

# TASKREC: A Task-Based User Interface for Smart Spaces

Vo, C.C., Loke, S.W., Torabi, T.  
Department of Computer Science & Computer  
Engineering, La Trobe University  
VIC 3086, Australia  
{c.vo,s.loke,t.torabi}@latrobe.edu.au

Nguyen, T.  
Department of Computer Networking &  
Communications, University of Information  
Technology, Ho Chi Minh City, Vietnam  
tuanna@uit.edu.vn

## ABSTRACT

A smart space is a physical space such as a seminar room or a university campus that is richly and invisibly interwoven with sensors and actuators into everyday objects, and consists of connected devices. This space is often difficult to use and its capability is often invisible from users' awareness, especially for those who are unfamiliar with this space. The difficulty results from the overload of features/configurations offered by individual devices and their combinations while the invisibility comes from the blend of computational elements into the environment. This paper describes TASKREC, a new system that automatically generates a list of high-level tasks supported within a smart space based on user's location, surrounding devices, and user's pointing gestures. TASKREC may automatically execute or guide the user through the accomplishment of the selected task based on the corresponding task model.

## Keywords

Task-Based User Interfaces, Location-Based Task Recommendation, Location Modelling, Task-Driven Computing.

## Categories and Subject Descriptors

H.5.2 [User Interfaces]: Graphical user interfaces.

## 1. INTRODUCTION

Smart spaces allow users to accomplish high-level tasks beyond tasks supported by single device and service. As these spaces increasingly embedded with technologies become more complex and sophisticated, everyday users often find themselves spending more time and effort in understanding, configuring, and exploiting them. Task-driven computing [11] appears to be a promising paradigm to address this problem as it shifts focus to what users want to do (i.e., on the tasks at hand) rather than on the specific means for achieving those tasks [6].

Loke [5] introduces a concept of taskable space, a space of connected devices and ubiquitous services is fitted with a task computing framework. The taskable space supports physical tasks, soft

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MoMM2011, 5-7 December, 2011, Ho Chi Minh City, Vietnam.  
Copyright 2011 ACM 978-1-4503-0785-7/11/12 ...\$10.00.

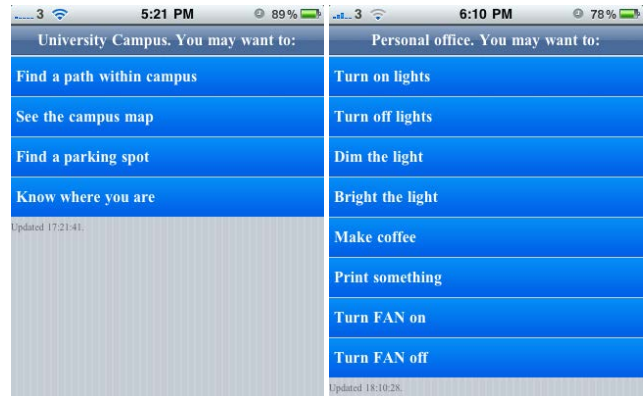


Figure 1: Task-based UIs based on user location. The left screen is a list of recommended tasks supported within the university campus. The right is recommended tasks supported within the personal office.

tasks, combined, and tasks which are accomplished by multiple devices working together. This paper presents TASKREC, a system that automatically generates a list of tasks (presented on the user's smartphone) supported within a smart space based on user's location, surrounding devices, and user's pointing gestures. The list of tasks are suggested by the system to the mobile user according to the capabilities of the current space in which the user is situated and according to what the user points his/her smartphone at. The user can then select tasks s/he wants to do from this task list. The system can then either automatically execute or guide the user through the accomplishment of the selected task.

Let's examine a scenario: When Bob drives into the university campus, the TASKREC client on his mobile phone automatically recommends him a list of available tasks as shown on Figure 1 left. When Bob enters and remains in his personal office, he is recommended a list of tasks supported within his office (see Figure 1 right). While he is walking forward and pointing the phone to the television, a list of television-supported tasks are recommended (see Figure 2 left); similarly Figure 2 right is a list of recommended tasks as he points the phone to the air-conditioner. When Bob selects a task to be accomplished, if the task is toggleable (e.g., turn on/off lights), there will be no user interface for that task; otherwise the system will guide him through the task accomplishment by executing the corresponding task model. The system also supports for multi-tasking. That means, Bob can switch between currently running tasks (e.g., 'Make coffee' and 'Watch TV') to control their executions. He can also define whether some tasks can be automat-

ically executed when his context changes e.g., entering or leaving his office.

Challenges for TASKREC are to accurately infer user’s location and to execute tasks. To address the first challenge, we use multi-model location technologies with various accuracies ranging from hundreds of metres (e.g., the global positioning system) to a few centimetres (e.g., the Cricket system<sup>1</sup>). Accordingly, we propose a multi-level location model to incorporate spaces and sub-spaces ranging from campus scales to device’s vicinity scales. This allows the user to switch between levels to get different task lists. We address the second challenge by using task models specified in the ANSI/CEA-2018 standard [1]. A task model is a formal description of activities involved in completing the corresponding task. It helps the machine understand how a task is performed [9]. Key requirements of task models are to be abstract enough to avoid being bound to individual devices/services; and to be able to incorporate context information into task execution. In the ANSI/CEA-2018 standard, tasks are defined in terms of subtasks; atomic tasks can be “grounded” to actual device functions and services.

TASKREC is implemented on top of a task computing framework fitted into a taskable space (so called space server). This server enables the manipulation, composition, and advertisement of task models for that space. Users of TASKREC use a handheld device (e.g., a PDA or mobile phone) or even their laptop or desktop PC to receive and execute relevant tasks. There is no need for installing a special program on user devices except a web browser because the TASKREC client is actually a web page. Most computational processes are at the space server.

The rest of the paper is organised as follow. First, we briefly present related work in this area. The next section elaborates on the conceptual models, followed by TASKREC’s architecture and implementation. We conclude with a discussion of implementation decisions and future work.

## 2. RELATED WORK

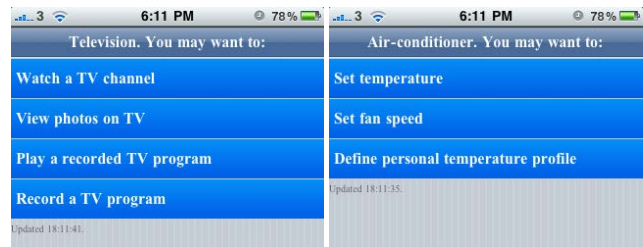
Much work has been done in recommending users applications, services, and resources which may suit their current context. For example, Cheng *et al.* [2] propose a system which recommends users applications installed on a mobile phone. However, having recommended applications, services, and resources, it is still difficult for the users to get them working together to achieve their intended tasks.

Research has focused on automating device integration and task execution in smart spaces. Messer *et al.* [7] implement the *Inter-Play* system that allows a user to express tasks via a pseudo-English interface then the system does the rest. However, because of the automation of device integration, the user may not be aware of tasks the system can supports at a particular moment. Also, the user needs to learn how to express accurately their tasks. These limitations can reduce user acceptance of this system. Similarly, Ranganathan [8] reports a task execution engine, where the user can select a task to be executed, then the system discovers and binds the best available resources for executing the task.

Several work has focused on methods for automatically generating interfaces for systems of multiple connected appliances like Huddle [4]. Huddle is designed to support multimedia tasks which rely on flows of data such as audio or video data. In contrast, we aim to support all types of tasks including soft tasks such as ‘Borrow a book from a library’.

## 3. CONCEPTUAL MODELS

<sup>1</sup><http://cricket.csail.mit.edu/>



**Figure 2: Pointing & Tasking metaphor. Two device-based task interfaces when the user points to a television or an air-conditioner.**

We explain the concepts and ideas we utilized in our task-based conceptualization of smart spaces in this section.

### 3.1 Task-Based User Interfaces

We refer a task-based user interface to as a user interface (UI) which is tailored to users’ current high-level goals called tasks. Such the interface is not to present the user a static matrix of buttons (e.g., for controlling a particular device) or a rigid hierarchical menu like traditional menus on PC; but to present relevant suggestions and guides for the users to accomplish their intended tasks. A typical example of a task-based UI is given in [9] where the UI guides a user through the completion of a task called ‘Borrow a book’.

### 3.2 Context-Aware Task Recommendation

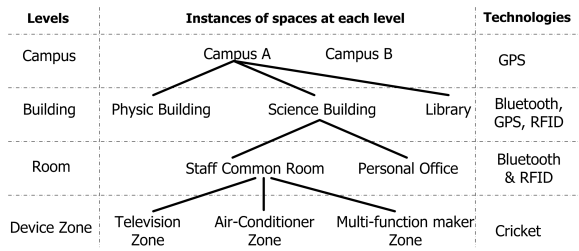
Users want to know or to be informed what tasks they could do in their situation at a specific space. Context-aware task recommendation is a method that uses context information for making task recommendation [10]. Context information can be any information as long as it is useful to characterise user’s situation [3]. In this paper, we consider user’s location for recommending tasks. We call this method location-based task recommendation. Specifically, the system continually updates a user’s location, then the tasks which are supported within the spaces containing that location will be recommended for the user. A space could be at a larger scale such as university campus, or at a medium scale such as library, or at a room scale such as personal office, or even at an object-zone scale (i.e., the surrounding of device, person, or physical object such as book). Figure 1 and 2 are examples of such location-based task recommendations.

### 3.3 Pointing & Tasking Metaphor

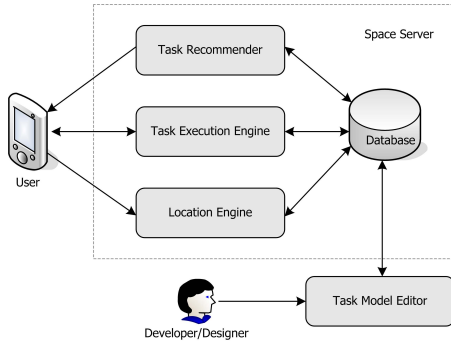
We aim towards scenarios where a user can retrieve a list of object-related tasks when he/she points to a particular object (e.g., a book, a television, or even a person). This scenario would be possible as physical objects in smart spaces are tagged with computational elements such as RFID tags or sensors. Imagining that you scan a book then you would be suggested a task called ‘Borrow the book’; or you approach close to a person at a conference then perhaps you would be suggested to get the profile (e.g., name) of that person. We implement this concept in our prototype where we points our phone which is attached with a special hardware (we will present this in the implementation section) to a television or air-conditioner then related tasks are shown to us (Figure 2).

### 3.4 Multi-level Location Model

Our location-based task recommendation approach needs to know the space the user is currently in. The more specific space is deter-



**Figure 3: Multi-level location modelling.** The first column consists of four different location levels. The second column presents examples of spaces at individual levels. The last column is location technologies used at corresponding levels.



**Figure 4: Architecture of the TASKREC system.**

mined, the more specific tasks are recommended. However, the granularity level of spaces is determined depending on technologies which are used to get user’s location information. It is not flexible if the system is designed to function only in one level of location granularity such as room level or campus level because the tasks which are supported within e.g. a university campus may not be supported within e.g. a specific room or the tasks which are supported within a room may not be supported by a specific device. Therefore, we propose a multi-level location model where spaces at larger scales are divided into sub-spaces at smaller scales. In our design, we have four levels of granularity including campus level, building level, room level, and device zone level. Figure 3 shows the four levels and their instances of spaces. The user is allowed to switch between levels to retrieve different task lists. We think that the more specific the spaces, the more relevant the tasks are to users. That is why currently we do not consider larger scales such as countries in our research.

#### 4. ARCHITECTURE

In the TASKREC system, a handheld device (e.g., a PDA or mobile phone) plays two roles. First, it is the interface between the user and the space. Second, it can provide location information if it has a built-in GPS unit and/or a Bluetooth unit. Users of TASKREC firstly need to register with the space server for that space. During the registration, he/she will active the address of their Bluetooth unit if have. An overall view of TASKREC architecture is shown in Figure 4.

TASKREC requires two types of input to function. The first is a repository of task models for possible tasks supported within the space. Designers use the Task Model Editor to specify task models in XML complying with the ANSI/CEA-2018 standard [1]. Task models can be published and advertised for others to search for and

reuse. We leave the development of a graphical tool for authoring task models in the future work. The critique of the reason of using the ANSI/CEA-2018 standard are given in [9]. The second type of input TASKREC requires is user’s location information. The Location Engine is responsible for updating user’s location and inferring the spaces the user is currently in.

The Task Recommender takes the current space of the user inferred by the Location Engine to recommend tasks. These components work in separate processes but on the same location data. The Task Execution Engine is to execute tasks and to guide the user through task accomplishment.

#### 5. IMPLEMENTATION

To get location information corresponding to the four levels of location granularity, we integrate four location technologies into TASKREC including GPS, Bluetooth, RFID, and Cricket. However, we do not limit our system to any future location technologies. Figure 3 outlines our decision of using these technologies for retrieving location information at different levels. Accordingly, we use the GPS technology to infer outdoor locations (i.e., campus level and building level) because of its high inaccuracy. We set an accuracy threshold to 20 metres. If the location accuracy reported by the GPS unit is greater than the threshold, the campus level will be used to infer the spaces; else the building level will be used.

The Bluetooth and RFID technology is used to infer indoor location information (e.g., buildings and rooms). Users can either use Bluetooth or RFID technology for providing their location information. The Bluetooth addresses and/or ID card numbers (i.e., of their personal ID cards) are provided to the system during users’ registration. The Location Engine has modules for scanning Bluetooth devices in surroundings and ID cards. These modules are assumed to be placed at various locations around the spaces and all connected to the Location Engine. Currently, we use a 13.56Mhz Mifare reader to scan for ID cards and Bluecove-based programs for Bluetooth devices scanning. The event of sweeping the ID card over the reader is a clue to infer user’s entering or leaving the room. Accordingly, if the user has specified some preferred tasks for these events, then they will be executed on the occurrences of corresponding events. We have experimented this concept with ‘Turn on/off light’ and ‘Turn on/off fan’ tasks as preferred tasks for the events of entering/leaving the personal office. Currently, the controlling of the light and fan is implemented using an X10-based HomeAutomation Kit though any other technologies are possible to be incorporated into our system as soon as the designers are provided URL services for controlling desired devices.

We use the Cricket system to infer user’s pointing gestures and device’s proximity. Cricket uses time-difference-of-arrival between radio frequency and ultrasound to obtain distance estimates. The accuracy can be between 1cm and 3cm. Cricket consists of beacons and listeners. A beacon can be worn by the user or attached to the user personal device (e.g., mobile phone). Each listener is attached to a device/appliance in the space and connected to the Location Engine. Cricket is able to provide listener identifiers and their distances to each beacon within their radio ranges at every moment. By comparing these distances, the Location Engine reveals the device/appliance which is currently nearest to the beacon (hence nearest to the user). Then the tasks supported by this device/appliance will be recommended.

To recognise the user’s pointing gestures, we use the line-of-sight connectivity between listeners and beacons. Theoretically, if a listener and a beacon are facing each other, the rate of receiving ultrasound signals at the listener is highest in comparison with a beacon facing a listener at some angles. We have set an experiment where









Positions	Beacons	Listeners	Signals received per 10 seconds
Facing			9.12
45° Angle			6.67
Right facing			1.67
Opposite			0.83

Figure 5: Experiment of beacon-listener facing positions.

a beacon was in turn placed at four different angles facing a listener from a two metre-distance including direct facing, 45° angle, right angle, and opposite (see Figure 5). We recorded statistically four averaged rates of receiving signals at the listener in the four corresponding positions for ten seconds each. The result proved the mentioned assertion. Based on this measurement, the Location Engine is able to infer the device/appliance the user currently points to, then recommend tasks for this device.

## 6. DISCUSSION AND FUTURE WORK

The TASKREC client running on the user device frequently requests for the updated list of tasks from the Task Recommender. The interval between two requests should depend on how frequent the user’s location changes. For example, at the room level if the user is walking within his office consisting of some taskable appliances, then his/her current position would rapidly change from appliance’s zone to another appliance’s zone just in one second. In this case, the interval would be set to one second. However, if the user is walking around the university campus, then his/her current location would only change from space to space in some seconds. Therefore, the interval would be set longer to some seconds. We propose to examine the trend in the user’s location changes in the last ten location updates. Then based on this, the Task Recommender suggests the TASKREC client to set the interval to an appropriate value.

As future work, we plan to develop a graphical tool for efficiently creating, composing, and searching for task models. The task execution engine should be extended to deal with how should the system inform users of task execution progress, errors, and failures. It should also support for migrating tasks (thus migrating UIs of tasks) and address UI adaptation when the user changes the space.

## 7. CONCLUSION

This paper has demonstrated the concept and the design of a task-based user interface for smart spaces. The interface automatically recommends a list of high-level tasks supported within a smart space based on user’s location, surrounding devices, and user’s pointing gestures; and guides the user through the accomplishment of the selected task. We have implemented and tested the system in several spaces including university campus, personal office, and devices/appliances.

We contend that the notion of the task provides users a much higher level of abstraction than individual applications, and this is also seen in operating systems (e.g., Windows), which display

lists of tasks for users, rather than just a menu of applications. We think that such a task abstraction should apply more broadly not just to desktop operating system tasks but also to everyday tasks that users want to perform in their respective spaces, and to things users want to do with their everyday appliances. Our work also leads to the idea of devices or appliances in the future which may or may not have a physical user interface - the modern television today does not have its control panel obviously visible since it is operated with using a remote controller; one can think of various appliances which are “faceless” or does not (need to) have an obviously visible control panel since one can bring up an interface to the device on the smartphone by simply pointing to the appliances (our use of the cricket system may be cumbersome for now but they serve a useful demonstrator - other directional RF technologies can be considered for the same purpose).

In the future, we can expect, apart from task lists being shown on smartphones, the tasks lists can be shown on wearable display devices<sup>2</sup> that resemble eye-glasses or even Internet-enabled contact lenses (with embedded microelectronics).<sup>3</sup>

## 8. REFERENCES

- [1] C. E. Assoc. Task model description (CE Task 1.0), ANSI/CEA-2018, Mar. 2008.
- [2] e. D. Cheng. Mobile situation-aware task recommendation application. In *NGMAST '08*, 2008.
- [3] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, Feb. 2001.
- [4] J. N. et al. Huddle: automatically generating interfaces for systems of multiple connected appliances. In *UIST '06*, pages 279–288, NY, USA, 2006. ACM.
- [5] S. Loke. Building taskable spaces over ubiquitous services. *IEEE Pervasive Computing*, 8(4):72–78, 2009.
- [6] R. Masuoka, B. Parsia, and Y. Labrou. Task computing—the semantic web meets pervasive computing. In *Proceedings of the Second International Semantic Web Conference*, pages 866–881, Florida, USA, 2003.
- [7] A. Messer, A. Kunjithapatham, M. Sheshagiri, H. Song, P. Kumar, P. Nguyen, and K. Yi. InterPlay: A middleware for seamless device integration and task orchestration in a networked home. In *PerCom*, pages 298–307, 2006.
- [8] A. Ranganathan. *A Task Execution Framework for Autonomic Ubiquitous Computing*. PhD thesis, University of Illinois at Urbana-Champaign, 2005.
- [9] C. Rich. Building task-based user interfaces with ANSI/CEA-2018. *Computer*, 42(8):20–27, 2009.
- [10] C. Vo, T. Torabi, and S. Loke. Towards context-aware task recommendation. In *Proceedings of the 4<sup>th</sup> International Conference on Pervasive Computing*, pages 289–292, Taiwan, 2009.
- [11] Z. Wang and D. Garlan. Task-driven computing. Technical report, School of Computer Science, Carnegie Mellon University, 2000.

<sup>2</sup>For example, see <http://www.media.mit.edu/wearables/mithril/>

<sup>3</sup><http://www.complex.com/art-design/2011/03/internet-contact-lenses>