CHAPTER 2

Characteristics of Mobile Applications

Intelligence is quickness to apprehend as distinct from ability, which is capacity to act wisely on the thing apprehended.

—A. N. Whitehead (1861–1947, British mathematician and philosopher) (Encarta 2004, Quotations)

Introduction

Mobile applications are different from their desktop counterparts, and it is worth spending some time and space exploring what makes them special. To be successful in mobile application development, it is important to have both a keen understanding of what mobile devices really are and how they differ from desktop and laptop computers that users also interact with.

This chapter is intended to set the context and get you "thinking mobility." Getting into the habit of thinking from a device perspective is important because it will have an enormous effect on how you go about designing your applications. To build great mobile applications, it is important to understand what key characteristics define great mobile applications. This chapter answers the questions "How does mobile application use differ from desktop use?" and "What are the most important characteristics of good mobile applications?"

Armed with this understanding, it will be possible for you to focus on the engineering challenges that matter most for mobile device software development.

Usage Patterns

Perhaps the most critical difference between mobile device applications and desktop applications is the way in which people use them. Picture

yourself at your computer desktop; what are you doing? If you are like most people, you are doing one of three things: browsing the Web, working on documents (word processing, spreadsheets, pictures, photographs, and so on), or communicating (e-mail, instant messaging, and so forth). These tend to be long-time duration and exploratory activities and the user will periodically switch between these activities midstream. If you are a developer (and the odds are pretty good you are if you are reading this book!), you can add software development to this list, which is absolutely a long-time duration and exploratory activity. If you are a player of computer games, you will know that many of the most interesting games for desktop computers are long-term and exploratory games allowing users to explore and interact with complex virtual worlds.

People who use both desktop applications and mobile device applications tend to use the two in different but complementary ways. While working with a mobile device, the user's activities tend to be short session length interrupt-driven or interrupt-causing interactions. The user is typically either responding to being interrupted or using the device to make an immediate request of some other person or process. Today's typical mobile phone is a great example of this. When you place a phone call or send an SMS text message, you are interrupting someone; when you get a phone call or an SMS message, you are being interrupted. This same usage model carries over to all kinds of mobile device applications. Successful mobile applications must be as natural and intuitive as making or receiving a phone call or SMS message.

Mobile device applications tend also to be more focused on enabling a few specific features very well as opposed to offering the general-purpose exploratory environment that successful desktop applications do. Because mobile devices are often operated using a single hand or by tapping a small screen with a stylus, it is important that users of the device be able to quickly discover and navigate to the information and features they want. The ability to quickly navigate to a small set of key features is an important aspect of a great mobile device experience.

Long-Duration vs. Short-Duration Activities

People using desktop computers tend to do so for long sessions. Working for an hour at a time is not unusual, and several-hour sessions are not atypical. In these sessions people mold and develop ideas over time and tend to do a great deal of iterating and revising. Because of this, startup time is not as important as giving the users the rich features they may need while exploring and working with their information.

Laptop computers tend to share many of the same usage patterns as desktops, but it is possible to see some mobile device characteristics emerging. Speeding up startup time (or wakeup time) and getting users back to the activities they were doing when they last shut down is an increasing priority. However, the usage patterns of laptops are still much more desktop oriented than they are mobile device oriented, and part of this is simply due to the relative physical size of the devices. Users can do an awful lot of things on a mobile phone in the time it would take them to get their laptop out of their bag, open the lid, start up the computer, get to the application they want to work with, and get online if required. This is very different from the "take it out of your pocket and use it right away" model that mobile phones or PDAs offer.

When using mobile devices, the same people's activities tend to be short term, ranging from several seconds to several minutes. Activities such as making a phone call, checking an appointment schedule, or entering or reading instant messages are all done using frequent but short-duration sessions. Even playing games tends to be a short-term activity on mobile devices. People playing games on mobile devices usually play for relatively short periods of time when they are in between activities. The games are intended to allow the user to pass extra time pleasurably while waiting for a train, sitting at an airport, or waiting to meet someone at the mall. Because of this focus on short-session usage and immediacy, mobile devices are usually in an "always on" or "instant on" state, where the time for users to access the device is on par with the time it takes for them to take the device out of their pocket.

It is important to note that although session time tends to be short, the underlying data the user works with is often long term. In the case of a mobile game, users may want to pick up where they left off the last time they played. For productivity applications, this trend is even more pronounced, with a classic example being the address book on a mobile phone (which must always be available). A common task when speaking to someone on the phone or in person is checking a personal appointment calendar and booking a follow-up meeting with the person. The user has a need to be able to quickly access the device, navigate to the person's contact information, create a new schedule item, and then resume the conversation without further delay. The underlying data is long term and durable, but the application usage pattern is short term and immediate.

Exploratory vs. Focused Activities

Desktop and laptop computers offer a very rich platform for exploring information. The large-screen real estate available enables the display of a great

deal of peripheral information that may be of interest to the user. The keyboard and mouse offer a rich way to navigate through that information in a random-access manner. This enables the user to jump rapidly to any of the information visible on the large screen, flip through several layers of windows that currently may be open, quickly enter and revise relatively large amounts of data in a short time, or "drag and drop" to move information around various documents. These are all exploratory mechanisms for working with information.

Browsing the Web on a desktop computer is a great example of this kind of exploratory experience. Common high-bandwidth network access offers the possibility of downloading large amounts of information locally. This allows more information to be downloaded than the user has immediate need for, just in case the user will find it of peripheral interest. The user is commonly presented with a rich downloaded document to explore with many areas of interest to potentially branch off to. During this exploration, users commonly branch out into other documents and often have several browser windows open simultaneously so that they can switch between them to find the information they are looking for. Anyone who has used the Web to find a lowest-cost airfare is familiar with this experience. Multiple browser windows to different travel sites are open, and different arrival and departure dates are priced. Data is often copy/pasted from browsers into e-mail or other documents. Faced with a rich Web page with a great deal of information and many links, the user's next specific action is very difficult to predict. Will the user click a link? Will the user type a new address into the Web browser access bar? Will the user copy/paste some information from a Web page into a document? Will the user start an e-mail or instant message session? The possibilities are almost endless, and because of this the user typically works in an exploratory mode.

Similar exploratory usage patterns exist for rich client applications on personal computers such as e-mail applications, spreadsheets, and content-creation applications such as word processors and paint programs as well as for development tools. Users "task switch" often and in random-access ways to get to the information they need from different sources and integrate it for their own use. A person typing into a word processing document will continually iterate over the text, making improvements to the content and layout, revising the text, and navigating the document in a random-access manner. Users may switch to another application to grab some relevant information and copy it into the document they are working with. Spreadsheets offer an even more exploratory user interface by giving the user a huge tabular space in which to enter, modify, and analyze data; to draw charts; and to ask what-if questions. Development tools are also immensely

exploratory and enable developers to quickly navigate around exploring their code and user interface design, browsing definitions, performing code transformations, and seeking reference materials relevant to the tasks they are trying to accomplish. Debugging code is by definition an act of exploring, experimentation, and analysis.

The best desktop applications enable users to explore information in a very free-form way. Conversely, great mobile applications enable the same users to zero in on specific information and services as quickly and as efficiently as possible with little or no navigation.

The same users deal with local documents differently on mobile devices than they do on desktop computers. It is often useful to be able to view and make small modifications to desktop-created word processing documents and spreadsheets on mobile devices, but these documents are almost never created and authored on devices. It is safe to assume that 90 percent of the content creation for these documents will happen on desktops and that devices will be used primarily for reading these documents.

In addition to applications used to surf the Internet and work with documents, there are also custom applications. Many kinds of custom applications exist on the desktop—line-of-business applications, communications applications, analytical applications, data-entry and tracking applications, and so on. All of these applications have mobile device variants. The same "explore vs. focus" difference discussed previously exists between desktop and mobile applications. To be useful, the mobile applications should offer a focused experience to the same person who may use a more exploratory version of the desktop application. Think of mobile applications as offering a different view on the same data and processes as a desktop application. The data and processes are the same as the desktop, but the view is transformed and focused into a way that offers instant access to the key elements that the user may need while mobile.

A common mistake developers make when trying to bring a desktop application to a device is trying to bring down the whole application verbatim and shoehorn it into a device. This never works satisfactorily because of the usage differences between desktops and mobile devices discussed here.

A second common mistake is to take a desktop application and user interface and start slicing pieces out of it until it runs on the device. Related to this are efforts to port pieces of a larger desktop user interface to a device by breaking the large interface into a series of nested dialogs that can be displayed on the device's screen. Doing this will usually result in an awkward and less-functional desktop application, not a well-honed mobile device application. The fundamental question that needs to be asked is "What is the view of the data and processes that will be most useful for

people using mobile devices?" From the answer to this question, you can start constructing the user interface from the ground up.

It is essential to think about the scenarios that users need to accomplish when mobile and to specifically optimize these tasks so that they work better than they do on the desktop. It is equally important to consider what parts of the application are not critical for mobile usage. These parts of the application should be de-emphasized or eliminated. Your mobile application should serve to behave as a kind of fish-eye lens for the users' needs, allowing a broad view but specifically zeroing on the most important tasks they want to accomplish.

A More In-Depth Look into Mobile Web Browsers

A mobile Web browser is an interesting example of a common rich client application running on a mobile device. Because of this, it is useful to examine how mobile Web browsers differ from their desktop counterparts. This will give some good metaphors to think about how other desktop applications may manifest themselves as mobile device applications and what design changes may be required to make the mobile experience a good one.

Many mobile devices now offer Internet browsers, but these browsers are very different in their usage models from desktop browsers. Although technologically similar (they both render HTML for display), their interaction with users is significantly different. Desktop browsers are designed to access the widest array of Web sites and to download and display complex documents. Mobile Internet browsers are designed to access mobile Web sites and download and distill documents to their essential information. Mobile browsers tend to work in one of two ways: (1) They download content specifically intended for mobile devices that is simpler in layout, has smaller-sized photos and images, and is designed to be read in a smaller window; or (2) they download generic Web-content and attempt to distill it down to its essence and display that to users.

Building a Great Web Experience for a Mobile Device Requires Cooperation Between the Device and Server

It is worth noting that trying to distill generic HTML pages into a good viewable experience for a mobile device is a complex problem. Given the huge variety of layout possible with HTML and the mixing of core information with peripheral content and advertising, it is challenging to pull out what the essential information is and fit that onto a mobile device screen.

Instead of putting all the burden onto the device, a better result can often be achieved by tackling the problem on the server. Often, popular Web sites such as MSNBC or the BBC offer separate mobile device versions of their content. These two sites take different approaches to solving this problem that are worth understanding:

- http://www.msnbc.com/news/MobileChannel/mmc.asp—This Web site automatically adjusts its response to the capabilities of the browser that makes the request. Results differ significantly when accessing the Web site from a desktop browser or a mobile device browser.
- http://news.bbc.co.uk/text_only.stm—This BBC Web site offers a Web view with low-resolution pictures and limited formatting that can be displayed effectively on mobile devices.

The goal of displaying HTML content effectively on a mobile device can be achieved much more effectively if the server is willing to meet the device half way and provide a simplified view of the data. Solving mobile device application challenges by building additional targeted server services is a good idea not just for Web applications but for all applications that access data on servers. If there is work you can do on a server to simplify design challenges faced by mobile devices, it often makes sense to do this.

In addition, although the address bar is prominently displayed and used in desktop browsers, it is often hidden and is much less used on mobile devices. Due to the smaller screen sizes and more limited input mechanisms, the increased time it would take to type a URL into a device and the space that the address bar takes up on the screen, the concept of an address bar is much less useful on a mobile device. Adding to this is the fact that much generic Web content will not render acceptably on mobile devices. All of this greatly lowers the utility of completely random-access navigation to Web addresses.

None of this should leave you with the impression that Web browsing for mobile devices is not useful or interesting; it is both! Many good mobile Web applications offer the user a lot of utility when accessed from devices. With the advent of server-side Web application technologies such as the ASP.NET Mobile Controls, it is now easy for Web developers to build "mobile views" to their existing Web content. Web browsing for mobile applications is a "killer application," but it is a different "killer application" from its desktop relative.

Form Factor

The need to fit comfortably into one's pocket is a key defining characteristic of most mobile devices. This physical constraint is the basis for the mobile device's utility; if it fits in your pocket, it is mobile. Some additional important form-factor considerations are as follows:

- *The ability to be used in crowded and noisy spaces*—This is a reason why speech input is not always a great idea even if it is technically possible. It is also important that a mobile device not be disturbing to those around it (a reason why voice response is not always a great idea). A train full of people arguing with their mobile devices is not a pleasant social environment. Think of having a silent mode or headphones if sound is required for your mobile application.
- *Single- or two-handed operation*—Many devices are intended to be operated by one hand. Most mobile phones meet this criterion. Some are intended to be held in one hand and operated with another. Most PDAs fall into this category. Laptops generally require two hands and a flat surface for efficient usage. Your mobile application should follow whatever paradigm the device imposes on usage; that is, don't build an application that requires two-handed use to get anything useful done if it's going to run on a mobile phone that people typically operate with one hand. This is an important aspect for the usability testing of your application. Another important consideration is the physical environment in which a handheld application is being used. A touch-screen display offers a rich environment for navigating applications, but all too often application designers build and test the user interfaces either running on a desktop-based device emulator or sitting at their desks. This results in user interfaces that are far too small for real-world usage. The real world is full of bumps, jitters, vibrations, and people who want to use their devices while walking down the street or by pressing the screen with their fingers rather than the stylus. Paradoxically, for real-world usage, touch-screen input on small mobile devices often requires larger user interface controls than stationary applications being used on personal computers. It is important to understand whether your application needs to be "one-hand friendly," real-world two-hand "stylus friendly," or "finger friendly" when used with a touch screen while the user is mobile.

- *No power cords or communications cables for long periods of time*—To be effective mobile devices, the devices need to be able to operate untethered for long periods of time. A good rule of thumb is that the devices and the applications running on them should not require being connected to wired power or communication sources more than once a day. How often your application requires the device to be running at full power and how often your application needs to be connected to online data are important design considerations.

These form-factor considerations should strongly influence your design. Engineering creativity is required to solve form-factor problems as well as to adapt software design to different form factors.

T9, a Great Example of a Smart Engineering Solution to a Mobile Device Constraint

T9 is a way to allow for rapid one-handed text entry on mobile phones that have the standard 12-key phone number pads. Prior to T9, users entering sentences of text into a mobile phone needed to tap in each letter by laboriously hitting the 1 through 9 keys up to 4 times to get the correct letter. Table 2.1 shows the number of key presses necessary with and without T9 to type the simple text "text message."

Table 2.1 Mobile Phone Key Presses Necessary to Type a "Text Message"

Desired Letter	Key Taps Before T9	Key Taps with T9
T	8, = t	8, = t
E	3,3, = d, e	3, = e
X	9,9, = w, x	9, = x
T	8,= t	8,= t
<space>	1,= space	1,= space
M	6,= m	6,= m
E	3,= d, e	3,= e
S	7,7,7,7= p, q, r, s	7,= s
S	7,7,7,7= p, q, r, s	7,= s
A	2,= a	2,= a

Table 2.1 Mobile Phone Key Presses Necessary to Type a "Text Message"

Desired Letter	Key Taps Before T9	Key Taps with T9
G	4,= g	4,= g
E	3,3 = d, e	3 = e
Total Taps:	21	12

Twelve taps compared to 21 taps represents a savings of more than 40 percent in key presses. In practice, the T9 time savings is even greater because you avoid confirmation delays incurred when you want to enter two sequential letters represented by the same numeric key.

For example, both *r* and *s* are represented by the number 7 on the keypad. If you want to spell *cars*, you must wait for a second after you enter *r* to allow the input software to realize you are done with that letter and move the insert point to the next letter to be entered. Anyone who has ever tried both ways for a few days will never go back to the pre-T9 way of entering short text.

How does it work? Statistics! When you enter the keys 8, 3, 9, the software looks in its dictionary and determines that the only likely words you could be typing are either *vex* or *text*, so these are the two options it gives you. When you enter the final key 8, the software finds that only one stored word in its dictionary meets this key combination, *text*, and that word is chosen for you. By maintaining a dictionary of words and key combinations in your local language, the software is able to greatly increase your efficiency in writing short messages. If you need to go outside of the words available in the dictionary, you can enter the unknown words via the old input mechanism.

Is this suitable for writing the novel *War and Peace* on your mobile phone? No, of course not; but it is perfectly suitable for typing the sentence "Just finished reading *War and Peace*—long book!" and sending it to your friend.

T9 is a great example of "thinking mobile" and solving a problem specific to entering typical information onto mobile devices. We can learn from this creative idea. The key message is "Don't solve the generic problem; solve the specific problem your users face and optimize, optimize, optimize."

Reliability Requirements

With regard to reliability requirements, mobile devices paradoxically resemble servers more than they do desktops. The reasons for this are as follows:

- *Much like servers, mobile devices and their applications are often left running 24 hours a day, 7 days a week*. Cell phones and PDAs are often left running all the time or have standby modes that ensure that when they start up they come up in a state that closely resembles the one they were last used in. Although desktop computers are also increasingly left on all the time, users still reboot them, log on and off occasionally, and start and shut down applications fairly frequently; this causes improperly held system resources to periodically get flushed. In contrast, because the applications on mobile devices are meant to be "instant access," the applications are often left running in a background state so that they do not need to incur startup delays and users can pick up where they left off with the applications. For these reasons, devices resemble servers in that they need to sit ready to provide instant services for their clients.
- *Much like servers, mobile device applications have to deal effectively with unexpected failures*. Mobile devices operate in a demanding environment. Communications failures occur often and midstream. Users think nothing of popping the battery out of the back of a mobile phone midstream if they are running low on power or think the device is behaving in an awkward way. The operating system itself may shut down a background mobile application if it is running low on resources. Worse still, devices get lost, are stolen, are dropped into puddles, and suffer all means of cruel and unusual punishment at the hands of their users. Even with this, users are devastated when their mobile phone disappears or stops working and they realize that they had important and hard-to-replace information on it. For all these reasons, mobile applications, like mission-critical servers, need to make sure that the important data and state they are managing is held in longer-term storage that can survive the application unexpectedly vanishing or failing. Mobile application developers need to think like mission-critical server application developers and ensure that data important to users is stored safely and in a form that can be recovered if data corruption occurs due to sudden failure. Developers should also consider enabling appropriate backup measures to allow automatic off-device archives that enable users to recover from catastrophic loss of important

information; users rarely back up their own important data even though the need to do so should be obvious.

- *Much like servers, mobile device operating systems and applications often do not use memory paging files*. Your desktop computer probably has a large paging file set up by default that enables it to swap out memory not being used to a file on disk. This file is called a memory paging file or a swap file. If new memory is requested by an application because it is starting up or because it has additional memory needs and the system is running low on physical memory, the operating system will swap out pages of memory that have not recently been used to a file on disk. If the paged-out memory is later accessed, it is swapped back into memory and other pages are swapped out as needed. In this way your desktop computer can function as if it has a much larger amount of RAM than it actually does. This is intended to enable users to run many applications simultaneously while keeping the one in the foreground as responsive as possible. It also allows leaked memory to be relatively harmlessly swapped out to disk on the grounds that the application that is leaking memory will probably be shut down before its leak becomes so large that it consumes all of the page file memory. Servers tend not to use this strategy because they are designed for maximum throughput. A server wants to keep everything in physical memory where it can be accessed quickly. Devices tend not to use page files because they do not have huge disk drives onto which they can quickly swap in and out pages of memory. Having this capacity on a device would be prohibitive from expense, size, speed, and power-consumption perspectives. You may object that "in theory memory could be swapped to some kind of flash memory on a device"; however, this is not practical because writing to flash is not quick and is not intended to be done over and over again in rapid succession.
- *Many mobile devices serve other critical purposes while running foreground applications*. If a mobile phone ceases to work as a mobile phone because your application crashes, drastically slows down in responsiveness, blocks the user interface, or otherwise misbehaves, the end user will not be very happy. Most mobile operating systems have levels of protection to guard the critical functions, but your application is certainly capable of making the device less useful for its other functions if it misbehaves. In server terms, this would be known as a "denial-of-service" problem. Like servers, many devices need to manage a series of critical services that must be available for users at all times.

For these reasons, it is important that your mobile device application be able to run reliably and efficiently over long periods of time.

Important Characteristics of Mobile Applications

This chapter has spent a fair amount of time and space comparing and contrasting mobile devices and their applications with their desktop and server counterparts. It is now appropriate to specifically list the things that define great mobile applications.

Startup Time

Quick startup time is an important characteristic of mobile applications. Because users tend to use mobile devices frequently and for short durations, the ability to quickly start up a mobile application is imperative. It may be undesirable to sit at a desk staring at a word processor, encyclopedia, or development tool splash screen for 6 seconds as it starts up, but this is a minor annoyance compared to the overall time you will spend using the application. For a mobile device application that the user wants to use for 20 seconds to check or update some small piece of information, 6 seconds is an outrageously long time to wait. A good rule of thumb is that the user's session time with an application must be much longer than any startup time he or she endures. Desktop applications have long session times, so users are willing to suffer larger startup times. Mobile applications have short session times, so proportionately shorter startup times are required. Startup time is important for desktop applications, but it is critical for mobile devices because people use them intermittently and for short durations.

Responsiveness

A mobile device looks like a small mechanical tool that fits in your pocket, and people expect it to behave that way. When people tap it, push a button, or do anything physical to it, they expect a physical response. If they do not get an immediate response, they will become impatient and try again. This can cause problems when the second tap, click, or poke is processed by your application or another application that errantly gets the second click. Therefore, it is of utmost importance that users receive some type of acknowledgment immediately upon performing an action on a device.

The best kind of acknowledgment is the completion of the requested action; nothing tops that. The second best response is an acknowledgment that the request has been received and is being processed in the background, leaving the application ready to take another request. The third best response is to acknowledge the request and show something like a wait cursor to let users know their request is being worked on; the application is not responsive, but users receive some indication that work has been initiated and is being done on their behalf. The worst response is to do nothing and leave users wondering whether their action was registered. As with some other characteristics we will explore, this requirement is not unique to mobile devices, but it does have a special relevance because of the way people work with devices and expect them to work for them.

Focused Purpose

Focused purpose is another characteristic of a successful mobile application. The application must have a clearly defined set of things it does very well; it must do them with a minimum number of clicks, taps, or other user gestures; and it must do them quickly. The importance to mobile devices of focused purpose is often exemplified by having special buttons on the devices that are assigned to specific tasks (for example, a button to jump right to the stored contacts database, a button to view schedules, or a button to read in a bar code number and send it to a specific application).

Your mobile application should strive to identify which common tasks it is going to make as easy as absolutely possible. This is true for high-level application features as well as low-level tasks that are commonly performed by users working with your application. If your mobile application needs to enable users to enter dates quickly, for example, make sure that choosing dates is as easy (and quick) as possible and measure the effectiveness of this process. If your mobile application needs to enable users to choose locations on regional street maps to give them directions on how to get somewhere, make sure the user's experience in doing this is as quick as possible.

A common error in building mobile applications it to write as little code as possible with the intention of keeping the application as small as possible. This is a noble goal but should not be done at the expense of user productivity. Spend the time to write the extra code to make sure your application has a focused purpose and enables users to achieve their goals as quickly as possible.

Customized Interactions with Off-Device Information Sources

It is important to understand that building a great mobile application is not just a matter of getting the code that runs on the device right. Thought must also be given to the off-device software that the mobile application interacts with. Information sources that expose services to mobile devices should be given proper consideration in the application's design to ensure that they are returning information in a way that is appropriate for mobile devices. A good example of this is e-mail services for mobile devices. Rich e-mail applications require server- and client-side software. The client accesses the server to get information about incoming messages and then downloads the relevant content to the local devices. Because mobile devices tend to utilize network connections that are intermittent in availability, lower bandwidth, and often more expensive than those personal computers use, the e-mail server services for mobile devices should be tailored to meet these constraints. This typically means providing server facilities to limit the size of content being downloaded or specifying filters to identify the information that is truly useful to a mobile user. A server-side service originally designed for desktop access may need to be extended to effectively support mobile scenarios. In addition, configuration mechanisms must be designed to run on servers, desktops, or the mobile devices themselves that enable users to specify their information needs and tune the information filters to suit their requirements. Design for mobile applications often extends well beyond the physical boundaries of the devices themselves.

Consistency of Experience

Because mobile devices are compact and self-contained, users naturally view the whole device as a single unified experience. Each mobile device has its own gestalt. Typically, successful mobile applications do not appear so much as discrete applications but rather as natural feature extensions of the mobile device's experience. For this reason, following style guidelines for each specific device you are building mobile applications for is important. How a user starts, stops, navigates through the features of a mobile device, and answers common prompts are very specific and learned behaviors unique to the target device. Users adapt unconsciously to a mobile device's user interface metaphors, and deviations from these patterns become very uncomfortable. Consistency of experience is important for desktop applications as well, but because desktop applications offer such rich experiences there are often multiple ways to accomplish a given task (such as keystrokes,

mouse clicks, menus, and toolbars). For mobile applications, there is often only one way to accomplish a given task, and the user gets implicitly trained on how to do this. It is far better to have four different versions of your device application, each tuned to the user interface metaphors of specific devices, than it is to have one general application that does not integrate well into any of the target devices.

Computer Architecture Differences

In terms of architecture, desktop and laptop computers are like large houses in the countryside. Mobile devices are like apartments and condominiums in the city. Both serve specific needs and pose specific restrictions on how they can most effectively be used.

Houses tend to be relatively large and have lots of storage space. Some storage space is close at hand, whereas some other space is harder to get to. In a house, the stuff you do not use very often is stored in the attic or the basement, and now and then you have to go look for it when you need it.

Apartments and condominiums, on the other hand, tend to have very little storage space. Things you need close at hand are kept there. Less frequently used things are discarded. Rather than own things you use infrequently, it is often easier just to rent them as needed.

The same holds true for mobile devices. Both the available memory as well as the long-term storage is geared toward keeping around what you use often. Less frequently used data should be pushed onto servers to be accessed when needed.

On a desktop computer, RAM and long-term storage space are separated. On devices, your RAM is often used both as working application RAM as well as mid- and long-term storage. Flash storage devices are also increasingly being used for longer-term storage, and the bulk of this storage space is often in the form of removable memory cards.

Some Very Quick Math to Prove a Point About Memory Sizes

A rich mobile device today may have 64MB of RAM. This RAM is often partitioned between program RAM and virtual file system. Let's assume that 32MB of this RAM goes to the file system to hold all the long-term data you work with (things such as photos, documents, music, and other information). This leaves 32MB for the operating system and applications to share. Assume that five applications are running simultaneously (not uncommon), all are using roughly equal amounts of RAM, and that the operating system further uses resources equal to an application. This leaves a ration of 5 to 6MB of RAM for each application to use. This size is considerable but by no means infinite. A few big digital photographs brought into memory could consume most of this RAM. Many mobile devices have considerably less RAM to work with and may be asked to run more applications simultaneously. The RAM available on the device sets the absolute limit and is non-negotiable. If you use up the available physical memory, things will not be moved to a page file on hard disk as it would on the desktop. Most likely you will run out of memory and your application will crash.

A few megabytes of working space is pretty good when used effectively in the same way that a one-bedroom apartment in Manhattan can offer a nice amount of space so long as you do not try to fill it with too much stuff. Keep piling more stuff in and you will reach a critical point where nothing else will fit and you can only navigate around the apartment with extreme difficulty. With enough stuff in it, your apartment will become unusable. So it is with mobile devices.

Summary

Mobile devices differ significantly from desktop computers and laptops. Because of this, applications for mobile devices differ from desktop applications in significant ways. Desktop applications tend to get used in long sessions, and great desktop applications offer the user an exploratory environment for the information they work with. Mobile device applications tend to be used in short spurts, frequently, but for short session durations. Because of this, great mobile device applications offer a focused and efficient experience for accomplishing specific tasks rather than a general exploratory environment.

Quick startup time, responsiveness, and focused purpose are the hallmarks of good mobile application design. All these things amount to a highly productive user experience when using mobile devices. Keep these goals in mind when designing, building, and testing your mobile device application.

Architecturally, mobile devices differ from desktop and laptop computers in that most mobile devices do not have a hard drive and often use the available RAM for both program execution and file storage. Increasingly, mobile devices can use flash memory for long-term file storage. Flash offers good long-term storage capability but is typically not used as an extension of program RAM the way desktop computers offer a disk-based paging file for virtual RAM extension. This means that efficient memory management is more important for mobile devices than it is for desktops because execution RAM is a more limited resource.

With regard to the need for reliability, mobile devices have more in common with servers than they do with desktop computers. Like servers, the available RAM drives the overall performance of the system, and devices are often left on for weeks or months at a time without rebooting unless the user detects that things have gone drastically wrong. Ensuring that your applications efficiently manage their resources and particularly that they do not leak memory will have a significant impact on the overall performance of the device and the satisfaction of end users. Managed-code runtime environments can be a great aid in this effort.

A useful metaphor is to think of personal computers as being analogous to big houses in the countryside with lots of available storage space. Mobile devices are analogous to apartments in a metropolis, small and efficient. Both can offer comfortable living environments with their own advantages, but to live effectively accommodations must be made to meet the physical realities of the space available.