From Computers Everywhere to Tasks Anywhere: The Aura Approach

João Pedro Sousa and David Garlan

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{jpsousa|garlan@cs.cmu.edu}
http://www.cs.cmu.edu/~aura/

Abstract

A critical problem for ubiquitous systems engineering is to exploit resource-rich environments without burdening the user with their configuration and management. A particularly important aspect of this problem is to support continuity in the face of mobility and dynamically varying resources. In this paper we argue that a solution to this problem requires new system-level components and architectures. Specifically, we describe our design of a personal "Aura" component that acts on behalf of a user to manage resources, provide continuity, and support high-level user tasks.

Key words: ubiquitous computing, software architecture, task modeling, task-oriented computing, fidelity-aware computing, context-aware computing.

Submitted for publication *20 April 2001*

1 Introduction

Ubiquitous computing presents both an opportunity for users and a challenge for system engineers. For users, a wealth of computing, informational, and communication resources available everywhere should allow them to work more effectively. For system engineers, however, there is the challenge of harnessing these resources without overburdening users with management of the underlying technology and infrastructure.

One particularly important aspect of this problem is to support continuity in the face of time-varying resources. While ubiquitous computing promises to make many more resources available in any given location, the set of resources that can be used effectively is subject to frequent change – both because the resource pool itself can change dynamically, and because a user may move to a new environment, making some resources available and others inaccessible.

As we detail later, traditional solutions normally associated with mobile computing are inadequate to solve this problem either because they are unable to exploit resources as they become available in a user's environment, or because users must pay too high a price to manage those resources. In Project Aura we are developing a new solution to this problem based on the introduction of a concept of a personal *Aura*. This solution builds on existing efforts to provide service location and discovery, and also incorporates research on fidelity-aware computing, context-aware computing, networks, and human-computer interaction.

The intuition behind a personal Aura is that it acts as a proxy for the person it represents: when a person enters a new location, her Aura negotiates the marshalling of the appropriate resources in support of the person's tasks.[1] The Aura shields the person from the variability of computing environments as well as from the instability of resources. Moreover, it can play a proactive role in anticipating the future needs of the person. Figure 1 illustrates this idea, using a thick arrow of distracting interactions between a user with no personal Aura and the pervasive infrastructure, on the left, and on the right, a thin arrow of meaningful, distraction-free interactions between a user and his personal Aura: all the hard work in setting up and managing the pervasive computing infrastructure is being done by the Aura itself. To accomplish this, an Aura needs to be knowledgeable about a user's personal preferences, policies, and on-going tasks.

While an appealing idea, from a systems engineering point of view there are a number of hard questions: Which should be the responsibilities of the system components embodying the concept of Aura? What kinds of information should they capture, and what forms should that representation take? How do those components interact with other parts of the computing infrastructure?

In this paper we present a set of answers to these questions. Section 2 outlines related work. Section 3 then identifies the challenges in ubiquitous computing that are addressed by our research, proposes criteria to evaluate solutions to those challenges, and gives concrete examples of the features and properties that our system can support. Section 4 describes the architecture that we have adopted to support the concept of a personal Aura, enumerates the responsibilities of each of the components in that architecture, and illustrates how those components interact in terms of a concrete scenario. An especially important role in this architecture is played by task descriptions: we enumerate the requirements that are imposed by such a role, identify the features of task description languages that address those requirements, and argue why existing workflow languages do not support all of those requirements. Section 5 presents an account of the state of the implementation of the Aura architecture, and Section 6 summarizes the principal contributions.

---

[1] When we refer to *tasks* in this paper, we mean tasks that are carried out by people using the ubiquitous computing infrastructure, rather than to tasks in the operating system sense of a schedulable process.

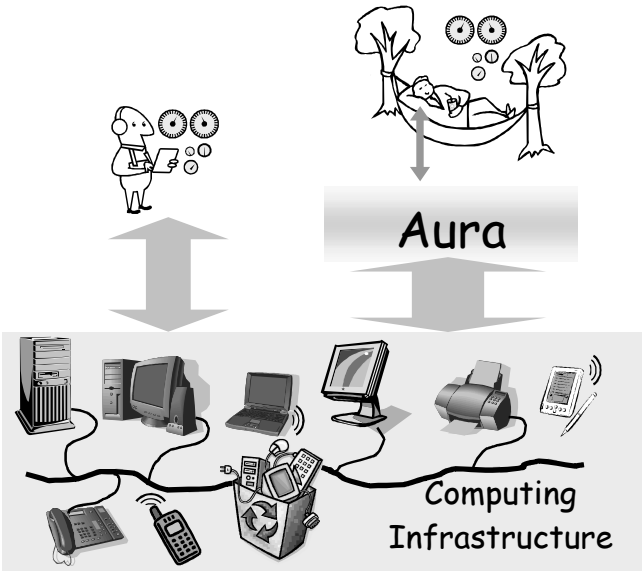

Figure 1. Aura as a proxy for people

2 Related work

The area of ubiquitous computing is populated with numerous research projects and approaches: see for instance [10], [11], [12]. Many of these, however, are primarily concerned with two key problems. One is the problem of dealing with "sensors everywhere" issues: given a highly instrumented environment containing various kinds of information gathering devices, how do we use this information effectively. Key sub-problems are information filtering, fusion, and abstraction. A second key problem is the construction of new devices and services that can function effectively in a ubiquitous computing setting. Research attacking this problem includes wearables, new sensors and actuators, adaptive portable computers, people location services, location-independent information access, adaptable communications and network infrastructure, etc.

In contrast, our research is primarily concerned with reducing user distraction by exploiting the resources[1] made available in a ubiquitous computing setting to shield the user from the underlying complexity and variability. Thus our research builds on much of the other ubiquitous computing infrastructure research, but with a new primary locus of concern – namely the scarce resource of user attention.

An important component of our research is the representation and management of user tasks. This work is closely related to existing work in academia and industry that addresses the modeling of tasks/workflows. In the area of distributed systems and robotics, various task description languages (TDLs) have been developed. In that work tasks are meant to be executed by computers: TDLs are akin to programming languages endowed with special communication and synchronization primitives [13], [14]. A different approach is taken by the business and requirements modeling community: workflows are used to dictate the behavior of the executing agents [15], [16], [17]. The human-computer interactions community uses task descriptions to model the admissible behavior of users, and derives interfaces from those descriptions [18]. In contrast, our research on task description and execution is driven by two principles: first, that the agent executing the task should not be strictly constrained by the task description; and second, that the primary pur-

[1] By *resources* we broadly include devices, networks, services, and applications.

pose of the task description is to enable proactivity by a third party in setting up the support for the task. These two assumptions determine the novel nature of the task execution and error handling strategies adopted in Project Aura, as we detail later.

3 The Problem

In a world where computing and communication infrastructures are rapidly becoming pervasive, assumptions based on workstation-oriented computing are increasingly inappropriate. First, people's computing needs are no longer confined to a desk in an office: computing can now also take place at home, on the train, at the park and even walking down the street. Second, people are increasingly surrounded by devices that enable computation and communication. It is not uncommon for a person to have a workstation at the office, a personal computer at home, maybe a laptop, a handheld computer and a mobile phone; all of them potentially interconnected through wired or wireless networks. In the future the number of potential sources of computation and information will only skyrocket, as will people's expectations of using these resources wherever they may be at a given time.

While such diverse, resource-rich, and pervasive computing environments promise to provide substantial new capabilities for users, there is one significant problem: people are spending more and more time managing the heterogeneity of technology and replication of information, instead of focusing on the productive *tasks* themselves.

Thus, arguably, the central challenge for ubiquitous computing is to find solutions that satisfy two competing goals. The first is *maximizing use of available resources* – that is, effectively exploiting the increasingly pervasive computing and communication infrastructure. The second is *minimizing the distraction and drains on user attention* that stem from managing those resources in a setting where users are mobile, and where the resources in a particular environment may change dynamically and frequently. These two goals are in opposition insofar as the more technology one exploits, the more that is required to make it all work together smoothly and adapt to changing circumstances.

One way that people are addressing this problem today is by supporting as much of their computing needs as possible on a machine that they can carry everywhere. This approach partially solves the problem of user overhead in adapting computations to new environments, since it provides a uniform interface to computation and information. Unfortunately, however, it suffers from the problem that it is not able to exploit the richness of local environments – such as external displays, processors, and input devices. Moreover, because of size, weight, and battery constraints, the computing capabilities of mobile devices will usually be weaker than the capabilities of their fixed counterparts with no such constraints.

A second alternative – one that deals with the limited power of mobile computing platforms – is to compute via remote access to a computing server that stores a users personal state and preferences, much as X-terminals do today. Like portable computing platforms, this approach presents a uniform view of the personal work setting, thus reducing user distraction. But also, like mobile devices, it suffers from the inability to take advantage of local capabilities to maximize the utility of the environment relative to the person's needs. Also, it typically requires a stable, long-lived, high-bandwidth link to the remote server.

We propose an alternative approach in which the computing infrastructure is configured automatically for the mobile user, potentially using whichever computing capabilities are available or reachable from the current location. The challenge lurking behind this approach is to understand how to enable the system to reestablish the personalized work setting of a mobile user, recovering the activities' execution state where it left off, and reconstituting it in the new setting. Further-

more, that reconstitution should take full advantage of the capabilities available at the local environment, augmenting any devices that the person is carrying to provide as high a quality of service as possible.

Currently available tools for service location and reservation, like Jini™ [19], take a first step in this direction, by providing a way to locate available services in an environment. But more is needed to fully address the challenge above, since service location capabilities don't by themselves deal with the problem of deciding which services should be used, and how they should best be configured to support a user's higher-level goals. Nor do they help maintain continuity for mobile users. Nor do they handle the problem of time-varying resources, for example as devices move in and out of range, or communications bandwidth changes due to external load.

The essential ingredient missing is a representation of users needs and desires. To see why this is necessary, consider the problem of adapting the computation to accommodate changing bandwidth. For instance, suppose that a person is viewing a video over a network connection for which the bandwidth suddenly drops. Should the system (a) advise that the activity can no longer be accomplished, (b) pause the video and try to find an alternative connection, or (c) reduce the fidelity of the video, and if so, should it reduce the frame update rate or the image resolution? The most appropriate choice depends on the person's intent for watching the video.

As noted in Section 1, our approach to addressing these issues is based on the introduction of an infrastructure layer between the person and the computing environment. The purpose of such a layer is to act on behalf of people: setting up the environment according to a person's needs, dealing with heterogeneity of platforms, variation of resources, and migration of work settings and the associated information. Playing such role requires knowledge about the person's preferences, the tasks she intends to carry out, and how that intent is best served by tradeoffs among limited resources. That role requires the ability to forage capabilities on the different environments a person roams into. People have traditionally played this role themselves – system administrators and users covering different aspects of it – now we want a part of the pervasive infrastructure to play it for us. We call that layer *Aura*, and, conceptually, each person has one instance of it.

3.1 Capabilities of the infrastructure

To make the problem more concrete, in our research we have adopted a set of benchmark scenarios that describe the kind of features we expect from a pervasive computing infrastructure such as the one we propose. While there are many possible scenarios, in this paper we consider five that are specifically related to task instantiation and adaptation in a mobile setting, and that briefly touch on the influence that the physical context has on tasks. We describe their main features below, along with a short synopsis for ease of reference. (A fuller description of benchmark scenarios is included in the Appendix.)

The first scenario illustrates a person leaving one environment and resuming the same task in another, albeit similar, environment.

Scenario 1 *- Commodity computing*

Fred is working on a task at home and leaves for the office. Aura sets up the environment at the office and resumes the same task as soon as Fred is recognized entering the office.

The following scenario illustrates a person moving through environments while having an urgent pending task, and finding the capabilities to accomplish that task in an unanticipated location.

Scenario 2 *- Marshalling services*

On his way home, Fred stops at a takeout restaurant. Aura proactively marshals services at the location, allowing Fred to complete an urgent task.

The third scenario illustrates a person resuming a task in adverse circumstances. In the first situation, Aura advises not to initiate a task due to lack of resources out of Aura's control. In the second situation, Aura distinguishes between what is essential to the task and what is not, assisting the person in making the most out of the available resources. In the third situation, Aura proactively scans the vicinity for resources and suggests the person to go to another location where the task can be carried out. The forth situation illustrates the exploration of possible degraded modes of operation, and in the fifth situation, Aura proactively scans forecasts of resource utilization to find a time when the task can be carried out with satisfactory quality.

Scenario 3 - *Helpful when it goes bad*

Fred is waiting at the airport and trying to resume a videoconference. Aura helps by identifying critical resources, where and when to get them, and what can be done with what is available.

The fourth scenario illustrates Aura's proactivity in setting up the next possible task steps, covering several – the most likely – possibilities.

Scenario 4 - *Proactive set up*

Fred may or may not stop at the local library on his way to the office. Aura prepares the environments at the different locations for the few of Fred's most likely tasks, and monitors his whereabouts to know exactly which work setting should be brought up and where.

The fifth scenario illustrates how the physical context places restrictions on the activities carried out by a person, and how Aura reacts to those restrictions, helping but not hindering the person.

Scenario 5 - *Context awareness*

Fred is working on a task that contains sensitive information. When someone enters the office, Aura checks the level of trust for that person and automatically hides the sensitive information.

Note that in all of these scenarios the interactions between the person and the computing infrastructure are in terms of the tasks the person wants to accomplish and their supporting information, and no longer in terms of individual applications or files.

4 The Aura design

We identify three high-level components to which we assign the responsibilities of addressing the challenges in Section 3: first, a *Task Manager* component embodies the concept of personal Aura. Second, an *Environment Manager* component takes charge of finding, setting up, and managing computing capabilities and resources. And third, a *Context Observer* component that reports on the physical context where people carry out their tasks, allowing both the Task Manager and Environment Manager to react to changes in the context. We elaborate the responsibilities of these components below, and Section 4.2 illustrates how they interact during a concrete scenario.

4.1 Components of Aura

Task Manager. When a roaming person enters a new environment, it is the responsibility of the Task Manager to consult the task information for that person and to proactively request the environment to set up the capabilities that support the person's tasks. The Task Manager assumes that the Environment Manager holds explicit information about what the local environment can offer – its available capabilities and resources – as determined by local devices, communication infrastructures, and applications. Furthermore, upon dynamic changes in the available capabilities or resources, the Task Manager negotiates the adjustments that best serve the person's intention with the Environment Manager.

To play such a role, there are four aspects to the knowledge a Task Manager has about the person it represents. First, what are the preferences of the person: what are the preferred providers that support a given capability (for instance a preference of emacs over vi for text editing,) and what are the personalized settings for those providers (personalized shortcuts, macros, etc.) Second, how the intent for the current activity is reflected in the preferred tradeoffs upon limited resources. For instance, upon limited bandwidth does the person prefer to watch that particular video later, or is it preferable to degrade the fidelity; and if so, which dimension of fidelity. Third, what is the encompassing task within which the current activity fit; what are the following steps, when are they expected, and what do they involve in terms of computing capabilities; and what is the person's likely trajectory for the immediate future. This knowledge is key for proactively setting up the computing infrastructure. Fourth, how the physical context surrounding the person affects each of the other three aspects; for instance in Scenario 5, how a person entering Fred's office prompts the hiding of sensitive information.

Environment Manager. The responsibilities of an Environment Manager are: First, matching capability requests issued by Task Managers to the offer of local or remote providers of services, cooperating with other Environments Managers as deemed necessary. Second, maximizing the overall utility of a requested set of capabilities in the face of limited resources, and according to preferences for the quality of service tradeoffs as expressed by Task Managers. Third, retrieving and restoring the configuration and execution state of services, so to enable the continuity of a task across different providers for the same service. And fourth, monitoring the environment, automatically taking the necessary actions to assure the continuity of service as negotiated with Task Managers. We detail some consequences of these responsibilities in the following paragraphs.

An *environment* at a given location is defined to be the set of capabilities that can be *used* from that location. Those capabilities include the ones resident on devices directly accessible by the person, i.e., on local devices, as well as on devices that can be used via some communications network, e.g. a remote file server. Contrast "usability" in this sense with "reachablility" in a strict communications sense: if the person is sitting in Las Vegas, printing a document at a (reachable) printer in Chicago may not be very handy – depending on the person's intention. That is, the set of *usable* capabilities is defined not only by the location of the person (i.e., by the devices around her), but also by the intention of the task that the person wants to carry out. For example, printing a document in a remote printer may be exactly what the person wants, if the document is to be reviewed by someone else.

However, it is not practical to expect an Environment Manager at a particular location to know about the capabilities on every location around the world. Therefore, although an Environment Manager may know about all the local capabilities, it always has to assume incomplete knowledge of the actual environment pertaining the task of a roaming person. For instance, it is the responsibility of the person's Task Manager to let the local Environment Manager in Las Vegas know that the person wants to use a printing capability near room X on building Y… in Chicago. This observation points to the need for a cooperation protocol between Environment Mangers, for finding capabilities and allocating services across different locations. Another example of using such a cooperation protocol is if the task requires some computing intensive activity for which the local capabilities are insufficient. In a case like this, the Task Manager may rely on the local Environment Manager to discover remote CPUs that comply with the requirements, or the task description itself may include a list of specific computers that are preferred by the person for those purposes.

The capabilities of an environment are qualified by properties observable by the person. For instance, a speech recognition capability may be qualified by the extent of the vocabulary it recognizes and by the average latency of recognition. The possible values of those properties are, in

turn, constrained by the availability of relevant resources; say CPU cycles, bandwidth to remote CPUs, battery charge, etc. So, tradeoffs arise in the face of limited resources: the person should expect higher latencies if she prefers a larger vocabulary to be recognized. Assigning the responsibility of maximizing the utility of a capability to the Environment Manager implies that, first, it should consider alternative providers for the requested capability, and second, it should tune the fidelity of that capability according to the preferred quality of service tradeoffs in face of the available resources.

Note that the goal of migrating tasks from one environment to another also poses requirements on the applications and devices providing the services that support those tasks: the new provider for a service must be willing to resume the execution state generated by some other provider serving that task in the previous environment.[1]

Context Observer. This component has responsibility for observing the physical context surrounding people. For instance, what is the physical location of the person (her office, the street…) who else is in the vicinity and what is the corresponding level of trust; and what is going on around the person (working by herself, in a formal meeting, driving a car…).

Note that recognizing the *constraints* that a particular context poses on the tasks is a responsibility for the Task Manager: for instance, in Scenario 5 in the Appendix, knowing which aspects of the task should be suspended when the Context Observer observes someone entering Fred's office. Likewise, it is the responsibility of the Environment Manager to know what to do with the devices that a Context Observer recognizes within reach of the person: for instance, in Scenario 2, matching the capabilities observed in the takeout Fred just walked in to the capabilities requested by Fred's Task Manager.

4.2 Aura in action

We now use Scenario 1 in the Appendix to illustrate the workings of Aura, emphasizing the protocols of interaction between the components identified in Section 4.1. For simplicity, we refer to an instance of a particular component at a given location as "the *component* at *location*;" for instance, "the Environment Manager at Fred's home," or simply, the "Home Environment."

In Figure 2, Fred is working at home when the Context Observer notices Fred leaving the house. The Context Observer lets Fred's Task Manager (TM) know that Fred is leaving through interaction (1). This causes Fred's TM to undergo state transition (a) where it realizes it should suspend the task ongoing at home. Fred's TM then requests to checkpoint the state of each of the services being provided as part of the ongoing task – interaction (2) – and in interaction (3), Fred's TM tells the Home Environment to deallocate those services.

After checking Fred's schedule, the TM infers that he is likely to head to the office, and (4) conveys that information along with an estimated time of arrival to the TM at the office.[2] That triggers state transition (b) in the TM at the office, causing it to (5) retrieve the state from the tasks Fred has been working on in the last location.[3]

---

[1] For instance, the execution state associated with text editing includes the edited file, the pane and cursor position within that file, and the editing preferences and macros.

[2] We are deliberately abstracting away implementation decisions and mechanisms concerned with the instantiation of Task Managers at different locations. Examples of such decisions are: whether to migrate or replicate a TM, whether to cache it in an environment where the person goes recurrently, etc.

[3] The state of a task includes the task description, personal preferences, the configuration and state of all the computing services supporting the task, as well as the manipulated information. Again we are abstracting

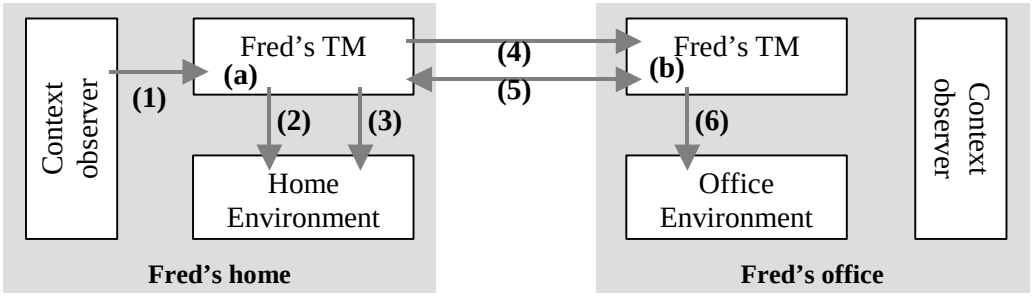

Figure 2. Fred leaves for the office

Given that, Fred's TM at the office requests the capabilities involved in Fred's task to the Office Environment, (6), indicating the estimated time they will be needed. By having this advanced notice on the requested capabilities, the Environment can do a better job at planning and managing the resources it has available. Now, there are two possible outcomes captured in the following two figures. In Figure 3, the Environment is plentiful in resources and is willing to setup in advance the services corresponding to the requested capabilities. So, it comes back to Fred's TM, (7), indicating the services allocated to cater for Fred's computing needs. Fred's TM is therefore able (8) to restore the personalized configurations and execution state of those services as left off by Fred at home. As soon as the Context Observer at the office recognizes Fred coming into his office, it informs the TM of that – (9). This causes Fred's TM to undergo state transition (c), resuming the task for Fred, and requesting the services to popup in the Office Environment (10).

In Figure 4, maybe due to local policies concerning the anticipated granting of services in face of constrained resources, the Office Environment does not get back to Fred's TM with a list of allocated services before Fred actually reaches the office. When the TM is informed by the Context Observer that Fred is coming into his office, (7), it makes a state transition (d) that prompts it to resume the task for Fred: requesting the immediate allocation of services (8); and upon such allocation, (9), restoring the personalized configurations and execution state of the allocated services – (10).

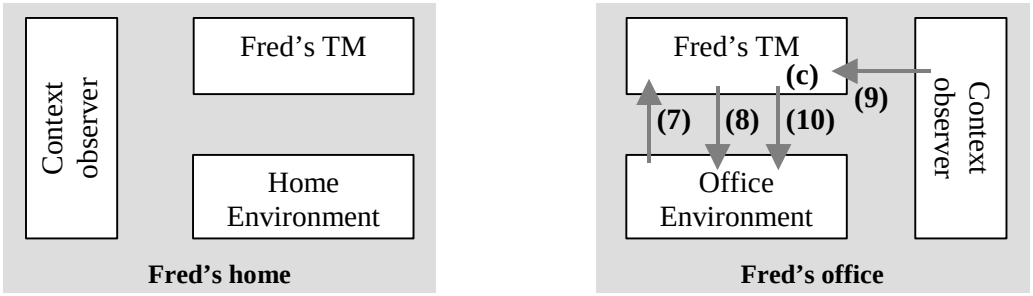

Figure 3. The environment is willing to setup the services before Fred reaches the office

away implementation issues like whether that state is totally or partially (updates only) transmitted in a point-to-point fashion, or by synchronization with a distributed file system.

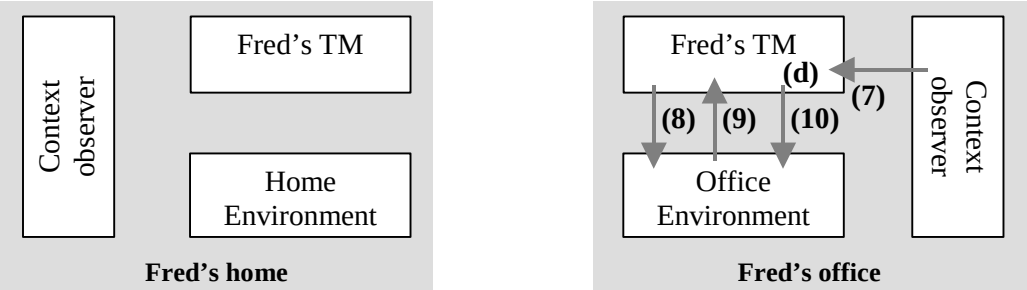

Figure 4. Fred reaches the office before the services were setup

4.3 Describing Tasks

Examples of tasks are found in the scenarios described in the Appendix: organizing a conference; debating some topic with a number of people; interviewing and evaluating candidates to some position; etc. Tasks are usually long lived – they span a number of *sessions* – and they can be carried out anywhere – each session can potentially be carried out at a different location.

Task descriptions in Aura obey to the following four requirements. First, we would like to be able to instantiate tasks in diverse environments. Second, we would like to reduce distractions by raising the level of interaction between the person and the infrastructure from individual applications and files to meaningful task steps. Third, we would like to enable proactivity in setting up the environment according to the person's needs for the immediate future. And fourth, we would like the infrastructure to play a supportive rather than restrictive role with respect to people's tasks. We now examine how each of these requirements is addressed in our approach.

First, it is much easier to support the migration of a person's work setting to a different environment if we have a description of what the person is doing – her task – in terms that are independent of particular platforms or application software. That is, by having a person's activity described in terms of the capabilities that are required to accomplish what the person wants. Examples of such capabilities are: video conferencing, text editing, printing, etc. As a person moves from one environment to the next, the available devices or applications that have a particular capability may be very different. For instance, text editing may be supported on a Unix workstation, on a PC running Windows, on a Palm Pilot, or on a set-top-box sitting on some airport lounge… – and the roaming person would like to take advantage of that support to carry out her activity. By asking the current environment for text editing capabilities, as opposed to a specific product, say, emacs, we are able to instantiate the task in heterogeneous environments without previous knowledge of which products support that capability.

Second, raising the level of interaction between the person and the infrastructure is achieved by remembering the set of capabilities associated to the particular activity the person intends to engage in, so that the person doesn't have to setup those capabilities individually. When the person moves from one environment to the next, we use that knowledge to retrieve the state of the task in one environment and reinstate it in the other.

Third, if a person's activity is looked upon as a *task step*, the infrastructure can be made aware of conditions that trigger a step transition. Examples of triggering conditions are:

- an explicit indication from the person wanting to move to another activity;
- an event originated by the activity itself, like disconnecting a phone call or closing a window on a window-based interface;
- derived from the context, like a person coming into or leaving a particular location; or

– based on a timing characterization, like a scheduled video-conference with an established start time and estimated duration.

Upon detecting the occurrence, or the imminence, of a triggering condition, the infrastructure proactively sets up all the capabilities required by the next task step, thus relieving the person of that chore.

Fourth, from the premise that people are to be assisted rather than constrained by the pervasive computing infrastructure, we monotonically increase the task description with a new possibility every time the person takes a step that is unexpected from a task description viewpoint, but legitimate from the viewpoint of security and privacy policies. To manage the proliferation of possibilities, we use probabilistic task descriptions, akin to Markov chains, that describe how likely each of the possible next steps is. This way, the infrastructure can adjust how much it goes into setting up less likely scenarios depending on how plentiful resources are at the time.

Existing Task Description Languages (TDLs) and workflow languages obviously address the third requirement above, and can easily be adapted to provide support for the first and second requirements. However, they make assumptions that are not compatible with the fourth requirement. As we noted in Section 2, current TDLs assume that the purpose of the description is to constrain the behavior of the executing agent to predetermined paths. In the context of ubiquitous computing, however, people are unconstrained agents, free to use the computing capabilities according to their best judgment – opportunistic personal judgment may override existing ways of carrying out a given task, allowing for the exploration of new ways. Furthermore, the purpose of a task description is foremost to support the proactivity of a third party – the infrastructure. That same infrastructure uses the task description to anticipate the needs of the person for the immediate future, setting up and configuring the capabilities corresponding to those needs. When faced with a deviation from the possible behaviors encoded in the task description, the infrastructure can react in three possible ways: a) consider it an error and nudge the person back to known paths. b) Replace the broken behavior by the new one; and c) monotonically increase the set of possible behaviors. Although by requirement four, alternative a) more often takes the form of providing feedback aimed at guiding the person in well-established behaviors, there are circumstances where the infrastructure may indeed refuse to accommodate the person's behavior. For instance, when that behavior collides with human law or coded privacy rules. Since having the system refuse an action by the person can be handled by traditional error mechanisms, we shall not belabor this aspect any further.

Alternatives b) and c) are contrasted in the following concrete example. Suppose that a person uses the pervasive computing infrastructure to help her plan and carry out trips. A high-level decomposition of the traveling task includes such task steps as browsing airline fairs, buying the tickets, getting to the airport and so on.[1] Suppose further that, someday, the person decides not to buy an air ticket after browsing the airfares, but rather decides to rent a car. Alternative b) replaces the behavior excerpt in Figure 5a) by the excerpt in Figure 5b). Alternative c) augments the description in Figure 5a) so to include the new behavior as an alternative, denoted by the + symbol in Figure 6.

[1] Each of these steps could be further decomposed into finer detail; for instance, buying a ticket can be made online or over the phone, each of these requiring different capabilities from the environment. Nevertheless, the presented detail is enough to illustrate the point. Also, the figures depicting the task descriptions are informal, and should not be taken as representative of any form of syntax suggested to express these constructs.

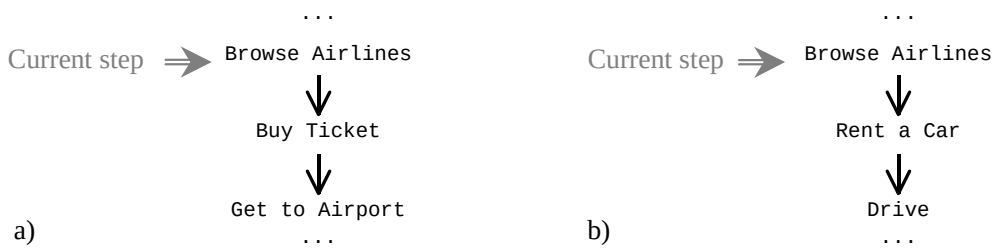

Figure 5. Two versions of a traveling task

One can argue that keeping the two versions of the task description in Figure 5, and toggling between them as appropriate, is equivalent to the augmented description in Figure 6. Indeed that is the case from an adaptation perspective, where the infrastructure reacts to the actions of the person. It is not equivalent, however, from a proactivity perspective: the description in Figure 6 allows the infrastructure to easily foresee all known possible next steps and proactively set them up for the person.

However, continually augmenting the set possible behaviors in the fashion suggested above may overload the infrastructure with respect to setting up a multitude of capabilities, with little added value for the person. This observation constrains the execution model followed by the infrastructure: proactivity is better served by task descriptions that include the relative likelihood of each possible next step. Using probabilistic task descriptions, akin to Markov chains, the infrastructure can adjust how much it goes into setting up less likely scenarios depending on how plentiful resources are at the time. It can also do a better job at guiding the person through most common paths. Figure 7 shows the probabilistic version of the traveling task excerpt, where the infrastructure can observe that the branch related to driving is 20 times less likely than the one for taking the plane. Note that the probabilities associated to alternative branches are not static – they are adjusted from one execution of the task to the next, given the observation of the actual behavior of the person.

Finally, we note that although not a primary goal for the pervasive computing infrastructure, the existence of a task description like the one in Figure 7 enables sophisticated systems to provide people with guidance in best work-practices. Task descriptions may be built to reflect the knowledge of domain experts, which then becomes accessible to less experienced people carrying out the same kind of task. For instance, someone inexperienced in buying a house could benefit from a task description that includes the steps from the market survey, to inspecting the prospective property, to legal procedures. Taking advantage of knowledge on both the criticality of the task step towards the overall intention, and on the context surrounding the person, such sophisticated systems can tune the level of feedback received by the person when deviating from known practices; for instance, no audible beeping during a formal meeting.

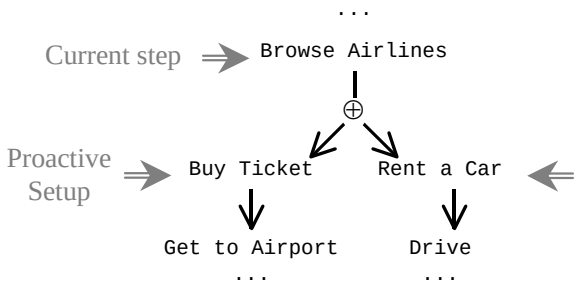

Figure 6. Augmented description for the traveling task

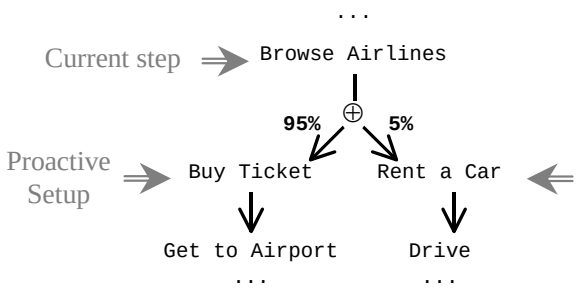

Figure 7. Probabilistic description for the traveling task

5 Aura implementation

The framework described above is being implemented in the context of Project Aura at Carnegie Mellon University. Broadly, the project addresses the design, development and evaluation of a pervasive computing infrastructure spanning wearable, handheld, desktop and infrastructure computers; wired and wireless networks [1]. Within that context, support for task description and user mobility are key components.

As a long-term, ambitious effort, Project Aura, is continuously adapting as new research and system construction proceeds. Our current implementation takes advantage of the wireless network that spans the entire campus, with a potential user base of 10,000. It currently provides capabilities in each of the key categories outlined above. In the area of environment management we are building on top of established service registration efforts, such as Jini, but also providing additional capabilities for monitoring the state of an environment's dynamic resources [8], [9], determining whether *ensembles* of services can work together, and evaluating the utility of alternative configurations [7]. We are also building services, operating systems, and networking infrastructure that is inherently more adaptable and resilient to resource variability [4], [5] [6]. In the area of context observation, we have been developing a number of services that support context-awareness, such as people location services [3]. In the area of task management we have an initial prototype that supports automatic capture of certain kinds of tasks – taken as snapshots of the work setting – and migration tasks across homogeneous platforms [2].

6 Conclusion and Future Work

In this paper we have identified as a core challenge for ubiquitous computing the need to simultaneously make appropriate use of the available resources in an environment, while minimizing distractions to the user, and to do this in a world where those resources are continuously changing because of user mobility or other environmental changes.

To meet this challenge we argued that current approaches will not suffice, and that what is needed instead is a new kind of system architecture that supports the encapsulation of a user's Aura – a set of preferences, tasks, and policies that allow the system to marshal the appropriate resources in support of that user's high-level tasks. To make the ideas concrete we enumerated the requirements of such an architecture in terms of the kinds of usage scenarios that it should enable. We then described a specific architectural framework consisting of an Environment Manager, a Task Manager, and a Context Observer that collectively realize those requirements. Focusing on task management as the key innovation in this framework, we argued that task representations must accommodate workflow-like descriptions, but permit dynamic changes to the flow, and indicate probabilities of paths through the graph. We also briefly described the current implantation of these ideas in Project Aura.

This paper, of course, leaves many details unexplained. How is utility best determined? What kinds of service interfaces are needed to support composability? How does a user specify or modify a task, and what mechanisms can be introduced to simplify this process? What is the right balance between system proactivity and non-intrusiveness? These are all interesting questions, and ones that form the basis of on-going research in this area.

7 References

[1] Aura Project at Carnegie Mellon University, http://www.cs.cmu.edu/~aura/.

[2] Z. Wang, D. Garlan. Task Driven Computing. *Carnegie Mellon University Technical Report CMU-CS-00-154*, http://reports-archive.adm.cs.cmu.edu/cs2000.html, May 2000.

[3] J. Small, A. Smailagic, D. Siewiorek. Determining User Location For Context Aware Computing Through the Use of a Wireless LAN Infrastructure, http://www.cs.cmu.edu/~aura/docdir/small00.pdf, submitted to the ACM Sigmetrics, 2001.

[4] B. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, K. Walker. Agile Application-Aware Adaptation for Mobility. *Proceedings of the 16th ACM Symposium on Operating System Principles*, October 1997, St. Malo, France.

[5] J. Flinn, M. Satyanarayanan. Energy-aware adaptation for mobile applications. *Proceedings of the 17th ACM Symposium on Operating Systems Principles*, December 1999, Kiawah Island Resort, South Carolina.

[6] J. Flinn, D. Narayanan, M. Satyanarayanan. Self-Tuned Remote Execution for Pervasive Computing. To appear in the *Proceedings of The 8th Workshop on Hot Topics in Operating Systems*, May 2001, Oberbayern, Germany.

[7] C. Lee. On Quality of Service Management. *Carnegie Mellon University Technical Report CMU-CS-99-165*, http://reports-archive.adm.cs.cmu.edu/cs1999.html, August 1999.

[8] N. Miller, P. Steenkiste. Collecting Network Status Information for Network-Aware Applications. *Proceedings of Infocom'00 – IEEE conference on Computer Communications*, March 2000, Tel Aviv, Israel.

[9] T. DeWitt, T. Gross, B. Lowekamp, N. Miller, P. Steenkiste, J. Subhlok, D.Sutherland. ReMoS: A Resource Monitoring System for Network-Aware Applications. *Carnegie Mellon University Technical Report CMU-CS-97-194*, http://reports-archive.adm.cs.cmu.edu/cs1997.html, December 1997.

[10] Oxygen at the Massachusetts Institute of Technology LCS. http://www.lcs.mit.edu/news/scifi.html.

[11] The Endeavour Expedition at the University of California, Berkeley. http://endeavour.cs.berkeley.edu/.

[12] Portolano at the University of Washington. http://portolano.cs.washington.edu/.

[13] D. Doubleday, M. Barbacci. Durra: A Task Description Language User's Manual. *Software Engineering Institute Technical Report CMU/SEI-92-TR-36*, http://www.sei.cmu.edu/publications/documents/doc.list/1992.htm, December 1992.

[14] R. Simmons, D. Apfelbaum. A Task Description Language for Robot Control. *Proceedings Conference on Intelligent Robotics and Systems*, October 1998.

[15] Little-JIL at the University of Massachusetts. http://laser.cs.umass.edu/tools/littlejil.html.

[16] I. Alexander. A Co-Operative Task Modelling Approach to Business Process Understanding. 12th European Conference on Object-Oriented Programming, Workshop W8: Object-Oriented Business Process Modeling, http://easyweb.easynet.co.uk/~iany/consultancy/papers.htm, July 1998, Brussels, Belgium.

[17] H. Simon. *The Sciences of the Artificial*, 3rd Edition, MIT Press, 1996, pp. 59-72.

[18] F. Paternò, I.Breedvelt-Schouten, N.deKonig. Deriving Presentations from Task Models. *Proceedings EHCI'98 - IFIP Working Conference on Engineering for Human-Computer Interaction, Kluwiert Publisher*, 1998, Crete, Greece.

[19] Jini Connection Technology, *http://www.sun.com/jini.*

Appendix – Benchmark Scenarios

Scenario 1 - *Commodity computing*

This simple scenario illustrates a person leaving one environment and resuming the same task in another, albeit similar, environment.

Fred is at home working on the organization of a conference in a remote place. He's gathering information on possible venues and getting budgets for catering. The web pages of some of the hotels include short videos featuring virtual visits to the premises and Fred already downloaded some of these for reference. Fred is also taking notes on a spreadsheet concerning his appraisal of each venue alongside with the alternative catering budgets.

Fred leaves home and heads to his office. Since Fred means to continue working on the organization of the conference, Aura sets up that task at Fred's office so that he can resume his work as soon as he is recognized entering the office: a web browser over the recently visited pages, the downloaded videos paused on the same places, and a spreadsheet containing all the entered figures. Since there is a big screen on the wall of Fred's office, that is preferred to stage the video and web browsing, releasing monitor space for the spreadsheet.

Scenario 2 - *Marshalling services*

This scenario illustrates a person moving through environments while having an urgent task pending, and finding the capabilities to accomplish that task in a maybe unsuspected location.

Fred leaves his office in the evening and heads home. During the day, Fred reached a point where he must decide between a very nice hotel, with a fat catering budget, or a slightly modest one with an attractive budget. Fred sent a mail to the manager of the nicer place asking him to review the budget, but the reply didn't get in by the time Fred left the office. However, the deadline for having a decision made, according to Fred's schedule, is making this a high priority task.

The email with the revised budget actually got in as Fred was walking home. Aura, aware that the wireless handheld that Fred is carrying does not have the capabilities to bring up the task, refrains from bothering Fred.

On his way home, Fred makes a stop at a local takeout. Upon entering the premises, Aura detects courtesy set-top-boxes available for customers. Aura buzzes Fred's handheld and asks if he would be ok with completing the decision while he waits for the food. Fred sits at a free set-top-box and gets authenticated. Aura brings up the mail with the reply from the manager of the nicer hotel, the videos of the final candidate venues and the spreadsheet with the evaluation notes, enabling Fred to work on his decision.

Scenario 3 - *Helpful when it goes bad*

This scenario illustrates a person resuming a task in adverse circumstances. In situation a) Aura advises not to initiate a task due to lack of resources out of Aura's control. Situation b) shows Aura distinguishing between what is essential to a task and what is not, assisting the person in making the most out of the available resources. Situation c) illustrates Aura proactively scanning the vicinity for resources and suggesting the person to go to another location where the task can be carried out. Situation d) illustrates the exploration of degraded modes in order to be able to carry out the task with the available resources. Finally, situation e) illustrates Aura proactively scanning forecasts of resource utilization so to find a time when the task can be carried out with satisfactory quality.

Fred is at his office, holding a videoconference with four colleagues. The incoming video feeds are being displayed on the big screen on the wall, and Fred is using the monitor on his desk to consult some notes and charts that are helpful to support his argument. But Fred has a plane to catch, so he leaves the discussion.

Twenty minutes later Fred is waiting at the airport, since his plane just got delayed. Fred opens his laptop and lets Aura know that he would like to join the discussion again.

a) Aura estimates the power consumption associated with the communication and rendering of four video feeds and recognizes it can only hold the laptop for 3 minutes on the remaining battery charge. Aura recommends not initiating the communication due to power limitations.

b) Aura is also aware of the limited display space on the laptop, so it informs Fred that even in case he decides to hold the conference, the supporting notes and charts cannot be displayed in an acceptable size.

Fred finds a power plug and accepts not having the auxiliary materials displayed, since by now Fred pretty much has them in his head.

c) In the meantime, a lot of people are getting to the next gate and Aura recognizes that the available bandwidth in the wireless cell is not enough for the minimal needs of the videoconference. Aura consults with the wireless network service at the airport and finds out that the only location with low bandwidth utilization is at the other end of the terminal.

d) Fred is not willing to walk that far, so he changes his preferences for the task, increasing his appreciation for getting service at a closer distance and relaxing his expectations on the quality of service. Aura checks the wireless network service again and finds that the relaxed quality of service parameters can be met six gates down the hall.

e) Fred is feeling tired, so he tries another approach: he relaxes his preference for starting the conference right away. Aura finds out that the local wireless cell is forecasted to clear in 10 minutes, when the boarding closes at the next gate. Aura starts the videoconference as soon as the wireless utilization comes down, and Fred is able to join the discussion until he gets called for boarding.

Scenario 4 - *Proactive set up*

This scenario illustrates Aura's proactivity when faced with uncertainty, covering several – the most likely – possibilities.

Fred is heading for the office and has no scheduled appointments in the morning. Fred has been working on and off on two tasks, and Aura infers from a probabilistic model of Fred's behavior that these are the most likely tasks Fred will be resuming. Aura has no way of obtaining beforehand certainty of which task Fred will decide to work on first, so Aura decides to make arrangements for the two tasks.

One of the tasks requires a literature search, and Fred often does that in the local library; so, Aura contacts the environment at the library to scan for available capabilities and make a tentative reservation for the resources Fred may be using.

Aura observes through the people locating system that Fred is indeed heading to the library, so Aura migrates the information Fred will be basing his search on to the local server, and confirms Fred's reservation indicating his estimated time of arrival. As soon as Fred is recognized sitting at one of the library's desks, Aura brings up Fred's work setting for the literature search.

Scenario 5 - *Context awareness*

This scenario illustrates how the physical context places restrictions on the activities carried out by a person, and how Aura reacts to those restrictions, helping but not hindering the person.

Fred is interviewing candidates for a position in his group, and he has scheduled a meeting with a candidate in a few minutes. Fred is reviewing the details of the job description, contrasting those with the candidate's application and writing his appraisal of the application. Fred classified the access to these three types of documents as follows: the job description is public access, all the candidate's applications can be accessed only by trusted people, and Fred's appraisals are to be accessed only by Fred.

When Aura recognizes the candidate entering Fred's office, his identity is checked against the list of trusted people, and Aura automatically hides all information not for public access. Of course, while the candidate is at the office, Fred can bring back up the candidate's application, if Fred explicitly makes that decision. However, if Fred tries to bring up the candidate's appraisal, classified as being for Fred's eyes only, Aura will prompt a confirmation so to prevent an accidental indiscretion. As soon as the candidate leaves the office, Aura lifts the restrictions on information display and the task can be fully resumed.