**Tools:** Developers require tools to enable rapid context-aware application development. With such tools, developers should be able to produce reusable components, feeding back into the ecosystem.

**Extensibility:** It must be easy to add new kinds of context data. Adding new context sources should not require changes to existing applications.

**Open Interchange:** In an inter-enterprise ecosystem, precise, open specifications of the syntax and semantics of context data will enable the exchange of context information among unrelated individuals and organizations. The use of open standards like XML will encourage interoperability, including data exchange protocols and data encodings. The syntax and semantics of context data must also encompass a variety of metadata, such as quality of information, data origin, or data cost.

**Aggregation:** The system must make it easy to aggregate, filter, transform, and summarize data. Application developers, and eventually data providers, will write composers and offer them as plug-ins to standard middleware offerings.

**Security and Privacy**: Robust notions of security and privacy are essential to the success of a context ecosystem, especially in application areas like healthcare where government requirements impose significant constraints. Less stringent constraints on the privacy of individuals and enterprises must also be accommodated, such as restricting to whom the location of a person can be disclosed.

**Scalability:** The system must support scalability along a number of dimensions. First, it must cope with a growing number of diverse and geographically distributed data sources that may dynamically come into and go out of existence. Second, because it must cope with a high volume of data elements coming from these sources, it must be capable of aggregating and filtering this information set. Finally, it may need to support a large number of applications and end users.

**Dynamic Discovery:** As this context ecosystem grows, querying the infrastructure dynamically for available context information becomes a critical requirement, as do the dynamic discovery of evolving sources and the rebinding of composer instances to new providers.

**Manageability:** Large context ecosystems will need to be autonomic systems, but will also require tools to manage and administer them, especially with context exchanges among enterprises. Autonomy and manageability must be considered at design time rather than when failures occur. As the ecosystem evolves into a multi-enterprise environment, it must cope with competing interests and the lack of a single administrative authority.

**Actuation:** Context-aware applications may act on the environment in addition to observing it. Although some applications may take direct action, others may need the middleware to transform high-level, abstract actions into low-level, concrete operations.

## <A>Previous Work

A significant body of work exists in the general area of context-aware computing systems. In this section, we identify representative examples in the areas of applications, middleware, and data sources, while commenting on how each satisfies a subset of the requirements just described.

### *<B>Applications*

Some of the earliest context-aware applications were built around the use of location. The ActiveMap project (Want et al., 1992) annotated the floor plans of a space with the location of people and objects. Similarly, the CyberGuide (Abowd et al., 1997) and Guide (Davies et al., 2001) systems concentrated on supporting tourists (indoors in the case of CyberGuide, and outdoors in the case of Guide). These early projects were focused on the positioning, networking, and device technologies. They showed the relevance of context information to specific application domains by offering proofs of the concept. As a consequence, they stopped short of any concerted effort to systematize their approach to some general form of context information. These applications remain good examples of the types of applications that a context ecosystem must support.

Another early system, Cooltown (Kindberg and Barton, 2001), examined the use of Web technology in supporting users of pervasive devices interacting with their environment. The authors explored how content can be pulled from the environment onto the pervasive device (e.g., display conference room services when entering a banquet hall) as well as how content could be pushed from the device to the environment (e.g., push the URL for a presentation to a projector, push an image from a user's camera to a local printer). The user’s proximity to and/or interaction with a device located in the environment enabled this interaction. The interaction permitted small devices to react to the user’s context and also allowed those devices to control aspects of the environment. The focus of this work was on understanding the interactions required to support a user within a particular location. It was not focused on supporting the more general context ecosystem that we have outlined here, though a general context ecosystem should support the types of interactions these authors envisioned.

## <B>Middleware

The Context Toolkit (Dey et al., 2001) was one of the first efforts to provide a more general framework for supporting context-aware applications. In this system, context widgets capture raw context information from devices or applications, and retain it for possible historical analysis. Context interpreters combine one or more pieces of context to produce a new item, and context aggregators present interfaces to all the context information related to a single entity. Services execute actions on behalf of applications, and are closely tied to widgets; indeed, the first implementation of services incorporated them into the associated widgets. “Discoverers” provide context registration and discovery services, and the authors also envisage “situations” as an abstraction for combining disjoint pieces of context with complex conditions. The Context Toolkit contains support for many of the requirements identified earlier, including aggregation, extensibility, and low barriers to entry, but faced challenges in the areas of privacy, tooling, and manageability.

iQueue (Cohen et al., 2002) and Solar (Chen and Kotz, 2002) represent another approach to enabling context-aware applications. These systems model context information as streams of events originating in a publish-subscribe system; events are encoded as serialized Java objects. There is a tree- or graph-structured hierarchy of *operators*, similar to what we have termed composers in our ecosystem, and a hierarchical naming scheme for context data. The iQueue system offers the first high-level abstraction (Cohen et al., 2002) to allow developers of context-aware applications to write queries for needed data. iQueue and Solar focused on many of the same requirements, including aggregation, extensibility, and scalability. iQueue provided more in the way of tooling than Solar, but Solar provided more support in the area of privacy and dynamic source discovery. Neither system provided support for manageability or actuation.

As part of the Aura project (Garlan et al., 2002), Judd and Steenkiste (2003) developed the Context Information Service (CIS). They model context with four types of providers: people, areas, networks, and devices. Client applications issue queries against the CIS that are processed by a query synthesizer to decompose them into simpler queries that can be handled by the various context-information providers. The providers may compute query results dynamically, retrieve them from a cache, or perform an SQL query against a normal relational database. Queries can include references to metadata such as accuracy, confidence, update time, and sample interval. CIS focuses on support for aggregation and for quality of information (an aspect of the syntax and semantics requirement). It also addresses scalability. The authors make no mention of how CIS approaches actuation, manageability, or dynamic source discovery, and it offers minimal support in the way of tooling.

## *<B>Sources*

Other research has examined sources of context information. Given the importance of location as a source of context, a significant amount of work has been done in this area, which we will not review here. Interested readers are referred elsewhere for a survey of this work (Hightower and Borriello, 2001).

Beigl and his colleagues have examined how to augment everyday items, such as coffee mugs, to provide and respond to context information (Beigl et al., 2001). Researchers at the MIT Media Lab have also investigated adding context to everyday items, such as beds and trivets (Leiberman and Selker, 2000).

The MUSE system (Castro and Muntz, 2000) explores the concept of fusion services that infer context information from sensor data. They use Bayesian networks and apply an information-theoretic approach to estimate a probabilistic indoor location from an IEEE

802.11 network. Though their system also provided some middleware support, the focus of this work falls into the area of data aggregation and interpretation.

Shafer and his colleagues explored visual context, multi-modal interactions, and automated behaviors as part of their Easy Living project (Shafer et al., 2001). They obtained visual context from cameras embedded in the environment, using gaze direction to resolve object references, and presence near a device to activate that device. This use of visual context is an interesting example. In this case, both the input data (e.g., the camera feed) and the interpretation of that input (e.g., user gazing at lamp) can be viewed as forms of context information, yet the analysis of that data may need to be performed in the middleware, especially if that interpretation depends not just on the camera input but also the location obtained from another source.

This brief survey shows the breadth of previous work on context-aware computing, and points out that none of the previous work has really addressed the more comprehensive notion of a context ecosystem and how it might be enabled in a systematic fashion. We address this in the next section.

## <A>Architecture

This section describes the technological infrastructure of a context ecosystem designed to address the requirements presented in the previous section. We also present a concrete example of the middleware component of such an ecosystem.

Figure 1 presents a layered view of the technology infrastructure underlying a context ecosystem, as illustrated by our CxS prototype (Cohen et al., 2005). At the lowest level of the infrastructure, raw information is obtained from sources such as sensors, vehicles, devices, and software. Examples of such information are the presence and contents of an RFID tag, the level of pollen sensed in the air, the weight and location of a vehicle, the contents of a person’s calendar, and the idle/busy status of a workstation.  The data sources can thus be either domain-specific (e.g., glucose or cardiac monitor) or domain-independent (e.g., location, availability). In general, context sources may push information to the middleware spontaneously, or may provide it in response to explicit

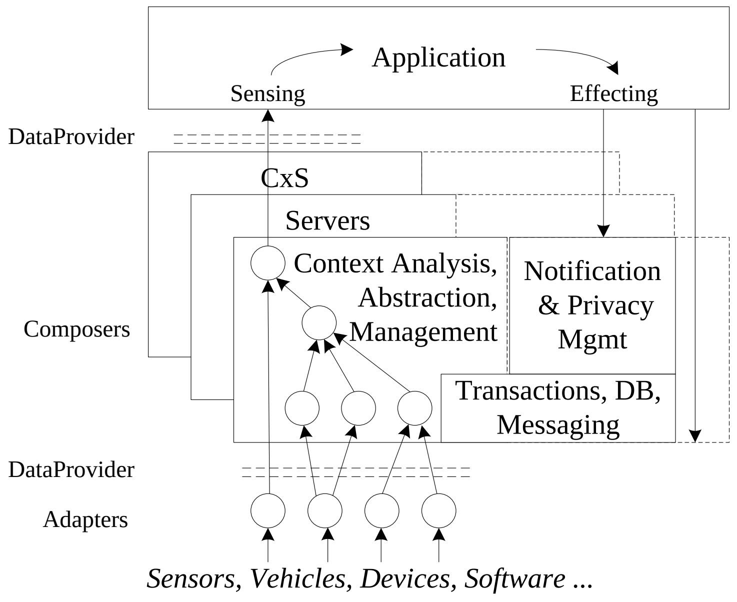
Application
Sensing
Effecting
DataProvider
CxS
Servers
Context Analysis, Abstraction, Management
Notification & Privacy Mgmt
Composers
Transactions, DB, Messaging
DataProvider
Adapters
Sensors, Vehicles, Devices, Software ...

**Figure 1. CxS Architecture**

requests. These raw sources of information are wrapped by *Adapters*, which transform the native information into a form that can flow across the *DataProvider* interface. In our prototype, data transmitted across this interface is encoded with XML according to a schema specific to each provider kind, and the transmission can follow either a push or a pull model. These data sources represent basic *data providers*.

The CxS middleware can be deployed on a network of server nodes. Inside one or more CxS servers, *Composers* process data objects received from adapters and other composers and make that data more easily consumable by applications. Composers can, in general, implement arbitrary algorithms over their inputs, such as aggregation, summarization, and combination. For example, a composer might use the state of a person's computer keyboard and mouse, telephone, and office door to provide applications with a good indication of "interruptibility" (Fogarty et al., 2005). Similarly, the current pollen count combined with a patient's location could be used by a medical application to suggest changes in medication dosage. At the highest level, composers in turn offer application-oriented context data via the DataProvider interface. The use of a common interface as input and output of the composers allows the infrastructure to support a hierarchy of composers.

Thus, the middleware layer is crucial to satisfying a number of our requirements. The adapters, composers, and data provider interface support low barriers to entry, extensibility, aggregation, and dynamic discovery. The use of XML enables the middleware to support the open interchange of context data, once the context ecosystem evolves to support open standards.

Valuable applications are the driving force behind a context ecosystem. In our prototype infrastructure, applications submit queries for data providers to the middleware and the middleware responds with one or more data providers satisfying the query. The application then chooses one (or more) data providers and requests (or subscribes to) the required data. As the actual data providers satisfying a query change, the flow of data through the composer hierarchy is adjusted dynamically (Cohen et al., 2005). In this way, the desired information is received despite, for example, physical movement of an RFID-enabled package past a number of individual RFID readers. This aspect of CxS helps satisfy our requirement for dynamic discovery.

Using CxS, we have constructed composers programmatically through Java programming interface, which we call the Java data-composition facility (JDCF), and through the use of an abstract, high-level description, which we call iQL, of the computations they perform (Cohen et al., 2002). We have also experimented with building Eclipse-based tools to support application and composer developers, but tooling remains an area in need of additional exploration.

Our prototype middleware is designed to allow CxS servers to run independently on many different machines, running in multiple administrative domains. This feature supports our scalability and manageability requirements. Our experience and tooling in these areas is, however, limited as this is currently an active area of research.

Although some applications will simply retrieve context information for use in supporting their users, others may need to go beyond simple sensing of the context to also effecting changes. Notification is one of the most common actions taken in response to dynamic data. For example, an application might respond to location and traffic data by telling a driver to turn left in two blocks to avoid a traffic jam. This concept is shown in

Figure 1 by the “effecting” arrow at the top and the arrow into the notification component. This aspect of CxS helps satisfy our requirement for actuation.

As we explored how to support context-aware applications, we recognized that one could not responsibly build such support or applications without also providing the ability to control access to the information. Many custom solutions we have seen either ignored the privacy issue or built their system to avoid it (making the system significantly less useful). Instead, we chose to address the problem directly. The goal of the *Privacy Management* component is to mediate the privacy concerns of both individuals and enterprises. No context information is released by CxS without consultation of the Privacy Management component. We recognize that different entities (e.g., the hospital, patient, physician, insurer) will have different policies about context data. This component allows CxS to control access in an appropriate fashion. This aspect of CxS helps satisfy our requirement for security and privacy.

## <A>Experience

We have begun prototyping a number of components that support the vision of a context ecosystem described earlier. The CxS middleware presented in the previous section is a central component. In addition, we have developed data providers and composers of various types. Most importantly, we have prototyped a number of applications in the demanding domain of healthcare.

We have built adapters that allow the context middleware to obtain data from more than a dozen different context sources. These sources include location information (obtained from 802.11 triangulation, active badges, and cell-tower data), calendar scheduling (obtained from Lotus Notes), instant messaging presence (obtained from Lotus Sametime), virtual activities (obtained from desktop computers and PDAs), connectivity information (obtained from the WebSphere Everyplace Connection Manager), telephone status (obtained from a PBX and from a VoIP soft phone), web services (obtained from a weather site and from a stock quotation site), and the like. Each new source of context requires a short time (in the order of a day or two in our experience) to integrate into the system.

We have developed a number of composers that aggregate these data sources into higher-level context information. These composers include one that monitors patient status by aggregating all telemetry for one patient. It generates alerts for unusual data reported by that patient's monitors. Another composer enhances context information with more static information maintained in back-end databases.

One application we have prototyped is a nursing triage application. The goal of this application was to increase the efficiency of the nursing staff by showing each nurse a summary of the status of each patient in his or her care. The application took data from mock biometric sensors (we did not have access to digital biometric sensors at the time), aggregated that data, analyzed that data, and presented a summary view in a web-based triage display. A per-patient view allowed the nurse to access detailed data for each patient. We expect to develop this prototype further with additional biometric sensors.

A second application we have prototyped is a resident-hours monitoring application. The goal of this application was to verify compliance to the recent U.S. regulations concerning the number of hours residents are permitted to work in a hospital. Violating these regulations can result in teaching hospitals losing their accreditation. For this application, residents are given active badges and the hospital entrances used by those residents are instrumented with RFID readers. In addition, the application has access to the residents' schedules. With this information, the application tracks the hours worked by each resident. If a resident violates the rules, the administrator charged with supervising the resident is alerted immediately. More interestingly, if completion of scheduled hours would place a resident in violation of the regulations, the resident and the supervisor are notified of the potential violation. This application is currently available as a solution demonstration, and we expect to deploy it in a hospital setting in the future.

A third application we have prototyped is an operating-room monitoring application. The goal of this application was to help ensure patient safety by verifying that the correct patient is in the correct operating room. For this application, patients are outfitted with active badges during their pre-operation procedures. When a patient enters the operating room, the system detects this event and verifies that the patient is in the correct operating room based on operating-room schedules. In the event of an error, numerous alarms go off, including both audible and visual alerts. In addition, electronic notification messages are sent to the staff in charge of the operating room. These alerts escalate until the situation is resolved. This application is also currently available as a solution demonstration.

We have thus far employed human notification as our sole actuator. We are currently engaged in the design and implementation of “effecting” interfaces on the downward path, as implied by the dashed boxes in Figure 1.

One lesson that we have learned in building the context ecosystem components to support healthcare is that context services can provide great business value without necessarily entailing high capital outlays. In the operating room and resident monitoring applications, relatively little capital outlay is required to obtain improvements in patient safety and operational efficiency. Furthermore, once a return on investment is shown in one application, the deployment can be extended and the ecosystem reused for the next application.

## <A>Conclusions and Further Work

Making businesses responsive to dynamic, changing environments is crucial, and context information is key to adapting to such environments. This paper has presented our vision of a context ecosystem designed to help enterprises respond to dynamically changing environments. This ecosystem is built on a comprehensive software infrastructure that supports context-aware solutions in an extensible, reusable, and scalable way, and a set of data providers, aggregators, and applications to solve specific business problems. This ecosystem and its constituent technologies facilitate collecting, processing, and analyzing environmental context data, and ultimately enable dynamic changes to the real world in response. The ecosystem is not specific to any particular domain and so context sources common to many domains can be shared. We have argued that such an ecosystem enables enterprises to derive value over the long term because applications become easier to build and applications evolve gracefully as technology improves.

This paper has also described a key component of the above ecosystem—a middleware system prototype called CxS that provides context information to applications and services. CxS separates the concerns of accessing context data from the concerns of aggregating and analyzing that data and, ultimately, provides applications with the data needed. A context service built around CxS can amortize the cost of incorporating new sources of context across many context-aware applications and services, and provide application developers with a common API for context information.

We envision that the creation of an ecosystem of standards and developers will enable us to access data, compose data, and create both vertical and horizontal context-based applications. In order for this context ecosystem to evolve, further work is needed in a number of areas. First, a critical mass of context data sources must be available to application developers "out of the box," to reduce the effort needed to make new applications context-aware and the benefits of using the middleware immediate and obvious. Second, there must be standardization of the exchange of context information. Third, the software infrastructure must be scalable to support large numbers of possibly verbose context sources. Finally, it is clear that we need to a rich set of tools that can effectively open up the power of context technologies to developers.

## <A>Acknowledgements

This paper has built upon the work of the entire Context  Technologies team in IBM, and has benefited greatly from the comments of Norman Cohen, the support of Barry Leiba, and the applications built by Marion Blount, John Davis, Archan Misra, Wolfgang Segmuller, and Daby Sow.

## <A>References

Abowd, G. D., C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton (1997). Cyberguide: A Mobile Context-Aware Tour Guide, *Wireless Networks*, 3(5):421–433.

Beigl, M., H.-W. Gellersen, and A. Schmidt (2001). Mediacups: Experience with Design and Use of Computer-Augmented Everyday Artefacts, *Computer Networks*, 35(4):401-409.

Castro, P. and R. Muntz (2000). Managing Context Data for Smart Spaces, *IEEE Personal Communications*, 7(5):44-46.

Chen, G. and D. Kotz (2002). Context Aggregation and Dissemination in Ubiquitous Computing Systems, *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, Callicoon, NY, 105–114.

Cohen, N. H., H. Lei, P. Castro, J. S. Davis II, and A. Purakayastha (2002). Composing Pervasive Data Using iQL, *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, Callicoon, NY, 94–104.

Cohen, N. H., P. Castro, and A. Misra (2005). Descriptive Naming of Context Data Providers, *Proceedings of the Fifth International and Interdisciplinary Conference on Modeling and Using Conference (CONTEXT-05)*, Paris, France (to appear).

Cohen, N. H., A. Purakayastha, L. Wong, and D. L. Yeh (2002). iQueue: A Pervasive Data-Composition Framework. *Proceedings of the Third International Conference on Mobile Data Management*, Singapore, 146-153.

Cohen, S. J., M. Weinberger, S. A. Mazzuca, and C. J. McDonald (1982). Perceived Influence of Different Information Sources on the Decision-Making of Internal Medicine House Staff and Faculty. *Social Science and Medicine*, 16(14):1361-4.

Conigliaro J., W. H. Frishman, E. J. Lazar, and L. Croen (1993). Internal Medicine Housestaff and Attending Physician Perceptions of the Impact of the New York State Section 405 Regulations on Working Conditions and Supervision of Residents in Two Training Programs. *Journal of General Internal Medicine*, 8(9):502-7.

Covell, D. G., G. C. Uman, and P. R. Manning (1985). Information Needs in Office Practice: Are They Being Met? *Annals of Internal Medicine*, 103(4):596-9.

Curley, S. P., D. P. Connelly, and E. C. Rich (1990). Physicians' Use of Medical Knowledge Resources: Preliminary Theoretical Framework and Findings. *Medical Decision Making*, 10(4):231-41.

Davies, N., K. Cheverst, K. Mitchell, and A. Efrat (2001). Using and Determining Location in a Context-Sensitive Tour Guide, *Computer*, 34(8):35–41.

Dey, A. K. (2001). Understanding and Using Context, *Personal and Ubiquitous Computing Journal*, 5(1):4–7.

Dey, A. K., D. Salber, and G. D. Abowd (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction (HCI) Journal*, 16(2–4):97–166.

Fogarty, J., S. E. Hudson, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. C. Lee, and J. Yong (2005). Predicting Human Interruptibility with Sensors, *ACM Transactions on Computer Human-Interaction*, 12(1):119-146.

Garlan, D., D. P. Siewiorek, A. Smailagic, and P. Steenkiste (2002). Project Aura: Toward Distraction-Free Pervasive Computing, *IEEE Pervasive Computing*, 22–31.

Geissbuhler, A., J. F. Grande, R. A. Bates, R. A. Miller, and W. W. Stead (1997). Design of a General Clinical Notification System Based on the Publish-Subscribe Paradigm. *Proceedings of the American Medical Informatics Association (AMIA) Annual Fall Symposium*, Nashville, TN, 126-30.

Hess, C. K. and R. H. Campbell (2003). A Context-Aware Data Management System for Ubiquitous Computing Applications, *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, Providence, RI.

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

HIPAA (2005). Information available at www.hipaa.org.

Judd G., and P. Steenkiste (2003). Providing Contextual Information to Pervasive Computing Applications, *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 133–142.

Kannry J. and C. Moore (1999). MediSign: Using a Web-based SignOut System to Improve Provider Identification, *Proceedings of the American Medical Informatics Association (AMIA) Symposium*, Washington, D.C., 28(6):550-4.

Kelly A., F. Marks, C. Westhoff, M. Rosen (1991). The Effect of the New York State Restrictions on Resident Work Hours. *Obstetrics & Gynecology*, 78:468-73.

Kindberg, T. and J. Barton (2001). A Web-Based Nomadic Computing System, *Computer Networks*, 35(4):443-456.

Kuperman, G. J., J. M. Teich, D. W. Bates, F. L. Hiltz, J. M. Hurley, R. Y. Lee, and M. D. Paterno (1996). Detecting Alerts, Notifying the Physician, and Affering Action Items: A Comprehensive Alerting System, *Proceedings of the American Medical Informatics Association (AMIA) Symposium*, Washington, D. C., 704-8.

Kuperman, G. J., J. M. Teich, M. J. Tanasijevic, N. Ma'Luf, E. Rittenberg, A. Jha, J. Fiskio, J. Winkelman, and D. W. Bates (1999). Improving Response to Critical Laboratory Results with Automation: Results of a Randomized Controlled Trial. *Journal of the American Medical Informatics Association*, 6(6):512-22.

Kushniruk A., T. Karson, C. Moore, and J. Kannry (2003). From Prototype to Production System: Lessons Learned from the Evolution of the SignOut System at Mount Sinai Medical Center. *Proceedings of the American Medical Informatics Association (AMIA) Symposium*, 381-5.

Leiberman, H. and T. Selker (2000). Out of Context: Systems That Adapt To, and Learn From, Context, *IBM Systems Journal*, 39(3-4): 617-632.

Román, M., C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt (2002). A Middleware Infrastructure for Active Spaces, *IEEE Pervasive Computing*, 1(4):74–83.

Shabot, M. M., M. LoBue, and J. Chen (2000). Wireless Clinical Alerts for Physiologic, Laboratory and Medication Data. *Proceedings of the American Medical Informatics Association (AMIA) Symposium*, Los Angeles, CA, 789-93.

Shafer, S. A. H., B. Brumitt, and J. Cadiz (2001). Interaction Issues in Context-Aware Intelligent Environments, *Human-Computer Interaction*, 16(2-4):363-378.

Tate, K. E., R. M. Gardner, and K. Scherting (1995). Nurses, Pagers, and Patient-specific Criteria: Three Keys to Improved Critical Value Reporting. *Proceedings of the Nineteenth Annual Symposium on Computer Applications in Medical Care*, New Orleans, LA, 164-8.

Thorpe, K. E. (1990). House Staff Supervision and Working Hours: Implications of Regulatory Change in New York State [see comments]. *Journal of the American Medical Association*, 263(23):3177-81.

Wagner, M. M., S. A. Eisenstadt, W. R. Hogan, and M. C. Pankaskie (1998). Preferences of Interns and Residents for E-mail, Paging, or Traditional Methods for the Delivery of Different Types of Clinical Information, *Proceedings of the American Medical Informatics Association (AMIA) Symposium*, Orlando, FL, 140-4.

Wagner, M. M., F. C. Tsui, J. Pike, and L. Pike (1999). Design of a Clinical Notification System, *Proceedings of the American Medical Informatics Association (AMIA) Symposium*, Washington D.C., 989-93.

Want, R., A. Hopper, V. Falcao and J. Gibbons (1992). The Active Badge Location System, *ACM Transactions on Information Systems*, 10(1):91-102.