Mobile Web Widgets: Enabler of Enterprise Mobility Work

Alison Lee
Nokia Research Center – Palo Alto
955 Page Mill Road
Palo Alto, CA 94306
+1-650-796-5392

Alison.Lee@nokia.com

ABSTRACT

Increasingly, many smartphones (i.e., Apple iPhone, Google Android, PalmPre, Blackberry Storm and Bold, and Nokia S60) today are equipped with standards-based Web browsers that facilitate access to and interaction with the Web. In some cases, like Nokia S60 series, the smartphones add a web application runtime environment that runs widgets. Like the desktop widgets, Web-based mobile widgets are lightweight mobile applications developed with standards-based Web technologies, such as HTML, CSS, JavaScript, and AJAX. They serve as front ends to Web 2.0 services or Internet content. Newer versions of the S60 web runtime will also have access to platform resources and capabilities that enable more personal and context-aware widgets. Platform service access and mobile work context provide two key differentiators of mobile Web widgets from their desktop counterparts. As such, the mashups enabled by mobile Web widgets have the potential to empower large numbers of smartphones with capabilities and services that are hitherto limited by heterogeneity of device operating systems.

Categories and Subject Descriptors

D.2.m [**Software Engineering**]: Miscellaneous – *rapid prototyping*. H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *web-based services, data sharing.* H.4.3 [**Information Systems Applications**]: Communications Applications – *Computer conferencing, teleconferencing, and videoconferencing, information browsers*. H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *graphical user interfaces, prototyping, screen design, user-centered design.*

General Terms

Design, Experimentation, Human Factors.

Keywords

AJAX, communication, contacts, location, mashups, mobile information access, mobile meetings, mobile Web, mobile Web server, personal information management, mobile Web widgets, platform resources, shared voice and data, smartphones, Web 2.0.

1. INTRODUCTION

While overall mobile phone sales are projected to decline, smartphone sales are expected to grow. This is due to the fact that mobile phones have grown beyond being mobile communication devices. Increasingly, they are used for a variety of other mobile tasks; the least of which is access to mobile Internet services. In the enterprise domain, there is growing interest for mobile phones to support mobile work. The multi-function capabilities of smartphones such as standards-based Web browser and rich media support and the availability of key features such as GPS, camera, and accelerometer are seen as critical to the development of rich, context-aware services. Such services are particularly important for mobile context where a variety of situational constraints can be addressed by leveraging contextual information.

Studies of work reveal that mobility is common in everyday work. At the macro-level, work requires local travel such as walking to other buildings at the same site or remote travel to meet customers or non-co-located colleagues. At the micro-level, mobility is a necessary precursor for some coordination and communication activities [1]. Studies of the mobile context and technology use reveal differences in how work is done compared to the desk context and which applications are used compared to PCs or laptops [9][16][23]. For example, the unpredictability of information access and unfamiliarity with mobile environment pose unique difficulties. As a result, people use coping strategies and perform preparatory work in anticipation of going mobile known as mobilization work [18][19]. Studies of mobile device use have also elucidated a number of tradeoffs people make and the need to align the effort with work, mobility, and social situation [18]. Tradeoffs include time, effort, resource deprivation, distraction posed by working with the device, "instant on" nature of device, and degree of interruption. Finally, studies of information practice such as people's use of information scraps while being mobile highlight many unmet design needs in current mobile personal information management tools [3].

These studies point to a large opportunity space for tools to support mobile information access, mobile communication, and other activities around mobile work. Many of these applications are integration of information access and functionality in a manner that is suited for the mobile context. Thus, mobile Internet services and mobile widgets have much to offer in this space by enabling specialized mashups [20]. The mashups are not limited only to aggregation, projection, and cross product of external data and resources but could include data and resources local to the smartphone such as contacts and GPS. As these mashups are developed with standards-based Web technologies, such as HTML, CSS, (asynchronous) Javascript, and XML, all that is required is a Web browser or Web runtime to host the mashup [20]. Many smartphones are equipped with a Web browser (i.e., Apple iPhone, Google Android, PalmPre, Blackberry Storm and Bold, Windows Mobile, and Nokia S60). Newer Nokia S60 smart

WWW 2009, April 20-24, 2009, Madrid, Spain.

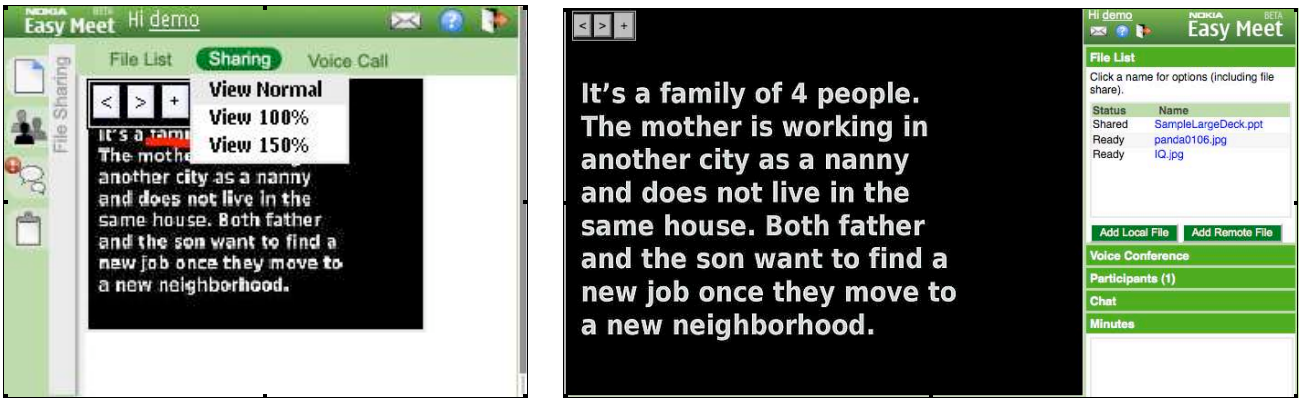

Figure 1: Two different screen layouts – tabbed, accordion.

phones also include a widget engine for mobile Web widgets. As a result, mobile Web mashups have the potential of running on heterogeneous smartphones.

The next section reviews the current state of mobile Web applications and mobile widgets. Then, we introduce a number of widgets that we have developed to expand on the power of mobile Web widgets to enhance mobile work. The widgets tap smartphone resources (e.g., contacts, cameras, GPS) and combine them with the Web of services and content available to support needs and activities of workers while being mobile. Each widget's development was motivated by and addressed different mobile work needs. The examples illustrate the value of the web runtime framework and demonstrate the potential of mobile Web widgets and smartphones as key enablers of mobile work. Section 4 explores issues related to mobile user experience, mobile information needs, and context modeling opportunities. We also highlight some key technical challenges that need to be addressed before mobile Web widgets can unleash the power of smartphones as an integral tool for mobile worker.

2. WIDGETS AND MOBILE WIDGETS

Widgets are lightweight, task-specific, mini-applications that leverage local and/or Web content. Widgets (also known as gadgets) execute within a runtime environment known as a widget engine. Thus, launching a widget does not require launching a Web browser and loading the widget. Like Web applications, the Web and mobile Web variants of widgets are created using HTML, CSS, Javascript and XML. Unlike Web browsers, the widget engines lack the browser UI elements and functionality such as back button or URL input field or history. Widgets occupy a small display footprint with all visible UI elements being associated with the mini-application.

Web widgets (e.g., Google gadgets) are reusable components of Web applications that run in a Web browser and are embedded and executed in a third-party Web page (e.g., social network site, blog, auction site). They are neither packaged nor installed on a client. They enable developers to share their Web applications and to make their Web sites accessible to audiences through third-party Web pages.

Device widgets, of which mobile widgets are one form, are downloaded and installed on a user's PC or mobile device. In recent years, device widgets have proliferated since Apple re-popularized them in Dashboard widgets [12]. Unlike Web widgets, device widgets can access device data and resource. Users can personalize the application. Yahoo! Widgets (previously known as Konfabulator), Apple's Dashboard widgets, and Google Gadgets (running on Google Desktop) are examples of desktop widgets [5]. Widsets, Opera Widgets, and S60 Web Runtime (WRT) widgets are examples of mobile Web widgets [10][17]. The first is built with java and run on java-enabled phones while the latter two are built with standard-based Web technologies running in a widget engine. In the case of the S60 WRT widgets, they are integrated with the mobile phone operating system. As such, they are installed, accessed, and managed like other S60 applications. Like other S60 applications, they have an icon and look and feel no different from other S60 applications.

Each device widget is a packaged file that contains one or more HTML, CSS, and Javascript, image and/or sound files. Each package contains a manifest file with metadata about the widget as well as an image file for the widget's icon. A widget package is created using a standard packager such as ZIP. A media type, one for each vendor, distinguishes the different vendor widget. If the appropriate widget engine is installed on the user's device, and the widget engine has correctly registered a media type and/or file extension, the Web browser should automatically associate the downloaded resource with the widget engine.

3. SEVERAL WIDGETS

We present four different widgets that we have developed. They arose from different needs for using mobile Web applications. They include a mobile meeting service, a remote content access service, mobile information access centered on scheduled events, and knowledge acquisition tool based on collective intelligence. We describe each in turn focusing on mobile work needs that inspired the service and how the widget versions arose.

3.1 Mobile Meetings

Nokia Easy Meet is a Web-based, real-time collaboration service that allows mobile users to participate in remote meetings using their mobile phones. It provides simultaneous voice and data sharing. Mobile users can view images, slides, chat, meeting minutes, and chalk marks on slides. A participant list provides an awareness mechanism and shows all the people participating in the meeting along with their contact information. Voice communication with data sharing is supported over a 3G network or 2.5G with wi-fi. By clicking the 'voice conference' option, participants can initiate a voice call to an audio conference bridge passing along the 'audio conference' id and pin without any further key entries from the user. Nokia Easy Meet is accessible

from any mobile device with a standards-compliant Web browser (S60, Opera, Firefox, IE7, Safari).

Meeting notifications with the meeting url are sent via SMS as well as email. The SMS notification is particularly useful when users want to join a meeting from the mobile phone. When clicked, the mobile Web browser is launched and a connection to meeting server is made. Meeting hosts can forward this url via SMS or email to other people at anytime. While registered users can create, edit, or host meetings, guest users can be invited on a per meeting basis to participate. Both invitation forwarding and guest access were designed to lower barriers to participation and to foster technology support of ad-hoc and planned meetings.

Meetings are not only restricted to those that are scheduled for a certain time and time period, they may include permanent meetings that are always active. Such meetings permit users to suspend and resume without the overhead of creating new meetings. They can become a place where conversations, shared content, and meeting minutes cumulate over a series of engagements that stretch over time. Studies of mobile work reveal that ad-hoc, serendipitous and informal meetings are common-place [1][2][16][24]. Unlike remote meeting tools that support stationary, desk-based interactions, Nokia Easy Meet enables face-to-face interactions of co-present participants in a variety of mobile settings such as meeting rooms, public spaces, hallways or on trains [1] [19]. This is a new dimension that differentiates a mobile meeting service from desktop-based meeting service.

Finally, information access while being mobile is typically unpredictable and uncertain [19]. Unless mobile users anticipate or opportunistically plan their information needs, there will be mobile occasions when users need to access information not available on their mobile phone but are on their remote computers. This is very likely in meetings when the need to share a report or slide deck arise. Nokia Easy Meet enables mobile phone participants to share files by uploading them to the server from their remote computers. The meeting service is integrated with a remote content access service Ovi Files (http://files.ovi.com). Each computer that a user wishes to access remotely must be registered with Ovi Files and a connector application installed. Thereafter, users may access their content from a mobile phone.

Nokia Easy Meet supports two different screen layouts to accommodate different mobile phone screen resolutions (see Figure 1). The screen layout used is determined by a simple heuristic – mobile browser display width. Widths that are QVGA (320x240 or 240x320) or smaller use the small, tab layout and larger mobile displays use the large, accordion layout. Nokia Easy Meet is accessible from any mobile device with a standards-compliant Web browser.

As part of this mobile Web application service, we also created a S60 WRT widget. The widget provides a subset of the service functionality that enhances user experience of Nokia Easy Meet. First, users can customize the widget with their user id and password. Second, the widget launches and connects to the service and logs the user in at the same time. Third, the widget retrieves and displays the user's meetings (see Figure 2). The meeting list can be refreshed by simply clicking the 'Refresh' short-cut. Thus, with about 2 clicks, the user can join a meeting without having to key in urls, user id, or password, remember meeting times, etc. All of the widget's functionality exists as part of the Web application accessed through a mobile Web browser but has been repackaged into a widget that can be installed on user's mobile phone and customized with the user's information.

3.2 Remote Content Access

While we initially integrated the remote content access functionality with Nokia Easy Meet, its functionality is useful outside of the meeting context. In fact, Ovi Files provides an S60 widget for mobile phone use. We have repackaged our remote content access functionality as a widget also (see Figure 3). There were two reasons for this. First, our widget supports a different UI from the one for Ovi Files but one consistent and familiar to users of Nokia Easy Meet. This familiarity should facilitate use outside of meeting service for these users.

Second, and more importantly, we are developing a context-aware model of content access that factors in time, location, and file-type. By repackaging the remote content access functionality as its own widget, a user-based context model of general access can be built through everyday use. Then, with more data on patterns of access, we plan to provide an alternate UI that suggests candidates for the remote content that the user wants to access based on the user's context (i.e., time and location). The non-predictive UI allows the user to traverse the file system until the file is located. In the case of deep file systems or the case when users are not sure

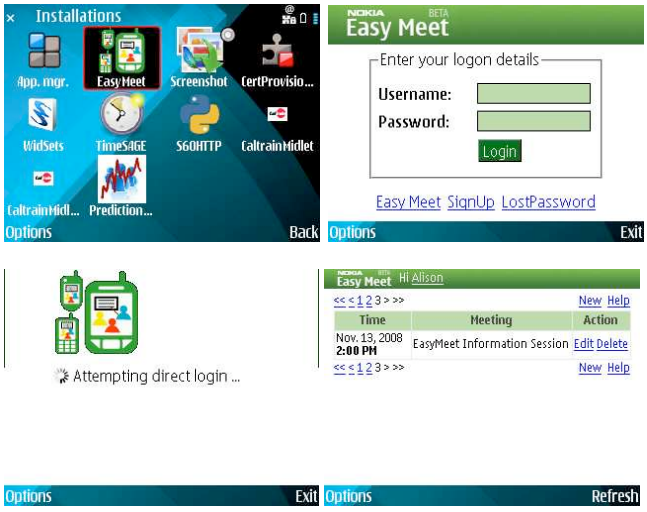

Figure 2: Nokia Easy Meet widget.

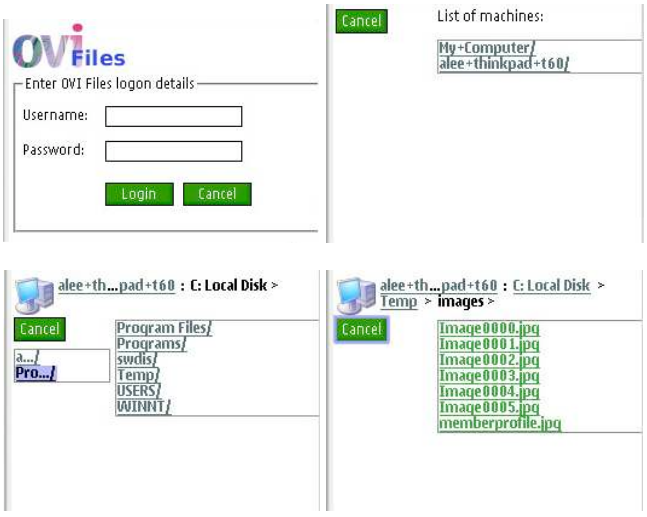

Figure 3: Remote content access – traversing files system.

of where the file is located, this can result in several user clicks to locate the file. An effective context-aware variant would reduce time and effort.

From the perspective of a widget, this remote content service has value when integrated with mobile meetings as well as standalone. In the latter case, rather than relying on human proxy or deferring until one returns to the machine, this service can provide access readily [16][19]. When made context-aware through mobile meetings or in terms of time and location, users can gain just-in-place, in-the-hand access.

3.3 Mobile Information Access

We recently conducted a number of informal interviews of mobile users in our organization to discover people's practices and activities prior to and en-route to meetings [13]. The study was exploratory in nature and was intended to provide insights for mobile services that support people's mobile information access needs. We obtained several interesting insights.

First, all the interviewees engaged in varying amounts of mobilization work with preparatory activities in anticipation of going mobile [2][3][16][18][23][24]. Second, all the interviewees made extensive use of their calendars not only for scheduling appointments, meetings, and events. They used these time containers to corral information that was peripherally as well as centrally relevant to the event such as email, phone numbers, URLs, documents, etc. That is, users had co-opted or overloaded their calendar with information scraps that they might need at the meeting or en-route to the meeting [3]. Many of them also relied on printing this information to have the information readily accessible when the need arise. Third, in probing for why they use paper, users indicated that they were not confident or thought the time, effort, or lack of device functionality to support "just-in-place" information access to the calendar data. Finally, in brainstorming services that could support their mobile information access needs, users described technology use cases that would make the calendar data actionable. That is, rather than simply being apprised of the calendar content such as who they were meeting, what the audio conference code and password information, or meeting address, users wanted to call the person they were meeting, dial the audio conference bridge for the meeting, or get directions to the meeting location.

This study led us to explore the development of a mobile Web calendar mashup that focuses on giving the mobile worker the flexibility and ability to use calendar information to perform a variety of activities that meet their mobile work needs. This service leverages a user's existing calendar and their practice of corralling information into calendars. Users can access the service from their mobile Web browser or an S60 widget. The service presents users with details of their upcoming meeting along with value-added data and functionality in support of their activities while en-route to their meetings. Users can also view upcoming

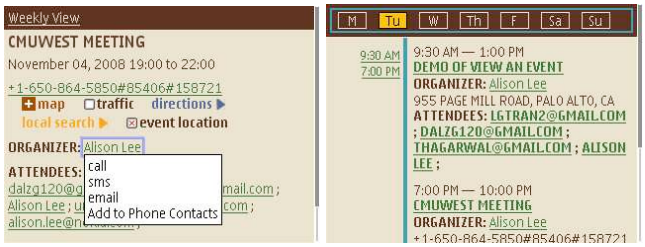

Figure 4: Individual and 7-day view in TimeSAGE.

meetings so that they can take preparatory actions in anticipation of mobile information access needs or activities. As a mobile mashup, a calendar event integrates a temporal context where a variety of information is already encapsulated.

3.4 In-Between Work – Collective Intelligence

The nature of work undertaken while being mobile is highly varied [19][24]. In particular, during business trips, work cannot be characterized as pertaining only to the specific purpose of the trip. Studies show that there are large amounts of time spent outside of a trip's scheduled activities that is in-between tasks and meetings. One of our interviewee's in our study described this time as "holding tank." The time is also referred to as "dead" time, "travel" time, "spare" time, or "wasted time" [19][24].

Understanding the exact form that "dead time" takes for mobile workers is important for understanding and supporting the different levels of information access required or possible in the particular situations and places where they arise and the organization and execution of work. Perry et al. [19] found that activities were rarely organized according to priority or urgency, but according to the context in which dead time occurred and the technological resources available. Hence, work was fit in as and when possible. They found that dead time was rarely used as efficiently as it could be. The challenge for designing devices and services is to allow mobile workers to make full use of dead time, taking into account the diversity of environments in which it occurs and the paucity of available resources. These studies reveal that such services should be lightweight and highly flexible, rather than highly specific, integrated or complex systems or single-use devices. There are numerous studies that reveal people's appropriation and co-opting of technologies including our own interview study.

Pursuant to this, we explored services that could be lightweight and suitable for "dead time" during mobility situations. One area in enterprise work that is critical to organizations is tapping knowledge diffused in an organization. A key element of the service was to tap this without great deal of cognitive or physical effort and without need for huge expenditure of time [23]. Such a service should also entice individuals to participate and to contribute their knowledge. We developed a game-like service that would tap individual's foresight about outcomes of future events and have them stake their opinions through wagers.

We created a widget that enabled users to select particular game and conduct trades in the particular game. Then, they would buy or sell contracts on outcomes of future events. Doing this in the form of a mobile widget seemed appropriate as people could engage in it whenever and wherever they had a little bit of time. Furthermore, if people become privy to information that they believe would affect the outcome of particular future events, then acting on this knowledge as quickly and as timely a manner would be important. The mobile phone seems both convenient and easy as it is always accessible and always with the user.

4. Issues

Mobile Web widgets and widget engines are available on mobile devices. Aside from the S60 WRT engine from Nokia, Opera recently began support of them in Version 9.5 [17]. However, there remain incompatibilities across widget engines [5]. There are now efforts within W3C to standardize widget [4]. Without such an effort, incompatibilities, such as a different media type for each vendor, will limit inter-operability of widgets across

different widget engines. This will cause confusion for end-users with mobile phones and widget engines from different vendors.

The focus of this section is not to rehash the incompatibilities' discussion as there are efforts afoot to deal with them. Instead, we raise the discussion up a level to mobile applications and the engineering of widgets to support mobile work. The discussion is based on a number of observations arising from our experiences and requirements with development of a variety of smartphone widgets for mobile work. This discussion revolves around three main topics: mobile user experience, software engineering practices, and systems engineering issues.

4.1 Personalization and Customization

One of the main advantages of a widget-based implementation of a service compared to a browser-based implementation is personalization of the user interface (UI), user experience and situational context. User preferences may often be due to personal tastes. As well, different mobile devices may have different screen sizes, screen resolutions, aspect ratios and input capabilities. When accessed with a mobile device versus a laptop browser, an effective user experience can be more than just personal preferences. It would depend on the capabilities of the device used by the user. The Nokia Easy Meet application supports two different screen layouts depending on screen size; a choice that can be easily determined based on a simple heuristic. However, other aspects of the device may not be as easily determined (i.e., presence of qwerty keyboard, touch interface).

A widget, with its persistent storage for preferences, is a useful feature that allows the user to set and store the selected UI options for the particular device. Thus, when accessed with non-qwerty, QVGA mobile phone, a user may select shortcut key options or handwriting to ease input interactions and opt for spoken output rather than having all information crammed into a small display. However, when accessing this same service from their laptop, the settings for the application may be different since keyboard and mouse is available.

In all the widgets described in the previous section, a common function included in the widgets is login to the service. User ids and passwords can be persisted using user preference storage mechanism of widget. This is done when the user configures the widget with their credentials. After that, users no longer need to key in this information as they would in the mobile browser counterpart. As well, negotiation of login can be automated by the widget on startup for protected services that users access.

Use of preference storage to keep user credentials has the benefit of reducing user interaction related to authentication to a minimum; when personalizing the widget for the first time or whenever they change their password. Having to key in their credentials every time can be inconvenient (e.g., mobile phone lacking qwerty keyboard) and disruptive to their workflow. This is analogous to users who have opted to use an "always on" device like a mobile phone to look up contact information rather than waiting to "bootup" a laptop to access the information. In fact, this was one of the user experience consideration that spurred the development the Nokia Easy Meet widget.

4.2 Proxying Authentication

Storing user credentials can have security benefits to the user akin to the security benefits of smart cards or credit cards. That is, the vehicle (in this case the widget) presents the credentials as part of negotiating authentication with service. This approach is also beneficial when accessing services that act as a proxy to the actual service containing protected resources or providing protected services. Both the remote content access widget and calendar widget are examples of using the services we developed as a proxy to actual services; Ovi Files and Google Calendar respectively. For example, Nokia Easy Meet never retains or persists the user's Ovi credentials but only facilitates the authentication with Ovi. Through a secure https connection the remote content widget passes along the user's id and password. In turn, the Nokia Easy Meet server passes this on to Ovi files using Single SignOn (SSO) APIs to authenticate the user to Ovi. Since this functionality was developed in support of the mobile meeting tool, reusing the same proxy service in developing our context-aware file service was handy.

Authentication with some services that use SSO can involve a complex series of redirects. Letting a trusted proxy service handle this rather than the widget itself simplifies the widget functionality. More importantly, some Single SignOn services require a large number of redirects that can time out over slow connections or worse yet, a widget engine may abandon the process thinking that it is trapped in infinite redirects. Using a trusted proxy with bandwidth and resources to deal with this is important for robustness. Finally, as our research goal is to rapidly experiment and prototype new services, use of a trusted proxy service means that authentication functionality can be solved once and used by many experimental services.

4.3 User Input and Output

AJAX applications are highly interactive Web applications that exploit a set of technologies that enable exchanging small amounts of data with the server behind the scenes [20]. As a result, Web pages do not need to be reloaded when this happens. The increased responsiveness and interactivity has enabled developers to create highly graphical and dynamic UIs. The resulting Web applications invariably require a mouse or 2-D pointing device to work smoothly. Unfortunately, a rich AJAX application can be cumbersome to use on a mobile device that lacks a touch interface or adequate 2-D pointing device.

On many mobile devices, a 5-way joystick (see Figure 5) is typically used for intra-page navigation and interaction. On S60 Web browsers, the Mini-Map and virtual pointer [11] are enhancements that facilitate Web browsing using the 5-way joystick. The virtual pointer technique combines mouse pointing with focus navigation to select focusable elements while pressing the 5-way joystick to navigate to the element. This allows focusable elements to be selected with minimal key presses. However, this still relies on 2-D pointing actions to move towards the focusable elements.

More importantly, in designing screen layout, designers take care in different segments of the display for application functionality. For example, sidebars contain links to content that are peripheral

Figure 5: 5-way joystick in the center of keypad.

to the content of a Web page whilst the main content area contain content central to the page. However, the same care is not extended to developing input operations for moving through such content areas in a logical way. By developing input mechanisms to move through such regions in a facile way can greatly improve input interactions with such content from a mobile device.

One input navigation technique for mobile device interaction would mimic keyboard navigation using the 5-way joystick. The ideas underlying AxsJAX, a framework for making AJAX applications accessible, are also useful for mobile Web applications where mobile device and context can challenge even able-bodied users [7]. In particular, by developing content navigation rules for AxsNav [21], a widget designer identifies different trails through the various semantic regions of a widget's UI (e.g., sidebar, main content). The content navigation rules include shortcut keys for traversing the links in a trail. On the mobile phone, the shortcut keys can be assigned to the buttons of the 5-way joystick. Such an approach would be more effective then tab navigation executed without knowledge of the semantic regions of an application.

The AxsJAX approach leverages reflection (i.e., discovering information about interaction widgets) capability provided by ARIA properties role and state and notification (i.e., detecting relevant changes) capability provided by ARIA live regions [7] [8]. The S60 Web browser and widget engine do not presently support ARIA although Opera 9.5 does. The lack of ARIA does limit the extent to which mobile widgets and applications to develop multimodal interaction capabilities that address constraints of mobile devices. While AxsNav is made easier with support for XPath in the browser, it can be supported without using Javascript implementation of XPath for browsers and widget engines that do not support it natively.

Many smartphones are equipped with other forms of input and output capabilities. These include touch input, voice input, voice capture, video playback, and video capture. As well, it is also common-place to find sensors and haptics technologies. Such capabilities unleash the breadth and alternatives to traditional input and output modalities capabilities that mobile Web applications and widgets can leverage. As an example, haptics can be exploited for input and output that facilitate eyes-free interactions.

4.4 Data Plans, Bit Rate, Power Utilization

The Nokia Easy Meet service is a real-time collaboration tool developed with AJAX techniques. When in a meeting, data and voice charges can mount. Data relating to shared content, chat, and other awareness information is provided to keep the mobile meeting participant apprised of developments. This is facilitated by asynchronous XML http polling requests. We took care in our design to control the amount of data exchanged, the transfer speed, and the frequency of the requests and updates. All three impact cost on user's data plan, the responsiveness of the application, and battery drain. Hence, one cannot directly apply techniques used to create desktop-based AJAX application to mobile Web applications and widgets. For example, excessive or unnecessary polling can have impact on costs, user experience, and use of the smartphone. Our approach to address this was to create an AjaxBroker to mediate and aggregate XHR requests [14].

Early in our development, we found that a one-hour conference could drain the battery because of over-polling or of spikes in power when setting up network connections when under-polling. As a consequence, when we developed the widget, we elected not to refresh their conference display periodically (see Figure 2) but instead provide a manual Refresh button. Users can launch the widget and leave it in the background. When they desire an update, the widget can be brought into foreground to be interacted with. Transitioning to more extensive service functionality is facilitated by the widget. That is, the widget can check for updates or launch the mobile Web browser to enable more detailed work such as a mobile meeting.

4.5 Bridging Work and Technology

Widgets provide specialized functionality. In the development of our services, we created browser-based Web applications. Widget functionality emerged to grease the user's interaction with the service. In the case of Nokia Easy Meet, it made accessing a mobile meeting easier by connecting to the service's URL, logging in for the user and enabling users to get periodic awareness of updates to their meeting. The remote access content widget emerged as a realization of enabling general and everyday access to one's remote content outside of mobile-meeting situation. . In the case of calendar application, the widget could be placed in the background much like the Nokia Easy Meet widget to support users while they are mobile by making the event information readily accessible (i.e., just-in-place mobile information access). The game widget arose from its natural affordance to support dead-time activity

4.6 Mobile Phone Platform Access

Both the remote content access service and the calendar service can leverage a user's current location as well as other resources on the mobile phone. In pilots of the Nokia Easy Meet service, we have received user requests to access their mobile phone contact list for people to invite when they create meetings. As well, users also indicated that once meetings are created, our service should update their calendar with the meeting. These examples show the desirability of being able to access platform resources and data to support the mobile Web services. This is a capability that is not yet supported on mobile Web browsers or widget engines.

In the implementation of the calendar and remote content services, platform data and resources are accessed via a personal S60 HTTP server that we developed. The server is a simple, HTTP server that responds to HTTP 1.1 Get/Post requests from applications running on the mobile phone. It is implemented in PyS60 [ref]. Web service APIs are installed in the script directory of the S60 HTTP server. These APIs are PyS60 or native scripts that access and interact with platform resources (e.g., camera, GPS) and data (e.g., contacts). The APIs return a JSON object [25].

Unlike the mobile Web server [26] the simplicity of the personal S60 server means that it neither supports external network requests nor does it support multi-threading for multiple, simultaneous requests. Keeping it simple and for personal access rather than being externally accessible has privacy, resource utilization, and security benefits. This approach was undertaken as platform resources via widgets are as yet unavailable in existing S60 devices. More importantly, the browser-based version of the application can use the same approach to access platform resources just like it accesses 3rd party APIs. Platform access for mobile browsers is not imminently available.

4.7 Enabling Rapid Prototyping

In all implementations of mobile Web applications and its widgets, we have followed a number of software engineering practices. First, all services maintained a separation between data, presentation, and logic. Django, a high-level python Web framework that supports object-oriented engineering principles, was used in the development of three of the four services. It provides a number of important building blocks such as object-relational mapper to describe database layout in python code, support for url design that is important in crafting web service APIs, and a template language to separate design, content and code. What is important to note is not specifically that Django was used but the selection of a framework that supports good object-oriented approach to service creation. The game service was developed using another framework that had similar building blocks.

To illustrate why selection of such a framework is important is to look at the contribution of a template system in the development of the service. We were able to create the UI for the browser and widget versions without replicating the work to create their views for each platform. As we iterated over the functionality of the service and re-factored selected functionality from the browser Web application for use as a widget, the template system enabled this. Finally, testing of widget was greatly facilitated by testing in a browser before final testing on the mobile devices. A number of issues and bugs were identified and addressed quickly. Debugging of widget code on the mobile device is not always easy for lack of good tools. Finally, by emulating the widget environment in the browser, it was easy to iterate on designs of the widget.

5. Conclusion

This position paper used studies of mobile work to show how it differs from desk-work. As well, we presented a number of mobile Web applications and widgets that we developed to illustrate our explorations of smartphone-based services to support mobile work. Widgets were developed to support mobile Web browser application (i.e., Easy Meet), were pulled out of mobile Web application for general use that could become contextualized (i.e., remote content access), were developed to mash up information and services related to calendar events, and were used as lightweight mechanisms to tap into collective intelligence.

As smartphones begin to take hold as more than communication devices, the role of widgets and mobile Web applications will take on an importance as device of mobile workers. We found that the use of mobile Web applications and mobile widgets to be extremely promising for exploring, prototyping, and piloting mobile services. In addition, mobile services in support of mobile work will need to integrate data and resource from the cloud as well as from the mobile phone platform itself. While incompatibilities and pervasiveness of the capabilities exist on the mobile platform, we have found that the smartphone platform and the mobile Web widgets to be promising technologies for exploring mobile work services. We have presented some of our experiences and issues with using these technologies. They represent challenges that need to be addressed before the power of smartphones and mobile mashups can be realized as integral tools for mobile workers.

6. REFERENCES

[1] Bellotti, V. and Bly, S. Walking Away from the Desktop Computer: Distributed Collaboration and Mobility in a Product Design Team. In 1996 Proceedings of the Conference on Computer-Supported Cooperative Work. ACM (1996), 209-218

[2] Bergqvist, J., Dahlberg, P., Ljungberg, F., and Kristoffersen, S. Moving Out of the Meeting Room: Exploring Support for Mobile Meetings. In S. Bodker et al. (eds.), *Proceeding of 6th ECSCW Conference*, Kluwer (1999), 81-98.

[3] Bernstein, M. van Kleek, M., Karger, D., and Schraefel, M.C. Information Scraps: How and Why Information Eludes our Personal Information Management Tools. ACM Transactions on Information Systems, 26(4), ACM (2008), 24-46.

[4] Bersvendsen, A. and Caceres, M. Widgets 1.0: APIs and Events. http://www.w3.org/TR/widgets-apis/, W3C (2009).

[5] Caceres, M. Widgets 1.0: The Widget Landscape (Q1 2008). http://www.w3.org/TR/widgets-land/. W3C Working Draft 14, (2008).

[6] Caceres, M. Standardising Widgets. http://datadriven.com.au/thesis/confirmation/confirmation.pdf (2007).

[7] Chen, C. and Raman, T.V. AxsJAX: A Talking Translation Bot Using Google IM: Bringing Web-2.0 Applications to Life. In Proceeding of the 2008 International Cross-Disciplinary Conference on Web Accessibility (W4A), ACM (2008), 54-56.

[8] Craig, J., Cooper, M., Pappas, L., Schwerdtfeger, R., Seeman, L. Accessible Rich Internet Applications (WAI-ARIA) 1.0. W3C (2009). http://www.w3.org/WAI/PF/aria/

[9] Cui, Y. and Roto, V. How People Use the web on Mobile Devices. In Proceeding of the International World Wide Web Conference WWW2008, ACM (2008), 905-914.

[10] Forum Nokia. Widgets. http://www.forum.nokia.com/main/resources/technologies/browsing/widgets.html.

[11] Grassel, G, Geisler, R., Vartiainen, E., Chauhan, D., Popescu, A. 2006. The Nokia Open Source Web Browser for S60. MobEA III Workshop held in conjunction with WWW2006 conference. http://www.research.att.com/~rjana/MobEA-IV/PAPERS/MobEA_IV-Paper_3.pdf

[12] Kaar, C. An Introduction to Widgets with Particular Emphasis on Mobile Widgets. University of Applied Sciences, Hagenberg. Technical Report 06/1/0455/009/02 (2007), 6 pgs.

[13] Lee, A., Adalumo, G., Agarwal, T., and Tran, T. Calendars as Mobile Habitats. Draft, (2009), 10 pgs.

[14] Li, D. and Anand, M. MaJaB: Improving Resource Management for Web-Based Applications on Mobile Devices. To appear in Proceedings of MobiSys 2009, ACM (2009),14 pgs.

[15] OAuth Workgroup. OAuth Core 1.0. (2007). http://oauth.net/core/1.0.

[16] O'Hara, K., Perry, M., Sellen, A, & Brown, B. Exploring the Relationship Between Mobile Phone and Document Use During Business Travel. In B. Brown, N. Green, and R.

Harper (eds.), *Wireless World: Social and Interactional Aspects of the Mobile Age*, Springer-Verlag (2001), 180 - 194.

[17] Opera. Opera Widgets. http://widgets.opera.com/.

[18] Oulasvirta, A. and Sumari, L. Mobile Kits and Laptop Trays: Managing Multiple Devices in Mobile Information Work. In Proceeding of Conference on Human Factors in Computing Systems, ACM (2007), 1127-1136.

[19] Perry, M., O'Hara, K., Sellen, A., Brown, B., and Harper, R. Dealing with Mobility: Understanding Access Anytime, Anywhere. ACM Transactions on Computer-Human Interaction, 8(4), ACM (2001), 323-347.

[20] Raman, T.V. Toward 2^W, Beyond Web 2.0. Communications of the ACM. ACM (2009), 52-59.

[21] Raman, T.V. and Chen, C. AxsJAX Content Navigation Rules (CNR) Reference. Google Code (2008). http://google-axsjax.googlecode.com/svn/trunk/docs/cnr.html.

[22] Scheible, J. and Tuulos V. Mobile Google Maps: Rapid Prototyping of Applications on the Mobile Platform. Wiley (2007).

[23] Sohn, T., Li, K.A., Griswold, W.G., and Hollan, J.D. A Diary Study of Mobile Information Needs. In Proceeding of Conference on Human Factors in Computing Systems, ACM (2008), 433-442.

[24] Wiberg, M. RoamWare: An Integrated Architecture for Seamless Interaction In between Mobile Meetings. In Proceedings of GROUP'01, ACM (2001), 288-297.

[25] Wikipedia. JSON. http://en.wikipedia.org/wiki/JSON.

[26] Wikman, J. and Dosa, F. Providing HTTP Access to Web Serbers Running on Mobile Phones, NRC-TR-2006-005. Nokia Research Center (2006). http://research.nokia.com/tr/NRC-TR-2006-005.pdf.