下一站：移動開發

擴展Android框架功能和底層模塊的技術要點

QCon
北京2011大会

By 高煥堂

大綱

1. 簡介：彈性面貌下的經濟關聯
2. 觀點：如何看待Android
3. 增值：差異化的動能
4. API ：保護增值利益
5. 途徑：讓魔豆冒出頭來
6. 搭橋：以Core Service銜接
   HAL驅動與框架API
7. 結語：推動技術持續提升

一、簡介：Android 彈性面貌下的經濟關聯

Android 的面貌

- 至今全球已有超過170種的Android設備上市了，Android具有高度開源、開放和擴展性。
- 未來Android開發者將會面對更多的Android底層模組和上層框架功能的擴展的需求。

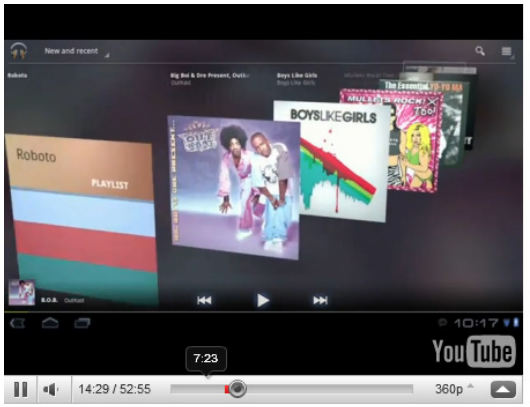
面貌下的經濟關聯

- 愈是開放的平台，愈容易實現底層模塊創新和框架擴展。
- 愈多選擇途徑，反而愈難決定出一條獲利的成功之路。
- 差異化的增值獲利途徑，是推動Android擴展和技術提升的關鍵環節。

二、觀點：如何看待Android?

開源 & 開放

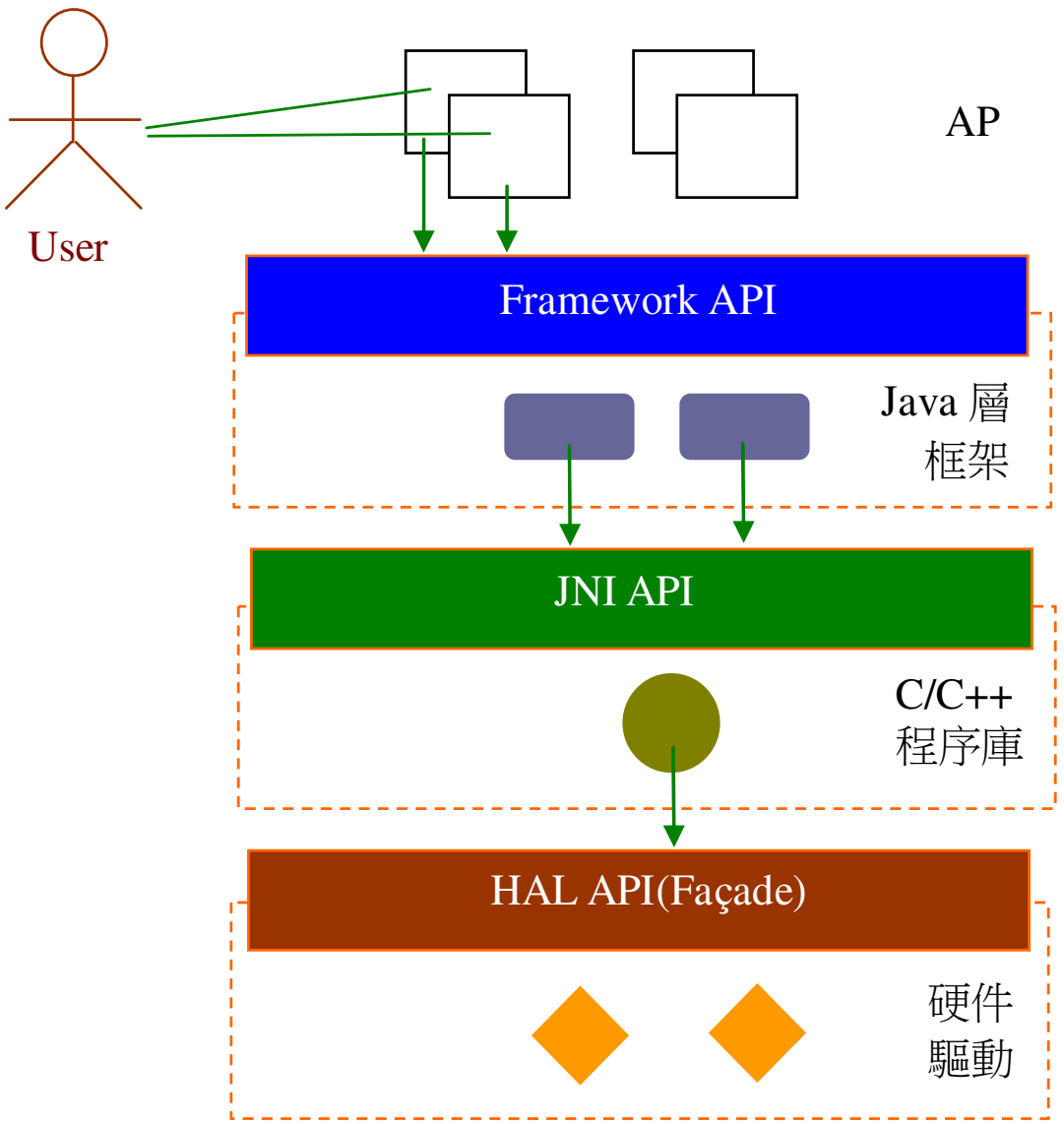
- Android本身是開源(Open Source)的。
- 在Android上開發，不需開源，包括
  - -- 底層驅動(Driver)模塊，
  - -- 層本地(Native)服務，
  - -- 上層應用(AP)框架和程序。



開放的框架


Java
APPLICATIONS
Alarm
Dialer
SMS/MMS
IM
Browser
Camera
Alarm
Home
Contacts
Voice Dial
Email
Calendar
Media Player
Albums
Clock
...
APPLICATION FRAMEWORK
Activity Manager
Window Manager
Content Provider
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
...
C/C++
LIBRARIES
Surface Manager
Media Framework
SQLite
OpenGL|ES
FreeType
WebKit
SGL
SSL
Libc
ANDROID RUNTIME
Core Libraries
Dalvik Virtual Machine
HARDWARE ABSTRACTION LAYER
Graphics
Audio
Camera
Bluetooth
GPS
Radio(RIL)
WiFi
...
Kernel
LINUX KERNEL
Display Driver
Camera Driver
Bluetooth Driver
Shared Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Driver
Power Management

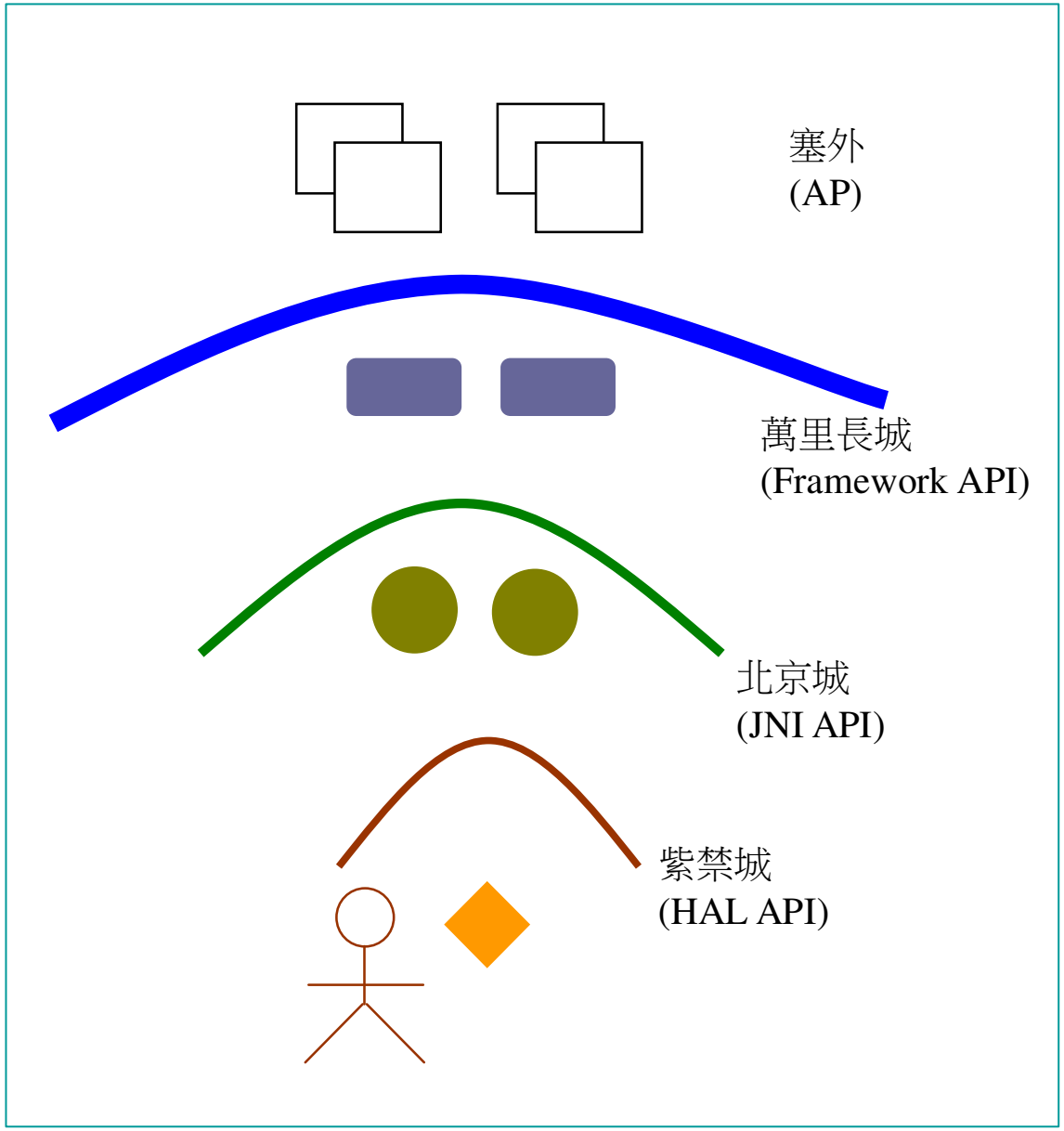
以傳統觀點看 Android 框架



此觀點的陷阱

- 底層提供服務給上層調用。
- User希望AP穩定不變。
- AP希望Java框架不變。
- Java框架期待C/C++模塊不變。
- C/C++模塊期待驅動穩定不變。
- 人人都期待底下的“平台”不變。

對框架的新潮觀點

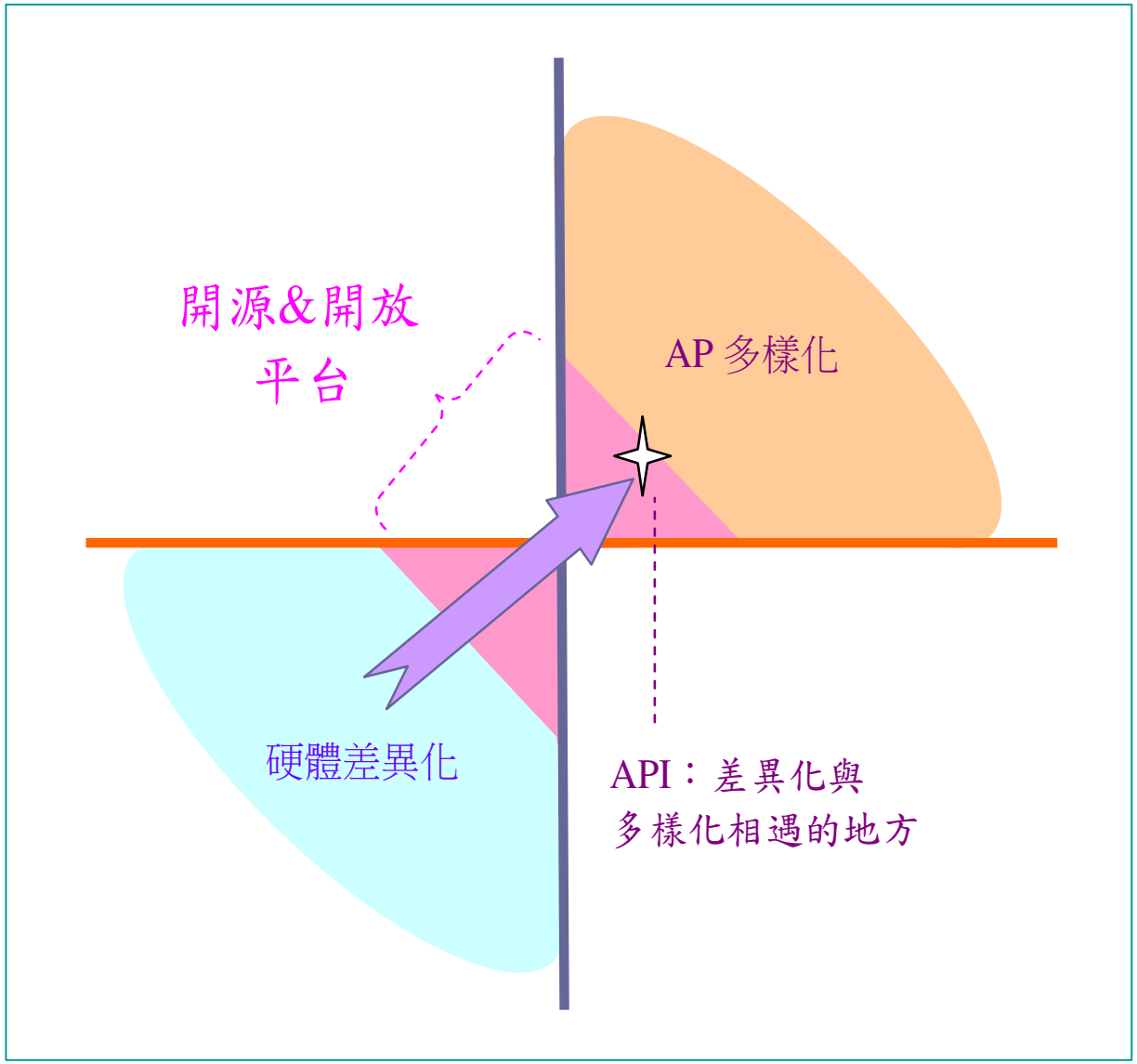


兩種觀點的差異

- **地位尊卑順序相反。**
- **行為決策相反。**老觀點，人人爭先恐後做 AP。反之，爭先恐後做框架和API。
- **底層先獲利。**萬里長城讓關內居民先獲利。所以HTC專注於開發底層硬件和驅動(關內部分)，成為Android手機大贏家。

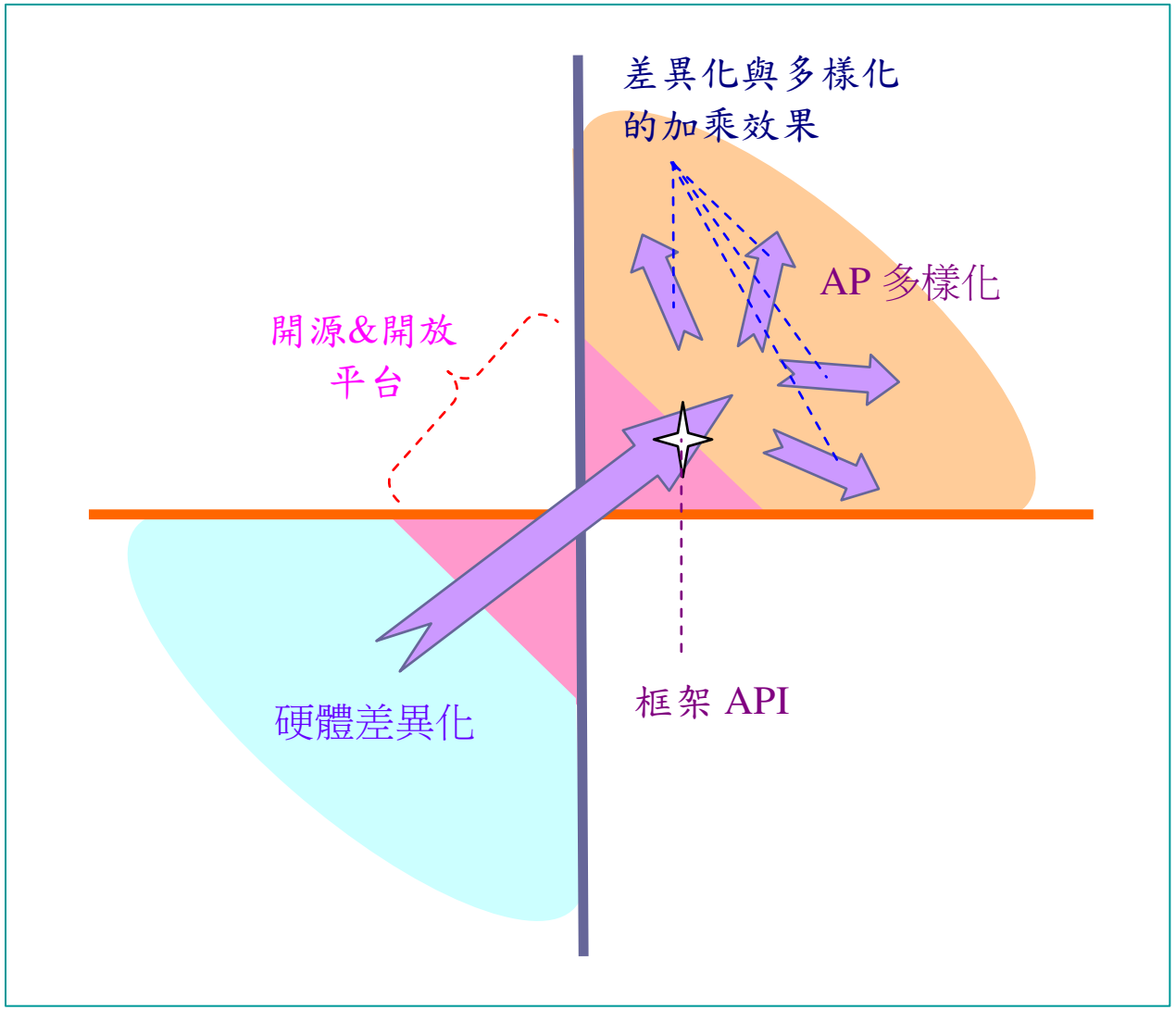
三、增值：
差異化的動能

1) 必要條件：開放&開源平台



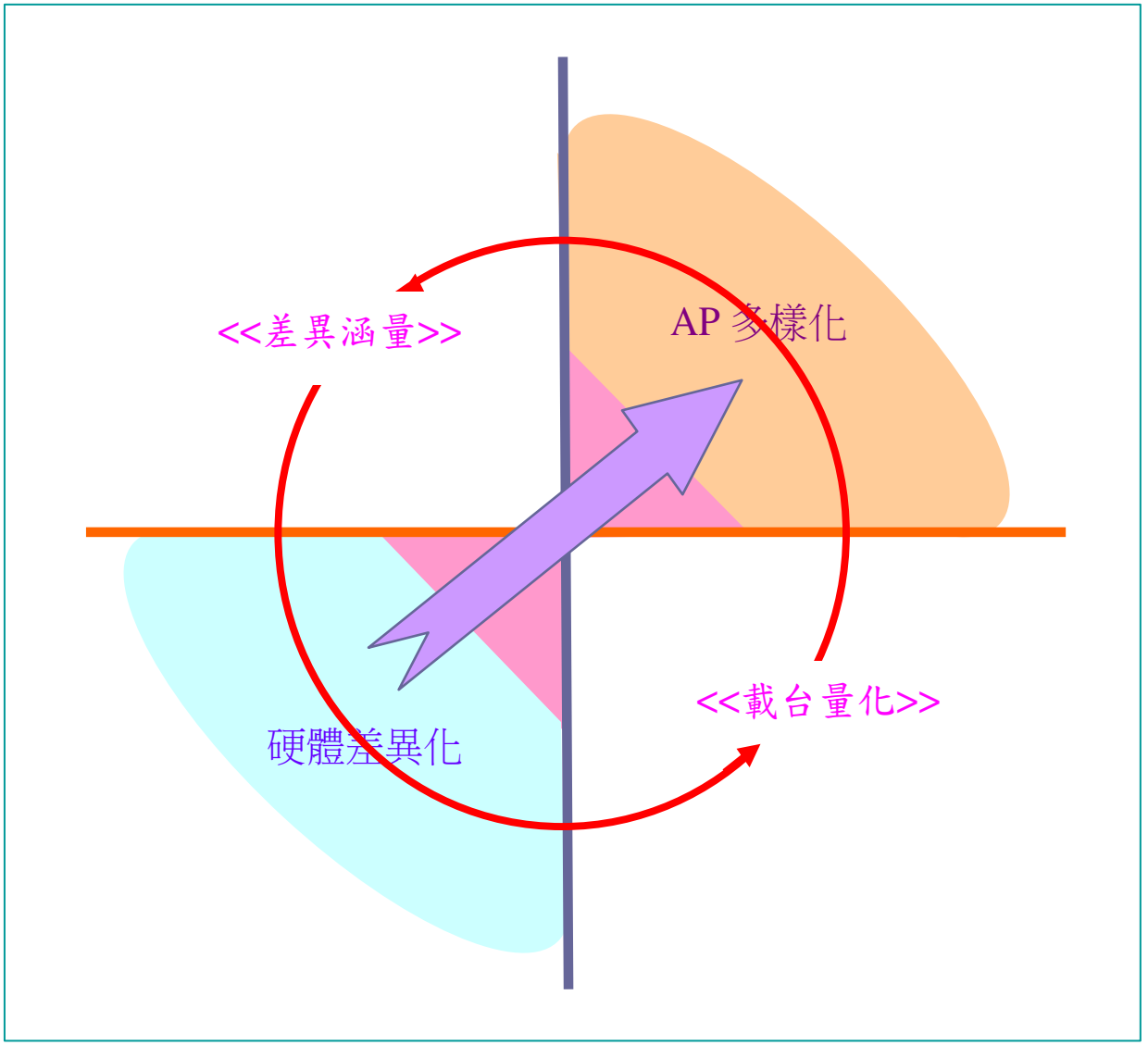
- 在封閉平台裡，硬件模塊就像魔豆被壓在水泥地板下，長不出來，其差異化就不見天日，無法呈現給客戶。
- 所以，開源&開放的平台是魔豆探出頭來的必要條件。

2)充份條件：掌握框架API



- 在此平台裡，差異化魔豆探出頭來，漣漪效應傳達到框架API，與眾多AP相會合。
- 於是，框架API成為差異化與多樣化相遇的地方。
- 掌握API，強力放送差異化。

3) 圓滿條件：發揮量化



- 硬件廠商善用「量化」，積極以硬件的巨大銷售數量，做為AP軟件的載台，替AP軟件創造大量複製的機會。

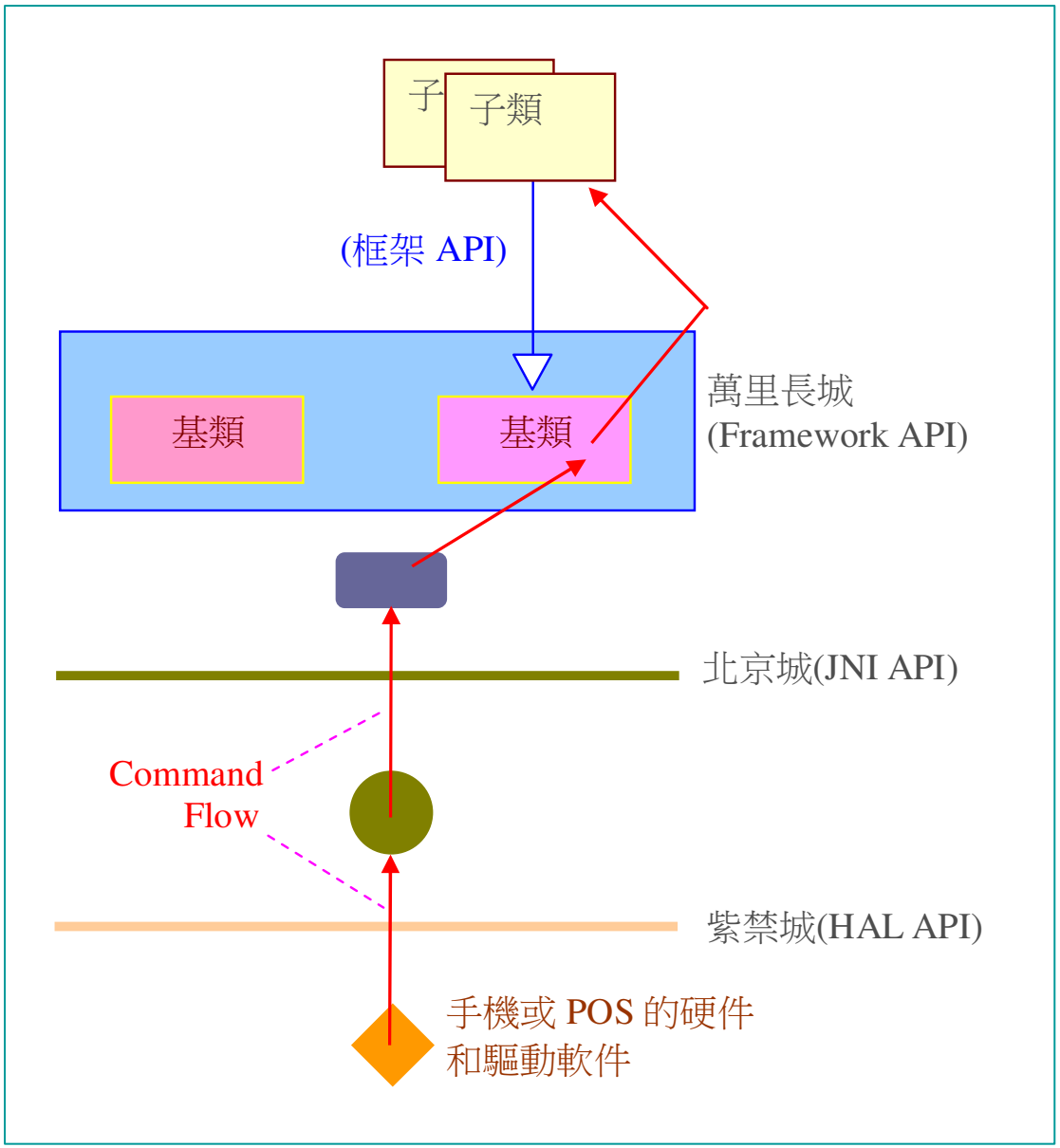
- 廠商愈積極善用「量化」來與AP開發者交換條件，進行軟硬整合，就愈能吸引AP開發者來接受大幅度的差異化。

四、掌控API:
保護增值利益

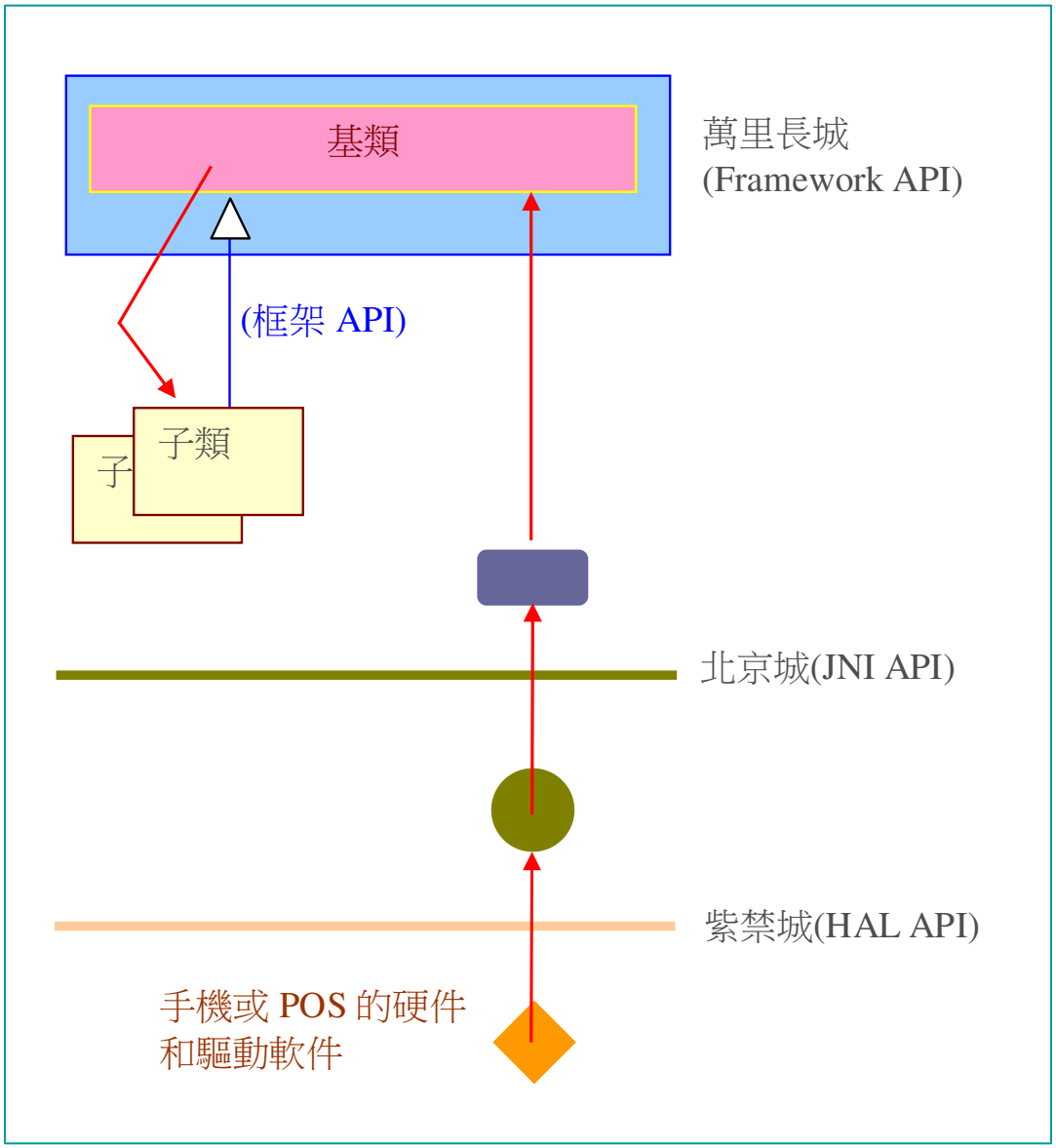
- 紫禁城內清朝皇帝的主導地位，必然會最第一道防線(即萬里長城)設立關口，並重兵駐守，成為具有高度主導性的接口，例如山海關、居庸關等。

- 唯有主導性API(或稱接口，或稱關口)才能確保命令的傳遞和執行。在此平台裡，差異化魔豆才能探出頭來，與眾多AP相會合。

API實踐技術：
基類(Super Class)

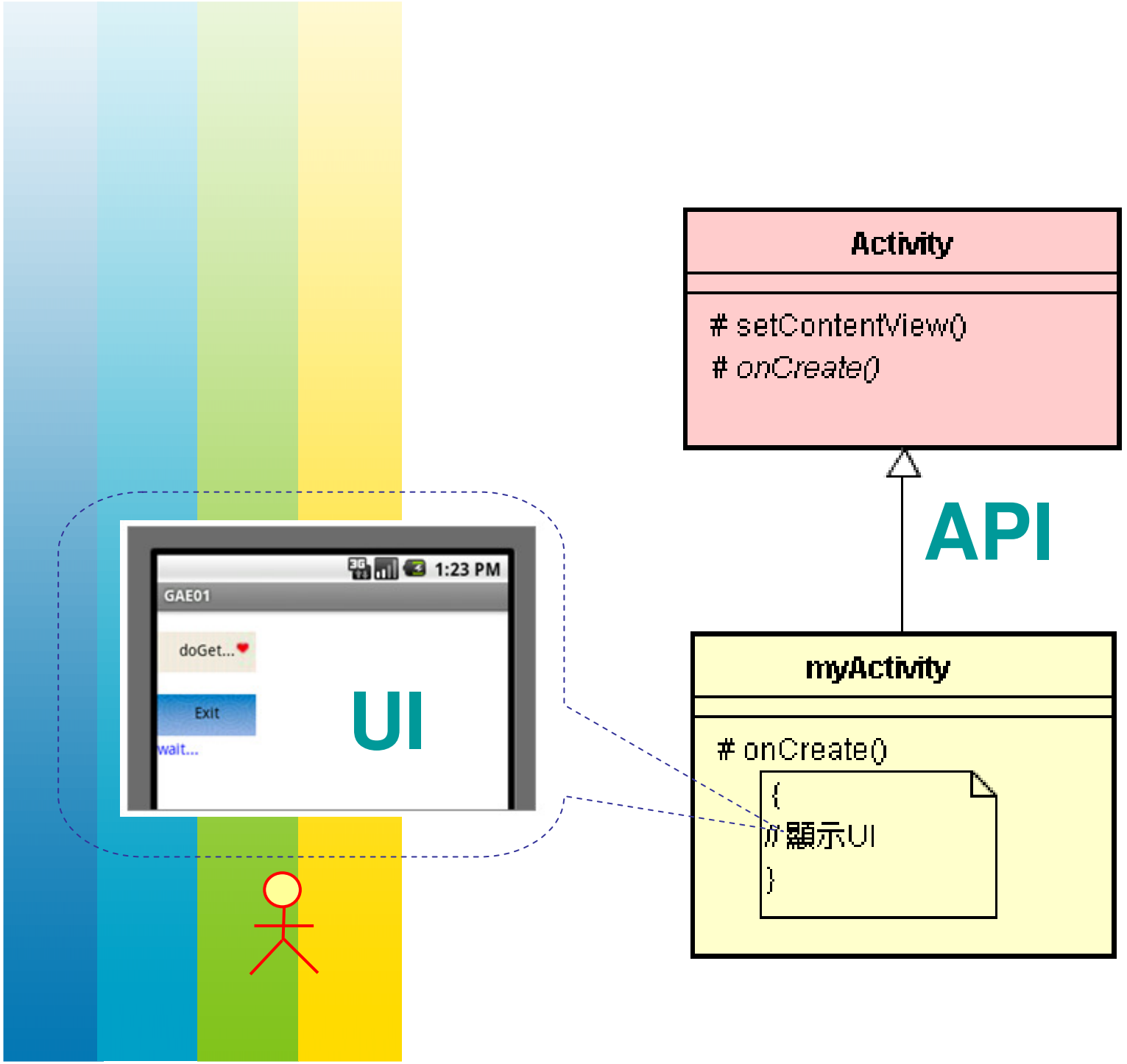


(同上圖)


基類
萬里長城
(Framework API)
(框架 API)
子類
子
北京城(JNI API)
紫禁城(HAL API)
手機或 POS 的硬件
和驅動軟件

區別API與UI

- “Interface” 分為UI (User Interface)和API (Application Programming Interface)。
- Android的分工模式：父類定義API；應用子類提供UI。

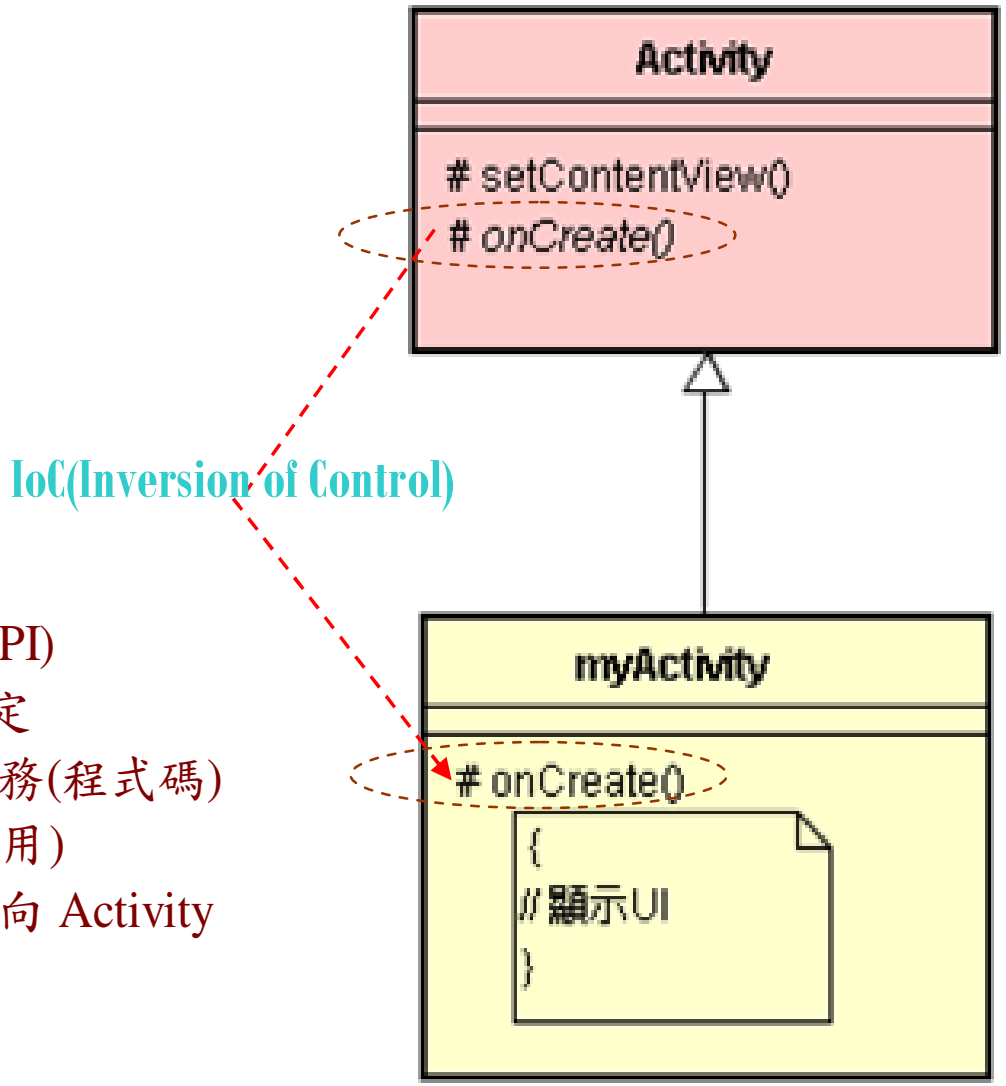

Activity
setContentView()
onCreate()
API
1:23 PM
GAE01
doGet...
Exit
wait...
UI
myActivity
# onCreate()
{
// 顯示UI
}

從API可看出強弱地位

- 接口(Interface)是雙方接觸的地方，也是雙方勢力或地盤的界線。
- 誰擁用接口的制定權，誰就掌握控制點，就能獲得較大的主導權，而位居強龍地位；而另一方則處於被動地位，成為弱勢的一方，扮演地頭蛇角色。

API不平等性

- 是雙方的接口協議(API)
- 由 Activity 開發者訂定
- 由 myActivity 提供服務(程式碼)
- 讓 Activity 來呼叫(調用)
- 服務從 myActivity 流向 Activity



- 從不平等性，就知道Activity擁有主動權；而myActivity子類則是弱勢的，受制於對方。
- 又可推論：Activity開發者處於強龍地位，而myActivity開發者則處於地頭蛇角色。

- 馬關條約是不平等條約，從它的「不平等」性質，就知道雙方的強弱勢地位了。

- 對於一個廠商，只要看看它的軟體模組是否擁有系統上的主導權，就能知道它的是否處於強龍地位；反之，就是位居地頭蛇角色了。

五、途徑：讓魔豆冒出頭來

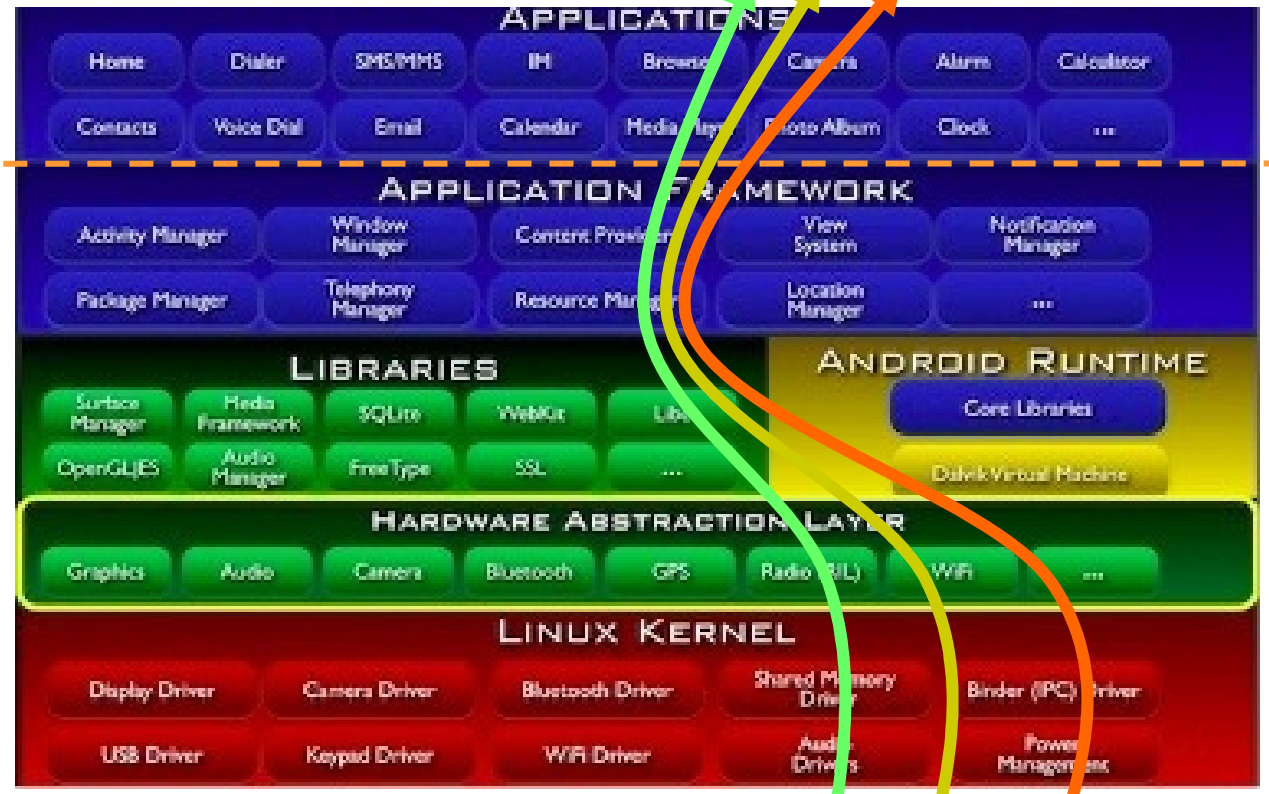
魔豆是什麼?

- 魔豆就是硬件的創新和差異化。
- 例如，不同品牌(如HTC和三星手機)之間形成顯著的差異化。
- 此外，同一品牌的各種型號(如HTC系列Android手機)之間也能持續創新、呈現極大差異化。

魔豆欲
見天日

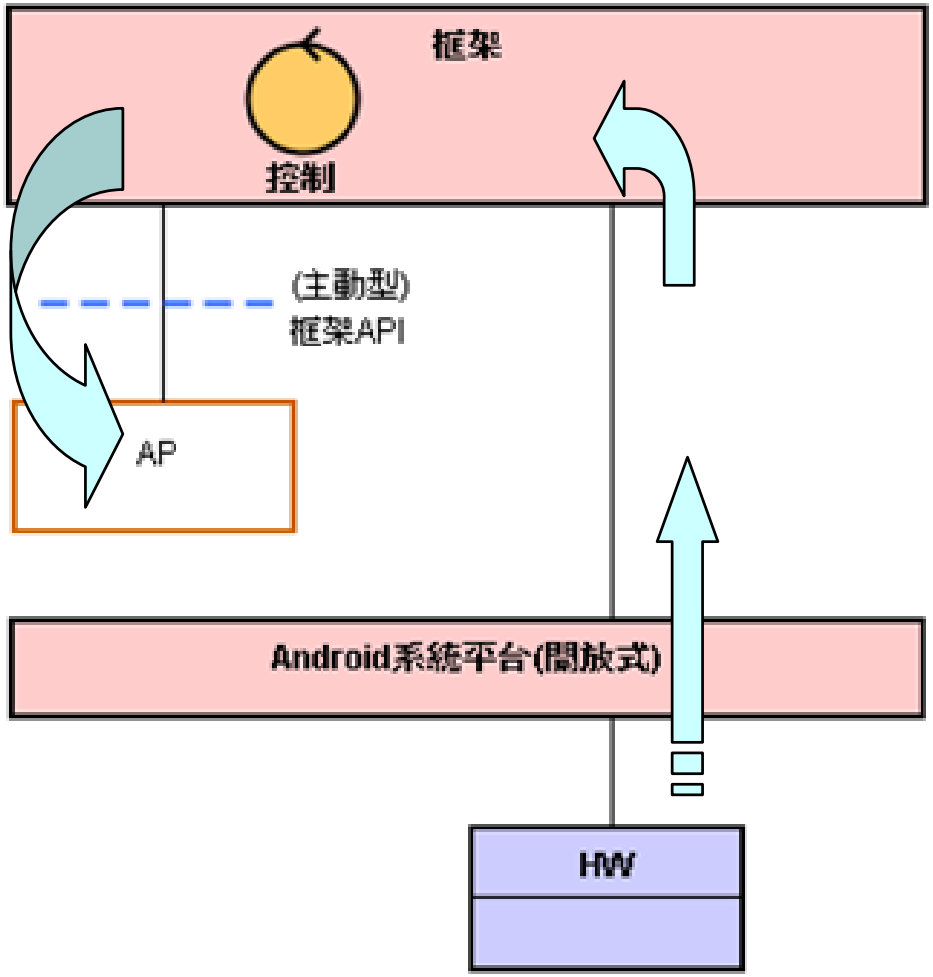
(框架API)

(差異化)



底層硬件模塊

(同上)



六、搭橋：以Core Service銜接驅動與API

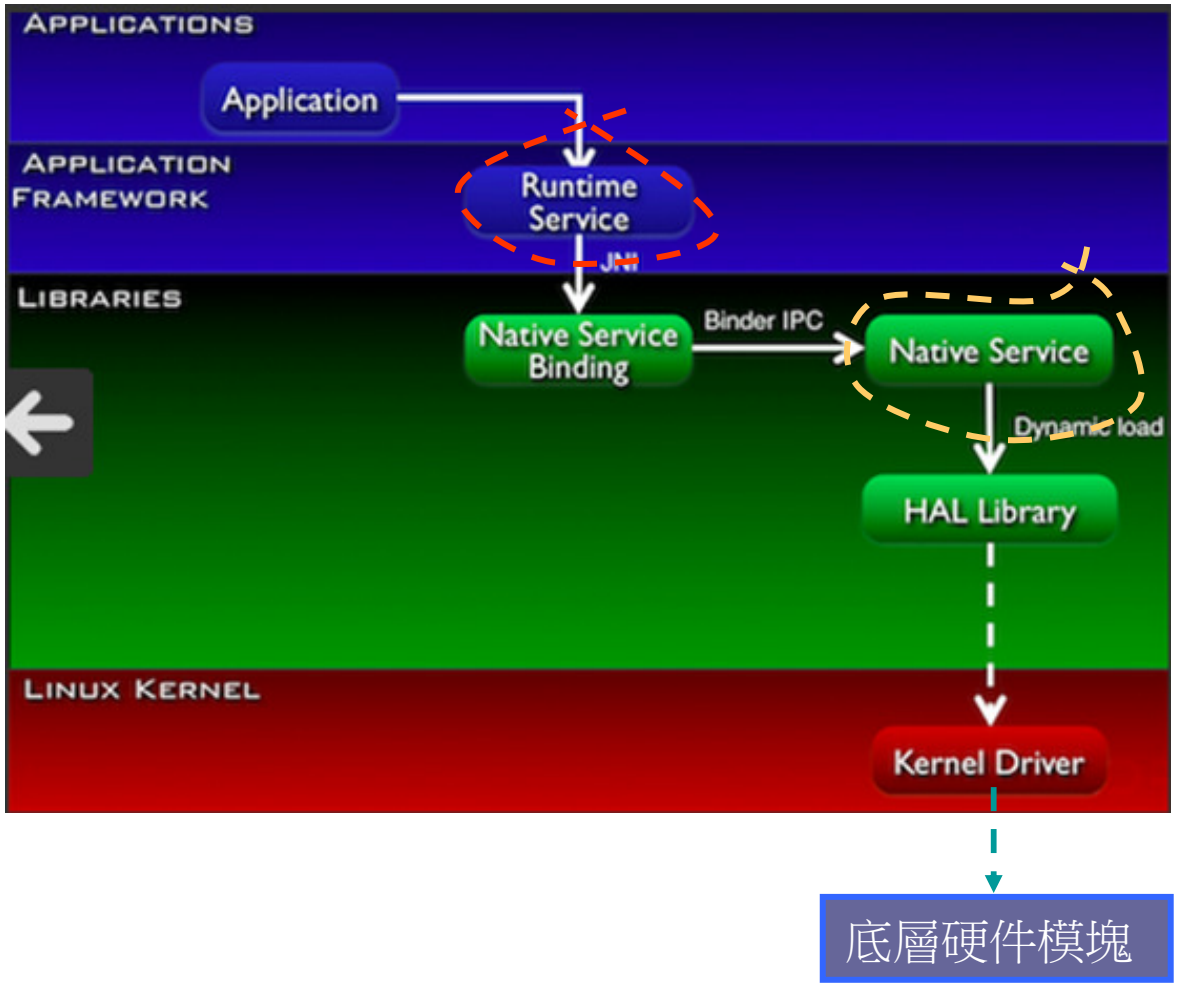
核心服務(Core Service)的角色

- 如果Android體系是一棵樹，則核心服務是「樹幹」部分。
- 它銜接了樹葉(即應用軟件)和樹根(即驅動軟件)。
- 為了發揮硬件差異化和創新特性，核心服務是讓上層AP來使用硬件的管道。例如攝像機、傳感器等底層硬件。

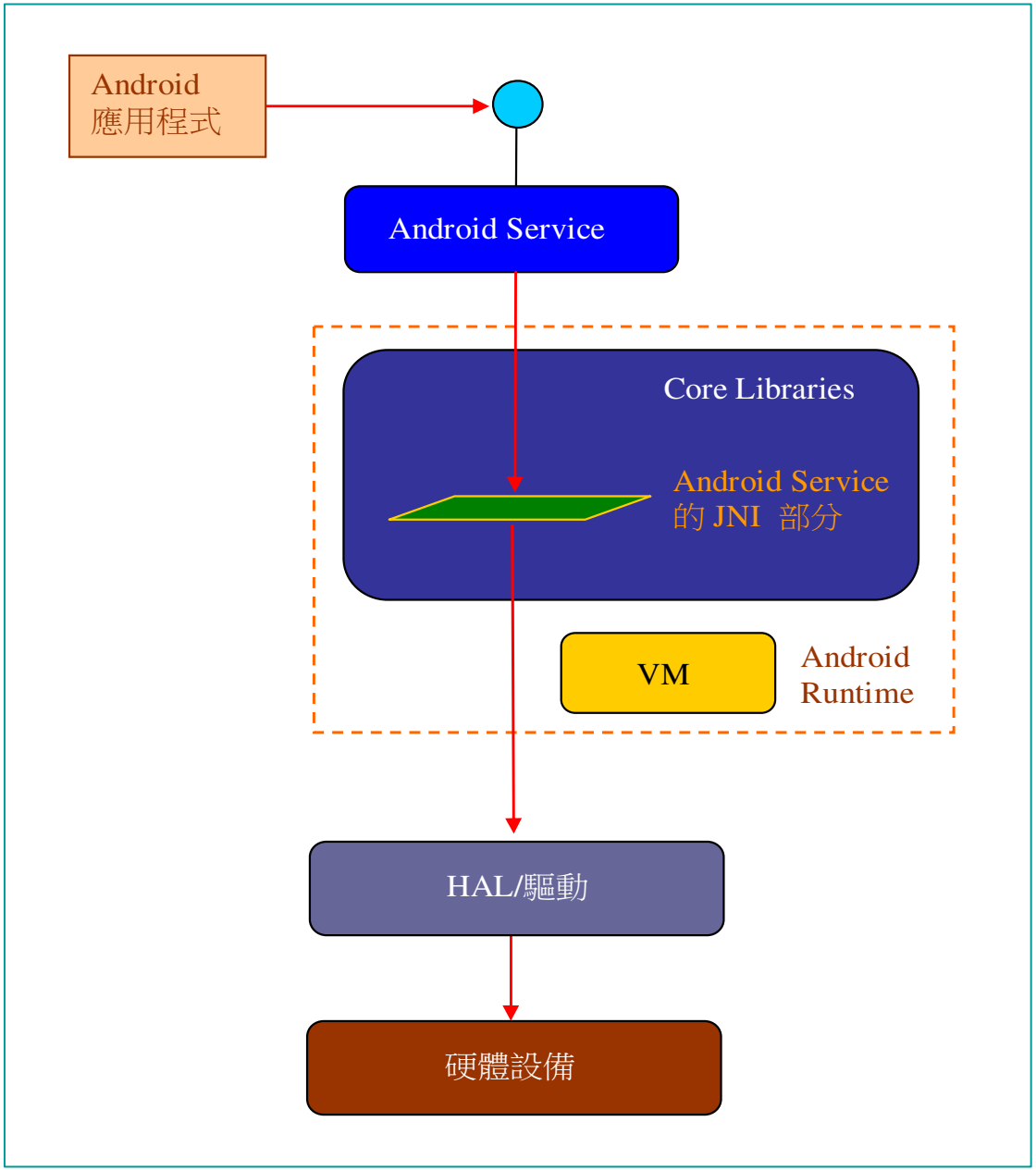
核心服務的特性

- 核心服務是在設備（如手機）啟動階段就啟動的系統服務。
- 核心服務可以用Java撰寫；也可以用C++撰寫。Java撰寫的核心服務稱為Android Service(如AudioService)；而以C++撰寫的核心服務稱為Native Service (如CameraService)。

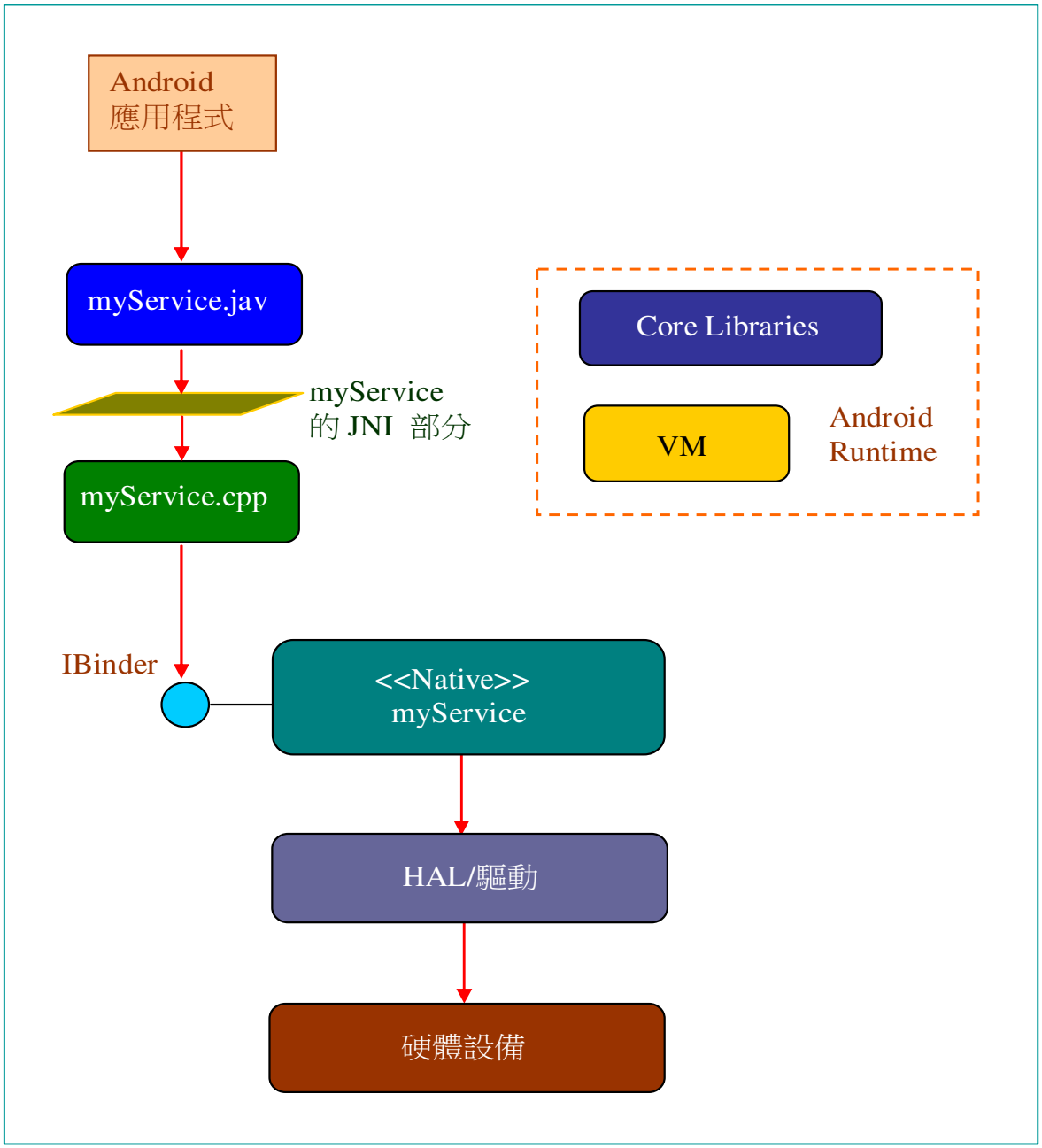
兩種
核心服務


APPLICATIONS
Application
APPLICATION FRAMEWORK
Runtime Service
JNI
LIBRARIES
Native Service Binding
Binder IPC
Native Service
Dynamic load
HAL Library
LINUX KERNEL
Kernel Driver
底層硬件模塊

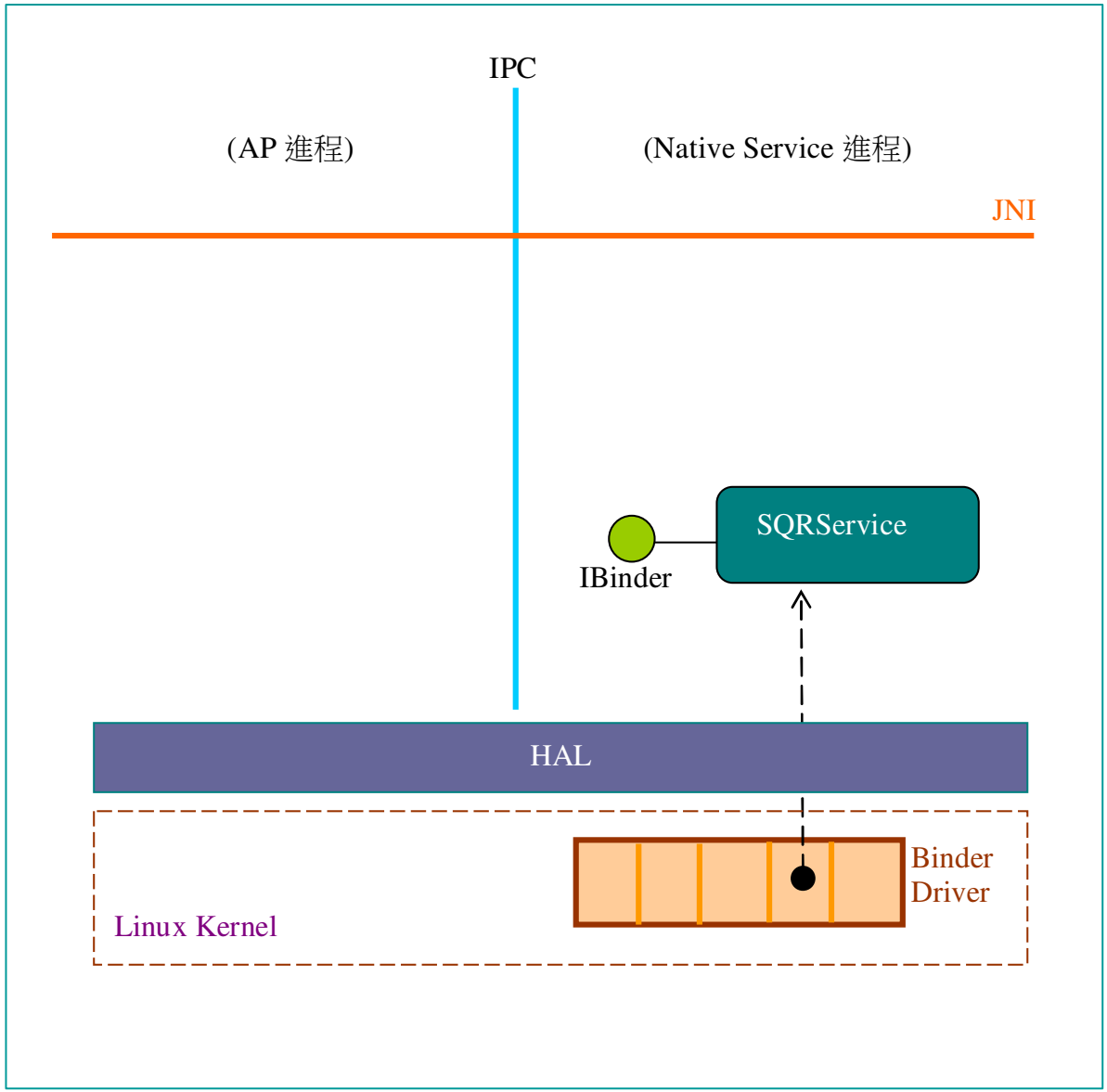
1) Android
(即Runtime)
Service



2) Native Service



核心服務開發範例
(Step-1)

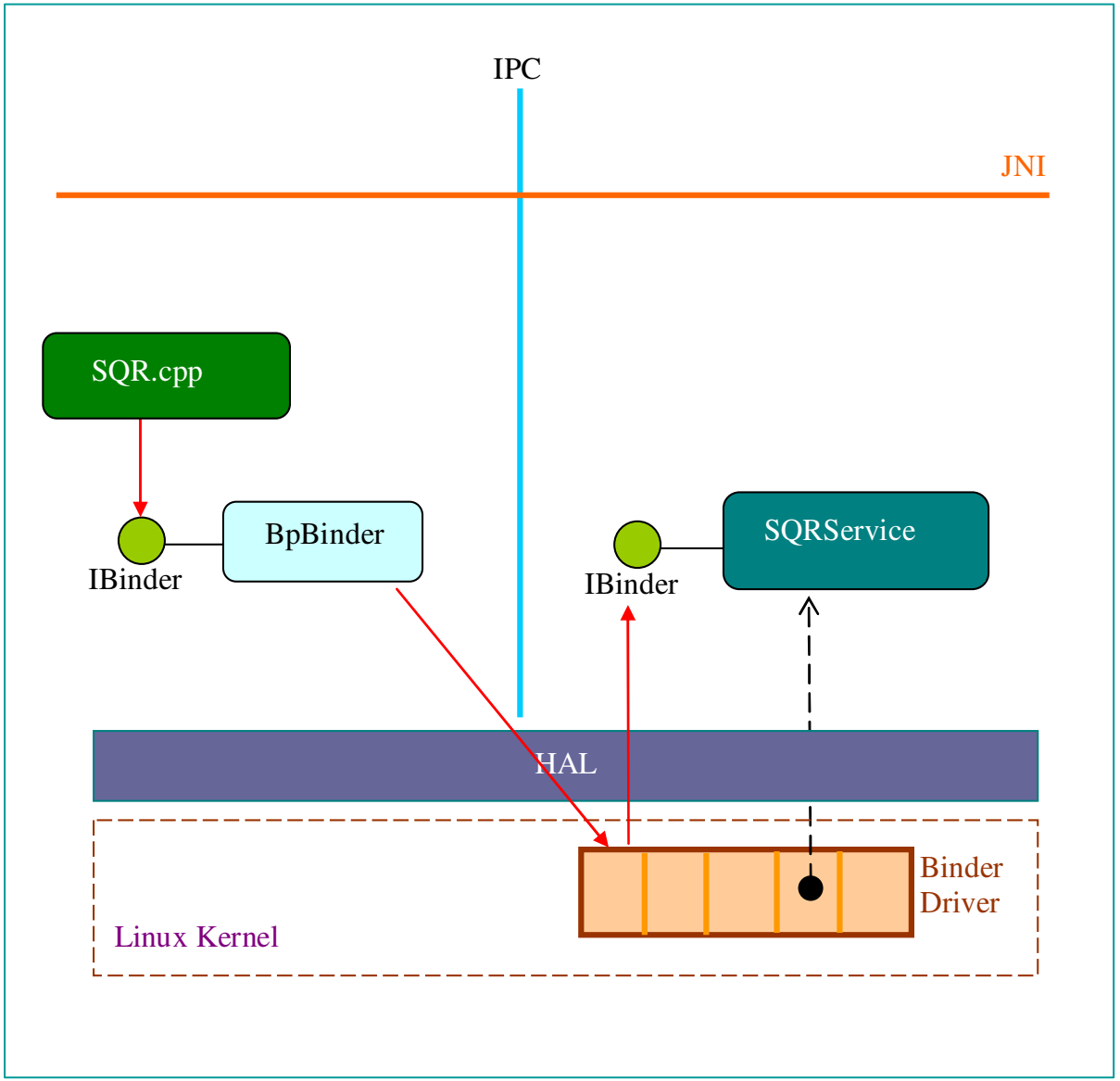


範例代碼

SQRService.cpp

```
// SQRService.cpp
namespace android {
enum {    SQUARE = IBinder::FIRST_CALL_TRANSACTION,    };
int SQRService::instantiate(){
   int r=defaultServiceManager()-
      >addService(String16("misoo.sqr"),
           new  SQRService());   return r;
    }
status_t SQRService::onTransact(
       uint32_t code, const Parcel& data, Parcel* reply, uint32_t
      flags) {
       switch(code) {
          case SQUARE: {
               int num = data.readInt32();
               reply->writeInt32(num * num);
               return NO_ERROR;    }
               break;
           default:  return BBinder::onTransact(code, data, reply,
      flags);
       } }
}; // namespace android
```

C++ Client
調用服務
(Step-2)



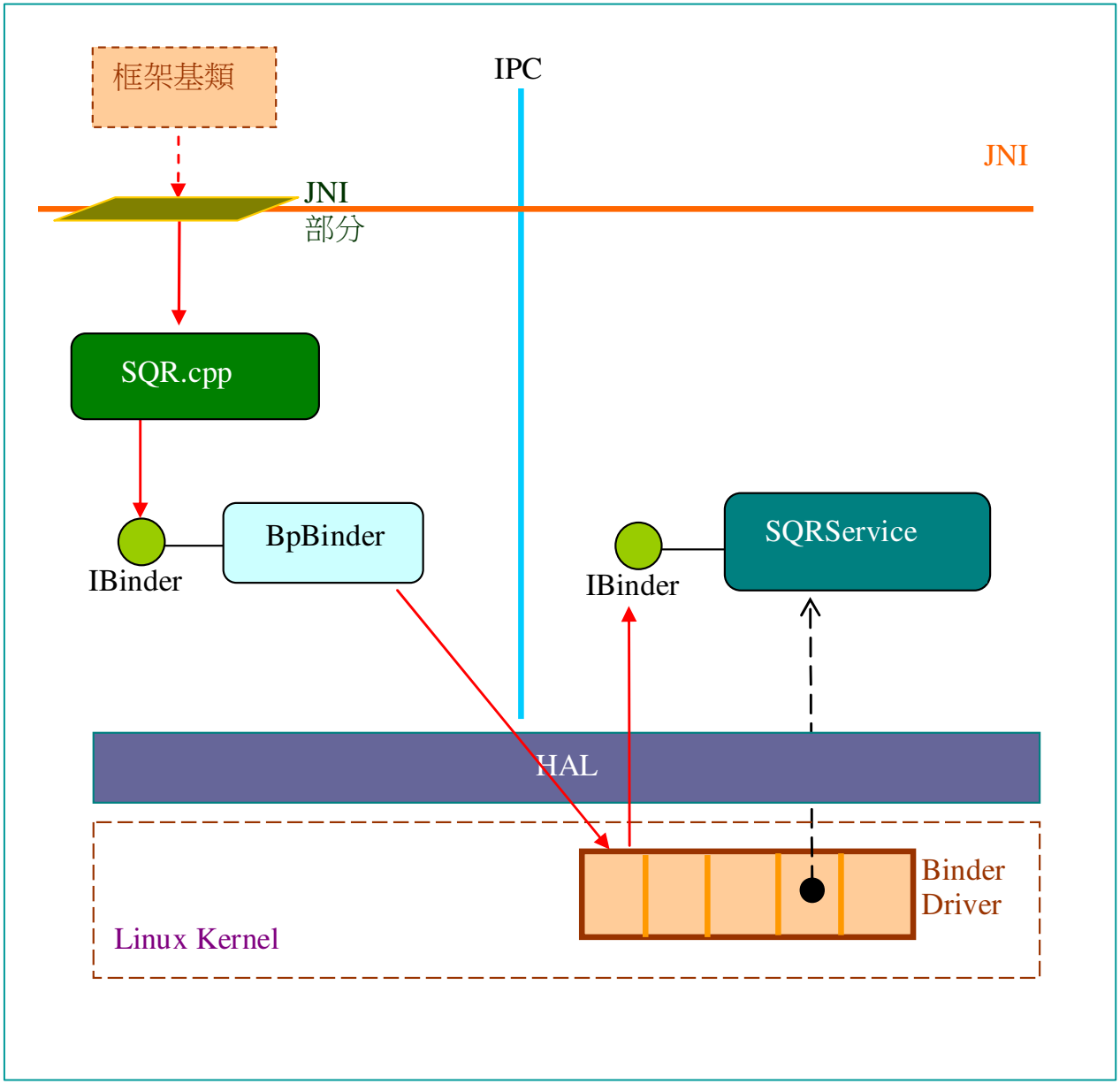
Client代碼

SQR.cpp

```
// SQR.cpp
namespace android {
    sp<IBinder> m_ib;
    SQR::SQR(){  getSQRService();  }
    const void SQR::getSQRService(){
    sp<IServiceManager> sm = defaultServiceManager();
    m_ib = sm->getService(String16("misoo.sqr"));
    return;
}

int SQR::execute(int n) {
     Parcel data, reply;
     data.writeInt32(n);
     m_ib->transact(0, data, &reply);
     return reply.readInt32();
   }
};
```

Java Client
調用服務
(Step-3)

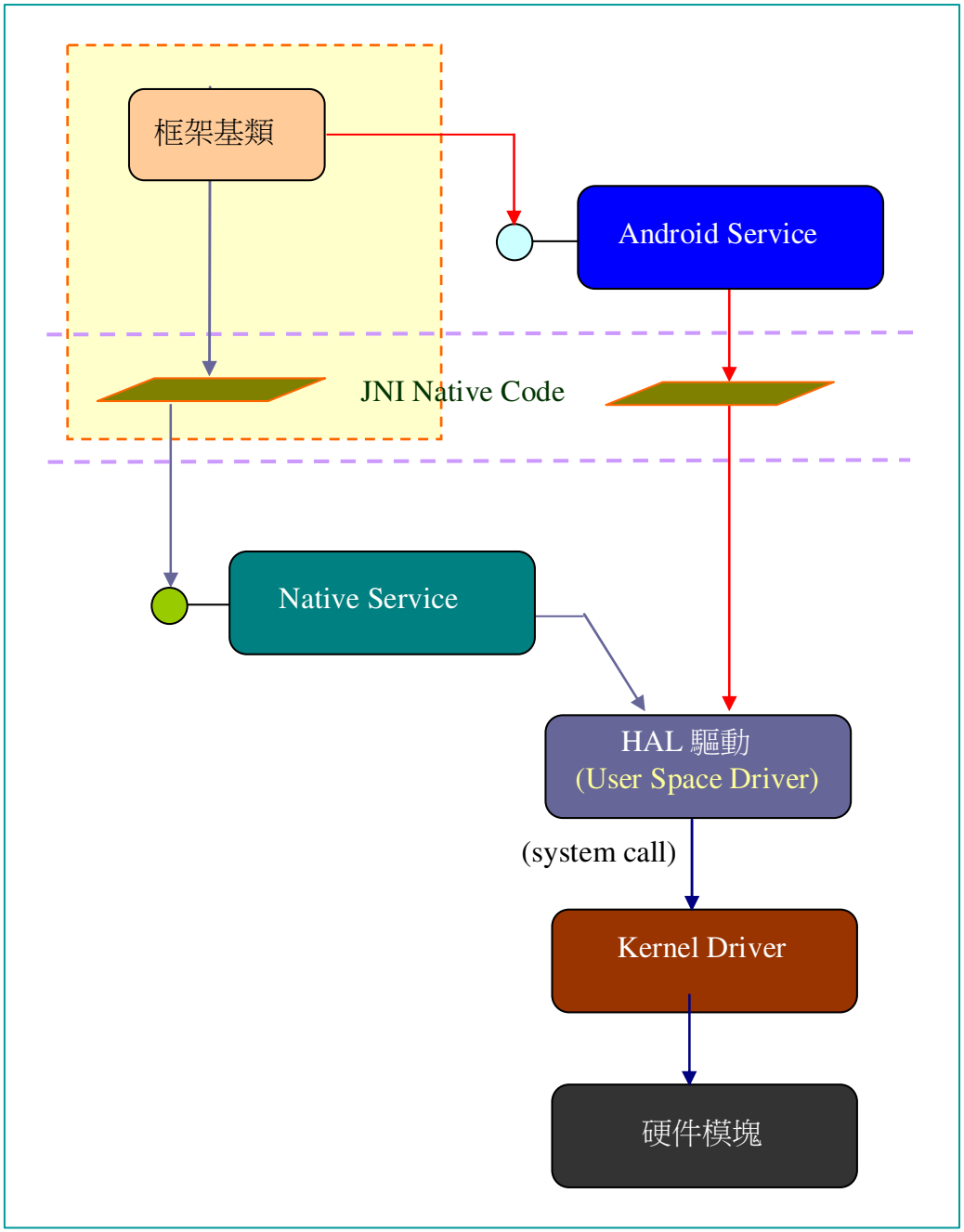


JNI代碼
Service_sqr01.cpp

```
/* com_misoo_service_sqr01.cpp */
#include "com_misoo_service_sqr01.h"
#include "../core_service/SQRService.h"
#include "SQR.h“

using namespace android;
JNIEXPORT jint JNICALL
    Java_com_misoo_service_sqr01_execute(
        JNIEnv *env, jobject thiz, jint x)
{
    SQR* sqrObj = new SQR();
    int num = sqrObj->execute(x);
    return num;
}
```

銜接框架到模塊


框架基類
Android Service
JNI Native Code
Native Service
HAL 驅動
(User Space Driver)
(system call)
Kernel Driver
硬件模塊

七、結語：推動技術持續提升

Summary

- 在Android平台上，會面對更多的底層模塊和上層框架功能的擴展的需求。
- 擴展模塊和功能會增加成本。
- 差異化增值可彌補成本，進而大幅獲利。

- 將差異化增值觀點納入擴展模塊和功能的技術要點之中。
- 差異化增值→擴展與創意

- 如此，推動技術的持續提升。

Q&A

android 台灣 Android論壇 www.android1.net

Chairperson

高 煥 堂

Tom Kao

+886-928-534-552
+86-1366-1542050
misoo.tw@gmail.com
android.tangsk.com (JP)

日本東京Android国际交流中心 主持人