Towards a Pervasive Computing Benchmark[1]

Anand Ranganathan, Jalal Al-Muhtadi, Jacob Biehl, Brian Ziebart, Roy H. Campbell, Brian Bailey
University of Illinois at Urbana-Champaign
{ranganat, almuhtad, jtbiehl, bziebart, rhc, bpbailey}@uiuc.edu

Abstract

Pervasive computing allows the coupling of the physical world to the information world, and provides a wealth of ubiquitous services and applications that allow users, machines, data, applications, and physical spaces to interact seamlessly with one another. In this paper, we propose a benchmark for evaluating pervasive computing environments. These proposed metrics facilitate the assessment and evaluation of different aspects of pervasive computing and its support for a wide variety of tasks.

1. Introduction

Pervasive computing is a new research field that encompasses a variety of elements from different disciplines including distributed systems, sensor networks, mobile computing, databases, AI, HCI, security, and networking. These technologies converge to deliver ubiquitous access to data and services and boost productivity. While there are standard metrics for evaluating more traditional fields like HCI [5, 6], system performance [7, 8], and software engineering [9], there are no standard metrics for evaluating different kinds of pervasive systems [1-4]. Pervasive computing introduces new ways of interacting with and using computers. Hence, new schemes for assessing and evaluating pervasive computing are required to guide the design and implementation of such systems. In this paper, we identify a number of metrics for the purpose of evaluating pervasive computing environments. For evaluation purposes, any pervasive computing framework can be divided roughly into three layers: system support, application programming support and end-user interface. Respectively, the metrics we identify in this paper can be categorized into three categories: system, programmability and usability metrics. Some of these metrics are unique and applicable only in pervasive computing environments; while others are based on metrics identified in other areas, which we extend and adopt for our purpose. The metrics identified in this paper are not exhaustive. However, we believe that they present suitable guidelines for steering the development of new systems and measuring existing ones.

Traditional benchmarks often measure system performance by observing performance on standardized task sets. Different systems are compared by how well they deal with these task sets. Because pervasive computing is in its infancy, there is no widely accepted notion of the task sets that would form a benchmark. Moreover, different pervasive computing environments have been built for supporting different kinds of tasks. Hence, it becomes difficult to compare them. To overcome this problem, we identify four broad classes of task, which many environments would support one or more of these tasks. We try to measure how well a pervasive computing environment supports these tasks in terms of usability, system and programmability metrics. The broad classes of tasks that drive our benchmark are: (1) Presentation tasks: involving displaying and navigating information (such as slideshows or web pages) (2) Notification/Trigger-based tasks: involving sending notifications to users or performing actions that are triggered by some condition. (3) Collaboration tasks: involving multiple users working together to achieve a common goal. (4) Information Finding tasks: involving finding information about users and resources.

In Section 2, we evaluate the system support for pervasive computing in terms of context-sensitivity, security and discovery metrics. Section 3 describes configurability and programmability metrics. Section 4 describes usability metrics. Section 5 concludes the paper.

2. System Metrics

Our metrics for evaluating the system support for pervasive computing cover the areas of context sensitivity, security and discovery. We did not include some of the more traditional metrics involving network, operating system or database performance since there are already well-established metrics for evaluating these aspects.

2.1 Context-Sensitivity Metrics

A key aspect of pervasive computing is sensing the current context and user goals and then using this information while helping users perform various tasks. Thus, context-sensing and context-based adaptation form a key dimension while evaluating pervasive environments. We split the evaluation of context-sensitivity into two parts: (1) Evaluating the quality of sensed or derived context information. (2) Evaluating the use of context information for enhancing the four different kinds of tasks.

In order to evaluate context-sensitivity, we define a taxonomy of the different kinds of contexts. Some of these contexts are sensed directly, while others are inferred from other sensed contexts. Our taxonomy consists

[1] This research is supported by grants from the National Science Foundation, NSF CCR 0086094 ITR and NSF 99-72884 EQ

of: Location, time, environmental contexts (temperature, light, sound level), informational contexts (stock quotes, sports scores), user contexts (gaze, orientation, health, mood, schedule, activity), group contexts (group activity, social relationships and networks, other people in a room), application contexts (email received, websites visited, previous files opened), system contexts (network traffic, status of printers), and physical object contexts (position or orientation of chairs and tables, properties of objects such as temperature and size). The quality of the above contexts can be evaluated using one or more of the following four metrics:

1. Confidence – expressed as a probability that the context has been sensed or deduced correctly
2. Accuracy – expressed as an error percentage of the sensed or inferred contexts.
3. Freshness – measured as the average time between readings of a certain kind of context.
4. Resolution – the area within which location information can be narrowed down to (room-level, building-level, etc.)

Context information can be used, proactively or reactively, for enhancing different kinds of tasks. We need metrics for evaluating how well context information is used. This involves comparing the same task in the pervasive environment when it uses and does not use context information. For example, a presentation application can be manually configured by the user for the current context (such as number and location of attendees, kind of presentation, displays available, etc.), or the application may configure itself automatically. The metric we propose for evaluating context-enhanced presentation tasks is the reduction in the number of configuration actions the user has to take. Similarly, context can enhance an information-finding task by automatically augmenting the query or filtering the results based on the location of the user or his current activity. Context can also be used to adapt the interface based on the kind of information being requested. The metric in this case is the reduction in the number of constraints the user has to specify in his query or the reduction in the number of actions the user has to take to get the desired information. Table 1 evaluates how context is used for enhancing different kinds of tasks.

Besides the above metrics, there are other aspects of context sensitivity that are more difficult to quantify and evaluate. For example, the overhead in deploying sensors for sensing contexts is difficult to quantify. We are still investigating ways of incorporating such metrics.

Researchers have recognized the fact that context information can have varying quality. For example, [10-14] allow location and context information to be associated with quality metrics, such as freshness, accuracy and confidence. While the quality of context information is an important metric, it is equally important to evaluate whether context information is being used to enhance various kinds of tasks. Hence, in our metrics, we try to cover both the quality of context as well as it's usage in various kinds of tasks.

Table 1. Evaluating context-based adaptation of tasks

Kind of Task	Context-Enhancement Metric
Presentation Task	Reduction in number of configuration actions that user has to take to configure environment in a context sensitive manner
Trigger-based/Notification Task	Reduction in number of times user was disturbed or annoyed by a proactive action taken by the system or by a notification. This is measured based on user feedback.
Collaborative Task	Reduction in number of configuration actions. Also, enhancement of seamlessness of interactions; ease of information retrieval, versioning and archiving processes measured by user feedback.
Information-Finding Task	Reduction in number of steps that user has to take to get some information or the number of parameters that user has to enter in his query.

2.2 Security Metrics

Addressing security and privacy issues in pervasive computing is vital to the real-world deployment of the technology. The security metrics we identify here try to gauge the ability of the security services to handle the ubiquity, context sensitivity, and rapid evolvement of the pervasive environment. We identify the following metrics:

1. Expressiveness of Security Policies: We measure the expressiveness of a security policy by its ability to incorporate the following in the policy's rules. 1) *Support for mandatory and discretionary rules*. Typical pervasive computing environments are composed of a tapestry of public spaces, devices, and resources, as well as personal devices and gadgets. Therefore, it is essential to be able to support mandatory policies set by the space administrators, as well as accommodating policies defined by users for their personal devices. 2) C*ontext sensitivity.* Security rules of a pervasive computing environment may vary according to the context of the space. Hence, the security policy language should be able to incorporate rich context information. 3) *Uncertainty handling.* Often, context information is not precise. Policies should be expressive enough to define how to act under imprecise or incomplete context information. 4) *Conflict resolution.* Expressive policies have the potential to conflict with each other, particularly when different users are allowed to set policies. Some mechanism for handling conflicts is necessary.

2. User control over private information: The physical outreach of pervasive computing makes preserving users' privacy a difficult task. Mechanisms are needed to give users control over their private information and how and when it can be disclosed. Cooper et al. [15] identify three kinds of privacy: content, identity, and location. *Content*

privacy is concerned with keeping data or content private. *Identity privacy* is concerned with hiding the identity of the user. *Location privacy* is concerned with hiding the location of the user. Our proposed metric takes into account the three different kinds of privacy (Table 2).

3. Unobtrusiveness of security mechanisms: Pervasive computing attempts to provide a seamless user-centric environment, where users no longer need to exert much of their attention to computing machinery. Therefore, the security subsystem should provide mechanisms that allow security services, like authentication for instance, to become transparent to some level, blending into the background without distracting users too much.

2.3 Discovery Metrics

An important facet of any large scale distributed system is discovery, or the ability to find resources that meet certain requirements. Different discovery protocols have been proposed and used such as CORBA's Trading Service, Jini's Lookup Service, INS, Salutation, etc. Besides, different pervasive computing environments have their own methods for service and device discovery. Our metrics for evaluating discovery protocols for pervasive computing environments consist of the following elements:

1. *Precision and Recall*. Precision is defined as the percentage of correct hits among all the answers returned. Recall is defined as the percentage of correct hits returned out of all the correct hits that exist in the environment.
2. *Context-Sensitivity*. Does the discovery protocol take into account the context of the requestor, the service or of the environment while matching requests to provide contextually appropriate results?
3. *Semantics*. Is the semantics of the query used, or is query-answering based solely on syntax (e.g. keywords)?
4. *Scalability*. Does the protocol scale for large-scale environments with a large variety of services and devices? This is measured based on the number of requests and advertisements that the system can handle per unit time.

3. Configurability and Programmability Metrics

An important metric for evaluating pervasive computing environments is the ease with which new applications and services can be developed and existing applications and services can be configured. We propose metrics that differentiate between end-users and developers. Developers have programming expertise, enabling them to create or modify applications and services. End-users normally lack these abilities and prefer using simple graphical interfaces to configure applications and services.

3.1 Application Properties

To measure the programmability and configurability of pervasive computing environments, we identified properties of applications in these environments that distinguish them from traditional desktop applications. These properties include multi-device partitioning, mobility, composability, context-sensitivity and automation. Our benchmark measures how easy it is for developers to program and for end-users to configure these properties for an application that performs one of the four broad classes of tasks described earlier (i.e. presentation, notification, collaboration, and information finding tasks).

Multi-device adaptation and partitioning: In a pervasive computing environment with many computing devices for each user, confining application interaction to a single device is overly restrictive. Applications that span multiple devices allow users access to a wider scope of interaction with the environment. An example of multi-device partitioning is allowing a user to control a slide-show on a wall-mounted plasma screen using a handheld device. Additionally, applications may need to adapt in order to run on a different device. We evaluate the effort required to support an application on a different device.

Application mobility: Many pervasive environments support mobile applications that can move with the user. Some of the issues in application mobility are maintaining consistent state and adapting to different devices. A related aspect is replicating an application across multiple devices, which is useful in collaboration applications.

Application and Service Composition: Composition allows components to be used in a variety of new tasks and also increases the reusability. A number of different approaches have been suggested for composition. These approaches can be evaluated in terms of the kinds of compositions they allow – sequential, parallel, conditional, recursive, manual, automatic, etc.

Context-Sensitivity: Applications may need to behave differently depending on the current context of the environment. Different infrastructures allow developers and end users to specify context-sensitive application behavior in different manners and through different languages. Our metric for evaluating context-sensitivity is the power and expressivity of the language used for specifying context.

Automation: Many pervasive systems try to configure applications and services based on users' preferences automatically. Common approaches include learning and user-specified macros or scripts describing what actions should be performed automatically. If learning is used, then the metric is the percentage of times correct decisions were made with regard to automation. If the user can use tools or scripts to specify automation, then the metric is based on the ease of use of the tool. Another metric is whether automation can be learned or specified in a context-sensitive manner.

3.2 Measurements

Pervasive environments generally provide application developers with libraries, frameworks or toolkits for supporting the above-mentioned properties in their applications. We measure their effectiveness using the traditional

metrics of man-hours and lines of code. Our programming metrics include the number of lines of code and man-hours that are required to create a new application performing one of the broad task classes. It then measures the number of additional lines of code and man-hours required for supporting a certain property for this new application. Alternatively, if GUI-based tools are available for developers, lines of code can be replaced by actions required on the GUI. End-users normally use GUIs or simple scripts to specify the desired properties. We evaluate the support offered by a pervasive environment to an end-user based on the level of skill required to configure the application, using the following scale: (1) Simple end user GUI. (2) Experienced user GUI. (3) End user command line. (4) Scripting language. (5) Advanced programming or source-code editing. Table 2 has a summary of our programmability metrics and also suggests how they can be measured.

4. Human Usability Metrics

In designing usable pervasive environments developers must consider both old and new usability challenges. As in traditional usability design, attention to user's performance (time to complete task), rate of error, recoverability from error and satisfaction are key factors in the assessment a system's usability. While these metrics and others [5] are still valid, by themselves, they fail to provide a complete assessment of a pervasive environment's usability. In these environments, the traditional interaction metaphor of one user to one computer is broken. Users are now interacting with multiple technologies while simultaneously collaborating with their co-located peers. They are moving about the environment, constantly refocusing attention while manipulating and relocating data across devices. New interaction metaphors present new challenges for measuring the usability of the system. Through observations of users collaborating in our Active Space we propose the following metrics for evaluating the usability of pervasive environments:

Head turns: We found head turns to be strongly correlated to how much a user's attention is divided across the workspace. Changing the focus of attention can be physically and mentally taxing and can quickly frustrate a user. The number of head turns can be measured easily and can help to provide designers with an understanding of an environment's ability to effectively interact with users.

Physical movement: Physical movement that is not a direct part of a user's task is superfluous, time consuming and causes interruption in users' thought process. The nature of pervasive computing is to have computing resources everywhere; excessive physical movement is contradictory to this model. Measuring the time a user spends moving in the space auxiliary to their main task helps designers assess the effectiveness of the system.

A priori user knowledge: One of the essential goals of pervasive environments is to provide interaction techniques that require little to no prior knowledge from the user. The best interfaces are those in which world knowledge provides enough understanding for a user to interact with the system [6]. Measuring the number of facts that the user has to know in order to perform a task is valuable in assessing the difficulty and learnability of the system's interaction techniques. For example, consider speech based interfaces where a user must first learn the system's vocabulary before being able to use it. Here the comprehensiveness of the vocabulary can be measured as a quantifiable number of facts required for successful interaction.

These metrics are not intended to be comprehensive, but rather complement traditional usability metrics to create a stronger, more extensive assessment of pervasive usability. Traditional metrics such as error rate, task completion time, and subjective satisfaction are still valid in these environments because they measure human characteristics which, with respect to traditional interfaces, are just as relevant to pervasive environments.

5. Conclusion and Discussion

In this paper we presented various metrics for evaluating pervasive computing environments. We derived these metrics from the result of four years of building and improving our pervasive computing system [4] and from performing various usability studies [16]. As an example of the kinds of results produced by our proposed benchmark, we evaluated our pervasive system using some of the metrics in our benchmark. The results of our evaluation are in [17]. We have also subsequently improved some features of our environment based on evaluation results. We believe that the benchmark will be broadly applicable to different environments because the broad goals of pervasive computing are common across most environments. While the list of metrics we identified are not exhaustive, we believe that they contribute towards clarifying a lot of the ambiguity that exist in evaluating existing systems, and give a good sense of direction for new pervasive computing infrastructures.

The classes of tasks we chose to be a part of our benchmark are meant to be a representative set of tasks in pervasive environments. We recognize that all pervasive environments may not support all kinds of tasks and that some environments may support other kinds of tasks as well. However, we believe that identifying such common tasks and evaluating pervasive environments based on these tasks is necessary in order to compare different environments. We also realize that some metrics are less precise and more difficult to measure than others. We hope that as we continue to use these metrics, these limitations can be overcome.

We invite feedback from other researchers in terms of important metrics that may be missing in our benchmark as well as alternative ways of measuring them. We also invite other researchers to evaluate their environments using our proposed benchmark to test the broad applicability of the benchmark. We are also working on expanding our benchmark to include other aspects such as fault-tolerance.

Table 2. Summary of Security, Programmability and Usability metrics and their Units of Measurement

	Metrics		Unit
Security	Expressiveness of the security policy		We identified 4 different features for security policy expressiveness. We measure this metric by using a value of 0-4, representing the number of features supported.
	User control over private information		0-3, where 0 = no control provided. 1 = system provides control over the disclosure of one kind of information (content, location, or identity), 2 = system provides control over two kinds of information. 3 = system provides control over all three kinds of information.
	Unobtrusiveness of security mechanisms		% of time used for interacting with the security subsystem (e.g. authentication) auxiliary to the main task
Programmability	Creation	New Application	Man Hours and/or lines of code
		Supporting Additional Devices	Additional Man Hours and/or lines of code
	Mobility	Programming support	Additional Man Hours and/or lines of code
		End-User ease of moving	1-5*
		End-User ease of replicating	1-5*
	Composition	Programming support	Man Hours and/or lines of code
		End-User ease of use	1-5*
		Expressivity	Kinds of compositions allowed
	Context Sensitivity	Programming support	Man Hours and/or lines of code
		End-User ease of use	1-5*
		Expressivity	Kind of logic used to specify rules
	Automation	Percent of user actions automatically reduced	0-100%
		Percent of correct automation decisions	0-100%
		End-user ease of use	1-5*
Usability	Head turns		Total number per task
	Physical Movement		% of time used for movement auxiliary to the main task
	A priori user knowledge		Total number of facts required to be known by the user to perform task
	Keystrokes, clicks, and other atomic input		Total number per task
	Error and Error Recovery		Total number of errors, and time spent recovering from error
	User Satisfaction		Subjective (1-5) scaling (5 = most agreement)

* based on end-user support scale in Section 3.2

6. References

[1] A. Fox, et al "Integrating Information Appliances into an Interactive Workspace," *IEEE Computer Graphics & Applications*, vol. 20, 2000.

[2] R. Grimm and e. al., "A system architecture for pervasive computing," In 9th ACM SIGOPS European Workshop, 2000.

[3] D. Garlan, at al, "Project Aura: Towards Distraction-Free Pervasive Computing," in *IEEE Pervasive Computing*, vol. 1, 2002, pp. 22-31.

[4] M. Roman, et al, "Gaia: A Middleware Infrastructure to Enable Active Spaces," *IEEE Pervasive Computing Magazine*, vol. 1, pp.74-83, '02

[5] M. Rauterberg, "Four different measures to quantify three usability attributes: 'feedback', 'interactive directness' and 'flexibility'," in Design Specification and Verification of Interactive Systems -- DSV-IS'95.

[6] D. Norman, *Design of Everyday Things*: Basic Books, 2002.

[7] D. DeWitt, "The Wisconsin Benchmark: Past, Present, and Future," *The Benchmark Handbook*, 1991.

[8] "SPEC benchmarks." http://www.spec.org/cpu2004/.

[9] B. Boehm, *Software Engineering Economics*: Prentice Hall

[10] M. Ebling et al, "Issues for context services for pervasive computing," in Middleware 2001 Workshop on Middleware for Mobile Computing, Heidelberg, 2001.

[11] P. Castro et al, "A probabilistic room location service for wireless networked environments," in UbiComp, 2001.

[12] A. Schmidt et. al, "Advanced interaction in context," in HUC 1999.

[13] P. Gray and D. Salber, "Modelling and using sensed context in the design of interactive applications," in 8th IFIP Conference on Engineering for Human-Computer Interaction, 2001.

[14] K. Henricksen et.al, "Modeling context information in Pervasive Computing Systems," in Pervasive 2002

[15] D. A. Cooper and K. P. Birman, "Preserving Privacy in a Network of Mobile Computers," IEEE Symposium on Research in Security and Privacy, 1995.

[16] J. Biehl, and B. Bailey. ARIS: An Interface for Application Relocation in an Interactive Space. in Graphics Interface 2004,

[17] A. Ranganathan, et.al. Evaluating Gaia using a Pervasive Computing Benchmark. UIUC technical report.