



AORTA

E911 dials for help! - "Position Location" Today and Beyond

The year is 1997. The Wireless industry is buzzing with a thing called "position location". One year earlier, the FCC, via a series of orders, had mandated all wireless carriers to provide automatic location identification (ALI) as part of phase II E911 (Enhanced 911 for emergency services) implementation starting Oct 1, 2001. Several new technology start-ups like SnapTrack, US Wireless, KSI, SigmaOne, Trueposition, etc., are focusing on this burgeoning area, and traditional telecom equipment providers like Lucent, Nortel, Nokia, etc., are scrambling to come up with products for a market that seems so imminent.

Fast-forward 4 years.

We are in 2001, the deadline is a mere couple of months away and in all probability the majority of the carriers are nowhere near providing any position location capability of the handset in a reliable and consistent manner. This column discusses the position location mandate, the solutions, and the missing pieces of the puzzle.

In the last four years, the FCC has adjusted the E911 requirements to better suit the reality of position location solutions. These solutions can be broadly divided into two main categories: network based and handset based. Techniques such as Angle of Arrival (AOA), Time Difference of Arrival (TDOA), and Multipath fingerprinting (MPF) are the commonly proposed network-based solu-

tions, while GPS or network-assisted GPS are the primary handset based solutions. Both solutions have their advantages and disadvantages (see table on page 2). First, let's review these techniques briefly.

A **TDOA** system works by stationing location receivers at three sites, with each site having an accurate timing source. When a signal is transmitted from a mobile terminal, it propagates at approximately 300m/μs to each antenna site. When the cell-site receives the signal, it is time stamped. By examining the differences in time stamps, TDOA can determine a mobile terminal's location by computing intersecting hyperbolic lines. To work correctly, the base stations must be time-synchronized to better than 100ns.

Also called Direction of Arrival, **AOA** has several versions. Small-aperture direction finding is the most common version and requires a complex antenna array at two or more cell site locations. The array consists of 4 to 12 antennas in a horizontal line. The antennas work together to determine the angle (relative to the cell site) from which a cellular call originated. When at least two sites can determine the angles of arrival of a given call, the caller's location can be determined from the point of intersection of projected lines drawn out from the

cell sites.

Multipath is a phenomenon of the wireless RF signal bouncing off solid objects like buildings, towers, and the like. While multipath deteriorates TDOA and AOA's accuracy in locating handsets, this technique utilizes the multipath characteristics (measuring the phase, the timing, and the amplitude path of all the RF signals from a single caller) of a RF environment to locate a handset. Multipath characteristic patterns within a block are analyzed and stored like fingerprints in the database. When a handset transmits its RF waveforms, algorithms try to match these RF characteristics to the fingerprints in a database, thus identifying the block where the call originated.

In a majority of the implementations, these network-based techniques are used together to augment performance.

The most commonly considered handset-centric option is the **Global Positioning System (GPS)**. It takes advantage of the multi-billion-dollar investment the U.S. government has made to establish a satellite infrastructure for location determination. For several years, 24 satellites have been operational, and they have

Inside this issue:

E911 dials for help! - "Position Location" Today and Beyond	1
Packet Shaping/Traffic Shaping Protocols (Part 1)	5
Designing an Effective User Interface for In-motion Computing	6
Using IBM's Reusable Dialog Components to Facilitate the Creation of VoiceXML applications	10

ALI	Pros	Cons
TDOA	<ul style="list-style-type: none"> Provides location for all handsets Can utilize the RF information to provide network design and monitoring services Indoor coverage is similar to indoor RF coverage 	<ul style="list-style-type: none"> Requires extremely precise time-synchronization. Requires TDOA systems at 80-100% of the cell sites, thus increasing the overall cost of the system. Requires at least three cell sites for accurate location determination. Multipath propagation effect (problem resulting from bouncing wireless signals) deteriorates the accuracy of the system. This technique is only able to track location at the start of the call. For CDMA systems, due to near-far problem, the signal-to-noise ratio (SNR) at the neighboring base stations decreases and leads to inaccuracies in position location estimation.
AOA	<ul style="list-style-type: none"> Provides location for all handsets Antennas could be used to improve capacity, reduce interference, and better system performance Indoor coverage similar to indoor RF coverage Can track the call in progress 	<ul style="list-style-type: none"> Requires at least two cell sites for accurate position location determination Requires AOA systems at about 100% of the cell sites thus increasing overall cost of the system Requires highly expensive antenna beams that need to be kept in a calibrated state all the time Since AOA tracks on the voice channel, additional processing is required to query MSC (Mobile Switching Center) to get the information about the user (MIN) For CDMA systems, suffers from the same near-far problem, as discussed in the TDOA section No privacy, carrier can track the user all the time (when the phone is on)
MPF	<ul style="list-style-type: none"> Single cell site can be used to locate and track multipath rays from a handset thus decreasing the capital and operating costs (as compared to TDOA and AOA solutions) RF characteristics analysis could be used for network design and optimization Works very well in urban and dense areas Can track the call in progress if on the same voice channel (no handoffs) 	<ul style="list-style-type: none"> Requires several iterations to build the fingerprint database. The fingerprints would change with weather or construction layout Antenna beams need to be kept calibrated No way of tracking hard handoffs Stationary handsets harder to track (in comparison to moving handsets) Poor performance in rural areas No privacy. Carrier can track the user all the time (when the phone is on)
GPS	<ul style="list-style-type: none"> Doesn't require expensive network modifications and infrastructure. The cost of putting GPS in a phone is declining fast, thus making it a very attractive long-term solution for carriers Much higher accuracy possible Provides privacy. Location could be available on demand 	<ul style="list-style-type: none"> No location of unmodified phones (FCC mandate requires location to be provided for all handsets and there are over 61 million handsets without GPS capability) GPS enabled phones will effect battery consumption Up to 2-3 minutes of initial warm up time Poor location determination in buildings and other shadowed environments Need more memory and software for handset

provided accurate, continuous, worldwide, three-dimensional position and velocity information at no charge. The basis of GPS is triangulation from satellites. To triangulate, a GPS receiver in the handset measures distance using the travel time of radio signals from the satellites. Since satellites already know their location, a precise location of the GPS receiver can be computed. Mathematically, four satellites are needed to determine exact position, but using some error correction and digital signal processing techniques, the number of satellites required for accurate measurements can be reduced to even one (3 satellites are required initially, 1-2 thereafter).

The network-driven GPS method places a minimal GPS front end in the handset and lets the wireless infrastructure equipment handle all the calculation and position determination. Autonomous GPS technology is based on complete GPS subsystem in a handset. Wireless carriers and handset vendors could place all location-determination functionality inside the phone, making virtually no changes in wireless infrastructure.

ALI Accuracy standards: The FCC adopted the following revised standards for Phase II location accuracy and reliability:

- *For handset-based solutions:* 50 m for 67% of calls, 150 m for 95% of calls, and
- *For network-based solutions:* 100 m for 67% of calls, 300 m for 95% of calls.

The FCC required wireless carriers to report their plans for implementing E911 Phase II, including the technology they plan to use to provide caller location by Nov, 2000. The top US wireless carriers – Verizon, Cingular, AT&T Wireless, Sprint PCS, and Nextel Communications account for roughly 76 million subscribers. In reviewing their plans and progress made for E911 deployment, apart from Sprint PCS, these carriers are way behind in their implementation schedule. For some, the service might not start until 2003. The issue is very evident at AT&T Wireless. They have filed for a waiver (VoiceStream got FCC waiver last year, Nextel's request is pending, and Alltel and Cingular are expected to file for waivers) as they have concluded that the network-based technology (E-OTD – Enhanced Observed Time Difference) that AT&T has tested isn't meeting FCC ALI accuracy requirements for its TDMA network. AT&T is proposing Mobile-Assisted Network Location System (MNLS) technology for its TDMA networks. The most significant problem facing this technology is its accuracy. AT&T gave an expected MNLS accuracy estimate of 250 meters for 67% of the calls and 750 meters for 95% of the calls, which falls short of FCC requirements. According to the FCC definition, MNLS is a network solution because it does not require modifications to legacy handsets and thus, it is subjected to the less stringent accuracy requirements of 100 meters for 67% of the calls and 300 meters for 95% of the calls. So even though the solution is less expensive and has some nice features, it probably won't make the grade.

The position location industry in US is very complex. Since there is such diversity in wireless standards (AMPS, TDMA, CDMA, GSM, GPRS, iDEN), wireless carriers have been struggling for the past four years to come up with a strategy that will suit both short-term needs and long-term goals. It is pretty clear that in the next 5 years or so, GPS-based solutions will become the norm for the industry, but it is the short-term compliance that is causing the carriers to split hairs. The majority of the carriers are being forced to consider a dual-implementation strategy: network-based solutions for their antique AMPS and second-generation networks, and GPS-based solutions for

GSM and next generation networks. For GPS-based solutions, since it is a handset-based solution, the existing handsets in the market need to be recycled. For the existing handsets, the only method of location is by using network-based solutions, which means infrastructure costs in millions, if not billions.

Another conundrum is recovery costs. Carriers want to see their investments recovered as quickly as possible; they would ideally like the state to fund the rollout. There are still several states, which are unclear on cost-recovery schemes. NENA (National Emergency Number Association), APCO (Association of Public-Safety Communications Officials International), and other public interest groups have been arguing on behalf of consumers for a speedy implementation of position location solutions so that they can start providing E911 services – the domino that was supposed to enthruse the industry with all sorts of applications and services.

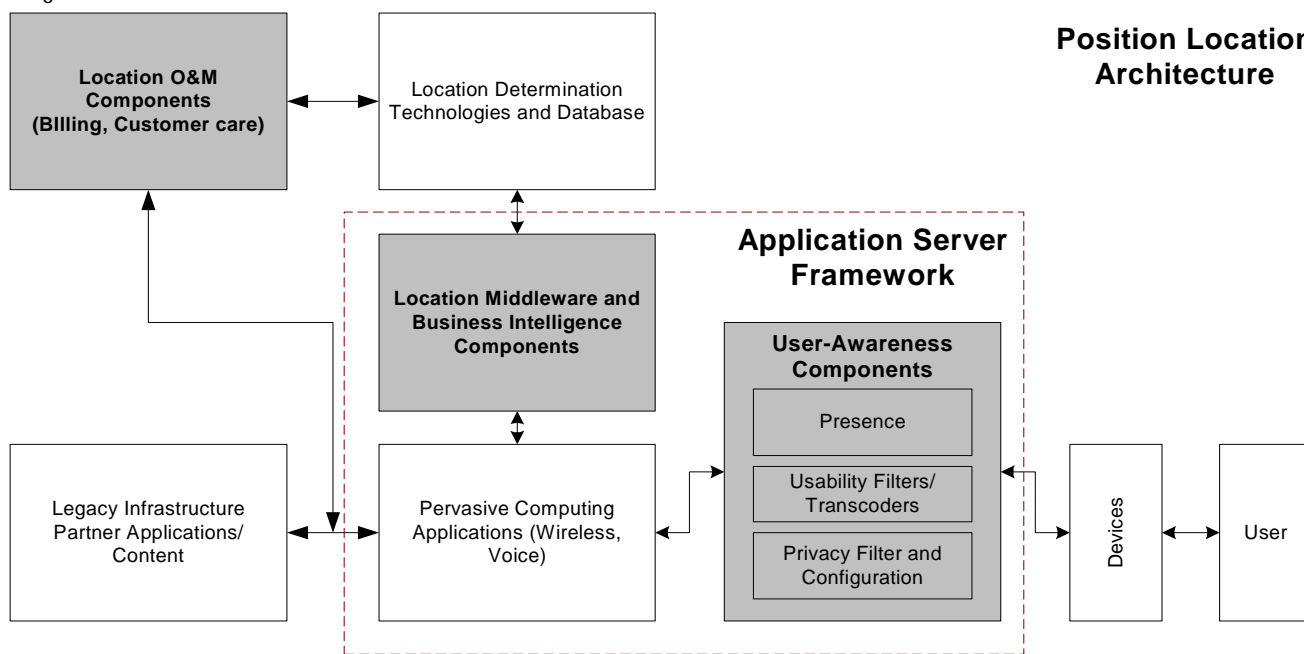
For handset-based solutions, handset manufacturers are leery of production capacity risks. Phone manufacturers are asking customers (carriers) to prioritize but nothing has been forthcoming, according to Nokia. Nokia believes that it would take two years from an order date to get to 50% activation of ALI-capable handsets for a single carrier and for that to happen, GPS needs to be available into the lowest-end handsets to meet the numbers. Also, since accuracy requirements are statistical, there is no general agreement amongst carriers regarding accuracy of the various technologies. The solution's accuracy varies depending on the topography – the accuracy of a solution is quite different in the fields and plains of Topeka, Kansas than it is in the urban canyons of Manhattan, NY.

There are also issues regarding interoperability. There is a need to establish a global forum to address the complexity and multiplicity of current solutions and market situation. With this purpose in mind, Motorola, Nokia and Ericsson established the Location Interoperability Forum (LIF – www.locationforum.org) in Sept 2000. The forum's goal is to define and promote an inter-operable location services solution that is open, simple, and secure. This solution should use appliances and internet-based applications to obtain location information from the wireless networks independent of their air interfaces and positioning methods.

Let's move past the location technology discussion to delivery issues. It's safe to bet that it will be another 2 years before position location-enabled applications and services are available widely in the US (due to implementation and handset delays). Though APCO and NENA have done an admirable job in pushing the issue (30% of 911 calls are from wireless phones according to CTIA), things just haven't moved quickly enough. Once the position location of any mobile device is a given, E911 application infrastructure (PSAPs, ALIs, ANIs, Routers) will be ready to make it work end-to-end. However, that isn't true for non-E911 applications and services. There are three key missing components (see figure on page 4) that need work in the position-location-enabled application architecture:

- *Location middleware and business intelligence components* to integrate location information into applications and to have the capability to do analysis to leverage trends and usage.
- *User-Awareness components* to update the user's presence (*where is the user and what device are configured for use at this very moment*) information, provide privacy filters (controlled by the user) and configuration options, and usability filters and transcoders to customize and render on need/demand.

Position Location Architecture



- *Operations and Management components.* Somebody has to track and bill for all this stuff.

Amongst all these, providing privacy filters (opt-in NOT opt-out policy) is absolutely essential for the industry, otherwise there is going to be a backlash against position location applications and services. We will probably see legislation of some sort to protect the consumer's privacy and unauthorized use of data. These three areas will also be fertile-ground for new, different business models and strategies – who owns and stores this VITAL user information? Companies like Gravitte, LocationNet, and SignalSoft are looking to provide the important components discussed above.

Also, some new application and services companies are springing up to get ready for the position location application boom. They should start spending some time with the carriers. It is pretty naïve to think that carriers will just GIVE you location information or just because the user is using your application (and driving airtime), you can assume a share of airtime revenues from carriers. It just doesn't work that way. Carriers are investing millions of dollars in network and handset rollouts and are going to do their best to SELL location information most appropriately. They should be working with content providers to ENABLE applications and services to be location-sensitive. Increase in such applications will lead to increase in airtime – the REAL source of carrier's revenues.

All this discussion begs the question: "So, what can you do today or in the short-term?" It will certainly be worthwhile to start experimenting with cell-sector-ID based position-location applications and services, especially for applications that don't need to pin-point location with high accuracy and start planning the road-map for future – as position location based revenues are bound to skyrocket, eventually.

There are three exciting columns in this issue of AORTA.

In the column, *Packet Shaping/Traffic Shaping Protocols*, Geoff Varrall from RTT Programmes discusses packet and traffic protocols that are becoming important in computing and communications solutions. Btw, Geoff is one of the best teachers in the wireless industry. He presents two-day seminars in concert with CTIA conferences. I will highly

recommend them.

In the column, *Designing an Effective User Interface for In-motion Computing*, Lisa Davis from Tangis discusses the current interaction methods in in-motion computing and lays down the critical success factors and guiding principles for in-motion user interface. Lisa is an accomplished UI and usability specialist.

Finally, in the column, *Using IBM's Reusable Dialog Components to Facilitate the Creation of VoiceXML Applications*, Lenora Wright from IBM's Voice Server Tools group educates us on the role of RDC's in building effective VoiceXML applications. Lenora is leading the RDC team at IBM.

Enjoy.

My *deepest gratitude* to Geoff, Lisa, and Lenora and their respective organizations for sharing their knowledge and for their promptness. I would also like to thank Molly Fitch at Tangis and Donna Keppen at IBM for their help in putting this issue together.

If you would like to share what you know, please let me know.

Your comments are always welcome.

Best wishes,

Chetan Sharma

Random Quotes

"The industry has to reduce the ratio of horror stories to subscribers."

Mitchell Kertzman, CEO, Liberate Technologies, on broadband (Vortex 2001)

Packet Shaping/Traffic Shaping Protocols (Part 1) – Geoff Varrall

*Reprinted with permission from RTT Programmes.
This article appeared in the monthly "RTT Hot Topics" column in July.*

There has always been a substantial gap between PSTN performance and Internet performance. PSTN end to end delay is typically 35 to 40 milliseconds, internet end to end delay is typically 350 to 400 milliseconds - ten times the time. This does not matter when delivering delay tolerant content. It does matter when delivering time sensitive content - voice or real time image, video and application exchange.

Packet network delay is the delay introduced by packet capture, routing and switching, buffering and queuing. Packet capture is the time taken to process an entire packet before forwarding to a router. Routing and switching delay is the time taken to check the header and routing table, queuing and buffer delay is the time spent by packets waiting in router buffers while routers deal with other packets - a process that typically takes 20 to 30 milliseconds.

One of the reasons for moving to IPV6 is to improve router performance. IPV4 has a variable length header and variable length payload. This is flexible and minimises address overhead but means that a router never quite knows what to expect. IPV6 defines a standard header size of 40 bytes. The 40 bytes are divided down into 8 fields rather than the 14 fields used in IPV4. The defined header size and reduced number of fields means router latency can be reduced and packet shaping/traffic shaping protocols more rigorously applied.

There are three primary packet shaping/traffic shaping protocols - RSVP sits at the edge of the network (for instance in Windows 2000) and defines four levels of service, **high quality/application driven** for applications that can declare their resource requirements - for example MP4 encoded 'declarative content'; **medium quality** where the application identifies the type of traffic flow needed, for example isochronous or non-isochronous but letting the network determine priority; **low quality network driven** - basic latency bounds and a minimum bandwidth guarantee and **best effort**. Multi Protocol Label Switching (MPLS) is then used to break the packet stream into fixed length cells, grouping packets within an IP session into a single flow which can be tagged to optimise router throughput. (A definition of a flow is a sequence of packets treated identically through a 'possibly complex' routing function - the idea is to pass down long lived flows, for example multi-media rich media streams to be switched by hardware). Finally, Diffserv is used to define four levels of service - platinum, gold, silver, bronze - and used as a mechanism for grouping traffic flows sharing similar QoS attributes. MPLS and Diffserv used together will typically reduce queuing delay from 20 to 30 milliseconds to 5 milliseconds.

Delay is however only part of the story. An additional parameter is delay variability (also known as jitter). Delay variability is a consequence of packet loss triggering send again protocols and is therefore related to the provisioning of buffer bandwidth. As traffic has become more asynchronous (increasingly bursty), buffer bandwidth has become increasingly harder to dimension. Essentially, bursty traffic can be accommodated by over provisioning buffer bandwidth - a 2 Mbyte buffer at 60% usage delivers a 4×10^{-2} packet drop rate, a 64 Mbyte buffer reduces the drop rate to 3×10^{-6} . Note that you could use UDP (user datagram protocol) to hide the packet loss (UDP allows packets to drop while TCP/IP requires dropped packets to be sent again) but dropped packets are bad news for differentially encoded rich media.

And there's the snag - packet routed networks promise greater bandwidth efficiency but need to deliver similar dynamic range to existing network topologies, ie to support real time rich media, packet shaping protocols have to deliver an order of magnitude improvement on existing Internet latency performance. The jury is still out as to where this is achievable. Even if it is, bandwidth efficiency will be little better than existing circuit switched networks.

Geoff Varrall is an executive director at RTT. Prior to RTT, Geoff worked at Philips Industries as a Product Manager, and over the next 10 years worked in a number of senior Product Management and Market Management positions within Philips, being finally responsible for a £50 million turnover business. Geoff Varrall regularly presents to audiences worldwide and is co-ordinator of 'The Oxford Programme' held on campus at Oxford University every July. A co-author of the *Mobile Radio Servicing Handbook* (Heinemann Butterworth, UK) and *Data Over Radio*, (Quantum Publishing, Mendocino, USA), Mr Varrall also writes regularly for a number of European trade journals.

RTT Programmes has been specialising in providing an international client base with technology assessment and technology related seminar programmes since 1986. The company's knowledge and experience is principally in mobile terrestrial communications. RTT works closely with members of the Mobile Experts Groups within ETSI (European Telecommunications Standards Institute), and the international academic, scientific and industrial research community. Over the past two years, RTT's research has extended to include the impact of technology convergence across six industries - computers, consumer electronics, IT, wireless, wireline, TV.

Designing an Effective User Interface for In-motion Computing – *Lisa Davis*

A frequent assertion in wearable computer discussions is that WIMP (windows/icons/mouse/pointer) interfaces are inappropriate for wearable computers, primarily because they place unreasonable cognitive and motor skill demands on users who are engaged in the real world. Laptop computers, handheld computers, and PDAs have attempted to address the problems of mobility by providing simplified user interfaces and mobile input devices. Unfortunately, the cost users pay for mobility is limited functionality, tiny screens, and inadequate input devices. This article briefly critiques the interaction methods currently in use, and proposes six criteria for successful in-motion interaction design. It also uses an in-motion user interface developed by Tangis Corporation to illustrate how the proposed design criteria might be fulfilled.

Introduction

It would hardly be controversial to claim that wearable computing needs more appropriate interaction methods than those commonly available on desktop and handheld systems. Researchers have raised this point repeatedly over the past few years, often citing the inadequacy of current methods as motivation for research into alternative approaches. In fact, at the 1998 IEEE Virtual Reality Annual International Symposium, one wearable computer researcher/user made a presentation with the arresting title, ‘WIMP interface considered fatal.’ [1] In this article, I’ll briefly critique the interaction methods currently in use and propose six criteria for successful in-motion interaction design. Where appropriate, I’ll use examples from the in-motion user interface developed by Tangis Corporation to illustrate how these criteria might be met.

Current interaction methods

First I’ll make a quick survey of the interaction methods currently available for in-motion computing, discussing the strengths and weaknesses of each method.

Point-and-click interfaces

WIMP (windows/icons/mouse/pointer) interfaces have dominated desktop computing for the past two decades. They offer two important advantages over the command-line interfaces (e.g., DOS or UNIX) that preceded them: they reduce the learning curve by making user options visible, and they support direct-manipulation interaction. However, standard WIMP interfaces developed for the desktop are fundamentally inappropriate for mobile computing, for a variety of reasons.

Perhaps most importantly, WIMP interfaces assume that human-computer interaction is the user’s primary task. The user is expected to focus his or her attention on the computing task more or less continuously until a task is completed. For in-motion users, interaction with the real world—performing hands-on work, navigating through physical space, interacting with other people, driving a vehicle, etc.—is likely to be more critical than interaction with the computer. In other words, the computer *assists* the user with real-world tasks instead of *being* the task. An in-motion user can’t be expected to focus his or her attention on the computer for long periods of time.

Another problem is that WIMP interfaces rely on pointing as the primary selection method. Unfortunately, pointing is difficult without a supporting physical environment, one that provides stability for the display, the pointing device, and even the user’s own body and limbs. Laptop computers, handheld computers, and PDAs have attempted to address the problem of mobility by providing simplified user interfaces and mobile input devices: pointing sticks, track pads, built-in trackballs, and pens. These devices are all intended to allow users to take their computing devices with them. Unfortunately, the price we pay for mobility is often limited functionality, tiny screens, and unsatisfactory input devices. Moreover, few of these devices are intended for while actually in-motion. It’s difficult to control a pointing stick or to enter handwritten data while walking. Depending on the situation, it might even be dangerous—for example, if the user’s attention were drawn away from a real-world hazard. A QWERTY keyboard might be the ideal input device if it didn’t require a physical surface for stability.

Several attempts have been made to produce full-fledged in-motion pointing devices, none completely successful. For example, the Xybernaut MA IV wearable computer has a built-in pointing device. Unfortunately, it is a rate-control device, which means that the force applied by the user’s finger determines the speed of the pointer’s movement. Even with fine-tuning enhancements such as negative inertia, rate-control devices aren’t easy to operate, especially when walking. Another input device, the Twiddler is a handheld combined chording keyboard and pointer used by many wearable computing enthusiasts. Some users report great proficiency using Twiddlers, but only after considerable practice. Also, the user’s hand size has a large impact on the likelihood of successful pointing. The Thumbelina is a mini-trackball also designed for handheld use. This device is easy to hold and use, but its tiny trackball doesn’t perform particularly well, even when the user is standing still.

With any of the pointing devices currently available, interacting with a typical window-based application is difficult—if not impossible—when walking. The jiggle that inevitably accompanies walking makes it difficult to see screen objects, much less interact with them. The Tangis design team spent several weeks investigating the extent to which different input devices are subject to unintentional pointer movements as a re-

sult of the user's movement through physical space. We even coined a term for this problem: ambient motion. We found it necessary to make a number of modifications in standard WIMP design to make point-and-click interaction effective for in-motion use.

Speech interaction

None of the input devices mentioned so far offers hands-free operation. Speech-based systems, however, could free the user's hands *and* eyes for interaction with the real world. Unfortunately, speech alone doesn't provide a blanket solution for the problem in-motion interaction because it's unsuitable in some circumstances. For example, the non-office world can be full of noise—traffic, construction equipment, manufacturing machinery, aircraft, people, music, etc.—that can interfere with the user's ability to hear or be heard. On the other hand, the world is sometimes very quiet, and speech interfaces can be unusable in those situations, too. In some social contexts, speech commands would probably be deemed intrusive by others, for example, in the lobby of the doctor's office, in the carpool, at the theater, in a restaurant, etc. Speech interaction with a computer, because it is unusual and 'unnatural' might be more distracting and unwelcome to bystanders than cell phone conversations. Even in places where speech interaction probably wouldn't be intrusive, a user's need for privacy might make it unsuitable. For that matter, speech might not be the ideal input method even when there are no environmental or social constraints on it. Imagine, for example, how inefficient it would be to draw a map or mark up a photo using a speech system.

In some circumstances, speech interaction might be the ideal solution—assuming, that is, that the speech system were easy to use and reliable. Relatively inexpensive speech recognition packages are available today; they give users the ability to control virtually their entire computing system with speech commands. However, simply adding a speech recognition layer onto existing applications doesn't make them easy to use. For one thing, direct manipulation interactions don't always have obvious speech counterparts. For example, the user must learn the correct command sequence for deleting the last three words in a sentence. Also, speech systems require significant training—of both the system and the user—to bring errors down to an acceptable level. Speech recognition systems need to learn about the user's speech patterns, and users need to learn the proper way to issue commands.

Of course, speech systems are improving all the time, and natural language systems—in which users talk to their computers as if they were human—promise to make human-computer interaction far more satisfactory. Technology developers have made huge gains in processing power, and language researchers have gained an increasingly sophisticated understanding of human communication. For now, however, natural language systems are best for addressing specific tasks in limited domains. Natural language interaction is unlikely anytime soon to be a practical interaction method for in-motion computing.

Non-command interaction

Post-WIMP or 'fourth-generation' user interfaces combine a variety of non-command interaction methods—for example, natural language, gestures, eye tracking, and/or machine learning. By definition non-command interfaces should have less need input devices of any sort. Although such systems may become widespread within the next decade, today they are still uncommon and largely experimental. Also, they often require expensive or cumbersome equipment in the form of sensors, video cameras, gloves, etc. Such solutions tend, moreover, to be application-specific, because accurate interpretation of the users' behavior requires an in-depth understanding of their goals, tasks, methods, environment, etc. It's unlikely that even a sophisticated non-command system could anticipate *every* user need and provide a natural interaction for it. It seems inevitable the user will sometimes need to direct the computer explicitly, either to perform a task that's beyond the system's experience or to correct a mistaken inference.

There's no single solution

Part of the problem with current computing systems is that no single interaction method is suitable for the wide range of non-desktop contexts that an in-motion user may encounter. For example, standard 2-D pointing might very effective when standing still. After all, for some tasks, direct manipulation is the quickest and easiest input method. Combined with a head-mounted display, 2-D pointing can also be very private. Unfortunately, 2-D pointing can be quite difficult when the user is walking or needs use of his hands. Constraining pointer movement to one dimension is a useful approach to reducing ambient motion.

Speech is also good for in-motion computing. For some tasks, it can be quicker than 1-D or 2-D pointing, especially if the system allows barge-in (i.e., the user can issue commands without waiting for the system to finish speaking). Speech interfaces also allow hands-free and eyes-up interaction. Even a handheld touch screen can be effective when moving, provided that target sizes are adequate. Touch screens provide a very direct form of interaction, and they combine input and output devices into a single unit. Moreover, users may find them more acceptable than head-mounted displays. However, a hand-held touch screen takes the user's eyes further away from the real world than does a see-through head-mounted display. Strap-on or fold-out keyboards are sometimes useful, especially for users that perform extensive alphanumeric data entry. Such users may even be willing to invest the time needed to master a chording keyboard, which requires the use of only one hand.

Interaction design criteria

Tangis Corporation has adapted the traditional WIMP interface for in-motion computing. Our interface design modifies and augments point-and-click interaction in an effort to make it suitable for in-motion computing in a wide variety of contexts. We provide software tools that allow developers to create applications that make use of this in-motion user interface. We've conducted numerous user studies [2] to assure that our

interface design is easy to use with a variety of input devices. The following criteria summarize what we've learned along the way about designing successful in-motion user interface.

One - The user interface must make minimal cognitive demands on the user

'Split attention' is a myth. Substantial research indicates that users have only one 'locus' of conscious attention. [3] In-motion users must shift their attention away from the real world every time they interact with the computer. To help users transition between the virtual and real worlds, the system should allow them interact with the computer in fast, easy, 'chunkable' transactions. The interface should not require long periods of user attention, and it should minimize its demand on the user's short- and long-term memory.

With the Tangis user interface, for example, interaction with the computer is a simple dialog, with the computer taking responsibility for moving the dialog forward whenever possible: the system prompts and presents choices to the user. The user responds. Each screen supports a single user decision. This design is simple enough to allow a quick 'in-and-out-again' interaction style that places minimal cognitive and motor skill demands on the user.

Two - The user interface must support a wide range of in-motion computing tasks

It wouldn't be cost effective to devise a custom interaction method for every application or computer hardware configuration. A general-purpose solution is needed if wearable computing is to become mainstream.

Before mocking up a single interface design, our design team spent several weeks identifying the essential 'building blocks' of human-computer interaction, an effort which we believe has paid off. Our basic user interface design is intended to be highly generic, and we've used it to implement applications for building inspection, aircraft maintenance, first aid, PIM functions, online food delivery, and stock trading.

Three - The user interface must support a variety of interaction methods and devices

No single interaction method will always work for in-motion computing. Users may interact with their systems in many different contexts, each favoring a different input mode. For example, an airplane mechanic, who works with her hands in a noisy outdoor environment, might need a head-mounted display and a belt-mounted input device. A real estate agent, who spends most of his time talking face-to-face with people, might be better off with a small wrist- or belt-mounted touch screen display.

Users should be able to interact with their computers using most common input devices, including 2-D pointing devices (e.g., mouse or trackball), 1-D pointing devices (e.g., scroll wheel and 2 buttons), physical keyboards, pens, touch screens, and speech recognition.



Illustration: touch screen keyboard vs. speech keyboard.

Four - Each interaction method must be highly usable

Most desktop applications provide a single user interface, and most are optimized for 2-D pointing and/or keyboard interaction. By assigning functions to the scroll wheel, they may add limited 1-D interaction. Third-party speech recognition software can be added onto the standard user interface to allow speech interaction. But this add-on approach inevitably produces sub-optimal interaction design for the added interaction methods. Even if you completely redesign a user interface to support multiple input modes, it is a difficult task that's likely to result in mediocre

usability for some or all modes. (I speak from experience as both a designer and a user of such interfaces.) In some situations, it's better to offer the user a variety of user interface designs, each optimized for a particular input method. For example, when we designed the interaction methods for alphanumeric data entry, we concluded that some devices needed their own virtual keyboard. The keyboard for 2-D pointing and touch screen uses a standard QWERTY layout. The speech keyboard we designed is quite different: it displays a phonetic alphabet, word equivalents for punctuation marks, and editing commands. Our preliminary design for a 1-D keyboard presents the letters in a vertical list that's ordered alphabetically.

A variety of auto-fill methods can be used with any type of keyboard to expedite data entry, including string matching, word prediction, and custom vocabularies. If an in-motion system makes several of these available, then an application could select the method(s) most appropriate to the user's task.

Five - Switching from one interaction method to another must be easy

An in-motion user may move from one context to another throughout the course of a day. For example, when the airplane mechanic goes to the parts warehouse, voice interaction may become possible and even preferable to touch screen input. When the real estate agent is en route to his next appointment, his hands and eyes are needed primarily for driving. If a change in context requires a change in interaction method, the interface should adapt as smoothly as possible. Ideally, the system should adapt automatically, making a seamless transition with no conscious effort on the part of the user.

At Tangis, we realized fairly recently that our system can identify the user's current input mode and automatically switch to the interface that's optimized for it. Currently, when the system detects a gross movement of the pointer, it switches to 2-D mode. When the scroll wheel is rotated, it switches to 1D mode.

We have also designed methods for detecting the other input modes for automatic switching. Detecting speech input is easy: the system switches to speech mode when it recognizes a valid speech command. The system can detect the difference between a 2-D pointing device and a touch screen by monitoring the pointer movement: touch screen interaction produces uniformly fast pointer movements as the pointer jumps to the place touched by the user. A 2-D pointing device produces movements that are fast at the beginning then slower as the target is approached. Automatic switching between input modes is a baby step toward non-command interaction. We are optimistic that our system will be able to infer the users' intentions from their actions in other ways. This is another area we've earmarked for future research.

Six - The interface must be easy to learn

Computer users have a huge interest in gaining immediate productivity, and wearable computing has a huge potential for increasing the productivity of in-motion workers. Nonetheless, although users might be willing to invest some up-front time learning to use an interface, it's unrealistic to expect them—or their employers—to commit to extensive training. Many ingenious input devices have languished because they require too much practice to become useful.

Conclusion

This paper proposes several criteria for the design of effective interaction methods for in-motion computing, and uses the Tangis user interface to illustrate how elements of WIMP interaction can be adapted to meet those requirements. Obviously, other designs could satisfy these criteria, and it's likely that the criteria themselves will undergo further refinement and augmentation. Nonetheless, this approach promises to make in-motion computing practical until the next generation interfaces reach maturity.

References

1. Rhodes, B.: WIMP Interface Considered Fatal. In the Proceedings of the 1998 IEEE Virtual Reality Annual International Symposium Workshop on Interfaces for Wearable Computers
2. Davis, L.L.: 2-D Pointing While Walking. In the Extended Abstracts of the 2001 ACM SIGCHI Conference on Human Factors in Computing Systems
3. Raskin, J.: The Humane Interface: New Directions for Designing Interactive Systems

Lisa Louise Davis is a user interface designer and usability specialist for Tangis Corporation. Prior to joining Tangis, she spent seven years as an independent consultant for such companies as Apple Computer, Intel Corporation, Attachmate Corporation, WatchGuard Technologies, and Saltmine Inc. Lisa is the inventor on three U.S. patents held by Apple Computer.

Tangis Corporation is a Seattle-based software company developing solutions for in-motion computing. Today Tangis creates applications for the in-motion workforce—people who need hands-free computing—permitting them to use information resources and corporate data while staying on-site and in-task. They are also developing software technologies that will soon make computers context aware, turning them into intelligent, proactive assistants that understand what people need and when they need it.

Using IBM's Reusable Dialog Components to Facilitate the Creation of VoiceXML Applications

- Lenora Wright

Voice technology is emerging as an application interface for the web and, as a result, existing web applications are being extended to accommodate it. Voice eXtensible Markup Language (VoiceXML) makes development of these voice applications easier, and its acceptance is dependent on how easy developers can create their own VoiceXML applications. The IBM Reusable Dialog Components provide a set of subdialogs and templates that are designed to facilitate the creation and delivery of VoiceXML applications. The IBM Reusable Dialog Components package is built upon the VoiceXML 1.0 specification and will work with any Voice Browser that adheres to this standard. This package also provides developers with an architecture that promotes open standards when creating their own reusable dialogs.

Here is an example of some of the subdialogs and templates included with the IBM package:

Subdialog Name	Description
Confirmation	Prompts the caller with a question. Valid inputs are defined in the built-in boolean grammar and negative.gram. You can choose between two types of confirmation: <ul style="list-style-type: none"> • Active confirmation allows the caller to respond to the question by saying Yes or No. • Silent confirmation requires the caller remain silent to indicate an affirmative response, or say No to indicate a negative response.
E-mail Address	Prompts the caller to say an E-mail address. This subdialog uses the grammar file emailaddress.gram, which enables the caller to say underscore, @, dash, hyphen, dot, or period in addition to the valid inputs listed for the Alpha Numeric subdialog.
Time Info	Prompts the caller to say a time. Valid inputs are defined in the built-in time grammar. You can choose between two types of times: <ul style="list-style-type: none"> • Fully specified times, which require hours, minutes, and either AM or PM. For example, the caller can say 1300 hours. • Partial times, which allow the caller to say a time without saying AM or PM; however, the caller may choose to say a fully specified time.
Template Name	Description
Address	Demonstrates how to collect a caller's address. This template uses the Alpha, Confirmation, Number, Street Type, US Major City, US Postal Code, and US State subdialogs.

Subdialogs are VoiceXML programs that receive parameters and can return values to and from other VoiceXML programs. The IBM subdialogs are building blocks that provide some basic core functionality that might be needed by VoiceXML applications, i.e., obtaining credit card or social security numbers from a caller. By providing these subdialogs, developers with little VoiceXML experience can begin to write basic functions quickly, thereby speeding application development. Templates, on the other hand, are combinations of subdialogs merged together to perform a more complex set of tasks. A template uses most of the constructs of a subdialog including the use of an ECMAScript Object file (discussed later); however, templates do not have parameters for passing.

A basic dialog in VoiceXML consists of a prompt that is used to manage the dialog progression or flow. Since the subdialogs utilize grammars (Java Speech Grammar Format) to denote valid caller responses, the prompt must be clear and concise enough to solicit the proper reply. When the caller's response is not part of the grammar, the subdialogs make use of event elements (`<nomatch>`, `<noinput>`, `<help>`) that assist in obtaining a valid response (self-revealing help).

There are many other constructs in the VoiceXML language that allow you to manipulate the overall dialog flow. However, the overall theme of a dialog seems to be fairly consistent, regardless of the context. Based on this consistency, we derived a general structure that all subdialogs should follow. Below is an outline of VoiceXML code that represents this structure (see code listing on this page):

Each subdialog has an associated ECMAScript Object File that is instantiated by the VoiceXML application. An ECMAScript Object File is an external ECMAScript file that

```

<vxml>
  <form>
    <field type="some grammar">
      <prompt></prompt>

      <nomatch 1></nomatch>
      <nomatch 2></nomatch>

      <noinput 1></noinput>
      <noinput 2></noinput>

      <help 1></help>
      <help 2></help>

      <error>
        <throw event="exit"/>
      </error>

      <filled>
        <return/>
      </filled>
    </field>
  </form>
</vxml>

```

is called using the VoiceXML `<script>` tag. This file contains one defined function along with all default prompts, subdialog specific parameters and self-revealing help. Although this ECMA script code could be embedded in a subdialog, it has been separated out for a voice application developer's ease of use and NLS translation. Each subdialog has its own partial interface (subdialog specific parameters), which makes it easier for the voice application developer to customize the subdialog. Customizations can be made globally or on a per instance basis.

To call the subdialog, first call the ECMAScript Object file, instantiate the object, and then pass it to the subdialog as shown below:

```
<vxml version="1.0">
  <script src="confirmation.es">
    var objConfirm = new Confirmation( );
  </script>

  <form>
    <subdialog name="test"
      src="confirmation.vxml">
      <param name="paramSubdialogObj" expr="objConfirm"/>

      <filled>
        <prompt>
          The answer was <value expr="test.returnConfirmation"/>
        </prompt>
      </filled>
    </subdialog>
  </form>
</vxml>
```

When you are ready to try your applications, we recommend that you test them using IBM WebSphere Voice Server SDK Version 1.5 which provides a complete testing environment through simulation of the telephone voice input and audio output.

Voice technology is rapidly becoming an integral part of computing. To this end, the IBM Reusable Dialog Components provide pre-written VoiceXML code that executes common function and that can be copied and reused throughout applications. Through the creation and distribution of the Reusable Dialog Components, a consistent, standards-based coding model is also provided. Therefore, the subdialogs and templates can make it easier to develop new applications and speed time to market by decreasing application development time.

For more information about IBM's Reusable Dialog Components visit our web site at: ibm.com/software/speech/enterprise

Lenora Wright is a development member of the IBM Voice Server Tools Group in Boca Raton, FL. Lenora has a BS in Computer & Information Science from the University of Florida. She is currently leading the Reusable Dialog Components team.

©Copyright IBM Corporation 2001

IBM Corporation

Department LG9A

8051 Congress Avenue

Boca Raton, FL 33487

Produced in the United States of America

All Rights Reserved

IBM and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries or both.

Other company, product and service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.



Next Month
*Ankur Laroia, Leader
 of Luminant's EAI
 Solutions Practice, in
 a 3 part series,
 introduces us to the
 world of EAI and
 looks at its future.*