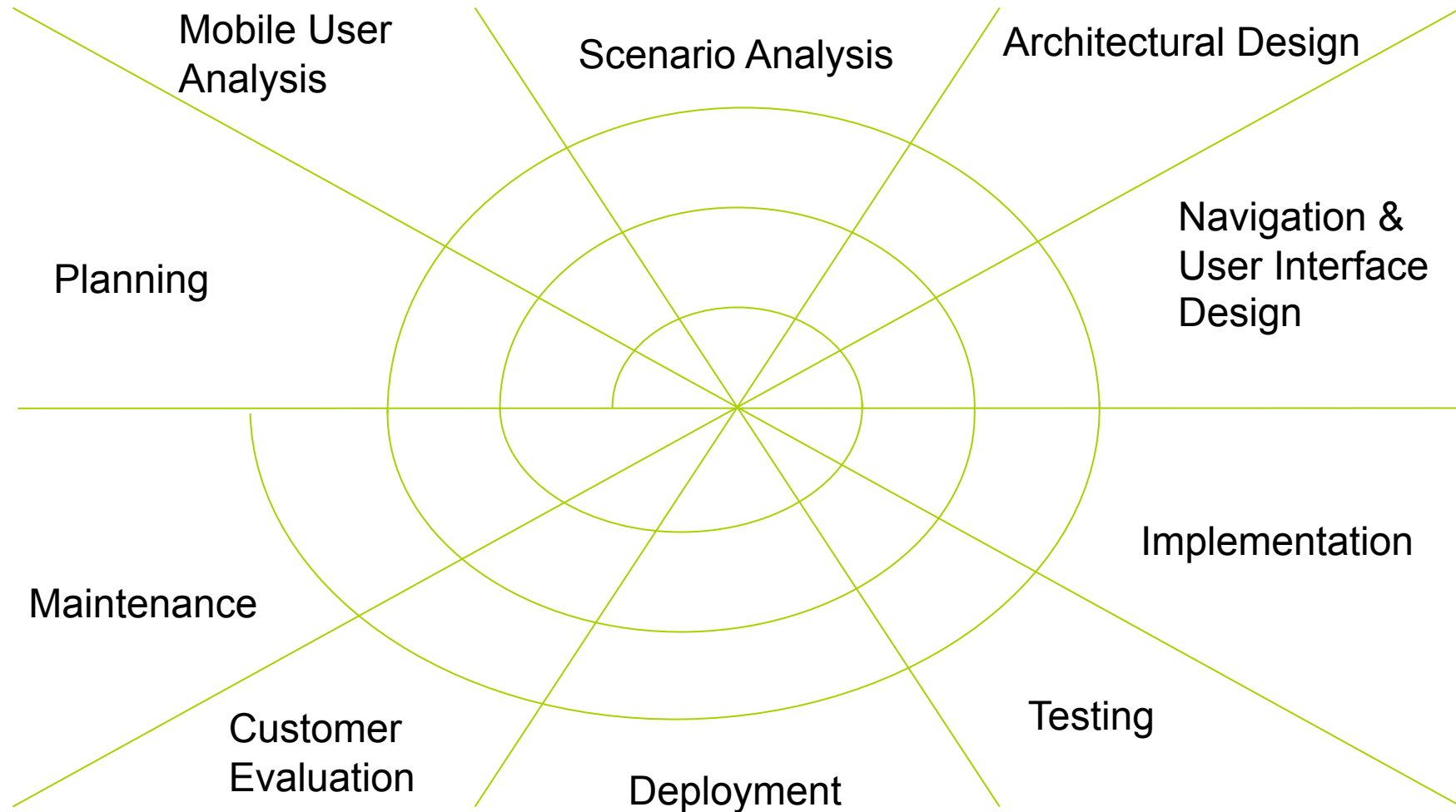


Building Mobile Applications



F. Ricci
2010/2011

Wireless Software Engineering Model



Planning

- ❑ Identifying the general **objectives** of the application
- ❑ Specifying the scope of the first release
 - This will be revised ... in a next step
- ❑ Estimating the possible cost of the project
 - Time required
- ❑ Analyzing the possible risks involved
 - Plan for repair measures
- ❑ Creating a tentative schedule
 - build a schedule of your project so that you can deliver the project at the end of January.
- ❑ **Start writing the report as soon as possible** so that you can insert the required info as you go ahead with the analysis and development.

1. Mobile User Analysis

- This phase deals with understanding our **target audience** for the application
- We need to examine the **types of users** that will use the application and any special requirements or functionality they require
- We can gather information by:
 - Asking experts in the field
 - Reading literature on the topic
 - Asking current users
- *Practically speaking: build an hypothesis and revise it with your colleagues.*

2. Scenario Analysis

□ **Usage Analysis**

- The functionality of the system, use cases should be considered to relay this information

□ **Screen and Interaction Analysis**

- How the user will interact with the system, what content and how will be displayed

□ **Environment Analysis**

- The interaction between this system and other networks, servers, and devices.

Briefly comment about the outcome of this analysis in the project report.

Take into account ...

- **A mobile user** may be:
 - Walking, running, or driving
 - Operating in lighting conditions that are too dark or too bright to see the screen or device keypad properly
 - Relying on one hand to initiate and complete tasks
 - Trying to complete a task by glancing occasionally at the screen.

3. Architectural Design

- ❑ This phase deals with the software architecture of the system when dealing with network based applications
- ❑ It should discuss how the system will be split into different subsystems and the challenges and benefits of such a design
- ❑ As discussed, attention must be paid to **message latency** and **application partitioning** to ensure **performance, reliability, and security**
- ❑ This analysis will have an impact in how the code is implemented (e.g., the usage of multiple threads).

Briefly comment about the outcome of this analysis in the project report.

Network

- ❑ Messages sent to clients or servers can be delayed due to a variety of reasons such as **overloaded network nodes** or servers, **dead or turned off cell phones**
- ❑ Applications must take this into account so as to avoid sending servers or clients **stale information** or requiring the user to **wait too long**
- ❑ In a **client-server** architecture the server can **store** messages that do not arrive at the mobile device and **attempt to resend** them at specific intervals
- ❑ Servers can also store the message and send it when the mobile device reconnects to the system
- ❑ **Let the user know if they receive a message that can possibly be out of date** or no longer valid, this could be done using timestamps (and explicitly pointing out this in the GUI).

Network

- ❑ Pass **as little messages as required** between the client and the server
- ❑ **Keep the messages as short as possible**, you can use symbols to represent commands for the server
- ❑ If your application must use a lot of bandwidth at least **notify the user of this fact**
- ❑ Wireless customers are forced to **PAY** fees to access the wireless network and internet
- ❑ While phones with WIFI capabilities allow for some users to have free connectivity at times it is important to **keep messages to a minimum and compact**
 - *Still WiFi may be intermittent*
- ❑ Applications that cost a lot to use **will not be popular** with many of the financially conscious users.

4. Navigation & User Interface Design

- ❑ The design of the **user interface** is critically **important** for the success of the system
- ❑ A poorly designed interface will detract the user from the system while a clean and easy to use interface will show professionalism and perception
- ❑ Keep in mind the challenges we have been discussing with user input, screen sizes, and display characteristics as these play important roles in designing an interface
- ❑ Screen **mock-ups** are important to display the look and feel to potential customers.

Briefly comment about the outcome of this analysis in the project report.

Display/Screen

- ❑ Mobile devices come in many different screen sizes
 - **Smartphones:** larger and higher resolution display screen
 - **Cell phones:** lower resolution and smaller display size
- ❑ Decide **what screen** (emulator) you will use for your project
- ❑ Take into account how much text is required by the user to input into your application and what kind of difficulties they may experience
- ❑ To overcome the problems that can occur with the different input devices **make input requirements concise**
 - the user should be able to perform the most common tasks in an application with **the least amount of button presses**
 - Provide users with **menus** when possible to help reduce the amount of button input required.

Optimize interaction

- ❑ **Clarity.** Do not make it more complex than it has to be. With the lack of screen real estate and challenges of manipulation, there is already enough complexity in the small device.
- ❑ **Simplicity.** Take every measure to simplify the actions users need to achieve a goal. Narrow down the functionality to what is essential.
- ❑ **Context.** Keep in mind both the physical and social context of use and any other difficulties users may encounter related to the environment.
- ❑ **Learning.** Take advantage of innovations that have worked in the past, such as design patterns. Do not reinvent the wheel just because it seems cool – look at good applications on your phone.

5. Implementation

- ❑ Implementation of the application is done in this phase with the use of any number of development tools
- ❑ Code conventions, class and object diagrams, API specifications can all be included in any documentation that is created at this point for the system
- ❑ This allows developers who join the project at later stages to follow the same format and style as the original authors.

Memory

- ❑ Developers must create applications which have a **minimal memory footprint** on the device while being of service to the user
- ❑ Memory must also be **carefully managed during the execution of any mobile application** as it can potentially render the phone unusable until termination of the application
 - 1) **Compact data representation** will help reduce the amount of memory it requires to load and use your application
 - 2) Use **optimization techniques** to reduce the amount of code required to write your application
 - 3) The bytecode **obfuscator** can reduce the size of your classes (access obfuscator: "properties" of the Netbeans project).

Memory Monitor

The screenshot shows the 'Memory Monitor Extension' window. The title bar reads 'Memory Monitor Extension-+5550000 - DefaultColorPhone - Sun Java(TM) Wireless Toolkit'. The menu bar includes 'File', 'Utilities', and 'View'. Below the menu bar are buttons for 'Open', 'Save', and 'Run GC'. The 'Objects' tab is selected, displaying a table with the following data:

Name	Live	Total	Total Size	Average Size
java.lang.String	325	1490	7800	24
java.util.HashtableEntry	171	171	4788	28
java.lang.Object[]	96	353	3724	38
VM Internal	23	223	3720	161
java.util.Vector	96	233	2304	24
java.util.HashtableEn...	16	25	1576	98
java.lang.String[]	40	103	1416	35
byte[]	14	39	1128	80
int[]	22	47	664	30
java.util.Hashtable	16	16	384	24
java.lang.Object	26	26	312	12

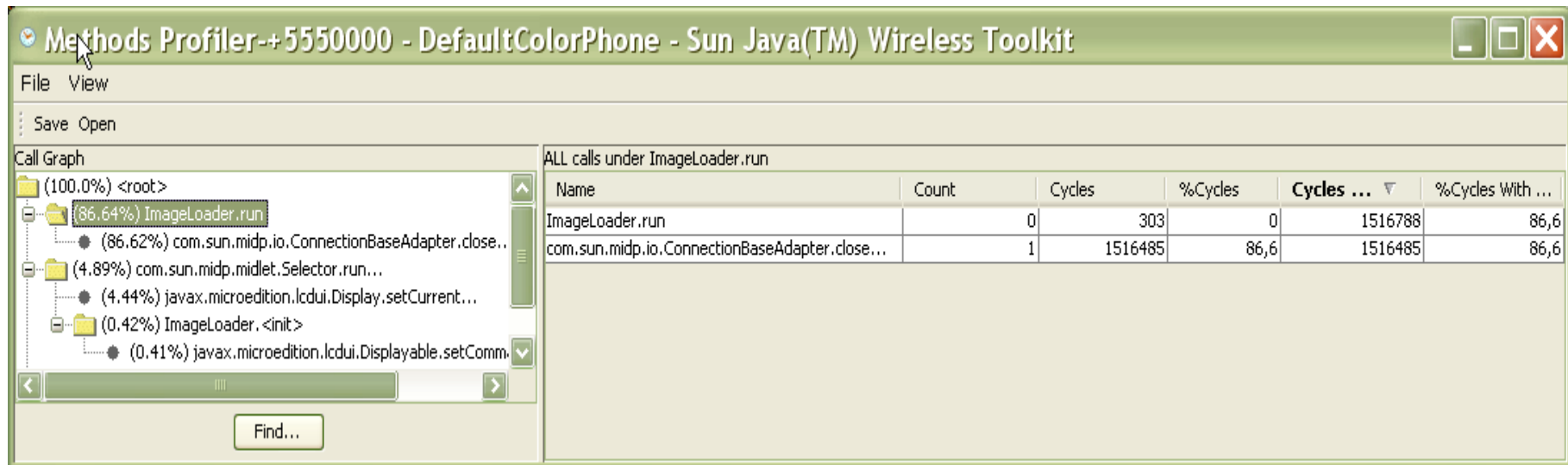
At the bottom of the table, it says 'Objects: 1153', 'Used: 68980 bytes', 'Free: 2028172 bytes', and 'Total: 2097152 bytes'. To the right of the table is a detailed view for the selected 'java.lang.String' object, showing its memory usage breakdown: (95.5%) java.lang.Class.runCustomCode(), (0.4%) com.sun.midp.lcd.ui.DefaultEventHandler\$QueuedEventHandler.run(), and (4.09%) ImageLoader.run(). There are 'Find...' and 'Refresh' buttons at the bottom of this panel.

- ❑ You can switch on the memory monitor from the "preferences" menu of the WTK
- ❑ **Name** - Class name of the objects
- ❑ **Live** - Number of instances. Some of these might be eligible for garbage collection
- ❑ **Total** - Total number of objects that have been allocated since the application began
- ❑ **Total Size** - Total amount of memory used by the objects
- ❑ **Average Size** - Average object size, calculated by dividing the live instances into the total size

Processing Power

- ❑ The **CPUs differ from phone to phone** and this must be taken into consideration by developers
- ❑ Developers cannot create applications that require the user to wait an unreasonable amount of time for the service to **load** or **execute**
- ❑ If the mobile application has a client-server architecture consider the **partitioning** of the application
- ❑ Allow the **server** to do the **brunt** of the calculations and processing work and pass the information to the **mobile device** for **less CPU intensive calculations.**

Profiler



- ❑ The call graph shows (when the midlet terminates) a hierarchy of method calls - Methods that call other methods
- ❑ Cycles shows the amount of processor time spent in the method itself.
- ❑ %Cycles is the percentage of the total execution time that is spent in the method itself
- ❑ Cycles with Children is the amount of time spent in the method *and* its called methods
- ❑ %Cycles with Children shows the time spent in the method and its called methods as compared to the total execution time.

Profiler in WTK 3.0

- Enable the profiler: right click on the emulator and access the emulator properties

The screenshot shows the IDE interface with the Profiler tool active. The Profiler window displays a Call Tree for the class 'com.sun.midp.events.EventQueue'. The table below shows the call tree data:

Class	Time [%]	Time	Invocations
All threads		468 ms (100%)	1
java.lang.Thread		421 ms (90%)	8
com.sun.midp.events.EventQueue		421 ms (90%)	1
com.sun.midp.lcd.ui.RepaintEventProducer		375 ms (80,1%)	29
com.sun.midp.lcd.ui.DisplayEventListener		31.0 ms (6,6%)	69
com.sun.midp.lcd.ui.ForegroundEventListener		15.0 ms (3,2%)	1
Self time		0.000 ms (0%)	1
com.sun.midp.midlet.MIDletEventListener		0.000 ms (0%)	1
com.sun.midp.lcd.ui.LCDUIEventListener		0.000 ms (0%)	44
java.lang.Object		0.000 ms (0%)	125
com.sun.midp.events.NativeEventPool		0.000 ms (0%)	71
Self time		0.000 ms (0%)	8
com.sun.midp.installer.InstallRetryHandler		0.000 ms (0%)	1
com.sun.midp.events.NativeEventMonitor		0.000 ms (0%)	74
javax.microedition.lcdui.LFTimer		0.000 ms (0%)	1
java.lang.String		0.000 ms (0%)	2
com.		0.000 ms (0%)	5
com.		47.0 ms (10%)	2

Right click here and select "properties"

6. Testing

- ❑ Testing is extremely important in mobile application development not only due to the heterogeneity of mobile devices
- ❑ It is important to test not only in an emulator but **on the physical device as well**, and to test on all or as many physical devices as the application can be located
- ❑ Testing also assists us to remove bugs and flaws in programs which become inevitable in larger systems as they become complex
- ❑ Use Cases are a helpful tool in generating test cases for the system.

7. Deployment

- ❑ Deployment of the application on physical devices will allow you to see the system in the real world
- ❑ Applications may be fine in an emulator but when transferred to a mobile device developers may find the application **slow, impossible to use, not functioning** all together, or consuming **too much bandwidth**
- ❑ It may not economically feasible to test on every possible device that a system may be used on but a wide variety of devices should be tested.

8. Customer Evaluation

- ❑ At this point the application is ready for download by customers in the network
- ❑ Customers should be given a way of providing feedback to the developers and reporting any issues they encounter when using the application
- ❑ Consider providing an email or web form where users can fill in the necessary info or provide an automated process which sends the error to the server.

9. Maintenance

- Maintenance of the system after deployment deals with several issues:
 - Resolving any bugs found in the application and creating necessary patches
 - Improving the quality of the application with upgrades
 - Providing new services and capabilities to customers.

References

- ❑ Mahmoud, Qusay H., and Zakaria Maamar. "Engineering Wireless Mobile Applications." Int. J. of Information Technology and Web Engineering 1.1 (2006): 58-73.
- ❑ Forumnokia. Getting Started with Mobile Design. Version 1.0; June 5, 2008.
- ❑ Forumnokia. Designing MIDP Applications For Optimization Version 1.0; August 31, 2004
- ❑ Mobile design resources. http://patterns.littlespringsdesign.com/index.php/Main_Page
- ❑ All these docs are available on the course web site.