

Atira: A Responsive Environment for Document Users

Deborra Zukowski, James Norris, John Rojas,
Arthur Parkos, John Braun, Hiram Coffy

Advanced Concepts and Technology
Pitney Bowes
35 Waterview Dr, Shelton, CT, 06484
{deborra.j.zukowski, james.norris, john.rojas, arthur.parkos,
john.braun, hiram.coffy}@pb.com

Abstract. We have been interacting with paper(-like) documents for millennia and have developed practices based on such technology. Likewise, we've been interacting with electronic documents for decades, developing a set of practices based on that technology. As the promise of ubiquitous computing is realized, we will soon have a rich new set of document technologies that will replace or augment our use of distinct paper/electronic documents. However, as McLuhan noted, "We shape our tools and thereafter our tools shape us." [1] Practices will co-evolve with our use of new technologies, and those practices may be very novel. At Pitney Bowes, we are trying to create a ubiquitous computing environment that will help us "discover" potential new document practices. The environment must be capable of capturing and leveraging users' interactions with documents as they work. Also, it must be flexible so that we can easily experiment with the interplay between technology and practices. This paper presents our ongoing work in developing such an environment.

1 Introduction

There once was a belief that paper would disappear as digital technology was more universally adopted, that all of our document-based practices would leverage only the digital technology. Yet, people continue to use paper. Digital technology even appears to promote its use, especially in office and related work environments. Researchers at the School of Information Management and Systems at the University of California at Berkeley have been tracking the growth of information, including that rendered on paper. In their 2003 study, they say:

Paperless society? The amount of information printed on paper is still increasing, but the vast majority of original information on paper is produced by individuals in office documents and mail, not in formally published titles such as books, newspapers and journals [2].

Social scientists have been studying the use of paper for decades, trying to better understand why it persists – and grows – as it does. In *The Myth of the Paperless Office* [3], Sellen and Harper have proposed that paper has affordances that help people use documents to better support their work. The Berkeley study substantiates their hypothesis.

There have been several different types of efforts in this area over the last 10 years [4], [5], [6], but as Wieser's vision of invisible computing [7] becomes a reality, we can revisit this work on his terms. Paper is ubiquitous. It provides a natural means for interaction with invisible computing environments. The environment that we build must be able to capture and identify these interactions, and must be able to present the context of document use in a way that can enrich a user's work experience.

At Pitney Bowes, a key and growing part of our business is mail and document management for our office and industrial customers. We provide services for the creation, production, distribution and storage of documents in both paper and digital form. To this end we are developing an immersive environment that we work within and that we can easily deploy for others. From this, we hope to see new document practices emerge as afforded by ubiquitous computing technologies. And as we learn, we need an environment that can easily evolve to better support those new practices.

We began by defining a responsive environment, called Atira. We emphasized the need for runtime flexibility, so that we could better experiment with a wider set of interactions. Our work has yielded an environment comprised of two pieces. First, we've defined a high-level architecture that includes a model for context, an infrastructure service that manages context, and an API that applications can use to leverage the service. Second, we've defined a framework, called the PAUR Pyramid, for building components that leverage sensed information to construct context and that determine and announce interesting happenings in the environment.

This rest of this paper describes key insights and our current approach, based on those insights. Section 2 briefly describes our initial prototype. Section 3 describes some insights we gained from building the prototype and interacting with the environment. Section 4 defines our second, and current, iteration on our high-level architecture and framework for building context. Section 5 discusses related work and how we might leverage the work as we progress. Section 6 presents issues and work that lay ahead of us.

2 Early Work

To build out an environment that was responsive to physical documents, we embarked on a three-step approach. First, we proposed an approach that would be capable of capturing interactions that document users would have with paper documents. This approach included attaching RFID tags to documents, thereby providing them with both identity and a means to communicate. Second, we defined a high-level architecture, as shown in Figure 1, that segmented the work needed to intercept, analyze, and leverage document-based activities. Third, we built an initial prototype that supported a range of activities, as described in an interaction script. The script included using physical documents as a means of transferring knowledge that was gained by others who read the document, keeping track of documents actively in use, finding related physical documents using digital search mechanisms, and explicitly retiring a document.

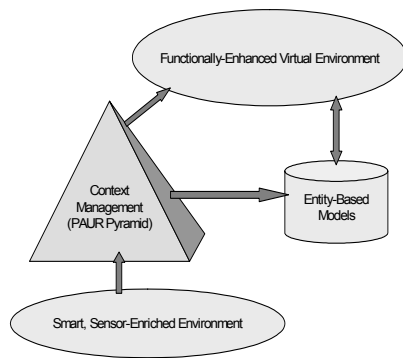


Figure 1

Given that physical entities (people and documents) are tagged with RFID, we enriched the environment by adding readers to doorways, work surfaces, and cabinet drawers. We also added “virtual sensors” to applications that support document interaction, e.g., searching for a document. With this sensor-enriched environment, when a person interacted with a document and this interaction was detectable by sensors, messages were generated and posted to a publish/subscribe message space. Messages were customized to the instance of the sensor, e.g., those deployed at doorways produced a message noting entry or exit. Those deployed on work surfaces produced a message noting containment.

The context management infrastructure built context, storing it as a set of properties within a set of entity models. These models represented all entities known to the environment, including people, documents, application, devices, and locations. We used RDF[8] to describe the state of each entity. The models were pre-populated with people, applications,

and devices known to the environment. These types of entities had to be explicitly managed. Documents, being more dynamic in nature, did not need to be explicitly identified. If an unknown document was sensed, then new instances were made in the model, and document owners could augment the state using a special annotation application.

The context management infrastructure also determined how the environment should respond to context changes caused by interactions sensed in the sensor-enriched environment. We designed a layered framework that allowed us to incrementally add “intelligence” to the infrastructure in support of user tasks within the environment. This framework, called the PAUR pyramid, is shown in figure 2. It is comprised of four layers of components. Components within a layer play the same overall role, but are capable of specialized function by subscribing to a subset of messages from the layer below and processing these messages to add functionality within the layer. This approach thus enabled us to incrementally add function within the layers as we built out our user scenarios.

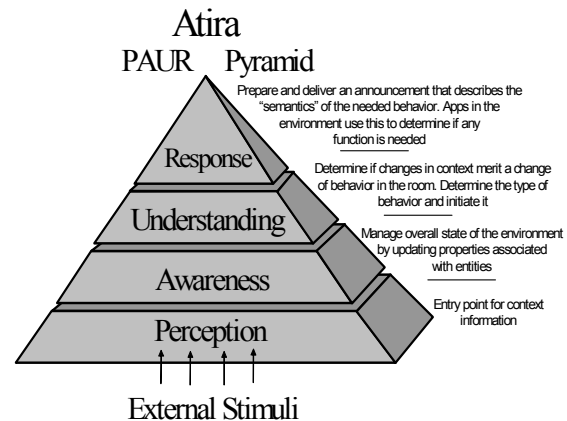


Figure 2

The bottommost layer, Perception, is used to filter the types of stimuli used to build the context. This layer is populated with *perceptor* components, including ones for entry/exit, containment, touch, and command. The perceptors ensure that the sensed entities are known to the environment. Documents, if unknown, are registered. The perceptors then post the valid interaction messages back to the message space.

The Awareness layer is populated with components, called *monitors*, that manage the state of active entities known in the context. Our prototype included document, application, and occupant monitors. The monitors determine activities – a person carrying a document, for example – that also may

indicate a change in state and update all affected entity models. The monitors post a message whenever a model is updated.

The Understanding layer is used to support intelligence in the behavior of the responsive environment. It is populated with components called *grokkers*, that determine the types of activities that are underway in the environment. Our prototype includes skeleton document, occupant, and application grokkers. We did not include much intelligence for our first attempts.

The top-most layer, composed of components called *responders*, is intended to semantically drive environment function. It posts messages that describe the types of actions needed to cause the behaviors identified by the grokkers. Our prototype did not exercise this level of functionality and the responders were basically a pass-through.

Our environment was enhanced with two prototyped applications, a browser-based interface to the environment called the Atira Browser, shown in Figure 3, and an application launcher. The browser showed the current state of the environment, including all persons physically or virtually participating in the environment and the set of documents being actively used by the person logged into the Atira Browser.

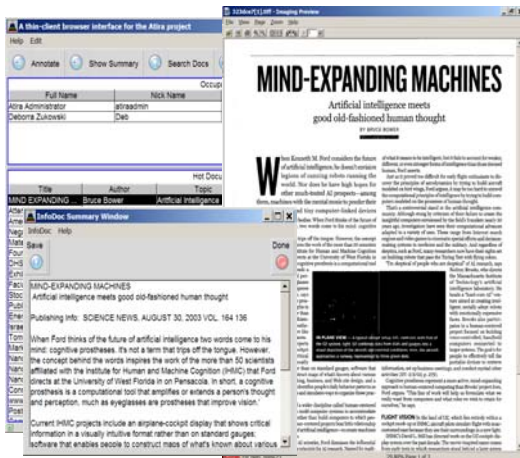


Figure 3

Our prototype allowed us to automatically sense the use of physical documents. When we used a document, the Atira Browser would list that document, and so we could easily access it to add online information about the document. We could then create a digital version of the content by scanning a document with a sensor-enhanced scanner. The digital version would automatically be associated with the physical version by linking the unique identifier of both renderings.

Our prototype also supported “functional spaces,” that is, spaces that imply intended behavior. One such

space, an archival cabinet, provided a tangible interface for retiring a document. When a document was placed in the archive cabinet, it would be removed from all active document lists.

3 Insights

The prototype and interaction script showed the value of using a responsive environment to help bridge the use of physical documents to the digital representation of information, and to provide tangible access to that information. We were also able to use the prototype to support a mini-experiment that investigated the usability of digitally capturing partial information from physical documents. This use of the prototype was unplanned, but it supported the experiment very well. In short, the overall approach looked sound and warranted a second iteration. For this next iteration, we focused on three findings:

- 1) Sensors should report only what they actually know,
- 2) Entity-based context management is unwieldy,
- 3) Ambiguity comes in two forms,
 - a) *Uncertainty*: Interactions are sensed but could mean multiple things and
 - b) *Ignorance*: Interactions may be missed.

First, sensors in our prototype generated messages based on their use. A tag reader at a doorway would introduce an “entry/exit” message into the environment. A tag reader under a desk surface would introduce a “containment” message. The perceptors were shallow pass-throughs for valid messages. This uncovered both practical and theoretical problems. From a practical perspective, having “use-sensitive” sensor deployment meant that the same type of sensor contributed to the environment in different ways. Moving a sensor from one place to another could mean more than a geographic change. It might also mean a functional change. Second, from a theoretical perspective, the higher-layer perceptors were dumb while the sensors below them were smart. Function/abstraction typically increases as information propagates upwards.

Second, entity-based context was unwieldy. For our use, we found that we were duplicating information across various entity instances. For example, the person entity included a list of active documents. The document entity included a list of active users. Both lists were needed to properly show the context of the entity. If a document was archived, the redundant information contained in the person entity had to be tracked down and removed. In general, systems that encourage duplicated information become complex quickly and are prone to errors when redundancy is not handled properly.

Third, the ambiguity introduced by sensor-enriched environments comes in two forms. Sensors may miss events. For example, a person may be missed by a sensor at the doorway of a room, but detected at a table within the room. The person clearly went through the door. Should the system rectify its *ignorance*? Sensors may also detect a set of entities, but be unable to provide enough information to distinguish the activities associated with the set. For example, the simultaneous appearance of a person and several documents within the field of a door sensor strongly implies that a person is walking through the door carrying the documents. However, if a second person enters at the “same” time, then the environment is *uncertain* about who is carrying the documents. Should the environment handle both types of ambiguity in the same way? In both cases, further events help to resolve ambiguity of what happened. With ignorance, however, context is omitted. When the ignorance is resolved, context is added. With uncertainty, context is present but may be probabilistic. Context may be removed as the uncertainty is resolved.

4 Atira As She Stands Today

For our second iteration, we’re focusing on the nature of context and the definition of the lower three layers. The view of the environment as the introduction of stimuli (at the bottom) and the initiation of behavior (at the top) are unchanged. The use of an infrastructure to build context and to notify the environment of changes in context is also unchanged. The biggest changes are in the semantics of our context models and the “intelligence” of our sensors. Both of these changes have impacted the definitions within the PAUR Pyramid framework. The following sections present our current architecture and framework definitions. As we are in the process of building a second prototype, we’ll defer the implementation discussion to a later paper.

4.1 Atira Context Model

To address insight 2, the models now differentiate between the intrinsic properties of entities and the relationships that entities may have with one another. Both entities and relationships are identified with unique URIs (Uniform Resource Identifiers)[9]. Entities have existence, e.g., people, documents, sensors, containers (i.e., spaces), and applications. Their properties describe aspects of their existence that are more statically defined in nature, e.g., name, author, and/or deployment information. Vocabularies for each entity type that determine the properties describing the entity are defined.

Relationship Models

The addition of relationships for representing interactions between entities is one of the bigger changes from our previous work. In our work, relationships are binary in nature; they show the relation between two entities. Relationships, as shown in Figure 4, associated two entities (subject and object) using a linking verb. These verbs are classified into categories, including presence, proximity, and work. Generally speaking, we’ve restricted the scope of what we call “context” to be the set of relationships that currently hold. Relationships that are no longer valid are removed from our context and placed in a historical context model where they remain available, using access methods similar to those for current context.

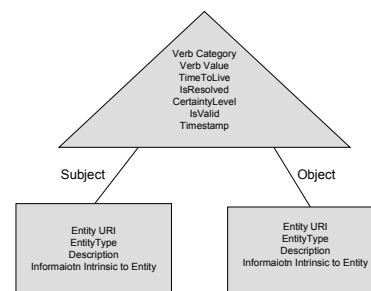


Figure 4

Sets of relationships can be used to show more aggregated context. For example, we will have individual relationships between documents and their users (Person Is_Using Document). To determine all documents in use by a person, we can query the context model for all relationships where the subject identity is set to the person’s URI, the linking verb set to “Is_Using,” and the type of the object is set to document type. A list of relationships for all the documents currently in use by the person is returned. To determine all persons using a given document, we can query the context model for all relationships with the type of subject set to person type, the linking verb set to “Is_Using,” and the identity of the object set to the document’s URI. A list of relationships for all persons actively using the given document is returned. With this flat hierarchy, no duplicated information is needed, nor is any conditional tree processing needed.

4.2 Atira Framework

We’ve revisited our framework, focusing on the bottom three layers. Figure 5 shows the updated PAUR Pyramid conceptual design. We’ve enhanced the semantics of the messages between layers and have revamped the roles played by components in the bottom two layers. We’ve also augmented the capabilities in the understanding layer to include support for initiating personalized behavior based on

changes to the underlying context, involving specific entities or entity types.

Sensors Report What They Know

Sensors now report only what they truly know, as suggested by insight 1 above. We currently have four types of sensors enabled in our environment, including those that detect entities, pressure (i.e. touch), motion, and user messages.¹ Messages sent by these sensors simply identify the sensor and pass information that the sensor is capable of detecting, e.g., the data in an RFID tag.

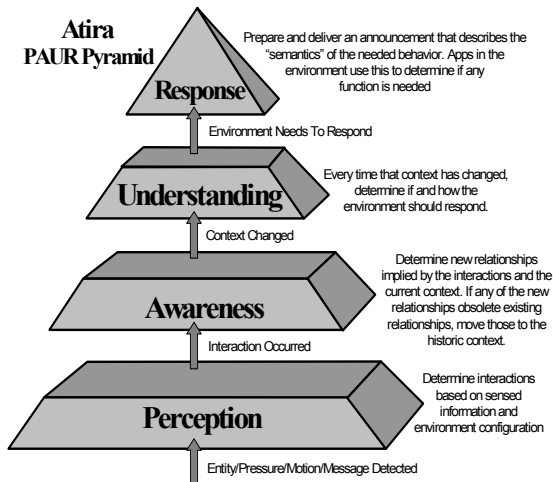


Figure 5

Perceptors Identify Interactions

Perceptors receive messages from sensors based on the class of sensor. They use the name of the sensor included in the message to access the sensor's deployment characteristics from the sensor model. Deployment characteristics include location where the sensor can sense, containers it "watches," and devices it may control. Perceptors use the information provided by the sensor message along with the deployment information to determine interactions like: "a person has just appeared in an office" or "a document has just moved to a new location." The perceptor posts a message reporting the new interaction to the message space.

Monitors Determine Relationships

Monitors listen for messages that describe new interactions and so may influence the current set of relationships known to the environment. All monitors are responsible for managing new relationships and cleansing the context of relationships that no longer

hold. Additionally, monitors provide an interesting level of abstraction in which to investigate ambiguity (identified by insight 3 above) caused by non-perfect sensing, and mechanisms to resolve or tolerate it. When the context has been updated, the monitors post a message to the message space for each new piece of context. These messages include references to context that was retired as a result of the interaction.

Grokkers Determine Interesting Happenings and Support them

Grokkers listen for messages that describe changes in context and are designed to drive the "responsiveness" portion of the responsive environment. Responsiveness can occur as a result of either newly generated or retired context. The type/occurrence of responsiveness can be personalized to a specific subject entity and a specific object, or based on any (type) of entity.

5 Related Work

During the last few years, researchers have begun to look into the area of context. They've asked: What comprises context? How is it represented? What types of methods provide the best access to it? From this, we've seen the beginnings of a rich set of ideas and papers. Among the earliest, the Context Toolkit[11] advocated a design-time approach to building context-aware applications. Applications are directly "wired" (via callbacks) to use widgets wrapped around sensors or widget aggregators. These may be discovered as the application runs. The context-based middleware[12] proposed by members of the Gaia[13] group define context producers, synthesizers, consumers, and discovery services for building awareness of context directly into applications. This work emphasizes the semantics of context drawing from methods used in the Semantic Web[14], and it explicitly builds high-level context using agent-based technologies. Researchers at Berkeley opted instead for an infrastructure-based approach[15] targeted at run-time support for context-aware applications. They've defined an infrastructure, called ContextFabric[16], for creating and storing context information, and letting applications access context using a specialized query language. This work has been extended recently to provide more semantics to the context by a project called ContextMap[17]. ContextMap provides a uniform tree-based representation of the underlying InfoSpaces supported in ContextFabric for both physical and social context. Researchers at Stanford support application-based context using a lightweight XML database, called Context Memory[18] on top of their EventHeap[19]. There are several other works that attempt to highlight definitions (or the proliferations thereof), issues, etc. These works include [20], [21], [22], and [23].

¹ Note that sensors can be hierarchical. If multiple sensors are added to a window to determine position of touch, as done in [10], a hierarchical sensor could be defined that would be capable of identifying position and pressure, as reported by leaf sensors. In the work cited, the "Knock classification/Position Estimation" software could be augmented with such a sensor.

Our work is very synergistic with several of these efforts and we hope to leverage some of them as we continue with our experimentation. We use the same notions of semantics as proposed by the Gaia context middleware. Our monitors are close in function to their context synthesizers, though we represent the entire context in a separate database, similar to the Context Memory approach. We feel it is important to provide one place for context because new context (relationships) is derived from newly perceived interactions and existing context. Having one place for context helps us better change the nature of the environment as we learn more about emerging work practices.

We distinguish intrinsic and relational context as is done in the ContextMap. However, we feel that a tree-based representation, proposed in ContextMap, suggests a hierarchy among entities. We prefer to use a non-structured, binary relationship model and a separate set of models for our “intrinsic” context. We plan to follow this work as it evolves because it has the potential to grow into a system that would meet our requirements for managing context models.

As for underlying infrastructure... We are currently using a publish/subscribe middleware based on JMS[24]. The EventHeap provides similar capabilities, and it would allow us to move beyond being simply a sensor-enriched environment. Potentially, we could adopt this approach and so be capable of leveraging more interaction devices in a more adaptable manner.

6 Current Status and Future Work

Our top priority is to enable enough of the environment to begin living in it. As part of that, we are trying to expand our document-interaction technologies beyond RFID. We plan to create “documents” that span rendering technologies and use these both electronically and digitally. By doing this, we hope to better understand current and emerging document practices for ubiquitous computing environments.

References

1. Marshall McLuhan, *Understanding Media*, MIT Press Edition, 1994.
2. Peter Lyman, Hal R. Varian, et. al, School of Information Management and Systems at the University of California at Berkeley. <http://www.sims.berkeley.edu/research/projects/how-much-info-2003/exccsum.htm>.
3. Abigail J. Sellen and Richard H. Harper. *The Myth of the Paperless Office*. The MIT Press, 2001.
4. Pierre Wellner. Interacting with Paper on the DigitalDesk, *Communications of the ACM*, July 1993.
5. Walter Johnson, Herbert Jellinek, Leigh Klotz Jr., Ramana Rao, Stuart Card. Bridging the Paper and Electronic Worlds: The Paper User Interface. *Proceedings of the INTERCHI*, April 1993.
6. David L. Hecht. Printed Embedded Data Graphical User Interfaces. *IEEE Computer Magazine*, March 2001.
7. Weiser M. (1991). The computer for the 21st Century. *Scientific American*, September 1991.
8. World Wide Web Consortium. <http://www.w3.org/RDF/>.
9. World Wide Web Consortium. <http://www.w3.org/Addressing/>.
10. Joseph A. Paradiso, et.al. Window Tap Technology. <http://www.media.mit.edu/resenv/Tapper/>
11. Anind K. Dey, Daniel Salber, and Gregory D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction Journal*, 2001, Vol. 16.
12. Anand Ranganathan and Roy H. Campbell. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. *Proceedings of the ACM/IFIP/USENIX International Middleware Conference*. June 2003.
13. Manuel Roman, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell and Klara Narhstedt,. Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*. Oct-Dec 2002.
14. Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*. May 2001. <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>.
15. James I. Hong and James A. Landay. An Infrastructure Approach to Context-Aware Computing. *Human-Computer Interaction Journal*, 2001, Vol. 16.
16. James. I. Hong. The Context Fabric. <http://guir.berkeley.edu/cfabric>.
17. Yang Li, Jason I. Hong, James A. Landay. ContextMap: Modeling Scenes of the Real World for Context-Aware Computing. *Adjunct Proceedings of the 5th International Conference on Ubiquitous Computing*. 2003.
18. Terry Winograd. Architectures for Context. *Human-Computer Interaction Journal*, 2001, Vol. 16.
19. Brad Johanson and Armando Fox. The EventHeap: A Coordination Infrastructure for Interactive Workspaces. *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications*. June 2002.
20. Karen Henriksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling Context Information in Pervasive Computing Systems. *Proceedings of the 1st International Conference, Pervasive 2002*. August 2002.
21. Guanling Chen and David Kotz. Supporting Adaptive Ubiquitous Applications with the SOLAR System. Technical Report TR2001-397, Dept. of Computer Science, Dartmouth College. May 2001.
22. James L. Crowley, Joelle Coutaz, Gaeten Rey, and Patrick Reignier. Perceptual Components for Context Aware Computing. *Proceedings of the 4th International Conference on Ubiquitous Computing*. October 2002.
23. M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *Personal Communications*. August 2001.
24. Java™ Message Service, Sun Microsystems, <http://java.sun.com/products/jms/>.