

Framing Pervasive Computing

Adnan Shaout and Hari Srinivasan

The University of Michigan – Dearborn
The Electrical and Computer Engineering Department
Dearborn, MI 48128 shaout@umich.edu

ABSTRACT

Pervasive computing is an environment in which computers can be embedded as part of any product or process, and communicate with each other and traditional enterprise systems. The core components of pervasive technology, embedded and networked modules, are not revolutionary in a society that has seen chips placed into everything from automobiles to refrigerators. What is new and distinguishes the paradigm of pervasive computing is the emergent behavior of these devices to create a fabric of computing that adapts around the user.

Despite the core principle of pervasive computing being to create this fabric of computing, the majority of research today focuses on finding the next killer application, centering on designing individual applications of pervasive computing. This paper presents a framework for the creation of pervasive computing environments as whole, and demonstrates that in order to successfully create an environment; bottom-up design methods must be used.

By using complexity theory as the bottom-up framework, seven characteristics to create a pervasive computing environment are presented. The paper will present a through overview of the field of pervasive computing and the framework used by IT architects and developers to bring their departments into the pervasive paradigm.

Keywords: Pervasive Computing, Framework, Design Principles, Nodes, and Emergent Behavior.

INTRODUCTION

In what would turn out to be the founding vision of pervasive computing, in a 1991 publication, then Xerox's Palo Alto Research Center's chief technology officer Mark Weiser painted a world where technologies "weave themselves into the fabric of everyday life", indistinguishing themselves and their information from physical space [1]. Today, in the spirit of Weiser's vision, the National Institute of Standards and Technology states that pervasive computing is, "(1) numerous, casually accessible, often invisible computing devices, (2) frequently mobile or embedded in the environment, (3) connected to an increasingly ubiquitous network structure." [1] These three attributes shift the focus of the computer age from 'computers' to 'computing', the core paradigm change that will be caused by pervasive computing.

To better understand the shift from computers to computing, imagine waking up in the morning and being asked by the alarm clock if you'd like coffee. You mumble yes, one of two words the clock knows, and it sends information over to the kitchen to brew a pot. You enter the bathroom and realize you're almost out of toothpaste, a quick scan and it enters into your grocery list, along with information from a weight-sensor in your fridge that your milk carton is near empty. The list is automatically sent to your grocer as well, and will be waiting for you when you arrive at the store. You find your way to the kitchen and sit down with the newspaper to enjoy your coffee. Seeing an article on real-time systems, you underline an important quote, and the pen transfers your message to an email. You've been up less than 10 minutes, and your coffee, groceries and some research are already complete. This is the vision of pervasive computing, to allow ubiquitous computing to improve existing functionality, be it simplifying the morning routine or manufacturing a car.

To attain that vision, pervasive computing must be an entire environment that adapts around the user. Still, the majority of research available is on how to design a single pervasive computing application, with little work on how to create the environment as a whole. These top-down design principles to solve singular problems are not robust enough to manage the complexities of the entire pervasive computing environment. This leads to a problem for the many IT personnel looking for general principles to guide the creation of robust environments for pervasive computing that can support many different functions and be flexible enough for future innovations, as the only creation frameworks available currently are based on achieving a particular goal.

This paper serves two purposes. The first is to provide the reader an overview of the current research and market drivers that are beginning to define the paradigm of pervasive computing. The second is to provide a framework (a model that governs the interaction of its components [2]) that can be translated to design principles for a pervasive environment. Section 1 looks at the business drivers, technical components, and systems challenges of pervasive computing today and in the near future. Section 2 examines the problems with currently available design principles [3] for pervasive computing and presents Complexity Theory as a framework by which pervasive environments can be formed. Finally, Section 3 uses the seven key characteristics of complexity theory as design principles that can conquer current challenges and lead to the successful creation of a pervasive environment.

1. STATE OF THE ART OF PERVASIVE COMPUTING

Not surprisingly, Mark Weiser's research team failed in its 1991 attempt to implement their vision. The last decade, however, has brought a culture embracing pervasive technology as well as technical advancements to meet demands.

1.1 Business Drivers

Increases in both customer and corporate demand have made pervasive computing a pertinent topic in today's business world. IBM analysts predict continued short-term growth, stating, "the pervasive computing marketplace today (2003) is \$136 billion and growing at 28% Compound Growth Rate through 2005, where it is projected to reach \$287 billion" [5]. The continued growth of the smart phone, the most prominent connected and personalized devices, represents the trend towards pervasive computing, with market research firm Canalys showing overall mobile handheld shipments increasing 83% between Q3 2003 and Q3 2004, with smart phone shipments increasing 190% [6]. The growth of pervasive computing is consumer driven, and companies that invested in pervasive computing are already seeing their solutions payoff. For example, Delta Airlines unveiled a plan allocating \$25 million to track baggage with RFIDs by 2007, far offsetting the \$100 million per year spent currently in misrouted baggage costs. Due to the strong return on investment, it is predicted that nearly 70% of enterprises will deploy a mobile, pervasive-oriented solution by 2005 [7]. Figure 1 provides some examples of pervasive computing in the market today.

Turkish mass transit riders once purchased 45 different kinds of passes to board some part of the public transportation network, causing commonality and logistics nightmares for city officials and headaches from having to hold and sort through the various types of tokens for transit customers. To conquer this, Turkish engineers looked to pervasive computing to find an intelligent token that could store proper cash amounts, wirelessly deduct differing fares when swiped for various forms of transit, and be small and durable enough for everyday use. As a result, 1.4 million key fobs, which hold embedded chips to keep track of money deposited, were issued to users, and is today the only form of token used throughout the Turkish mass transit system [8].

Singapore's Electronic Road Pricing (ERP) system uses radio frequencies (2.54 GHz band) to connect with smart cards inside vehicles. When a user enters a highway, sensors on the road deduct a given amount of money from the smart card. The amount of money deducted is determined by an algorithm that adjusts payments based on traffic speed (thus, when vehicle speeds are slow, the price increases, and lessening further traffic). It should be noted that the algorithm does not change in real time, but trends traffic-data and updates charges every 3 months. Violators' (those without enough cash on their smart card) license plates are immediately photographed and have a summons to court automatically generated. Overall, it is a good example of multiple wireless, embedded devices and bottom up control. [9].

Mexican Attorney General Rafael Macedo de la Concha and 160 of his employees, all who work at the anti-crime information center in Mexico City, had rice-grain-sized RFIDs chips implanted in their arms for authorization into the building and to manage future security issues.

Figure 1: Some real-life examples of pervasive computing in the market place today

1.2 Technology Drivers

Technology to support an environment of computing has also matured. In truth, these technical changes are not revolutionary, but are simply the next evolution of the computer industry. The one mainframe to many users model represented the first phase of computing and peaked in sales and demand near 1975. By the year 2000, the personal computer, with a one machine to one user model also began to plateau [4]. To enable the coming era of the "one user, multiple computers" model of pervasive

computing, various technologies are needed, including microprocessors, sensors, networks, compact data storage and software. The common theme between the optimization of these components for pervasive computing is that they all aim to act as a system in order to allow the fabric of computation to disappear from users' consciousness. A more in-depth examination of each technology and its predicted future growth, that can be used to predict the growth of the pervasive paradigm, is presented in figure 2.

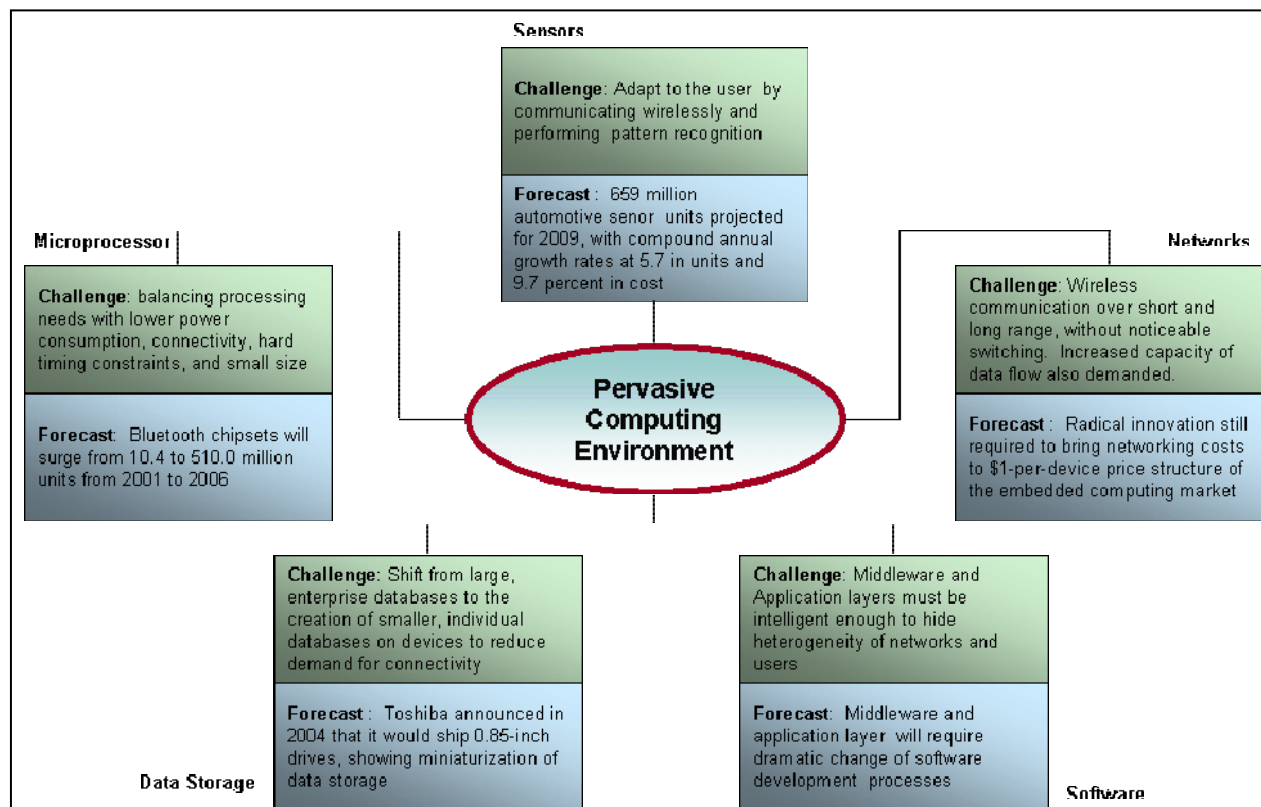


Figure 2: The future forecasts and challenges of the technical components of pervasive computing

1.2.1 Embedded microcontroller:

While most media attention focuses on Moore's law and the fast growth of computing power (in 2000, a mobile phone had more processing capacity and memory than a mainframe did in the 1960s), general-purpose microcontrollers used in pervasive oriented, embedded spaces are technically characterized by balancing processing needs with lower power consumption, wireless methods of connectivity, hard timing constraints, and small size [10]. An examination of laptop processors demonstrates the stride microcontrollers have made towards enabling small devices with large computing power. The mobile Pentium III allows the processor to run at 300MHz when the notebook is running on its battery,

where it operates at less than 1 volt and consumes on average less than half a watt of power [11]. The growth of nanotechnology and system-on-chip (SoC) processes promises to take chips to a new level, with micron wide devices that can hold even more processing and connectivity capabilities [12].

The forecast for the embedded microcontroller shows that the move beyond the desktop in embedded chips is well underway, largely driven by the transition to an event-based, real-time economy. US government research statistics for the year 2000 show shipments of new microcontrollers outnumbered those of new computational microcontrollers by a factor of almost 50, and, according to Intel, already more than 95

percent of devices containing microcontrollers do not present themselves to users as computers [13]. Future trends show that while the overall semiconductor industry is forecast to experience a fairly modest 6.1% compound annual growth rate from 2003 to 2008, embedded chips with wireless functionality will have large growths, with for instance, Bluetooth chipsets predicted to surge from 10.4 to 510.0 million units from 2001 to 2006, a five-year 118% Compound Annual Growth Rate (CAGR), with silicon revenue rising to over \$1.8 billion in 2006 [14].

1.2.2 Sensors and actuators:

Connecting the physical world to the world of computation is done through sensors and actuators. Sensors translate a form of energy (light, heat, movement) into information, and actuation converts the information into action. The growth of micro-electromechanical systems (MEMS) technology have been the hardware enabler for many sensor-driven applications; MEMS revolutionized the industry by allowing mechanical devices (accelerometers, barometers) to be constructed on a silicon chip, with lower power consumption, reduced costs and improved performance. Advances in power supplies allowed these chips to become individual, independent nodes. Software has helped deal with the uncertainty of sensors, another inherent problem in sensor, by reducing latency and filtering for more accurate data [15].

The last five years have shown that the weaving of sensors into society has begun. Between 1998 and 2002, the number of New York Times articles with the word "sensors" doubled, and in 2004 the city of Chicago announced plan to link 2,250 camera sensors to police officials to spot criminal behavior. The proliferation of sensors can be seen more clearly through the increase of sales. As sensors have now reached the steepest part of the cost-reduction curve, like processors did in the late 1990s, their sales have drastically increased [13]. Frost and Sullivan estimate about 447 million sensors and \$2.18 billion in North America production in 2002. Growth to 659 million units and \$4.15 billion is projected for 2009, with the compound annual growth rates associated with these numbers at 5.7 in units and 9.7 percent in cost [16].

1.2.3 Wireless Networks:

The improvements of wireless networks, in areas such as data transfer speed, available frequencies, and accessible locations are also a driver in the 'all the time, everywhere' vision [17]. Today, a continuum of wireless networking options is available to pass

information between two devices. For long-range wireless connections, the WiMax standard, also known as 802.16a, can transfer about 70Mbit/sec over a distance of 30 miles to thousands of users from a single base. Many cell phone companies subscribe to another system, 3G based on GSM standards, that sends packet-based transmission of text, digitized voice, video, and multimedia at data rates up to and possibly higher than 2 megabits per second (Mbps). For short-range wireless connections, or Wireless Local Area Networks (WLANs), industry consortiums have developed solutions with separate frequencies and standards such as Radio Frequencies, 802.11, Bluetooth, ZigBee, and Infrared. As can be imagined, due to the large continuum of standards, what is important in achieving ubiquitous networking is the ability for devices to deal and handle the different forms of communication needed for its operation.

Still, pervasive networking is the biggest challenge to the new paradigm, largely caused by the scalability issues that will be caused when devices, not just people, demand network services. As Ian Barkin, a senior analyst at business consulting firm Harbor Research, states, "Think about having a house with 10 devices each reporting 15 data points just three times a day, even that can get into trillions of data points being thrown at servers somewhere" [18] This ever-growing group will need an extended backbone infrastructure to meet demand and will require a revamping of existing protocols and standards for full integration.

1.2.4 Data Storage:

In the late 1990s, the trend for knowledge sharing began to shift from large, enterprise databases to the creation of smaller, individual databases. While technologies to create smaller storage differ, improvements in read-write head size, tracks per inch to allow more data density, antiferromagnetically coupled media (to enable data recording at ultra-high densities while maintaining data integrity), and higher data transfer rates have allowed vast improvements in storage. Cornell University demonstrates the progress of these technologies with Nanomagnets, which hold storage capabilities in spaces less than 100 nm wide.

A glance at consumer electronics today shows that cell phones can store pictures, music files and video clips, all of which increased memory demand [19]. To meet this demand, various technology providers have started to produce small data devices that can hold more information, mixing memory techniques, such as caching, prefetching and archiving with

hardware design [10]. For example, Hitachi's Microdrive, a one-inch, 16 ounce, 4GB hard drive, began to ship in 2003, and was quickly purchased by companies including Kodak, Minolta Co., Ltd., Nikon, Olympus, Pentax Corporation, and Sony to be used in future products [20]. Similarly, Toshiba announced in 2004 that it would ship 0.85-inch drives, about one-fifth the volume and weight of its current 1.8-inch drives, which are used in digital music players such as Apple Computer's iPod and sub-notebook computers [21]. If the trend continues, by 2012, a Tbyte of data will fit into a one-square inch device [10].

1.2.5 Software Frameworks:

While many frameworks exist on how to architect software solutions for pervasive computing, two layers have been the focus of the major of research for software that can adapt to its environment: Middleware and Application. Middleware mediates with the networking kernel, while applications take cues from the environment to guide middleware and networking issues [22].

Middleware mediates interactions between the network and end-user application and will become a major enabler of pervasive computing [23]. The proliferation of pervasive devices, combined with the lack of a single system administrator and dynamic nature, requires middleware capable of evolving and adapting to automatically solve problems. To increase the challenges faced by middleware developers, the components that middleware glues together will also be dynamic in nature. The issues

of mobility, disconnection, plug and play devices and heterogeneous need for resources of devices, as well a different latency and bandwidth of networks, present constantly varying networks and device interactions. While middleware, like all software, can always be optimized, new design patterns and platforms in mobile computing have shown success at conquering the challenge of pervasive computing. Applications are much more function-specific than middleware, run on pervasive devices and relay environment information to middleware and networking layers. As pervasive computing will be woven into the environment, applications must be intelligent enough to trigger action that needs network resources. For instance, if a data from a sensor indicates that a car is overheating, the application must raise the proper alarm and notify the network. While numerous passive wireless devices exist, more task-specific, smart application structures are being developed to enable pervasive computing.

1.3 Issues and Challenges

While the core technical features and business demand have grown to define a present society prime for pervasive computing, its future depends on the field's ability to conquer its unique challenges. Pervasive computing involves a much more integrated system than previous computing evolutions, leading to societal and technical challenges that have yet to be conquered. These challenges are summarized in Figure 3 and examined below.

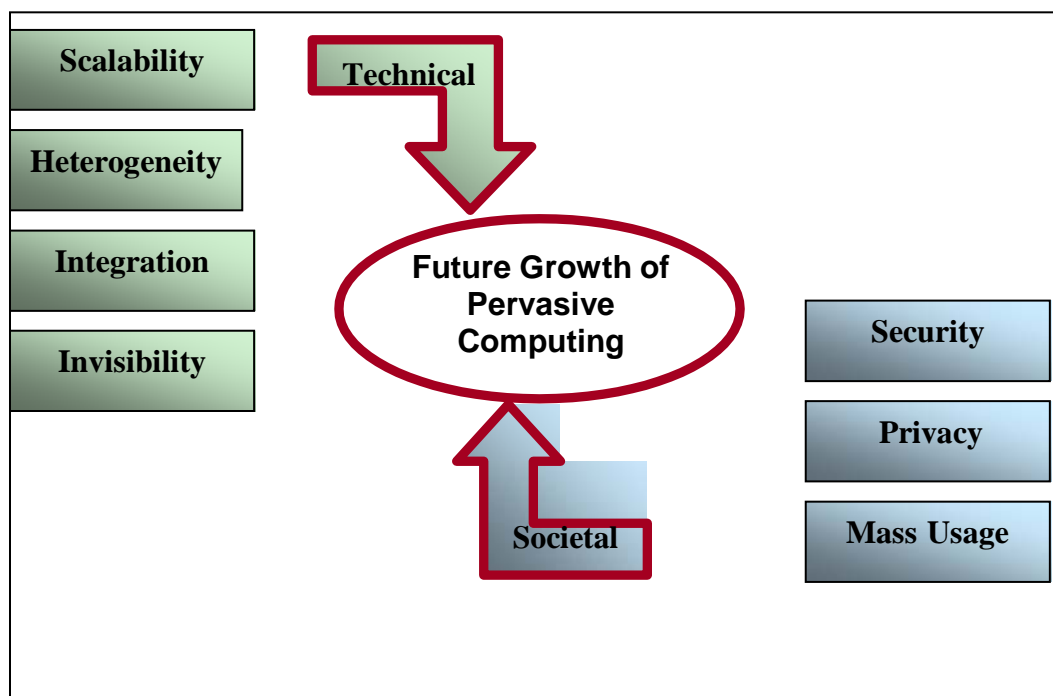


Figure 3: The issues and challenges affecting the growth of pervasive computing

1.3.1 Technical:

The technical barriers to pervasive computing are well documented, and are in many ways a continuation of the challenges traditionally associated with mobile and distributed systems. The following four principles are summarized from two monumental papers [22], [24]:

1.3.1.1 Scalability – The expected proliferation of users, devices, and networks will greatly strain the computing environment. By the numbers, by 2005, machine-to-machine technologies will create 35 billion microcontrollers and 750 million smart sensors that all demand to be connected to a pervasive network [25]. This proliferation creates stresses to the development and distribution of pervasive applications. In development, recreating application logic for each new device to be added, as is often done today in mobile computing, it would be impossible to meet demand. In deployment, explicitly installing each of the billions of applications will also clearly be impossible.

1.3.1.2 Heterogeneity – The availability of resources will differ across locations, thus creating noticeable differences in computing, detracting from users being able to depend on pervasive technology and not allowing computing to fade out of the user's consciousness. For instance, a university and a highway may have vastly different "smartness". Mobile computing has made great strides in hiding differences in coverage from the user, and middleware can use similar ideas to manage network resources. At the application layer, however, the challenge still remains on developing application that can run regardless of the platform.

1.3.1.3 Integration – Even more difficult to manage than the growth of devices and users are the interactions between all the components of pervasive computing. Coordination is needed for message routing and sharing of resources, as well as data integrity. In addition, cooperation is needed to allow a system-level optimization to best adjust to the user. Finally, capacity must be increased (for example on servers) to handle growth. The methods by which this challenge is solved will have implications on reliability, quality of service, invisibility and security implications.

1.3.1.4 Invisibility – While Weiser used the term invisibility; Satyanarayanan gives a more practical technical goal of minimal user distraction. For the most part, environments and nodes should be able to tune themselves, such as dynamic network

reconfigurations. Furthermore, they should bother the user as little as possible, blending in seamlessly with physical spaces. The abilities to act as expected and bother the user as little as possible are core ideas to the field of pervasive computing.

1.3.2 Societal:

Societal challenges, challenges not traditionally in the scope of pervasive computing research, will also affect the growth of the paradigm. The following are some of the societal issues:

1.3.2.1 Security – In addition to technical challenges, security fears must be overcome for wide usage of pervasive computing. Security fears take on three forms: virus protection, encryption mechanisms and controls. The first two categories have largely been addressed through mobile growth, with IDC predicting future growth of anti-virus and anti-spam mobile applications. Nokia is offering SSL encryption for Web-based applications, and after recent attacks like Cabir, the first-ever computer virus capable of spreading over mobile phone networks, companies such as Fsecure have created antivirus software services for devices with mobile operating systems [26]. Controls, however, must undergo a drastic change for pervasive computing. Currently, controls are largely accomplished through user authentication to verify identity and grant access rights. In pervasive computing, however, all users are not predetermined, there is no central control, and processing power and processing time are precious commodities, making current standards inadequate for the increased flexibility offered [27].

1.3.2.2 Privacy – Privacy encompasses granting users confidentiality (preventing unauthorized reads), integrity (preventing unauthorized writes) and availability (ensuring authorized users are not denied service) [28]. The difficulty in the "one user, multiple computers" model to provide these three functionalities is currently one of the major inhibitors of pervasive computing. From a developer's standpoint, solving privacy issues largely falls into the security category above, but there are larger societal issues. First, there are questions on who should be authorized to see certain information. For instance, currently a company can decide who logs on to its computers and networks, but does the government own the right to information at a park? Similarly, is it right for marketing companies to be able to purchase Internet profiles from service provider? Second, there is a liability question related with availability. The European eCall system, for

example, requires all vehicles traveling European Union roads to have technology to alert emergency services in the event of a crash, either manually by the driver or by electronic sensors in the vehicle [29]. Thus, what is the failure rate the system and each device are allowed to have and what in the system location range available? The answer to these questions will direct the future of pervasive computing.

1.3.2.3 Mass Usage – Technical solutions that work individually in the research lab may not work in the mass quantities needed to support a computing environment for two reasons. First, every computing device must be cost effective at high production numbers. For instance, for Radio Frequency IDs (RFIDs) to be used, a company must be able to justify the purchase of hundreds of thousands of tags with labor savings. Second, as devices must be embedded into varying physical domains, they must be durable enough for mass usage. For instance, in-vehicle sensors must be durable enough to last in varying temperatures and for the life of vehicle, and it has been shown that many rear-view cameras fail when the lens gets dirty, rendering the device inoperable. While technologies may be ready for everyday usage in a lab, the mass production challenges must still be overcome.

2. PERVASIVE COMPUTING DESIGN FRAMEWORK

To solve the challenges of pervasive computing a vastly different design framework is needed. Currently, the majority of this research has been focused on finding a killer application that would

draw significant interest and funds to the field [30], thus creating top-down design principles to solve specific problems. But these top-down design principles are made to solve a specific task and reach an end result, and thus are not suitable to design entire environments of pervasive computing, which have no specific end goal.

A more viable guiding design framework can be achieved by realizing that Weiser’s vision of pervasive computing was to create a complex system. What distinguishes a complex system from a merely complicated one is that there is no blueprint that controls end behavior, but ordered behaviors still emerge as a result of patterns of relationships between elements. For instance, the human body is formed through the bottom-up interactions of millions of cells, none of which are directly guided to form a person, but do anyways due to a number of biological rules and constraints. Complexity theory can also be seen in computing as well; the Internet grew through a set of simple rules and protocols. Figure 4 shows examples of complex systems in computing based on Holland’s complexity characteristics [32]. Emergence is perhaps the most important property of a complex system, as it creates a new layer of existence that, like the human body, exhibits its own unique behavior with no resemblance to its components. In Weiser’s vision, pervasive computing is an emergent layer, where the collective behavior of individual nodes forms a system that, without being explicitly programmed, creates a layer of computing beneficial to the user. As pervasive computing is a complex system, design principles that can guide a complex system are needed in pervasive computing.

John Holland's Complexity Characteristics	Internet [31]	FTP	Napster (P2P)	Linux	SETI@home (largest distributed computing project in history)	Hyphos is a wireless, self-organizing digital network. [44].
Aggregation	Network of Networks, aggregation of networks	Network of Nodes	Network of user base stations	Linux is the brain child of thousands of developers putting ideas together	More than a million PC users on the Internet donate unused CPU cycles to search for extra-terrestrial life by scanning through 35 gigabytes of signals received daily at the Arecibo Radio Telescope in Puerto Rico.	A network collection of mobile and autonomous nodes. They can organize in multiple ways

Tagging	Every website has a tag - URL's, Every group of websites has a tag - .edu, .com You can search by tag - Google, Yahoo!	Every file has a tag and every location has a tag	Every file and user has special identification	Every Linux instance has a unique identification	Every PC has unique identity	Each node is autonomous acting as its own router
Nonlinearity	Growth of the Internet users is exponential	Users grew rapidly and fell rapidly with P2P	Users grew rapidly	Linux Kernel distribution grew exponentially between 1990 and 2001	Number of users grew exponentially and then tapered	Not yet in usage (in research only)
Flows	The idea of the internet is network flows	Based on flow of information	Based on flow of files, music, etc.	Linux is based on free information flow can make a better OS	The idea of SETI is to break up computation into parallel chunks and flow data for computation to a client which returns results	Each node in the network communicates only with its immediate neighbors. Neighbors relay messages to their neighbors in turn until the message reaches its destination.
Diversity	Clearly, diversified by users base and information	Can send nearly any file	Technology has been used in open source code to music to subscription services	Can be seen by the number of drivers being written. Wide usage of Linux today	Every computer brings varying computational speeds and processing ability	Nodes can be in any form.
Internal Models	If we see a single website as an agent, many can predict demand and adjust around users. Amazon for example, can recommend choices and adapt based on users with internal, expert system rules	Each file has an internal model that allows it to accept/reject transfers. Each computer can adjust its network based on connectivity speed	Nodes that cannot send information quickly are not used and can be weeded out of the system	Every instance of Linux has a unique architecture that can be used as shareware if wanted and given to other developers	Every PC has its own computational framework based on its internal configuration. To ensure that none of them give erroneous results, redundancy is used where multiple machines do the same calculations and voting is used for correct answers	Each node is constructed in its own way, with its own communication standards and power supply
Building Blocks	TCP/IP	TCP/IP	TCP/IP + varying technologies	The different flavors of Linux all came from the same internal kernel	Every computer is given a SETI load as its basis. Computers that can compute more, get more data to use.	Standard protocols for communication

Figure 4: Examples of Complex Systems in computing.

3. PERVERSIVE COMPUTING DESIGN CHARACTERISTICS

Directing a complex system to drive success requires a unique design and development process. To build a house, every detail can be specified and the end result can be pictured before hand. With complex systems,

such as the Internet, however, blueprints must be forsaken for more general bottom-up rules that direct emergence. Regardless of where they're found in nature, all complex systems share some similarities. In studying these similarities, John Holland has listed seven characteristics shared by all complex system [32]. By using these characteristics as design principles, or essential objectives that provides a theoretical framework for design decisions, a robust pervasive computing environment, that conquers many of challenges impeding Weiser's initial vision, can be created.

A more detailed look at each characteristic as a design principle and the way the characteristic relates to the challenges remaining in pervasive computing is presented in figure 5. Example methods of implementation for each of the characteristics are also provided; it should be noted, however, that how each characteristic implementation can be a research topic on its own and examples are given only to demonstrate the strength of the framework.

Holland's Complexity Attributes	Translation of concept into Pervasive Computing	Implementation	Related Challenge/Issue
Aggregation	Patterns of behavior in nodes that produce an emergent layer of computing	Kubiatowicz's P2P principle (Stability through Statistics) used to guide emergent layer	Integration, Privacy
Tagging	Each node must have an individual tag	Tag all users, devices and environments or tag data	Heterogeneity, Security
Nonlinearity	Behavior of the environment will not be reflective of individual nodes	Separation of application from environment	Scalability
Flows	Information flow between nodes	Creation of effective networks	Invisibility, Cost Effectiveness
Diversity	Heterogeneous nodes are considered the norm, but differences must not be apparent to the user	Adaptable applications that expect change and irregularity	Heterogeneity
Internal Models	Every node has implicit rules that are hardwired and explicit rules that require a decision. Rules help the node adapt to the environment	Hardware design for implicit rules and intelligent software for explicit rules	Invisibility
Building Blocks	Basic core services can be reused among nodes and lead to fast development. Their combination is guided by need	Reusable services implemented in middleware	Scalability, Cost Effectiveness

Figure 5: Complexity theory can generate design principles for the creation of pervasive computing environments

3.1 Aggregation:

All complex systems portray the property of aggregation, a term that has two meanings depending at the level from which it is viewed¹. At the agent level, aggregation is a form of categorization that lets each individual agent cope with its environment. From the behavior of these agents, patterns of organization emerge that lead to hierarchical organization; for instance a city is made up of its

inhabitants, but patterns of organization, such as commercial districts, automatically grow. As pervasive computing, in Weiser's original vision, is a complex system, aggregation is an essential objective for all pervasive computing environments.

One method of implementing aggregation into a pervasive environment is demonstrated by John Kubiatowicz, who has created design methods aimed at collective layer of peer-to-peer systems, rather than at the individual nodes [33]. In a technique referred to as "Stability through Statistics," a form of statistical mechanics is used to design a system that can exhibit stable behavior by exploiting the "latent

¹ Note that the definition of aggregation provided does not relate to sensors, where the term aggregation has long meant the combination of two signals such as "aggregating data from two cameras."

order" of its multiple internal components. More specifically, by providing certain mechanisms, an environment can be created that is more stable than any of its parts. These mechanisms are:

Redundancy: More resources should be active than the minimum required for operation. For instance SETI@home, the largest distributed computing network in the world, has multiple computers to perform identical computations and excludes bad results through voting, thus providing stabilization despite unstable individuals

Replacement: Some technique must recognize failure, shutting down failing resources and switching to functioning ones. EmberNet is a company that aims to provide a self-organizing, self-healing, wireless embedded networking platform that can be used to monitor pipe temperatures to battlefield monitoring of hostile conditions [34]. Their system can optimize themselves to a task, and thus changing routing patterns when a particular organization reports a high failure rate.

Restoration: Some processes must act to reduce entropy and restore order. Restoring order calls for a constant process where a system spends energy to examine and improve its own functionality. One of the more successful methods to accomplishing this introspection has been the use of feedback loops, which devote spare computing resources to analyze system behavior, through methods such as Bayesian analysis.

If these mechanisms are used to create a stable aggregate of nodes, the current challenges of integration and availability (part of privacy) become easier to solve. To meet the challenge of integration, aggregation allows managing nodes at the level of the collective layer, allowing for an optimization of interactions at the system level. Thus, interactions become much easier to organize and monitor, as each individual connection does not have to be specially monitored. Similarly, the ability to ensure the system has a lower failure rate than any single node ensures availability.

3.2 *Tagging:*

Every entity in a complex system has a unique identity, a mechanism that facilitates proper interaction, selection and usage. This identity is represented through tagging. For instance, every web page possesses some address, and parts of the address (.gov, .com, etc.) can group like sites. To bring out Weiser's view of pervasive computing as a complex system, some mechanism of tagging is needed in pervasive computing.

Tagging is difficult to implement in pervasive computing because of the "one user, multiple computers" model of the paradigm. First, with the Internet a tag can only be hung on a virtual webpage, but due to the interactive and diverse nature of pervasive computing, tags on users, nodes and environments must be provided. Second, as pervasive computing mixes the physical and virtual world, tags may be placed on physical items, from humans to missiles to manufacturing equipment. Finally, the large numbers of devices that demand connectivity and the uncertainty of what nodes exist at any given time add complexity. Currently there is no uniform method of tagging, but some tagging methods are growing in popularity. For physical items, Radio Frequency ID (RFID) allows for the transmitting of a unique barcode to identify itself. For tagging environments, mobile computing has made progress in providing a common tag that represents the strength of a signal, and a similar method can represent the smartness of an environment. Lastly, many nodes use IP address to communicate to the Internet, such as a smart refrigerator that sends one's grocery list to the dealer. Many of these problems may be solved, however, with an interesting emerging trend in tagging that shifts to naming data from node rather than the node itself; as sensors can reorganize and move, often what is most important to facilitate selection and cooperation is the location and time data originated, not the sensor that relayed it [35].

In pervasive computing tagging can help solve the challenges of heterogeneity and security. With heterogeneity, tagging an area due to its "smartness" level can indicate the state a node should select. For instance, an area low on resources might send certain applications into a low power state, and shut off others not currently demanded by the user. In security, new techniques which use a form of tagging are being used to solve control issues. Trust-based security calculates a reliability rating to manage risk; in other words, by reading a certain tag, a node can tell how trustworthy another node is. Thus, a smart room can give certain access to a person with an id badge, showing how tagging leads to improved security. Tagging can also be used to secure physical locations, extending pervasive computing controls into that of physical objects. Henrici and Muller provide a good review of the use of RFID for security and privacy of locations [36].

3.3 *Non-linearity:*

As complex systems grow, their resulting emergent behavior cannot be decomposed into the behavior of its agents. This is a property unique to complex

systems, as non-complex systems simply are a magnification of its individuals, such a chorus that sounds like the magnification of a single voice. Nonlinearity is seen in Weiser's vision, as a pervasive environment cannot be linearly decomposed into just its devices and networks, but rather is a layer of computing made up from the interaction of these components. Therefore, environments must be explicitly designed to have nonlinearity.

To ensure that a pervasive environment is not directly reflective of its pervasive nodes, system behavior should be dissociated from node behavior. As application logic details behavior, separation of applications from the system must be accomplished. In their paper [37] Grimm et al. provides three ways to accomplish this. To separate the design of applications from its environment, common APIs can be created. Currently, multiple APIs lead to different common functions per platform, forcing development of different formats of the same application [38]. To separate distribution and installation of an application from its environment, a common binary distribution format can be used. The Java Virtual Machine (JVM) is an example of a common platform, albeit currently optimized for the desktop, which compiles to a binary format portable to any other java machine. Thus, a developer must only write an application once, and any device with a JVM can run the application [39]. Research is currently occurring to modify the concept of the virtual machine to the embedded framework, conquering the challenges of manageability, performance, security, and scalability [40]. Finally, to separate system data from functionality, new forms of encapsulation can be used in applications. As Grimm et al. state, for the definition of an application to a personal computer, it was beneficial to combine both data and functionality into one class, as class internals were changed more often than their relation to the rest of the code. In pervasive computing, however, these assumptions change. Companies usually compete on functionality, while large governing bodies define data formats that are relatively stable. Furthermore, objects are complicated and add overhead as well as lead to security breaches, rather sending a file (like a .pdf) that is reasonably safe. The end result is that data and functionality must be separated, allowing each to evolve separately and be stored separately. In short, by separating the design, deployment and data from application functionality, the behavior of an application becomes independent from the behavior or an overall system.

If designed properly, nonlinearity can help solve the scalability challenges of pervasive computing today.

By modularizing the application level from system functionality, applications do not have to be rewritten for each pervasive platform. As Saha and Mukherjee state in their explanation of the challenge of scalability, "Even if an enterprise could generate new applications as fast as it adds new devices, writing application logic only once- independent of devices- would have tremendous value in solving the applications scalability problem" [22].

3.4 *Flows:*

All complex systems contain methods for its agents to interact and exchange information. Clearly, by the paradigm's definition, Weiser's vision demands the flow of information between nodes to create a pervasive computing environment.

There has been much research devoted to creating smooth information flows in pervasive computing. To understand why implementing smooth information transfer in pervasive computing is so difficult, it is important to realize that pervasive computing information flow is fundamentally different from the traditional networks. Rather than using the network to connect computers when being used by people, pervasive computing entails device-to-device (D2D) connections, with a user directing or initializing the connection. And the devices do not have to be microcontrollers. Sensors may be spread throughout the environment, with a network needed to communicate an occurrence to backend infrastructure computers to process. Implementation attempts have largely taken on two forms [35]. First, self-configuring networks are created, largely using the design principle of aggregation, and using systems optimization to coordinate node schedules and attain the proper tradeoffs between fidelity, latency and efficiency. Second, research has increased on tiered architectures, which not only affect networks, but computing power and storage as well. In a tiered architecture, small elements with low computational power and bandwidth, allow for short-range sensing and operations. More powerful elements are used to process more complex information, such as digital signal processing.

Improvements in networks can help solve issues with invisibility and cost effectiveness. Flows can help solve invisibility by allowing information to transfer without needing human intervention. For example, self-configuring networks can be used to increase the time a network can go unattended [35]. In terms of cost effectiveness, proper flow will largely drop the price of pervasive computing environments. Intel director of research David Tennenhouse states that the price of wireless connectivity is still relatively

high for many tasks and that, "Radical innovation will be required to bring networking costs in line with \$1-per-device price structure of the embedded computing market" [13]. Tiered architectures have helped address this problem by utilizing cheaper networks and computation power for short-range information transfer, while core networking technologies improve and drop in price.

3.5 *Diversity:*

All complex systems also contain varying agents that differ both in their role and behavior. Still, at the collective level, these differences disappear. For instance, while a rainforest contains many trees, at the collective level, what is seen is simply a single forest. This property is also a necessary objective in pervasive computing. While multiple different nodes with varying behaviors may exist in an environment, the collective layer of computing must be invisible to the user.

One method to build diversity is to develop applications that depend very little on a stable, predictable environment, and thus embraces the natural diversity in pervasive computing. This is in contrast to the design methods used today to deal with remote resources that were based on the personal computer, where the failure or unavailability of a resource, such as network, was seen as an extreme case with little expectation of change. In fact, from a programmer's point of view today, diversity is so hidden that remote resource calls are seen the same as local resource calls through masking. To remedy this and embrace the diversity of pervasive computing, adaptive computing techniques must be used. One specific technique calls for applications to explicitly bind all resources (data, communication, etc.) they use [37]. In this method, leases are used to control the bindings, which demand that application renew them and provide timeouts to avoid locking onto an unavailable resource. By using late binding, applications lessen their dependence on a stable environment, similar to allowing an application to avoid type declaration and thus not bind a function to a certain data type.

By using these methods, applications can begin to be more adaptive to their environment, and thus better prepared to handle the heterogeneity inherent to the field of pervasive computing. By assuming that smartness of environments will never be consistent, the principle of diversity acknowledge that heterogeneity will be present and finds ways to constantly search and adapt to new situation. This all happens behind the scenes from the user, providing a consistent front for the entire system.

3.6 *Internal Models:*

Every agent in a complex system follows internal rules that produce certain behaviors and guide interactions with its environment. For instance, in an ecosystem, every creature has rules that help it find food and avoid hazards. Implicit models are hard-wired rules of behavior, while explicit rules are representations, stored in memory, that allow an agent to explore alternatives to each situation. These models act on information from the environment, and thus are a large part of the field of context sensitivity, or a software system's ability to sense and analyze context from various sources [41]. In pervasive computing, both mechanisms of internal models must be in each device application to allow it to adjust and adapt around the user.

Implicit models are implemented through hardware at the transistor level, while explicit models can be implemented by software states. For example, an internal model may contain a rule to use as little power as needed. This can be hardwired at the transistor level, with designs that have less leakage of electricity and less distance for current to travel have decreased power consumption, such as Motorola's MC9328MX1 with Dual Vt and Well-biasing [42]. Conversely, more complex architectures can be created that contain states that provide a given service level with needed energy consumption. Using intelligent software, these states can be selected based on lowest power needed for a certain task [10].

Both explicit and implicit models can help solve the problems of invisibility, with adjusting services provided and tuning themselves without user intervention. Furthermore, as complex environments continue to grow and nodes become more autonomous, the ability to predict all the possible scenarios that a node will encounter will be greatly reduced. To keep systems reliable, internal models will be depended on to ensure that a task operates correctly. Thus, the creation of internal models is a key principle in pervasive computing.

3.7 *Building Blocks:*

The final characteristic seen in every complex system is building blocks or simple components that combine to produce diverse agents. This mechanism can be seen in the human body, where similar cells combine to produce vastly different human beings. It should also be noted that natural selection limits the number of random combinations of these blocks to direct behavior in all systems. Thus to truly create a pervasive environment, building blocks are needed to

allow the fast growth and optimization of the paradigm.

One of the most effective methods to implement building blocks is through services. Services are implemented in middleware and provide functionalities that an application can use independent of the platform on which they reside. In separating hardware from applications, services allow both reusability of code and simplify development by providing primitives to the developer. Prasad and Nelson identify three types of services to demonstrate the variety and adaptability of the idea [43]. The first is a service to perform an action, for instance setting the volume in a radio. The second is a subscription to periodic updates of information. Finally, a service can provide notification of an event. By combining these basic services, complex software architectures can be created quickly. Furthermore, the market, a form of natural selection that ensures limits and directs the way services combine to larger entities, directs the combinations of these services.

By providing a method for the quick creation of pervasive nodes from basic devices, the challenges of scalability and cost effectiveness are minimized. This ability for a developer to reuse tested components allows for much quicker generation of code, similar to the reuse of object and classes in object-oriented programming. The need to not rewrite basic components further allows for decreased software development costs, lowering the price of creating and replicating nodes in pervasive computing.

4. CONCLUSIONS

Pervasive computing is an exciting field that is primed for future growth. To meet emerging demands, however, both the technical and societal challenges of the paradigm must be solved. Currently, research to find solutions to these challenges follows the traditional application-based framework; thus, some form of top-down design principles is used. In Mark Weiser's vision of pervasive computing, however, an environment of computing is formed with no end goal, and thus application development methods are largely insufficient.

Complexity Theory is presented as a framework that is more suited to pervasive computing, as it allows for bottom-up integration. By translating the seven core characteristics of complexity theory into design principles, it is shown that many of the challenges of

the pervasive computing today can be better met. Examples of how to implement the principles are also provided.

It is the hopes of the authors that the framework and design principles present can be translated into a formal Solution Delivery Methodology to successfully guide the creation of future pervasive computing environments.

5. REFERENCES

- [1] About pervasive computing, National Institute of Standards and Technology. Available: http://www.nist.gov/pc2001/about_pervasive.html (2001).
- [2] E. Lee, What's ahead for embedded software, IEEE, pp 18-26 (2002).
- [3] Seattle Land Use Zoning Code, City Department of Planning, Seattle, WA. Development. Available: <http://www.cityofseattle.net/dclu/CityDesign/DesignEducation/Glossary/>
- [4] M. Weiser, The computer for the twenty-first century, Scientific American, pp. 94-10 (1991).
- [5] http://www-306.ibm.com/software/pervasive/business_partners/proposition.shtml
- [6] Canalys Research Firm, Global smart phone shipment treble in Q3, Canalys Press Release, Oct. 2004. Available: <http://www.canalys.com/pr/2004/r2004102.htm> (2004).
- [7] <ftp://ftp.software.ibm.com/software/pervasive/info/BPIIntro.pdf> (META group)
- [8] <http://db.maxim-ic.com/ibutton/applications/index.cfm?Action=DD&id=8>
- [9] A. Menon, ERP in singapore- A perspective one year on, TEC Volume 41, Issue 2 (2000).
- [10] R. Want, G. Borriello, T. Pering, K.I. Farkas, "Disappearing hardware," IEEE Pervasive Computing, vol. 1, no. 1, Jan.-Mar. 2002, pp. 36-47.
- [11] <http://support.intel.com/support/processors/mobile/pentiumiii/sb/CS-007530.htm>
- [12] http://domino.research.ibm.com/comm/pr.nsf/pages/news.19990222_chip.html
- [13] M. McCullough, Digital Ground, Cambridge, MA: The MIT Press (2004).
- [14] <http://www.instat.com/press.asp?ID=503&sku=IN020017MI>

- [15] D. Estrin, D. Culler, K. Pister, G Sukhatme, "Connecting the physical world with pervasive networks, IEEE Pervasive Computing, pp 59- 69 (2002).
- [16] North American automotive original equipment sensors market, Frost and Sullivan (2003).
- [17] R. Kurzweil, Innovation in an era of accelerating technologies, presented at the Second Annual Emerging Technologies Conference at MIT, Cambridge, MA (2004).
- [18] M. La Monica, "Supply chain reaction," CNET.news, Sept. 2003. Available: http://news.com.com/2009-1008_3-996043.html
- [19] M. Yokotsuka, Memory motivates cell-phone growth, Wireless System Design Magazine, April 2004. Available: <http://www.wsdmag.com/Articles/ArticleID/7916/7916.html>
- [20] Hitachi Microdrive 3K4 digital media, Hitachi Global Storage Technologies, San Jose, CA (2003).
- [21] M. Williams, Toshiba has big plans for small drive, PCWorld.com, Sept. 2004. Available: http://www.pcworld.com/news/article/0,aid,117952,0_0.asp
- [22] D. Saha, A. Mukherjee, Pervasive computing: A paradigm for the 21st century, Computer, pp. 25 –33 (2003).
- [23] E. Nelson, K. Prasad, V. Rasin, and C. Simonds, An embedded architectural framework for interaction between automobiles and consumer devices, the proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, Canada (2004).
- [24] M. Satyanarayanan, Pervasive computing: Vision and challenges, IEEE Personal Communications, (2001).
- [25] Let the circle be unbroken, Harbor Research Inc, (2003).
- [26] L. Rohed, "Nokia phone adds virus protection" PCWorld.com, Sept. 2004. Available: http://www.pcworld.com/news/article/0,aid,117904,0_0.asp
- [27] L. Kagal, T. Finin and A. Joshi, Trust-based security in pervasive computing Environments, IEEE Computer, pp 154-157 (2001).
- [28] F. Stajano, Security for whom? The shifting security assumptions of pervasive computing, proceedings of International Security Symposium (2002).
- [29] J. Thomson, European automakers to jointly develop emergency call system, WardsAuto.com. (2004).
- [30] G. Abowd, E. Mynatt, and T. Rodden., The human experience, Pervasive Computing, pp. 48-57 (2002).
- [31] D. Lankes, Building and maintaining internet information services, Dissertation, School of Information Studies, Syracuse University, Syracuse, NY (1998).
- [32] J. Holland, Hidden Order, Reading, MA: Addison Wesley Publishing Company, 1995.
- [33] J. Kubiawicz, Extracting guarantees from chaos, Communications of the ACM- Vol.46, No. 2 (2003).
- [34] <http://www.ember.com/successes/applications.html>
- [35] D. Estrin, D. Culler, K. Pister, G Sukhatme, Connecting the physical world with pervasive networks, IEEE Pervasive Computing, pp 59- 69 (2002).
- [36] D. Henrici, and P. Müller, Tackling security and privacy issues in Radio Frequency Identification Devices, the proceedings of the 2nd International Conference on Pervasive Computing, Linz/ Vienna, Austria (2004).
- [37] R. Grimm, Systems directions for pervasive computing, the proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Elmau, Germany, pp. 128-132 (2001).
- [38] F. Dabek, B. Zhao, P. Druschel, J. Kubiawicz and I. Stoica, Towards a common API for structured P2P overlays, Presented at IPTPS 2003, Berkeley, CA (2003).
- [39] <http://java.sun.com/docs/books/vmspec/>
- [40] E. Sirer, R. Grimm, A. Gregory, and B. Bershad, Design and implementation of a distributed virtual machine for networked computers, Operating Systems Review, 202-216 (1999).
- [41] S. Yau, F. Karim, Y. Wang, B. Wang, and S. Gupta, Reconfigurable context-sensitive middleware for pervasive computing, IEEE Pervasive Computing, pp. 33-39 (2002).
- [42] M. Moore, N. Marshall, and B. Bone, Creating performance with stamina for wireless communications, Motorola Freescale Semiconductor White Paper (2004).
- [43] K. Prasad and V. Nelson, Automotive infotonics. An emerging domain for service-based architecture. The 2003 Formal Methods Europe Symposium Workshop on Service Based Software Engineering (SBSE2003), Pisa, Italy (2003).
- [44] <http://www.media.mit.edu/pia/Research/Hyphos/>