國立交通大學

資訊科學系

碩 士 論 文

風 格 化 使 用 者 介 面

Style-based User Interface

研 究 生：藍彥琨

指導教授：李嘉晃　教授

中 華 民 國　九 十 三　年 六 月

風格化使用者介面

研究生：藍彥琨　　指導教授：李嘉晃 教授

國立交通大學電機資訊科學院 資訊科學研究所碩士班

中文摘要

在消費性電子產品的設計上，過去的重點強調在功能多、效能快等技術層面的開發，現在則越來越講求造型新穎、產品個性化這樣的使用者感受。有鑑於此，本篇論文提出一種風格化的使用者介面架構在手機上的應用，讓手機使用者可以藉由不同的介面描述方式，改變不一樣的操作環境。

本篇提出的架構是用 XML 和 Java 來實作。利用手機是一個特殊應用的操作環境這樣的特性，我們使用 XML 來定義出各種可能的操作介面，進一步利用這樣介面描述語的方式，將使用者操作環境呈現出風格化樣式的顯示。而藉由 Java 跨平台的特性，更希望達到在任何支援 JVM 功能的手機平台上，能夠以定義好的介面描述檔在不同的手機平台上執行，而不需要做任何修改。另外，由於介面描述檔是 XML 的文字敘述，所以不論是手機設計者或是使用者本身都可以自行加以編輯修改，設計出想要的操作介面。

Style-based User Interface

Student: Yen-Koun Lan Advisor: Prof. Chian-Hoang Lee

Department of Computer and Information Science
National Chiao Tung University

Abstract

With the progress of technology, the requirement for appearance design in consumer products is getting more important. In addition to functions or performance, fantastic exterior with style-based design becomes the dominated consideration of the users. Be aware of this trend, this thesis proposes a styled-based user interface architecture in cellular phone environment.

The proposed architecture is implemented with XML and Java. Since the applications used in cellular phone environment are limited, we take XML as the user interface description language to describe the possible frame. Furthermore, we can use such user interface description to construct a style environment in cellular phone. Besides, take advantage of cross-platform characteristic of Java, we hope that such method can run any cellular phone with JVM ability without changing any thing.

誌謝

本論文得以順利完成，首先要感謝指導教授李嘉晃老師，他細心懇切的指導，豐潤了本文的內容，使其不致於結構鬆散、文字冗長。此外，老師嚴謹的研究態度，切中要題的研究精神，也使得我在兩年的學術研究生涯獲益良多。也感謝口試委員王勝德教授、李肇林教授、以及陳登吉教授的指正與建議，使得本論文能更加的完善。另外，也感謝聯發科技提供這樣的合作計劃，讓我能夠學以致用。

此外，還要感謝實驗室的學長、同窗及學弟們對我的關心與幫助，充實了我在實驗室的生活。

最後，我要感謝陸旭芬及我的家人，因為他們的支持是我努力的最大動力。僅以此篇論文，獻給我的摯愛們。

圖目錄

表目錄

第一章 緒論

1-1 概述

隨著科技的日益進步，除了講求功能多、效能快之外，愈來愈多的消費性電子產品開始朝向講求造型特殊、設計新穎的視覺風格方向邁進。1998 Apple iMac 的出現，帶動了一股這樣的設計風潮。對使用者而言，吸引眾人注意的第一印象是 iMac 的視覺造型設計，而非其高效能的硬體規格。這種現象說明了現在使用者對科技產品不再以功用、效能為主要的考量，另外對產品本身的觀感要求也愈來愈更加重視。

CPU: PowerPC 750
CPU Speed: 233 MHz
Bus Speed: 66 MHz
Data Path: 64 bit
Minimum RAM Speed: 100 MHz
Maximum RAM: 256 MB
Level 1 Cache: 32 kB data, 32 kB instruction
Level 2 Cache: 512 kB backside, 1:2
Monitor: 15"
VRAM: 2 -6 MB SGRAM
Maximum Resolution: 16 bit 1024x768
Dimensions: 15.8" H x 15.2" W x 17.6" D
Minimum OS: 8.1
Maximum OS: 10.3.3
Introduced: August 1998

圖 1：iMac 及其規格

(來源：www.apple.com)

以整個研發的時程來看，過去的研究著重在技術層面的開發，如改善軟硬體的結構、新的演算法等，來改善效能增進執行速度；現在則開始強調使用者的感受，如造型新穎、易於使用等，來吸引使用者的注意。整個研發的重心已從過去開發設計者的角度轉到以使用者的角度來看待科技產品，而產品的研發愈來愈重視 user friendly 的機制，讓使用者能夠方便的使用。這種強調以 “ 人 ” 為主的設計概念，可以從許多廠商的產品發展策略反應出來：Nokia 的廣告語 “ 科技始終來自於人性 ”，Samsung 強調產品

的高時尚感等，它們的訴求已不著重在產品的功能面上，而開始講求產品對人的價值。這些種種現象，所突顯出的是強調風格設計的時代來臨。

1-2 動機

這種開始著重在風格設計的轉變，可以從現在許多產品的趨勢反應出來。XPC (迷你準系統) 的出現，改變了一般人對 PC 外觀的刻板印象，引起許多廠商的爭相仿效而蔚為一股風潮；如今許多 PC 的設計更朝向可以作為客廳擺設的電子產品，著重在外觀造型的亮眼。而在手機市場上，鈴聲下載、來電答鈴這些加值服務的流行，顯示出使用者已不滿足現況；手機可換外殼的設計及內容 Theme 的可變換功能，更表現出使用者想與眾不同的心態。這些現象，表現出的就是使用者開始著重於產品的 “ 軟性 ” 功能。

圖 2：浩鑫 ST61G4 XPC 及 華碩 Digimatrix

有鑑於此，除了這些外觀造型 ”硬體形式 ” 的特殊化、個人化之外，本篇提出在手機上以風格化為主的使用者介面架構 (Style-based user interface)，這種 “ 軟體形式 ” 的特殊化、個人化，而有別於傳統以 “ 功能 ” 為主的的操作模式。舉例而言，Microsoft 的 Windows 系列或是 Apple 的 Mac OS 即是最佳的例子，任何人只需從使用者的操作介面即可立即判斷此為何種作業系統，而不是需要真是操作或是查詢系統資訊之後才會知道。

1-3 目標

一般手機內容的設計都是將底層軟、硬體的機制功能和使用者介面視為一體考量，這樣的設計方式，即使只是將使用者介面稍作變化，也都必需更動到整個系統的開發；而本篇提出的方式主要是將使用者介面由此一體設計的概念抽離出來，針對使用者介面獨立設計。藉由定義出的介面描述語法，呈現出使用者的操作環境，因而可將底層的功能開發和使用者介面設計分開，如此可藉由改變不同描述語法的設定，而呈現出不同的顯示結果。

本篇以 JAVA 語言來實作，主要是運用 JAVA 跨平台的特性，讓我們可以以定義好的介面描述語法能在不同的手機平台上執行，而不需做任何修改。當系統運作時，只要在任何可執行 Java Virtual Machine 的手機平台上，使用者的操作環境即可依據所給予的介面描述檔而呈現出相對應的畫面。另外，整個設計概念是將手機底層的軟、硬體機制和操作介面分開，所以使用者可藉由改變不同描述語法的設定，做成不同的介面描述檔，在同一個手機上而呈現出不同的操作環境。圖 3 的例子說明，假設我們有三種不同風格樣式的介面描述檔 A、B、C，藉由選用不同的介面描述檔，我們即可在同一款手機上呈現出三種不同的使用者操作環境。

圖 3：不同描述檔呈現在同一個手機上

甚至更進一步可藉由同一個介面描述檔，即使底層的手機硬體不同，只要這些手機都能支援 Java Virtual Machine，也可在這些不同硬體手機的平台上呈

現出相同類似的使用者操作環境。舉例而言，假設有 Panasonic，OKWAP，Sony-Ericsson，這三家手機廠商支援 Java Virtual Machine 的手機，只要有定義好的一個介面描述檔，這樣就可以使得這三支不同的手機擁有相同類似的操作環境，如圖 4。

圖 4：同一描述檔呈現在不同手機平台上

這樣子的設計重點著重在於使用者可以方便的藉由介面描述語的設定，變更為自己想要呈現出的操作環境，而不需要受限於手機廠商所設定的畫面。另外，藉由風格設計的概念，手機廠商可以提供許多不同風格的介面描述檔，讓使用者可以下載使用，變更操作介面，而不是只能更動手機桌面的畫面或是改變內容顏色而已。舉例來說，若有的使用者十分喜愛 Mac OS 的風格，手機廠商就可以根據這樣的風格，將手機內容設計成一款類似的操作介面，讓使用者可以更動來使用。

因此，為了達成上述目的，我們有兩項因素需要考量：一、介面描述語法如何定義，二、系統如何運作。介面描述語法部份，我們參考 XML 相關技術，以標籤（tag）作為介面描述語的定義。 XML 的檔案形式為文字檔，可以讓手機介面設計者或是手機使用者容易明白。系統架構設計上，則將一般原來支援 Java 功能手機的執行流程做一些修改，讓畫面的顯示透過我們定義好的方式呈現，在第三章-系統設計會詳細探討。

論文架構

本篇其餘部份，分為：第二章-相關背景知識，第三章-系統設計，第四章-系統實作，第五章-跨平台的討論，第六章-結論及未來工作和參考文獻。

第二章 背景知識

2-1 概述

近幾年來手機市場的蓬勃發展，使得這塊領域已成為各家兵爭之地。Nokia, Motorola, Samsung 及國內各家手機廠商推陳出新的展出各種新型手機，莫不希望提高市場佔有率。TI, Qualcomm 等通訊晶片設計大廠不斷提出新的平台架構，希望能夠提昇手機的效能。Symbian, Palm, 甚至 Microsoft 等研發各種手機平台作業系統及應用，讓手機結合各式功能以達到最佳的使用效果。在各方的努力之下，許多手機相關的技術及應用日漸成熟， Smart Phone、 Java Phone 等已成為日常所見的名詞。而 Sun Microsystems Inc.所推的 Java 功能，甚至已成為中、高階手機的基本配備。

Sun 於 1991 為了在消費性電子產品上的應用及因應網路的興起，發展出了一套完全物件導向，不受平台限制的 Java 語言。Java 基本上由應用程式介面 (API) 及虛擬機器 (Virtual Machine) 兩部份所組成。 Java API 是各個軟體元件的集合，提供各式類別給人使用。 Java Virtual Machine 是一個定義好的標準規格，可由軟體或硬體來實作。 程式執行的流程，是由 Java 編譯器將 Java 原始碼編譯成 bytecode 後，再載入虛擬機器執行。

Java 語言最大的特色優點即是跨平台的特性。程式開發者只要寫好開發的程式，即可移植到任何可以支援 Java Virtual Machine 的系統平台上執行，而不需要做修改。此一特性，也是本篇選擇以 Java 來實作的原因。

2-2 Java 2 Platform, Micro Edition (J2ME)

由於嵌入式系統的發展逐漸大量的成長，於是 Sun 於 2000 年推出 Java 在嵌入式系統上的發展平台，Java 2 Platform, Micro Edition (J2ME)。嵌入式系統在設計上原本就受限於硬體資源的有限，因此在設計的過程中，對於資源的如何有效運用，就顯得十分重要。 而整個 J2ME 的軟體架構層級如圖 5 所示。

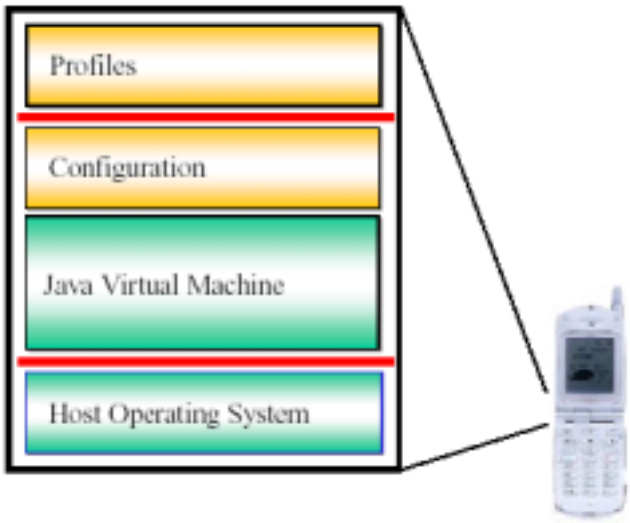


圖 5：J2ME 軟體架構（來源：java.sun.com）

J2ME 定義了在嵌入式及消費性電子產品上發展所需的相關規格，如 Java Virtual Machine (JVM) 及相關的程式庫和 API 等等。主要的應用可以分為兩類，一類為 Connected Device Configuration (CDC)，如 數位電視、settop-box 等；另一類為 Connected Limited Device Configuration (CLDC)，如行動電話、PDA 等。

在 J2ME 的軟體架構中，Configuration 定義了一群類似的裝置中所需要的最小平台需求，如記憶體的容量或運算能量等。其包含了 Java 語言、開發裝置所需的最小類別程式庫及虛擬機器的特性。目前 J2ME 定義了兩種 Configuration: Connected Device Configuration (CDC) 及 Connected Limited Device Configuration (CLDC)。CDC 主要應用在固定、可分享資訊的可連結裝置上，而 CLDC 主要應用在個人化、可移動性的可連結裝置上。

而 J2ME 中 Profile 定義了與各領域相關的類別程式庫和 API，與使用者息息相關。Profile 是架構在 Configuration 之上，其主要目的是保證各個裝置在某個 Configuration 之間的相容性。目前 J2ME 定義了一個架構在 CLDC 上的 Mobile Information Device Profile (MIDP)，應用在行動電話上開發環境中。而在 MIDP 環境上開發的應用程式稱為 MIDlet。基本上， J2ME 在行動電話上的開發平台架構如圖 6。

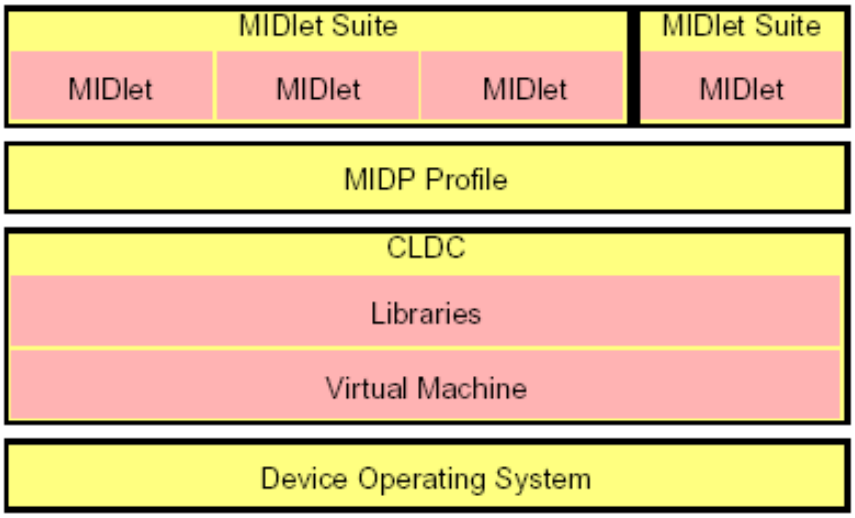


圖 6：J2ME 在行動電話上的開發平台架構（來源：java.sun.com）

2-3 Symbian Java Phone 架構

根據市調機構 IDC 的資料顯示，2003 年 Symbian 平台智慧型手機的出貨量在市場上的占有率有六成多（Nokia 占 56.9%，Sony-Ericsson 占 8.5%）。Symbian 是一家來自歐洲的嵌入式系統軟體廠商，其 Java Phone 的環境是架構在 Sun 所規範的 PersonalJava Application Environment 版本，並提供 Java Phone API 的實作。整個平台環境最底層為 EPOC，是 Symbian 專門針對行動資訊設備所設計的嵌入式作業系統，其上為 Java 虛擬機器及類別，再上一層是 Java Phone 的應用介面類別群組，最上層則是應用程式。

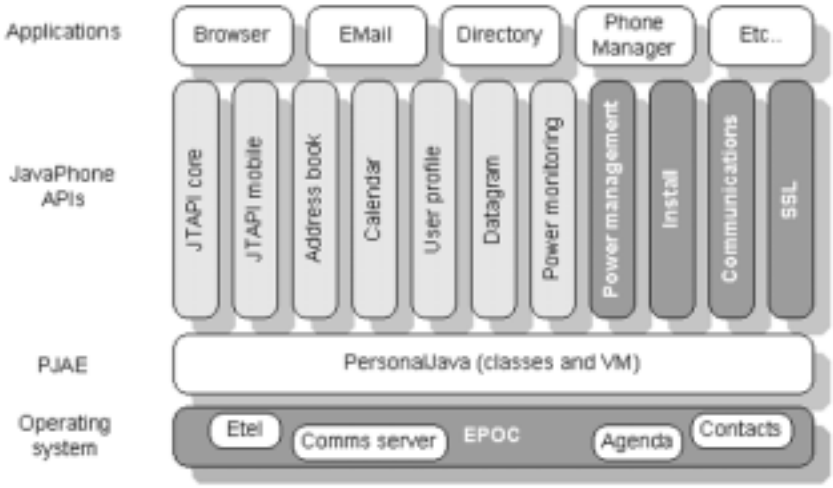


圖 7：Symbian Java Phone 架構 (來源：www.symbian.com)

圖 7 顯示出 Symbian Java Phone 的架構。架構最底層的 EPOC 是一個 32 位元多工的嵌入式作業系統，最主要全部以物件導向的概念來設計並以 C++的語法來實作，其核心採用主從式架構 (client-server architecture)，系統執行時所有資源的分配都需要向 kernel server 來要求。另外，Symbian 還實作了 Java Phone 的 API，將手機相關應用所需要用到的功能，提供給上層應用程式開發者容易使用。

2-4 Java Virtual Machine (JVM)

Java Virtual Machine (JVM) 是一個定義好的規格標準，擁有自己定義的指令集及對記憶體的操控等等。因此只要任何符合這項規格，不論是以硬體或是用軟體來實作，皆視為 JVM。目前較為常見的有 Sun 的 JVM，Microsoft 的 VM，或是其它廠商自己開發的 VM，如 IBM 等。另外，也由於這是一個開放的規格標準，所以有許多開放原始碼的 JVM 實作計劃，例如：Kaffe、Latte、SableVM 等等。

2-5 Kaffe

Kaffe 是一個以 C 語言來實作 JVM 規格的開放原始碼計劃，其版權為 Gnu

Public License (GPL)，任何人都可以修改原始碼並加以發佈，目前已經被移植到相當多的硬體平台上，包括 x86、ARM、M68K、PowerPC、Alpha、Sparc、MIPS 等等。Kaffe 在設計上將各個相關的功能設計成模組，包括 driver module 負責 kaffe 執行時控制整個流程的主要部份，class management modules 負責載入類別的 bytecode、查詢和更新各類別、物件、方法及例外、並驗証類別在內部使用的一致性，class path management module 負責管理對 Java 執行的相關類別路徑，class execution modules 負責連結 driver module 和 Java 解譯器 (interpreter) 或是 Java Just-In-Time (JIT) 並做 bytecode 的驗証和執行，data area management modules 負責管理 kaffe 在執行時的資料，memory management modules 負責管理所有 kaffe 所使用的記憶體，native support module 主要是實作 Java Native Interface (JNI)，helping modules 負責處理跟系統相關執行緒 (thread) 的工作。圖 8 是 kaffe 的架構。

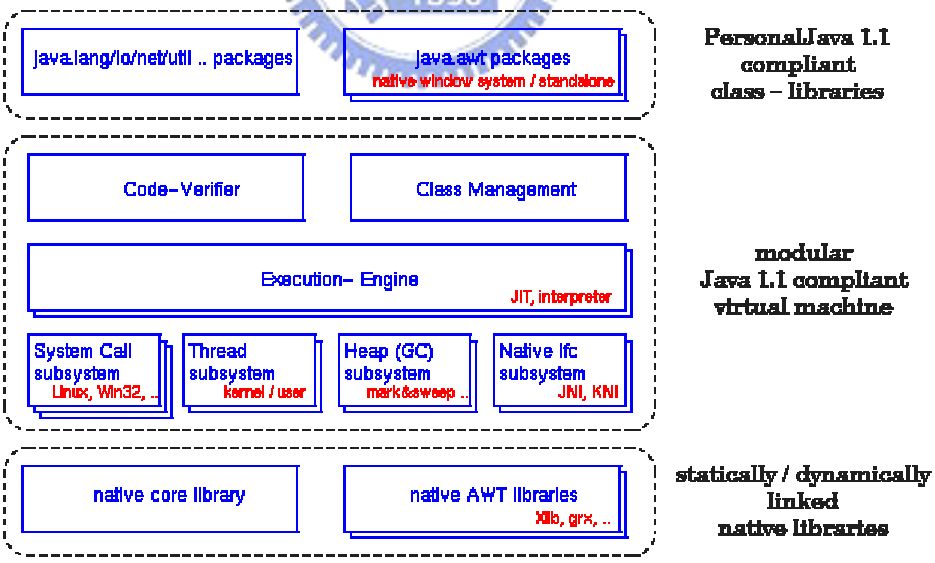


圖 8：Kaffe 架構 (來源：kaffe.org)

第三章 系統設計:

3-1 概述

本篇所提出的目的是希望在支援 JVM 手機功能的平台上，以一個介面描述檔作為使用者操作環境的呈現，更進一步的運用這種介面描述的方式，形成一種風格式的使用者操作環境。

目前手機在設計的流程上，通常將底層機制和使用者介面視為一體，因此任何使用者介面的更動都必需修改原始碼，使得手機在設計上極為不便。另外，這樣的設計方式也由於是和底層的軟、硬體息息相關，所以若要重新開發新的手機，等於要重新整個設計流程。針對這些設計上的不便，本篇提出一種以介面描述語的方式，將使用者的操作環境以風格化的方式呈現出來。

3-2 目前手機設計架構

圖 9 是目前支援 Java 手機功能的平台架構：

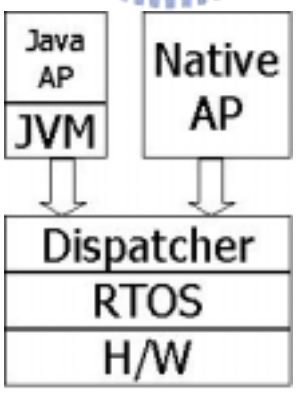


圖 9：支援 Java 手機平台架構

在硬體平台上執行一個即時作業系統，目前較為常見的可能包括 SymbianOS、Windows Mobile、PalmOS 以及 Linux 等。在作業系統上層為應用程式，原生的應用程式 (Native AP：例如以 C 語言寫的程式) 直接利用作業系統提供的相關 API 來執行；而 JVM 亦被視為應用程式的一種， Java 的應用程式在執行時，

則執行在已經移植好的 JVM 上。

系統運作時，原生應用程式和 Java 應用程式基本上是獨立分開，當原生應用程式在執行時，程式的執行結果即直接顯示在螢幕上；同樣地，當 Java 應用程式在執行時，執行的結果也會透過 JVM 顯示在螢幕上。這樣的系統架構，畫面的呈現各自由各個應用程式負責，顯示出執行的結果。因此，這樣的架構並無法形成一個風格一致的畫面呈現方式。

3-3 本篇架構

本篇提出以介面描述語法來呈現使用者操作環境，因此對於任何希望顯示在螢幕上的畫面，都要能夠適當的呈現出來。所以我們的設計考量在於，如何將畫面的呈現以一種一致的方式呈現出來，因此我們將原來支援 Java 功能的平台修改成圖 10 的架構：

圖 10：本篇提出的系統平台運作架構

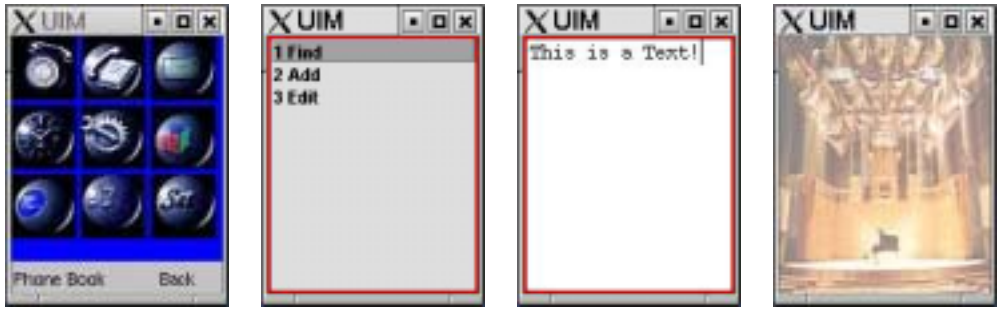
和原來平台架構有所不同的差異在於所有有關畫面的處理，都是以 JVM 為主。原生應用程式在執行時，其原本欲顯示在螢幕畫面上的輸出結果，在更改的架構下現在則透過 Java 所定義的 Java Native Interface (JNI) 經由 JVM 來呈現。另外在 JVM 的上層，設計一個介面管理程式 (User Interface Manager)，將使用者介面相關的動作統一藉由使用者管理員來處理。使用者管理員負責提供在手機環境應用上，可能用到的畫面方式。而原本 Java 應用程式的運作方式不

變，只是在執行結果的畫面輸出上，藉由介面管理員來呈現。這樣的架構，可達到本篇提出以介面描述語來呈現使用者操作環境的一致效果，又無需過多的修改而可執行現存的應用程式。

這樣的運作模式類似在 Linux 上執行 X-Window 的圖形介面環境。X-Window 運用主從式架構 (client-server) 的方式，提供相關的圖形程式庫給程式開發者使用，讓應用程式的外觀顯示呈現出同一種樣式。目前在 Linux 上較為常見的圖形程式庫包括建立在 Qt 的 KDE 和使用 GTK 為基礎的 GNOME 這兩種圖形介面環境。和本篇不同的地方在於，本篇是利用介面管理員來處理畫面而不是使用程式庫。

3-4 使用者介面管理員 (UIM) 設計

在本篇提出的系統平台運作架構中，使用者介面管理員最主要負責所有顯示畫面的呈現。基本上，由於手機是一個特定目的應用的環境，運作的範圍有限，所以我們將各種可能的操作介面概略分為幾類：第一種是圖片的畫面，畫面最主要是顯示各種圖形；第二種是文字的畫面，畫面最主要是呈現各種跟文字相關的訊息；第三種是有選項的畫面，畫面最主要是有多個選項讓使用者來選取，圖 11 列出這幾種畫面。



icon 選項畫面　條列式選項畫面　文字畫面　圖片畫面

圖 11：各種類型的畫面

另外，在手機的應用環境上，基本上每個畫面的作用都是事先規畫好的固定

方式，而且各個畫面之間相連結關係，是依據上一層的選項往下一層選取，類似父-子（parent-child）的關係，所以我們採用樹狀的階層式架構來表示這種相連結的結構。在各個畫面的設計上，仿效樹狀結構的節點，依照各種不同畫面的類別，記錄著父節點（parent node）或是子節點（child node）相關的連結資料，讓各個畫面即依照這樣的連結方式，因使用者不同選取的選項而呈現出相關的畫面。

UIM 在整個設計上利用 XML 的標籤（tag）方式，將每個畫面依據上、下階層不同的選項關係相互連結，形成樹狀結構。透過這樣的運作方式，再將畫面設計成風格一致性的顯示效果，即可讓手機的整個操作環境呈現出風格化樣式的感覺。

3-5 介面描述語法設計

在本篇提出的系統平台運作架構中，使用者介面管理員（UIM）依據介面描述的方式而呈現出相關的畫面。在介面描述語法的設計上，我們定義了 menu，icon，listItem，fnComponent 等 4 種標籤，然後因為不同畫面的型態而需要不同的參數。以下就各個不同畫面的作用一一加以敘述：

1. menu：呈現出包含有選項的畫面。主要的參數有 name，type，previous，及 next。額外的選項參數有 leftSoftkey，middleSoftkey，rightSoftkey，topLabel。

name	定義此畫面的名字
type	畫面的種類
previous	連結前一個相關的畫面
next	連結下一個相關的畫面

表 1：menu 畫面的主要參數

leftSoftkey	畫面左下方 “ 軟鍵 ” 的顯示
middleSoftkey	畫面中下方 “ 軟鍵 ” 的顯示
rightSoftkey	畫面右下方 “ 軟鍵 ” 的顯示
topLabel	畫面上方提示作用的顯示

表 2：menu 畫面額外的選項參數

另外 type 又分為 3 種：

IMAGE_CONTAINER	包含一個以上圖片的畫面
ICON_CONTAINER	包含各種 icon 選項的畫面
LIST	包含各種條列式選項的畫面

表 3：menu 畫面的型態

而不同型態所需的額外參數有所差別：

menu 的型態	所需的相關參數
IMAGE_CONTAINER	image：指定相關的圖片
ICON_CONTAINER	icons：畫面包含的 icon 數目
LIST	lisTems：條列式選項的數目

表 4：menu 畫面不同型態所需的額外參數

舉例而言：

<menu name="ISP" type="IMAGE_CONTAINER_PANEL" image="idle.gif"

previous = "ISP" next="MainMenu"> 表示這是名為 ISP 的包含多個圖片的畫面，其圖片為 idle.gif，上一個畫面為自己，下一個畫面指到名為 MainMenu 的畫面。

2. icon：呈現在 ICON_CONTAINER 的 menu 畫面中的 icon 選項。參數包含 name，image，posX，posY，parent，link。

name	定義此 icon 的名字
image	此 icon 表示的圖片
posX, posy	此 icon 在 menu 畫面中的位置
parent	此 icon 所屬的 menu 畫面
link	選取此 icon 所連結到的畫面

表 5：icon 畫面的參數

3. listItem：呈現在 LIST 的 menu 畫面中的條列式選項畫面。參數包含 parent，name，link。

name	定義此 listItem 的名字
parent	此 listItem 所屬的 menu 畫面
link	選取此 listItem 所連結到的畫面

表 6：listItem 畫面的參數

4. fnComponent ：呈現應用程式執行後所需的畫面。主要參數包含 name，type，previous。基本上，應用程式執行的結果所顯示出的畫面有兩種可能，文字相關訊息或是圖片相關的顯示。因此我們定義了 TEXT 及 IMAGE 兩種型態作為應用程式執行所需的畫面結果。

name	定義應用程式執行所需畫面的名字
type	畫面的種類
previous	連結前一個相關的畫面
image	IMAGE 畫面所表示的圖片

表 7：fnComponent 畫面的參數

另外，除了這些畫面的定義之外，還定義了一個 global 的標籤（tag）作為整個介面描述呈現所需的設定。

Panels	整個介面描述所需的畫面數目
Style	所定義好的介面風格
BgColor	所做用的背景顏色

表 8：global 標籤的參數

最後，我們舉出一個例子做為參考。圖 12 的介面描述檔描述著”主選單”是以小圖示的方式來呈現，而在”電話簿”的選項下是用條列式的選單畫面來表示，在”尋找”的這個選項上，最後執行是以文字畫面來呈現，另外在”新增”的這個選項上，執行的畫面是圖片的結果。

```
<global panels="13" style="telephony">Global setting </global>

<menu name="ISP" type="IMAGE_CONTAINER_PANEL" image="image/idle.gif" previous="ISP" next="MainMenu"> ISP
 <menu name="MainMenu" type="ICON_CONTAINER_PANEL" leftSoftkey="left" middleSoftkey="ok" rightSoftkey="Back" icons="9"
  previous="ISP">Main Menu
  <icon name="Phone Book" image="image/phone/address.png" posX="0" posY="0" parent="MainMenu" link="PhoneMenu"> Phone icon
  <menu name="PhoneMenu" type="LIST_PANEL" topLeftLabel="PhoneMenu" leftSoftkey="left" listItems="3" previous="MainMenu">
  <listItem parent="PhoneMenu" name="1 Find" link="1 Find"> Find listItem
     <fnComponent name="1 Find" type="TEXT_PANEL" previous="PhoneMenu"> Find Component </fnComponent>
   </listItem>
   <listItem parent="PhoneMenu" name="2 Add" link="2 Add"> Add listItem
     <fnComponent name="2 Add" type="IMAGE_PANEL" image="image/phone/image_test.jpg" previous="PhoneMenu"> Add Component
     </fnComponent>
   </listItem>
   <listItem parent="PhoneMenu" name="3 Edit" link="3 Edit"> Edit listItem
     <menu name="3 Edit" type="LIST_PANEL" topLeftLabel="EditMenu" listItems="2" previous="PhoneMenu"> Edit Component
       <listItem parent="3 Edit" name="EditTest1" link="EditTest1"> Edit Test 1
         <fnComponent name="EditTest1" type="TEXT_PANEL" previous="3 Edit"> Edit Test 1 Component</fnComponent>
       </listITEM>
       <listItem parent="3 Edit" name="EditTest2" link="EditTest2"> Edit Test 2
         <fnComponent name="EditTest2" type="IMAGE_PANEL" image="image/test.jpg" previous="3 Edit"> Edit Test 2 Component
        </fnComponent>
       </listItem>
     </menu>
   </listItem>
  </menu>
 </icon>
```

圖 12：介面描述檔的範例

第四章 系統實作

4-1 概述

本篇是以 Java 語言來實作，主要是利用 Java 語言跨平台的特性，開發出一套可在任何支援 JVM 平台上執行，而不需做修改的程式。另外，考量到本篇應用在 kaffe 這個開放原始碼計劃的 JVM 上所支援的功能及在手機嵌入式系統環境上硬體的限制，本篇實作採用 Java Abstract Window Toolkit (AWT) 的類別函式庫而不是 Java Swing 的類別函式庫。

Java 的類別函式庫 AWT 主要提供基本 GUI 圖形介面的元件，在本篇實作中比較需要注意到的是 Java AWT 的 Event Model。不同於一般 Java 程式的應用，由於手機的操作環境是一個以按鈕為主的操作介面，因而在事件驅動或是事件轉換中，程式主體必需處理元件事件焦點 (Focus) 的轉移，而不是藉由滑鼠的點選。

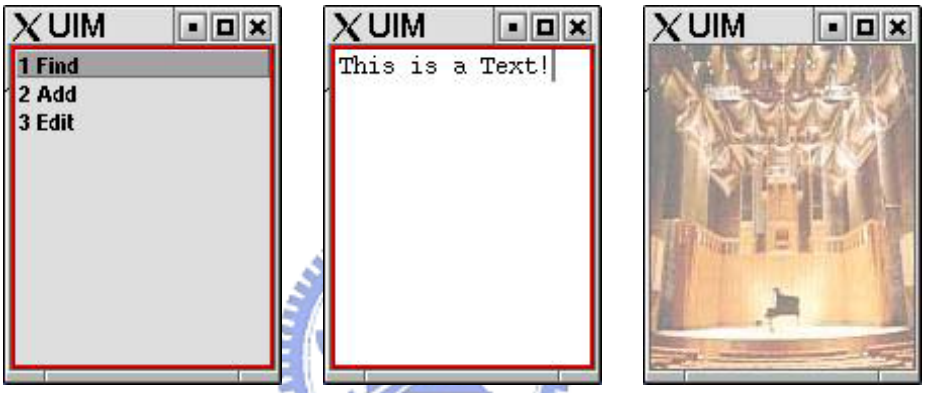
本篇使用 kaffe 做為 JVM，目前最新版本為 February 18, 2004 所發佈的 1.1.4 開發版本 (kaffe.org)。本篇選用 Kaffe 作為實作的平台而不是 Sun 的 JVM，主要的考量在於 J2ME 在整個行動電話的開發環境上受限於 Configuration 及 Profile 上的定義。在整個開發的過程中，無法超出 Configuration 及 Profile 定義的範圍，因而造成許多限制。另外，由於 kaffe 是一個開放原始碼的計劃，將 kaffe 移植到嵌入式的環境中可以把不需要的功能加以移除，使得資源能更有效的利用。

4-2 使用者管理員 (UIM) 實作

UIM 的實作主要包含三個部份：各種畫面的類別建構子 (constructor) 及相關的類別方法函式 (method)，介面描述語的剖析器 (parser)，及鍵按處理程式 (key listener)。

在第三章系統設計中提到，由於手機是一個特定目的的應用環境，所以我們

將可能運用範圍的畫面分為三種：第一種是圖片的畫面，畫面最主要是顯示各種圖形；第二種是文字的畫面，畫面最主要是呈現各種跟文字相關的訊息；第三種是有選項的畫面，畫面最主要是有多個選項讓使用者來選取。我們分別將這幾種定義成相關的類別 ImagePanel，ListPanel 及 TextPanel，這些類別的主體就是 Java AWT 的元件。實作時，這三種類別各自繼承 AWT 的 Panel 元件，在 ListPanel 內定義 AWT 的 List 元件，在 TextPanel 內定義 AWT 的 TextArea 元件，及在 ImagePanel 內定義 Image 的元件。顯示出的畫面如圖 13。



ListPanel　　TextPanel　　ImagePanel

圖 13：三種類別的畫面

另外還定義了繼承 Java Canvas 的 Icon 類別，作為在選單畫面中的選項的小圖示，如圖 14。

圖 14：包含 9 個 Icon 類別的畫面

在類別方法函式部份，最主要的就是如何在整個樹狀結構的連結中找出相對

應的畫面。我們利用 Java AWT 元件中繼承自 Component 的類別方法函式 setName() 和 getName()，設定及取得定義出的物件名稱。再利用 Java LayoutManger 將指定的物件顯示出來。

介面描述語的剖析器部份，程式載入介面描述檔後，剖析器根據介面描述語法定義的相關標籤，呼叫相關的建構子建構出相關的類別，並依據標籤語法定義的參數意義，建構出整個樹狀結構。例如：

```
<menu name="PhoneMenu" type="LIST_PANEL" listItems="3" previous="MainMenu">
```

剖析器即根據 LIST_PANEL 型態，呼叫 ListPanel 的建構子，並設定相關的類別方法函式，建構出名為 PhoneMenu，前一個畫面為 MainMenu 且包含 3 個選項的畫面。

手機目前是一個以按鈕操作的使用環境，所以我們利用 Java AWT 所提供的 KeyListener 來擷取任何對鍵盤按鈕的動作。其中最主要的就是 AWT 元件在事件驅動或是事件轉移時，由於在我們定義的類別物件中的主體是各類別的 AWT 元件，如 ListPanel 中的 List，而不是 ListPanel 本身，所以 UIM 在處理事件轉移時必需將元件的焦點設定為各類別的 AWT 元件。

UIM 程式的整個執行流程如圖 15 所示。

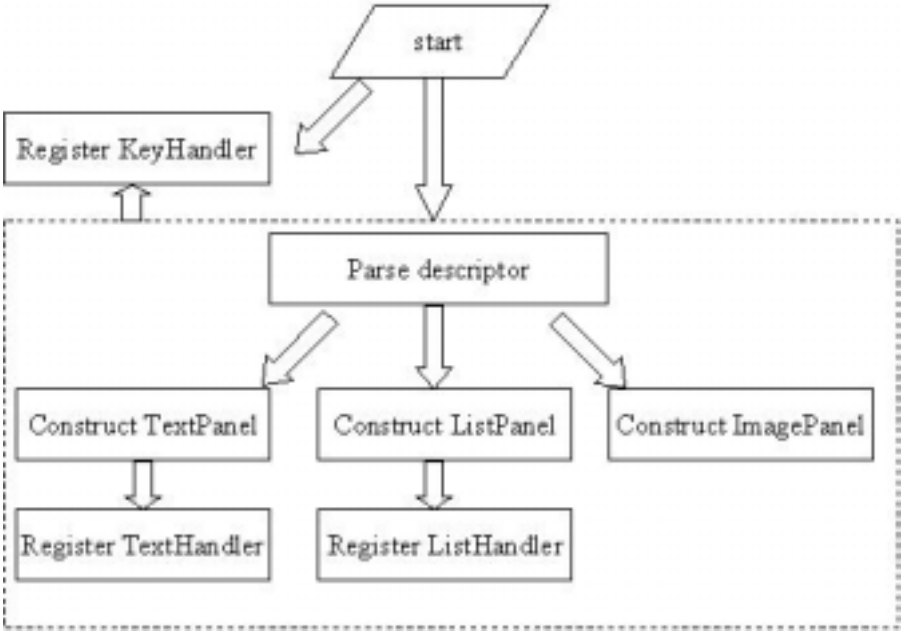


圖 15：UIM 執行流程

目前整個系統的模擬環境平台是將 kaffe 安裝在 Linux 。圖 16 為程式執行在主選單畫面的情形。

圖 16：程式執行的情形

4-3 介面描述語編輯器實作

本篇提出的介面描述語法是利用 XML 的標籤（tag）方式，基本上這是一個文字檔，所以可以利用任何文字編輯器來編輯。但是由於一些標籤定義上的參數

過於繁瑣，在編輯上可能產生錯誤或是不便，所以我們另外實作了關於本篇定義的介面描述語法的編輯器。

介面描述檔和程式本身的執行是獨立分開，不受限於 kaffe 支援的功能，所以我們利用 Java 提供功能較強大的 Swing 類別函式庫來實作。在第三章系統設計-介面描語設計的部份中，我們提到整個介面描述語是以樹狀結構將所有的畫面連結起來，所以我們利用 Java Swing 提供的 JTree 類別來實作介描述語編輯器。JTree 的相關類別中，提供了 insertNodeInto()、removeNodeFrom() 等類別方法函式讓程式開發者可以方便的在設定的 JTree 中增加或移除任何節點。

在實作上，對於任何要增加節點的動作，我們提供了在介面描語上所定義的畫面型態，而根據不同的畫面型態在增加的節點上會顯示出不同型態所需要的必要參數，讓使用者變更設定。最後，則根據使用者所設定的各個節點型態及相關的參數設定，輸出成所設計的介面描述檔。圖 17 顯示出介面描述編輯器的執行情形。

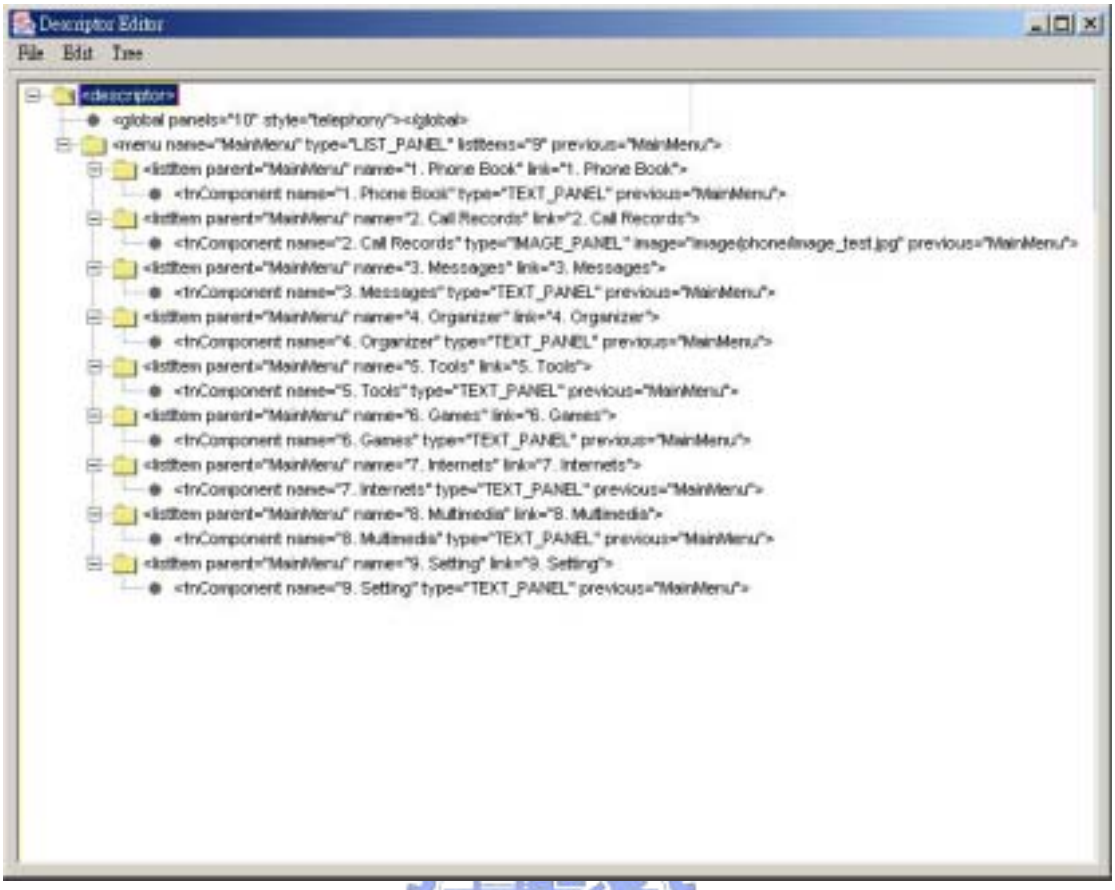


圖 17：介面描述編輯器的執行

4-4 範例

在本節中，我們呈現出幾種不同風格顯示的手機操作環境。強調出只需透過不同的介面描述檔，就能讓手機的操作環境呈現出不一樣的感覺，甚至透過 JVM 跨平台的特性，也能讓不同的手機硬體平台呈現出相同類似的操作介面，表達出本篇所提的概念。

相較於目前的手機設計所能變換的只是背景顏色或是背景圖片，本篇所提出的方法可以改變介面的呈現方式為純文字選項或是圖形選項，也可以將手機選項的小圖示更改成不同的樣式，這樣的做法所表現出的就是讓使用者能自由的將手機的操作環境更改成自己喜歡的風格樣式，達到徹底的個人風格化，而不只是受限於手機廠商所設計成的樣子。

範例 1 呈現的是純文字風格的手機操作環境。早期手機受限於硬體功能，無

法呈現出圖形化的介面，假設現在有的使用者對於目前以圖形化顯示的手機感到過於雜亂，只想以簡單的介面來使用手機，就可以將整個操作環境變更成這樣的純文字風格，又可以使用現在手機所有硬體支援的功能。

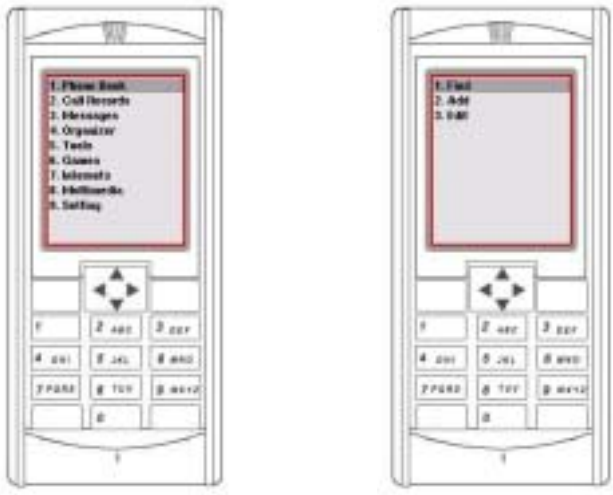


圖 18：純文字風格的手機操作環境

範例 2 呈現的是目前手機常見的圖形化操作環境。手機中的主選單是以各種小圖示來做為各個選項的呈現，讓使用者選取；而在子選單中另外以條列式方式來顯示。

圖 19：目前手機常見的圖形化操作環境

範例 3 希望呈現的是手機的造型設計和內容風格一致的樣式。假設我們有一

款外型線條分明、外觀樣式簡單的手機造型，搭配著內容風格是以簡單線條為主的圖式選項，呈現出“內”“外”一致的感覺。甚至可將內容的顯示文字替代成與風格相符的有意義文字而不是傳統手機上使用的”電話簿”“訊息選單”等等，讓整個手機呈現出完全的個人化風格。

圖 20：風格一致的手機設計

圖 21 展示的是 siemens Xelibri 系列，幾款造型較為特殊的手機。像這樣新穎的外觀設計樣式，改變傳統原來使用者對手機的想像，若只是以傳統手機的內容方式來做為搭配，實際上對追求個人化與眾不同風格的使用者來說，效果上就無法突顯。若能改以相配外觀造型的使用者操作介面，甚到能讓使用者自己設計需要的介面環境，這樣就更符合使用者的需求。

圖 21：幾款 siemens Xelibri 系列手機（來源：www.xelibri.com）

最後，我們想展現的是何謂風格化內容。一般來講，在圖形化的手機操作環

境中，對於代表選項的小圖示最為醒目，但除了這些小圖示之外，文字性的描述也十分重要。對於一個我們所認定的風格化設計內容需要包括小圖示的顯示搭配著相對應的文字性描述。舉一個慣用的例子來說明：在現在的手機上，對於一個電話的圖示  ，常代表著與電話相關的文字描述，如 ” 電話簿 ” 或是 ” 通話紀錄 ” 等。任何人一看到這樣的圖示選項，大概都不會會錯意，我們稱這種為電信風格。而相較於這種原來在手機上使用的電信風格，如果我們嘗試著發展另一種在手機環境上的風格介面，除了必需考慮到相對應的意義之外，還需要顧及到這樣的描述所代表的後設語言是否充份的表達出原來的意思。圖 22 的範例呈現的是我們嘗試發展的音樂風格選單。舉例來說，相對於一般在手機上使用 ” 通話記錄 ” 來表示，我們以 ” 時間表 ” 來代表同樣的概念。

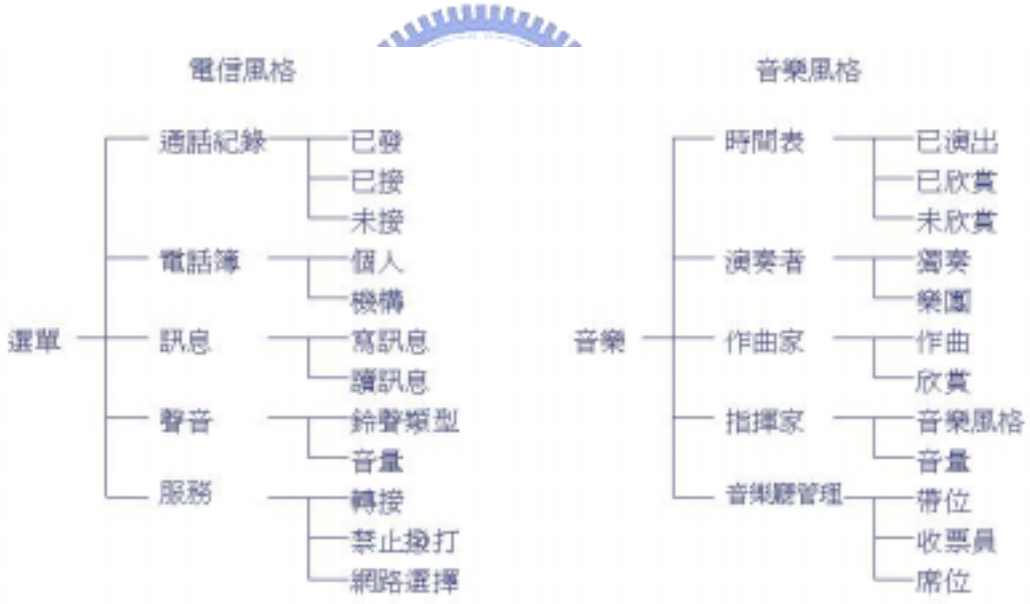


圖 22：音樂風格選單範例

第五章 跨平台的探討

5-1 概述

本篇提出的架構中，最主要是利用 Java 跨平台的特性，達到在任何支援 JVM 手機的硬體平台上以介面描述語來呈現出各種樣式的畫面。而系統在運作上，除了最上層的使用者介面之外，事件的處理主要還是由底層的機制來負責。因此如何將上層的介面環境和底層的系統平台相結合，達到同步的動作，是本章節所要討論的主題。

5-2 嵌入式作業系統

目前在眾多嵌入式作業系統中有許多開放原始碼計劃，其中 Embedded Configurable Operating System (eCos) 由 RedHat 所主導，其主要設計的概念是將各個功能以模組化的方式讓系統設計者可依不同的需要加以設定，最後產生一個符合特定目的所需的嵌入式作業系統。和一般作業系統不同的地方在於這種可規劃式的模組方式，讓系統設計者只需替換不同的模組元件即可更換出不同的需求。

eCos 支援相當多的硬體平台架構，目前包含 ARM, Hitachi H8300, Intel x86, MIPS, Matsushita AM3x, Motorola 68k, PowerPC, SuperH, SPARC 和 NEC V8xx 等等，而這些不同的硬體平台和各部份相關的程式庫架構在系統核心的硬體抽象描述層(Hardware Abstraction Layer)。當系統設計者在開發各個不同硬體平台時，只需以不同的模組元件來替代既可。

eCos 提供系統開發者許多的功能來開發在嵌入式系統上的應用，其核心包括了各種排程的策略 (scheduling policies)、中斷的處理、同步的機制和執行緒的控制等等。

在排程者 (scheduler) 的設計上，核心提供了 bitmap 和 multi-level queue 兩種方式：bitmap scheduler 提供了快速且可決定性的排序處理，其主

要是系統提供 32 個 priority queue 讓程式開發者使用，但每個 queue 只准許存在一個執行緒；而 multi-level queue 則准許在相同的優先權下能在多個執行緒。

在同步的機制上，核心提供了 mutex、condition variable、counting semaphore、mail box 和 event flag 五種不同的做法：mutex 可以讓不同的執行緒安全的共用一個資源，condition variable 可以讓執行緒在某些修件下開始執行，counting semaphore 可用來指示某特定事件的發生，mail box 允許事件觸發時可交換資料，而 event flag 可使用來等待某些不同事件的觸發。

在中斷處理設計上，核心使用兩種層次（two-level）的方式。當中斷發生時，中斷處理常式（ISR）會儘快的處理相關的設定然後結束讓系統核心繼續能夠處理中斷的發生。而中斷處理常式只能處理少數的核心呼叫，如果中斷處理常式偵測到一個 I/O 動作的完成，它就會呼叫等待的執行緒，由這個執行緒呼叫相關的延遲中斷服務常式（Deferred Service Routine）來執行。利用這種兩層次的方式避免核心程式在中斷發生時造成的同步問題。

另外，除了這些核心設計之外，eCos 也提供了在嵌入式環境上所需要的功能，包括記憶體管理、驅動程式、例外處理及 ISO C 程式庫等等。而在整個開發設計上 eCos 還提供一個開發環境介面讓設計者方便使用。圖 23 為 eCos 開發環境介面執行的情形。

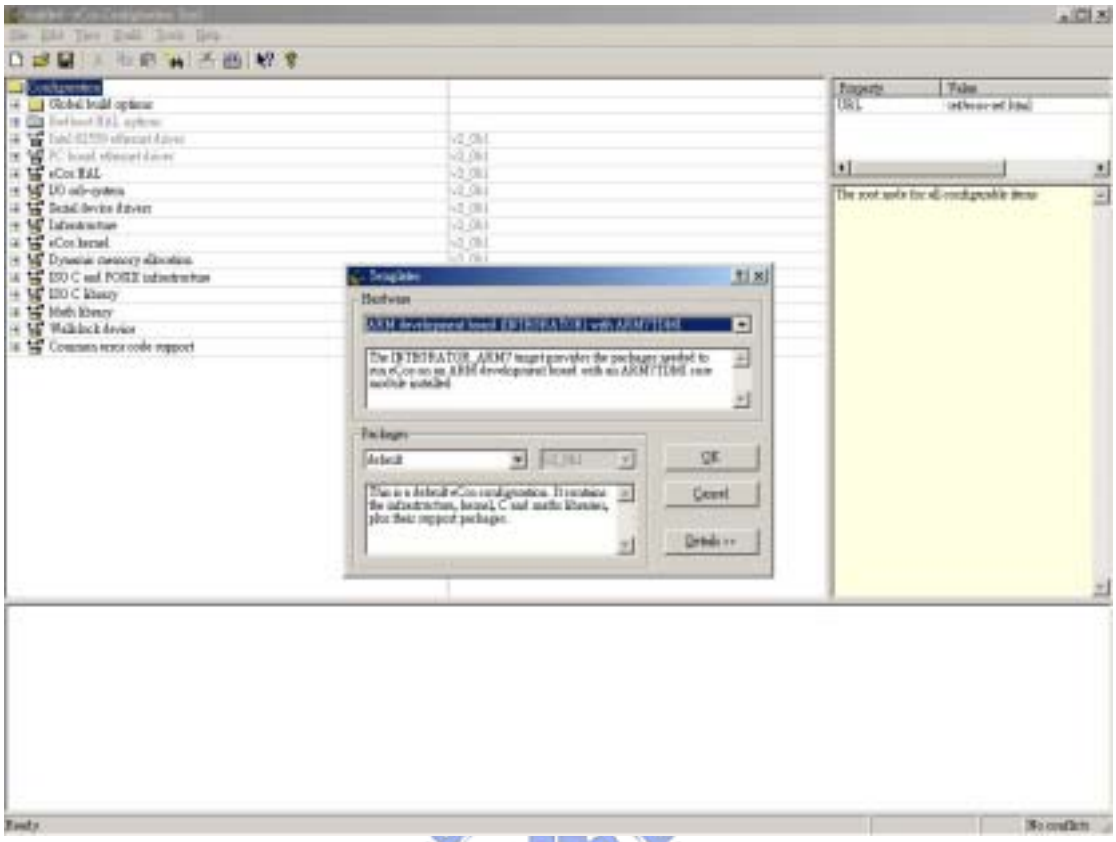


圖 23：eCos 開發環境介面執行的情形

5-3 Java Virtual Machine

在 JVM 相關的開放原始碼計劃中，除了本篇使用的 Kaffe 之外，還有一些計劃在發展中，如 ElectricalFire、Jupiter、SableVM 等等。

ElectricalFire 是 Mozila 計劃下利用 Just-In-Time (JIT) 技術開發的 JVM。JIT 是將 Java 轉成機器碼執行來增加 Java 執行速度的一種方式，其基本的運作方式是 Java 程式執行到之前未使用過的函式時，程式會暫停，直到編譯器將 Java Class 的函式編譯成機器碼之後，再繼續執行。EletricalFire 大部份是以 C++來實作，少數和平台相關的部份由組合語言來完成，目前支援的平台包括 x86 Linux 和 Winodws。

Jupiter 是以模組化和具有擴展性的概念來實作 JVM，其主要是利用類似在 UNIX 下用 pipeline 來執行程式的想法，在定義好的介面之間，將各個模組串接

起執行。圖 24 是 Jupiter 的執行架構。

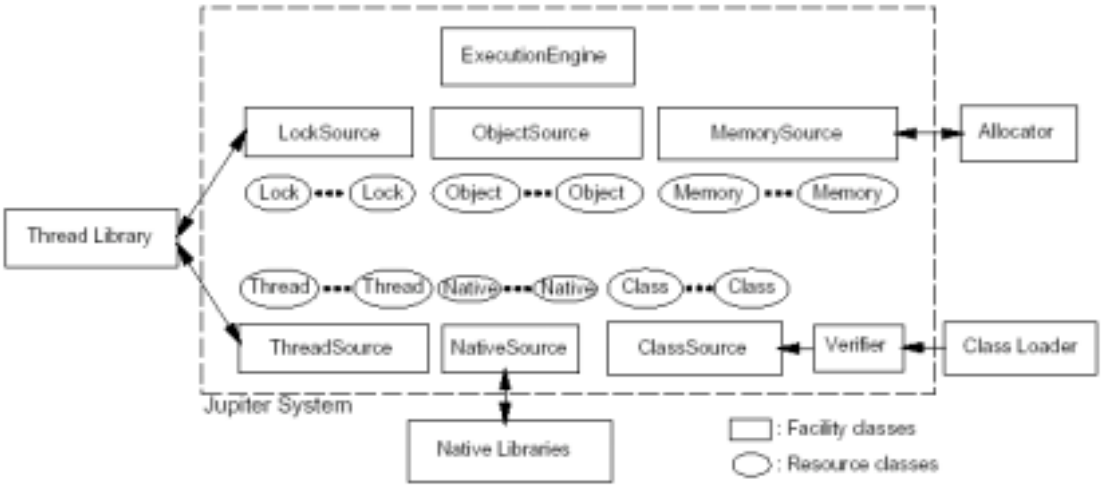


圖 24：Jupiter 執行架構 （來源：www.eecg.toronto.edu/~doylep/jupiter/）

SableVM 是一個以 C 來實作 JVM 在 Linux 執行的研究計劃，其主要目的是希望開發出一個有效率的 VM，在實作上提出以雙向物件配置（bi-directional object instance layout）的方式來增快記憶回收、並將虛擬機器的記憶體依不同需求分成不一樣的區塊等來改進 JVM 的效能。圖 25 是 SableVM 的架構。

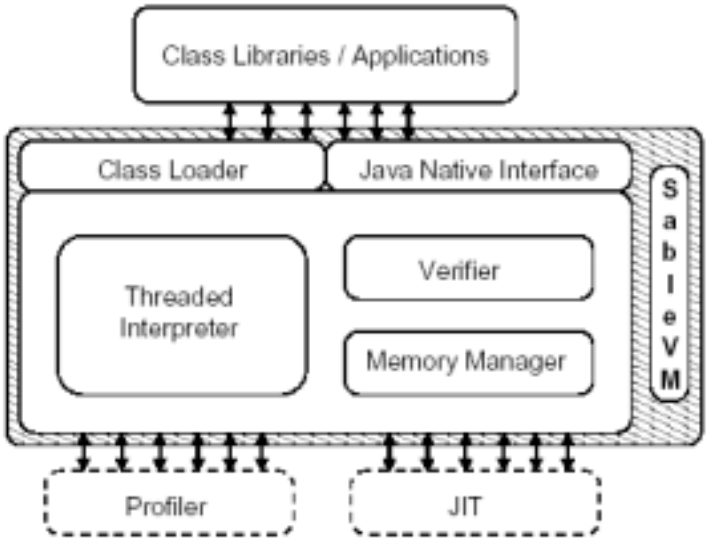


圖 25：SableVM 架構（來源：www.sable.mcgill.ca）

5-4 模擬實作

我們利用狀態機（state machine）來模擬底層的平台機制如何上層的介面

環境如何達到同步的動作。基本上，我們將系統運作時分為三種狀態：第一種是底層的系統在執行的狀態，如處理中斷等；第二種是正常狀態，表示系統在一般的狀態下，任何程式皆可正常執行；第三種是鎖定狀態，如手機螢幕鎖定時的狀態等。運作的流程大致如下：一般情形下系統執行在正常狀態，使用者可以任意操作手機。當有任何事件發生，如對方來電等，系統這時切換到系統狀態處理相關機制，處理完畢之後回到正常狀態。當使用者設定鎖定時，則切換到鎖定狀態，這時使用者無法任意操作手機，直到使用者解除鎖定，系統則恢復到正常態。

第六章 結論

6-1 概述

在科技日新月異的今日，消費性電子產品除了在硬體功能上講求高效能，多功能之外，越來越多的產品研發者開始注意到使用者對產品的 ” 軟性 ” 訴求。不論是在產品外觀上的造型推陳出新，別出用意，或是在操作介面上強調易於使用，容易安裝，這些現象，反應的是設計者開始從使用者的角度來思考產品的研發。

而除了外觀設計新穎，使用容易之外，現在的使用者對消費性電子產品也越來越講求個人化風格。他們希望自己擁有的產品能夠具有獨特性，甚至還可以表現出自己的特色，所以有越來越多的產品設計者開發許多加值性服務，例如手機的可換殼功能、來電鈴聲等等。這些額外的功用，表現的就是使用者想要與眾不同的心態。

6-2 結論

有鑑於這些使用者想要將產品表現出具有自我特色的想法，本篇提出一種在手機上能夠呈現出具有風格化操作介面的應用。透過以 XML 的方式做為介面描述的語法，再運用 Java 跨平台的特性，設計出一種在任何支援 JVM 的手機平台上，能夠呈現出具有風格化的使用者介面。

在設計上，我們利用一個使用者管理員程式 (UIM)，提供各種在手機運用環境上可能使用到的畫面種類，並根據介面描述檔的設計，呈現出相關的內容畫面。而由於介面描述檔是以 XML 方式的文字敘述，所以不論是手機設計者或是使用者本身都可以自行設計出想要的操作介面。

6-3 未來工作

在本篇的實作上，由於缺乏硬體的設備，目前的執行環境是將 kaffe 安裝在

Linux 平台上做模擬。在未來，我們可以嘗試將 kaffe 移植到實際的硬體設備上，在實際的硬體上安裝即時作業系統，如 eCos 或是其它的嵌入式作業系統，讓整個系統架構運作起來，看看實際上執行的效果如何。

另外，由於手機是一個現存的應用環境，存在著很多已經開發好的應用程式，所以如何整合現存的應用程式透過 Java Native Interface (JNI) 在本篇的架構下運作，以節省開發的資源，也是未來需要工作的目標。

最後，除了運用 Java 提供的 AWT 元件之外，如何開發設計更具有風格特色的 GUI 元件，或是在現有的元件之上，增加風格式的設計，也是未來可以努力的方向。

參考文獻

[1] E. Gagnon and L. Hendren, “SableVM: A research framework for the efficient execution of java bytecode”, In Java Virtual Machine Research and Technology Symposium, Monterey, CA, April 2001

[2] Patrick Doyle and Tarek Abdelrahman, “Jupiter: A modular and extensible JVM”, In Proceedings of the Third Annual Workshop on Java for High-Performance Computing, ACM International Conference on Supercomputing, Italy, June 2001

[3] Horst Eidenberger and Christian Breiteneder, “A Framework for User Interface Design in Visual Information Retrieval”, In IEEE Fourth International Symposium on Multimedia Software Engineering, December 2002

[4] Connected Limited Device Configuration (CLDC) Specification, http://java.sun.com

[5] Mobile Information Device Profile (MIDP) v2.0 Specification, http://java.sun.com

[6] Java, http://java.sun.com

[7] XML Core Working Group, “Extensible Markup Language (XML) 1.0 (Third Edition)”, http://www.w3.org/TR/REC-xml, February 2004

[8] UIML, http://www.uiml.org

[9] Symbian, http://www.symbian.com

[10] Kaffe, http://www.kaffe.org/

[11] eCos, http://ecos.sourceware.org/

[12] ElectricalFire, http://www.mozilla.org/projects/ef/

[13] SableVM, http://www.sablevm.org/

[14] Jupiter, http://www.eecg.toronto.edu/~doylep/jupiter/