Phony Programming
(Series 60 Symbian Phones)

Larry Rudolph
MIT 6.883
Feb 13, 2007

What's so special about phones?

- **Ignorant Owner**
  - perhaps a really ignorant owner or user
- **Scarce Resources**
  - Power -- slow processor, small memory, small disk (expanding)
  - Size -- Screen and keyboard area
  - Heat -- To keep this low, must constrain above
  - Price -- Different than PC, since each feature costs the same, due to volumes
  - Reboot -- these are rare events
  - Brand loyalty -- hardly any, mistakes are costly
- IT'S NEW AND EXCITING!

The introductory lecture briefly addressed the issue as to why phones are different. Mobility was the big reason, and we will see that this simple term has many implications.

We are all used to using computers. At times we may get frustrated and at times we may be impress, but
no matter what, we understand how to use computers. People understand how to use a telephone. Turn it on, dial a number and talk. It is an appliance with a well defined basic interface.

Is important that any user will always be able to use the device as a phone? The answer is important to the
design. If the answer is yes, then one must make sure that an ignorant user will not keep poking at things
causing lots of damage. If the answer is no, we can assume that the device is a personal one and can be tailored to the owner.

There are several features of phones that make them different. Conservation of power is

What's so special about phony programming?

- Inherently very little -- but in practice ...
- History
  - originally: phones were h/w appliances
  - past: s/w - h/w co-design reduced costs
  - near past: new features all s/w
  - near future: 3rd party software
- In computer world, two approaches
  - Intel: H/W, Microsoft: O/S, separate h/w from s/w
  - Apple: H/W & S/W, easier to integrate, high costs $
- Companies have no history of being open !!

Why do we need to think about programming the phone in any special way? PDAs and handhelds are not so different than PCs. Phones seem much different, why?

There are the constraints, but there is also history. Phones grew up as being appliances with fixed functionality, and then to appliances with additional features. Currently they are devices that are still pretty much co-owned by the operators and manufacturers. Although we buy them, it feels like we are renting them from the operators. Many have the name of the operator inscribed on the device itself.

Operators do not want their subscribers call the service department. It is very expensive. On the other hand, it seems clear that software helps sell hardware. The killer application is what makes people choose one device over another. Earlier, we tried to argue that it is fashion. Which do you think is more important.

But the trends are hard to fight. Software and interesting applications will be developed by third parties.

Current Choices

- Most smart phones have two ASICs
  - Modem/Telephone service (DSP)
  - Embedded processor and extra features
- Palm-Based: Slow to integrate
- Pocket-PC: Appears to be aggressive
  - Can they maintain two OS's (Windows & WinCE)?

What is interesting is that the choice is so limited. In the early days of computers, and the early days of any technology that is high risk, high payoff, there are lots of choices. As time goes on, the choices narrow.

Actually, there are many operating systems for phones and handhelds. The only problem is that we do not get to see them or program with them. Most of the operating systems are hidden and embedded into the phone. We only see the user interface.

Palm Phones

5

Palm was one of the companies that started the PDA revolution. They had a wonderful user interface. It is still surprising to see people with PDAs that are not also cell phones. They do one thing right.   Many people do not want the compromises that are inherent in "convergent" devices.   The palm phones are more expensive, have smaller screens and other limitations. They are not such great
phones either, having touch screens, which many people do not like on their phones.

But what really happened to Palm? It had a very loyal customer base. Any opinions?

Pocket PC Phones

6

PocketPC phones are popular for the obvious reasons -- familiarity and presumed streamlined integration. Of course there is a difficult balancing act. Microsoft operating systems are well known to require lots of resources -- memory and cpu. The strength of microsoft has been its ability to handle a wide range of third party hardware devices. This is of little help on an integrated phone. The other feature is the familiar "look and feel". But the look and feel that is geared for a large screen and easy to use multi-button mouse does not easily translate to the phone device's constraints.

Does anyone know how much microsoft charges manufacturers for PocketPC?

The good news is that PocketPC is programmable. There are lots of third party applications. One web site claims over 20,000 applications. Anyone have any experience with them?

I have had two bad experiences with pocket-pc. The first is the fact that third party applications get installed into sdram and not onto flash. There is a second backup battery for keeping the data in sdram valid, but if the device fully discharges, all the software must

Linux Based Phones

7

OpenMoko promises to be highly programmable. They are trying very hard to keep the software open. Some of us are paranoid and only when we can see the code, do we trust that it is not doing anything that might compromise our privacy. But even with open software, there is still the treat that the phone could be upgraded over the air without our knowledge.

Apple's iPhone

But its not like the walled garden has gone away. You dont want your phone to be an open platform, meaning that anyone can write applications for it and potentially gum up the provider's network, says Jobs. You need it to work when you need it to work. Cingular doesnt want to see their West Coast network go down because some application messed up.

We define everything that is on the phone, he said. You dont want your phone to be like a PC. The last thing you want is to have loaded three apps on your phone and then you go to make a call and it doesnt work anymore. These are more like iPods than they are like computers.

The iPhone, he insisted, would not look like the rest of the wireless industry.

These are devices that need to work, and you cant do that if you load any software on them, he said. That doesnt mean theres not going to be software to buy that you can load on them coming from us. It doesnt mean we have to write it all, but it means it has to be more of a controlled environment.

8

We can have an interesting discussion about the iPhone. It appears that it will not be so easy to write your own code to run on the phone. That would be too bad, since I believe there is lots of innovation that is just waiting
to happen.

Symbian Phones

9

Symbian is a different operating system and currently, it is the easiest platform on which to program. The choice was based on Python, which makes it easy to prototype and test out features. Symbian is trying hard to be the platform of choice for 3rd party software. As usual, in the beginning, support was only for official collaborators. There was little help for non-companies. That seems to be changing, and the more we do in this class the better.

Symbian Epoc OS

- Originally developed for the Psion handheld computer
  - competition with Palm
  - single user, small memory, instant-on, no network
- EPOC operating system
- Symbian independent company
  - partly owned by Nokia, Sony/Ericsson, Panasonic, Seimans, Samsung (no one controls them)
  - EPOC and Symbian names became intermixed
- Nearly all documentation and tools were for commercial developers but things are changing
  - high start-up cost,
  - Three different major OS's: for Nokia, Sony/Ericsson, NTT (more on this later)

CSAIL   Pervasive Computing MIT 6.883 Spring 2007 Larry Rudolph   10

Although most people know of Windows and various flavors of Unix operating systems, there have been a huge number developed for all different types of special purpose applications.   Real time systems is an are of much diversity as are embedded systems. Symbian grew out of the embedded PDA world and somehow manage to survive. The situation is unique. Symbian is now jointly owned mostly by Nokia, and Sony/Ericsson -- and to a lesser degree, Pansonic, Seimans, Samsung -- but Nokia depends the mostly on them. It is unique because in many ways Nokia is too dependent on them. There are various versions, Symbian is up to version 9. Nokia calls them Series xx (e.g. 40, 60, 80 ...)

Deja Vu

- Despite the fact that
  - Programmable Mobile Phones are new artifacts
  - the OS and programming repeat mistakes
- Designs that make sense for disconnected, single use devices, remain as devices become connected and multi-use and multi-tasked.
  - Inertia is a powerful force

We see this time and time again. Microsoft DOS is a prime example, but there are many others. A system is developed without thinking about the implications of multi-user, multi-tasking, security. It then gets adapted to be connected and security becomes a big problem. Perhaps there is a "law" -- something like, every computer-based device eventually evolves to be interconnected with other devices.

I am sure I will continue to repeat my fear of the new ability to update the firmware of a phone over the air (OTA). This has so many serious implications.

Programming Languages Series
60 Phones

- C++ for Symbian
  - access to all the phones functions
  - not so easy
- Java
  - highly sandboxed.
  - no access to file system, phone, and more
  - not the choice language for virus writers
- Python
  - will have interface to all Symbian API's via extensions
- Adobe Flash

For a variety of reasons we will be programming Nokia Symbian Series 60 phones. Unfortunately,
not all series 60 phones are the same; it is an unfortunate naming scheme, but we are stuck with it.

Symbian was first developed as object oriented languages were starting to become popular but the technology was not quiet mature yet. Symbian choose C++ as the language for its OS. But there is no automatic garbage collection, and there was no good exception mechanism either. They added their own and that has made it difficult to program in Symbian C++. We will use it, but only when needed.

Java should have been the ideal language for programming phones, especially with the write once -- run everywhere philosophy. There are several problems. 1. Java is owned by Sun and so Sun controls the JVM. You cannot just put a JVM on your machine without their approval. 2. Java, at least on
many of the Nokia versions I know about, is highly sandboxed and does not allow access to things like the file system, mailboxes, telephone functionality, and address book. 3.

Symbian OS Basics

- Kernel: Protected Mode; Controls H/W

- Server: Manages one or more resources
  - no UI (user interface)
  - yes API (application program interface)
  - may be device driver or kernel service

- Application: A program with a UI
  - Each application is a process; own virtual address
  - If interacts with server, can be called a client

- Engine: part of app. manipulates its data not UI

Symbian is organized around servers. Resources are managed by servers. Servers run in the background and do not have a user interface. They may be part of the operating system and provide basic service, like incoming phone call notification or controlling the audio driver. They may also be user defined servers. Since it is difficult for a user to see if a server is running, it makes a great place to hide a virus.

There is usually only one copy of a service for each resource. They run in their own address space, and they are interrupt driven. They do not run without an event first waking them up.

An application interacts with one or more servers. An application has a user interface. It runs in its own virtual address space (but there is no swapping).

Symbian also uses the term <u>engine</u> which is kinda like a server. It is part of an application but without the user interface part. I like this distinction, as the user interface is crucially important for devices with limited screen area and keys. I think the difference between an engine and a server is that an engine exits when the application exits. It is like a thread of

Processes & Threads

- Three boundaries
  - DLL or module: cheap to cross
  - Privilege: medium to cross
  - Process: expensive to cross

- Processes, Threads, Context Switches
  - Process: has its own address space (256 MB)
  - Thread: Unit of execution within a process
    - Preemptively scheduled; 2MB nonshared ==> 128 thds/pro

- Executables
  - exe: single entry point
  - dynamic link library (DLL): multiple entries
    - Shared (OS) vs Polymorphic (app)

The truth is that I keep confusing symbian's definitions with those of unix. There are subtle differences, mostly in restrictions. I am always very careful when using threads in symbian (and nearly always get caught with some hard to track bug).

RAM Memory Parts
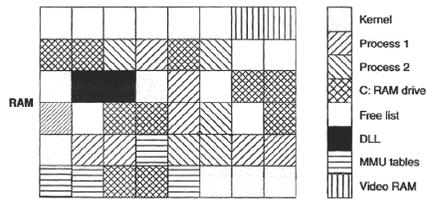
15

The ARM v6 processor has virtual memory and paging, however, Symbian does not use the paging facility. When physical memory runs out, processes die and new ones cannot be started.   It also does not use the variable sized pages the ARM supports. However, virtual memory provides lots of good protection. When a process terminates, its pages are freed.

Symbian makes a big deal about trying to prevent or limit memory leaks. This is because applications may run for a very long time -- e.g. the calendar application. But things like games, or our prototype applications, should not have to worry too much about memory problems.
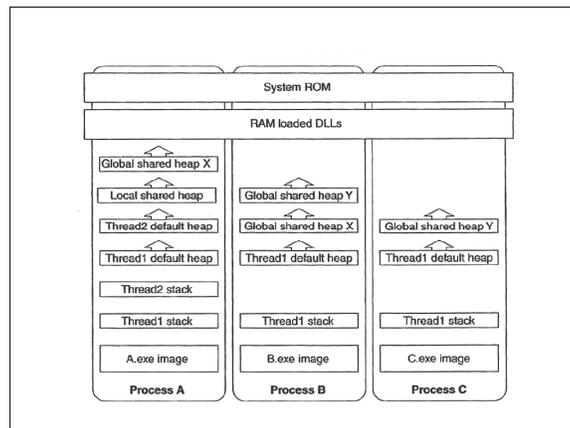
16

Memory

- RAM: partitioned into 4k pages
  - kernel, process, DLL, MMU tables, Video, C:
  - Thread memory: Shared:Heap, Nonshared:Stack
- DLL -- no writable static data (yes for exe's)
  - requires multiple copies of instantiated dll
- Files (does this remind you of something)
  - C: RAM -- r/w file system. Zero'd on cold boot
    - restored from ROM on cold boot
  - Z: ROM -- can be reflashed (not easy)
  - D: Memory Card
    - 512 byte blocks written atomically; VFAT format

Event Handling

- Efficient handling major OS design
  - native OS server is single event-handling thread
- Symbian organized as Event-driven
- Active Objects: non-preemptive event handling
  - and client-server structure

Response to key-press
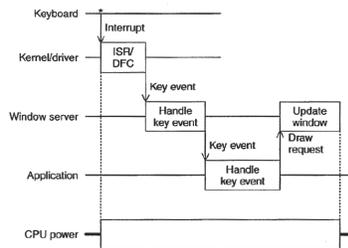
Active Objects

- Each active object has virtual member function called **RunL()**
  - gets called when event happens
  - events have associated priority
- Active Objects execute **non-preemptively**
  - RunL() should execute for short time
  - no need for mutex, semaphores, critical sections
  - fewer context-switches
- Compute-intensive threads:
  - Simulate using pseudoevents
  - split task into pieces, generate low-prio event
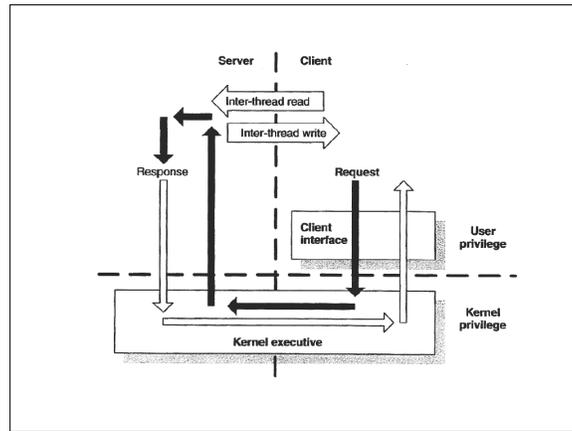
21

Java Programming

- Lots of examples at www.forum.nokia.com
  - great for games and network connectivity
- Compile on server
- Install on phone via:
  - bluetooth connection
  - mms message (email)
  - upload from web server
- On phone, java applet must be opened to install before being run.

22

C++ Programming

- The real stuff but documentation is difficult

- Everything depends on the SDK

  - Software Development Kit runs under Windows
  - The processor on phone is ARM (same as iPaq)
  - Must do cross-compilation
    - compile with different libraries and assembly instructions

- There is an "Emulator" -- but it is really a simulator

  - e.g. it executes x86 code not arm code
  - so must compile either for ARM or   x86
  - emulator is needed for debugging

C++ Programming

- Tutorials: C++ for Java programmers
  - http://www.cs.brown.edu/courses/cs123/javatoc.shtml
  - There are many others on-line
  - Easier to go from java to c++ (java has less weirdness)

Symbian Layers

Group	Description
Base	Provides the fundamental APIs for all of Symbian OS, which I've described in this chapter.
Middleware	Graphics, data, and other components to support the GUI, engines, and applications.
UI	The system GUI framework including the Shell (in UIQ, the Application Launcher) application.

Applications	Application software can be divided into GUI parts (which use UIQ) and engines (which don't deal with graphics). Some applications are simply thin layers over middleware components: others have substantial engines.
Communications	Industry-standard communications protocols for serial and sockets-based communication, dial-up networking, TCP/IP, and infrared.
Language systems	The Java runtime environment.
Symbian OS Connect	Communications protocols to connect to a PC, and services such as file format conversion, data synchronization for contacts, schedule entries and e-mail, clipboard synchronization, printing to a PC-based printer.

Symbian 60 Phone Programming in Python

Pervasive Computing MIT 6.883 Spring 2007 Larry Rudolph

Installing stuff

- a package or installation file in symbian:
  - application_name.sis
- Get it onto the phone
  - push via bluetooth (or send message)
- Open up message and install
  - if there is flash memory, install it there

Pervasive Computing MIT 6.883 Spring 2007 Larry Rudolph

S60 2nd Edition, Feature Pack 2 S60_2nd_fp2_msb.zip PythonForSeries60_1_2_for_2ndEd_FP2_SDK.zip	6630, 6680, 6681	
S60 2nd Edition, Feature Pack 1 S60_sdk_2_1_NET.zip PythonForSeries60_1_2_for_2ndEd_FP1_SDK.zip	7610, 6620, 3230	
S60 2nd Edition S60_sdk_v2_0.zip	6600, 3650	

---

import appuifw
appuifw.note(u'Hello World',u'info')

- import appuifw # the application user interface fw(?)

- u'Hello World', # u' ' for a unicode string. All GUI

- # strings are unicode. Others can be

```
import appuifw
planet = appuifw.query(u'Which planet?',u'text')
appuifw.note(u'Hello '+planet , u'info')
```

- "query()" pops up a dialog with prompt string and input type

- other types can be 'number' 'date' 'code'

- the '+' concatenates the two unicode strings

31

```
import appuifw
planets = [ u'Mars', u'Earth', u'Venus' ]
prompt = u'Enter your home planet'
index = appuifw.menu(planets, prompt)
appuifw.note(u'Hello '+planets[index] , u'info')
```

- The 'menu' method pops up the list of items in first param

- It returns with an index into the list of menu items

- Note that the prompt param must also be a unicode string

32

Our own interface:
S833.py

There are a bunch of annoyances in the current UI

Let's put wrappers around basic calls

We should go back and do this for location

CSAIL   MIT Massachusetts Institute of Technology

33   Pervasive Computing MIT 6.883 Spring 2007 Larry Rudolph   33

```
# this is file s883.py
# wrappers to appuifw, e32, and bluetooth

def note( str , type = 'info'):
    appuifw.note( unicode(str),  type )

def query( str , type = 'text' ):
    return appuifw.query( unicode(str),  type )

def menu( list, prompt = 'select one' ):
    ulist = [ unicode(u) : for u in list ]
    return appuifw.menu( ulist , unicode( prompt ) )
```

34

```
import s883
planets = [ 'Mars', 'Earth', 'Venus' ]
prompt = 'Enter your home planet'
index = sma.menu(planets, prompt)
sma.note('Hello '+planets[index] )
```

35

- It is easier to go through the python reference document, rather than reproducing it all here...

36