Redefining Requirements of Infrastructure for Pervasive Computing Systems

Authors
Institutions
Email addresses

***Abstract*—To enable the vision of invisible computing, an infrastructure for pervasive computing systems is required. This infrastructure is successful if it meets requirements assigned to it. Hence, it is extremely essential to correctly determine its requirements. Much research has endeavoured to figure out these requirements. However, it seems that the research on realising the potential of invisible computing is encountering with difficulties and falls far short of the expectation of the user community. We argue that the reason for this is the lack of an appropriate fundamental model of pervasive computing environments (PCEs). In this paper, we see real world computing objects (including local and global) as universal multi-modal interfaces for users and other computing objects to interact with. These objects are universally networked interactive interfaces while the network capability becomes a trivial concern and computational processing, decision making, and data management are placed on powerful central computers. According to this view, the paper redefines the correct requirements of infrastructure for pervasive computing which must be addressed to realise the potential of invisible computing.**

I. Introduction

Making computing invisible is an inarguable expectation of a knowledge-based and technology-rich modern society. For this vision, computing is able to be a truly intelligent tool supporting user daily life activities. Moreover, beyond a passively supporting tool (i.e. computing only appears when asked), embedding computing into surroundings environments and objects makes them become intelligent and active tools (i.e. computing appears automatically and distraction-freely without asked). This can contribute to the computerisation of human daily life activities and the change of the way people own and use knowledge especially general knowledge (e.g. where is the museum? How to get there?). Computing can help them to achieve their desired tasks without this kind of knowledge.

[2]–[7]

We argue that researchers are encountering with difficulty implementing the objectives of pervasive computing. Despite of the fact that there have been many efforts aiming to achieve these objectives, for two last decades (since the view of invisible computing initially appeared), those endeavours still remain as individually models or domain-specified applications which are hard or even impossible to apply/extend/develop/deploy in large scale applications. As a result, those projects are still mostly remaining in research labs while applying them in the practices is seen as long time fiction stories. It is clear that the implementation and application of pervasive computing in reality have not matched to our current expectation and potentially existing technologies.

Much work has been conducted to find out the reasons of this contradiction. Some have supposed that we are lacking of a theoretically foundation for development/deployment of applications in pervasive computing environments. Some others have concluded that because pervasive computing is a multi-discipline that covers the areas of such as artificial intelligence, data mining, network communication, pattern recognition, and distributed software architecture, ones who would like to success with applications of pervasive computing must own and have an ability of applying those areas of knowledge for their applications. But there are not many of such these researchers.

Other reasons of the failures of bring the vision of pervasive computing to the reality [4]:

- Hiding distribution and relying on technologies such as RPC or distributed file systems;
- Using single-node programming methodologies;
- Hide remote interaction.

We add to these another reason:

- Using single-modal design methodologies.

We, while adopt the second hypothesis presented above, are finding the resolution to the first one, i.e., the underlying fundamental for pervasive computing. As an initial step, this work aims to redefine a general model of pervasive computing environments as presented in Section 2. Our approach derives from the principles of user-centred design in the field of human-computing interaction. Based on our model, in Section 3, we re-identify the requirements of infrastructure for pervasive computing environments. Then, we discuss and evaluate our model in Section 4. Section 5 presents the related work while Section 6 concludes the paper and outline directions for future research.

II. Our proposed model of pervasive computing environments

A. Assumptions of future computing

- Computing devices (not completely computers) are embedded anywhere and powerfully inter-networked among them and between them and central stations. Applications are hosted and managed by powerful computers at centres.
- Storage is huge.

- Processors are extremely powerful.

- an underlying service-based distributed architecture - the emphasis is on task enablement, rather than support for device-specific application such as high-end work processors or games.

B. Advances in technologies

This Section outlines and presents our prediction about advances of technologies regarding to pervasive computing. We are interested in multi-functionalities integrated computing device which is embedded pervasively in daily life environments. This leads to multi-modal interfaces of pervasive applications. Our second consideration is the increasing achievement of addressing the bottleneck problem in centralised computing architectures thank to incessantly rapid development of more powerful processors, greater and more reliable storages, and wider and more powerful networks. These considerations will form a foundation of our model.

Norman [8] asserted that computing devices are more and more specified in functionality and their sets of features becomes limited. This vision is no longer acceptable. As clear proofs for this, it is obvious that today a mobile phone is not merely used for human-human communication as its originally designed function; instead it becomes a multi-function device. Similarly, a printer is not just used for usual printing, it can email, photocopy, scan, make a call, fax, and so forth. Consequently, pervasive computing models based on this vision are not appropriate any more.

The greater advantages of centralise computing model compared to decentralised one are unrefutable. The problems and limitations of this model such as extendibility, scalability, and bottleneck are successfully addressed benefited from rapid improvements of processing power, storage capacity, and network bandwidth and reliability. Much research has tried to develop applications for pervasive computing environments based on decentralised models. However, there are difficulties encountered due to heterogeneity of these environments. To open a path for successful development of pervasive applications, it is essential to re-examine the application of the centralised model for pervasive environments as its inherent potentiality. We do not argue that the decentralised model is completely failed. This model should be suitable in infrastructure-less environments such as disasters, open seas, deep forests, or between cars on roads. Even in such these cases, we still have infrastructures located elsewhere for a centralised model thanks to such as satellite networks and WiMAX.

One of the benefits of the peer-to-peer model is its ad hoc feature which enables elements in an environment to enter or disjoin at any time. However, this feature can be perfectly remained in the centralised model by providing the elements a plus and play mechanism through local/global networks.

Multi-modal interaction is seen as new paradigm of user interface design for pervasive applications with which the user is able to interact with applications (which may involve a range of computing elements such as users, devices, and locations of interests) through the interaction with these computing elements embedded in the surrounding. Note that, each computing element provides a various capability of interaction. We argue that these computing elements should play a simple role being user interfaces for interactions similar to desktops, keyboards, and mouses of the second generation of human-computing interaction rather than the hosts for the entire application or parts of the application. Migrating applications onto devices just makes difficulties for us while it would more straightforward if we host applications on powerful computers in the centralised model. Powerful networks and powerful computers in the future will enable our vision. *Cloud computing* has initial steps toward this view as an evidence for our argument. Cloud computing suggests that instead of buying, installing, configuring, and maintaining applications on your own computers, we rent them and use them through networks such as Internet. In this case, your powerful computers are even purely interface units rather than processing units.

Based on above analysis of the advances in technologies, we introduce our high-level conceptual model of pervasive computing environments in the following section.

C. A high-level conceptual model

Our model of pervasive computing environments is a two-layered and centralised model. The first layer is the interaction layer and the second is the processing layer.

The interaction layer is responsible for computing-computing, computing-human interaction (i.e. interfaces between human and computing or computing and computing) as well as computing-environment interaction (i.e. computing impacts on environments which otherwise provide computing context information which is sent back to the processing layer for processing). Elements of the interaction layer are users, computing devices. The computing devices of interest of a particular application form a logically star network in which the elements of the processing layer play a role of central station of the star.

The processing level consists of powerful computer with supporting hardware and software tools. They are the centre system of database, processing, decision making, adaptation, knowledge interpretation, and communication with elements of other processing levels of other applications. Communication between the two layers relied on network infrastructures which allow on-the-fly formed logically star networks for a specific application.

Elements of the interaction layer are seen as peripherals of a centre system (i.e. the processing layer) which is similar to the traditional computing paradigm (a desktop PC has its keyboard, mouse, and monitor). The diffidence here is that the peripherals are distributed pervasively in the environment.

III. CHALLENGES

- boos sung theem: khong con hop li de chugn nghi rang thiet bi di dong la mini-desktops va ung dung la nhung chuong trinh chay tren cac thiet bi do. - chuyen huong tu quan niem: devices phai co kha nang kham pha mot cach tu dong dich vu, tai nguyen san co [2] sang quan niem: he thong tu dong kham pha dich vu, tai ngueyn, thong tin va thiet bi san co de hoan tat tac vu trong moi truogn da cho de adapt app cho phu hop voi nhung gi dang co. - thay doi quan diem ung dung in it physical vicinity, ung dung any where in terms of virtually vicinity meaning apps of interest located elsewhere in powerful computer but we can see them through networks. - trong nhieu truong hop, chung ta khong can gan them computing vao thiet bi nhu dien thoai ban hay may in de cac thiet bi computing khac (mobile phone, PDA) co the phat hien ra no, mo chi can trung tam biet va dieu khien duoc no, khi do thiet bi se duco ket noi den trung tam va nho trung tam de su dung theit bi may in hay dien thoai do. - thay doi quan diem: developers must phat trien cac ung dung co kha nang constantly adapt to a highly dynamic computing environment [4] thanh quan diem: he thong infrastructure phai cho giup cac apps de adapt to ... - Chung toi argue that, mot ung dung thanh cong trong tuong lai kong chi su dung thong han hep co duoc trong moi truong hien thoi ma can phai truy xuat vao kho thong tin chang han internet. thong tin ma thiet bi so huu co the khong dap ung nhu cau de thuc thi tot mot task. do vay, du muon du khong thi device cung phai connect to wide network chang han internet hoac cac may trung tam manh me de khai thac thong tin khong can thiet bo sung.

- co kha nang giai quyet su tang dan len cua cac thiet bi moi trong moi truong. - dap su hop tac lan nhau khong biet truoc (impromptu) adaptation to situation ma ta khong luong truoc. - quan li moi truong ma khong can su ho tro hoac administrators. - anh uong xa hoi cua thiet bi va cong nghe moi cua pc. - dam bao tinh reliability de ho tro su tin cay/su uy thac cua nguoi dung vao he thong. - suy luan khi doi mat voi su toi nghia ambiguity.

Chung ta phai lam the nao de tranh ra duoc cac vien canh sau:

- nguoi dung bi lu lut voi thong tin; - user bi frustrate khi ho truy cap hang hoa, dich vu, thong tin ma le ra ho de dang su dung. - doi hoi, yeu cau, kiem tra mot cach cung nhac khong can thiet hay trong nhung trong hop ngoai le.

De lam duoc dieu do, cac thach thuc sau day phai dc giai quyet:

- understandability cong nghe phai transparent to user. tuy nhien phai co cac giao dien truc giac cho nguoi su dung dieu khien. ontologies and natural language processing - tich hop: cong nghe phai lam viec voi nhau, va ho tro nhau, nhung khong doi lai nguoi dung. agent - adaptivity: phai dap ung tot mong muon cua nguoi dung va khong yeu cau qua nhieu su can thiep cua nguoi dung: learning machine.

IV. Requirements

V. A new application model a programming model

- Identifying abstract interaction elements
- Specifying an abstract service description language: A means is needed to express the expected function of a service, allowing for different services to provide this function when the application is running. This must allow for services to be declared optional as well.
- Creating a task-based model for program structure. The application should be delineated into tasks and sub-tasks. A task includes the abstract interaction and the application logic, including the use of the services. The structure is used by the system to generate device specific ”presentation units”; e.g., screens.
- Creating a navigation model. The navigation specifies what causes a task to begin and end (e.g., a user action), and what tasks precede and follow it. This information is complementary to the task structure, and is used by the system to automate the flow of the ”presentation units” when the application is running.

Program structure: - abstract user interfaces, abstract services needed, abstract resources needed. - tasks and subtasks and which tasks are presented to users? –¿ context, how task can be done –¿ context. pre-conditions and post-condition, what tasks truoc va sau mot tasks. - moi task co its requirements, thiet bi, moi truong, ngu canh se kiem chung lieu no co the thuc hien duoc khong va thuc hien theo cach nao la tot nhat. - what task does the user want to accomplish? if the task is a composite of many subtasks, how are these defined to assist the user in his overall task? - what is the flow through the tasks? How does each task begin? how does it end? how does a subtask initiate another in a dynamic framework? - what is the user interaction for each task? what user actions are needed to perform the task and how actions can be identified? - what information needed? where it come from? - how task performance adapt itself to the given environment? - how deal with fails? how user manage the task performance? - design-time - ung dung ma giai dien cua no tuy vao cac thiet bi ngoai vi co san. khong phu thuoc vao thiet bi- device-natural. - dich vu ma ung dung su dung khong nen duoc gan voi mot cai ten cu the. - load-time - run-time

VI. Development Methodology

This methodology would allow a programmer to build an application by answering questions such as:

1) What task does the user want to accomplish? If the task is a composite of many subtasks, how are these defined to assist the user in his/her overall task?
2) What is the ”flow” through the tasks? How does each task begin? How does it end? How does one subtask initiate another in a dynamic framework?
3) What is the user interaction for each task? What user actions are needed to perform the task? How are user actions a reflection of user intent?
4) What information does the user need to perform the task? Where does this information come from?
5) What logic does the system perform for each (sub)task? Is it possible for the (sub)task logic to adapt itself to a given environment?
6) An application must be specified in terms of its requirements
7) Modelling device characteristics and application requirements: The characteristics that axe relevant for differentiating between devices must be codified, and a metric

for each of these characteristics must be developed. The application requirements must be specified in the same terms.

8) Developing negotiation protocols. Such protocols are necessary for a device to ascertain what subset of applications and services can be hosted within the bounds of its resource limitations.
9) it may be desirable to split the execution burden between the device and available servers. This split, which we call apportioning, uses information about the currently available resources and the resource demands of the application. Incorporating fast and efficient apportioning algorithms.
10) it may be desirable to have multiple abstract representations of the application interface, one for each combination of interface modality and form factor.
11) The system needs to support dynamic selection of an appropriate application interface from a set of available interfaces, based on the device's resources and form-factor. The presentation selected in this manner will be specific to an interface modality and form factor. Further adaptation may be necessary for the characteristics of a particular device.
12) The system needs to seamlessly integrate the applications and services found in the environment. This involves composing the functionality as well as the user interface. The composition is subject to the constraints and resource limitations of the device and the composition restrictions of the discovered entities.
13) the run-time must monitor the resources for the currently active portal, or portal set, and appropriately adapt the application to those resources.
14) the run-time must respond to changes initiated by the user. For example, the user may choose a different set of portal devices.
15) The run-time should support handoff of task context from one environment (e.g., office) to another (e.g., car), possibly through a disconnected state.
16) the run-time must be able to take advantage of services provided by the environment and the physical resources available within it.
17) The run-time must handle unexpected failures, such as exhausting batteries or a service crash. Existing failure detection and recovery mechanisms
18) requires the run-time to detect changes in the resources of any portal device or environment hosts that participate in application execution. Resource changes include changes in available network bandwidth, introduction of new devices into the environment, introduction of new users and/or applications, etc. In response to detected changes, the run-time must initiate a reapportionment and/or relocation of application components. Resource changes may impact the user's interaction with the application.
19) Transient resource changes should be recognised as such and should not impact the application. When changes are significant and long-lived, the application should be automatically re-apportioned, with minimal impact on the user.
20) User initiated re-apportioning. The user may initiate re-apportionment of the application. Reasons for reapportionment may range from anticipated change in the connectivity of devices to a mobile user entering the proximity of new devices. In the latter case, the user should be given a choice of whether to use the new devices or not.
21) If the network connection between client and server is detected to degrade via run-time monitoring, the apportioner may react by (1) migrating code from the server to the client to reduce the application's demand for communication, (2) lower quality service; (3) change devices,...
22) explicit support for disconnected operation needs to be added to the model.
23) The run-time should prepare for disconnection without a user's intervention whenever possible.
24) Failure Detection and Recovery

By answering these questions, the programmer will have specified an application at a high level of abstraction. Given the programming model explained above, the implementation will be made up of a task structure annotated with navigation flow, an abstract user interface for each task, and scripting logic that details the task function. The major challenge here is to build a development environment that supports the above methodology.

VII. Technologies Enabling This Model

- User Interface Management Systems [1]
- Client-Server computing model
- Java computing model
- web technology
- service technology

VIII. Advantages

- eliminating the synchronisation problem. When the user updates a phone number, that phone number is the same regardless of the device through which it is accessed.
- the application is built to be run on any device.
- the concept of "upgrading" software may quickly become anachronistic.

IX. Products

- development environment (IDE) - a algorithm for generating device-specific renderings of an abstract interface specification - a layer to allow uniform access to distributed services - a mechanism to be introduced to dynamically vary application apportionment between client and service at run time [9]. - support for failure, disconnection, recovery - develop the interfaces and mechanisms needed to allow an application to identify and use a service at runtime that was unanticipated when the application was written.

X. Discussion and evaluation

Nhung diem moi trong mo hinh cua chung toi:

1. chung toi khong xem cac thiet bi tin hoc trogn vai tro vua la interface vua thuc hien cac chuc nang xu li va dua ra cac quyet dinh. Cac theit bi tin hoc cua chung toi hoan toa chi la giao dien va/hoac sensor tuong tu nhu man hinh, ban phim, chuot, may in, speaker. chung dong vai tro tuong tac va thuc hien cac actions hoac bieu dien thong tin den nguoi dung hoac ket noi voi cac thiet bi khac de lam nheim vu nay. Nhu vay, cac thiet bi khong can thiet chua cac modul phan mem tren no, no cung khong can nhieu cac thanh phan phan can nhu bo nho, bo vi xu ly, trong nhieu truong hopj, no co the co nhu may in co the co bo nho cache hay co mot bo vi xu li don gian (tuy nhien chung ta khong goi chung la computer).

Su phan biet hai levles nhu vay la rat quan trong. No giup cho cac nha design thiet bi va cac nha phat trien phan mem co the tap trung vao linh vuc da duoc chuyen mon hoa cho ho. Dieu nay de dang thay rang se giup cho su phat trien cac he thong se nhanh chong hon. Tuong tu nhu su phan biet giua cac nha san xuat may in, ban phi, man hinh, loa va cac nha san xuat phan mem su dung cac thiet bi do. Chung toi su dung cach tiep can ma chung ta da dung do la su dung cac package ma ta goi la driver de cho may tinh trung tam co the dieu khien duoc cac thiet bi ngoai vi. Cac drivers giup cho he thon trung tam biet duoc kha nang cua cua thiet bi ngoai vi lien ket voi no va lam the nao dieu khien cac thiet bi ngoai vi do. Nhiem vu cua nguoi phat trien phan mem la lam the nao de tich hop cac chuc nang cua cac thiet bi ngoai vi de tao ra cac ung dung ma ho quan tam. Cac thiet bi duoc bao gom trong mot ung dung cu the nao do thanh lap nen mot mang cach thiet bi ma ta goi la a logically/virtually local network. Chung ta goi la logical boi vi thuc te , cac thiet bi trogn mang nay co the o khap noi tren the gioi chu khong phai la trong mot pham vi vat li cu the nao do.

XI. Related work

Nghien cuu cua Henricksen et al. mo hinh PCEs bao gom thiet bi, nguoi dung, thanh phan phan mem, va giao dien nguoi dung. Trong cach tiep can nay, tac gia cho rang cac thanh phan phan mem co the duoc hosted va thuc thi ngay tren cac thiet bi nhung cai von yeu ve kha nang xu li cung nhung nho hep ve bo nho luu tru. vi the, tac gia cho rang, doi hoi can co mot co che adaption to nhung thay doi trong qua trinh du nhap va thuc thi cac component. Quan diem chua chung toi khong bac bo mo hinh phat trien ung dung huong thanh phan, nhung mo hinh cua chung toi khong cho phep cac thanh phan phan mem hosted tren cac thiet bi yeu. Thay vao do, chung duoc hosted va xu li tren cac may tinh trung tam powerful, ket qua se tra ve cho cac thiet bi ngoai vi de chung tuong tac voi nguoi dung hoac thuc thi tac vu nao mong muon.

By [2]:Devices, applications, and environments.

- A device is a portal into an application/data space, not a repository of custom software managed by the user.
- An application is a means by which a user perform a task, not a piece of software that is written to exploit a device's capabilities. app as task.
- The computing environment is the user's information-enhanced physical surroundings, not a virtual space that exists to store and run software.

By [4]: - A framework for building apps, not an infrastructure hosting apps and help them in terms of adaptation dieu nay de lai cho cac nha phat trien ung dung nhung nhiem vu nang ne.

XII. Conclusion and future work

References

[1] Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster. Uiml: an appliance-independent xml user interface language. In *WWW '99: Proceedings of the eighth international conference on World Wide Web*, pages 1695–1708, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[2] Guruduth Banavar, James Beck, Eugene Gluzberg, Jonathan Munson, Jeremy Sussman, and Deborra Zukowski. Challenges: an application model for pervasive computing. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 266–274, New York, NY, USA, 2000. ACM.

[3] Nigel Davies and Hans-Werner Gellersen. Beyond prototypes: Challenges in deploying ubiquitous systems. *IEEE Pervasive Computing*, 1(1):26–35, 2002.

[4] Robert Grimm, Janet Davis, Eric Lemar, Adam Macbeth, Steven Swanson, Thomas Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, and David Wetherall. System support for pervasive applications. *ACM Trans. Comput. Syst.*, 22(4):421–486, 2004.

[5] T. Gu, X. Wang, H. Pung, and D. Zhang. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004.

[6] Yong-Bin Kang and Yusuf Pisan. A survey of major challenges and future directions for next generation pervasive computing, 2006.

[7] Philip Moore, Bin Hu, and Jizheng Wan. Smart-context: A context ontology for pervasive mobile computing. *The Computer Journal*, pages bxm104–, March 2008.

[8] D. Norman. *The invisible computer: why good products can fail, the personal computer is so complex, and information applicances are the solution*. MIT Press, 1998.

[9] H.S. Stone and S.H. Bokhari. Control of distributed processes. *Computer*, 11(7):97–106, July 1978.