Space-based middleware for loosely-coupled distributed systems

Charles E. Hughes[a], James T. Burnett[a], J. Michael Moshell[b], Christopher Stapleton[b], Barry Mauer[c]

[a] School of Elec. Eng. and Computer Science, Univ. of Central Florida, Orlando, FL 32816-2362
[b] Digital Media Program, Univ. of Central Florida, Orlando, FL 32816-1990
[c] English Department, Univ. of Central Florida, Orlando, FL 32816-1346

ABSTRACT

We describe two distinct distributed systems, MeasureMe and GeoPresence. As with all distributed software systems, these require support for communication, coordination, task distribution and persistent shared data. In this paper, we show how the tuple spaces paradigm provides the required communication and coordination, and how its use facilitates our achieving scalability, extensibility, data persistency and ease of software component upgrades.

Keywords: Distributed systems, location-aware services, peer-to-peer, middleware, tuple spaces.

1. INTRODUCTION

The systems we describe here, MeasureMe and GeoPresence, attack very different problems. The MeasureMe system provides an infrastructure for shared experience in science centers by connecting stations where visitors perform physical measurement and statistical analysis activities. The GeoPresence system allows a potentially unlimited number of users to share experiences enhanced with spatial locality. What both share in common, besides being distributed systems, is a need for anonymous and scalable communication, with long-term data persistency.

GeoPresence's need for anonymity arises from the desire of people to record experiences without being publicly identified. MeasureMe's need for anonymity stems from the need of public facilities to avoid maintaining uniquely identifiable information about visitors without informed consent. However, in both cases, anonymity is rarely complete; and identity is usually shared on a limited, or optional, basis. For example, I want my children to know that I left them messages; a teacher wants to identify students when activities are carried over into the classroom. Scalability concerns in MeasureMe arise as science centers around the world collaborate for a shared exhibit. The foundations of GeoPresence grow on a universal interest in sharing digital experiences and potentially accommodate worldwide involvement. Persistency in MeasureMe arises from the desire for people to compare their performance against everyone who has ever visited the exhibit, anywhere and at any time. Persistency with GeoPresence is obvious; I want my messages to be available to my grandchildren and even their grandchildren.

What we found so attractive about space-based middleware is that it solves these problems and more, providing us the opportunity to extend our systems in ways we cannot conceive of today, in most cases without making any changes to the existing systems.

2. MEASUREME

The MeasureMe exhibit at the Orlando Science Center is an interactive experience through which visitors measure various aspects of themselves; both physical dimensions and performances doing selected tasks. At each stage, visitors learn how they "measure up" against all others who have ever interacted with this type of measurement station. By measuring various aspects of their own physical beings, visitors are drawn into the science. By being involved in a diverse set of playful activities, the attention of the visitor is kept long enough for their data to become a meaningful part of the statistical mix.

Once visitors have completed at least one measurement station, they can go to the analysis stations at which they can compare their performances on any subset of activities to any group they choose, e.g., their performance on “Hang Time” and the length of their right foot versus all others of their age and gender. One possible goal is to find a combination that makes a person unique. In the process, visitors learn about statistical means, medians, and distributions, and about how easy it is to deceive through statistics, just by a careful choice of the population to whom you compare yourself.

The implementation of the MeasureMe exhibit is carried out by a mixture of physical apparatuses, computer workstations, measurement devices that communicate values from the apparatuses to the computers, and a network that connects all the computers together. Figure 1 depicts one of the apparatuses (Hang Time) and the computer station associated with this activity. Hang Time is an example of a measurement station. Here the station records how long a user is able to hang.

The current exhibit contains eleven measurement stations and six analysis stations. We have designs for seven more measurement stations. Out of sight, there are also servers providing database, web and JavaSpaces services. Before discussing the technologies, we will do a simple case study of how the system is used. We then discuss the technical strategies employed in the context of the actual problem that we were solving.

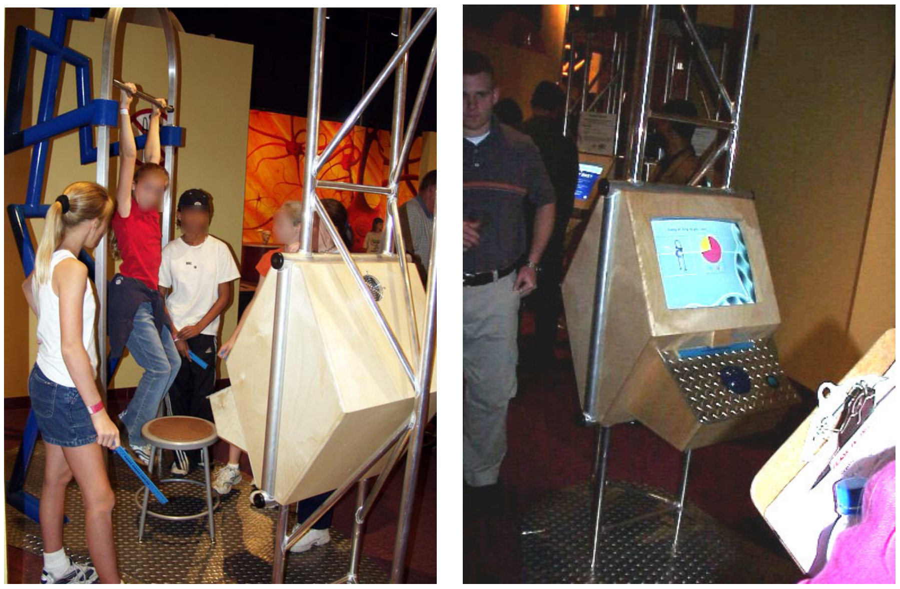


Figure 1: Hang Time

2.1. Case Study – experiencing the exhibit

Each user who enters the MeasureMe exhibit area picks up a ruler from a bin. The ruler has a unique barcode (uniquely identified with a single person for any given day). The user places this ruler in a tray below the screen of any station. Beneath the tray is a concealed barcode scanner that recognizes this individual by the ruler’s code. If the user removes the ruler, a special null barcode (located above the tray) is exposed, which signals the station of the user’s departure. Rulers are recycled daily. This is possible since each user’s performance is uniquely identified by the date and barcode. Although individual identities are not stored, it is possible and desirable to create such an association if activities are to be continued in a classroom setting after the science center visit. We support this, providing teachers with a way to reload their students’ performances at some later date to allow a class to continue the analysis experience. This feature of permanently retaining all data supports educational research. It also provides us with test data that can be used to exercise the system in order to improve its performance or to better understand its deficiencies. For example, if we observed that participants rarely go to more than four data stations we might hypothesize that people want more up-front questions – presently we acquire demographics at the first three stations only. Actually, we have noticed this about young children; they are disappointed when we stop asking them questions about themselves. On the other hand, most older children just want to get to the activity.

Interaction at each measurement station starts with the capture of demographic information. As already noted, this occurs on the first three stations, since we presently gather only age, place of residence and gender. We have steadfastly held to the principle that we should ask at most one question before each activity, and that each question/answer session should typically take between 10 and 30 seconds. An exception to this occurs when we are doing educational studies on groups for which we need to determine such attributes as learning styles.

Once a user has provided the requested demographic data, the measurement activity is begun. The length of time is quick for measurements related to physical characteristics, e.g., foot length, hand span and height. The time for some performance measurements, like Hang Time or Balance, is as long as the visitor can continue to carry out the task. The time at an analysis station is quite variable, depending upon how many different ways the user wishes to measure his or her performance versus that of selected populations of people who have ever participated in the MeasureMe exhibit. Many people use the analysis stations more than once in order to observe incremental changes in the results of various analysis tasks as a consequence of visits to new measurement stations or even revisits to ones on which they would like to improve their performances.

Users should return their rulers to a recycle bin after completing activities for the day – this can take a long time as we set no limits on how long an individual can take, or even if that person leaves and returns later in the day. However, if a user discards one ruler and picks up another, MeasureMe thinks it has a new person since the barcode is the only unique identifier for any given day.

Each measurement station has a unique appearance, designed to reflect its corresponding activity. Figure 2 shows Hand Grip and Flexibility (two performance measurements), arm span (a physical measurement) and a wide shot of the MeasureMe exhibit with a bank of analysis stations in the background. Notice the uniqueness, in color and shape, of the geometric objects at the bottom of each station's name banner. These geometric objects are used to visually identify each activity on the user's screen. We indicate all those that a user has visited on the startup screen. This helps them identify activities that remain to be done. These icons are also used on the analysis stations as selectors for which activities the user wants comparisons to be made.

2.2. An Overview of the Technical Infrastructure

The measurement workstations in the MeasureMe exhibit have several tasks to perform. First, they must know the kind of device with which they are communicating. Our original intention was to have each station use a unique null barcode through which it could identify itself. Unfortunately, there is no way to directly query the barcode; the devices just send data through the keyboard interface whenever the code changes and the steady state in which the machines start is not a change so far as the barcode scanner is concerned. As a consequence, each measurement station reads a simple text file from its own hard drive. This file provides the name of the station. This name may then match up against an entry in "tuple space" that describes the station parameters (we'll talk about spaces later; for now, if this is new to you, think of them as blackboards on which notes can be read or written) [1,2]. If no match exists, then the station assumes there are no changes since the last update, and instead reads a file on its local hard drive. In either instance, the read provides a serialized object – a persistent version of the parameters that bind the station to the type of exhibit it is managing. Each station then writes this serialized parameter object on its local disk, so it can be used in a subsequent boot, if needed.

Once a station learns the identity of its device, it again goes back to the tuple space to find cumulative data associated with each age group that has ever used this device. The actual data is returned from a persistent database whose values are updated each evening. In all cases the data comes as a matrix (100 by 256). The 100 rows represent cumulative data for ages 0 through 99, respectively. The 256 columns reflect how many people of a given age got a score of 0 through 255, respectively. This means that all data are scaled to a range from 0 to 255. Thus an overachiever can get no higher score than 255 (for instance no one can record grip strength higher than 255 pounds per square inch). The data found in entries of the table are used to show new users how they performed relative to their own age groups. Since age is the first demographic data obtained, a user can always be compared against others along this dimension.

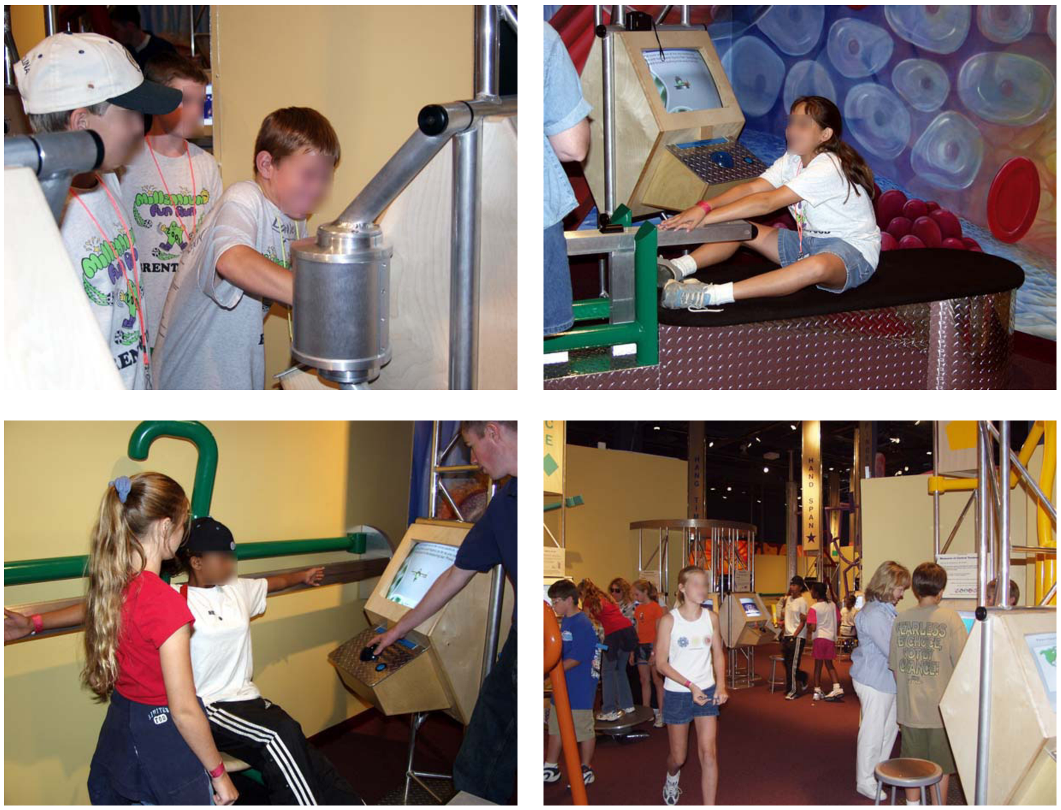


Figure 2: Various Stations

After a measurement station discovers its device, it awaits a user. When a user arrives, the station tries to take (that's a destructive read) an entry from tuple space based on the user's barcode id and today's date. If no such entry exists, one is created inside the station. The station then logs in to the device and listens to values returned. Occasionally, this involves multiple "handshaking" phases. The exact codes sent and received, and even whether the interaction is triggered by the user (pressing a button) or the device (recognizing an event like the user grabbing or releasing the hang time bar) depends upon the specific device. All devices communicate to the workstation computer via a serial connection. Once data has been acquired, a station updates its local database (the 100 by 256 matrix described above), as regards performance by age group. If the user chooses to exit now, the user interface is reset to reflect availability, and an updated entry for this user is written to the tuple space. These entries carry each user's history from station to station and make certain we do not ask any demographics question more than once. These also are used at the end of the day to update the persistent database, so tomorrow's starting values will reflect today's activities.

The MeasureMe software provides error checking and recovery during the visit of a user to a station. The biggest issue is handling the early exit of a user. If this occurs, the station must stop current activity, write the user's entry back to tuple space and reset the user interface to reflect the availability of the station.

Figure 3 depicts the relationship between the exhibitor thread, the primary thread of execution controlling the user's experience, and the devices. As seen here, this thread is isolated from the actual devices; all its communication is via a device proxy. This approach, similar to what we do with the database, allows changes to be made in the device protocol without those changes propagating to other parts of the system. It also allows us to replace the real devices with software simulations in order to debug off-line.

To see how this works, we can start at the SerialConnection. When it gets data from the physical interface, it informs the Device object of this via a callback. After manipulating the data passed to `callback`, the `callback` method on

`Device` calls `propertyChange` on the associated `DeviceProxy`. Every physical device has a `Device` class and a `DeviceProxy` class. The client application always interacts with the `DeviceProxy`, not the `Device`. The proxy defines two methods of interest: `waitForDeviceMeasure` and `measure`. `waitForDeviceMeasure` is a method that blocks until the activity is initiated. For instance, calling `waitForDeviceMeasure` on Hang Time will block until the user starts hanging. In this way, the GUI can call this method, then when it's done display the hanging animation. The `measure` method returns the results. In Foot Length, it simply returns the last value that was sent to it. In Balance, it returns the difference of the beginning time and the end time.

The above focused on what happens at the measurement stations. The analysis stations are a bit less complex, in that they do not need to communicate with any physical devices. However, they do have some added complexity in their user interface and computational demands – users have more choices of what to do and these stations provide more statistical analysis. In general, though, analysis stations are not much different than measurement stations – they must manage the entrance and exit of users; they must get an entry that describes the current user (demographics and activities); and they must employ each user's description to tailor an interface that reflects this user's "trail" of activities. Analysis stations, however, do not update entries, as no new measurements are taken here. Thus, the visitor tuple written from an analysis station is exactly the one it took when the user entered the station.

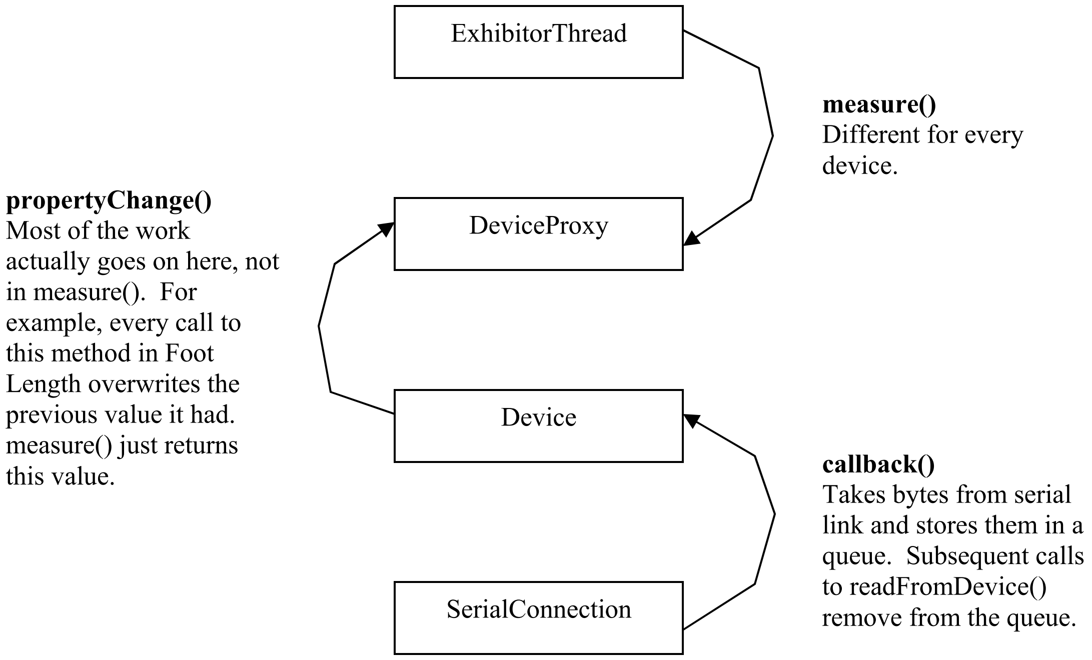

ExhibitorThread
measure()
Different for every device.
DeviceProxy
propertyChange()
Most of the work actually goes on here, not in measure(). For example, every call to this method in Foot Length overwrites the previous value it had. measure() just returns this value.
Device
callback()
Takes bytes from serial link and stores them in a queue. Subsequent calls to readFromDevice() remove from the queue.
SerialConnection

Figure 3. Flow of activity at a measurement station

3. GEOPRESENCE

GeoPresence is a very different application than MeasureMe, but it shares the common need for reliable communication, coordination and persistent storage. It also shares the requirements that communication be

- anonymous – senders and receivers don't know who is "on the other end".
- asynchronous – senders and receivers do not have to be present at the same time.
- content-based – a receiver finds data based of content, only receiving data in which it is interested.

The goal is to support the notion of Geospatial Storytelling, the art of crafting a story in which the author not only uses traditional forms of storytelling but also a relation to physical place. By tying a story together with a geographical place authors gain an additional tool through which they can relate to their audiences. For example, a picture of one's grandparents might be associated with their family farm; a wedding video and a list of attendees might be tied to the place where the marriage occurred; and a neighborhood announcement might be placed at the location of a community center. Using interface software to the GeoPresence system allows people to tell a story about whatever inspires or moves them at any place, and then deposit this story in a persistent space, indexed by location, genre, author, time period, etc.

At this time we have developed two methods for authoring and retrieving stories. The first is through a map-based interface running as a web application. The second, and preferred mode, is to operate at the site of the story, using a location-aware device.

Mapster is our current example of a map-based interface. Sample screens can be seen in figure 4. On the top left is a map that associates landmarks with content and on the top right is the content at the landmark labeled "Rose Garden." Below are contents for landmarks North Gate and South Gate.

In the "real world" mode, we use a portable with an attached Global Positioning System (GPS) and a wireless LAN adapter. This could work equally well with any Personal Digital Assistant (PDA) or cellular phone, so long as it is location-aware and has wireless Internet access. Of course, using appliances like these places limits on the kinds of media that can be supported and the quality of such delivery, but these limitations will quickly pass.

We developed this system with the goal of providing a tool that can be used to preserve culture and history, whether personal or institutional. The actual project for which GeoPresence is currently used is called Earth Echoes. In effect, the Earth Echoes project allows the general public to "drop" stories on the earth, which then "echoes" these stories back to people who pass by this place, today, tomorrow, next week or even a hundred years from now. Early uses of this software have included collection of stories from Eatonville Florida, America's oldest incorporated African American community, Winter Park, another Florida town founded at the same time as Eatonville, the Leu Gardens, an Orlando horticultural park, the trails to Mount LeConte in the Great Smoky Mountain National Park, and the historical buildings and cultural information along the Lynx Lymmo bus route in downtown Orlando.

3.1. Case Study – Leu Gardens

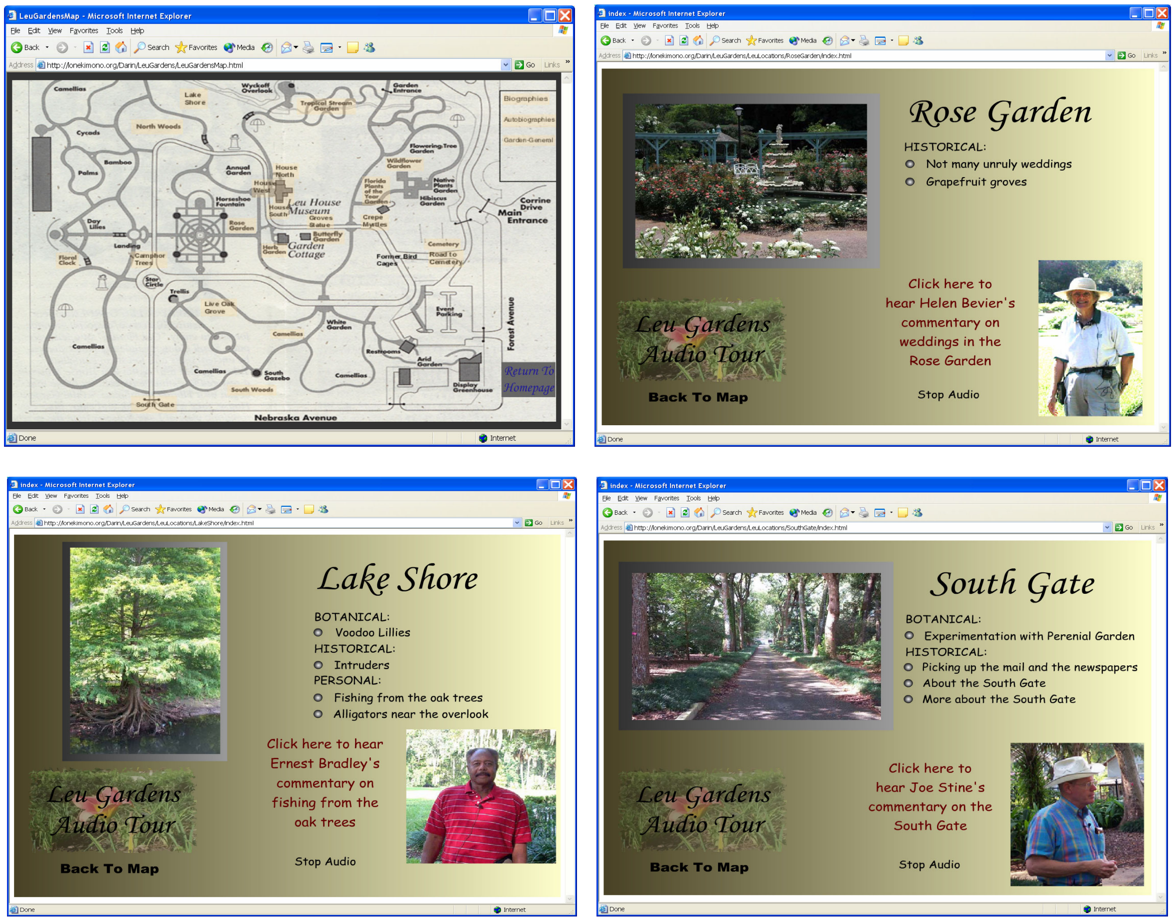
The Leu Gardens property was first settled in the pioneer days when Orlando was a small outpost. It was a private residence for four generations before becoming a public garden. We chose this location so we could demonstrate that the GeoPresence technology could serve both the needs of the local community and the needs of visitors. Our goal was to preserve three kinds of stories: historical, biographical, and botanical.

The map in figure 4 has labels (active hyperlinks) at each site where those we interviewed provided us with interesting stories. To gather information we interviewed four people, ranging from a tour guide/botanist to a person who grew up on the property before it became a public facility. Each provided a different perspective. Using Mapster, one clicks on a landmark to get its content. You then may choose any of the stories, observing appropriate pictures (here the speakers) and listening to audio. The software does not limit one to audio; video clips, flash animations, text, etc. may be used, but audio seemed the most appropriate medium for this particular experience.

The material presented here in a web browser may also be acquired on-site, using a portable or PDA with GPS. The goal is to make this available through any appliance, so long as it is location-aware, has wireless access and supports a competent media player. In the on-site presentation, what the user hears is determined not only by GPS location but also by the individual's preferences. Thus, if I am primarily interested in botanical information, those stories matching this genre would be given preference in my personal presentation. Others who are interested in history would have the stories presented using a different priority, appropriate to their preferences.

3.2. An Overview of the Technical Infrastructure

GeoPresence software provides a peer-to-peer network allowing individuals to share stories with selected groups of people. At present, we are using a Gnutella style network to share data. All data is in the form of XML documents, with tuple spaces used to maintain searchable criteria for the selection of data.


LeuGardensMap - Microsoft Internet Explorer
http://lonekimono.org/Darin/LeuGardens/LeuGardensMap.html
Biographies
Autobiographies
Garden General
Return To Homepage
Nebraska Avenue
Main Entrance
Forest Avenue

Rose Garden

HISTORICAL:

- Not many unruly weddings
- Grapefruit groves

Click here to hear Helen Bevier's commentary on weddings in the Rose Garden

Stop Audio

Leu Gardens Audio Tour

Back To Map

Lake Shore

BOTANICAL:

- Voodoo Lillies

HISTORICAL:

- Intruders

PERSONAL:

- Fishing from the oak trees
- Alligators near the overlook

Click here to hear Ernest Bradley's commentary on fishing from the oak trees

Stop Audio

Leu Gardens Audio Tour

Back To Map

South Gate

BOTANICAL:

- Experimentation with Perenial Garden

HISTORICAL:

- Picking up the mail and the newspapers
- About the South Gate
- More about the South Gate

Click here to hear Joe Stine's commentary on the South Gate

Stop Audio

Leu Gardens Audio Tour

Back To Map

Figure 4: Mapster Screens – stories associated with Leu Gardens

Access to stories, whether from a web interface or an automated PDA interface, is based on location, genre and rights. Location matching is rarely an exact match – in reality, choices may be biased not only by where you are, but also by the speed and direction at which you are traveling through a region. Genre filtering is a weighted process, accounting for an individual's many preferences and moods. Access rights involve both group and individual privileges, and require encryption/decryption to ensure privacy, where desired.

While we do have several prototypes of the GeoPresence infrastructure built, this is a work in progress. For instance, one prototype (the simplest) just uses location to select content; another uses location, speed, direction and time-of-day; yet another is more biased to genre. We are constantly fine-tuning and adding access features, a process we expect to be working on for several years.

4. TUPLE SPACE: THE KEY TECHNOLOGY

Since the MeasureMe exhibit involves measurement, analysis, database server, JavaSpaces server, web server and development/maintenance stations, there is the obvious need for communication and coordination. Similarly, GeoPresence's support for the sharing of data by a large (potentially massive) community of peers has the same requirements, but on a much larger scale. Each uses a strategy based on the tuple space concept first put forth in 1982 [1,2], and more recently popularized by Sun Microsystems in its JavaSpaces implementation [3], by IBM in its TSpaces implementation [4] and by GigaSpaces Technologies Ltd. in its GigaSpaces implementation [5].

The essential idea behind a tuple space is that of a blackboard on which entries can be written, read (copied) or even destructively taken (copied and erased). You can think of the tuple space as a place where notes can be left in order to share information or to pass around tasks. The receiver of a note looks for one that matches its needs. This approach has many advantages over traditional means. First, it does not require that a message receiver be listening or even present at the time the message is sent; communications can be picked up at any time. Second, it does not fix a task to be done by any particular worker; any appropriate worker can pick a message off the board and work on that activity. Third, it can be used as a coordination mechanism; if an appropriate message is not yet on the board, a worker can wait until one is placed there before carrying out its next task. Fourth, communication is content-based; the receiver looks for matches to some pattern that describes the kinds of messages that interest that receiver. Each of these features contributes to the properties of scalability and unfettered system evolution so necessary in modern distributed systems.

Of course, there are some interesting issues raised when a worker process tries to find a match, and no such match exists. In some applications, the worker should block (become idle) until a match arrives. This is very convenient if the worker is a specialist, and can only be productive with the right type of task. However, in other applications, a failure to match actually provides us with information on which we can proceed. For instance, in MeasureMe, the absence of a descriptor for some individual means that this is the first station visited by this person. Under that circumstance, the station should create a new entry, populate it with data based on the user's responses and activities, and then place that entry on the board when the user leaves. In the case of GeoPresence, the absence of a match in one cluster of machines sharing information in a single tuple space means that the search for matching content may proceed outward to other clusters. The recursive nature of our clustering, based on spaces have handles to other spaces, makes this search mechanism implementable by a very simple recursive algorithm. Scalability of the algorithm is not a challenging issue.

Another interesting case in a tuple space occurs if more than one entry matches a worker's request. What entry do you copy to this user? In the tuple space paradigm, the answer is any one of the matches; it's a random choice made by the system. Thus, tuple space adds a bit of non-determinism, in that we cannot predict what will happen when multiple matches are possible. This is okay in most cases; a worker is just looking for something appropriate to do. However, this can be very annoying if a worker wants to copy but not remove all matching entries. The purist way to do this is to remove every match until none are left, look at what you got, and then return them all. JavaSpaces, the tuple space implementation from Sun, is rather (but not totally) pure, so this is how we do it. Other implementations of tuple space, like IBM's TSpaces, extend tuple spaces operations to include one that can scan for and return all matches. (Actually, there's a subtle way to do this even in JavaSpaces via an operation associated with an AdminIterator administrative object, but this is not part of the core specifications.)

All the above requires polite behavior. If only one worker should consume a given kind of message at a time, we have to provide some mechanism to enforce that rule. Thus, any implementation of the tuple spaces concept must handle such race conditions. JavaSpaces does so, and thus MeasureMe and GeoPresence do not need to concern themselves with this possibility, so long as we do things right. This means that we must merely take (copy and remove) a tuple if only one worker should have access to this information at any given time. Reading, rather than taking, will allow multiple copies to be made. In general, MeasureMe uses 'takes' to gain exclusive control whereas GeoPresence almost always uses 'reads' to allow many copies of an entry.

In any complex system, failures are inevitable – they must be planned for and handled gently. For instance, if a worker is to take information from the board, carry out some activities using this, and then return an updated entry, we have to be concerned about the possibility that this worker quits halfway through the task. If this should happen, we want the original entry to reappear on the board, just as if nothing ever happened.

Graceful recovery is a common issue in databases and so the solution used comes from that field in the form of a "transaction" – a collection of activities that appears atomic to all observers. A transaction either completes in its entirety, or appears to have never taken place.

The way this model is used in MeasureMe can best be illustrated by what happens when a user arrives at a station. Again, the user's ruler uniquely identifies that individual (through the barcode). The station looks on the blackboard (tuple space) for an entry that matches the code. If it finds none, then this is a new user. If it finds one, then this is an old user for whom we have already collected some data. Under this circumstance, the station will take the tuple, so at most one copy associated with any unique barcode can exist at any given time.

Once a station has taken or created an entry for a given individual, it uses the data found there, updates it if appropriate and returns the data, once the user has left the station. As pointed out above, there is the possibility that a station will fail while it has a user's entry. This is where transactions play an important role, in that a failure just means that the data collected on that one station may be lost, but all data collected prior to this event will be retained as a result of the transaction's failure leading to a reappearance of the original entry in the tuple space.

Transactions are normally supported through locks, checkpoints and rollbacks. However, RUPLE [6], RogueWave tuple spaces implementation, does not support transactions through such mechanisms; rather it handles conflicts by having all database access processed through a single thread that handles one transaction at a time. Their focus is on XML-only tuples, with access provided by XQL queries. The advantage here is that complex queries, rather than simple associative matching, is supported, and query processing can be done in a rapid fashion due to the absence of locks. Whether or not this is a good tradeoff for the absence of multiple threads is still an open question, but our intention is to experiment with this system to see how well it performs in our context.

4.1. Software Upgrades

While our current versions use tuple spaces primarily for communication and coordination, software component upgrades can also be provided by this paradigm. For example, when a node in MeasureMe acquires its parameter data, it can also check for the existence of an upgrade, in the form of new "jars." These are then copied locally, providing new versions of software components. Such a technique does away with the annoyance and the attendant inconsistencies of having to manually install upgrades. Moreover, it does this within the existing framework that requires all communication to be mediated by tuple spaces.

5. CONCLUSIONS AND FUTURE DIRECTIONS

In both of the projects just discussed, we clearly do not want to lose currently collected data through obsolescence. We also want to be able to handle a potentially very large number of participants. Thus, we require a design and a supporting infrastructure that can evolve gracefully, both in size and capabilities.

The tuple space approach, in general, and its various Java implementations, in particular, provides a natural path to expanding MeasureMe to other science centers and to allowing GeoPresence clients access to enormous data resources, organized into clusters supported by a hierarchy of tuple spaces. This is possible since any JavaSpace (TSpace) can contain handles to any number of other JavaSpaces (TSpaces). Thus, using this paradigm, data gathered at one science center (associated with one GeoPresence cluster) can be shared with an unlimited number of other centers (GeoPresence clusters). Scaling the solution in this manner is a low effort, natural way to upgrade and evolve the capabilities of any loosely-coupled distributed application, as exemplified by MeasureMe and GeoPresence.

GeoPresence uses TSpaces rather than JavaSpaces for the simple reason that TSpaces supports XML querying. An alternative space system supporting XML is RUPLE, discussed above. This additional capability allows us to forgo the back-end databases used in MeasureMe and GeoPresence, storing persistent data in tuple spaces (MeasureMe treats the space as a semi-persistent store that is refreshed on a daily basis; GeoPresence uses spaces for dynamically changing content and indices).

Spaces have been seen as a natural mechanism for agents. In particular, agents can use the space to observe behaviors, detect circumstances under which their particular expertise is appropriate, and communicate with each other. We employ such agents in MeasureMe to support evaluation and plan to use this in GeoPresence to build profiles and improve query performance.

One potential problem faced in a tuple spaces implementation is how one reasons about the behavior of such systems. Of course, reasoning about complex interactions in distributed systems is a hard problem anyway, made especially complex by the fact that communication is unreliable and events may be observed by many participants, each having its own, potentially imperfect, view. Traditional reasoning techniques based on a single perfect observer serializing observable events do not allow us to formally study some important properties of systems like MeasureMe and GeoPresence. These properties include acceptable behavior in the presence of parallel events some of which may be missed by one or more observers (cooperating nodes). A proposed solution to this is "view-centric" reasoning as described in Smith et al. [7] One of our current efforts and future goals is to use this formalism to reason about the systems such as those described here, and space-based systems in general.

5. ACKNOWLEDGEMENTS

This work was supported in part by the Orlando Science Center, the UCF Office of Research, and the National Science Foundation under grant EIA 9986051. MeasureMe software was developed by Dean Reed and later maintained by Mathew Lowery, both UCF Computer Science undergraduate students at the time of this activity. Shaun Murphy, a Computer Engineering undergraduate, developed software interfaces to measurement devices and hardware interfaces created by volunteers and employees of the Orlando Science Center, under the direction of Eileen Smith, Director of Exhibits. Leu Garden content was developed with the assistance of students in UCF's English Department, most notably Tyler Fischer and Alex Katsaros. Web presentation of the Leu Gardens and other Earth Echoes content was developed by Darin Hughes, an Information Technology student at UCF.

6. REFERENCES

1. N. Carriero and D. Gelernter, "Coordination Languages and their Significance," *Communications of the ACM* **35**(2), February 1992, pp. 97-107.
2. N. Carriero and D. Gelernter, "A Computational Model of Everything," *Communications of the ACM* **44**(11), November 2001, pp. 77-81.
3. E. Freeman, S. Hupfer and K. Arnold, *JavaSpaces: Principles, Patterns, and Practice*, Addison-Wesley, 1999.
4. P. Wyckoff, S. W. McLaughry, T. J. Lehman and D. A. Ford, "TSpaces." *IBM Systems Journal* **37**(3):pp. 454-474, 1998.
5. A. Kariv, *GigaSpaces Cluster White Paper*, [Online] available at http://www.gigaspaces.com/download/GSClusterWhitePaper.pdf, GigaSpaces Technologies, Ltd., January 2002.
6. Rogue Wave, *RUPLE: A Loosely Coupled Architecture Ideal for the Internet*, [Online] Available at http://www.roguewave.com/developer/tac/ruple/Ruple.pdf, Rogue Wave Software, May 2002.
7. M. L. Smith, R. J. Parsons and C. E. Hughes, "View-centric reasoning in modern computing systems," *3rd International Conference on Internet Computing (IC '02)*, Las Vegas, Nevada, June 24-27, 2002.