Towards Ubiquitous Government Services through Adaptations with Context and Views in a Three-Tier Architecture

Dickson K.W. Chiu[1], *Senior Member, IEEE,* Dan Hong[2],
S.C. Cheung[2], *Senior Member, IEEE, and* Eleanna Kafeza[3]
[1]*Dickson Computer Systems, 7A Victory Avenue, Kowloon, Hong Kong*
[2]*Department of Computer Science and Engineering, Hong Kong University of Science and Technology*
[3]*Department of Marketing & Communications, Athens University of Economics & Business, Greece*
email: dicksonchiu@ieee.org, csdhong@cse.ust.hk, scc@cse.ust.hk, kafeza@aueb.gr

Abstract

With the recent advances in mobile technologies and infrastructures, citizens start to demand for not just mobile but also ubiquitous access to e-government services. Further with the invention of new interaction devices, the context in which the service is being used becomes an integral part of the activity carried out with the system. This demands a new paradigm for system requirements elicitation and design in order to make good use of such extended context information. Instead of redesigning or adapting existing e-government services in an ad-hoc manner, we introduce a methodology for the elicitation of context-aware adaptation requirements and the matching of context-awareness features to the target context by capability matching. For the implementation of these adaptations, we propose the use of three tiers of views: user interface views, data views, and process views. This approach centers on a novel notion of process views to ubiquitous government (u-government) service adaptation, where mobile users may execute a more concise version or modified procedures of the original process according to their context. The process view also serves as the key mechanism for integrating user interface views and data views. We demonstrate our methodology by extending an e-government appointment service into a mobile one with context support.

1. Introduction

The advancement and widespread of mobile technologies has resulted in an increasing demand for the extension of Internet electronic government (e-government) services to anytime and anywhere access. Although the computing power and bandwidth of these mobile devices are improving, their capabilities are still significantly inferior to desktop computers over the wired Internet. As most existing e-government services are not designed to support users on mobile platforms, they have to be adapted to accommodate these limitations and adequately address citizens' new needs [1].

With the widespread use of mobile devices and the need for ubiquitous computing, the issue of "context" now becomes a hot topic in human computer interaction (HCI) research and development. Let us consider the case study of this paper, extending an e-government appointment service into a mobile and ubiquitous one. There are several reasons why context is important. First, context reduces the input cost and improves efficiency. Explicit input interrupts the user's thoughts and slows down the speed of the interaction. By sensing the environment and interpreting explicit actions, mobile devices could provide a rich and implicit context, such as the users' location and device platform. Second, context may provide an exciting user experience without much effort on the users' part. With the help of context, interactive services such as appointment reminder are accessories that can make citizens' lives easier, by taking account into not just the citizens' location and environment (such as traffic conditions and weather), but also their preferences. Third, citizens benefit from context sharing. We may assume that peers and relatives have similar preferences, which means something that benefit one group member has a higher probability of being preferred by other members.

As such, by sharing the context, systems could provide better services. We have earlier illustrated some advantages of using context for requirement elicitation in ubiquitous tourist services [15]. Local authorities (such as the city government) are the most efficient and competent to coordinate and host these services because they have the most resources or the best knowledge about local contexts [8]. This motivates us to extent the context concept to ubiquitous government (u-government) service adaptation because we believe that other services for citizens and visitors should be evolving in a similar way.

However, moving the interaction beyond the desktop presents many exciting and new challenges to HCI. First, the interface is moving from humans vs. computers to humans vs. context-aware environments. With the popularity of multimodal interactions, more varieties of input and output devices are now being used. A user may have multiple devices while a device may be shared by different people. Resolving the possibly conflicting input of different cooperating devices becomes more crucial than before in the design process. Second, "knowledge in the world" becomes more important [13]. The goal of HCI design is creating a convenient user experience. To better predict a user's behavior, it is important to understand the subtleties of everyday activities. Third, ubiquitous activities are not so task-centric while the majority of usability techniques are. It is not at all clear how to apply task-centric techniques to informal everyday computing situations [3].

To address these requirements of ubiquitous computing, we extend the notion of context, the constantly changing environment, into three categories [6] [25] instead of the narrow perspective that just focuses on location. *Computing context* refers to the hardware configuration used, such as the processors available for a task, the devices accessible for user input and display, and the bandwidth. *User context* represents all the human factors, such as the user's profile, calendars, and profiles. *Physical context* refers to the non-computing-related information provided by a real-world environment, such as location, time, lighting, noise levels, traffic conditions, and temperature. The three categories are equally crucial and they, as a whole, determine the appropriate and customized interaction between the user and the service. Therefore, we propose to use this notion of extended context as a basis of the requirement elicitation for u-government service adaptations.

As for the implementation of these context-based adaptations, we consider the fact that most web-based services are developed with a three-tier architecture. Motivated by the our earlier work [7], we associate these context-based adaptations through different views at all the three tiers, namely, *user-interface views* at the front-end tier, *process views* at the application tier, and *data views* at the back end database tier. These views adapt services to match the corresponding to the requirements of heterogeneous platforms. We demonstrate the applicability of our approach to u-government services adaptation with a case study of extending an e-government appointment service into a mobile one with context support.

In the rest of this paper, section 2 reviews background and related work. Section 3 introduces our methodology and the conceptual model of our approach. Section 4 discusses some typical context-aware requirements and design issues with reference to our model. Section 5 highlights how the adaptation features are implemented with a three-tier view approach. Section 6 concludes the paper with some directions of future research.

2. Background and Related Work

What is context? By the definition of the Oxford Dictionary, context is a circumstance in which something happens or in which something needs to be considered. Many researchers are not satisfied with such a general definition so they have tried to come up with a more accurate one. Schilit et al. [25] claimed that the three important aspects of context are: where you are, who you are with, and what resources are nearby. Chen et al. [6] redefine context as the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user. Moreover, Dey et al. [10] define it as any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Contexts are typically locations, identities and states of people, groups, and computational and physical objects.

A system is context-aware if it can extract, interpret, and use context information and adapt its functionality to the current context of use [19]. The challenge for such a system lies in the complexity of capturing, accessing, and processing contextual data. To capture context information some additional sensors and/or programs are generally required. Context can be acquired several ways. Location information, which is the most popular user context, is easily obtained by sensors (e.g., the Global Positioning System (GPS) in an outdoor environment). Moreover, sensors could sense the temperature, humidity, sound, or even movement in both outdoor and indoor environments. Portable mobile devices such as cell phones, PDAs, and notebooks also provide context by collecting and interpreting user's interaction responses. Abundant context could propagate through wireless cellular networks, wireless LAN networks, wireless Personal Area Networks (PAN), wireless Body Area Networks (BAN), and wired networks [6]. Sensing techniques and wireless network technology together bring context-aware computing into our everyday life. Context is not only used in an application, but is also shared among different applications. In order that context information is accessible among different applications, the context needs to be represented in a common format and properly categorized [9]. In addition to being able to obtain context-information, applications need to have some "intelligent" component which functions as a predictor of a user's intentions. Developers can intelligently use context information in four primary ways [26]: 1) resolving references, 2) tailoring lists of options, 3) triggering automatic behaviors, and 4) tagging information for later retrieval. Thus, our extended notion of context serves as a legitimate basis of the requirements elicitation for ubiquitous applications.

There are many context-aware applications available for mobile devices with features like proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions [25]. Such applications usually present information to a user, automatically execute a service for the user, and tag the context to information to support later retrieval [10]. However, most of these applications are concerned mainly with physical contexts. Here are some examples.

One of the application areas with the most context-aware software is office and meeting tools. The **Active Badge Location System** [31] by Olivetti Research Laboratory in the 1990's is the first context-aware application. System users wear badges that transmit signals providing information about their locations to a centralized location service through a network of sensors. The purpose of the system is to provide a human receptionist with useful information in order to direct calls to the nearest phone. Later experiments have been conducted for automatic phone call forwarding. **Office Assistant** is an agent that interacts with visitors at the office door and manages the office owner's schedule [32]. Based on information such as the owner's working status and available time slots, whether there is a visitor arriving or leaving, and the interaction history, the system could assist users in changing their interaction behaviors. Examples are interactions with other visitors, advising them on their appointments, and updating calendar entries to reflect recent appointments. **CybreMinder** is a context-aware prototype implemented by the Georgia Institute of Technology. It supports users in sending and receiving reminders involving rich context, such as time, place, and working status [11]. The reminder could be delivered via SMS on a cell phone, email, or a nearby display screen.

As for tourist information systems, **Cyberguide** is a mobile context-aware tour guide system for visitors in a tour of the Graphics, Visualization and Usability (GVU) Center Laboratory during open hours [1]. It moves all the information into a hand-held, location-aware unit. By using context information such as location, the direction of movement, and the user's previous locations, the system could suggest some places of interest according to the user's preference. The **Mobile Location-Aware Hand-held Event** is an event planner [14] provides a tourist guide service based on GPS location acquisition. Users may also use this system to send or receive emails. The system also allows the user to set up event reminders. Moreover, the system takes the privacy issue into account. A user could set the visibility based on the persons or groups, which could help to control who can see whom.

Context-aware applications are not limited to research laboratories. For example, Microsoft Direct Watch [22] is a new, specialized wireless service that delivers personalized information through watches that combine style and technology. Services include news, weather, stock quotes, appointment reminders, and personal messages. However, this application requires users to set them up manually since there is no other equipment (i.e., sensors) that will cooperate with them to provide context. Though these techniques are not mature currently, it reveals that the context-aware world is coming.

In order to facilitate the implementation of context-aware software, the Context Toolkit [12] developed by the Georgia Institute of Technology infers the pieces of context automatically from sensors in a physical environment. It separates the acquisition and representation of context from the delivery and response to context by a context-aware application. It aims at the rapid prototyping of a rich space of context-aware applications. However, one of the disadvantages is that the relationship between a widget and a sensor is one-to-one mapping; that is, it cannot support sensors to work together in order to get a data item. Furthermore, one application could not reuse the code of another application due to its way of handling the context interpretation and aggregation.

Burrell et al. [5] summarize some previous context-aware applications systems and propose the Semaphore, a context aware collaborate tool used in wireless networked environments at campus. Lei et al. [20] provide a middleware infrastructure for the design of context-aware applications and try to address the extensibility and privacy issues at the same time. Baradram [4] presents the design of a context-aware pill container and a context-aware hospital bed, both of which react and adapt according to the context in its special domain, a hospital. Kim et al. [18] categorize context-awareness into only non-computational contextual design, non-computational context-aware design, and computational context-aware design. Based on these categories, they analyze how to design context-sensitive appliances. Xu and Cheung [33][34] present techniques to match and detect the inconsistency of contexts based on their semantics. Inconsistent contexts can be resolved by different repairing strategies.

As for mobile government, Sharma and Gupta [27] present a Web services architecture together with issues and challenges, but not the application of context. Abramowicz et al. [1] discusses the issues of user interface design and studies of mobile user needs representation and service description challenges, but do not provide a detailed technical solutions. In summary, all these designers have not explicitly provided a methodology for the requirements elicitation for the design of context-aware applications in a ubiquitous environment and relate them to a systematic implementation based on views.

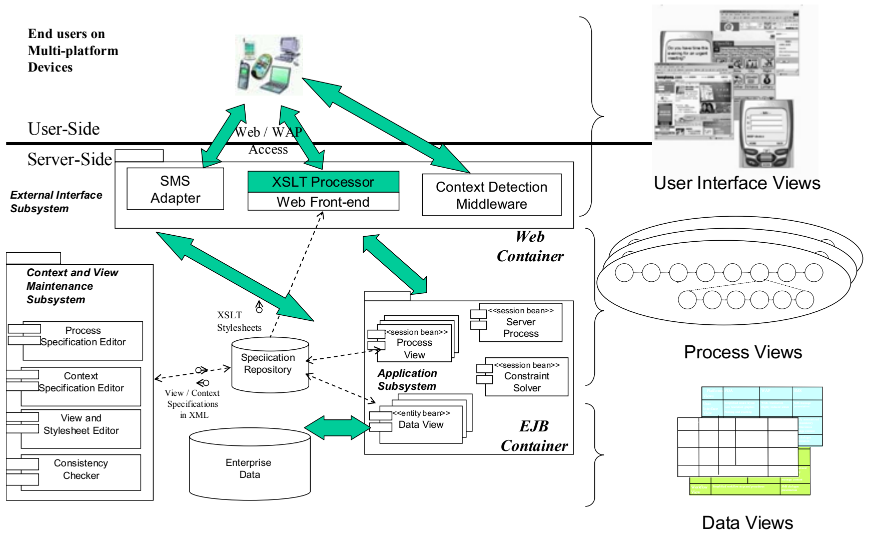

End users on Multi-platform Devices
User-Side
Server-Side
Web / WAP Access
SMS Adapter
XSLT Processor
Web Front-end
Context Detection Middleware
External Interface Subsystem
Web Container
User Interface Views
Context and View Maintenance Subsystem
Process Specification Editor
Context Specification Editor
View and Stylesheet Editor
Consistency Checker
XSLT Stylesheets
Speciication Repository
View / Context Specifications in XML
Enterprise Data
<<session bean>> Server Process
<<session bean>> Process View
<<session bean>> Constraint Solver
Application Subsystem
<<entity bean>> Data View
EJB Container
Process Views
Data Views

Figure 1: Three-Tier system architecture for the realization of three tier views in context-sensitive m-government services

3. Methodology Overview

To elicit the requirements for context-aware u-government service adaptation, we have to consider the two ultimate goals of Interaction Design [23]. One is the usability goal. An interactive system needs to provide effective and efficient services as well as making it easy for the user to learn how to use the system and to remember how to use it. Another is user experience goals. In order to attract people to use interactive services, they must be designed to be usable, effective, and pleasurable to use. The obvious elements for ubiquitous computing are the computing facilities, the human user, and the environment. This is the main reason we extend the notion of context to three main context categories: computing contexts, user contexts, and physical contexts.

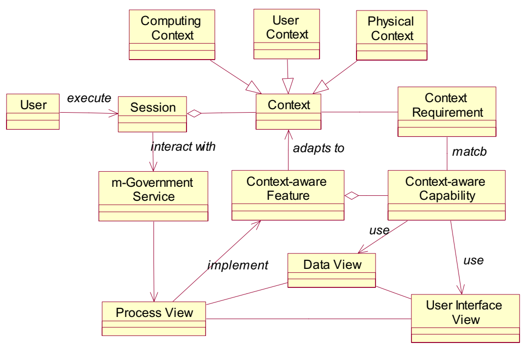


Figure 2. A Meta-model for context-aware service adaptation

The services provided by the context-aware applications need to satisfy users' goals. So understanding the users' expectations becomes crucial in the design process. In order for the system interactions to match the users' common usage, the behaviors of users have to be studied too. The relationship between the environment and users and that among users themselves could be considered as context.

Internet applications are generally developed with a three-tier architecture comprising front ends, application servers, and back end databases. Each of these tiers hosts a set of views as shown in Figure 1, which also depicts our system architecture. Views help balance trust and security, that is, only information necessary for the process enactment, enforcement, and monitoring of the service is made available to the concerned user, in a fully control and understandable manner. At the front-end tier, user interface views provide mobile users with alternative presentations and interfaces to interact with the process views hosted at the application servers tier. At the logical tier, a process view is a projection of a u-government service process that concerns a user and one's context. Process views are synchronized and enacted by application servers. At the backend, a data view consists of multiple tables that collectively represent a projection of data that are required in the enactment of a u-government service, according to process and context requirements.

Figure 2 presents a meta-model for the requirements elicitation of a context-aware u-government service, based on the implementation with the three-tier views. User interface, process, and data views are associated with many-to-many relations. In other words, a user interface view may provide the interface to multiple process views, each of which may in turn supports multiple user interface views. We choose process views as the central mechanism to associate users in different contexts to process views for service enactment as explained in the subsequent sections.

The scenario of extending an existing web-based application to support mobile users is usually more typical than designing a context-aware service from scratch. In addition, the following discussion is also applicable to phased projects in which usually standard web-based application are developed as the first phase, and supporting mobile users as subsequent phases. We propose the design of the views in three tiers to be carried out in the following major steps:

1. Determine the different target groups of users that the system is focusing on.
2. Estimate the typical contexts that such user groups may be situated at, especially those that possibly affect the system's interactions.
3. For each of these contexts, enlist the requirements specific for the context.
4. For each service, investigate how different contexts (from step 2) may impact on its interactions and therefore determine the required context-aware features to adapt to such contexts.
5. For each context-aware feature, detail the desired context-aware capabilities.
6. Make sure that the context-aware capabilities match and satisfy all the requirements elicited from step 3. Otherwise, it would mean that some required context-aware feature are missing and we should re-iterate from step 4 to handle those unaddressed requirements.
7. Design process views that capture typical sets of context-aware features for adaptation.
8. Design data views for each of the data sources based on the requirements of the process views and context-aware capabilities.
9. Design user interface views based on platform dependent restrictions.
10. Perform validation of view consistency.

By adopting this approach, our meta-model further provides a basis for the system to select the required context-aware features at run-time according to the current context of each user session, so that the system uses the same criteria (matching of context requirements with context capabilities) to provide the most appropriate user interactions.

4. Context and Requirements Elicitation

Based on the methodology presented in the previous section, we proceed to analyze some typical context-aware requirements and some design considerations under the three main context categories: computing contexts, user contexts, and physical contexts by considering the requirements of an appointment u-government service. Table 1 tabulates some typical context relevant to ubiquitous computing classified into these three context categories.

Computing Contexts	User Contexts	Physical Contexts
Available devices	Preference	Location
CPU	Purpose	Time
Memory	User calendar	Destination
Screen size	Personal information	Traffic condition
Energy	Facilities	Physical limitations
Bandwidth	Disability	Weather

Table 1 Classification of Some Typical Contexts

4.1. Computing Contexts

In order to design a good interactive service, we must understand the characteristics of the new computing environment. This involves many new input and output devices that may lead to new interactive styles.

One important feature of context-aware services is that users may use many mobile devices. Which devices should be used in the interaction becomes very important. People prefer a device with multiple functions. The system needs to be integrated into the existing equipment (i.e., mobile phones or PDAs) and a heavy device is not acceptable [29]. In this sense, u-government services should ideally run easily on any device that is easily available to a user. Further, one significant difference to the traditional HCI is that multiple devices may be available to a user in a specific situation in ubiquitous computing. Gesture recognition devices, handwriting devices, and embedded cameras together provide more opportunities for users to interact with the environment than with each of them alone. Handwriting devices, voice recognition devices, etc., enable people (especially the elderly) to input their information in a convenient way in different situations. Moreover, with the help of eye-tracking devices, u-government services could benefit those with disabilities since they help automatically choose a target device from a device set.

Another difference between a context-aware environment and a desktop-based scenario is the dynamic nature of the available devices supporting the user's interactions [26]. For example, the devices may be moving dynamically, especially where there are wireless connections among the devices. How to efficiently send the changing context back to the application has attracted a lot of researchers' attention. A lot of work has been done in adhoc networks and sensor networks. The dynamics of devices challenges the routing of information. Furthermore, the dynamic changes to the devices should be transparent to the users. Manual change of device selection should be avoided.

In a desktop scenario, it is reasonable to assume that user sees every message displayed on the screen or hears a notification played through the speakers since the user is focusing on the computer. In a ubiquitous environment, there may not be a focal point for the user to fully concentrate on [26]. It is common for a user to miss a call in a noisy environment since the ring tone is not loud enough to be heard. Moreover, users may no longer be static in an environment. It is also important that users maintain continuous interactions without noticing the changes in the device.

Mobile devices have a limited computing (CPU) power. Even the most advanced PDA could not compete with common desktop computers. The limited CPU power challenges the application installed on mobile devices. For example, the CPU of a mobile device not only has to compute the user's location information, but also to render graphical information. Therefore, most of the computation is preferred to be done at the server side. Memory is another important computing context during the interaction, since it is used to store the user profile and the context-ware applications as well as to provisionally keep some application data. For example, maintaining a complex map of governmental offices (and definitely not a 3-dimensional visualization at the current state of the art) and location for appointment should probably be avoided because of CPU and memory concerned.

In traditional interactions, who is currently using the system is usually not an issue because there is often only one user or the identity can be easily found from the login information. In a context-aware environment, there are likely many people interacting with different devices concurrently on either independent or cooperative tasks. Each user may have multiple devices interacting and each device may be shared by multiple users. Detailed information provided by the system may not satisfy a user's needs [17]. The database must maintain enough detailed information to suit the needs of diverse users. For the u-government appointment service, the database might choose to contain only the detailed paths in the city around the government offices and provide customized maps to users according to their individual contexts. Moreover, it would be helpful if the u-government services could work with other databases (such as a public transportation or route database) in order to provide other extended services.

Monitors and big screens are no longer the only display devices available. The screens on mobile phones or PDA are more convenient when a user moves around. However, the screens on mobile devices are usually much smaller. Avoiding too much scrolling requirements of the mobile user interface would require careful design.

When people buy mobile devices, they care about the duration of battery power. No one wants to buy a mobile phone which can only be used for a short while. Context-aware applications suffer the same problem. It is important for mobile device users to select a reasonable protocol to communicate with the service providers in order to save energy. Moreover, in the design process, developers might take the energy savings during system idle times into consideration because to display a map on a colorful screen is very battery power consuming. In a ubiquitous environment, devices communicate by wireless LAN, Bluetooth, or infrared, etc. The bandwidth is limited compared with the wired networks. Therefore, to develop a context-aware application, we should not assume a broad bandwidth for every user. We also need to con-

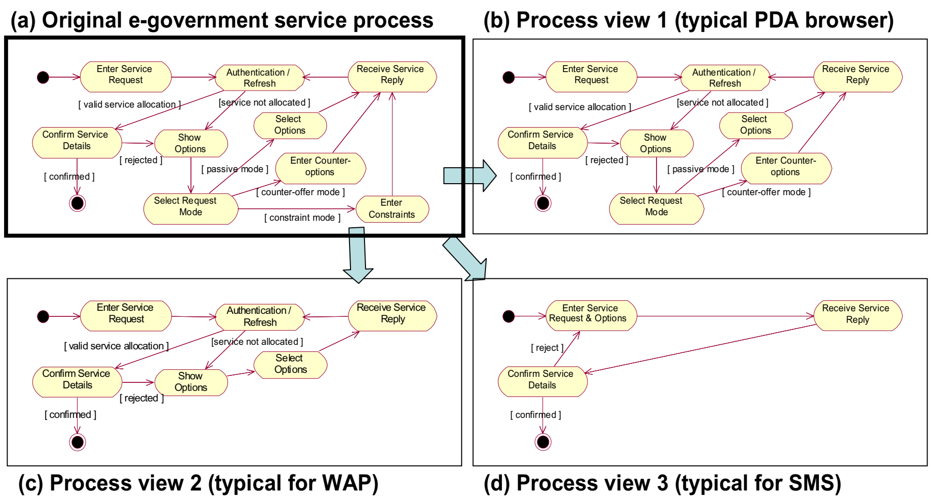


Figure 3. E-government appointment process and u-government process views for different adaptations.

sider the network delay and bandwidth limitation.

Therefore, the traditional approach of considering only the current available device is inadequate. We have to consider other factors for the required adaptation. Moreover, the adaptations are not only restricted to user interface design, but also to the data and logical tiers.

4.2. User Contexts

Users usually have their own preferences that determine their choices. It is very useful for applications to make use of such preference information. Further, if the system knows a user's purpose of an activity, wrong recommendations or interactions can be avoided. For example, if a user's purpose for an appointment is urgent, the system may try to arrange a venue by minimizing the traveling time together with the waiting time.

Context-aware applications may collect some personal data to deduce the interaction between the user and ubiquitous environment. Integrating the appointment service with the user calendar is definitely effective as appointment time at the users' occupied time slots can simply be avoided. Other information, such as the user's working and home addresses are also useful for suggesting a venue.

The ubiquitous computing environment makes privacy data management different from other application domains [16]. The ubiquitous sensors are unobtrusively deployed and work all the time, but they are often invisible. This non-intrusive nature of ubiquitous computing causes user concern because less information can indicate if privacy data are collected and used. However, it is inappropriate to send a permission request to the privacy data owner each time when such data are used. Therefore, how to protect privacy information becomes an important design issue for future research.

Further the system should have special concerns for the disabled. For example, a governmental office with easy access facilities and good transportation connections should be chosen for those in wheelchairs.

4.3. Physical Contexts

Physical contexts refer to non-computing-related information provided by a real-world environment. These are best explored by other existing ubiquitous applications and therefore we try to keep the discussion brief in this paper. Among them, the user's current location is the most important context, especially when appointment venue is concerned. Not only is this information critical to most interactions, it is also the key to many other system functions, decisions, and recommendations. Further, the user's destination (i.e., the location where the user wants to go) together with the current location are the major determinant to the route of the user. In determining a route, traffic conditions such as information about traffic jams and road repair projects help users to avoid these problematic locations. Further, weather (sunny, cloudy, or rainy) and temperature impact on the recommendation of some the appointment activities such as outdoor gatherings and functions.

In general, users need topical information [17]. When users are traveling, context-sensitive information such as weather report, transportation information, and local news may become invalid and need to be updated in a timely fashion. For example, when a user is traveling away and is far from the appointment venue, the system may consider the user not attending and perhaps reassign the current appointment slot to those waiting for services at the venue. In this way, government resources could be better utilized.

5. Three-tier View-based Implementation of Context-based Adaptations

5.1. Process Views

Motivated by views in federated object databases, we propose the use of process views as a fundamental mechanism for flexible u-government service adaptation. A process view is a structurally correct subset of a process definition (as defined in [7]). The concept of process views enables different users (on different platforms) to access different customized version of the same service. Process views are also useful for security applications, such as access restriction (like the use of views in databases). Thus, process views serve as the centric mechanism in our approach, that is, process views represent customized service processes that integrate with data views and user interface views.

Figure 3 illustrates some process views of our motivating example, service appointment u-government service, in the Unified Modeling Language (UML). Upon successful authentication to a service request, users may specify their options in three modes, namely, passive mode, counter-offer mode, and constraint mode. In the passive mode, the user just selects the acceptable alternatives from the options list. However, in the counter-offer mode, the user actively suggests other counter-options through simple form-filling. In the advanced constraint mode, the user specifies constraints representing his/her preferences through complex form-filling or by using a constraint editor.

Now, let us consider the limitations of various context and therefore the required adaptations. The passive mode is simple enough for any browsers including those for Personal Digital Assistant (PDA) with wireless connections as well as mobiles phone supporting the Wireless Application Protocol (WAP). For the counter-option mode, a PDA is fine, but entering options through forms in a WAP phone is quite inconvenient and is therefore not recommended. For the constraint mode, a desktop computer is normally required because of the process complexity and the required screen size. Further considering the message based communication pattern of a mobile phone using Short Message Service (SMS), entering a request together with its options saves one additional message and is therefore more appropriate. Summarizing these requirements, we can arrive at the adapted view design as depicted in Figure 3(b)-(d), respectively. It should be noted that the task *Show Options* is not customized at the process tier, but at the *user interface tier* instead. This is because just the appearance of the screens and input are different while they are referring to the same set of information items, as explained in the section on user interface view.

In order to capture the requirements for users of a mobile platform, a commonly adopted approach from object-oriented design is to carry out use-case analysis. We should concentrate on the difference of the requirements for the mobile users from those under standard browsers. These differences are compared with the standard processes to formulate views. We should identify similar or identical tasks to maximize reuse, and in particular, consider the possibility of customizing them with data views and user interface views instead of rewriting them.

Usually, a complete detailed service process is too complicated for a mobile environment. Therefore, typical adaptations are simplification of the process, reordering of work steps, task delegation to other personal or to a software agent. For example, a user with SMS support can only make very simple decisions or feedback. As this kind of operating environment is often error-prone and may have security problems, we suggest not allowing critical options in these process views.

It should be noted that the views just discussed serve as templates only. User platform is just one of the context types. It is often difficult to tell if a user can tolerate a complex process just because of the device. It could be due to the user's operating environment, preferences, or even due to the user's mood. Therefore, it is always a good idea to consider more context types and information to determine the required and preferred adaptation.

5.2. Data Views

A data view is a set of tables comprising a projection of the enterprise data required for the enactment of some process views. Note that a data table corresponds to either a class or an association. Table columns are represented by the class attributes. With a process view defined, we can proceed to analyze its data requirement. Usually, only part of the application data is required for a process and part of them in turn are required for a process view. Each work step requires data from the database in some form. In particular, we should identify mandatory fields, optional fields, and fields that are to be skipped in the view, in order to cope with the simplification required for mobile users. However, additional fields those have to be computed for summarizing information and knowledge may be required. In case that the mobile user cannot provide mandatory information or input them effectively, we may need to modify the process so that the mandatory information can be provided later or by someone else. Therefore, like designing software, the process is not a linear one. In addition, security requirements should also be considered. Sensitive information may have to be restricted to users within the office or to those of pre-approved locations. *Data views* may also be employed to hide less important data field or to show alternate summary columns.

In this example, the main tables concerned related to the appointment service are schedules, options, and constraints. For each specific type of service, the tables and their rows required are only a subset. For each user's context, we can further define specific data views to restrict such access and for the purpose of customization. In particular, the WAP platform often has restrictions and needs summaries while the SMS platform always needs this (as shown in Figure 5). Moreover, even if integration with the user calendar is implemented, only the relevant part should be exchanged because of privacy reasons.

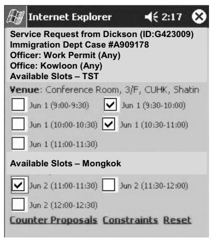
5.3. User Interface Views

User interface views provide users with appropriate interfaces to interact with process views within the capabilities of front end devices. This means the user interface views for a web user is different from that for a WAP user. As shown in Figure 2, we use XSL technology [21] for the implementation of user interface views. Figure 4 presents two possible user interface views that support the *Show Options* task in the process views for PDA users and WAP users, respectively. Information to be presented at a user interface view is structured as XML document objects by a process view. Both Figure 4 (a) and (b) correspond to the same XML document object but are rendered by two different XSL style sheets. For example, the presentation objects for a WAP user interface view are decks and cards in the Wireless Markup Language (WML) while a PDA browser interface still uses HTML.

We usually need to remove graphics or reduce the resolution, provide panning and zooming, shorten fields or provide summarized ones instead, break one web page into several screens, etc. Due to device limitations, multimedia contents, such as pictures, animations and videos, are omitted from the screens in the user interface view for WAP users. For user input, we should consider the difficulties in entering data (especially typing) on mobile devices, and provide menu selections as far as possible. For PDA interface, the main problem is just a smaller screen, some of which may be black-and-white. If the original full-function user interface is too complicated (e.g., too many unnecessary features or high resolution screen layout), another simplified *user interface view* is probably required. Pictures and documents may require to be shown in lower resolution and documents may be outlined and level-structured. Panning and zooming (supported by most browsers) should also help. For users on a WAP interface connecting to the application server via a WAP gateway, the screen is extremely small. A *user interface view* is mandatory to map the original one into WML.

It should be noted that other than devices, context such as the user's eye-sight, lighting, environment, and preference also have impact on the user interviews.

(a)



Service Request from Dickson (ID:G423009)
Immigration Dept Case #A909178
Officer: Work Permit (Any)
Office: Kowloon (Any)
Available Slots – TST

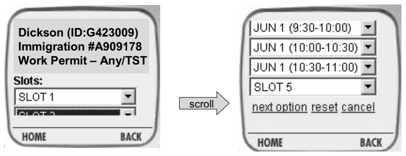
Venue: Conference Room, 3/F, CUHK, Shatin

☐ Jun 1 (9:00-9:30) [✓ Jun 1 (9:30-10:00)]
☐ Jun 1 (10:00-10:30) [✓ Jun 1 (10:30-11:00)]
☐ Jun 1 (11:00-11:30)

Available Slots – Mongkok

[✓ Jun 2 (11:00-11:30)] ☐ Jun 2 (11:30-12:00)
☐ Jun 2 (12:00-12:30)

(b)



Dickson (ID:G423009)
Immigration #A909178
Work Permit – Any/TST

Slots:
SLOT 1

JUN 1 (9:30-10:00)
JUN 1 (10:00-10:30)
JUN 1 (10:30-11:00)
SLOT 5

Figure 4. Screens of the Show Options task for (a) PDA users and (b) WAP users

6. Conclusion

As demonstrated in this case study, adaptation of e-government services to mobile ones is more than merely patching user front-ends: adaptation of the processes is essential to cope with different user contexts. We have proposed the use of context to elicit the requirements of the adaptations and a view-based mechanisms to all the tiers of a three-tier architecture (namely, *process view*, *user interface view*, and *data view*) to implement such adaptations, driven by the process view mechanism. In general, there is no universal recipe for designing process views, as this process is analogous to designing a piece of software. However, a practical guideline is to analyze the requirement of the service against that of the target context, and then select among the alternative viable adaptations. The consistency of such adaptations can be verified against its definition based on process algebra and automata theory [7]. In this way, we can also identify similar or identical views at each of the tiers to maximize possible reuse of the original services under different contexts.

We can also observe that even for simple and common services, adapting them toward a ubiquitous one is far much complex then previously expected, if we take into account various possible improvements that can be introduced through the consideration of context. Our view-based approach further facilitates systematic planning and deployment of such ubiquitous services in phases. This is necessary also because the general experience in building context-aware services is new. We further believe that such adaptation for government and public services is worthwhile despite of the complexity and they should be the pioneer because (i) they usually have the largest number of possible users (citizens and visitors), (ii) better governmental services help increase the productivity of the general public, (iii) their access to various citizens' contexts are relatively easy, (iv) they are usually more trusted as compared with other private services. Further, after gathering the experiences, the government can play an active role in coordinating the access of citizens' context for their benefit, including legislation and enforcing legal civil usage.

Further enhancement of u-government services could possibly with handling exception and alternatives due to the unavailability of the required personnel, network delay, interrupts, disconnections, and so on. In the worst case when a process is aborted, the system should consider additional requirements for compensation processes (for example, a clean up process for a service cancellation) under a mobile environment. We admit that some existing services have to be adapted or optimized for the introduction of new service parameters and information involved after the introduction of user mobility. Though this cannot be solely addressed by the three-tier view architecture, our approach still helps identify these extra adaptations and keeps changes systematic and reusable.

As the specification of process views is based on the standard artifact of UML activity diagrams while the three-tier architecture and contemporary XML are widely employed, the techniques presented in this paper can be applicable to other systems. Through our approach of incremental and systematic adaptation of m-service, fast deployment as well as reduction in development and maintenance cost can be facilitated. Thus, u-government service provision can be streamlined.

Our research direction focuses on the requirement elicitation for context-aware applications, especially on how composite context information and their requirements could be effectively elicited with the consideration of possible conflicts and implications, especially privacy issues [16]. We are especially interested in applications for ubiquitous collaboration and agent-based computing under this emerging computing environment. We are also interested in evaluation metrics for the selection among viable adaptations.

References

[1] Witold Abramowicz, Andrzej Bassara, Agata Filipowska, Marek Wiśniewski, Paweł Żebrowski. Mobile Implications for m-Government Platform Design. Cybernetics & Systems 27(2-3):119-135, 2006.

[2] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. Wireless Networks, 3(5):421–433, Oct. 1997.

[3] G. D. Abowd and E. D. Mynatt. Charting past, present, and future research in ubiquitous computing. ACM Transactions on Computer-Human Interaction, 7(1):29–58, March 2000.

[4] J. E. Bardram. Applications of context-aware computing in hospital work: examples and design principles. In 2004 ACM Symposium on Applied Computing, pages 1574–1579, Nicosia, Cyprus, March 2004.

[5] J. Burrell, P. Treadwell, and G. K. Gay. Designing for context: usability in a ubiquitous environment. In Conference on Universal Usability, pages 80–84, Arlington, Virginia, United States, June 2000. ACM.

[6] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College, 2000.

[7] D.K.W. Chiu, S.C. Cheung, E. Kafeza and H.F. Leung. A Three-Tier View Methodology for adapting M-services, IEEE Transactions on Systems, Man and Cybernetics, Part A, 33(6):725-741, Nov 2003.

[8] D.K.W. Chiu and H.F. Leung. Next Generation Multi-platform Tourist Assistance Systems: A Multi-agent and Semantic Web Approach, ICEC 2005, Xian, China, August 2005.

[9] C. Ciavarella and F. Paternò. The design of a handheld, location-aware guide for indoor environments. Personal Ubiquitous Computing, 8:82–91, 2004.

[10] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology, June 1999.

[11] A. K. Dey and G. D. Abowd. Cybreminder: A context-aware system for supporting reminders. In The Second International Symposium on Handheld and Ubiquitous Computing (HUC2000), pages 157–171, Bristol, UK, 2000. Springer-Verlag.

[12] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 16:97–166, 2001.

[13] A. Dix, J. Finlay, G. D. Abowd, and R. Beale. Human-Computer Interaction. Prentice Hall, Harlow, England, third edition, 2004.

[14] R. Fithian, G. Iachello, J. Moghazy, Z. Pousman, and J. Stasko. The design and evaluation of a mobile location-aware handheld event planner. In MobileHCI 2003, pages 145–160, Udine, Italy, September 2003. Springer-Verlag.

[15] Dan Hong, D.K.W. Chiu, and V. Y. Shen, Requirements Elicitation for the Design of Context-aware Applications in a Ubiquitous Environment, ICEC 2005, Xian, China, August 2005.

[16] Dan Hong, Mingxuan Yuan, and V. Y. Shen. Dynamic Privacy Management: a Plug-in Service for the Middleware in Pervasive Computing. In proceeding of MobileHCI'05, Salzburg, Austria, pp. 1-8, 2005.

[17] E. Kaasinen. User needs for location-aware mobile services. Personal Ubiquitous Computing, 7:70–79, 2003.

[18] S. W. Kim, S. H. Park, J. Lee, Y. K. Jin, H.-M. Park, A. Chung, S. Choi, and W. S. Choi. Sensible appliances: applying context-awareness to appliance design. Personal Ubiquitous Computing, 8(3-4):184–191, July 2004.

[19] M. Korkea-aho. Context-aware application surveys. http://www.hut.fi/ mkorkeaa/doc/context-aware.html, 2000.

[20] H. Lei, D. M. Sow, J. S. Davis, G. Banavar, and M. R. Ebling. The design and applications of a context service. SIGMOBILE Mobile Computing and Communications Review, 6(4):45-55, 2002.

[21] Y.-B. Lin and I. Chlamtac. Wireless and Mobile Network Architectures, John Wiley & Sons, 2000.

[22] Microsoft. Microsoft direct. http://www.msndirect.com/what/faq.htm, 2004.

[23] Object Management Group. Foreword UML specification 1.4, September 2001.

[24] J. Preece, Y. Rogers, and H. Sharp. Interaction Design: Beyong Human-Computer Interation. John Wiley and Sons, New York, 2002.

[25] B. N. Schilit, N. Adams, and R. Want. Context-aware computing applications. In IEEE Workshop on Mobile Computing Systems and Applications, pages 85-90, Santa Cruz, CA, US, 1994. IEEE.

[26] S. A. N. Shafer, B. Brumitt, and J. Cadiz. Interaction issues in context-aware intelligent environments. Human-Computer Interaction, 16:363–378, 2001.

[27] Sushil K. Sharma and Jatinder N.D. Gupta. Web services architecture for m-government: issues and challenges. Electronic Government 1(4):462-474, 2004.

[28] B. Shneiderman and C. Plainsant. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley, San Francisco, USA. Fourth edition, 2005.

[29] Y. Sumi, T. Etani, S. Felsyand, N. Simonetz, K. Kobayashix, and K. Mase. Cyberguide: A mobile context-aware tour guide. In Community Computing and Support Systems, pages 137–154, Kyoto, Japan, 1998. Springer-Verlag.

[30] W3C. Platform for privacy preferences (p3p) project. http://www.w3.org/P3P/.

[31] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. ACM Transactions on Information Systems, 10(1):91–102, Jan. 1992.

[32] M. Weiser. The computer for the 21th century. Mobile Computing and Communications Review, 3(3):94–104, 1991.

[33] Chang Xu, S.C. Cheung and Xiangye Xiao, Semantic Interpretation and Matching of Web Services, in Proceedings of the 23rd International Conference on Conceptual Modeling (ER 2004), Shanghai, November 2004, LNCS 3288, pp. 542-554.

[34] Chang Xu and S.C. Cheung, Inconsistency Detection and Resolution for Context-Aware Middleware Support, in Proceedings of the Joint 10th European Software Engineering Conference and 13th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2005), Lisbon, Portugal, September 5-9, 2005. To appear.