Context Awareness: an Experiment with Hoarding

João Garcia, Luís Veiga, and Paulo Ferreira

Distributed Systems Group, INESC ID Lisboa, Portugal
http://www.gsd.inesc-id.pt/
{jog,lveiga}@gsd.inesc-id.pt, paulo.ferreira@inesc-id.pt

Abstract. Computer mobility allows people to use computers in varied and changing environments. This variability forces applications to adapt thus requiring awareness of the computational and physical environment (e.g. information about power management, network connections, synchronization opportunities, storage, computation, location-based services, etc.).
An important application for mobility is hoarding, i.e. automatic file replication between devices. To be accurate and not obstructive to the user, the hoarding mechanism requires both context awareness (e.g. amount of usable storage) and estimation of future environment conditions (e.g. network connection, tasks to be performed by the user in the near future, etc.). However, making applications context-aware is hindered by the complexity of dealing with the large variety of different modules, sensors and service platforms, i.e. there is no middleware supporting such applications and their development in a uniform and integrated way.
This paper presents the architecture for an environment awareness system (EAS) and how it applies to hoarding. EAS is a middleware component that acts as an intermediary between applications and all mechanisms that assess the surrounding environment. It lets applications query and combine environment properties in a standardized way. Crucial for the success of automatic file hoarding is the EAS's capability of supporting environment prediction based on simple reasoning and pattern detection. Thus, applications may advise users accordingly or even make decisions on their behalf.

1 Introduction

Mobile computer technology has led people to use their computers in a wide variety of environments, e.g.: a PC at the office, a laptop at the airport, a PDA in a taxi, etc. Users want to work continuously in this data ubiquitous world taking advantage of available resources and not worrying about any system problem that may occur (such as missing files).

Achieving such ubiquity is hard and depends on many applications. Automatic file replication between devices, i.e. hoarding, is a solution for the fundamental problem of data availability in mobile environments. To be accurate

while not obstructive to the user, the hoarding mechanism requires both context awareness (e.g. knowing the amount of available memory) and estimation of future environment conditions (tasks to be performed in the future, etc.).

However, creating or adapting applications, like file hoarding systems, is encumbered by the variety of different modules, sensors and service platforms. This is due to the absence of middleware supporting such applications and their development in a uniform and integrated way.

Furthermore, users want to take advantage of any resources as they move around (for example cheap wireless connections). Therefore, the increasing geographical mobility of devices and the mobility of data among devices require applications to be aware of the environment around them and its risks and opportunities. An optimal evaluation of the context should include not only current conditions but also conditions that may be found in the future. For example, if a user is leaving her office, she wants to take along on her laptop all files needed in order to keep on working on her ongoing tasks. But, if her personal computer knew that she is leaving for a meeting related to a specific task, it would only transfer files related to that task.

Power management is another domain where knowing resource availability is highly relevant: if a device were aware of how long its batteries are supposed to last, it could adjust its energy consumption accordingly.

Fulfilling the above mentioned users expectations raises two major problems for application development and execution:

- The heterogeneity of networks, sensors, platforms and services increases the difficulty of building such applications, making a middleware layer supporting applications and their development in a uniform way clearly desirable;
- Current approaches don't take into account users' past habits and future actions. Thus, developers of context-aware applications and users in general must take a large number of decisions concerning the best usage of computational and physical resources.

Existing methods to provide applications with structured context information[1, 2], either limit the information provided to applications to specific domains (relative location, user/device identification) or require programmers to modify their applications each time they want to query a new environment property.

This paper presents the architecture for an environment awareness system (EAS) and how it applies to automatic file hoarding. EAS is a middleware component that supports the interaction between applications and any computer-based mechanism able to provide clues regarding the surrounding environment. It lets applications query and combine environment properties in a standardized way by means of an API providing access to the device's context sensors in a uniform way. Each sensor is represented by an environment perception module (EPM), which is a software component capable of polling and/or forecasting a specific environmental property.

In addition to providing a framework for existing and future EPMs, the EAS enables many synergies between EPMs by aggregating and/or applying logic

expressions over data from several EPMs. Most important, an EAS is able to forecast future conditions and/or user actions by detecting patterns in the time series of past conditions and maintains a history of past, scheduled and forecast environment conditions. The EAS analyzes personal information manager data and the history of past events to estimate future location, future user activities, etc... For instance, in the case of automatic file hoarding, the EAS is designed to compare the subject of a user's meetings and the file access patterns during previous meetings with the keyword on her files to determine which files need to be replicated to a user's laptop for the meetings on her immediate schedule.

In summary, managing personal files effortlessly is one of the major problems raised by an environment where users have multiple mobile devices. Automatically replicating files among devices, hoarding, can be greatly aided by an EAS because it enables a comparison between users' file accesses and specific environment conditions. Additionally, since the EAS provides an estimation of future conditions, this can be used to decide which data will be most useful in the predicted future.

In the rest of this paper, we begin by presenting the architecture of the EAS. Then, we list a number of EPMs that could currently be integrated into an implementation of an EAS. We show how the EAS is applied to file hoarding and compare our design with existing systems. Finally, we present some of our conclusions and future work.

2 Architecture

The environment awareness system (EAS) is a middleware component, which provides applications with a simple and structured mechanism to query a device's computational and physical environment.

Applications can perform queries on the current situation or request callbacks when certain conditions are met. Queries return an indication of whether an environment property is within a certain range of values.

An EAS event is a timestamped set of property-value pairs representing a change in environment properties. Each property, such as "AbsoluteLocation" or "NetworkConnectivity", is detected by a particular sensor. Each attribute has a domain describing the values it can assume, e.g. "NetworkConnectivity" can be "None", "Poor", "Medium" or "Good". A value describes the status of a property such as "443.23N 217.98W" (a possible value for "AbsoluteLocation") or "45 min" (a "BatteryTime" value). Queries and callbacks can be directed to the local device or to a remote device.

Additionally, the EAS lets programmers specify that certain conditions are to be associated with a particular label: for example, assigning GPS positions to known locations ("home", "office", etc.) or particular circumstances to specific activities (being in room 13 on Monday morning means the person is in a "Staff Meeting"). Users can submit these labeled situations, which are stored by the system and can be referred to in subsequent queries. The hierarchical organization of environment properties and events combines on one hand a simple

way of manipulating information about the hardware and, on the other hand, an expressive mechanism to describe situations to which applications want to respond.

The EAS (Fig. 1) is composed of:

- An API layer accessible to applications,
- A callback registry,
- A schedule of EPM probe actions,
- A repository of past, scheduled and forecast events,
- A forecasting model for calculating probable future conditions,
- Several environment perception modules (EPMs),
- A remote invocation interface.

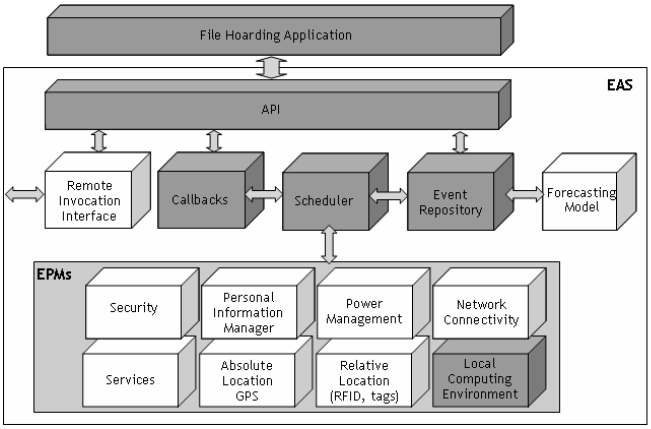

Fig. 1. A file hoarding application on top of the EAS (dark grey boxes were implemented in the prototype of Sec. 3).

Callbacks Callbacks allow applications to be notified of future conditions. They are associated to changes in the values of one or more environment properties. Callbacks are stored in a data structure indexed by environmental property and remain active until they expire. Whenever a change occurs in an environment property, e.g. "PowerSupply", all callbacks that refer to that property are reviewed to check whether all necessary conditions have been met and the application, which submitted them should be notified.

Probe Scheduler The EAS periodically probes the physical devices that provide environment property values. Many environment properties are machine characteristics that don't change frequently, if at all. Therefore, they can be

stored in the EAS and only be recalculated when the local hardware is reconfigured. There is a schedule of the next moment when each of the available devices must be probed in order to update the properties it detects. Probe actions may be disabled during periods for which there are no registered callbacks.

Event Repository The event repository stores all changes in environment properties that are detected by the EAS. Observed events are the result of actual probes of EPMs (see Sec. 2.1) whereas forecast events are calculated by the forecasting model. Scheduled events are explicitly inscribed in personal organizer information.

Forecasting Model A forecasting model was included in order to be able to forecast future situations and to allow applications to perform proactive actions regarding future environment conditions. Many future conditions can be deducted from personal organizer schedules but frequently that leaves a significant amount of time with unknown occupation. This can be complemented with future events forecast based on past recurring conditions. This is performed by an ARIMA[3] forecasting model for discrete variables, which periodically analyzes the event log and tries to detect temporal patterns for each of the detected environment variables and stored labeled situations and inserts new events, adequately tagged as "forecast" into the event repository.

Remote Invocation Interface An EAS provides a remote service that enables queries between different devices. This opens up the possibility of sharing information among a group of devices so that in some cases other trusted devices can work as extensions of a device. In general, the remote interface is used for queries that refer to neighbouring devices and are most useful to allow applications to take advantage of available resources around it.

2.1 Environment Perception Modules

Environment Perception Modules (EPMs) are the EAS components that interact with physical devices and assess current environment properties. Building them is the greatest challenge in implementing an EAS due to hardware and OS heterogeneity. Each EPM has to include code to assess its properties by probing different devices in different OSs. Currently many of the OS modules and devices that provide environment properties are accessible through standard APIs, which may simplifies the EPM code greatly.

Personal Information Manager Many computer users run personal organizer software, which can be a source of valuable information regarding the device's future environment. This information isn't presently taken into account by computing systems.

For example, Microsoft's personal organizer, Outlook, provides automation objects APIs, which can be used to programmatically acquire information and store it into a future location table in the EAS. This information could also be acquired from other similar software with a specific EPM.

Once integrated in the EAS, this information can be used to feed the forecasting model. A schedule can be further processed to provide applications with hints and to determine, of those events stored in the event repository, the ones whose corresponding callbacks should be invoked invoked. Thus applications can, asynchronously, send results to hardware not present, assured that data will be sent to it when it connects to the network and becomes available.

Whenever a location is provided by a personal organizer appointment, the situation hierarchy is scanned to check whether that location has been submitted as a situation label. If so, for each of the attributes contained in that situation, scheduled events are inserted into the event repository. Naturally, events scheduled in the personal organizer override forecast events.

Location is a particularly important characteristic of the environment. Therefore, a index of previous locations, and of the corresponding values of the environment properties, is kept in order to derive more information about future locations which are known from scheduled events (see 3.7). Moreover, events, which have a high probability of occurring at a scheduled events location, are added to the event repository as probable events.

Local Computing Environment This EPM allows applications to obtain information about the software environment in the device, such as, which applications are running, which is the current foreground application and which files and folders are being accessed. Knowing which files and applications are accessed by a user at each device is fundamental in order to configure devices which will be used in the future. A good example of this form of adaptation is the hoarding application described in Sec. 3.

Other Naturally, the EPMs that were presented above are a fraction of those that will exist and be of interest to applications in the future. Important environment properties we did not discuss are, for instance:

- Absolute location: This EPM would provide GPS coordinates to applications or confirm previously provided labelled locations;
- Relative location and tagging: Currently, there are many ways to provide relative location within a restricted space (WiFi, Bluetooth, RFID). The EAS can uniformly inform applications of their location or of the presence of other device or persons.
- Services: EAS can also be used to integrate and standardize existing service location infra-structures;
- Processing power: CPUs can be benchmarked and classified for applications using a simple description domain;

– Power supply: Time and percentage estimates of battery time are available on most computers today. They can be easily represented as an environment property within the EAS and be used to adjust energy consumption and/or CPU speed.

We can envisage many situations were other environment information and services become relevant: assessing lighting to manage solar charged batteries, using voice processing services at nearby devices, etc. As the EAS' implementation evolves it will incorporate such novelties as new attributes in its event repository.

3 Context-Aware File Hoarding

We chose to use the EAS to address the problem of hoarding, i.e. automatically select the relevant files to be replicated when a user moves from one computer to another (as described in Sec. 1). Automatic file management is an relevant requirement in mobile environments because more and more users have several devices (PC, laptop, mobile phone, PDA) and the storage and bandwidth between them is not constant and unlimited.

The ability to keep a user's files updated on her current device requires first of all that the relevant files be present at that device. It isn't feasible to transfer all of a person's files because many devices have limited storage and bandwidth is often a bottleneck for large transfers. It is impossible to rely on transferring files on demand because network connections aren't constantly available. And finally, selecting files to be replicated manually is time consuming and error prone.

SEER[4] has shown that hoarding files based on the history of most recently used files is the best known heuristic. However, it should be pointed out that this results from a small scale exploration of the parameter space of their algorithms and from assuming that people use only one type of computing device. There are many situations where more sophisticated information is needed. For example, many users do different tasks depending on where they are (at home, commuting, in the office, at a meeting, etc.) and which device they are using (PC, laptop, mobile phone, etc.). Letting users' habits and access patterns determine which files will be hoarded hasn't been tried and can be achieved using the EAS. This would, for example, enable a user to leave her office to do a presentation elsewhere, without worrying about her slides, because the EAS on her PC would have detected that it was necessary to transfer them to the PDA she carries with her.

The first step of hoarding is clustering files so that semantically related files are moved together. Detection of file accesses is performed by the Local Computing Environment (LCE) EPM. Detecting past patterns automatically by correlating file access and environment properties is the job of EAS's forecasting model. Forecasting future actions is essential for a hoarding algorithm that is more sophisticated than just hoarding the most recently used files.

We have implemented a file hoarding application on top of a EAS prototype(Fig. 1). This application monitors user activity and clusters and selects

data that should be hoarded in case a user decides to move to another device. Users can assign folders, extensions filenames and applications to any given task. Using that information, the hoarding applications clusters accessed files according to the user's preferences. Only when it is unclear which task a file belongs to, the application then asks whether the file is to be assigned to the current or some other task. So far our experience has shown that the disturbance caused by asking the user to organize her files, quickly fades away. Currently, we log all accesses to selected folders enabling the detection of time patterns in the history of file (and consequently task) accesses. Additionally, personal calendar data allows the EAS to compare the subject of scheduled activities with task and file keywords, thereby improving our hoarding estimates.

4 Related Work

We have been witnessing a trend towards integrating several kinds of devices in an increasingly network-centric manner. Research work developed to deal with the issues raised by this shift have been divided in a number of fields. The most relevant efforts have been made in the areas of location awareness, device/object identification and service location.

Regarding location awareness, in [5] routing efficiency and quality of service is maximized in ad-hoc wireless networks, i.e., networks composed of dynamically repositioning mobile hosts. In this work, location awareness is used at a lower level than in our work. It is used to improve routing algorithms and packet forwarding. Our approach aims at providing applications broader information about their computing environment, its host and its neighbourhood i.e., information about every capability their execution environment supports, and noticing them when these capabilities are subject to temporary or permanent changes.

In the Xerox ParcTab[6] experience, broad work about location and context-awareness, for proximate selection and automatic reconfiguration, ranging from hardware design, user interface customization based on context, and location information is also presented. Context information is based on information about neighbouring hosts. A specially developed predicate language was included for programming context-triggered actions. The system is highly dependent on outside information like responses to homing beacons and positioning devices.

Naturally, these notifications about the execution environment, in the case of network centric applications running in mobile hosts, must include information about nearby devices that may be consulted to obtain such information. This must be implemented with some kind of distributed event processing. There are several approaches to this issue[7–10]. Bates[7] defends a framework for a federation of heterogeneous components connected, transparently, by distributed events in a publisher-subscriber model. There is an event taxonomy based on event sub-classing. There is an event composition algebra that allows some degree of control over dependency checks between different sets of events and to enforce certain event sequences. Events are logged for future replay or querying.

In Jini[8] the emphasis is put on resource and service discovery. There is only one event class so it lacks expressiveness although each event may contain an arbitrary data object.

This architecture was further refined and extended in Rio[9] with extended event description, capabilities detection for proxy execution, operational strings to represent resources and services and quality of service matching between device capabilities and application requirements.

The Universal Plug and Play[10] approach aims at very similar goals than the previous two but is also centered on remote device and service detection, and data exchanging without any sort of code download. It is supported in a series of standards that makes it more platform independent though less flexible in the dynamic code download aspect. It supports dynamic IP addressing, device and service discovery; control actions are based on SOAP URL invocations and received events are implemented in XML messages defined in GENA.

All these technologies try to take advantage of some form of awareness about the computing environment in some specific way. None of them, though, comprehensively attends to all the properties mentioned in this paper or aims to standardize the representation of environment properties in an extensible manner.

There have also been attempts to create generic context-awareness platforms[11, 2, 1]. Our decision to design the EAS, came from the realization that some of these platform either were aimed at specific context properties (relative location and user/device identification as in [11, 2]), while others require programmers write specific code for each new environment property [1] and that none of them considered knowledge of future conditions as relevant input for device adaptation. For example, Gaia[2] results in modified applications that interact with an omni-present infra-structure whereas we would simply like applications to become aware of encircling resources.

5 Conclusions and Future Work

This paper presents an architecture for an integrated environment awareness system (EAS) that allows applications to assess and adapt to the computational, network and physical environments. The system can anticipate future user behaviour based on past patterns in order to take full advantage of the resources available in the future.

We also demonstrate how an EAS can aid the task of automatic file management, in particular file hoarding. Making well informed hoarding decisions requires complex information about users' habits and patterns and these can be acquired and structured by an EAS. We present a prototype of a hoarding application based on a EAS which collects information from the local computing environment in order to perform file clustering and estimates future user file needs by comparing the keywords of accessed user tasks and the subject of calendar scheduled activities.

As future work, we are currently obtaining performance and user experience results on the EAS main features (queries, callbacks) and on the hoarding prototype. We are also refining the design of the EAS architecture modules that weren't implemented for the prototype.

References

1. Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: Aiding the development of context-enabled applications. In: CHI. (1999) 434–441
2. Shankar, C., Al-Muhtadi, J., Campbell, R., Mickunas, M.: A middleware for enabling personal ubiquitous spaces. In: Workshop on System Support for Ubiquitous Computing (UbiSys '04) at the Sixth Annual Conference on Ubiquitous Computing (UbiComp 2004), Nottingham, UK (2004)
3. Makridakis, S., Wheelwright, S., Hyndman, R.: Forecasting: methods and applications. third edn. John Wiley and Sons, New York (1998)
4. Kuenning, G., Ma, W., Reiher, P., Popek, G.: Simplifying automated hoarding methods. In: Proceedings of the 5th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'02), ACM (2002)
5. Tseng, Y.C., Wu, S.L., Liao, W.H., Chao, C.M.: Location awareness in ad hoc wireless mobile networks. Computer **34**(6) (2001) 46–52
6. Want, R., Schilit, B.N., Adams, N.I., Gold, R., Petersen, K., Goldberg, D., Ellis, J.R., Weiser, M.: The parctab ubiquitous computing experiment. Technical report, Xerox Corporation Palo Alto Research Center (1995)
7. Bates, J., Bacon, J., Moody, K., Spiteri, M.: Using events for the scalable federation of heterogeneous components. In: EW 8: Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications, New York, NY, USA, ACM Press (1998) 58–65
8. Waldo, J.: The Jini Specifications. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2000)
9. Sun Microsystems: Rio Architecture Overview. (2001)
10. Microsoft Corporation: Universal Plug and Play Device Architecture. (2001)
11. Ferscha, A., Vogl, S., Beer, W.: Context sensing, aggregation, representation and exploitation in wireless networks. Scalable Computing: Practice and Experience **6**(2) (2005) 71–81