CS 423 – Operating Systems Design

Lecture 25 – Symbian OS

Klara Nahrstedt
Fall 2011

Based on slides from Andrew S. Tanenbaum textbook and other web-material (see acknowledgements)

Overview

- Administrative Issues
  - Last week for re-grading
- Symbian OS Overview
- Ultra-Mobile Development Principles

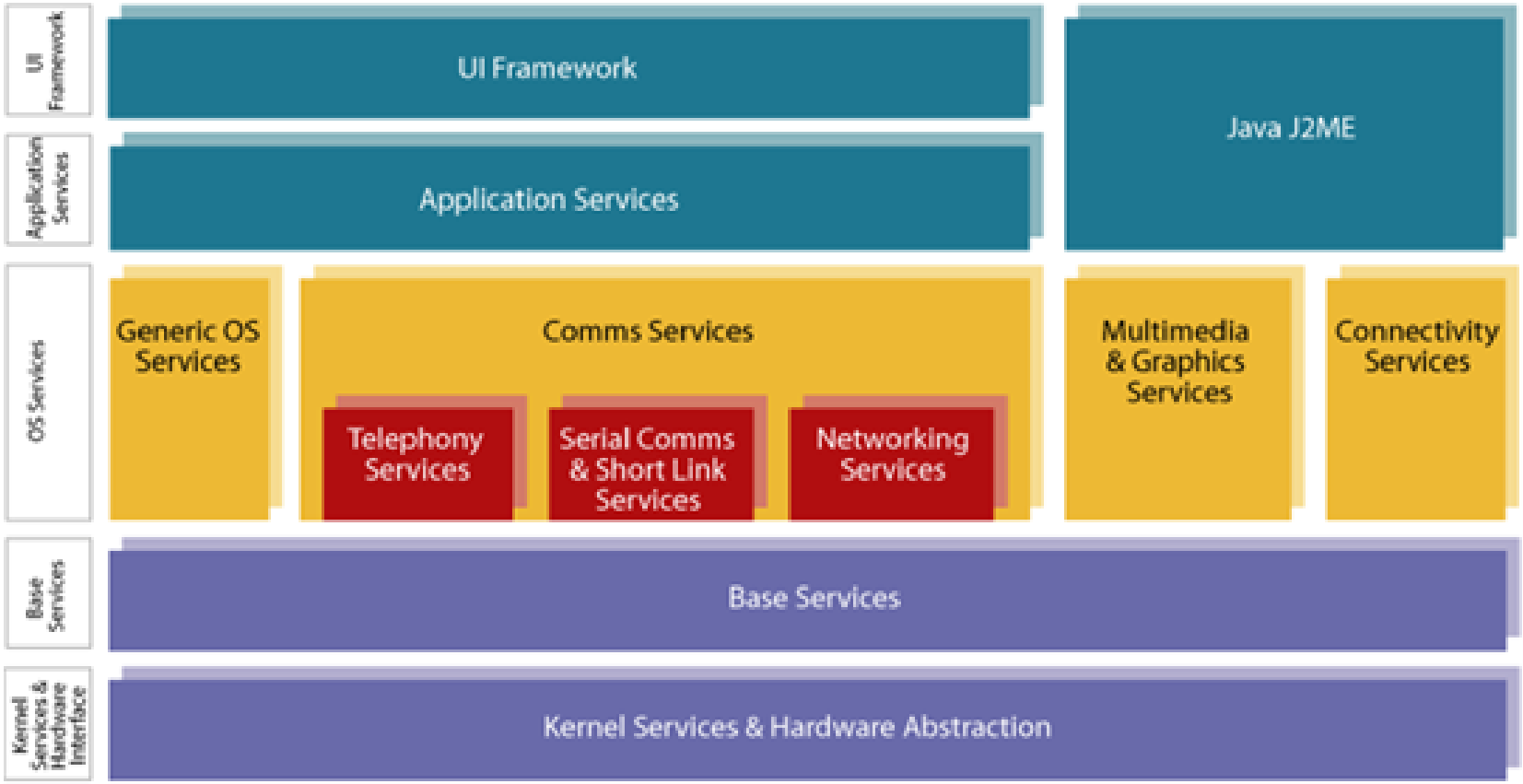
History of Symbian OS

- 1996 – Psion designed 32-bit OS that
  - Supported pointing devices on touch screens, used multimedia, and was communication rich
  - Was more object-oriented
  - Was portable to different architectures and device designs
- Result EPOC OS Release 1
- Further expansion towards touch screen, generalized hardware interface, …
  - EPOC Release 3 and 5 (ER3, ER5) run on platforms Psion Series 5 and 7.
- 2000 - Psion and its EPOC OS as basis of Symbian OS on mobile phone platforms Nokia, Ericsson, Motorola, Panasonic

Symbian OS Overview

- Is Object-oriented
- Inherited from EPOC
- Uses micro-kernel design
  - Minimizes kernel overhead and pushes non-essential functionality to user-level processes
- Uses client-server architecture
- Single-User
- Supports multi-tasking, multi-threading
- Supports extensible storage system
- Inherited multimedia and communication emphasis

Symbian Architecture



Object Orientation

- Object-oriented design
  - Creates an abstract entity called object of data and functionality of system component
- Object
  - provides data and functionality
  - Hides details of implementation
  - Can be removed, replaced by other object as long as its interface remains the same
- Object-oriented kernel
  - Provides kernel services through objects
  - Application obtains handle to kernel-side objects
- Object orientation is designed into the entire OS framework

Micro-kernel Design

- Minimal system functions in kernel
- Many system functions pushed out to user space servers
  - Servers do their work by obtaining handles to system objects and making system calls through these objects into the kernel
  - User applications interact with these servers rather than make system calls.
- Micro-Kernels
  - take much less memory space upon boot
  - the structure is more dynamic and flexible
  - servers start as needed, not all servers required at boot time
  - implemented as pluggable architecture for systems modules that can be loaded as needed and plugged into the kernel.

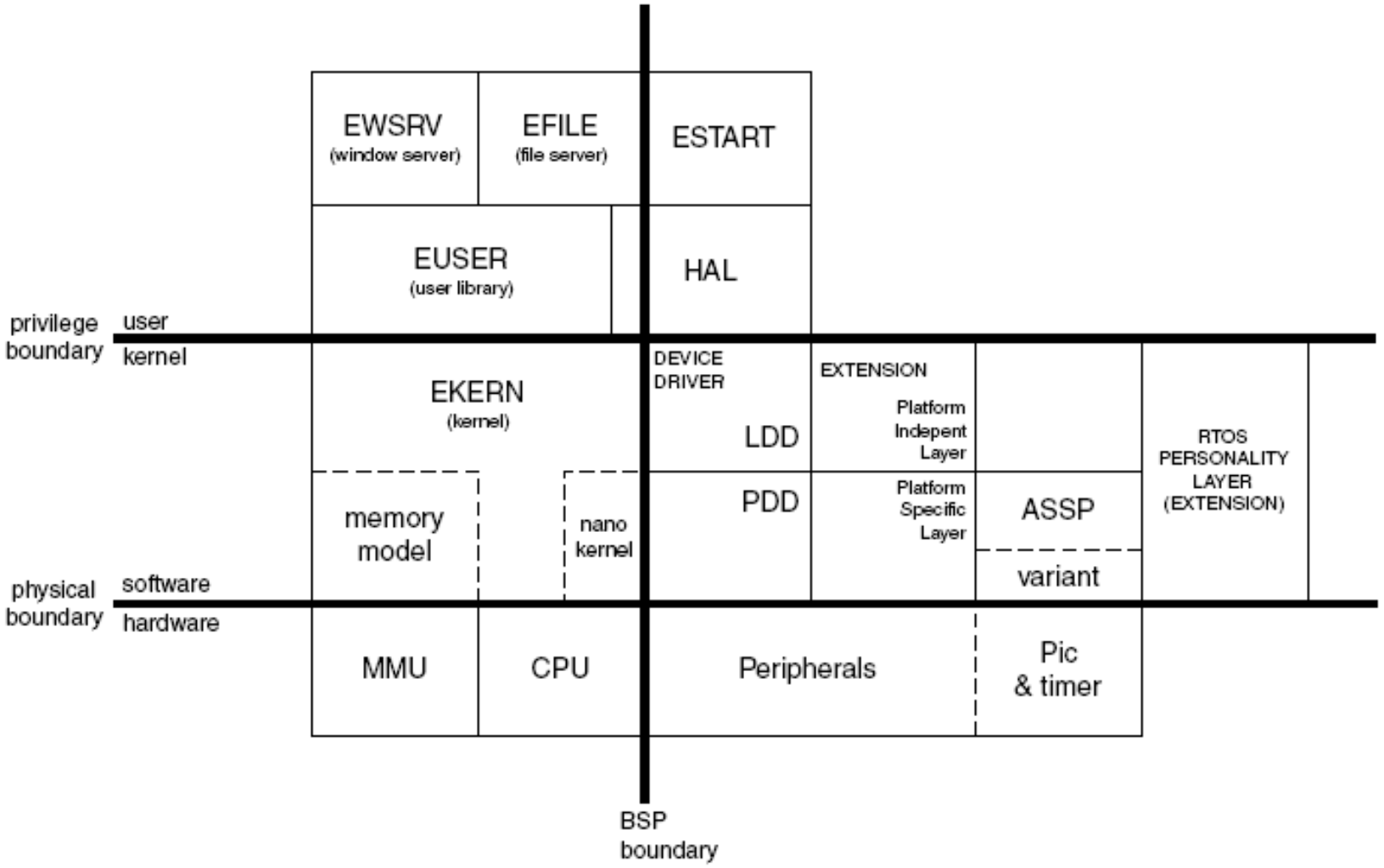
Micro-kernel Design

- Some issues
  - Microkernel uses message passing and performance suffers because of added overhead of communication between objects
  - Efficiency of functions that are outside of kernel diminishes
    - Since messages pass between user and kernel space objects, switches in privileges levels occur, complicating performance
    - Message passing and privilege switching implies that two or more address spaces must be used to implement microkernel service request
- Symbian OS puts emphasis on minimal, tightly focused servers

Symbian OS Nanokernel

- Design of Symbian OS separates
  - functions that require complicated implementation (kernel layer) from
  - most basic functions (nanokernel layer)
- Nanokernel - most basic (primitive) functions in Symbian OS
  - Scheduling and synchronization operations
  - Interrupt handling
  - Mutexes, semaphores
  - Most functions at this level are preemptible
- Kernel Layer –more complicated kernel functions
  - Complex object services
  - User-mode threads
  - Scheduling and context-switching
  - Dynamic memory
  - Complex synchronization
  - Object and inter-process communication

Microkernel Overview



Client-Server Resource Access

- Applications that need access to system resources are clients; servers are programs that OS runs to coordinate access to these resources
- Example: open file
  - Make connection to a file server
  - Server acknowledges the connection
  - Client requests 'open' request with the name of specific file
- This design
  - Protects resources
  - Is effective for managing multiple access to system resources
  - Each server is easily upgradeable and swapped out for new designs

Communication and Multimedia

- Pluggable messaging architecture
  - New message types can be invented and dynamically loaded by the messaging server
  - New transport methods can be introduced by implementing new object and loading into the kernel
- Multimedia devices and content are handled by special servers
  - Allows user implement modules that describe new and existing content and functions
  - Supports various forms of objects, designed to interact with each other

Processes and Threads (1)

- Multi-asking – Symbian OS favors threads and is built around thread concept
- Thread support is based in nanorkernel with nanothreads
- Nanokernel provides nanokernel scheduling, interthread synchronization and timing services
  - Nanothreads cannot run in user mode
  - Nanothread needs minimal set of data location of its stack, how big stack is
  - OS keeps control of everything else (e.g., code of each thread uses, stores thread's context on tis run-time stack)
  - Nanothread states: suspended, fast semaphore wait, DFC (Delayed Function Call) wait, sleep, other

Processes and Threads (2)

- Process
  - Processes are Symbian OS threads grouped together under single process control block structure with a single memory space
  - Scheduling a process means scheduling a thread and initializing the right PCB to use for its data needs
- Threads in one process work together, share scheduling parameters, share memory space objects, including device and object descriptors
- When a process terminates, all threads in process are terminated by kernel.

Active Objects (3)

- Active Objects - Specialized forms of threads
  - Introduced due to Symbian OS focussing on communication
    - Apps have similar pattern of implementation: they write data to socket or send data through pipe, and then block and wait for response from receiver
  - Active objects are designed so that when they are brought back from this blocked state, they have single entry point into their code that is called.
- Advantage of having this simplified thread model:
  - Scheduling – while waiting for events, all active objects reside within single process and can act as a single thread to the system
  - All active objects can be coordinated by a single scheduler implemented in a single thread
  - Active objects form efficient and lightweight version of standard threads
  -

Memory Management (1)

- Memory model restricted and does not use virtual memory/swap space model
- No virtual memory with demand paging
- Storage = memory, no disk drive
- Two types of memory
  - RAM and flash memory
  - RAM stores OS code
  - Flash memory used for operating memory and permanent (file) storage
  - Possible to add extra flash memory to device
    - Secure digital card – exclusively for permanent storage

Memory Management (2)

- Concepts
  - Paging, address translations, virtual/physical address abstraction
  - We have pages but pages cannot be swapped from memory to external storage
  - Abstraction of memory pages is used
  - Pages are replaced, but the page being replaced is just discarded
    - Only code pages can be replaced since on they are backed on the flash memory

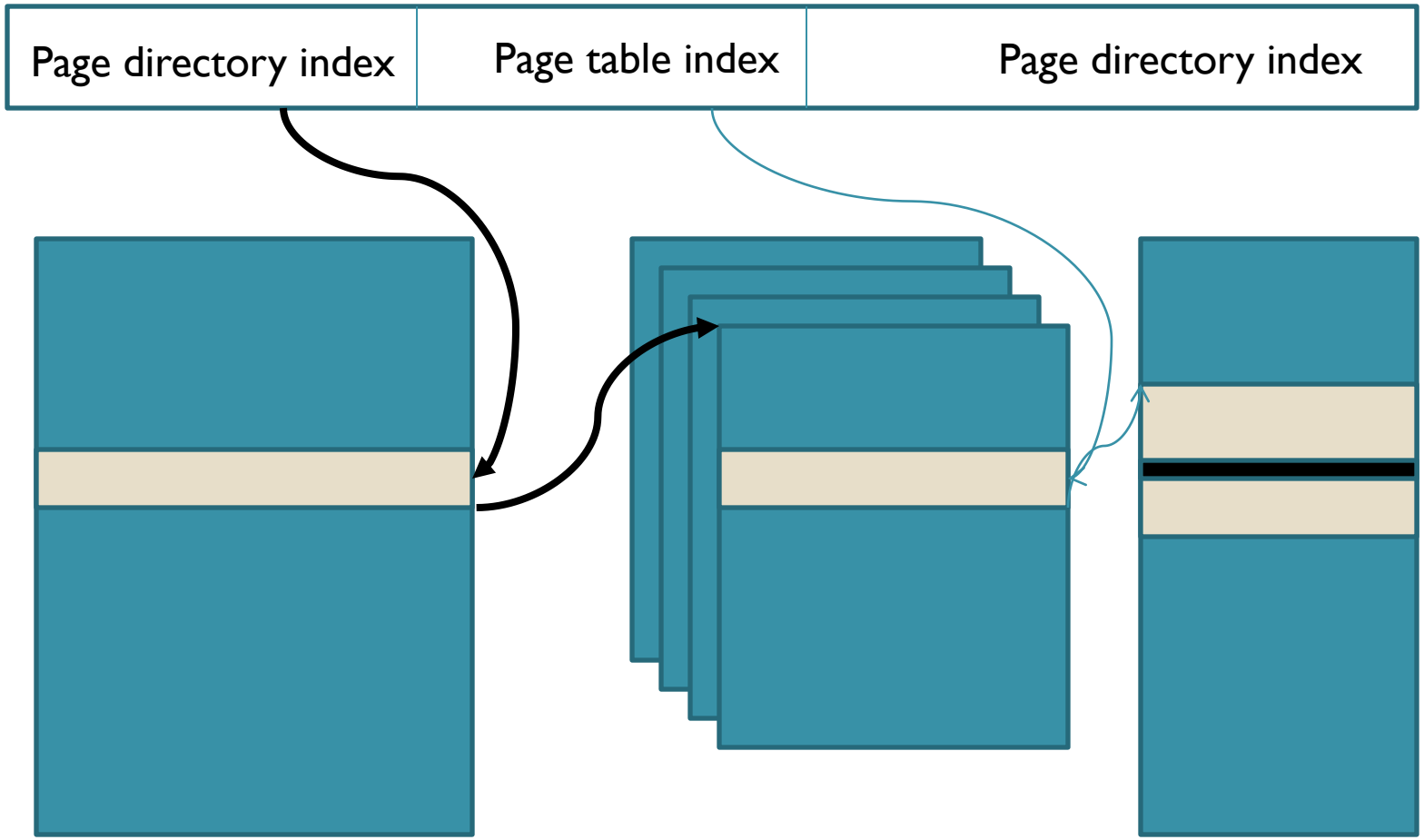
Memory Management (3)
(Tasks of Memory Management)

1. Management of application size
   - Application size needs to be small and object-oriented design
2. Heap management
   - Heap – space for dynamic memory allocation – must be managed very tightly
   - Heap space is bounded to force programmers to reclaim and reuse heap space
3. Execution in-place
   - Flash memory is mapped into virtual address space and programs can be executed directly from flash memory without copying them into RAM first
4. Loading DLLs
   - Loading all DLLs when application is first loaded into memory is more acceptable, but it is a choice. (users accept more DLL loading delays at the beginning when loading the app than during the app execution)
5. Offload memory management to hardware
   - If there is available MMU, use it – system performance is better

Memory Management (4)

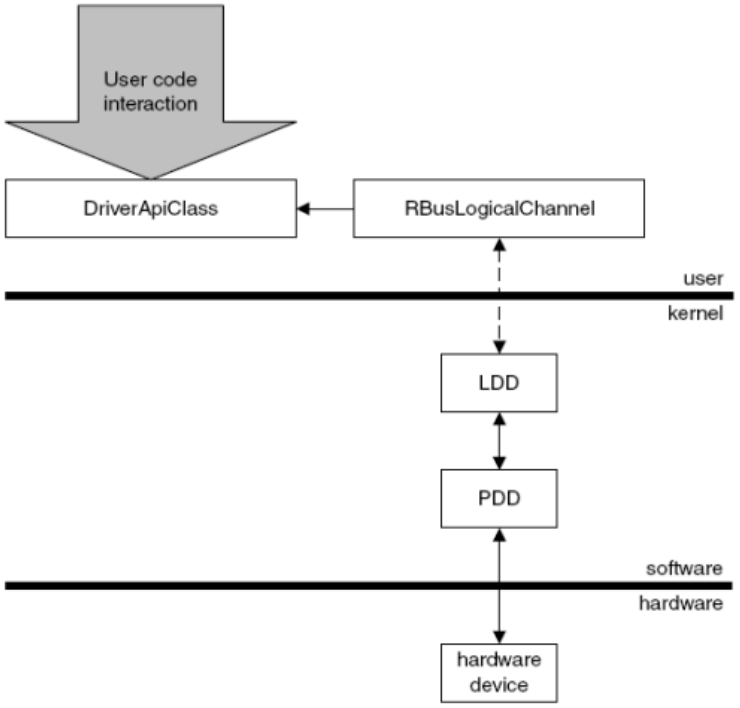
- Frame size 4KB
- Two-level page table strategy
- First level, called page directory
  - Kept in memory and is pointed to by TTBR (translation table base register)
  - Points to second level of page table
- Second level points to pages
- Hardware assists in virtual-to-physical memory address mapping calculation

Two-level Page Table



Input and Output (Device Driver)

- Drivers execute as kernel-privileged code
- Drivers split into two levels
  - Logical device driver (LDD) - interface to upper layers of software
  - Physical device driver (PDD) – interacts directly with hardware
- LDDs and PDDs can be dynamically loaded by user programs if they are not already existing in memory



I/O – Kernel Extensions

- Kernel extensions – device drivers that are loaded by Symbian OS at boot time
- Different from normal device drivers
- Most device drivers are implemented as LDDs paired with PDDs and loaded when needed by user-space apps.
- Kernel extensions are built into boot procedure
  - They implement special functions crucial to OS
    - DMA services, display management, bus control to peripheral devices (e.g., USB bus)

I/O – DMA

- Device drivers make use of DMA hardware
- DMA hardware consists of controller that controls set of DMA channels
- Each channel provides single direction of communication between memory and device
  - Bidirectional communication requires two DMA channels
  - At least one pair of DMA channels is dedicated to screen LCD controller
- DMA service, provided by DMA hardware, is used by PDD
- Between PDD and DMA controller, Symbian OS implements two layers of software
  - Software DMA layer
  - Kernel extension that interfaces with the DMA hardware

Storage Systems

- File Systems for Mobile devices
  - Arbitrary names of files
  - Hierarchical directory-based file system
  - Block sizes typically 512 bytes to 2048 bytes
  - *Flash memory cannot simply overwrite memory, it must first erase first, then write*
  - *Entire blocks must be erased*
  - *Erase time for flash memory are long*

File Systems

- Hence – specific flash file system design needed
  - Wear-Leveling
    - When flash store is to be updated, the FS will write a new copy of the changed data over to a fresh block, remap the file pointers, and then erase the old block later when it has time.
- Symbian OS uses FAT-16
- Symbian File server implementation is built much like Linux Virtual File System

Summary

- Symbian OS
  - Object-oriented OS for smart phone platforms
  - Microkernel design with nanokernel core
  - Many features of general purpose OS
  - But some specific features
    - Active objects make waiting on external events much more efficient
    - Lack of virtual memory makes memory management more challenging
    - Support of object orientation in device drivers uses two-layer abstract design

Ultra-Mobile Development Principles

- Responsiveness
- Power Management over Performance
  - Power limiting factor
- Tight Memory Management
- Flash memory limitations
  - Slow writes, limited writes
- Security and Privacy

References

- http://www.developer.nokia.com/Community/Wiki/Symbian_OS_Internals