Joel Jones jones@cs.ua.edu

My formal training and work in Computer Science has been both broad and deep. I have successfully done research in several areas of computer science, in academic, industrial, and government research settings. This work has resulted in papers, presentations, workshops, and most importantly, working software. In addition to my formal responsibilities, I have always had “side” projects, working either individually or with others. Many of these projects have come from ideas formulated many years before. Below are the details, ending with some “new” ideas—excerpts from my idea journal, ranging from deep to silly, in chronological order.

Training and Employment

University of Illinois, Urbana-Champaign. Computer Science, Ph.D. 2002 .
Arizona State University, Tempe. Computer Science, Master of Science. 1989.
University of Tennessee, Knoxville. Computer Science, Bachelor of Arts.1986.

University of Alabama, Assistant Professor. 2000–Present
Graduate Intern, Hewlett-Packard Laboratories, Palo Alto, CA, Summers 1994 and 1995.
 Compiler optimization for a VLIW machine.
Programmer, U. S. Army Construction Engineering Research Lab, Champaign, IL, Fall 1993–Spring 1995.
 Integrating databases with a GIS (Geographic Information System).
Intern, Apple Computer, Cupertino, CA, Summers 1992 and 1993.
 Code generator for Power PC. Sound drivers for Apple’s first Unix-based system (A/UX).

Representative Results

Joel Jones. “Anonymous Caching: Managing Duplication and Dependencies.” Pattern Languages of Program Design. (2001).
 Software design pattern.

Allen Parrish, Joel Jones, and Brandon Dixon. “Extreme Unit Testing: Ordering Test Cases to Maximize Early Testing.” *Proceedings of XP Universe.* (2001).
 Software engineering testing scheme for agile development methodologies.

Joel Jones and Samuel Kamin. “Annotating Java Class Files with Virtual Registers for Performance.” *Concurrency: Practice and Experience,* V.12, N. 6, (2000): 389–406.
 Compilation technique for Java .class files.

Apple Scripts for Claris Organizer/Palm Desktop
 Work flow software distributed on millions of Mac OS CDs.

“Introduction to HTML,” presentation for UIUC ACM Student Chapter Meeting, Fall, 1993, Urbana, IL.
 Workshop in HTML. N. B. date.

“Logging I/O Activity in a Supercomputing Environment,” poster presented at National Net ‘91, Washington, DC.
 Software for logging networking and file I/O characteristics of supercomputing applications.

Joel Jones. *CDAT — A Tool for the Graphical Display of Data Structures from Static Source Analysis,* Masters Thesis, Arizona State University, 1989.
 Visualization technique for program understanding.

Forward Thinking

6 August 1997 **Mathematica for CS**

What would a symbolic math package for discrete mathematics be like? Type theory? Category theory?

7 August 1997 **Algorithm Animation through Dance**

20 August 1997 **Cord Ivy**

To make office environments look cooler, wrap artificial ivy around power and computer cords.

8 October 1997 **Desktop Devices**

Use PDA like pads connected to USB to desktop computer as input devices, but they also operate externally as PDAs.

6 November 1997 **Omnia research area name**

It's not CSCW (computer supported collaborative work) because it is not necessarily collaborative, unless collaborative doesn't imply collaboration with another person. It's not HCI (human computer interaction) because the interaction may be with something that doesn't fit into the typical computer model. For example, if the system is used to simply file a voice recording, without recognition of what is spoken (the audio equivalent of "ink"). One possible term is human-machine communication, but does this give an incorrect impression if the machine is only used as an intermediary (for example, a voice-controlled telephone)?

19 September 1999 **Distinguishing Collisions from other Interference forms**

In wireless networking, collision detection may involve two sources of interference --- that from other network transmitters and that from non-network transmitters. In wired networking, interference only involves one source of interference --- other network transmitters. There, it is appropriate to do an exponential back off. In a wireless network, it is better if the source of the interference is not from another network transmitter to retransmit immediately, rather than doing an exponential back off. To detect the difference, compare the set of gathered network neighbors against the difference between the received signal and the transmitted signal. If the difference at the transmitter physical address time is one of the neighbor addresses, then do the normal exponential back off.

11 July 2000 **RF Tunneling and Building Design**

Increased use of wireless communication technology presents an interesting constraint on the construction of offices and homes. Some wavelengths should pass through walls, others shouldn't. Therefore, make subareas of a building RF opaque and provide repeaters connecting them. These repeaters should have programmable network band pass filters to allow changeable frequencies to pass through.

22 January 2001 **Voice Recognition Groupware**

Barak Perlmutter's work on distinguishing sources in a noisy signal environment can be used to enhance groupware. Some example applications: voice control of display system, transcripts with speaker information, vote taking, directed note taking (e.g. CRC cards).

Mobile and Personal Ubiquitous Computing

Joel Jones
Department of Computer Science, University of Alabama

We have already seen the beginnings of the vast integration problem before us as our environments become increasingly populated with devices that communicate and with a staggering variety of wireless services. End-users increasingly expect a multi-media experience, even when using simple messaging systems. This raises great challenges with regards to user-interface design and to limited wireless bandwidth. Our research and design processes must change, expanding to include integration across domains with a focus on how the technology can serve the user.

Mobile

The solution to the bandwidth problem will not be simply "design for latency, you can always buy more bandwidth"—unlike wired communications, wireless bandwidth is precious and shared. The solution will involve combinations of technology in operating systems, networking, compilers, databases, and human-interface design.

One vertical slice of the integration problem is providing for the delivery of applications in a distributed, mobile environment with soft real-time constraints. Such a system has many practical uses, including emergency medical response, disaster relief work, and construction. The missing technology is quality of service (QoS) guarantees across the underlying system components (computer languages, compilers, networks, and databases.)

To clarify system functionality, consider a prosaic example, consumer information. Imagine someone standing on a sidewalk in the North Beach neighborhood of San Francisco, California. The North Beach area is awash with Italian restaurants. A user with a mobile computing device (PDA) issues the query "What are the closest Italian restaurants?" An integrated system is needed to answer this query in a timely fashion.

Initially, the PDA uses wireless networking to find a query server to handle the query. The query server does not store applications, so it issues queries to application servers which contain the requested information, then merges the responses and transmits them to the PDA.

In this model, the ability to deliver applications as well as information drives the rest of the research questions. In this scenario, such programs might allow the user to make a reservation, view an animation of steaming lasagna, or initiate a phone call to the restaurant. This is the "application delivery" portion of the system.

Since PDAs are heterogeneous, (i.e. the processor and other aspects of the PDA may vary from user to user), the variability of processor is handled by having the delivered applications be expressed as code for some sort of virtual machine, such as the Java Virtual Machine (JVM). This is the "mobile" portion of the system.

The system has soft-real constraints, since an individual will not wait an indefinite amount of time to receive an answer to their query. As yet another constraint, if PDAs are in moving vehicles, information must be delivered before they have moved out of the area. This is the "soft real-time" portion of the system.

Furthermore, businesses want to have control over the user-experience of their customers. A business will place references to its applications on query servers responsible for handling wireless networking in their service area. Therefore, query servers will not have a unified database of applications, and must query multiple application servers to process a clients query. This results in the "distributed" portion of the system.

The scenario outlined above exposes many interesting research issues which have not been addressed in this kind of integrated application. They include resource discovery, use of machine-independent performance profiles for soft-real time scheduling, distributed real-time spatial databases, and network bandwidth scheduling.

Personal

Another vertical slice of the integration problem is the integration of personal or home based computing and multimedia. Half of the U. S. homes have internet access, 90% of homes have cable-television, 80% have VCRs, etc. Today, a moderately "wired" individual will have a home computer, one or more televisions with cable or satellite television, a cell phone, a cordless phone, an office computer, possibly a laptop, etc. These devices have little integration, particularly in the area of messaging. There are good reasons for this lack of integration. It is natural to segregate work related communications from personal communications. It is also natural to have means of communication which you will share with friends, but not with work colleagues.

This environment also begs for integration of technology from various domains. When a new device is introduced into a home, it should be recognized and incorporated into the control system for the home. Consider the introduction of a hard-disk based video recorder, such as a TiVo^{TM}. Once the device is unpacked, the device should become responsible for integrating itself into the home environment. Once the necessary power connection is made, the device can guide the user through the remaining installation instructions. By communicating wirelessly with a home-controller, the recorder could tailor the instructions to the particular environment. If the user has a single television with a cable television feed, then the user would be directed to connect the cable feed to the recorder and to connect the recorder to the television. In a more complex environment, the user might be directed to connect the recorder to some sort of AV/mux. The technology integrated here includes wireless networking to communicate configuration information to the recorder and expert system technology for determining the proper system (re)configuration.

After the device is configured, its capabilities will be known to other relevant devices in the system. The system controller will now understand the commands that the recorder knows and the recorder will provide a mapping from a limited-domain natural language grammar to the appropriate device controls. The system controller can now accept voice commands from the user for controlling the recorder. The technology integrated here includes speech recognition in the system controller and a meta-data system for the recorder to advertise its capabilities.

The user of the system can provide instructions to the system controller such as "Whenever I answer the phone and the television is on, start recording to the recorder until I hang up." This capability extends beyond speech recognition into the realms of artificial intelligence, specifically context-sensitive natural language processing.

New Scientists and New Science

In Kuhn's theory of scientific revolutions, it is only when the old practitioners have passed on that a paradigm shift is complete. An important aspect of facing the grand challenge of ubiquitous computing is to train students to think about problems in a larger context. Even the most mundane applications, such a payroll processing, have a complex interaction with the systems and societies they are embedded in. This requires a through review of our curriculums, not just in terms of the courses offered and required, but of how we teach classes. In the area of compilers, for example, it is traditional to spend a significant amount of time in the introductory course on different classes of languages and how to construct lexers and parsers for those languages. In a revised curriculum with an eye towards integrative technology, focus would be shifted partially from details of lexer and parser construction to the use of lexer/parser generators and their use in small language design and implementation. Similar shifts of emphasis must occur in other core courses, such as operating systems, networking, databases, etc.

We must also correspondingly change the way we do research and design. The evaluation criteria for computer science researchers should be expanded to reward those who produce innovative, useful artifacts. We must also avoid the "CSL syndrome" [1], where researchers have the tendency to only communicate within their own sub-discipline, even when in an interdisciplinary environment. We must reach out to others outside our own sub-disciplines and to other disciplines such as architecture, psychology, library science, etc.

New Scientists and New Science

The real grand-challenge before us is not just developing new solutions to new problems or even new solutions to old problems. We have the grand challenge of solving old problems with old solutions, integrating and combining what is already known and understood. The grand-challenge is to have software researchers and practitioners seek answers outside their domain and work with others to integrate a broad range of technologies into a solution.

[1] CSL is the Coordinated Science Laboratory, an interdisciplinary research center at the University of Illinois.