# Challenges: Environmental Design for Pervasive Computing Systems

Ravi Jain* and John Wullert II
Applied Research
Telcordia Technologies, Inc
445 South St, Morristown, NJ 07960
(973) 829-4204
wullert@research.telcordia.com

## ABSTRACT

We argue that pervasive computing offers not only tremendous opportunities and exciting research challenges but also possible negative environmental impacts, particularly in terms of physical waste and energy consumption. These environmental impacts will come under increasing government and consumer scrutiny, and like other disciplines (e.g. architecture, transportation), pervasive computing will have to adapt accordingly. Further, we argue that software–related issues will play an increasing role in reducing the environmental impact of computing. We thus propose that an important challenge for pervasive computing is to develop research in new architectures, design methodologies, metrics, algorithms and operating systems to minimize these impacts. We then discuss specific research issues and questions that arise in three phases of the device lifecycle: minimizing resource usage for manufacture and operation, maximizing device lifetime, and improving recyclability.

## Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: Network architecture and design – *wireless communication, distributed networks.* D.2.0 [**Software engineering**]: Design tools and techniques.

## General Terms

Design, Algorithms, Economics, Human Factors.

## Keywords

Environmental impacts, pervasive computing, green computing.

* Current address: R. Jain, NTT DoCoMo USA Labs, 181 Metro Drive, San Jose, CA 95110. Phone: (408) 451-4767. Fax: (408) 573-1090. Email: jain@docomolabs-usa.com.



*MOBICOM'02*, September 23-28, 2002, Atlanta, Georgia, USA.


## 1. INTRODUCTION

Computer systems have come a long way from the early single-application mainframe behemoths used by specialists, often scientists and engineers, in closed, protective environments. Today, an incredible variety of computers and applications, spanning the globe and beyond, have deep and far-reaching impacts on all parts of society. Developments in mobile computing and networking, leading to the idea of pervasive computing, promise to have even greater impacts.

Pervasive (or ubiquitous) computing has been driven by ambitious, exciting and noble goals – to make computing as useful and unobtrusive as utilities like electricity and water; to produce "calm" rather than distraction; and to bring the benefits of computers to everyone by developing not only powerful, costly machines but "tiny inexpensive ones" [Weiser93, Weiser96]. While this vision has yet to be realized, much progress has been made, and researchers and engineers around the world are chipping away at the obstacles.

However, little or no thought has been given to the physical final end result of pervasive computing: devices of varying size, weight and complexity, that are useless, obsolete, malfunctioning, or simply broken – in other words, that are garbage. Further, these devices, by their very design and function, are ubiquitous, massively distributed, and embedded in numerous everyday objects and the environment. From digital jewelry and clothing, to networked appliances, PDAs, cell phones, sensor networks, smart floors and cyber homes, pervasive computing offers us not only a glittering future of convenience, comfort and connectivity, but possibly also a legacy of deadening clutter and dangerous trash: plastics that do not biodegrade, heavy metals that are carcinogenic, gases from production and incineration that are toxic, and landfills that threaten generations to come.

To gauge the extent of this possibility, it is worthwhile glancing at the physical waste produced by the current – i.e., non-pervasive – use of computers. It has been estimated that over three-quarters of all computers ever bought in the U.S. are stored in people's attics, basements, office closets and pantries [MCC96]. By the year 2004, experts estimate there will be over 315 million obsolete computers in the US; many destined for landfills, incinerators or hazardous waste exports [NSC99]. Consider that computer equipment is a complicated assembly of over a thousand materials, of which many, such as lead, cadmium and mercury, are known to be highly toxic. Thus the environmental

impacts of computers have been a subject of growing concern over the past decade [DoE].

Already, the growth of waste electrical and electronic equipment is about 3 times that of other municipal waste [AEA97]. The average lifespan of a computer tower has shrunk from 4-6 years in 1997, and is estimated to fall to 2 years by 2005. Pervasive computing will only add to this "mountain of obsolete PCs" [Watson99], both by increasing the nature and quantity of physical devices and the rate at which they become obsolete. Not only computing, but also communications devices and integrated computing and communications devices are expected to proliferate. Analysts have estimated that 13 million Bluetooth devices alone were shipped in 2001 and predict that by 2005 there will be over 780 million new Bluetooth devices shipped

Generally most pervasive computing devices – although by no means all – will have one significant environmental advantage over traditional computers: small physical size that inherently consumes less material. However, they have disadvantages as well: they will be far more numerous; low cost will encourage rapid replacement; less mature technology will become obsolete faster; disposable versions of some devices, like disposable cell phones [HopOn01, Telespree01], will soon emerge; and they will tend to use batteries, which often contain environmentally unfriendly heavy metals. In addition, their small size, weight, embedding in other materials and overall design for ubiquity will disperse them widely, making them more likely to be lost, forgotten, or simply abandoned, and making proper collection, recycling or disposal harder. Finally, pervasive computing devices, to meet their goal, will be truly global, bringing computer environmental impacts to regions of the world where little or none exist at present.

So far we have focused on the physical waste aspects of pervasive computing. Energy consumption is another significant environmental concern. It has been estimated that computing, telephony and networking equipment now account for a significant fraction of the total energy consumption in the U.S. [EStar]. While mobile devices are becoming more energy-efficient, the overall energy consumption due to such devices continues to increase as their total number increases rapidly, they integrate more sophisticated and energy-consuming peripherals (larger displays, built-in wireless interfaces, CD-R/W etc.), and the applications and system software (and even screen savers) become ever more complex [Paradiso00]. Once again, in the case of pervasive computing, this environmental impact is greater than current computing because of the use of batteries.

There are many efforts targeted at increasing the rate at which electronic devices, including computers, are recycled, and at making it more tenable to extract useful material from electronic waste e.g. see [EESymp, RecycleW, MIT]. Most of these efforts have addressed the physical aspects of electronic devices and computers: circuit boards, batteries, displays and the like.

Missing from most of this work is any effort specifically within the domains of computer science and engineering aimed at minimizing the environmental impact of computers. Just as manufacturers of household goods such as appliances and cosmetics increasingly seek to project and differentiate themselves as "green" (to respond to or avoid government regulations and consumer pressure), so will computer system manufacturers. Just as diverse disciplines such as architecture, transportation engineering, and materials science have developed principles and techniques for environmentally sensitive and sustainable design, it is likely that pervasive computing will need to follow suit. And these principles must apply to all aspects of electronic devices, both hardware and software.

## 2. DESIGN IMPLICATIONS

The intent of this paper is not to present an alarmist view of global environmental collapse due to computers, but to argue that computers in general, and pervasive computers in particular, pose an environmental risk that must, and can, be addressed.

As researchers we tend to focus on innovation, and in our view there are interesting opportunities for reducing the environmental impacts of pervasive computing by innovation. However, we believe strongly that effective reduction of resource consumption, reuse of resources, and recycling of materials – the reduce/reuse/recycle mantra – needs to be an integral part of the design process, not an afterthought. This view has been articulated for manufactured products in general, e.g.: "Very few objects of modern consumption were designed with recycling in mind. If the process is truly to save money and materials, products must be designed from the very beginning to be recycled …" [McB98]. In fact, unlike current computing, where environmental concerns were only raised after the proliferation of computers, pervasive computing offers us a unique opportunity to apply environmental consciousness *while we are still at the start* of the next wave of technology proliferation. We believe it is important to rethink pervasive computing in the sense that minimizing total lifecycle environmental impact should become one of the important factors in pervasive computing design.

Of course, much of pervasive computing research focuses on design to minimize resource consumption, mainly because of limitations of device size, weight and capabilities that make the research interesting in the first place. However, the research is generally driven by the goal of squeezing more functionality out of the resources available, where the resources in turn are limited primarily by cost, or sometimes by availability. At present, post-manufacture environmental impacts are external to the producer, and are not reflected in costs for electronic devices. This is true in the US, although recently in Europe "Extended Producer Responsibility" (EPR) policies are shifting the burden of waste electrical and electronic equipment to the manufacturer [EC00][1]. In contrast, we argue that pervasive computing design should explicitly consider and minimize environmental impacts as a separate parameter from cost. In particular, the costs of environmental impacts should be included in the design process, and methodologies developed so that the present value of these future costs can be compared on an apple-to-apples with other costs. This is not as radical as it may sound; many corporations have voluntarily embraced principles of minimizing environmental impact.

We also argue that research should consider minimizing not only production and operation costs but total lifecycle impacts, i.e., choosing techniques to reduce the costs of reuse, recycling, and

[1] Nonetheless, very recently some proposed legislation has been introduced in the U.S. Congress to address the issue of electronic waste, and this may become one of the key incentives for environmentally sustainable design.

disposal. This complicates the design process, but makes it more challenging – trading off not only between functionality and operational or production cost, but also total lifecycle impacts. At present there are few metrics and techniques to carry out or evaluate such designs, and developing these techniques for pervasive computing is a challenging endeavor.

**Example Scenario.** To make these ideas more concrete we sketch a scenario that is quite conservative in the sense that it is within the realm of technical realization. Alice leases or purchases a *Rethink* brand cell phone whose casing is made from biodegradable or recycled plastic. It contains a large proportion of recycled electronic components, and connections on its circuit boards are made with lead-free solder. The Rethink phone comes with a Shoe Battery installed and a spare that is rechargeable, when Alice walks, by means of a piezo-electric charging apparatus in her shoe. The phone is sufficiently integrated, programmable and convenient that it eliminates several other devices, such as a PDA, wallet, and various household remotes (TV, VCR, stereo, fan, garage, car, etc.). The phone has a software radio to accommodate air interface changes when Alice travels or when new technology is developed. It also has an open, standard API so that new applications (e.g. currency converters, foreign language phrasebooks) can be downloaded to it on demand, extending its function as well as life. The hardware is designed for modularity and replacability, rather than being integrated to minimize initial cost. Thus when Alice decides to upgrade, the phone can be easily disassembled by the store clerk or Alice herself, and many of the physical materials (casing, display, keys, battery circuits, etc) are reused. The manufacturer has an incentive to avoid needless obsolescence so as to avoid disposal costs. Similarly, the software is in component form and is replaced only if necessary, and with minimum delay and inconvenience. Finally, after many years, when the product wears out or is actually obsolete, Alice has an incentive (e.g. deposit refund or trade-in value) to return the phone to the manufacturer or third party for recycling.

In the rest of this paper we elaborate on the computer science and engineering challenges presented by environmental design for pervasive computing.

# 3. COMPUTER SCIENCE AND ENGINEERING CHALLENGES

Rethinking pervasive computing to minimize environmental impact has implications at all aspects of system design, including computer science and software. In fact, we believe that software will play an increasing role in the environmental impacts of pervasive computing. One reason is that, over time, more and more computer system functionality is being implemented in software rather than hardware. As hardware becomes faster and cheaper, it becomes feasible to obtain the flexibility, adaptability and programmability offered by software while still meeting system performance and cost constraints. For example, each new generation of telecommunications switching system has unbundled application logic from system software and allowed greater programmability [Anjum01, Lazar97]. As another example, software radios are being developed that will allow radio channel modulation for cell phones and other wireless communications devices to be defined in software rather than hardware [Mitola00].

We organize our discussion of research challenges and techniques in three categories. The first set of techniques involves using less physical material per device in the first place. This can involve making devices physically smaller or ensuring that they only include the needed components. The second grouping deals with using devices longer, for example by increasing their reliability or ensuring their re-usability. The final group looks at creative means of disposal, including "smart disposal" to feed information back into the product design phase. In some cases, because data is more easily available, we use examples from current computing platforms like PCs to make our suggestions more concrete.

## 3.1 Using less

An obvious first step towards reducing environmental impact is smarter design to use less materials or energy for the same functionality and performance.

### *3.1.1 Minimizing physical materials*

There are several approaches to reducing material usage when considering collections of devices. We specifically look at functional integration, resource sharing and modular design

Functional integration, the process of combining several functions into a single device (e.g. a cell phone that is also a PDA, remote control, etc.) is often pursued to increase user convenience and reduce costs. The primary user convenience is the reduction in number of devices that must be managed, tracked, carried, etc. This reduction could aid in reducing the waste stream. However, integration alone will not solve the problem of proliferation. There is a counter trend toward specialization: small special-purpose devices, particularly those embedded in other objects, such as wearable devices of many types.

Another approach to reducing the overall quantity of electronics is resource sharing. One option is to make it easier for multiple users to share a device by supporting personalization and privacy features that are invoked, for example, by biometric identification. Taking this a step further, in some cases large-scale centralized solutions are more environmentally effective than individually owned devices. For example, the physical and energy environmental impacts of telephone answering machines are almost 10 times greater than those of a centralized voicemail service [Taiariol01].

More fundamentally, computing today is viewed largely as a private resource owned by organizations or individuals, but it is clear that most computers are tremendously underutilized. One possible method to reduce the total number of computers is to allow the existing computational resources to be shared. Sharing global computer resources in this manner is being pursued [AndKub02], but largely to accomplish goals that would be unaffordable or simply impossible using dedicated computers. Early successful examples of such uses are large-scale computations required for finding prime numbers, where 130,000 users have freely contributed computing resources [GIMPS02], or simulations to understand protein folding behavior, where over 20,000 users have contributed [Pande02]. Generalizing these experiments raises two fundamental and deeply challenging problems: developing what amounts to an Internet Scale Operating System (ISOS) to manage resource allocation, security and coordination; and developing economic models and mechanisms to provide incentives for private owners to lease out their computing resources [AndKub02]. We believe these

challenges are all the more worthwhile pursuing because of the potential large-scale environmental benefits that can accrue. Moreover, environmental cost should be a factor in the design and operation tradeoffs of such an ISOS; this does not seem to have been considered. One example in [AndKub02] is to utilize the idle resources of hundreds of computers to cheaply deliver a streaming movie on demand to a participating user. However, choosing which idle computers to use for this application could involve environmental concerns in addition to cost and performance. For example, the ISOS could preferentially activate computers in colder geographic regions and let those in warmer areas go into a low-power idle state, conserving energy for heating and cooling.

The ISOS idea can also be beneficially applied to smaller-scale situations, such as the pervasive computing environment in a user's home, office and car. Thus rather than adding more devices, especially for special purpose applications, it may be possible to leverage the unused computation and communication resources in the environment seamlessly. Designing such a pervasive computing environment operating system presents many of the same challenges as an ISOS, along with additional ones of fewer resources, limited energy, restricted user interfaces, device and communication media heterogeneity, and slow or intermittent communication.

It is also worth considering how to reduce materials within individual devices. In particular, it should be possible for a user to buy a device whose modular design allows customization, so that only the specific hardware and software required by the user is included. Personal computers can typically be assembled in this way, but to be cost-effective the option units are typically large, e.g. disk drives and peripheral cards. Design and fabrication techniques that allow customized assembly for smaller and cheaper devices, and with smaller option units, need to be developed. Much as compact flash storage can be added in varying quantities to smaller devices like digital cameras, it should be possible to add other resources as well. We will discuss how the device can evolve as the user's requirements change in section 3.2.

### *3.1.2 Minimizing energy usage*

There has been a significant amount of research within the mobile computing, networking and devices communities on reducing the operational energy usage of pervasive computing devices. However, almost all of the research focuses on extending the battery life.

While this is worthwhile, a more fundamental research approach is also desirable. We believe formal models of energy consumption are required that explicitly consider energy instead of (or in addition to) CPU cycles in order to motivate design of algorithms (e.g. for data indexing or for wireless computing). This is analogous to the formal models developed to support the design and comparison of I/O-efficient algorithms [Shriver96]. Some work along these lines is in [Ellis01]. Formal frameworks can provide a basis for long-term scalable improvements in energy efficiency.

In addition, a focus on operational energy usage can be shortsighted from an engineering point of view. Analysis shows that a recent-model cell phone under a typical usage consumes about 6 kJ per day from the battery. However, if the charger is left plugged in all day (as is not uncommon) the total energy usage is 110 kJ, an efficiency of about 5% [Nicolaescu01]. The resulting difference in $CO_2$ emissions in these usage scenarios is about 5 kg per phone per day. This is an example of a situation where, for at least one class of devices, very significant energy and environmental savings could be achieved by better design.

Another interesting finding in the same study is that a half-duplex multiparty call (e.g. like the Nextel Direct Connect™ "push to talk" feature used in dispatch applications) uses roughly half the power of a regular voice call. We note, however, that many PCS and cellular system designs preclude this functionality (thus in the U.S. none of the other major nationwide carriers currently offer it), pointing to the need for research in architectures and protocols that can support low-power applications efficiently. Further, standby operation for this device consumes over 60% of the total energy. This is consistent with estimates that nearly 90% of the energy usage in other devices such as cordless phones and answering machines is in standby mode [EStarCordless02]. Clearly, focusing on operational energy savings of devices alone is insufficient; recognition of architectures, design methodologies and tools for minimizing total energy usage is required.

Finally, we briefly mention that research in alternative sources of energy for pervasive computing is desirable so as to limit the use of fossil fuels to charge batteries and potentially the use of batteries themselves, particularly for embedded and wearable devices. Further work on using renewable sources such as human and solar energy is required (e.g. human footfalls can generate over 50 mW [Paradiso00].)

## 3.2 Using it longer

People dispose of things for many reasons. One is that the item no longer has sufficient functionality; another is that it breaks or malfunctions; and yet another is that the device becomes outdated, for example it does not support the latest applications. We consider intelligent computer science and engineering techniques to address these issues.

### *3.2.1 Self-destroying data*

One reason for computing devices becoming outdated is that they run out of storage space. However, in many cases, it is likely that the storage space contains information that is of no use to either the system or the user. Some estimates indicate that 30-60% of disk space on a computer is wasted [Wang96, LyVar00]. For example, out-dated user information and multiple copies of the same information can occupy storage space needlessly. Such a situation can be created by users (e.g., keeping multiple revisions of a single document) or by systems (e.g., partially installed or incompletely removed software). This *data sprawl* from information unnecessarily stored far beyond its useful lifetime not only consumes storage, but it also costs energy (for search and management) and contributes to “virtual clutter” and usability issues. Individuals find it easier and cheaper to expand system resources (larger disks and faster processors) than to manually manage even personal information such as mail and web logs. The proliferation of multimedia content and the widespread use of digital personal libraries (e.g. already many babies in the U.S. start out with a web page) will make this a non-trivial issue.

We suggest that systems be designed to minimize data sprawl through better indexing, retrieval, on-line or automatic compression, and knowledge management techniques. For

instance, some word processors store information in mysteriously inefficient ways and most store every document revision as a copy instead of storing just the revisions. Few offer comprehensive and convenient journaling features so that data can be self-managing (e.g. the most recent revision can be located easily) or self-destroying (e.g. after a certain specified date or after a certain number of copies have been made.) While recent operating systems do make recommendations about deleting files, these capabilities need to be made far more convenient and applicable to a user's entire (pervasive) computing environment. Similarly, e-mail attachments are needlessly copied to multiple mailboxes (and sometimes to multiple devices for a given recipient) rather than being automatically stored in a logically central location. The list of such inefficiencies can easily be made much longer, mainly because the "storage is cheap" refrain hides the environmental costs of obsolete or useless data.

### *3.2.2 Programmability and just-in-time/just-right software upgrade*

One of the primary reasons people have for buying new electronic devices is to gain access to the latest software applications. A clear example of this is home video game consoles, where the latest games only work on the latest consoles. Even applications with nominally the same functionality show an ever-increasing demand for storage space and processing power. For example, as shown in Table 1, over the last 7 years the storage requirements for an office productivity software suite have increased by a factor of over 5 and the processing requirements by a factor of 40. Similarly, from 1994 to 1999, Linux kernel size (in uncommented lines of code) grew as the square of the number of days since the release of version 1.0 [GodTu01]; to about 2.5 million lines in 2001 [Wheeler01]. While some of this *software sprawl* not only increases storage and energy usage but, after only a few releases, can also make hardware obsolete. While some of this growth is related to added functionality, the complexity of the resulting applications results in much of this functionality going under-used.

Mechanisms to discover system capabilities, auto-configure systems, and allow secure, just-in-time plug-in and assembly of required system software components (similar to what is done in Web browsers) are required. Related to this are:

- *Dynamic application usage* can extend the useful life of a device by avoiding software sprawl and allowing convenient upgrade and application leasing. This involves dynamic application discovery, download, and billing, all of which has to be done with resource-limited devices.
- *Languages and APIs for programmability, modularity and extensibility are required* so that system as well as application software can be designed for reuse, replacement, and upgrade.

One indication that it is possible to address the storage problem is apparent by comparing the disk requirements shown in the last two rows of Table 1. Between 2000 and 2001, the minimum storage requirement actually decreased slightly. This decrease is attributed to an on-demand installation process that adds features only as they are actually invoked by the user. A remaining challenge is to use this approach for designing applications to address the processing requirements.

**Table 1: Application requirements example**

| Rel. Date | Disk Requirements | CPU Requirements [Rosch02] |
|---|---|---|
| 1995 | 40 MB compact;<br>87 MB typical;<br>126 MB complete | 16 MHz, 386DX.<br>Est. MIPS: 5.5 |
| 1997 | 73 MB compact<br>121 MB typical<br>191 MB complete | 33 MHz 486.<br>Est. MIPS: 27 |
| 2000 | 252 MB minimum<br>527 MB recommend | 75 MHz Pentium.<br>Est. MIPS: 126 |
| 2001 | 245 MB minimum | 133MHz Pentium.<br>Est. MIPS: 219 |

To make this process systematic and bring it to a finer grained level, hardware requirements should be written not only for entire software applications, but based on particular user-level features. Software development methodologies, and in particular requirements engineering processes, to support this must be developed.

Observe that this concept of on-demand feature-based software upgrade is different from software reuse. While software reuse generally deals with making software development more efficient through modular source code, here we are talking about installing and/or deploying executable code on an as-needed basis.

An additional problem with running newer software on older hardware is that software vendors limit the range of hardware on which they test the software. Such limitations are necessary because software testing is a time and labor intensive process. Design, development and testing processes that support modular hardware, as described below, could help to address this. Test automation, for both functional and interface testing, could enable manufacturers to certify software operation on a wider range of devices, effectively extending their lifetime. In addition, it is necessary to develop automated test techniques that can verify that the software will operate correctly when feature-level software modules are added on-demand in response to (implicit or explicit) user input.

### *3.2.3 Just-right hardware upgrades*

Another strategy that could increase the average lifetime of electronic equipment is to design the hardware and software to support system hardware upgrades.

**Example.** We sketch a simplified example as follows: suppose a PC is designed to hold multiple processors and also multiple disks. Further, suppose the operating system can support multiple, heterogeneous processors and can treat multiple disks as a single logical unit. These features increase the price of the original system by, say, 10%. Suppose a user buys this system with a single processor with adequate memory and disk space and decides to upgrade after 2 years. For about 25% of the cost of the original system (using typical current PC component costs of 10% for a new processor, 10% for a new disk drive and 5% for added

memory), the user can substantially improve the system performance. Specifically, progress in storage and processing technologies during the two years (i.e., Moore's law) doubles processing power per dollar and triples storage per dollar. Further, the user keeps the original equipment, rather than replacing it. Thus the user has a system that has three times the processing power and four times the disk space of the original system. This exceeds the capabilities the user would have if he or she bought a new, single processor/single disk system. Under this scenario, the user might continue to use the original processor and disk drive -- as well as supporting electronics -- for four years instead of replacing the entire system after three years to have similar capabilities. Software that could take advantage of multiple processors efficiently would enable this scenario. Note that this rough calculation example does not take into account the environmental savings, which would become visible if disposal costs were included in hardware prices as under EPR policies [EC00].

Thus it should be possible to treat the hardware components as flexible, modular elements that can be added, removed or replaced as needed, and the system software and applications would functionally adapt to meet the available resources. Such a scheme would require innovative packaging to support component removal and replacement, as well as functionally adaptable software.

A key to this ability to upgrade hardware modularly is an operating system that can deal conveniently with multiple heterogeneous processors and treat multiple disks as a single logical unit[2]. The extension of current, high-end, multiprocessor software technology to mass-market systems with heterogeneous capabilities and non-specialist users is an open issue.

While we described this design for upgrade in terms of personal computers, it might be even more applicable to certain types of pervasive computing systems. For example, while supporting upgrades in embedded, tightly packaged systems such as PDAs could dramatically alter the form factor in undesirable ways, wearable computers, as have been described for pervasive computing [Siegel95], can be distributed over a large area, and could potentially include slots or sockets to enable system upgrade.

### *3.2.4 Tolerating component failures*

All devices will eventually malfunction. To reduce the flow of devices into the trash, it is desirable to extend their lifetimes. One option for doing this is to design device systems with inherent redundancy while a second is to make devices repairable.

Fault tolerant systems, which strive to maintain computer state across failures, and high-availability systems that strive to minimize downtime, have generally been reserved for sophisticated applications because they are expensive. Such expense is not compatible with the aim of making computing pervasive.

Could redundant designs based on more relaxed requirements be made more cheaply? High availability systems, with their reduced requirements, are generally less expensive than fault tolerant systems [Aartsen94]. Can further relaxation of requirements result in inexpensive redundancy? Such relaxed requirements could leverage the modular nature of the software, as described in sec. 3.2.2. For example, rather than maintaining a certain capability, the system could be designed for graceful degradation, to tolerate the loss of individual underlying hardware elements in a manner that reduces application capabilities in a gradual, rather than catastrophic, fashion.

The design described in sec. 3.2.3 to support incremental hardware upgrades could also be leveraged to make hardware failure tolerant and repairable. In addition, much the way RAID technology was developed to produce reliable, high-performance storage using inexpensive disk drives [Patterson88], a distributed approach to pervasive computing systems could be built from inexpensive, previous-generation processors.

While design for upgradeability requires software that can sense system capabilities and adapt, convenient replacement of failed parts would require software that could perform intelligent hardware diagnostics, perform workarounds where possible (e.g. if the system is designed in a RAID manner), and alert the user conveniently. While some of these capabilities are present in some expensive, high-reliability systems and networks, the challenge is to make them available in pervasive computing.

Functionally adaptable software would be capable of adjusting the capabilities presented to the user based on the surviving resources. Work on self-adaptive software [Oriezy99] has been targeted at more sophisticated applications and platforms, but may be applicable as well. The challenge presented here is to create a mapping between software functions and specific computing resources and then using that mapping to adjust application behavior to resource changes.

## 3.3 Smart disposal

Eventually even the most frugally used and judiciously upgraded device must be disposed of. While recycling is preferable to simply tossing into the trash, current recycling leaves much to be desired.

There has been an increase in recent years in recycling PCs and computers to extract raw materials [Matthews97, Goldberg98, RecycleW]. Cell phone recycling has recently been instituted in Japan [Belson, 2002] and elsewhere. Cell phones can be crushed and useful metals extracted from them, yielding about 24 micrograms of gold per phone and substantially more of other precious metals. However, the process is expensive, low-margin and time consuming; in Japan, the recycling company pays about 7 cents per phone while the metals extracted are worth about 21 cents. In addition, there are concerns that the recycling process itself may pose environmental hazards as well as risks to worker health and safety. “Down cycling” existing cellular telephones to extract raw materials is apparently not a very viable or environmentally responsible practice.

In essence crushing devices into raw materials loses the vast majority of their value. Thus smart disposal and recycling techniques that identify and reuse of subassemblies should be pursued. Labeling components (possibly with RF tags) to record their identities and capabilities could possibly help this process.

[2] Adding disks today involves either replacing existing drives and re-installing the operating system, or dealing with separate logical drives that compartmentalize data, neither of which is particularly convenient for the user.

Smart disposal should also attempt to close the loop of product information: provide definitive quantitative feedback to system designers about the actual usage and upgrade (including the timing of use and upgrade) of software and hardware components and features. This would allow design of better, modular, right-sized and upgraded systems. Hardware and software techniques to provide this information conveniently, in a scalable manner, and while preserving privacy would need to be developed.

## 4. CONCLUDING REMARKS

We have argued that environmental design of pervasive computing is an essential and inevitable challenge for the future. While environmental impacts are typically viewed in terms of minimizing physical material usage and waste, we argue that in the case of pervasive computing, software will increasingly be key to reducing hardware impacts. Doing so requires examining our system design processes with a new metric: reducing environmental cost. With this overarching theme, we have surveyed a wide range of new computer science and engineering techniques that are required, at various levels of system design, that can help reduce material and energy usage, help reuse and prolong the life of devices, and help smarter disposal and recycling.

Many of these techniques, including software modularity, Internet-scale operating systems and self-diagnosing hardware, are being investigated in other contexts. This is advantageous in that it provides multiple reasons both for performing the research and for deploying the resulting technologies. Exploiting these techniques for green pervasive computing is will be a challenge in itself because of the power, size cost and processing constraints imposed by pervasive computing. Further, and as importantly, we believe that entirely new avenues of research need to be pursued, such as developing new formal models and metrics for environmental costs, design for renewable energy sources and total-lifecycle energy management, as well as techniques for smart disposal and usage or upgrade feedback into the design process.

Developing, evaluating and refining these techniques is the heart of the environmental design challenge for pervasive computing.

## References


[Aartsen94] M. E. Aartsen, High Reliability in New York Telephone and New England Telephone Development Projects, Twincom Workshop in Slotje Limburg at Oosterhout, October 25,1994. http://pws.prserv.net/playspace/papers/HAVAIL.htm

[AndKub02] D. P. Anderson and J. Kubiatowicz, The worldwide computer, *Scientific American,* 40-47, Mar. 2002.

[AEA97] AEA Technology, Recovery of WEEE, Economic and Environmental Impacts, June 1997.

[Anjum01] F. Anjum, et al., CitiTime: A system for rapid creation of portable next-generation telephony services using third-party software components, *Computer Networks,* vol. 35, 579-595, 2001.

[Belson02] K. Belson, Mining cellphones, Japan finds El Dorado, *New York Times,* page G1, Feb. 26, 2002 [Cahners01] Cahners In-Stat Group, Bluetooth Overtakes 802.11x with 2001 Shipments on Track, Report MM01-18BW, Nov. 2001. See http://www.instat.com/abstracts/mm/2001/mm0118bw_abs.htm

[EC00] European Commission, Commission tackles growing problem of electrical and electronic waste, DN: IP/00/602, Press Release, June 13, 2000. See http://europa.eu.int/rapid/start/cgi/guesten.ksh?p_action.gettxt=gt&doc=IP/00/602|0|RAPID&lg=EN

[DoE] U.S. Dept of Energy, The Energy Star Program, See http://www.eren.doe.gov/cities_counties/saving1.html

[EESymp] Proc. of IEEE Symp. on Electronics and the Environment, 1994-2001.

[Ellis01] C. S. Ellis, The Milly Watt Project, 2001. See http://www.cs.duke.edu/~carla

[Estar] U. S. Dept of Energy, Press Releases: "EPA Administrator Looks to Telecommunications Industry for Increased Energy Efficiency Opportunities", 11/16/2001; "Energy Star to Launch Label for Telephony Products at Consumer Electronics Show in Las Vegas, January 8-11, 2002"

[EStarCordless02] U. S. Dept. of Energy, Cordless Telephones, Answering Machines, and Combination Cordless Telephones and Answering Machines, 2002. See: http://yosemite1.epa.gov/estar/consumers.nsf/content/cordlessphones.htm

[GIMPS02] Great Internet Mersenne Prime Search, See http://www.mersenne.org

[GodTu01] M. Godfrey and Q. Tu, Growth, evolution, and structural change in open source software, *Proc. Intl. Workshop on Principles of Software Eng. (IWPSE)*, Sept. 2001.

[Goldberg98] C. Goldberg, Where do computers go when they die? *New York Times,* Mar. 12, 1998.

[HopOn01] Hop-On Communications, See http:// www.hop-onwireless.com

[Lazar97] A. Lazar, Programming telecommunication networks, *IEEE Network*, 11, 5, 8-18, 1997.

[LyVar00] P. Lyman and H. R. Varian, How much information, 2000. See: http://www.sims.berkeley.edu/research/projects/how-much-info

[Matthews97] H. S. Matthews et al, Disposition and End-of-Life options for personal computers, *Carnegie Mellon Univ. Green Design Initiative, Rep. 97-10,* July 1997.

[McB98] W. McDonough and M. Braungart, The NEXT Industrial Revolution, *Atlantic Monthly*, October 1998.

[MCC96] Electronics Industry Roadmap, Microelectronics and Computer Technology Corporation, Austin, TX, 1996.

[MIT] MIT Green Computing: An examination of the environmental effects of computers at MIT, See http://ecocomputers.mit.edu

[Mitola00] J. Mitola, *Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering*, Wiley, Oct. 2000.

[Nicolaescu01] I. V. Nicolaescu and W. P. Hoffman, Energy consumption and cellular telephones, *Proc. IEEE Symp. Electronics and the Environment*, 2001.

[NSC99] Electronic product recovery and recycling baseline report, National Safety Council, Washington, DC, 1999.

[OReilly99] J. O'Reilly, Down but not out: Floppy disks far from dead, *Tape Disc Business,* 1999.

[Oriezy99] P. Oreizy, et al., An Architecture-Based Approach to Self-Adaptive Software, *IEEE Intelligent Systems,* Vol. 14, no. 3, May/June 1999, pp. 54-62.

[Pande02] V. Pande, The folding@home project, See: http://folding.stanford.edu

[Paradiso00] J. Paradiso, Renewable energy sources for the future of mobile and embedded computing, MIT Media Lab, Mar. 2000.

[Patterson88] Patterson, David A., Garth Gibson and Randy Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," Proc. of the 1988 ACM SIGMOD Conf. On Management of Data, June 1988.

[RecycleW] Recycler's World. See http://www.recycle.net

[Rosch02] W. L. Rosch, The Winn L. Rosch Hardware Bible, 2002. See http://www.hardwarebible.com/Microprocessors

[Siegel95] J. Siegel, R. E. Kraut, B. E. John, K. M. Carley, An Empirical Study of Collaborative Wearable Computer Systems" Conf. on Human Factors in Computing, May 7 - 11, 1995.

[Shriver96] E. Shriver and M. Nodine, An introduction to parallel I/O models and algorithms, in R. Jain, J. Werth and J. C. Browne, *Input/Output in Parallel and Distributed Computer Systems,* Kluwer, 1996.

[Taiariol01] F. Taiariol, P. Fea, and C. Papuzza, Environmental impact of a telecommunication service, *Proc. IEEE Symp. Electronics and the Environment*, 2001.

[Telespree01] Telespree, Inc, http://www.telespree.com

[Wang96] P. Wang, "How to Save Half Your Disk Space Automatically." *HPWorld 96*, http://erpnews.com/pubcontent/interact/jul96/07pwang/pwang.html

[Watson99] T. Watson, "USA sitting on mountain of obsolete PCs", *USA Today*, June, 22, 1999.

[Weiser93] M. Weiser, Some Computer Science Problems in Ubiquitous Computing, Communications of the ACM, July 1993.

[Wesier96] M. Weiser and J. S. Brown, The coming age of calm technology, Xerox PARC, Oct. 1996. See http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm

[Wheeler02] D. A. Wheeler, More than a gigabuck: Estimating GNU/Linux's size, June 30, 2001. See: http://www.dwheeler.com.