Towards a System for Recommending Tasks to Users based on User Intentions and Environment Capabilities

Chuong Vo, Seng W. Loke and Torab Torabi
Department of Computer Science and Computer Engineering, La Trobe University, Australia
ccvo@students.latrobe.edu.au, s.loke@latrobe.edu.au, t.torabi@latrobe.edu.au

ABSTRACT

The burden of managing and utilizing Pervasive Computing Environments (PCEs) comprising a collection of devices on users and devices embedded in the environment (e.g. user's comprehension of PCE capabilities, system configurations, service compositions, and management of system failures) currently falls on users. Users might be overwhelmed by information and capabilities that an environment might offer – for example, there could be thirty to a hundred computers within a living room of the future, each providing various functionalities which the user should then learn how to use (some functionalities perhaps provided by a combination of such devices). Even if the user has a mobile device through which task requests can be issued to this collection of devices, the user would still need to know what are feasible/possible tasks s/he could perform here. Our research aims to minimise this overhead by proposing a task recommender system named TASKREC that manages the PCEs (e.g. services, resources, and context information) where the user is located and suggests to users (via his/her mobile device) tasks that are relevant to them and which can be accomplished by the PCE in which the user is located. Via reasoning with knowledge about current environment capabilities, context information, and user intention, the system recommends to the user possible and relevant tasks when he/she asks for it. In this position paper, we present a formal problem of task recommendation. We propose a conceptual architecture for TASKREC and then provide a scenario for the vision of TASKREC.

Categories and Subject Descriptors
D.2.11 [**Software Architectures**]: Domain-specific architectures.
H.3.3 [**Information Search and Retrieval**]: Selection process.
H.3.4 [**Systems and Software**]: User profiles and alert services.

General Terms
Measurement, Design, Human Factors

Keywords
Task Computing, Recommendation Systems, Context-Awareness, Pervasive Computing.

1. INTRODUCTION

The world is moving towards universally connected information spaces where 'technologies weave themselves into the fabric of everyday life' [11]. Spaces in such a world are called pervasive computing environments (PCEs) [9]. They are collaborative spaces consisting of mobile users, software services, interconnected electronic devices (e.g. set-top boxes, smart-phones, PDAs, laptops, displays, cameras, speakers), and pervasive networks (e.g. HomeRF, Bluetooth, cellular network, Wi-Fi, WiMAX, mobile broadband).

Although PCEs enable us to access information and services anytime and anywhere, they are overwhelming us with an overload of information, services, and complex configurations [2, 4, 10, 12]. This is leading towards a heavy burden to users when they want to accomplish a computing task. Moreover, because of the increasingly sophisticated and feature-rich software applications and computing devices, controlling even one of these devices or applications is increasingly difficult. As a result, controlling and exploiting a PCE which consists of a dynamic range of high-tech devices, feature-rich applications is even more difficult. To exploit a given PCE, the users must (1) recognise feasible tasks in the environment in order to issue feasible tasks; (2) map their high-level goals of tasks to the low-level operational vocabularies of capabilities of devices and functionalities of applications embedded in the environment; and (3) properly specify the constraints for their tasks subject to the context information of their surroundings. These requirements may be beyond ordinary users as the complexity, diversity and sheer number of devices (as well as well different combinations of ways they might work together for the user) continually rises and the amount of information in their surroundings increases exponentially – the user is surrounded by a proliferation of devices in the immediate environment but *might not know what tasks the user can perform through these devices in the environment*. Also, there is a need for solutions which free users from system managements and configurations so that they just focus on their intended tasks.

To address the problem of information overload, there have been research efforts that aim to recommend information and services. However, users are still required to manually integrate the recommended information and services in order to achieve their goals. This calls for task recommender systems. Few efforts (e.g. [1, 6, 7, 8]) have focused on task recommendation. However, the existing systems do not take the capabilities of a PCE as a whole into their recommendation processes. This is critical because accomplishing a task in a PCE often involves a range of services, devices, and context information; the environment capabilities should be utilised in recommending feasible tasks.

We present a context-aware task recommender system (TASKREC) which aims to recommend or even automatically accomplish 'relevant' tasks. Our approach proposes to use three similarity measurements in the task ranking process for recommendation: the similarity of user intention to task objective, the feasibility of tasks, and the autonomy of task performance. TASKREC can help a user make decisions on the question: given the current situation (including the current environment, the current context, and the current user's intentions) what tasks can and should be performed? And how much distraction/attention/obtrusiveness do the selected tasks effect on the user? Further, our proposed system can be integrated into context-aware applications which can adapt their behaviours to changes in the environment and user intention.

The rest of the paper is organised as follows. Section 2 presents a motivating scenario to which our solution is applied. Section 3 defines concepts in our approach. Section 4 presents an overview of our solution for task recommendation and elaborates on the three similarity measurements used by the process of task ranking. We describe a general architecture for TASKREC in Section 5. Section 6 outlines the related work. Finally, a conclusion and an ongoing research agenda are given in Section 7.

2. A MOTIVATING SCENARIO

In this section, we demonstrate the applications of the proposed TASKREC with some aspects of a scenario.

"At a university, a seminar room named 'SEMS room' comprises software-controlled appliances (such as a projector, a large plasma display, a web camera, IP speakers), a desktop computer with presentation applications (such as Microsoft Power Point and Acrobat Reader), and a wireless Ethernet. These facilities are functioning properly and currently provide users the capabilities such as 'slideshow', 'video conference', and 'video playing'.

Bob working at another university is invited to present a seminar on 'Industry-based Learning: Principles and Practices' at the 'SEMS room' at '3pm next Monday'. While preparing for the seminar, he would like to integrate some video clips into the presentation and his colleague, Alice, at a Japanese university is asking him about remotely attending the seminar. However, he may not anticipate the capabilities of the SEMS room so that he would not properly make decisions on these intentions. Fortunately, TASKREC installed on his personal computer infers his intentions and the capabilities of the SEMS room, it recommends to him 'Embedding video clips into his presentation' and 'Accepting his colleagues' request for 'video conference'.

On the day of the seminar, when Bob is driving his car and about to approach the car park for guests at the university where he is giving his seminar, TASKREC on his PDA infers that Bob wants to find a free parking spot (this is because TASKREC uses a collaborative filtering based recommendation method which recommends a solution according to observing previous behaviours of other similar people whose contexts are similar to Bob's context). It detects and recommends the service provided by the university which guides him quickly to a free parking spot.

Fifteen minutes before the seminar, Bob tries to contact Alice to remind her of his seminar. However, he does not know Alice is in a meeting which is finishing in ten minutes. TASKREC on his PDA infers his intention and Alice's situation. It also recognises that her preference for receiving communication at this moment is by 'voice message', 'text message', 'email', and 'contact later'. Hence, TASKREC recommends to him to 'Leave a voice message'.

Before Bob steps into the SEMS room, because TASKREC on his laptop detects the services provided by the Manager System of the room and determines a familiar situation, instead of recommending something, it automatically invokes these services to turn the music off, dim the lights, load his presentation onto the local computer, warm the projector up, and setup the video conference."

3. CONCEPTS AND MODELS

3.1 Task Modelling

A *task* is a computer-supported human activity which is performed by collaborative interactions among the user and devices located in the environment. We conceptualise a task as a human activity, not device functionality. For example, 'Presenting seminar' or 'Watching movie' are tasks but 'Displaying slideshow' or 'Playing movie' are not tasks; they should be called services instead. This is because 'Presenting seminar' or 'Watching movie' are carried out by humans while 'Displaying slideshow' or 'Playing movie' are carried out by machines.

We propose that each task is specified in a structured document called *task specification* which describes *task objective*, *capability requirement*, and *task flow*. Task flows define a sequence of *actions* and/or *sub-tasks* sufficient to achieve the task objective. There are three levels of autonomy of an action/task: *manual*, *semi-autonomous*, and *autonomous*. A manual action/task is accomplished by the user only. At the semi-autonomous level, the action/task is accomplished by collaboration between users and devices. At the autonomous level, the environment automatically performs the entire action. The level of an action/task is determined subject to the current environment capabilities.

A *task space* of a PCE is a repository containing specifications of possible tasks in this environment. The task space can be updated (e.g. add, modify, and remove task specifications) at runtime. Learning techniques should be used to update the task space. The following is an example of adding a new task. The system initially recognises that the seminar room is only used for seminar tasks. However, it observes that the room has been operated as follows: during a session, there is no light on; the video player is connected to the projector; the speakers are turned up; there is no presenter but a large audience; and some video disks are used instead of presentation files or pens. Sure enough, compared to the operation of a movie theatre, a new task perhaps called 'Watching movie' should be added to the task space of this room.

3.2 Environment Modelling

We model a PCE with its current capabilities (hereafter called *environment capabilities*) as one *single virtual device.* We propose to use a virtual device service gateway such as VDSG [3] for discovering and composing device capabilities.

We term a *master device* as one which is carried by or near the user. It is the display of recommended tasks, the initiator of task invocations, and the controller to manipulate all involved secondary/auxiliary devices in accomplishing the task.

It is assumed that the requested capabilities of a task and the provided capabilities of the environment are specified using the same ontology. Therefore, the concepts of the requested capabilities are directly mapped to the concepts of the provided capabilities.

Formally, environment capabilities can be defined as a vector of variables $\mathrm{C} = (c_1, ..., c_n)$, where $c_i, i \in (1, ..., n)$ represents a particular capability of the environment. $dom(c_i)$ is the space of possible values of c_i. The current value of c_i is denoted as v_i. For example,

$$\begin{aligned} \mathrm{C} &= (\text{VideoConference, SlideShow, ColourPrinting}) \\ &= (\text{True, True, False}). \end{aligned}$$

That is, the current environment can exhibit slideshows with the video conference support.

3.3 User Intention Modelling

The *current user intention* (hereafter called *intention*) is the user's instantaneous expectation which is inferred from the user context such as activity, calendar, situation, feedback, profile (e.g., roles, habits, and preferences), location, and time.

Roughly, the description of an intention at a particular point of time defines a goal desired by the user and how these goals should be achieved (i.e. 'constraints'). For example, one of Bob's intentions is to present a seminar on 'Industry-based Learning: Principles and Practices' at the SEMS room at 3pm next Monday. There are some video clips in his presentation. In this example, the task is 'Presenting a seminar' while the constraints of the task are the presentation file 'IBLseminar.ppt', the location 'SEMS room', the time '3pm next Monday', and the video clip files.

We model an intention as a tuple of $\langle doAct, onTarget, useTool, atPlace, atTime, withPerson \rangle$. *doAct* expresses the activity the user desires to perform; *onTarget* is the target object(s) which is the content of or influenced by the task; *useTool* is the device(s) hosting the task or the service(s) to be employed; *atPlace* is place the task occurs in; *atTime* is the moment(s) of the beginning of the task; and *withPerson* is person(s) involved in the task.

4. TASK RECOMMENDATION PROBLEM

TASKREC attempts to be aware of the user's intention and environment capabilities. Therefore, our approach of ranking tasks for the user *u* in the environment *e* with the task space *T* is based on three factors: the intentions I_u, the environment capabilities C_e, and the task specifications of each task *t* ($t \in T$) including task objective O_t, requested capabilities $\mathrm{R}_t = (r_1, \ldots, r_n)$, and task flows P_t. If we define $\mathrm{W}_{t,u,e}$ as the *rating* of a task *t* for the user *u* in the environment *e*, then

$$\mathrm{W}_{t,u,e} = \mathrm{I}_u \times \mathrm{O}_t \times \mathrm{R}_t \times \mathrm{P}_t \times \mathrm{C}_e$$

We introduce three measurements as foundations for our task ranking algorithm. First, *task relevancy* is to measure the similarity of intentions with task objectives. Second, *task feasibility* is the similarity of requested capabilities of a task with provided capabilities of the environment. Finally, *task autonomy* expresses the autonomy of performance of a task. It is a percentage of the task (comprising sub-tasks) which can be automatically carried out by the environment. Task autonomy is measured based on the task flow of a task which is determined depending on the current environment capabilities and the objective of the task.

4.1 Assumptions

To prioritise tasks, we use the three measurements above together with the following assumptions:

- Users always prefer the task which optimally satisfies their intention in any case;
- If there are many tasks which bring users the same degree of satisfaction, the more feasible the task is, the more preferred the task is;
- If two tasks are equally feasible, then the task which is more autonomous has a higher priority.

4.2 Definitions

4.2.1 Task Feasibility

The task feasibility of a task *t*, called F_t, is the feasibility degree of performing the task in a given environment *e*. It is calculated by measuring the similarity between requested capabilities of the task R_t with the environment capabilities C_e:

$$\mathrm{F}_t = 1 - \frac{\sum_{i=1}^{n} dis(r_i, v_i)}{n},$$

where $r_i \in \mathrm{R}_t$ and $v_i \in \mathrm{C}_i$ be values of the concept of a particular capability $c_i \in \mathrm{C}_e$, respectively. Also, $dis(r_i, v_i)$ is the distance between two values r_i and v_i. Here we assume that the importance of a requested capability is equal to another (i.e. their importance weights are all assigned to one).

4.2.2 Task Autonomy

The task autonomy of a task *t* called A_t is the percentage of task performance which can be automatically carried out by the environment. For example, the autonomy of the task 'Present seminar' using a video projector and a desktop computer for showing slides would be greater than the autonomy of the same task in the environment only supporting an overhead projector for showing transparencies.

As mentioned previously, the task flow of a task defines actions or/and sub-tasks sufficient to complete the task. We call $a_1, a_2, \ldots, a_m$ (where *m* is the number of actions/sub-tasks within a task flow) the autonomy degrees of these actions/sub-tasks, then the autonomy degrees of the entire task is:

$$\mathrm{A}_t = \sum_{i=1}^{m} a_i \ .$$

Note that, a feasible task may not be an autonomous task while an autonomous task must be a feasible.

4.2.3 Task Relevancy

The task relevancy of a task *t* denoted S_t indicates the relevance of the task for the current user *u*. It is measured as the similarity between the task objective $\mathrm{O}_t = \{o_1, o_2, \ldots, o_k\}$, and the user intention $\mathrm{I}_u = \{i_1, i_2, \ldots, i_k\}$:

$$\mathrm{S}_t = 1 - \frac{\sum_{j=1}^{k} dis(o_j, i_j)}{k} \ .$$

Note that, o_j and i_j are values of the same concept representing a particular objective/goal. Here, we assume that the importance weights of all intentions/objectives are equal. $dis(o_j, i_j)$ is the distance of two values o_j and i_j.

4.2.4 Task Rating

Let $\mathrm{W}_{t,u,e}$ and $\mathrm{W}_{t',u,e}$ be the ratings of tasks *t* and *t'* for the user *u* in the environment *e*. We say that $\mathrm{W}_{t,u,e} \geq \mathrm{W}_{t',u,e}$ if one of the following clauses is true:

$$\begin{array}{lr} \mathrm{S}_t \geq \mathrm{S}_{t'} & (1). \\ (1) \text{ is false and } \mathrm{F}_t \geq \mathrm{F}_{t'} & (2). \\ (2) \text{ is false and } \mathrm{A}_t \geq \mathrm{A}_{t'} & (3). \end{array}$$

4.2.5 The Most Relevant Task

A task *t* is called the most relevant task for the user *u* in the environment *e* if

$$\mathrm{W}_{t,u,e} = \max\left\{ \mathrm{W}_{t',u,e} \mid \forall t' \in \mathrm{T} \right\}.$$

This section has presented an algorithm for task recommendation based on three measurements: task feasibility, task autonomy, and task relevancy. The next section describes a general architecture for TASKREC.

5. SYSTEM ARCHITECTURE

The main purpose of TASKREC is to rank tasks and recommend to the user the relevant tasks based on their ratings. The ranking process is performed once the system detects the changes of the environment and the user intention or once the user asks for recommendation. The result of the ranking process is a minimal number of the relevant tasks which is initially displayed on the master device (e.g., a smartphone or a large public display near the user) for the user to choose. However, some of these recommended tasks may be performed automatically without asking the user if their relevancy, autonomy, and feasibility degrees are high enough or the user may establish their preference to allow these tasks to be automatically invoked when he is in the similar contexts.

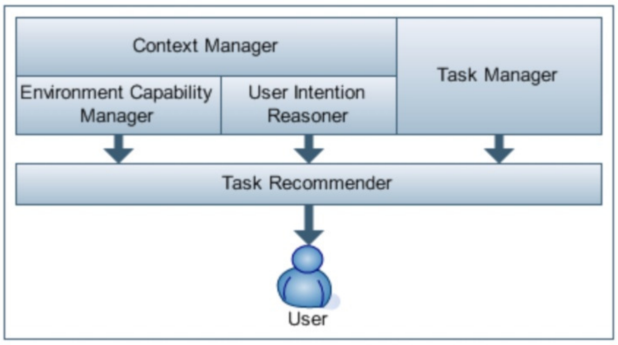

Figure 1. The conceptual architecture for TASKREC.

As indicated in Figure 1, the system consists of five components. Context Manager captures contextual information and distributes it to Environment Capability Manager, User Intention Reasoner, and Task Manager. Task Recommender uses information about environment capabilities, user intention, and task specifications provided by Environment Capability Manager, User Intention Reasoner, and Task Manager for its recommending process. Task Manager is able to learn new task specifications from the environment. User feedbacks and user profiles are captured and managed by User Intention Reasoner. Task Manager executes tasks chosen manually or automatically from recommended tasks.

6. RELATED WORK

There have been much research on task computing [4, 5, 12]. However, little work has been done in recommending tasks given the user and a PCE in which the user is currently situated.

Cheng *et al*. [1] propose an application recommendation system which learns latent situations from usage history and compares them to the current situation to find similar situations. The applications typically performed in these similar situations are ranked and recommended. This approach actually recommends applications while our approach is to recommend tasks which would require multiple applications and devices to be accomplished.

Messer *et al.* [6] propose a middleware called InterPlay for seamless device integration and task orchestration in a networked home. The users express their tasks using a pseudo-English interface and the system will achieve these tasks with minimal user intervention. This approach requires users having in their minds feasible tasks to be accomplished while our approach can recommend to users relevant and feasible tasks in new environments which they can invoke to achieve their intentions based on current environment capabilities.

Ni *et al.* [7] propose an algorithm to discover 'active' tasks in the current context of the user. Their solution can find out feasible tasks but not relevant tasks.

Rantapuska and Lähteenmäki [8] develop a system named Homebird which discovers features of other devices automatically and suggests to the user certain tasks that can be performed together with those devices. Because this approach does not consider current user intentions and current context, its suggestions would disturb users rather than help them. In other words, Homebird can recommend feasible tasks which may be not relevant tasks.

7. CONCLUSION AND FUTURE WORK

In this paper, we introduced the problem of task recommendation in PCEs. Our solution for task recommendation uses user intention, environment capabilities, and task specifications for predicting relevant tasks. As a result, we proposed an algorithm for ranking tasks based on three measurements: task feasibility, task autonomy, and task relevancy. We illustrated the architecture for the task recommender system called TASKREC. The system can benefit users even in unfamiliar environments without pre-training the system.

TASKREC consists of components such as Task Manager, Context Manager, User Intention Reasoner, and Environment Capability Manager. We are currently investigating optimal solutions for implementing these components. In addition, as indicated in the application scenario, a TaskRec client installed on user's mobile devices needs to talk to the local services, retrieve local context information captured by local sensors, and discover local environment capabilities. To enable the system to access such locally unfamiliar environments, we plan to propose a 'cloud infrastructure' which plays the role of a universal middleware for managing and sharing the local knowledge about environment context and capabilities.

8. REFERENCES

[1] D. Cheng, H. Song, H. Cho, S. Jeong, S. Kalasapur, and A. Messer. Mobile situation-aware task recommendation application. In *The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, 2008.

[2] G. Fischer. Articulating the task at hand and making information relevant to it. *Human-Computer Interaction*, 16(2):243–256, 2001.

[3] R. Y. Fu, H. Su, J. C. Fletcher, W. Li, X. X. Liu, S. W. Zhao, and C. Y. Chi. A framework for device capability on demand and virtual device user experience. *Journal of Research and Development*, 48(5-6):635–648, September-November 2004.

[4] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project aura: toward distraction-free pervasive computing. *Pervasive Computing, IEEE*, 1(2):22–31, Apr-Jun 2002.

[5] R. Masuoka, B. Parsia, and Y. Labrou. Task computing – the semantic web meets pervasive computing. *The SemanticWeb - ISWC 2003*, pages 866–881, 2003.

[6] A. Messer, A. Kunjithapatham, M. Sheshagiri, H. Song, P. Kumar, P. Nguyen, and K. H. Yi. Interplay: A middleware for seamless device integration and task orchestration in a networked home. In *PERCOM'06*, pages 296–307, Washington, DC, USA, 2006. IEEE Computer Society.

[7] H. Ni, X. Zhou, D. Zhang, and N. Heng. Context-dependent task computing in pervasive environment. *Ubiquitous Computing Systems*, pages 119–128, 2006.

[8] O. Rantapuska and M. Lähteenmäki. Homebird–task-based user experience for home networks and smart spaces. In *PERMID 2008*, 2008.

[9] M. Satyanarayanan. Pervasive computing: Vision and Challenges. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 8(4):10–17, 2001.

[10] S. G. Thompson and B. Azvine. No pervasive computing without intelligent systems. *BT Technology Journal*, 22(3):39–48, July 2004.

[11] M. Weiser. The computer for the 21st century. *Scientific American*, 3(265):94–104, 1991.

[12] D. G. Zhenyu Wang. Task-driven computing. Technical report, School of Computer Science, Carnegie Mellon University, 2000.