Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user. Since we started this work at Xerox PARC in 1988, a number of researchers around the world have begun to work in the ubiquitous computing framework. This paper explains what is new and different about the computer science in ubiquitous computing. It starts with a brief overview of ubiquitous computing, and then elaborates through a series of examples drawn from various subdisciplines of computer science: hardware components (e.g. chips), network protocols, interaction substrates (e.g. software for screens and pens), applications, privacy, and computational methods. Ubiquitous computing offers a framework for new and exciting research across the spectrum of computer science.

A few places in the world have begun work on a possible next generation computing environment in which each person is continually interacting with hundreds of nearby wirelessly interconnected computers. The point is to achieve the most effective kind of technology, that which is essentially invisible to the user. To bring computers to this point while retaining their power will require radically new kinds of computers of all sizes and shapes to be available to each person. I call this future world "Ubiquitous Computing" (short form: "Ubicomp") [Weiser 1991]. The research method for ubiquitous computing is standard experimental computer science: the construction of working prototypes of the necessary infrastructure in sufficient quantity to debug the viability of the systems in everyday use, using ourselves and a few colleagues as guinea pigs. This is an important step towards insuring that our infrastructure research is robust and scalable in the face of the details of the real world.

The idea of ubiquitous computing first arose from contemplating the place of today's computer in actual activities of everyday life. In particular, anthropological studies of work life [Suchman 1985, Lave 1991] teach us that people primarily work in a world of shared situations and unexamined technological skills. However the computer today is isolated and isolating from the overall situation, and fails to get out of the way of the work. In other words, rather than being a tool through which we work, and so which disappears from our awareness, the computer too often remains the focus of attention. And this is true throughout the domain of personal computing as currently implemented and discussed for the future, whether one thinks of PC's, palmtops, or dynabooks. The characterization of the future computer as the "intimate computer" [Kay 1991], or "rather like a human assistant" [Tesler 1991] makes this attention to the machine itself particularly apparent.
Getting the computer out of the way is not easy. This is not a graphical user interface (GUI) problem, but is a property of the whole context of usage of the machine and the affordances of its physical properties: the keyboard, the weight and desktop position of screens, and so on. The problem is not one of “interface”. For the same reason of context, this was not a multimedia problem, resulting from any particular deficiency in the ability to display certain kinds of realtime data or integrate them into applications. (Indeed, multimedia tries to grab attention, the opposite of the ubiquitous computing ideal of invisibility). The challenge is to create a new kind of relationship of people to computers, one in which the computer would have to take the lead in becoming vastly better at getting out of the way so people could just go about their lives.

In 1988, when I started PARC’s work on ubiquitous computing, virtual reality (VR) came the closest to enacting the principles we believed important. In its ultimate envisionment, VR causes the computer to become effectively invisible by taking over the human sensory and affecter systems [Rheingold 91]. VR is extremely useful in scientific visualization and entertainment, and will be very significant for those niches. But as a tool for productively changing everyone’s relationship to computation, it has two crucial flaws: first, at the present time (1992), and probably for decades, it cannot produce a simulation of significant verisimilitude at reasonable cost (today, at any cost). This means that users will not be fooled and the computer will not be out of the way. Second, and most importantly, it has the goal of fooling the user -- of leaving the everyday physical world behind. This is at odds with the goal of better integrating the computer into human activities, since humans are of and in the everyday world.

Ubiquitous computing is exploring quite different ground from Personal Digital Assistants, or the idea that computers should be autonomous agents that take on our goals. The difference can be characterized as follows. Suppose you want to lift a heavy object. You can call in your strong assistant to lift it for you, or you can be yourself made effortlessly, unconsciously, stronger and just lift it. There are times when both are good. Much of the past and current effort for better computers has been aimed at the former; ubiquitous computing aims at the latter.

The approach I took was to attempt the definition and construction of new computing artifacts for use in everyday life. I took my inspiration from the everyday objects found in offices and homes, in particular those objects whose purpose is to capture or convey information. The most ubiquitous current informational technology embodied in artifacts is the use of written symbols, primarily words, but including also pictographs, clocks, and other sorts of symbolic communication. Rather than attempting to reproduce these objects inside the virtual computer world, leading to another ”desktop model” [Buxton 90], instead I wanted to put the new kind of computer also out in this world of concrete information conveyers. And because these written artifacts occur in many different sizes and shapes, with many different affordances, so I wanted the computer embodiments to be of many sizes and shapes, including tiny inexpensive ones that could bring computing to everyone.
The physical affordances in the world come in all sizes and shapes; for practical reasons our ubiquitous computing work begins with just three different sizes of devices: enough to give some scope, not enough to deter progress. The first size is the wall-sized interactive surface, analogous to the office whiteboard or the home magnet-covered refrigerator or bulletin board. The second size is the notepad, envisioned not as a personal computer but as analogous to scrap paper to be grabbed and used easily, with many in use by a person at once. The cluttered office desk or messy front hall table are real-life examples. Finally, the third size is the tiny computer, analogous to tiny individual notes or PostIts, and also like the tiny little displays of words found on book spines, lightswitches, and hallways. Again, I saw this not as a personal computer, but as a pervasive part of everyday life, with many active at all times. I called these three sizes of computers, respectively, boards, pads, and tabs, and adopted the slogan that, for each person in an office, there should be hundreds of tabs, tens of pads, and one or two boards. Specifications for some prototypes of these three sizes in use at PARC are shown in figure 1.

This then is phase I of ubiquitous computing: to construct, deploy, and learn from a computing environment consisting of tabs, pads, and boards. This is only phase I, because it is unlikely to achieve optimal invisibility. (Later phases are yet to be determined). But it is a start down the radical direction, for computer science, away from attention on the machine and back on the person and his or her life in the world of work, play, and home.

Hardware Prototypes
New hardware systems design for ubiquitous computing has been oriented towards experimental platforms for systems and applications of invisibility. New chips have been less important than combinations of existing components that create experimental opportunities. The first ubiquitous computing technology to be deployed was the Liveboard [Elrod 92], which is now a Xerox product. Two other important pieces of prototype hardware supporting our research at PARC are the Tab and the Pad.

Tab
The ParcTab is a tiny information doorway. For user interaction it has a pressure sensitive screen on top of the display, three buttons underneath the natural finger positions, and the ability to sense its position within a building. The display and touchpad it uses are standard commercial units.

The key hardware design problems in the pad are size and power consumption. With several dozens of these devices sitting around the office, in briefcases, in pockets, one cannot change their batteries every week. The PARC design uses the 8051 to control detailed interactions, and includes software that keeps power usage down. The major outboard components are a small analog/digital converter for the pressure sensitive screen, and analog sense circuitry for the IR receiver.
Interestingly, although we have been approached by several chip manufacturers about our possible need for custom chips for the Tab, the Tab is not short of places to put chips. The display size leaves plenty of room, and the display thickness
dominates total size. Off-the-shelf components are more than adequate for exploring this design space, even with our severe size, weight, and power constraints.

A key part of our design philosophy is to put devices in everyday use, not just demonstrate them. We can only use techniques suitable for quantity 100 replication, which excludes certain things that could make a huge difference, such as the integration of components onto the display surface itself. This technology, being explored at PARC, ISI, and TI, while very promising, is not yet ready for replication.

The Tab architecture is carefully balanced among display size, bandwidth, processing, and memory. For instance, the small display means that even the tiny processor is capable of four frame/sec video to it, and the IR bandwidth is capable of delivering this. The bandwidth is also such that the processor can actually time the pulse widths in software timing loops. Our current design has insufficient storage, and we are increasing the amount of non-volatile RAM in future tabs from 8k to 128k. The tab’s goal of post-it-note-like casual use puts it into a design space generally unexplored in the commercial or research sector.

The pad is really a family of notebook-sized devices. Our initial pad, the ScratchPad, plugged into a Sun SBus card and provided an X-window-system-compatible writing and display surface. This same design was used inside our first wall-sized displays, the liveboards, as well. Our later untethered pad devices, the XPad and MPad, continued the system design principles of X-compatibility, ease of construction, and flexibility in software and hardware expansion.

As I write, at the end of 1992, commercial portable pen devices have been on the market for two years, although most of the early companies have now gone out of business. Why should a pioneering research lab be building its own such device? Each year we ask ourselves the same question, and so far three things always drive us to continue to design our own pad hardware.

First, we need the right balance of features; this is the essence of systems design. The commercial devices all aim at particular niches, and so balance their design to that niche. For research we need a rather different balance, all the more so for ubiquitous computing. For instance, can the device communicate simultaneously along multiple channels? Does the O.S support multiprocessing? What about the potential for high-speed tethering? Is there a high-quality pen? Is there a high-speed expansion port sufficient for video in and out? Is sound in/out and ISDN available? Optional keyboard? Any one commercial device tends to satisfy some of these, ignore others, and choose a balance of the ones it does satisfy that optimize its niche, rather than ubiquitous computing-style scrap computing. The balance for us emphasizes communication, ram, multi-media, and expansion ports.

Second, apart from balance are the requirements for particular features. Key among these are a pen emphasis, connection to research environments like Unix, and communication emphasis. A high-speed (>64kbps) wireless capability is built into
no commercial devices, nor do they generally have a sufficiently high speed port to which such a radio can be added. Commercial devices generally come with DOS or Penpoint, and while we have developed in both, they are not our favorite research vehicles because of lack of full access and customizability.

The third thing driving our own pad designs is ease of expansion and modification. We need full hardware specs, complete O.S. source code, and the ability to rip-out and replace both hardware and software components. Naturally these goals are opposed to best price in a niche market, which orients the documentation to the end user, and which keeps price down by integrated rather than modular design.

We have now gone through three generations of Pad designs. Six scratchpads were built, three XPads, and thirteen MPads, the latest. The MPad uses an FPGA for almost all random logic, giving extreme flexibility. For instance, changing the power control functions, and adding high-quality sound, were relatively simple FPGA changes. The Mpad has built-in both IR (tab compatible) and radio communication, and includes sufficient uncommitted space for adding new circuit boards later. It can be used with a tether that provides it with recharging and operating power and an ethernet connection. The operating system is a standalone version of the public-domain Portable Common Runtime developed at PARC [Weiser 89].