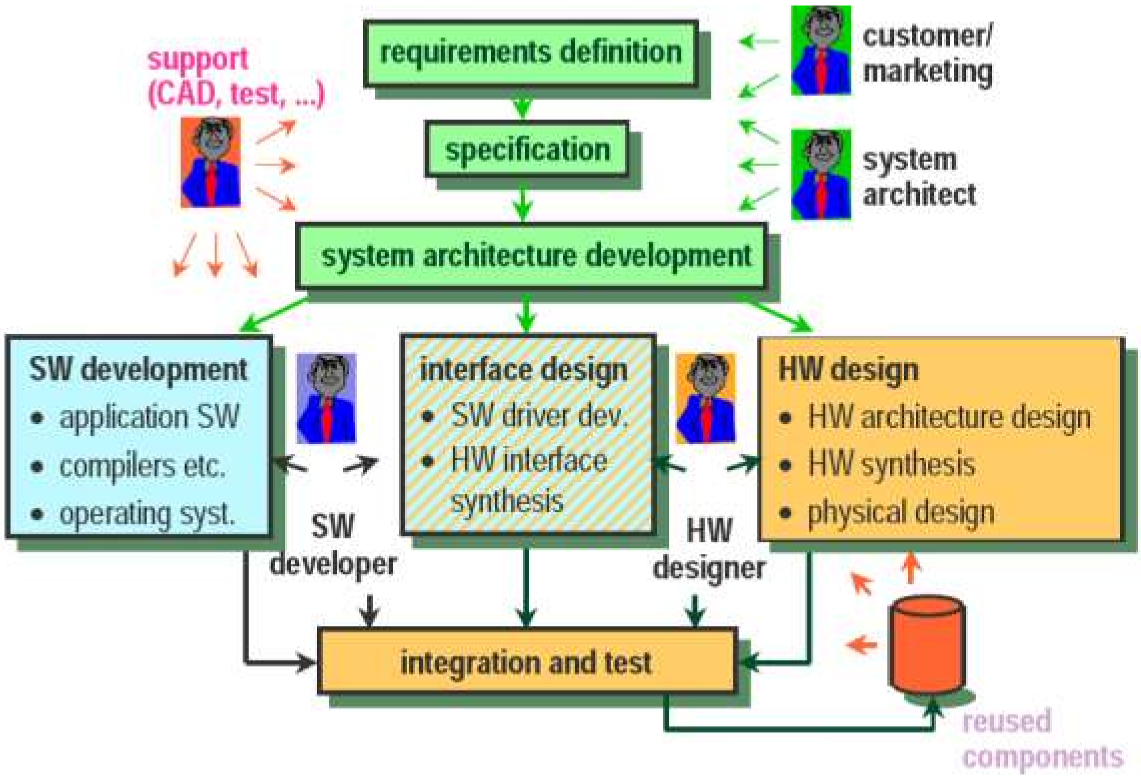
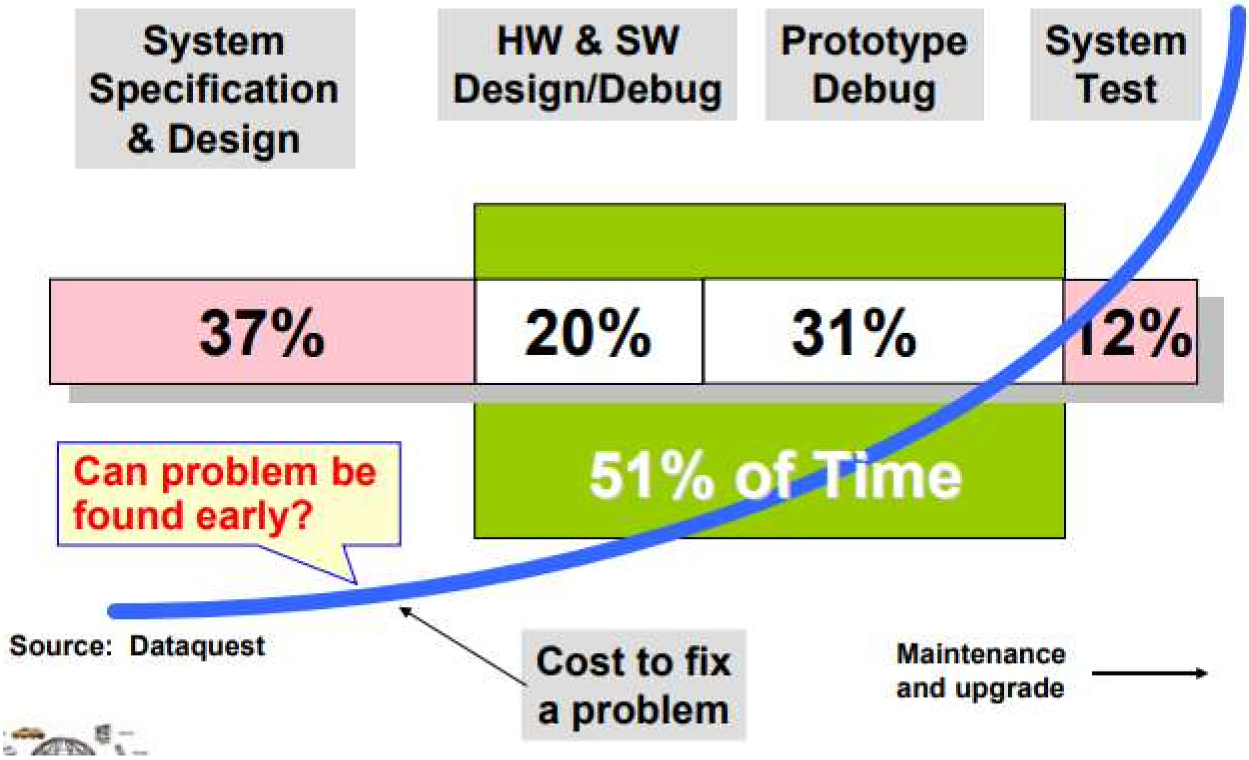
Embedded System

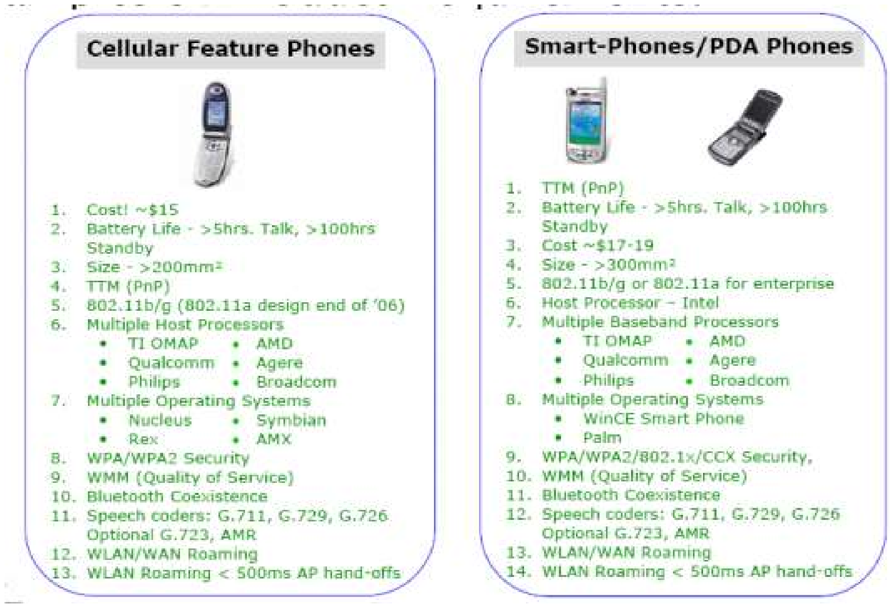
Lecture Note 2. Development

I. The Developing Process





這裡我們給一個 Requirement 例子



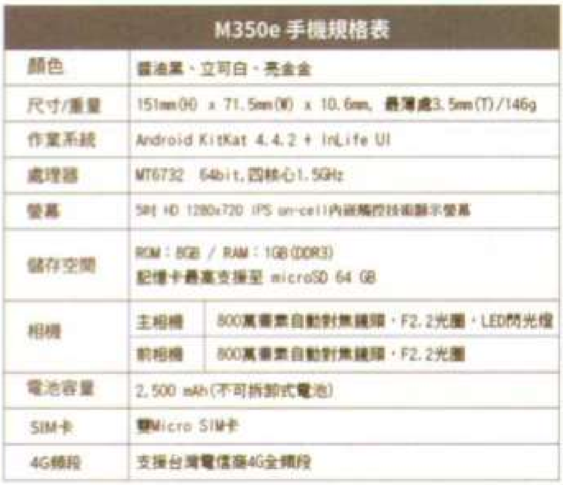
Cellular Feature Phones

1. Cost! ~$15
2. Battery Life - >5hrs. Talk, >100hrs Standby
3. Size - >200mm²
4. TTM (PnP)
5. 802.11b/g (802.11a design end of '06)
6. Multiple Host Processors
   - TI OMAP
   - AMD
   - Qualcomm
   - Agere
   - Philips
   - Broadcom
7. Multiple Operating Systems
   - Nucleus
   - Symbian
   - Rex
   - AMX
8. WPA/WPA2 Security
9. WMM (Quality of Service)
10. Bluetooth Coexistence
11. Speech coders: G.711, G.729, G.726 Optional G.723, AMR
12. WLAN/WAN Roaming
13. WLAN Roaming < 500ms AP hand-offs

Smart-Phones/PDA Phones

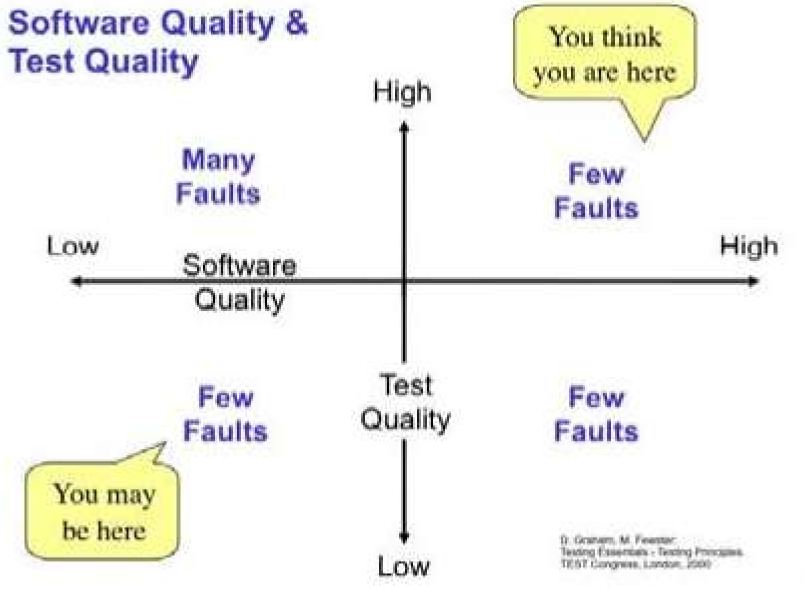
1. TTM (PnP)
2. Battery Life - >5hrs. Talk, >100hrs Standby
3. Cost ~$17-19
4. Size - >300mm²
5. 802.11b/g or 802.11a for enterprise
6. Host Processor – Intel
7. Multiple Baseband Processors
   - TI OMAP
   - AMD
   - Qualcomm
   - Agere
   - Philips
   - Broadcom
8. Multiple Operating Systems
   - WinCE Smart Phone
   - Palm
9. WPA/WPA2/802.1x/CCX Security,
10. WMM (Quality of Service)
11. Bluetooth Coexistence
12. Speech coders: G.711, G.729, G.726 Optional G.723, AMR
13. WLAN/WAN Roaming
14. WLAN Roaming < 500ms AP hand-offs

Spec 例子



M350e 手機規格表		
顏色	醬油黑・立可白・亮金金	
尺寸/重量	151mm(H) x 71.5mm(W) x 10.6mm, 最薄處3.5mm(T)/146g	
作業系統	Android KitKat 4.4.2 + InLife UI	
處理器	MT6732 64bit,四核心1.5GHz	
螢幕	5吋 HD 1280x720 IPS on-cell內嵌觸控技術顯示螢幕	
儲存空間	ROM：8GB / RAM：1GB(DDR3) 記憶卡最高支援至 microSD 64 GB	
相機	主相機	800萬畫素自動對焦鏡頭・F2.2光圈・LED閃光燈
	前相機	800萬畫素自動對焦鏡頭・F2.2光圈
電池容量	2,500 mAh(不可拆卸式電池)	
SIM卡	雙Micro SIM卡	
4G頻段	支援台灣電信商4G全頻段	

在 Testing&Debugging 時還有一個需要注意的點是

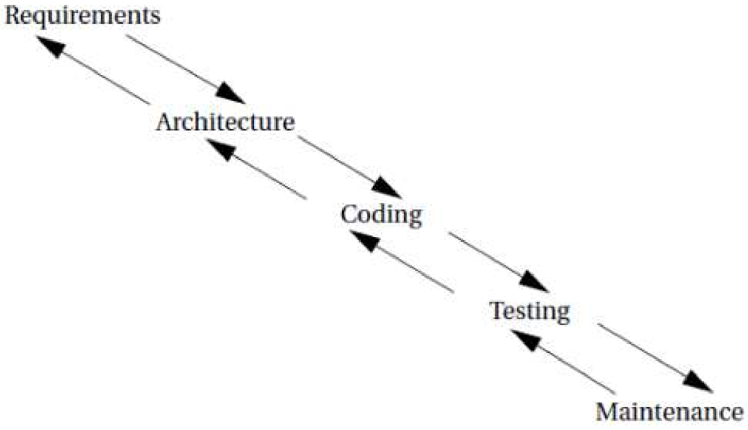


D. Graham, M. Fewster: Testing Essentials - Testing Principles. TEST Congress, London, 2000

II. Embedded System Design

Common models of software development：

1. Waterfall model (Plan-driven)



缺點非常明顯，瀑布模型太過理想化

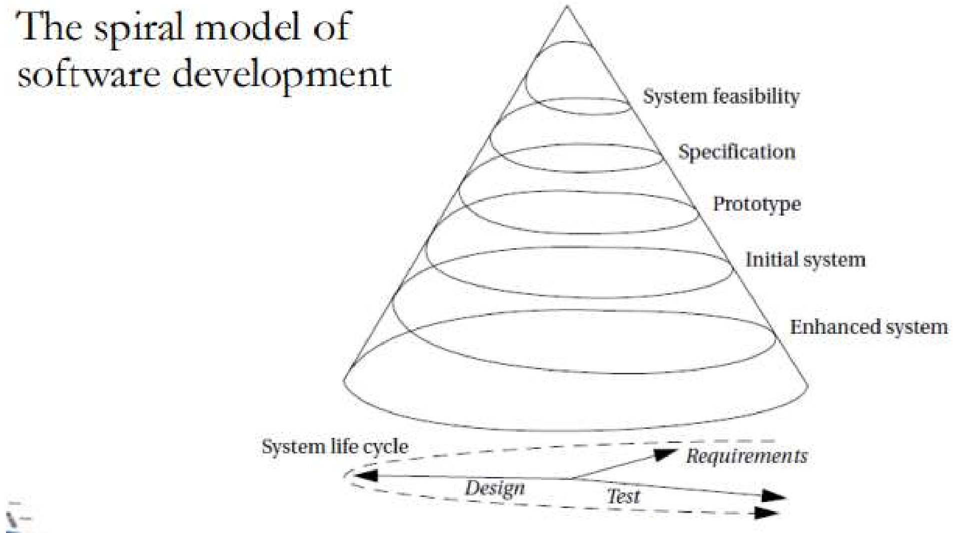
(1) 各個階段的劃分完全固定，階段之間產生大量的文檔，極大地增加了工作量。

(2) 由於開發模型是線性的，用戶只有等到整個過程的末期才能見到開發成果，從而增加了開發風險。

(3) 通過過多的強制完成日期和里程碑來跟蹤各個項目階段。

(4) 瀑布模型的突出缺點是不適應用戶需求的變化。

2. Spiral model (Risk-driven)



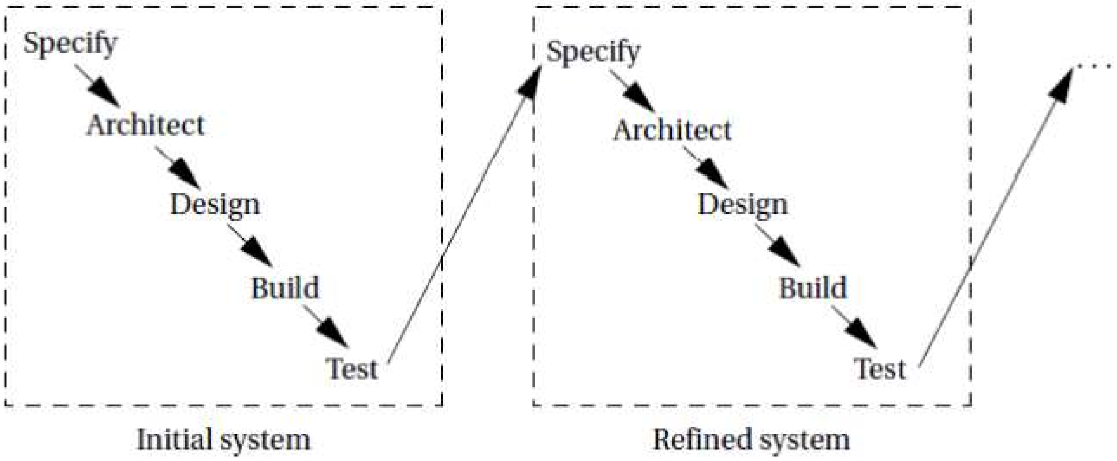
雖然在迭代的階段植入了軟體測試，因此可以很自由的變化，但還有有些缺點的

(1) 過分依賴風險分析經驗與技術，一旦在風險分析過程中出現偏差將造成重大損失

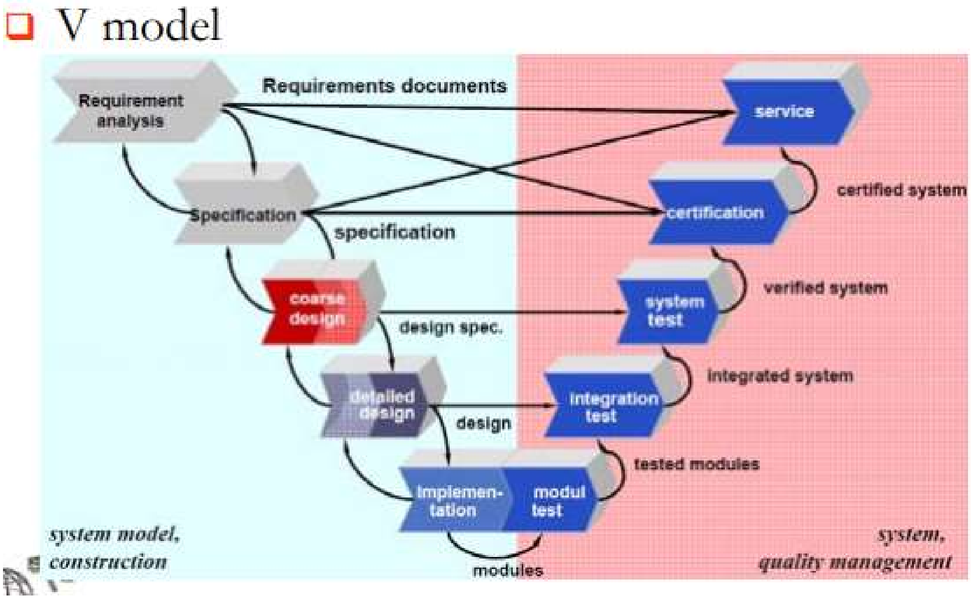
(2) 過於靈活的開發過程不利於已經簽署合同的客戶與開發者之間的協調

(3) 由於只適用大型軟體，過大的風險管理支出會影響客戶的最終收益

3. Successive refinement model



4. V model

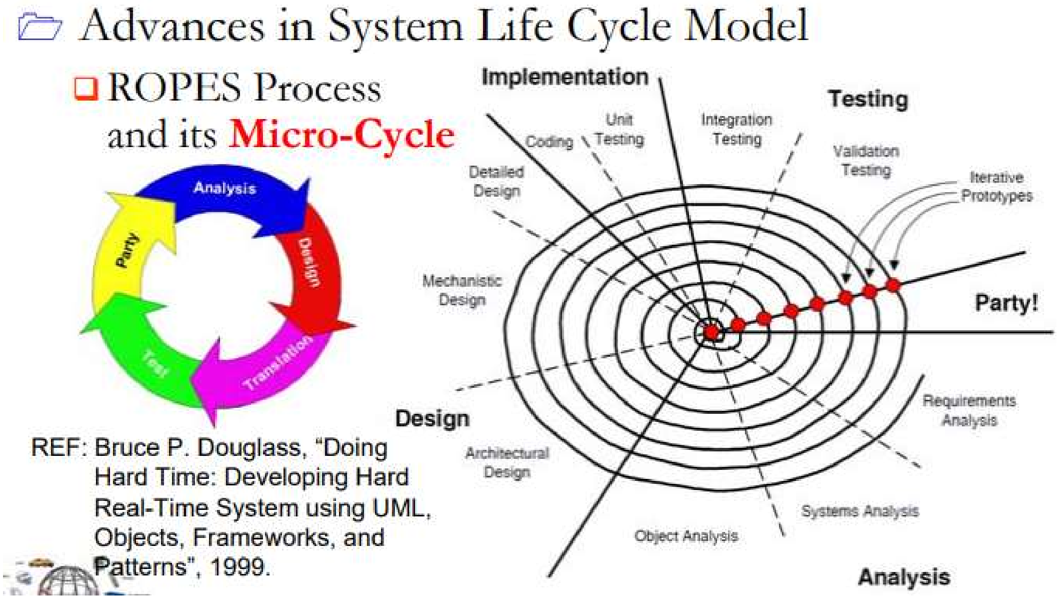


❑ V model

V 模型的左側是需求的分解，並且產生系統的規格，V 模型的右側是各部份的整合以及確認，確認可以說是在問「做的是正確的東西嗎？」，而驗證可以說是在問「做的方式正確嗎？」，**V 模型是瀑布模型的變種，瀑布模型存在的問題 V 模型也存在**。

5. ROPES

Object-oriented design is usually part of an overall development process. Here's a development process called ROPES (Rapid Object-Oriented Process for Embedded Systems)



Advances in System Life Cycle Model

❑ ROPES Process and its Micro-Cycle

REF: Bruce P. Douglass, "Doing Hard Time: Developing Hard Real-Time System using UML, Objects, Frameworks, and Patterns", 1999.

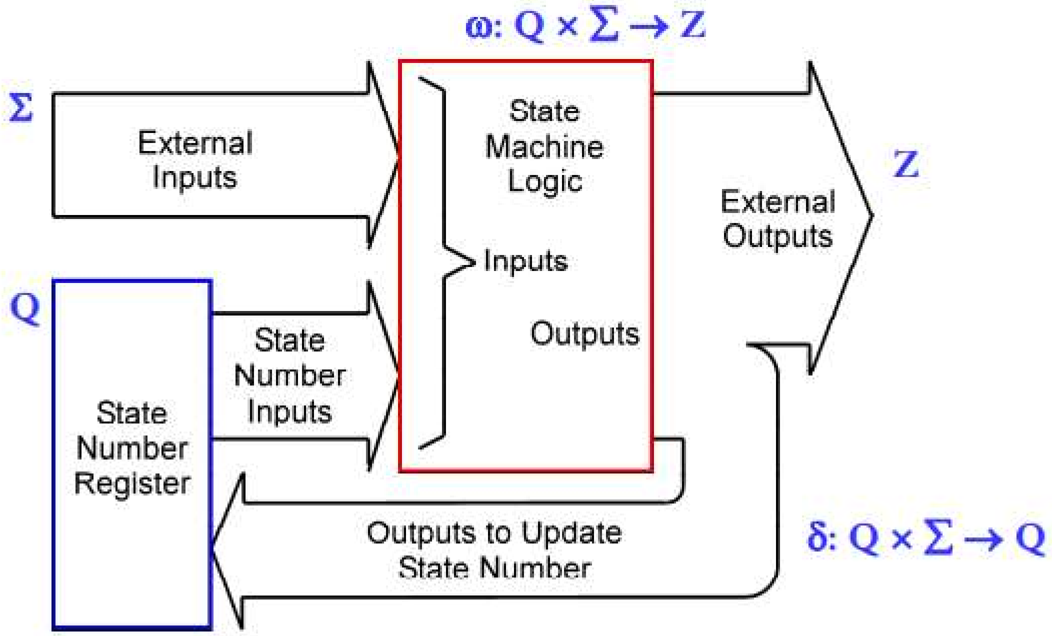
Systems engineering identifies a high-level subsystem or component architecture and decomposes the system-level use cases to subsystem-level use cases that map to individual subsystems.

III. System Design with State Machine

1. 定義

State Diagram: $SM = (Q, \Sigma, Z, \omega, \delta, q_0, F)$

- □ Representation describing finite state machines
- □ Components:
  - ☑ Q – a finite set of states
  - ☑ Σ – a finite collection of input symbols (alphabet)
  - ☑ Z – a finite collection of output symbols
  - ☑ ω – output function representing $Q \times \Sigma \rightarrow Z$
  - ☑ δ – representation of transitions: $Q \times \Sigma \rightarrow Q$
  - ☑ q_0 – the start state ($q_0 \in Q$)
  - ☑ F – a finite set of accepting states ($F \subseteq Q$)



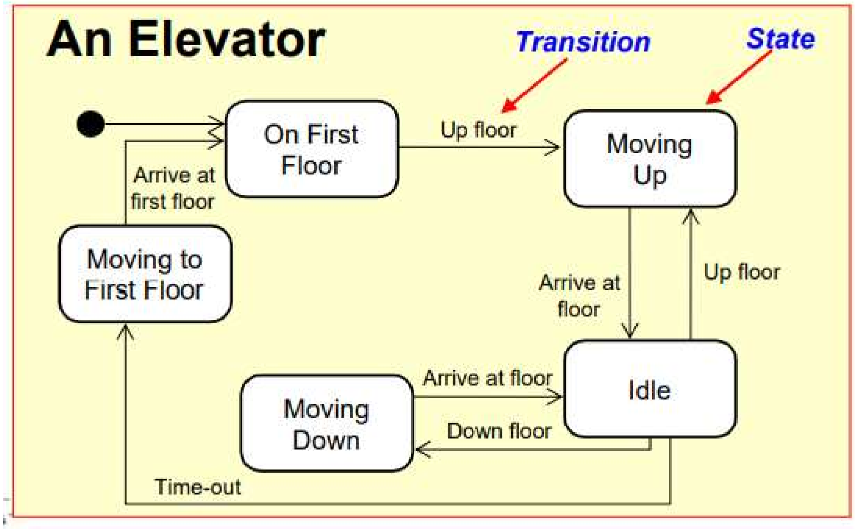
(1) A way to represent the specification of a system :

what the system **must do** (and **must not do**)

(2) A way to check whether the system satisfies its specification in its operating environment

(3) A way to model the behaviors of a system

這裡我們舉個簡單例子

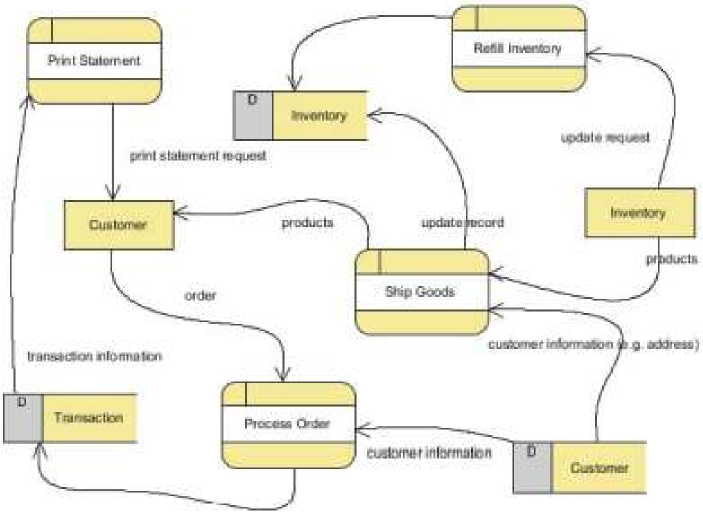


當然還有更多方式描述嵌入式系統設計

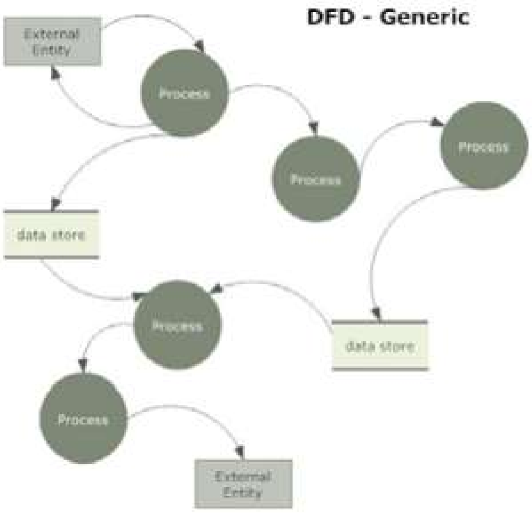
- UML Diagram
- Block diagram
- Data flow diagram (DFD)
- State diagram (Statechart)
- Flow chart
- Pseudo-code
- Actual program code
- PCB layout
- Circuit diagram
- …

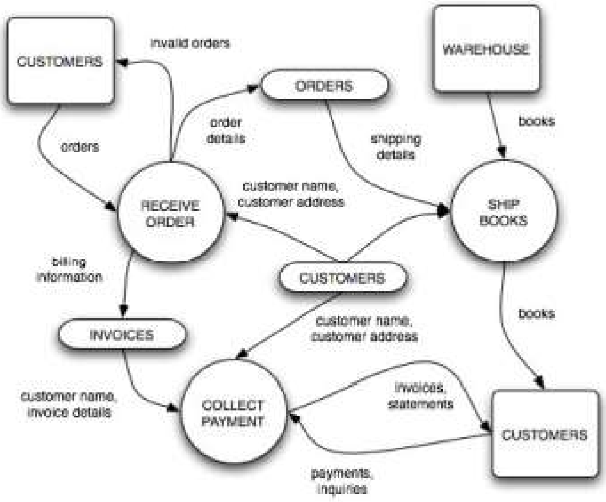
Data Flow Diagram (DFD)

Notation	Description
Process	Process
D DataStore	Data store
Entity	External entity
→	Data flow
↔	Bidirectional data flow

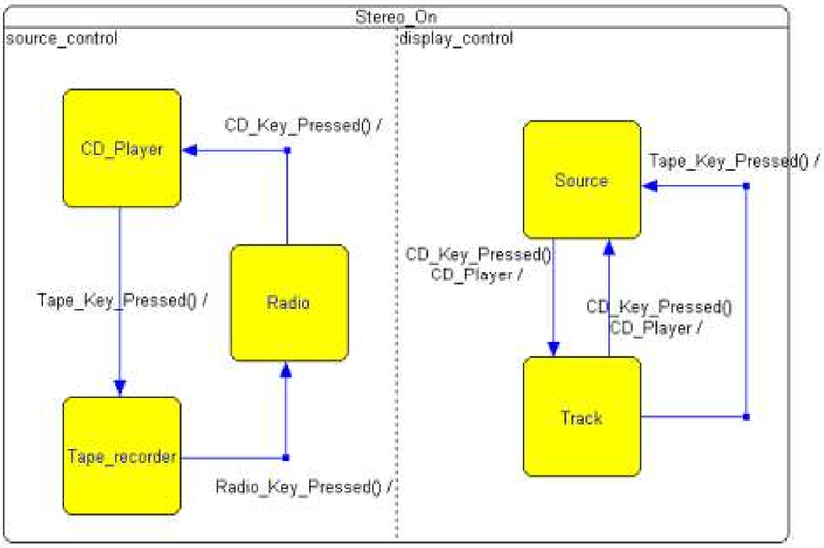


🗁 Example of Data Flow Diagram (DFD)





🗁 Example of State Diagram – CD/Stereo



IV. UML

UML 主要 Focus 在 Object-Oriented Concept and Modeling，這裡有 3 個門派

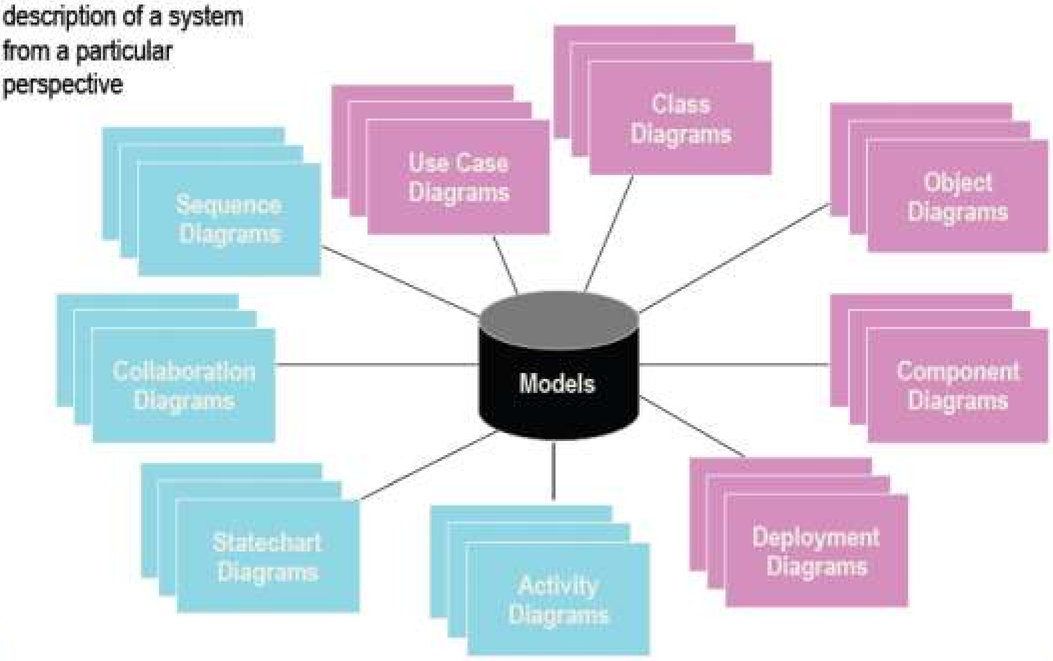
❖ The primary leaders of the effort were

☑ Grady Booch (Booch method)

☑ James Rumbaugh (OMT)

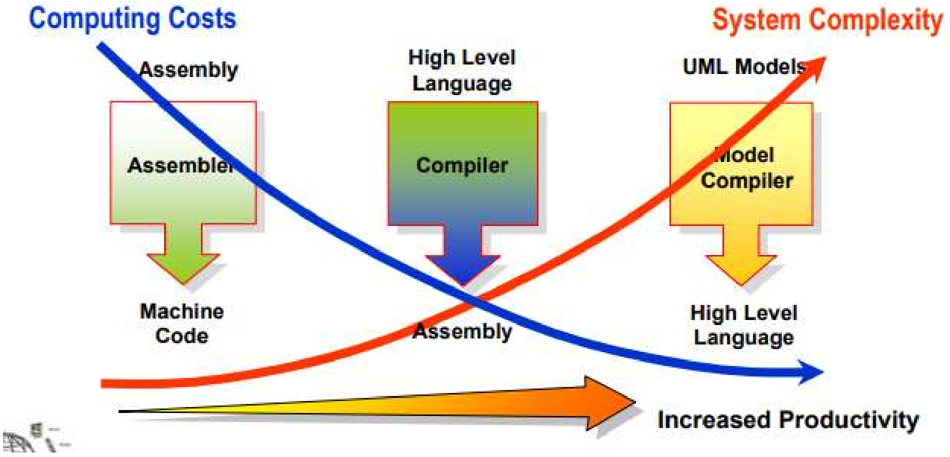
☑ Ivar Jacobson (OOSE)

A *model* is a complete description of a system from a particular perspective



很有趣的一張圖，可以看出 Model Compiler 的強大

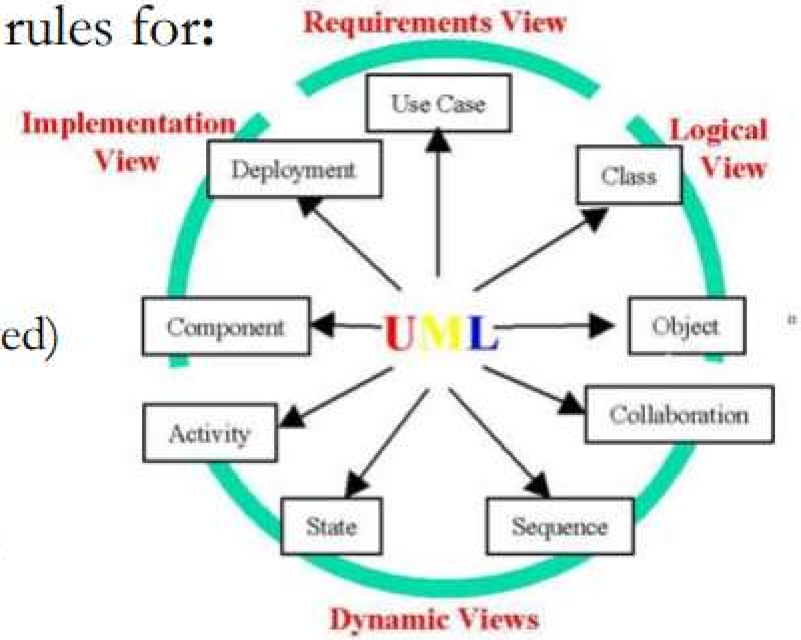
❑ Evolution of software development:



1. Structure

UML = UML Syntax + UML Semantics

- UML specifies rules for:
  - naming
  - scoping
  - visibility
  - integrity
  - execution (limited)
- Views:
  - Static views
  - Dynamic views



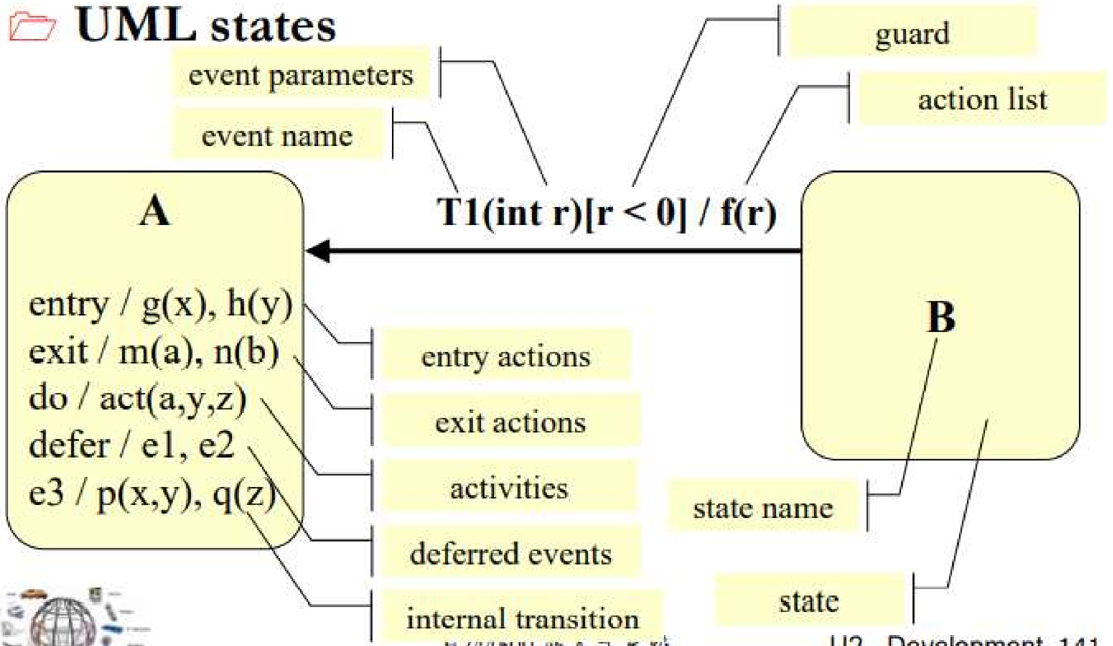
2021/04/06

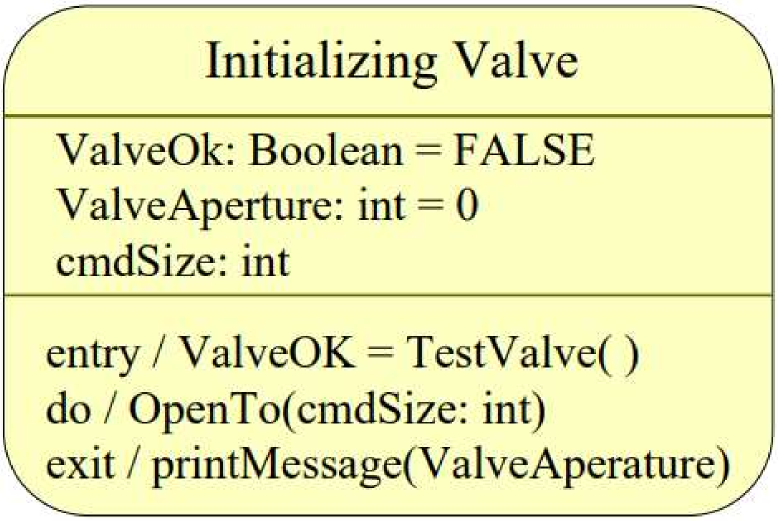
UML standard diagrams:

- Static Views
  - ✓ use case
  - ✓ class
  - ✓ object
  - ✓ component
  - ✓ deployment
- Dynamic Views
  - ✓ sequence
  - ✓ collaboration
  - ✓ startchart
  - ✓ activity

2. UML States

UML states



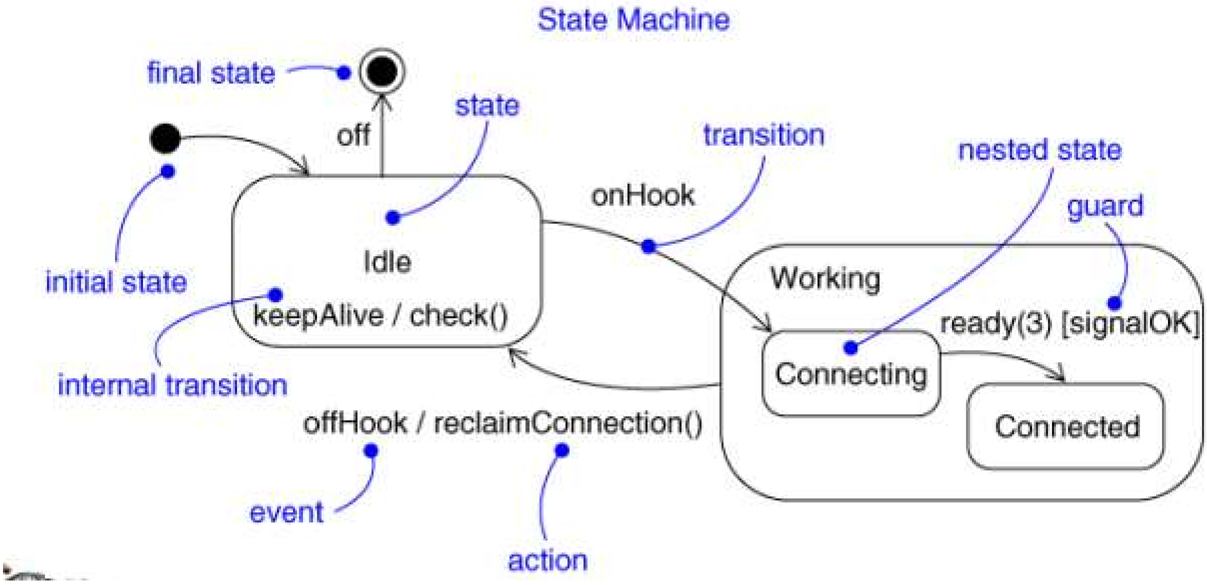


Initializing Valve

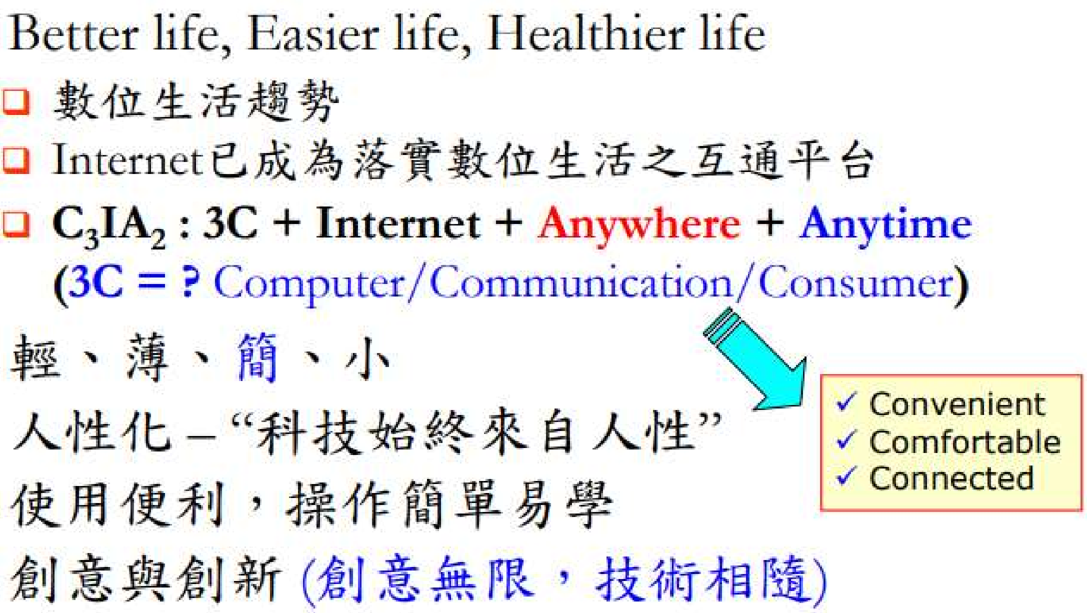
ValveOk: Boolean = FALSE
ValveAperture: int = 0
cmdSize: int

entry / ValveOK = TestValve( )
do / OpenTo(cmdSize: int)
exit / printMessage(ValveAperature)

- ❑ UML state diagram



V. 消費性嵌入式系統產品設計考量



Better life, Easier life, Healthier life

- □ 數位生活趨勢
- □ Internet已成為落實數位生活之互通平台
- □ C_3IA_2 : 3C + Internet + Anywhere + Anytime
  (3C = ? Computer/Communication/Consumer)

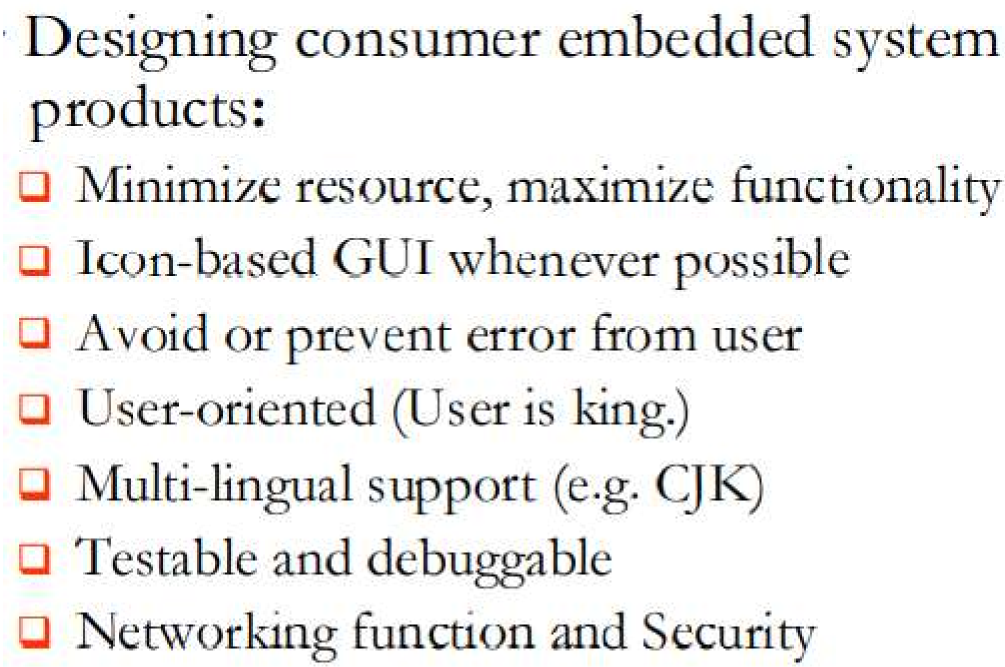
輕、薄、簡、小

人性化 – “科技始終來自人性”

使用便利，操作簡單易學

創意與創新 (創意無限，技術相隨)

✓ Convenient
✓ Comfortable
✓ Connected



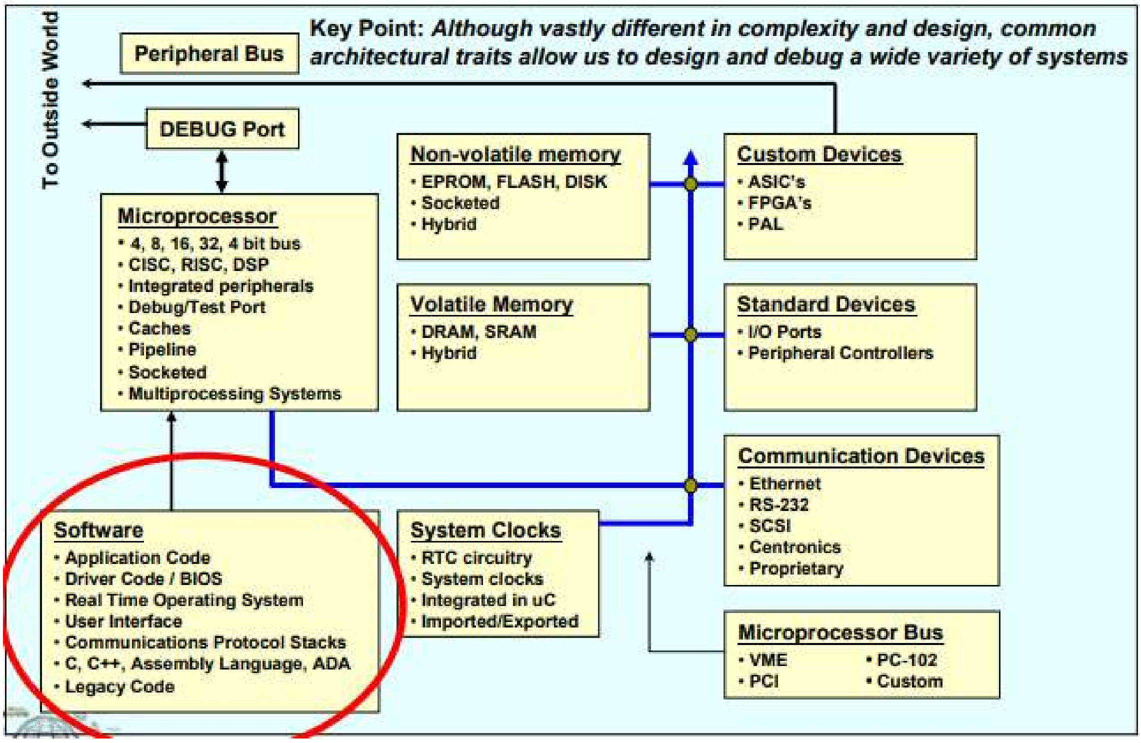
Designing consumer embedded system products:

- □ Minimize resource, maximize functionality
- □ Icon-based GUI whenever possible
- □ Avoid or prevent error from user
- □ User-oriented (User is king.)
- □ Multi-lingual support (e.g. CJK)
- □ Testable and debuggable
- □ Networking function and Security

VI. 嵌入式軟體

1. Embedded Software is
賦予系統晶片(半導體)生命力的要角，創造附加價值並提供硬體產品多元化之應用

2. 嵌入式系統產品所執行之軟體
(1) 控制、賦予產品智慧並內建於微電子產品中作為不可分離之元件的軟體
(2) 儲存於 ROM, Flash Memory 等非揮發性記憶體中的軟體程式，專司硬體驅動、控制與操作介面等處理功能



Key Point: *Although vastly different in complexity and design, common architectural traits allow us to design and debug a wide variety of systems*

To Outside World

Peripheral Bus

DEBUG Port

Microprocessor
- 4, 8, 16, 32, 4 bit bus
- CISC, RISC, DSP
- Integrated peripherals
- Debug/Test Port
- Caches
- Pipeline
- Socketed
- Multiprocessing Systems

Non-volatile memory
- EPROM, FLASH, DISK
- Socketed
- Hybrid

Volatile Memory
- DRAM, SRAM
- Hybrid

Custom Devices
- ASIC's
- FPGA's
- PAL

Standard Devices
- I/O Ports
- Peripheral Controllers

Communication Devices
- Ethernet
- RS-232
- SCSI
- Centronics
- Proprietary

Software
- Application Code
- Driver Code / BIOS
- Real Time Operating System
- User Interface
- Communications Protocol Stacks
- C, C++, Assembly Language, ADA
- Legacy Code

System Clocks
- RTC circuitry
- System clocks
- Integrated in uC
- Imported/Exported

Microprocessor Bus
- VME
- PCI
- PC-102
- Custom

2. Basic program structure

(1) Start-up and Initialization

有沒有 OS 差很多, Boot Code 的部分要考慮

(2) Running/Executing functionality

Perform pre-defined operations

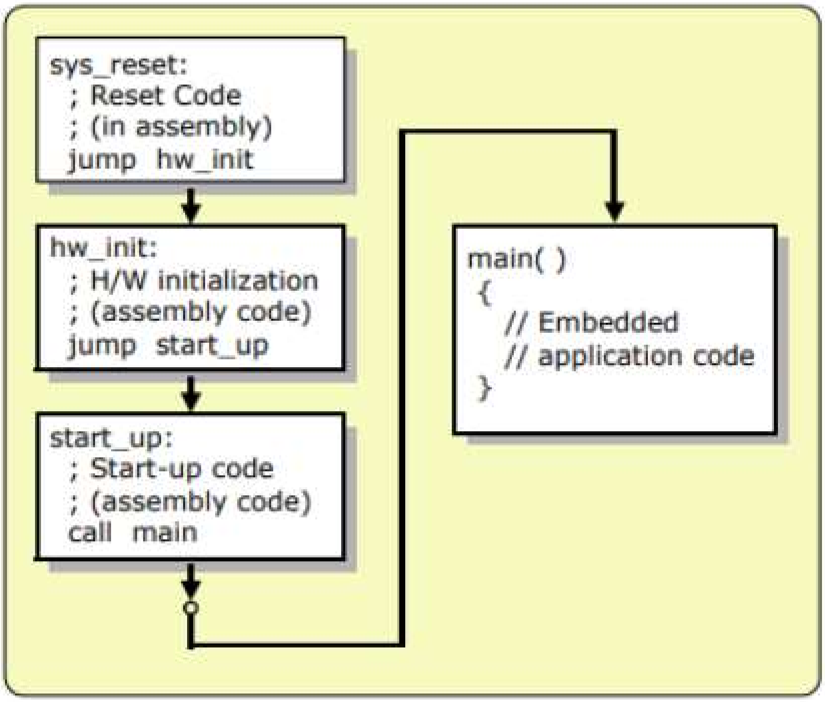
Processing user requests

(3) Shutdown (optional)

Save status and configuration

Stop running/Halt/Power-Off

Boot code of starting up an embedded system :



```
sys_reset:
  ; Reset Code
  ; (in assembly)
  jump  hw_init
```

```
hw_init:
  ; H/W initialization
  ; (assembly code)
  jump  start_up
```

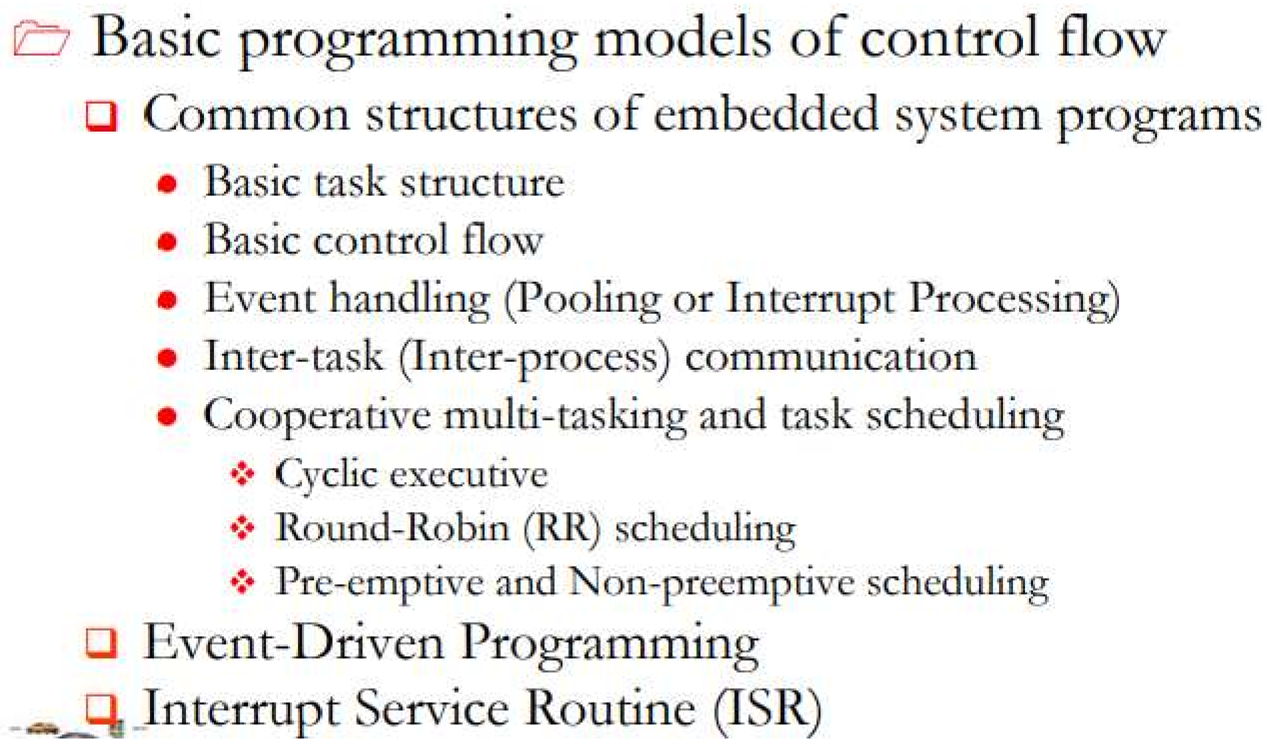
```
start_up:
  ; Start-up code
  ; (assembly code)
  call  main
```

```
main( )
 {
   // Embedded
   // application code
 }
```

以上, 若沒有 OS 幫助, 前三個函數方塊需要考慮

另外可以看見前兩個方塊使用到 jump, start_up 的部分才是 call main function

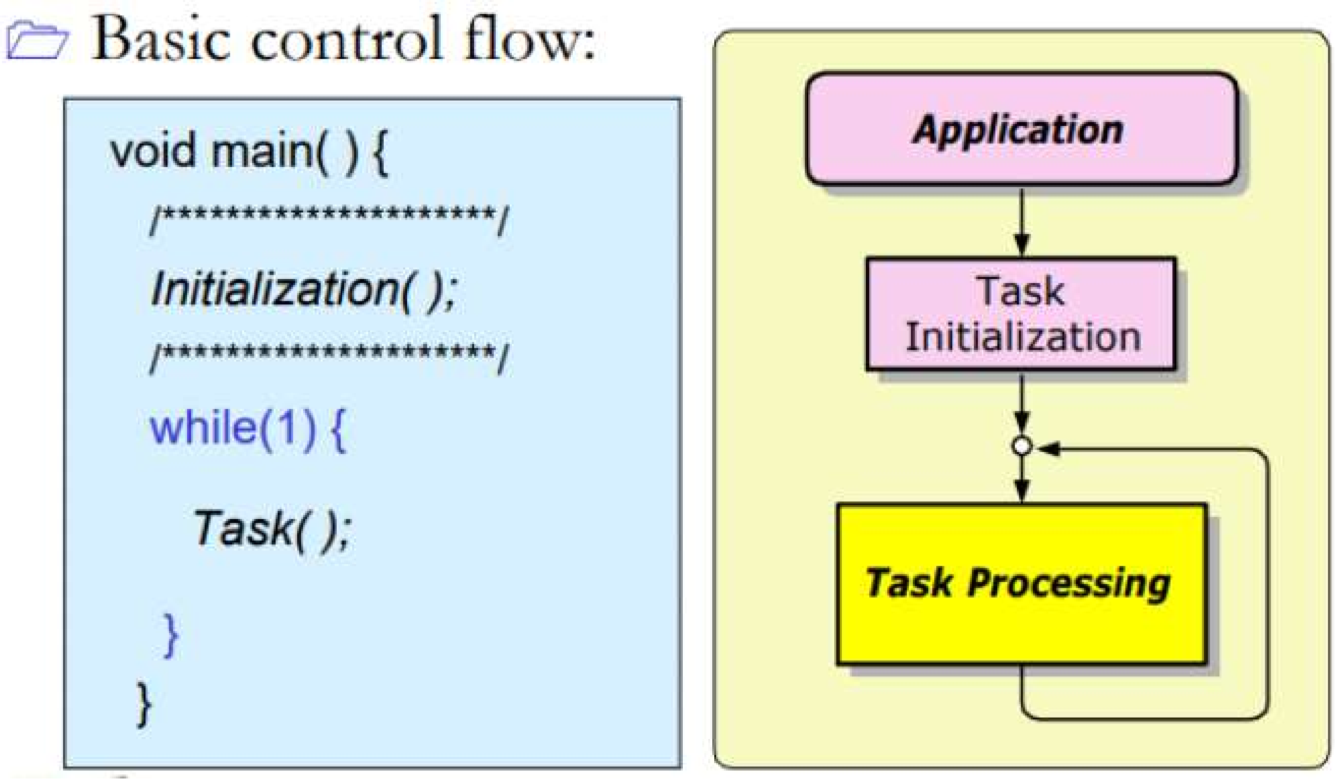
3. Basic Task Structures



Basic programming models of control flow

- Common structures of embedded system programs
    - Basic task structure
    - Basic control flow
    - Event handling (Pooling or Interrupt Processing)
    - Inter-task (Inter-process) communication
    - Cooperative multi-tasking and task scheduling
        - Cyclic executive
        - Round-Robin (RR) scheduling
        - Pre-emptive and Non-preemptive scheduling
- Event-Driven Programming
- Interrupt Service Routine (ISR)

(1) Free-Running Task



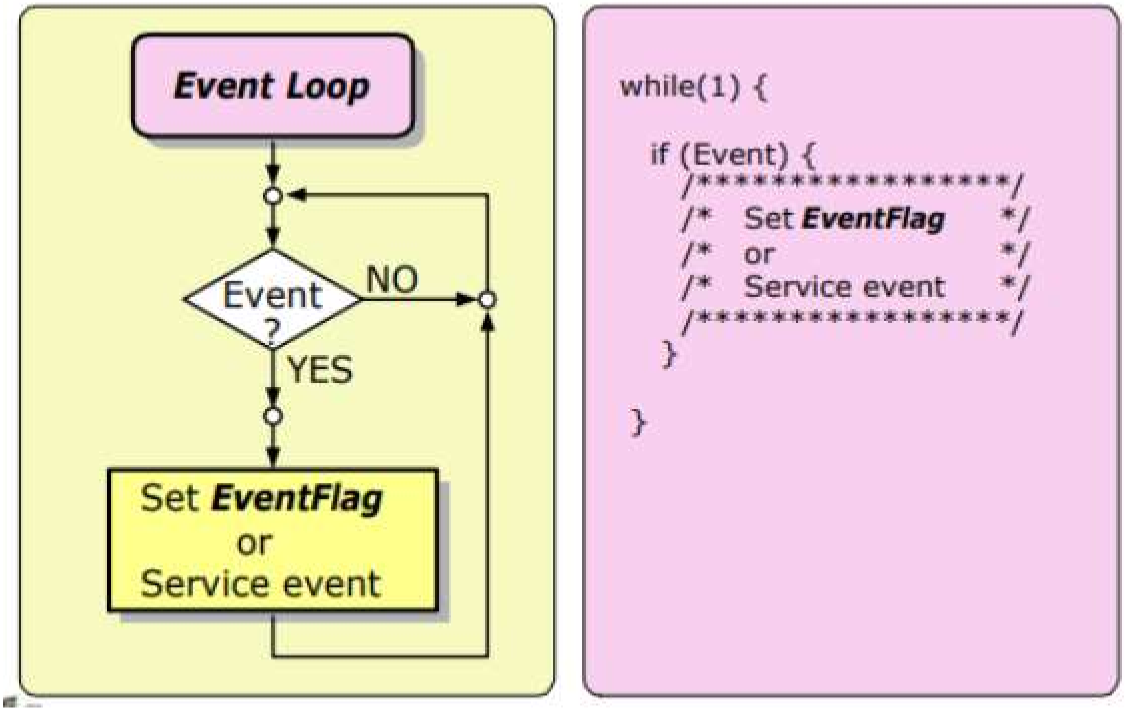
Basic control flow:

```
void main( ) {
   /*********************/
   Initialization( );
   /*********************/
   while(1) {

      Task( );

   }
}
```

(2) Polling Processing



```
while(1) {

  if (Event) {
    /*****************/
    /*  Set EventFlag  */
    /*  or             */
    /*  Service event  */
    /*****************/
   }

 }
```