國立交通大學

資訊學院　資訊學程

碩 士 論 文

GART －圖形化介面/音訊回歸測試系統

GART – GUI/Audio Regression Testing framework

研 究 生：陳冠文

指導教授：黃世昆　教授

中 華 民 國 一 百 零 二 年 六 月

GART － 圖形化介面/音訊回歸測試系統

GART – GUI/Audio Regression Testing framework

研 究 生：陳冠文　　Student：Guan-Wen Chen
指導教授：黃世昆　　Advisor：Dr. Shih-Kun Huang

國 立 交 通 大 學
資訊學院　資訊學程

碩 士 論 文

A Thesis
Submitted to College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master of Science
In

Computer Science

June 2013

Hsinchu, Taiwan, Republic of China

中華民國一百零二年六月

GART － 圖形化介面/音訊回歸測試系統

學生: 陳冠文　　指導教授: 黃世昆

國 立 交 通 大 學　　資 訊 學 院　　資 訊 學 程 碩 士 班

摘要

本論文研製之圖形化介面／音訊回歸測試系統，能夠控制鍵盤／滑鼠並進行播音／錄音，進行最需要人力的介面測試。介面測試最常遇到的問題，就是測試資料在介面更動過後，就需要進行同步的更新；本系統提出的「資訊樹物件探索」(ITOS）方法，能夠改善這個問題，在介面上的物件更動後（包括位置、大小、圖片、文字），測試資料仍然能夠正常運作，因而減少維護測試資料所需的人力。

本系統建置成動態資料庫，將不同的功能放在不同的模組裡，方便使用又具有彈性，能依照不同的需求，加載需要的功能；模組化的資料庫，也減少了本系統的開發及維護工作所需的人力。

GART – GUI/Audio Regression Testing framework

Student: Guan-Wen Chen Advisor：Dr. Shih-Kun Huang

Degree Program of Computer Science
National Chiao Tung University

ABSTRACT

GART provides a regression testing framework that can control keyboard/mouse, play sound and record sound. Through these functions GART can perform GUI regression testing that needs much manual work. The major issue of GUI testing is the test patterns needed to be updated after the GUI changed. We propose to use Information Tree Object Searching (ITOS) method to solve this issue. Using ITOS, the test data can be reused after the objects on the GUI changed. These changes include position, size, pictures, and text. In this way we can reduce the human work of data maintenance.

GART is built as Dynamic Link Library (DLL). It can load different modules on demand. Since it puts the different functions in different DLL modules, GART is easy to develop and maintain.

誌謝

首先感謝我的指導老師－黃世昆老師，給予論文研究方向，並適時提點，讓論文架構更完整。也謝謝實驗室學長姊的幫忙，在每週的研討會上分享研究進度，也讓我對自己的研究有不同的想法。謝謝同儕的砥礪，適時的提醒，才能在二年的時間內完成這篇論文。

謝謝公司的長官與同事，在我研讀學業的二年間，在工作上給予很多的協助，分擔壓力，讓我有更多的時間能完成學業。

感謝一路上支持我的家人，尤其是我最愛的老婆，寶寶剛好在我學業最忙碌時出生，因此很多時候我無暇幫忙照顧寶寶，老婆一手擔起責任，讓我能放心地在學業上用功，回家時看到老婆和寶寶的笑容，是我能完成學業的最大動力。

這篇論文的完成，是很多人在背後支持我的成果，謝謝所有我沒有在此提到，卻一直默默幫忙我的人，謝謝你們！

目錄

表目錄

圖目錄

一、 緒論

1.1 研究動機

軟體測試在產品研發過程中是很重要的一環，其中圖形化使用者介面（GUI）的測試常常需要大量的人力。

目前主要的工作是維護一個音效控制軟體（以下簡稱 VPanel）。VPanel 的主要功能，是提供使用者一個統一的介面，讓使用者能方便地控制 Windows 下有關音效的相關設定。介面測試是由人工進行的；但由於產品眾多，時間有限，即使投入大量的人力，也無法每次都測完所有的功能。我們需要一個能夠測試 GUI 的測試系統，來幫助我們提高測試的效率，以增加程式的穩定度，並減少人力的花費。

1.2 研究目標

建置一個測試系統，將曾經發生過的問題，進行回歸測試，以確保產品的穩定性。測試的目標是有圖形化使用者介面（GUI）的音效控制軟體－VPanel，因此需要模擬鍵盤／滑鼠的操作，控制 GUI 上的物件來複現問題，並且能驗證聲音上的正確性，確認音效的設定沒有問題，產生紀錄檔作為測試的結果。

目前有許多模擬鍵盤／滑鼠的工具，可以簡單地將錄製結果播放出來。由於 VPanel 支援數種不同樣式的介面和 9 種語言，不同的介面和語言其按鈕的大小、位置、圖片、文字也不同，再加上音效設備是動態生成的（詳見 2.2.4），所以各個物件建構的順序也不固定，因此一般的鍵盤／滑鼠工具無法達成我們的要求。我們需要一個能在不同介面上都能成功複現鍵盤／滑鼠操作的工具。

1.3 主要貢獻

提供一個回歸測試系統，能夠針對 GUI 及音訊相關的已知問題，進行回歸測試，增加測試範圍並減少人力支出。

有些軟體的介面設計不只一種樣版，同一個物件在不同樣版上可能有不同的大小、位置、圖片，若有支援多國語言，則物件上的文字也會不同，這讓 GUI 測試需要針對不同的樣版、語言，建置不同的資料庫。本系統提出的「資訊樹物件探索」方法，即使物件的樣式改變了，也能正確地找到目標物件，讓不同樣版、不同語言的 GUI，都能使用相同的測試資料庫，減少了建置和維護測試資料庫的人力成本。

一個需要長時間維護的軟體，最大的問題，是隨著時間的演變，程式變得越來越龐大，越來越複雜，再加上維護人員的更替，而使得維護工作難以順利進行。本系統的模組化設計，讓開發和維護工作變得容易。因為將功能切割，不會因為一個模組的改動而影響到程式的其他部份；而每個模組因為功能單一，程式碼也容易了解，讓維護人員的交替工作變得更簡單。

1.4 論文大綱

本論文共分為六個章節：

第一章 緒論，說明研究動機與研究目標。

第二章 研究背景，介紹與論文研究相關的知識，包括軟體測試、音訊、MFC、JSON 等等。

第三章 相關研究，介紹不同種類的自動腳本軟體。

第四章 研究方法，說明系統架構、測試流程、測試資料的建立、滑鼠／鍵盤的錄製與播放、資訊樹物件探索等等。

第五章 實驗結果與分析，會與類似功能的軟體作比較，並展示實際測試的結果。

第六章 結論與未來展望，為本論文的結論與未來研究方向。

二、研究背景

2.1 軟體測試

軟體測試是在軟體開發過程中不可或缺的一環，其主要目的就是提早發現錯誤，以增加軟體的可靠度和正確性。

2.1.1 介面層次的測試

Andreas／Zeller[1]將介面層次的測試分為 3 種方法：

1 Low-level：以傳送命令的方法來控制物件。

2 System-level：控制整個系統來進行測試，通常會使用虛擬機器。

3 Higher-level：真正找到介面上的控制項，並對其進行操控。

Andreas／Zeller 也提到，介面層次的測試，很容易因為一個小改變（例如：解析度的變化），而造成測試資料失效。

2.1.2 回歸測試

回歸測試（Regression Test）的目的，是確保曾經測試過的部份不會出現錯誤。在新增或修改程式代碼的過程中，可能會影響到其他未被修改到的部份。回歸測試會在代碼更動過後，重複之前全部或部份的測試工作，以確保程式能維持原本的功能。

2.1.3 單元測試

單元是指程式中最小的獨立單位，可能是一個函式（function）或是一個類別（class）……等。單元測試的目的，是確保每一個獨立的單元符合規格且運作正常。單元測試的優點如下：

- 程式容易更改／維護：能確保更改後依然正確，若發生錯誤能更容易找到問題點。
- 可作為說明文件：單元測試定義了每個單元正確與錯誤的用法，而且與程式碼同步，可以作為說明文件使用。
- 可替代軟體設計規格：用單元測試替代文字說明，不僅能清楚了解需求，實作完也可立即測試。

單元測試僅能確保單元正確，無法檢測出整體性或系統層的錯誤。因為需要寫測試碼，因此不可能測出所有情況，也可能因為測試碼錯誤而造成錯誤的結果。而更新程式碼的同時，也需要同步更新單元測試碼，否則單元測試也將失去作用。

2.2 音訊

2.2.1 WAV 格式

WAV（Waveform Audio File Format）是在個人電腦上一種標準的音樂儲存及播放格式，其包含了二部份：標頭（Header）及資料（Data）。標頭的部份定義了取樣率（Sample Rate）、位元深度（Bit Depth）、聲道數目（Channels）、及資料大小。WAV 的資料通常是沒有經過壓縮的。

2.2.2 WAV 音訊分析

在作音訊測試時，必需對音訊資料作分析；因為 WAV 的資料沒有經過壓縮，很適合作為測試的音訊格式。

測試的音源通常為單一的正弦波（Sine wave），並經過傅立葉轉換（Fourier transform），將音訊資料由時域（Time domain）轉到頻域（Frequency domain）作分析。最常見的分析標準是 SNR（Signal-to-noise Ratio）和 THD（Total Harmonic Distortion）。

SNR 是訊號（P_{signal}）和雜訊（P_{noise}）的功率比值，越大代表雜訊越少，音質越好。（圖 1）

THD 是除主頻外的諧波和（$\sum_{i=2}^{\infty} P_i$，P_i 代表不同週期的諧波）和主頻（P_1）的功率比值，越小代表失真越少，音質越好。（圖 2）

$$\text{SNR} = \frac{P_{signal}}{P_{noise}}$$

圖 1　SNR 公式

$$\text{THD} = \frac{\sum_{i=2}^{\infty} P_i}{P_1}$$

圖 2　THD 公式

2.2.3 HD Audio

HD Audio(High Definition Audio)是 Intel 在 2004 年提出的音訊技術[2]。相較於 AC97[3]能提供更高的取樣頻率及更多的聲道;最高能達到 192kHz、32-bits、2 聲道的音質（或是 96kHz、32-bits、8 聲道）。

HD Audio 也規範了 Unsolicited Response，其功能是在使用者將音源線插入音源孔時，硬體會主動發出訊息通知驅動程式，讓驅動程式可以作出相對應的動作（詳見 2.2.4）。

2.2.4 Windows 音訊架構

微軟（Microsoft）在 Windows Vista 之後的作業系統，配合 HD Audio 提出了 Universal Audio Architecture (UAA)的音訊架構[4]。在此架構下，所有符合 UAA 規範 HD Audio 裝置，都只要用統一的驅動程式（Windows Audio Class Driver）就可以正常工作（各家廠商也可以提供自己的驅動程式）。

UAA 為了省電，規範了動態裝置（Dynamic Device）的架構。在此規範下，任何閒置的裝置都應該自動斷電；而在被使用到時（如：開始播音／錄音）能夠自動運作。因此一個音訊裝置會有幾個狀態：

- DEVICE_STATE_ACTIVE：正在工作中。
- DEVICE_STATE_DISABLED：停用狀態。
- DEVICE_STATE_NOTPRESENT：無法使用的裝置。
- DEVICE_STATE_UNPLUGGED：沒有插入音源線。

除了 DEVICE_STATE_ACTIVE 之外的 3 個狀態，都會被視為閒置。如圖 3，左側為音源線未插入的閒置狀態；右側則是插入音源線後，變為使用中的裝態。

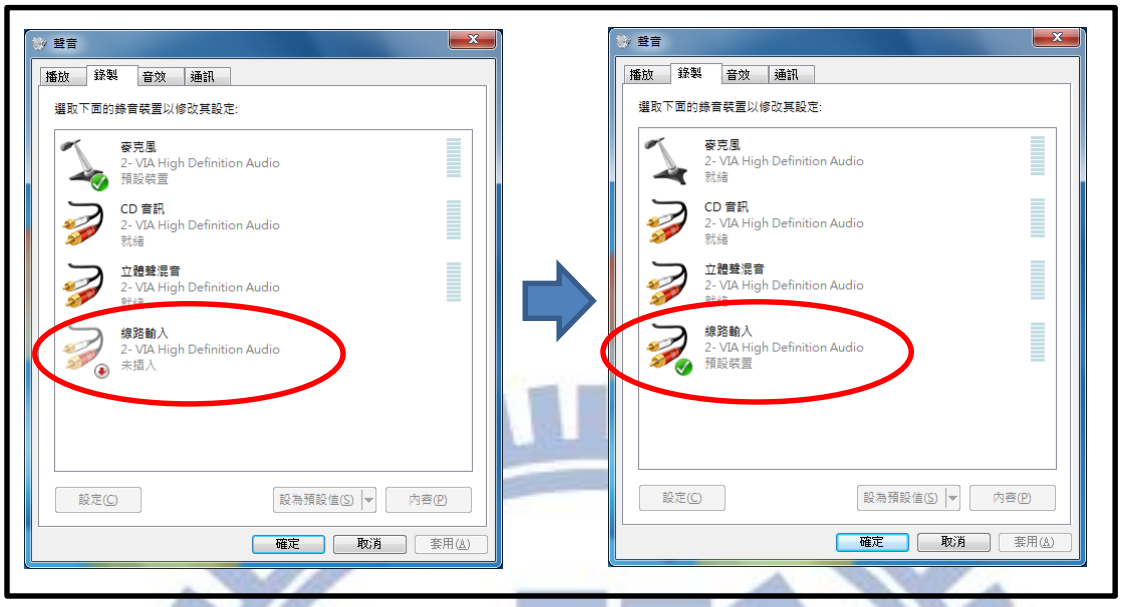


圖 3　動態裝置，左側為閒置，右側為使用中。

2.2.5　WASAPI

WASAPI（Windows Audio Session API）[4]是在 Windows Vista 之後，微軟提供給應用程式的應用程式介面，以用來控制作業系統的音訊內容，包括播音、錄音、音量……等。應用程式也可以透過 WASAPI 來取得音訊裝置的各種資訊，包括狀態、音訊格式、預設裝置……等。

2.3　MFC

MFC（Microsoft Foundation Class）[4]是微軟提供的函式庫，提供了許多樣板以建立應用程式的使用者介面（GUI）。

在 MFC 下的所有視窗（Windows）物件都來自 CWnd 類別。視窗物件可能是對話方塊（Dialog）、表單、控制項（按鈕、下拉式方塊）……等。每一個視窗物件都有其母視窗（Parent Window），最底層的視窗就是 Windows 桌面。一個應用程式稱為一個模組（Module），這個模組的最底層稱為此程式的根視窗（Root Window）。在視窗物件下的其他物件，都稱為子視窗（Child Window），一個視窗物件可能擁有許多子視窗。

2.4 JSON

JSON（Java Script Object Notation）[5]是一種純文字的資料交換語言，其語法包括：

- Object（物件）：

{ String : Value, . . . }

- Array（陣列）：

[Value, . . .]

- Value（值）：

可以是 Object、Array、String、Number、null、true、false

- String（字串）：

"..."（以雙引號包住）

- Number（數字）：

沒有雙引號包住的就是數字。

JSON 的格式簡單，容易瞭解，易於存取，雖然沒有 XML 強大，但其輕量化的特性，可以減少資料的檔案大小，適合用於網路傳輸。

三、 相關研究

3.1 滑鼠自動控制程式

滑鼠是目前個人電腦上最常使用的輸入方式，有時會遇到一些需要重覆點擊滑鼠的操作,因此目前有許多滑鼠自動控制軟體,能自動操控滑鼠，免去使用者自己點擊的麻煩。而各個軟體其控制滑鼠的方式不盡相同，以下以功能作分類，介紹常見的類型。

3.1.1 座標

最常見也最直覺的作法是紀錄滑鼠的座標,如:Mouse Recorder Pro[6]、TinyTask[7]、Point-N-Click[8]、WinMacro[9]。利用錄製／播放，或是編輯腳本的方法，把滑鼠的座標紀錄下來，自動執行時會將滑鼠移到指定的座標並執行點擊動作。優點是簡單、容易操作；缺點是和程式的位置、螢幕解析度有很大的關係，常常因為換了電腦或是移動程式位置後就無法正確執行。

3.1.2 顏色／圖片

這類型的程式，會以特定的顏色或圖片來定位滑鼠位置，如：按鍵精靈[10]、Sikuli[11]、Mouse Macro Recorder[12]。因為是尋找特定的圖片，因此即使程式位置移動、大小改變，仍然能夠找到正確的按鈕並點擊。有些時候圖片是會改變位置的（如遊戲中的物件），利用這個方式，即使圖片移動，仍然能夠找到正確的位置。缺點是如果程式本身有多種佈景主題，在改換之後，就無法找到原本的顏色或圖片了。

3.1.3 物件 ID 及文字

在 Windows 下，利用 GUI 物件的屬性（控制 ID、文字）來尋找特定物件，如：Autoit[13]。這類型的程式主要是利用物件本身的屬性來尋找特定物件，最常用的屬性為控制 ID 和文字。在同一個視窗物件裡，不會有相同控制 ID 的物件，因此可以用來尋找特定 GUI 物件。雖然控制 ID 在同一視窗下不會重覆，但一個應用程式可能有數個視窗，所以可能會找到重覆的控制 ID。若給定的物件屬性相同，Autoit 會給每個相同屬性的物件一個編號，第一個編號為 1，第二個為 2，以此類推，這樣便能找到程式中特定的物件了。

3.2 單元測試框架

單元測試是很有用的工具，專業的程式開發套件（如 Visual Studio[14]）都已經內建單元測試功能。也有一些開放原始碼的單元測試套件，如：JUnit[15][16]、CUnit[17]、CPPUnit[18]等，幾乎所有常用的語言都能找到適合的單元測試框架。

四、研究方法

4.1 系統架構

要驗證音訊，就必需抓取音訊資料。我們的系統是在有音訊晶片的個人電腦上，經由音源線的串接，作本機測試（圖 4）。

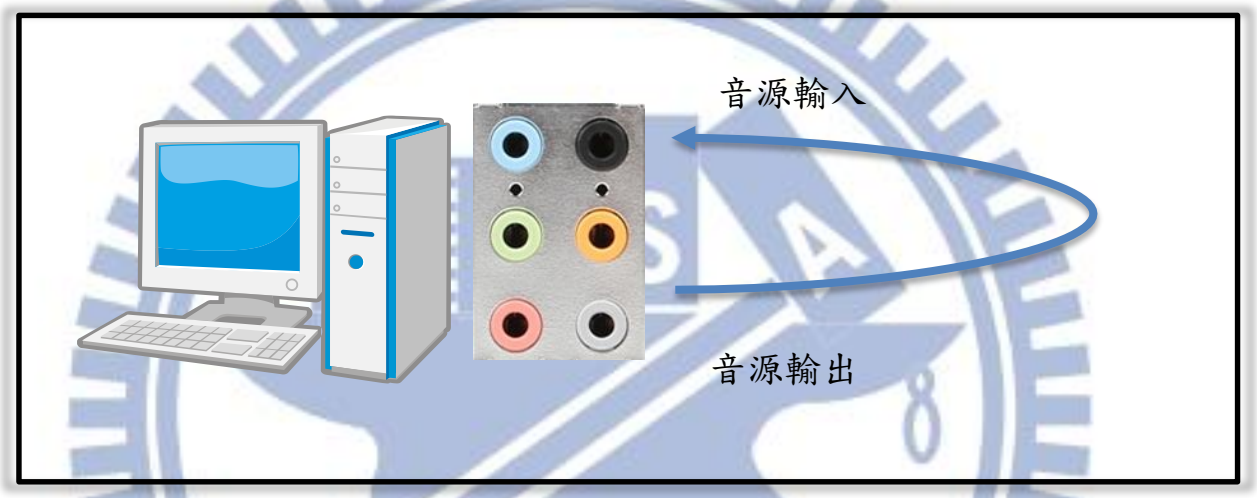


圖 4 硬體線路連接圖

系統需求：Windows Vista＼Windows 7＼Windows8、GART 測試系統、待測軟體。（圖 5）

GART 是一套函式庫，全部建成動態連結檔（DLL），包含主要的 GART.DLL 和其他 DLL 插件（詳見 4.8），再加上測試資料，就是 GART 的主要元件。待測軟體可以是任何有 GUI 的應用程式，在本論文中則是 VPanel。

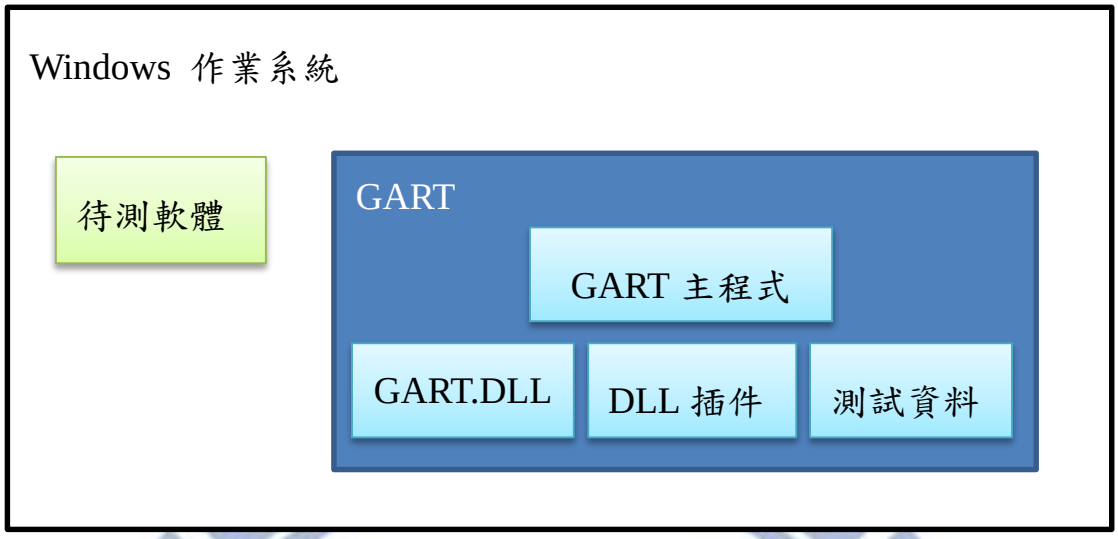


圖 5　系統架構

4.2 測試流程

如圖 6，GART 會先從測試資料庫中取出測試資料，並確認目前的系統符合測試條件（作業系統、音效晶片、VPanel 版本……等），再依照該筆測試資料的參數，設定目前的環境（如：音量大小、預設裝置……等）。初始化完成後就會開始執行測試，並將結果紀錄下來，再去測試資料庫裡尋找下一筆測試資料。

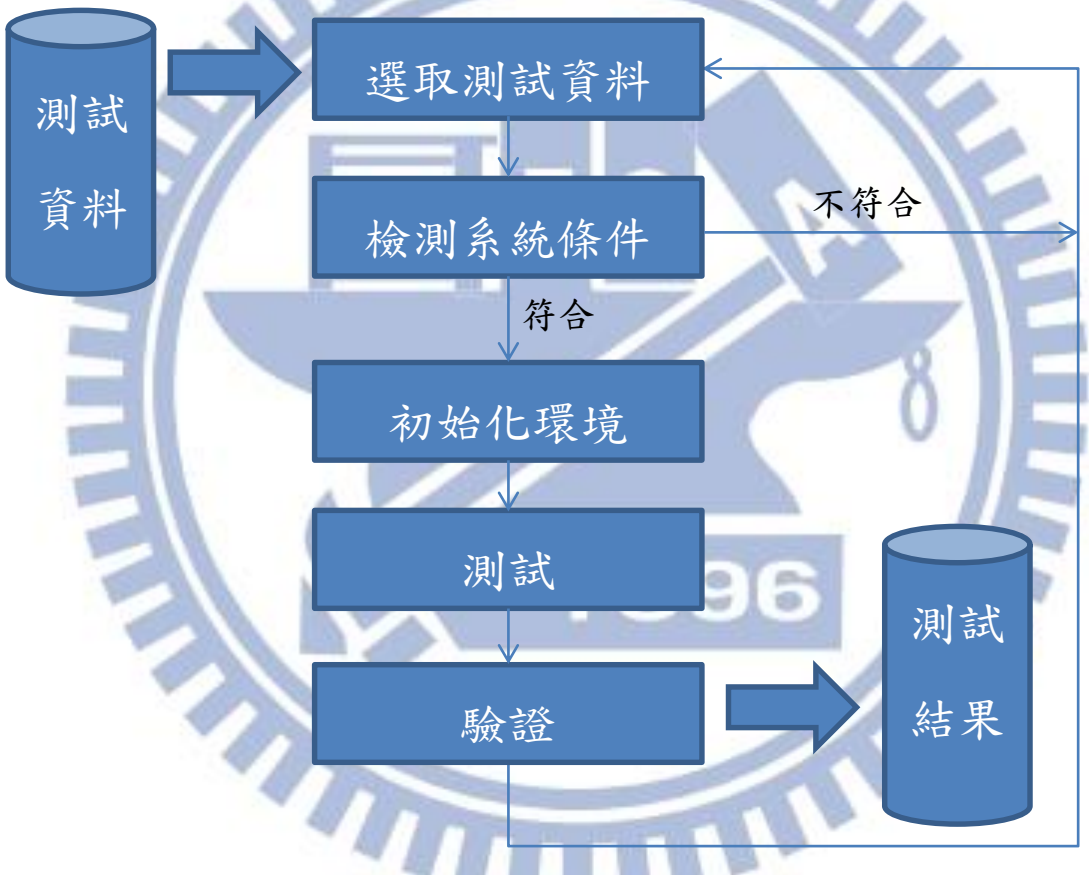


圖 6　測試流程圖

4.3 測試資料需求

GART 的測試資料應至少包含以下部份：

1. 系統資訊：因為產品線眾多，需要知道此筆測試資料執行的作業系統、晶片型號……等。
2. 初始設定：問題產生的初始狀態，如音量、聲道、預設格式……等。
3. 鍵盤／滑鼠的腳本：包含控制鍵盤／滑鼠的訊息，以及目標物件的資訊樹（詳見 4.5）。
4. 資料驗證：當測試動作結束，需要知道結果為正確或錯誤。

測試資料的生成，是由測試人員根據問題回報的資訊，用滑鼠／鍵盤操作一次，由 GART 錄製成腳本；對腳本進行編輯，加入系統及驗證的條件成為測試資料。由於腳本的格式為 JSON，可以利用一般的 JSON 編輯軟體（如：JSON Format[19]、JSON Editor Online[20]）或文字編輯器進行編輯。

4.4 滑鼠／鍵盤錄製

為了讓測試資料的生成更容易，GART 的滑鼠／鍵盤錄製支援以下功能：

1. 抓取鍵盤、滑鼠資訊：會自動濾除不必要的訊息（例如：滑鼠移動），簡化腳本。
2. 取得 GUI 上的物件資訊：取得目前鍵盤／滑鼠焦點上的物件資訊，自動添加在鍵盤／滑鼠的動作之前。詳見 4.5.1。
3. 保留時間戳記：有些問題和時間是相關的，保留時間戳記能更完整地複製問題。

4.5 資訊樹物件探索（ITOS）

VPanel 有數種不同的樣式主題與 9 種多國語;在不同的主題或語言下，相同功能的按鍵可能會有不同的位置、大小、圖片、文字。為了讓相同的測試資料能在不同的樣式主題和多國語言下運行，只錄製滑鼠的點擊位置是不可靠的， GART 使用了資訊樹物件探索（Information Tree Object Searching，ITOS）的方法來解決這個問題。

ITOS 在尋找物件時，除了物件本身的資訊，還利用了和此物件相關的視窗資訊，我們稱這些為資訊樹（Information Tree）。在錄製滑鼠動作（4.5.1）時，會將資訊樹保留下來；在播放腳本（4.5.2）時，則會利用資訊樹來幫助尋找物件。

4.5.1　錄製滑鼠動作

Windows 下的 GUI 物件是階層式的，在同一視窗（CWnd）下的物件各自擁有其唯一的 ID（但可能和其他視窗下的物件重覆）。如圖 7，B_1 和 B_2 同屬於 W_B，因此其 Control ID 在 W_B 中是唯一的；而 B_1 和 B_3 屬於不同的視窗，其 ID 便可以相同。

ITOS 不只抓取物件 ID，會連該物件的視窗資訊全部紀錄下來。視窗資訊抓取物件的母視窗（Parent CWnd），母視窗的母視窗……直到程式的根視窗（Root Window）為止。以圖 7 為例，當按鈕 B_1 被點擊時，除了 B_1 本身的資訊，ITOS 會連 W_B（B_1 的母視窗）、W_A（W_B 的母視窗）和 W_R（B_1 的根視窗）的資訊也一併保留。

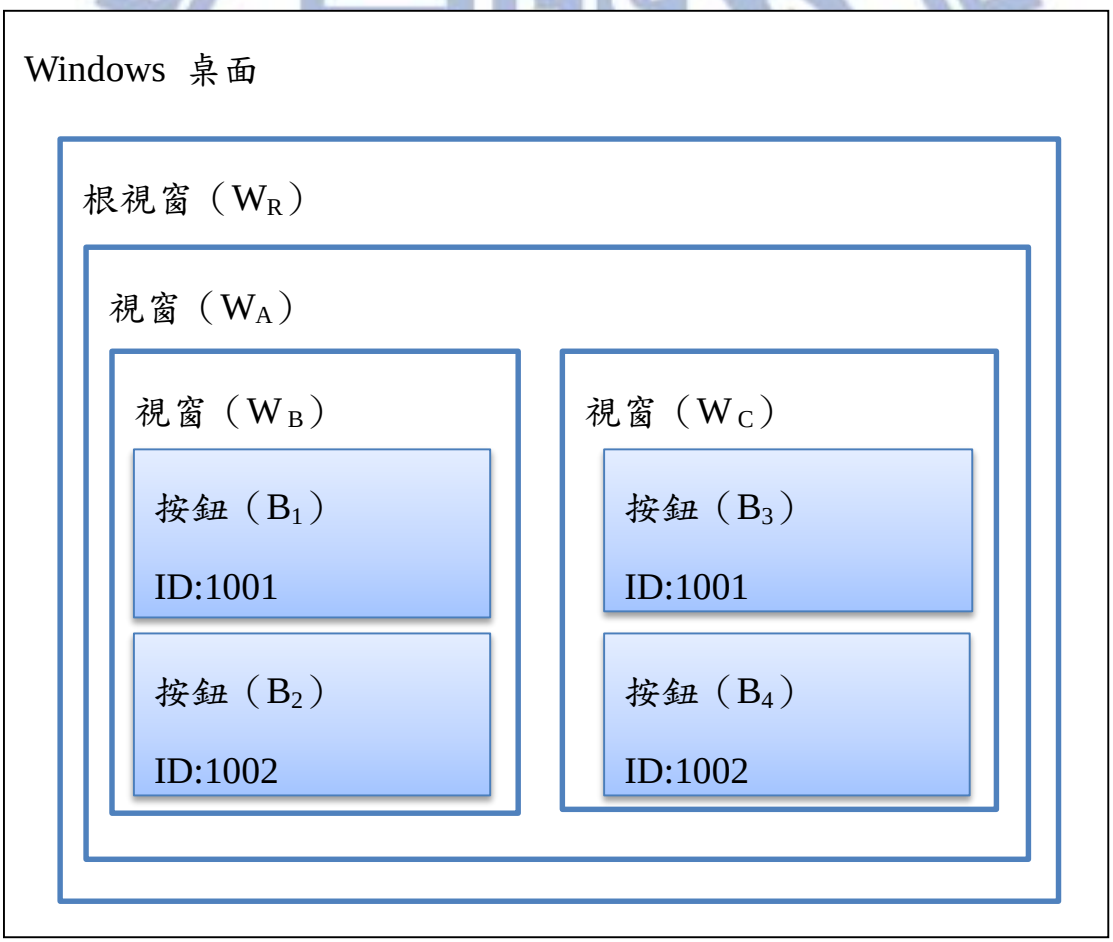


圖 7　視窗階層示意圖

4.5.2 播放腳本

在播放腳本時，ITOS 會根據保留下來的資訊樹，試著找到該物件。由根視窗開始，以深度優先搜尋法，尋找符合的子視窗（Child CWnd），最後找到目標物件。

一個視窗只有一個母視窗，一個母視窗卻可能擁有很多子視窗，而這些子視窗可能擁有同樣的屬性（如圖 7 中，B_1 和 B_3 的資訊樹完全相同）。在遇到這種問題時，Autoit（詳見 3.1.3）為每個相同屬性的物件添加了自定義的編號，但這個編號會隨著視窗中物件生成的順序不同而跟著改變。VPanel 因為需要控制音訊設定，而音訊設備在 UAA（詳見 2.2.4）下是動態生成的，所以 VPanel 中各個視窗物件的生成順序不會固定，會使得 Autoit 的編號在每次執行時都會變動，而無法找到正確的物件。

為了找到正確的物件，ITOS 利用了可視（Visible）這個動態的視窗屬性。因為是 GUI，要讓使用者控制的物件，一定是可視的；不可視的視窗或物件不可能是我們尋找的目標。利用這個屬性，可以精確的找到目標物件。

有些情況下，母視窗下可能有二個同時可視的子視窗，如圖 7，若 W_B 和 W_C 同時為可視的，則無法分辨哪一個是我們的目標。此時需要額外的條件來分辨它們，本系統使用視窗的位置及大小來區分。在這種狀況下，由於固定了位置及大小，就無法通用於不同的樣式主題，只能為不同的樣式主題建置不同的腳本。

4.6 音訊驗證

VPanel 是音訊控制軟體，我們若要驗證其功能是否正常，就必需作播音／錄音的動作，來確保各種音訊設定是否正確。GART 提供了這樣的功能，能夠執行音訊相關的測試。

GART 為了方便作音訊驗證，播音／錄音使用的格式為 WAV(2.2.1)，使用這種格式的好處，是可以直接取得音訊資料而無需作解碼的動作。音訊驗證有四種依據：第一種為波峰值，可驗證音量大小及判斷有無聲音。第二種為主頻，可驗證格式是否正確。第三、四種為 THD／SNR，可以檢驗音質的好壞，以確定音訊是否正常，有無雜音等。主頻／THD／SNR 的檢測只能針對單頻的正弦波，無法對一般的音樂檔作分析，所以 GART 也能產生正弦波的音源。

四種檢測依據的標準值是由人工設定的，可以取全部或部份的檢測依據，並給定範圍值。在分析音訊資料時，GART 會先作快速傅立葉轉換(Fast Fourier Transform，FFT)，在過程中會計算峰值，最後求得主頻／THD／SNR，並和標準值作比對，並將結果紀錄下來。

4.7 紀錄檔

測試過程中會產生紀錄檔，除了測試結果外，有時也需要一些額外的訊息。因為測試資料庫會不斷擴增，所產生的紀錄檔也會相當龐大，為了更方便管理，GART 中為紀錄檔設計了不同的層級：測試失敗、測試錯誤、測試成功、警告、偵錯、註解。當測試完畢，工程師可以依照不同的層級，取出需要的紀錄，不會因為資料過於龐大而找不到需要的資訊。

4.8 模組化設計

為了方便維護及擴展功能，GART 將功能都模組化為動態連結資料庫（DLL），除了主要的 GART.DLL 外，可以加入不同功能的 DLL，讓 GART 擁有不同的功能。

GART 讀取一個 JSON 物件（object）時，其名稱（name）就代表了這個模組的功能；接下來再到設定檔裡去找這個功能所在的模組並呼叫它。

設定檔的格式如下：

```
[{"PlugIn":{
    "Key":"功能名稱",
    "Dll":"Dll 檔名",
    "Function":"處理函式"}}]
```

設定檔在 GART 初始化時就會讀進來，代表 GART 所支援的功能。在讀取 JSON 測試腳本時，會比對腳本和 GART 支援的功能，呼叫對應的處理函式；若不在設定檔內，則會忽略此 JSON 腳本。

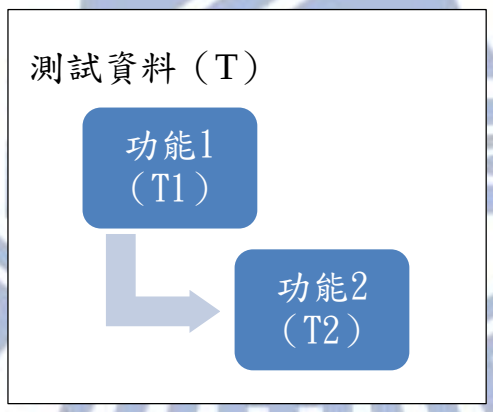


圖 8 測試資料例子

在功能模組內，有時會需要用到其他模組的功能，因此 GART.DLL 有一個回呼函式（Callback Function）：當功能模組遇到無法處理的 JSON 腳本時，可以呼叫回呼函式，讓 GART.DLL 來處理這個腳本。GART.DLL 在處理完後，會再回到原本的功能模組內繼續執行。功能模組也可以選擇忽略無法處理的 JSON 腳本（如：不合法的腳本），端看該功能模組的需求而定。

以圖 8 的測試資料為例，此筆測試資料有功能 1 的腳本，而功能 1 的腳本裡又包含功能 2 的腳本。執行此測試資料的流程如圖 9 所示：

1. 主程式讀取測試資料（T）並傳送給 GRAT.DLL。
2. 判斷測試資料（T1），並呼叫功能 1.DLL。
3. 執行功能，當遇到功能 2 的腳本時，將 T2 傳回給 GART.DLL。
4. 判斷測試資料（T2），並呼叫功能 2.DLL。
5. 執行功能，結束後返回。
6. 返回功能 1.DLL。
7. 繼續執行，結束後返回。
8. 結束此筆測試資料。

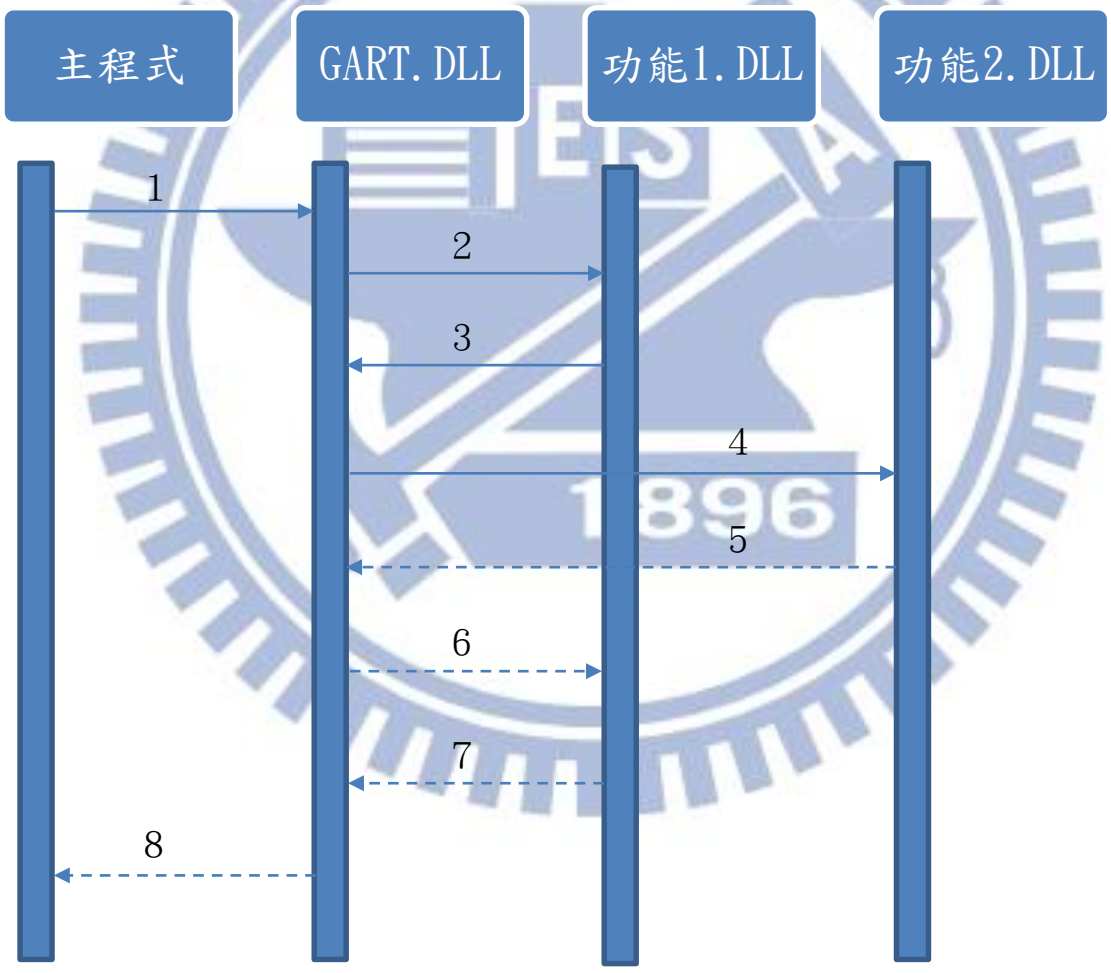
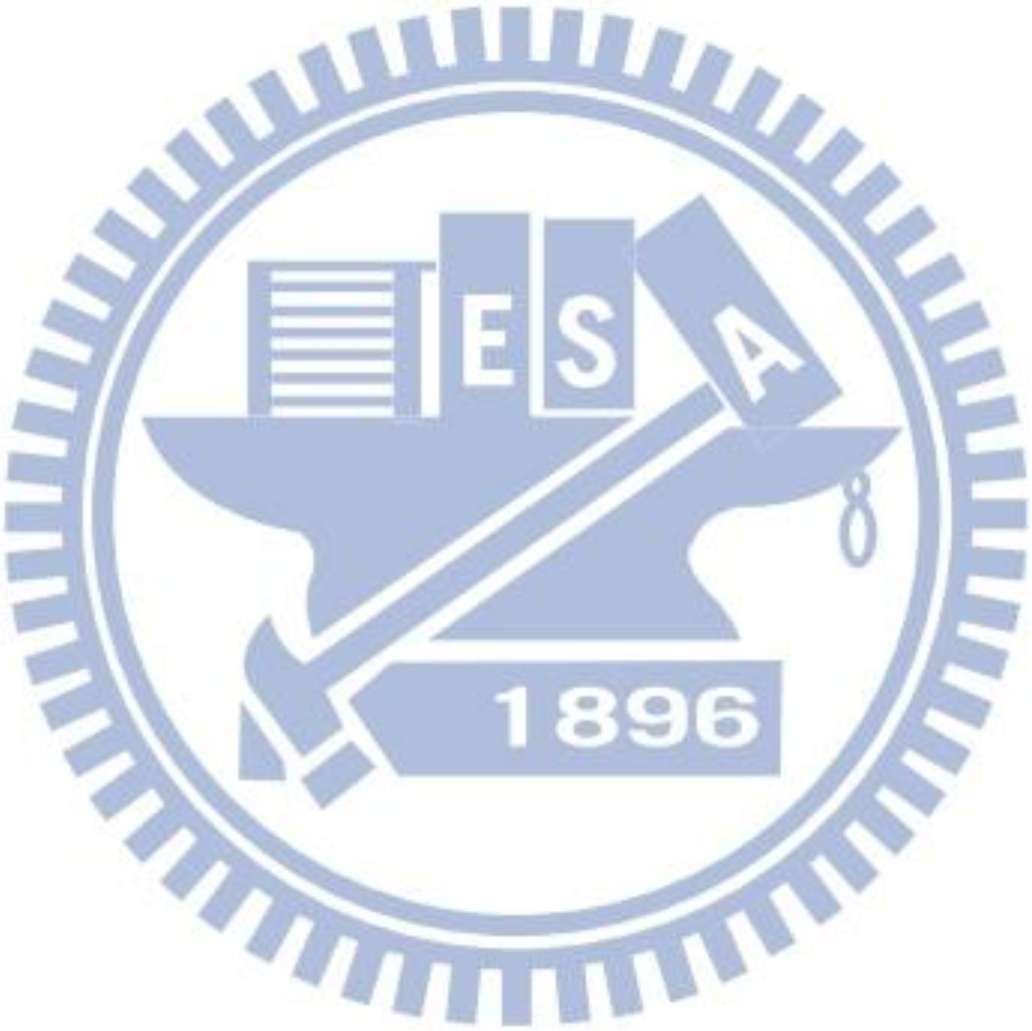


圖 9　範例測試資料（T）執行流程

這樣的設計提供了 GART 強大的彈性，可以依照使用需求，掛載需要的 DLL 模組；也可以依照不同的需要，設計自己的模組插件，讓 GART 支援不同的功能。這樣的設計也讓 GART 的維護變得更容易，不會因為新增的功能（DLL）而影響到原本的系統；每個模組因為是獨立的 DLL，能將程式作適當的切割，不會因為功能日漸強大而變得難以維護。



五、 實驗結果與分析

5.1 GUI 物件尋找

表 1 是 GART 與一些較具代表性的自動腳本工具作比較。GUI 物件尋找是指用什麼方式在 GUI 上找到想要控制的物件。

表 1 物件尋找方式比較

物件尋找方式	Autoit	Sikuli	按鍵精靈	GART
位置	○	○	○	○
顏色			○	
截圖		○	○	
物件類+編號	○			
資訊樹				○

5.2 測試資料生成

一般腳本工具所提供的錄製／播放功能，其所產生的腳本，只包含滑鼠的位置資訊。這樣的腳本很容易因為視窗位置的改變而失效，難以重覆使用。GART 的錄製功能，能自動抓取物件的相關資訊，以方便建立測試資料（自動產生之腳本範列請參閱附錄一）。表 2 列出了常見的腳本工具錄製／播放的功能比較。

表 2 錄製腳本功能比較

錄製腳本	Autoit	Sikuli	按鍵精靈	GART
位置	○	○	○	
物件資訊				○

Autoit 有提供尋找物件的語法，但需要配合額外的工具（Autoit v3 Window Information[13]）。GART 則是在錄製過程就能自動產生腳本，簡化了編輯過程（表 3）。

表 3 腳本製作步驟比較

	Autoit	GART
1.	使用 Autoit v3 Window Information 工具，抓取欲控制的物件資訊。	錄製滑鼠動作。
2.	將 1.中所得的資訊寫入腳本代碼裡。	
3.	對每個想要的滑鼠點擊動作，重覆 1.和 2.。	

使用 Autoit v3 Window Information 工具抓取物件資訊還會遇到一個問題，那就是無法抓取類似圖 10 中的物件。像這類失去滑鼠焦點就會自動消失的彈出式工具視窗，便無法使用工具去抓取資訊。而 GART 是直接錄製滑鼠動作，因此可以避免這個問題。

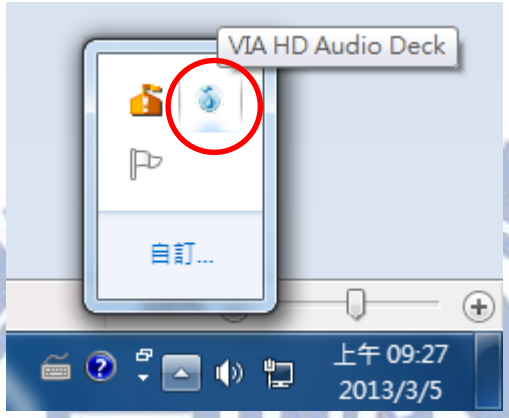


圖 10　彈出式視窗範例

5.3 實現的功能與測試資料

以下會列出目前 GART 已經完成的功能模組（5.3.1）與測試資料（5.3.2）。

5.3.1 實現的功能

表 4 列出目前已經完成的模組插件及其主要的功能。除了_GART.DLL 是必要的檔案之外，其餘的模組插件都是可以動態掛載的。

表 4 功能模組列表

模組插件	功能
_GART.DLL	滑鼠／鍵盤的錄製與播放；呼叫其他模組插件。
_GARTPlugIn_AudioPlay.DLL	播放弦波。
_GARTPlugIn_AudioRecord.DLL	錄音及音源分析。
_GARTPlugIn_Environment.DLL	判斷作業系統及音效晶片版本。
_GARTPlugIn_FindWindow.DLL	以 ITOS 的方法尋找物件。
_GARTPlugIn_If.DLL	AND/OR 判斷條件式。
_GARTPlugIn_Log.DLL	輸出紀錄檔。
_GARTPlugIn_Mixer.DLL	控制系統音效的各種設定。
_GARTPlugIn_Repeat.DLL	重複執行。
_GARTPlugIn_RunCommand.DLL	執行應用程式。
_GARTPlugIn_ScriptFile.DLL	讀取並執行 JSON 腳本。
_GARTPlugIn_Sleep.DLL	睡眠。
_GARTPlugIn_TestData.DLL	測試資料庫資訊。

5.3.2 實現的測試資料

表 5列出了目前已經實現的測試項目。一個測試項目可能有各種組合，因此需要多筆測試資料才能完成測試。以下會詳述測試項目所執行的功能。

表 5 測試資料列表

測試項目	測試內容	測試資料數目
喇叭測試功能	喇叭測試功能是否正常	160
麥克風增益	麥克風增益功能是否正常	32
總計		***192***

喇叭測試功能項目（詳細項目列表請參閱附錄二）：

1. 檢測目前系統（作業系統和音效晶片版本）是否符合測試條件。
2. 設置音量及預設格式，固定測試條件，以免影響測試結果。
3. 滑鼠有 4 個點擊動作，分別點擊 4 個不同的按鈕，之後 VPanel 會播放喇叭測試音樂。
4. 將聲音錄下來，並驗證其正確性。
5. 產生結果紀錄檔：將結果存成 JSON 紀錄檔。

麥克風增益項目（詳細項目列表請參閱附錄三）：

1. 檢測目前系統（作業系統和音效晶片版本）是否符合測試條件。
2. 設置音量及預設格式，固定測試條件，以免影響測試結果。
3. 滑鼠會有 2 個點擊動作及 1 個滑動條拉動，滑動條會設定麥克風增益大小。
4. 播放正弦波並將聲音錄下來，驗證其正確性。
5. 產生結果紀錄檔：將結果存成 JSON 紀錄檔。

表 6 中列出了測試資料的檔案大小,其中可以重覆使用的部份超過90%，不僅能夠節省儲存空間，也節省了編寫測試資料所需的時間。以目前實現的測試資料為例，在製作前 24 筆測試資料時，大約花了 40 個小時（包含測試與調校),而之後的 168 筆測試資料，由於都利用之前寫好的腳本，只花了 6 個小時便建製完成。

表 6　測試資料大小

測試項目	測試資料大小(KB)	平均一個測試大小(KB)	平均一個測試可重複使用部份(KB)	重複使用率[1]
喇叭測試功能	264	16.3	15.3	93.8%
麥克風增益	146	21.8	20.6	94.4%

[1] 重複使用率＝（平均一個測試可重複使用部份）／（平均一個測試大小）。

5.4 實際測試結果

我們實際在 VPanel 上執行已經實現的 192 筆測試資料，以下會列出執行的結果（結果範例請參考附錄四）。

5.4.1 測試時間

表 7 是在繁體中文下的不同樣版上執行的結果；表 8 是在樣版 1 下的不同語言上執行的結果。若以 18:00 下班到隔天 9:00 上班有 15 個小時來計算，一個晚上至少可以跑 6300 筆測試資料，因此在實務上是可行的。

表 7　不同樣版之測試時間（單位為秒）

測試項目	樣版 1	樣版 2	樣版 3	樣版 4	樣版 5	***平均***
喇叭測試功能	1356	1358	1359	1363	1356	***1358.4***
麥克風增益	288	289	287	330	287	***196.2***
總計	1644	1647	1646	1693	1643	***1654.6***

表 8　不同語言之測試時間（單位為秒）

測試項目	繁體中文	英文	簡體中文	德文	法文
喇叭測試功能	1356	1355	1355	1358	1348
麥克風增益	288	251	251	288	250
總計	1644	1606	1606	1646	1598
測試項目	西班牙文	日文	韓文	俄文	***平均***
喇叭測試功能	1350	1350	1350	1359	***1353.4***
麥克風增益	287	288	288	288	***275.4***
總計	1637	1638	1638	1647	***1628.9***

5.4.2 測試錯誤檢測

我們另外設計了會產生錯誤的實驗：在測試時將側置喇叭（7、8 聲道）的連接線拔除，則在 160 筆喇叭測試中，應該會有 16 筆與側置喇叭相關的測試會回報錯誤，並產生發生錯誤時的錄音紀錄檔（WAV 檔）。表 9 是在繁體中文下的不同樣版上執行的結果；表 10 是在樣版 1 的不同語言上執行的結果。

表 9　不同樣版之測試結果

	樣版 1	樣版 2	樣版 3	樣版 4	樣版 5	***平均***
測試時間(秒)	1359	1351	1352	1345	1357	***1352.8***
回報錯誤數量	16	16	16	16	16	***16***
Wav 檔數量	16	16	16	16	16	***16***

表 10　不同語言之測試結果

測試項目	繁體中文	英文	簡體中文	德文	法文
測試時間(秒)	1359	1359	1358	1359	1358
回報錯誤數量	16	16	16	16	16
Wav 檔數量	16	16	16	16	16
測試項目	西班牙文	日文	韓文	俄文	***平均***
測試時間(秒)	1359	1359	1359	1360	***1358.9***
回報錯誤數量	16	16	16	16	***16***
Wav 檔數量	16	16	16	16	***16***

由以上的測試結果，證明了 GART 的確可以使用相同的測試資料，在不同的樣版和不同的語言上執行測試，並且都能夠正確執行，找出錯誤，得到預期的結果。

5.5 限制

1. GART 是針對應用程式上的元件而開發的，無法分辨出同一元件上的不同功能。如瀏覽器，一個網頁對應用程式來說就是一個物件（window），GART 無法辨認同一網頁上的不同連結。
2. GART 使用資訊樹（詳見 4.5）作為尋找物件的主要依據，如果在程式改版時，刪除或插入了某些視窗或物件 ID，可能造成資訊樹改變，而讓測試資料失效。因此在軟體改版時，要注意不能更動原本物件的資訊樹，否則測試資料將會產生錯誤。
3. 如果 GUI 上的物件 ID 是隨機生成的，便無法使用 ITOS 的資訊找到正確的物件。如圖 3 的聲音視窗，有可能因為新增或移除音訊裝置而在重新關機後造成物件 ID 改變。
4. ITOS 雖然保留了更多資訊，讓我們更容易找到目標物件，但也會遇到二個物件完全無法分辨的情況。此時需要利用額外的資訊，GART 使用位置來辨別不同的物件，也失去了在不同介面使用同樣測試資料的好處。
5. GART 雖然能讓同筆測試資料跑在不同介面上，但如果介面改變時，將橫（直）向的滑動條改為直（橫）向，將造成滑鼠拖曳的方向錯誤。

六、 結論與未來展望

6.1 結論

GART 所提出的資訊樹物件探索（ITOS），能夠準確地找到 GUI 上的物件，不因物件的大小、位置、顏色、文字……等屬性改變而受到影響。相較於以往的 GUI 測試工具，因為利用物件本身的屬性作為判斷，因此常常因為介面設計上的更動，而需要去更改測試資料，耗費多餘的人力。如今有許多應用程式，為了迎合使用者的需求，開發了各種不同的佈景主題與多國語言的支援；如果為了測試這些不同樣版和語言的 GUI，就需要重新編寫測試資料，將讓測試資料的維護工作變得相當龐大，甚至無法維護；GART 使用 ITOS 的方法，能夠在不同的介面上使用相同的測試資料，降低維護工作所需的人力，讓測試人員更能專注在功能測試上。

GART 全部以動態連結檔的方式建構，並將功能切割成不同的檔案，讓 GART 的維護變得更簡單。一個軟體的維護，比起開發更為重要，尤其在業界，一個專案可能需要許多人共同維護，也可能因為各種原因，更換維護的負責人員。GART 的設計，將各個功能切割，不僅在發生問題時，能縮小尋找範圍，也不會因為一個功能模組的問題（Bug），而影響到程式的其他部份。更重要的，因為 GART 將程式模組化，能讓軟體工程師更容易了解整個系統，也更容易執行分工，讓專案的轉移更簡單，不會因為人員的更換，造成系統無法維護。

GART 將功能模組作為插件使用，除了新增功能變得簡單，也能利用這樣的特性作不同的應用。例如作內部測試時，可能會使用較多的插件，執行詳細的測試；而拿掉部份插件，便可以作為外部測試工具，提供給客戶使用。具有彈性的設計，讓 GART 能因應不同的情境，不用更改程式碼，便能有不同的應用。

6.2 未來發展方向

GART所提出的資訊樹物件探索(ITOS),利用了GUI物件階層的概念，除了物件本身,還儲存了整個階層的資訊,能夠準確的找出GUI上的物件。ITOS 可以應用在任何擁有物件階層的 GUI 上，如 Android、網頁……等，只要能分析出 GUI 的物件關係，便能運用 ITOS 來提高準確度。

由於 GART 的可擴充性，讓它能作到一般自動化腳本工具所能作到的事；例如：自動點擊、批量操作、軟體測試……等。GART 的錄製／播放功能，也能作為錯誤回報工具；有時客戶回報的問題，很難用文字或圖片描述，若使用錄製／播放的功能，便能清楚地了解問題產生的步驟。

GART 使用 JSON 作為腳本語言，編寫測試資料仍需對程式編寫有一定的了解，但有些測試員並不熟悉程式語言。為了讓編寫測試資料更簡單，一個好的腳本編輯軟體是很有用的。編寫 JSON 的編輯軟體比起其他腳本語言簡單，不用另外寫語言分析器（Parsing Tool），能對 JSON 直接存取，不會有額外的工作。

GART 作為回歸測試系統，必定會產生大量的測試結果，需要一個系統來作管理。目前只有實現結果分級的部份，未來還需要更多的功能；或是將 JSON 的結果檔案轉到資料庫中,用現有的資料庫軟體(如 MySQL[21]）來管理。

參考文獻

[1] Zeller, Andreas, Why programs fail : a guide to systematic debugging, Elsevier, Amsterdam ; Boston, 2005.

[2] Intel, I., & Hub, O. C. 6 (ICH6) High Definition Audio. AC, 97, 1-207.

[3] Devices A., "Audio Codec'97; Component Specification; Revision 1.03; Sep. 15, 1996.", 1997.

[4] Microsoft, “MSDN”. URL:http://msdn.microsoft.com

[5] Crockford D., “JSON: The fat-free alternative to XML”, 2006. URL:http://www.json.org/

[6] Nemex, “Mouse Recorder Pro”. URL:http://byshynet.com/software.php?id=3

[7] Vista Software, “TinyTask”. URL:http://www.vtaskstudio.com/support.php

[8] Polital Enterprises, “Point-N-Click”. URL: http://polital.com/pnc/

[9] S. Senthil Kumar, “WinMacro”. URL: http://winmacro.codeplex.com/

[10] 兄弟軟體, ”按鍵精靈”, 2001-2013. URL:http://tw.vrbrothers.com/qmacro/index.htm

[11] Yeh T., Chang T. H., Miller R. C., “Sikuli: using GUI screenshots for search and automation.” Proceedings of the 22nd annual ACM symposium on User interface software and technology, pp. 183-192, ACM, October 2009.

[12] MouseMacroRecorder, “Mouse Macro Recorder”, 2013. URL:http://www.advanced-mouse-auto-clicker.com/

[13] Bennett J., “Autoit”, 2011-2012. URL:http://www.autoitscript.com/site/autoit/

[14] Microsoft, “Visual Studio”. URL:http://msdn.microsoft.com/zh-TW/vstudio/

[15] Beck K., Gamma E., “JUnit Test Infected: Programmers Love Writing Tests”, Java Report, 3(7), July 1998. URL:http://JUnit.sourceforge.net/doc/testinfected/testing.htm

[16] Beck K., Gamma E., “JUnit A Cookís Tour”, Java Report, 4(5), May 1999. URL: http://JUnit.sourceforge.net/doc/cookstour/cookstour.htm

[17]Kumar A., Clair J. S., “Cunit-a unit testing framework for c”, Acesso em, 25, 2005. URL: http://cunit.sourceforge.net/
[18]Eric S., Michael F., Jerome L., “CppUnit”. URL:http://sourceforge.net/projects/cppunit/
[19]JenSense, “JSON Format”. URL:http://jsonformat.com/#jsondataurllabel
[20]Jos de Jong , “JSON Editor Online”, 2011-2013. URL:http://jsoneditoronline.org/
[21]MySQL, A. B., "Mysql 5.1 reference manual.", Sun Microsystems, 2007.
[22]Memon A., Nagarajan A., Xie Q., “Automating regression testing for evolving GUI software”, Journal of Software Maintenance and Evolution: Research and Practice, 17, pp. 27-64, 2005.
[23]Gomez L., Neamtiu I., Azim T., Millstein T., “RERAN: Timing- and Touch-Sensitive Record and Replay for Android”, International Conference on Software Engineering (ICSE 2013), San Francisco, CA, May 2013.
[24]Ecava, “Jsonvalue”, 2010-2011. URL:http://code.google.com/p/jsonvalue/
[25]Mills A. J. S., "Log4J". URL: http://logging.apache.org/log4j/1.2/

附錄一

點擊一個按鈕所自動產生的腳本：

```
[ { "If": {
        "And": [ {
            "FindWindow": {   //物件資訊樹
                        "message": 512, // 滑鼠資訊
                        "mousePos": { "X": 266, "Y": 515 },
                        "mouseClientPos": { "X": 53, "Y": 16 },
                        "mouseWheel": 0,
                        "dwDelayTime": 500,   // 延遲時間
                        "RootWindow": {   // 根視窗
                            "CtrlID": 0,
                            "Text": "VIA HD Audio Deck",
                            "Class": "#32770",
                            "Style": 2496266316,
                            "ID": 0,
                            "Width": 799,
                            "Height": 599 },
                        "Window": {   // 物件資訊
                            "CtrlID": 1011,
                            "Text": "",
                            "Class": "AfxWnd110su",
                            "Style": 1342177280,
                            "ID": 1011,
                            "Width": 79,
                            "Height": 42 },
                        "Parent": [     // 相關的母視窗
                            {
                                "CtrlID": 0,
                                "Text": "",
                                "Class": "#32770",
                                "Style": 1342701644,
                                "ID": 0,
                                "Width": 570,
                                "Height": 370
                            },
                            {
```

```
        "CtrlID": 0,
        "Text": "",
        "Class": "#32770",
        "Style": 1342701644,
        "ID": 0,
        "Width": 570,
        "Height": 400
    },
    {
        "CtrlID": 0,
        "Text": "",
        "Class": "#32770",
        "Style": 1342701644,
        "ID": 0,
        "Width": 640,
        "Height": 554
    },
    {
        "CtrlID": 0,
        "Text": "",
        "Class": "#32770",
        "Style": 1342701644,
        "ID": 0,
        "Width": 670,
        "Height": 540
    },
    {
        "CtrlID": 0,
        "Text": "",
        "Class": "#32770",
        "Style": 1342701644,
        "ID": 0,
        "Width": 658,
        "Height": 554
    },
    {
        "CtrlID": 0,
        "Text": "",
```

```
                    "Class": "#32770",
                    "Style": 1342701644,
                    "ID": 0,
                    "Width": 640,
                    "Height": 554
                } ] } } ],
"Check": 1,
"Action": [  // 滑鼠動作
    {
        "MouseStep": {
            "message": 513,  // Mouse Down
            "mousePos": { "X": 266,  "Y": 515 },
            "mouseClientPos": { "X": 53, "Y": 16 },
            "mouseWheel": 0,
            "dwDelayTime": 500,
            "RootWindow": {
                "CtrlID": 0,
                "Text": "VIA HD Audio Deck",
                "Class": "#32770",
                "Style": 2496266316,
                "ID": 0,
                "Width": 799,
                "Height": 599
            },
            "Window": {
                "CtrlID": 1011,
                "Text": "",
                "Class": "AfxWnd110su",
                "Style": 1342177280,
                "ID": 1011,
                "Width": 79,
                "Height": 42
            } } },
    {
        "MouseStep": {
            "message": 514,  // Mouse Up
            "mousePos": { "X": 266, "Y": 515 },
            "mouseClientPos":  "X": 53, "Y": 16 },
```

```
"mouseWheel": 0,
"dwDelayTime": 125,
"RootWindow": {
        "CtrlID": 0,
        "Text": "VIA HD Audio Deck",
        "Class": "#32770",
        "Style": 2496266316,
        "ID": 0,
        "Width": 799,
        "Height": 599
},
"Window": {
        "CtrlID": 1011,
        "Text": "",
        "Class": "AfxWnd110su",
        "Style": 1342177280,
        "ID": 1011,
        "Width": 79,
        "Height": 42
}   }   }   ]   }   }   ]
```

附錄二

喇叭測試功能，測試資料列表：

編號	聲道	取樣頻率(Hz)	位元深度(bit)	喇叭
Test00000	2	44100	16	前置
Test00000	2	44100	16	前置左
Test00001	2	44100	16	前置右
Test00002	4	44100	16	前置左
Test00003	4	44100	16	前置右
Test00004	4	44100	16	後置左
Test00005	4	44100	16	後置右
Test00006	6	44100	16	前置左
Test00007	6	44100	16	前置右
Test00008	6	44100	16	後置左
Test00009	6	44100	16	後置右
Test00010	6	44100	16	中置
Test00011	6	44100	16	重低音
Test00012	8	44100	16	前置左
Test00013	8	44100	16	前置右
Test00014	8	44100	16	後置左
Test00015	8	44100	16	後置右
Test00016	8	44100	16	中置
Test00017	8	44100	16	重低音
Test00018	8	44100	16	側置左
Test00019	8	44100	16	側置右
Test00020	2	48000	16	前置左
Test00021	2	48000	16	前置右
Test00022	4	48000	16	前置左
Test00023	4	48000	16	前置右
Test00024	4	48000	16	後置左
Test00025	4	48000	16	後置右
Test00026	6	48000	16	前置左
Test00027	6	48000	16	前置右
Test00028	6	48000	16	後置左
Test00029	6	48000	16	後置右
Test00030	6	48000	16	中置
Test00031	6	48000	16	重低音

Test00032	8	48000	16	前置左
Test00033	8	48000	16	前置右
Test00034	8	48000	16	後置左
Test00035	8	48000	16	後置右
Test00036	8	48000	16	中置
Test00037	8	48000	16	重低音
Test00038	8	48000	16	側置左
Test00039	8	48000	16	側置右
Test00040	2	96000	16	前置左
Test00041	2	96000	16	前置右
Test00042	4	96000	16	前置左
Test00043	4	96000	16	前置右
Test00044	4	96000	16	後置左
Test00045	4	96000	16	後置右
Test00046	6	96000	16	前置左
Test00047	6	96000	16	前置右
Test00048	6	96000	16	後置左
Test00049	6	96000	16	後置右
Test00050	6	96000	16	中置
Test00051	6	96000	16	重低音
Test00052	8	96000	16	前置左
Test00053	8	96000	16	前置右
Test00054	8	96000	16	後置左
Test00055	8	96000	16	後置右
Test00056	8	96000	16	中置
Test00057	8	96000	16	重低音
Test00058	8	96000	16	側置左
Test00059	8	96000	16	側置右
Test00060	2	192000	16	前置左
Test00061	2	192000	16	前置右
Test00062	4	192000	16	前置左
Test00063	4	192000	16	前置右
Test00064	4	192000	16	後置左
Test00065	4	192000	16	後置右
Test00066	6	192000	16	前置左
Test00067	6	192000	16	前置右
Test00068	6	192000	16	後置左

Test00069	6	192000	16	後置右
Test00070	6	192000	16	中置
Test00071	6	192000	16	重低音
Test00072	8	192000	16	前置左
Test00073	8	192000	16	前置右
Test00074	8	192000	16	後置左
Test00075	8	192000	16	後置右
Test00076	8	192000	16	中置
Test00077	8	192000	16	重低音
Test00078	8	192000	16	側置左
Test00079	8	192000	16	側置右
Test00080	2	44100	24	前置左
Test00081	2	44100	24	前置右
Test00082	4	44100	24	前置左
Test00083	4	44100	24	前置右
Test00084	4	44100	24	後置左
Test00085	4	44100	24	後置右
Test00086	6	44100	24	前置左
Test00087	6	44100	24	前置右
Test00088	6	44100	24	後置左
Test00089	6	44100	24	後置右
Test00090	6	44100	24	中置
Test00091	6	44100	24	重低音
Test00092	8	44100	24	前置左
Test00093	8	44100	24	前置右
Test00094	8	44100	24	後置左
Test00095	8	44100	24	後置右
Test00096	8	44100	24	中置
Test00097	8	44100	24	重低音
Test00098	8	44100	24	側置左
Test00099	8	44100	24	側置右
Test00100	2	48000	24	前置左
Test00101	2	48000	24	前置右
Test00102	4	48000	24	前置左
Test00103	4	48000	24	前置右
Test00104	4	48000	24	後置左
Test00105	4	48000	24	後置右

Test00106	6	48000	24	前置左
Test00107	6	48000	24	前置右
Test00108	6	48000	24	後置左
Test00109	6	48000	24	後置右
Test00110	6	48000	24	中置
Test00111	6	48000	24	重低音
Test00112	8	48000	24	前置左
Test00113	8	48000	24	前置右
Test00114	8	48000	24	後置左
Test00115	8	48000	24	後置右
Test00116	8	48000	24	中置
Test00117	8	48000	24	重低音
Test00118	8	48000	24	側置左
Test00119	8	48000	24	側置右
Test00120	2	96000	24	前置左
Test00121	2	96000	24	前置右
Test00122	4	96000	24	前置左
Test00123	4	96000	24	前置右
Test00124	4	96000	24	後置左
Test00125	4	96000	24	後置右
Test00126	6	96000	24	前置左
Test00127	6	96000	24	前置右
Test00128	6	96000	24	後置左
Test00129	6	96000	24	後置右
Test00130	6	96000	24	中置
Test00131	6	96000	24	重低音
Test00132	8	96000	24	前置左
Test00133	8	96000	24	前置右
Test00134	8	96000	24	後置左
Test00135	8	96000	24	後置右
Test00136	8	96000	24	中置
Test00137	8	96000	24	重低音
Test00138	8	96000	24	側置左
Test00139	8	96000	24	側置右
Test00140	2	192000	24	前置左
Test00141	2	192000	24	前置右
Test00142	4	192000	24	前置左

Test00143	4	192000	24	前置右
Test00144	4	192000	24	後置左
Test00145	4	192000	24	後置右
Test00146	6	192000	24	前置左
Test00147	6	192000	24	前置右
Test00148	6	192000	24	後置左
Test00149	6	192000	24	後置右
Test00150	6	192000	24	中置
Test00151	6	192000	24	重低音
Test00152	8	192000	24	前置左
Test00153	8	192000	24	前置右
Test00154	8	192000	24	後置左
Test00155	8	192000	24	後置右
Test00156	8	192000	24	中置
Test00157	8	192000	24	重低音
Test00158	8	192000	24	側置左
Test00159	8	192000	24	側置右

附錄三

麥克風增益測試，測試資料列表：

編號	取樣頻率(Hz)	位元深度(bit)	增益大小(db)
Test00160	44100	16	0
Test00161	44100	16	10
Test00162	44100	16	20
Test00163	44100	16	30
Test00164	48000	16	0
Test00165	48000	16	10
Test00166	48000	16	20
Test00167	48000	16	30
Test00168	96000	16	0
Test00169	96000	16	10
Test00170	96000	16	20
Test00171	96000	16	30
Test00172	192000	16	0
Test00173	192000	16	10
Test00174	192000	16	20
Test00175	192000	16	30
Test00176	44100	24	0
Test00177	44100	24	10
Test00178	44100	24	20
Test00179	44100	24	30
Test00180	48000	24	0
Test00181	48000	24	10
Test00182	48000	24	20
Test00183	48000	24	30
Test00184	96000	24	0
Test00185	96000	24	10
Test00186	96000	24	20
Test00187	96000	24	30
Test00188	192000	24	0
Test00189	192000	24	10
Test00190	192000	24	20
Test00191	192000	24	30

附錄四

麥克風增益測試所產生的紀錄檔：

```
{ "LogRead": [  {
        "Level": 4,    // 紀錄檔的分級，1 代表 Fail，4 代表 Pass
        "Module": "_GARTPlugIn_AudioRecord.dll",    // 由哪個功能檔產生的訊息
        "Message": [  {
            "Result": 1,    // 最後的結果，0 代表 Fail，1 代表 Pass
            "LeftResult": {  // 左聲道的結果
                "Result": 1,
                "Signal": {  // 主頻
                    "Result": 1,
                    "fMin": 900,  // 標準值下限
                    "fMax": 1100,  // 標準值上限
                    "fValue": 1007.8125  }, // 測試的結果值
                "SNR": {  // SNR 的結果
                    "Result": 1,
                    "fMin": 10,
                    "fMax": 20,
                    "fValue": 15.912925  },
                "THD": {  // THD 的結果
                    "Result": 1,
                    "fMin": -14,
                    "fMax": -4,
                    "fValue": -9.792718  },
                "PeakMax": {  // 峰值的結果
                    "Result": 1,
                    "fMin": 25600,
                    "fMax": 32000,
                    "fValue": 25954  }
            },
```

```
"RightResult": {   // 右聲道的結果，說明同上
    "Result": 1,
    "Signal": {
        "Result": 1,
        "fMin": 900,
        "fMax": 1100,
        "fValue": 1007.8125    },
    "SNR": {
        "Result": 1,
        "fMin": 10,
        "fMax": 20,
        "fValue": 15.954743    },
    "THD": {
        "Result": 1,
        "fMin": -14,
        "fMax": -4,
        "fValue": -9.817779    },
    "PeakMax": {
        "Result": 1,
        "fMin": 25600,
        "fMax": 32000,
        "fValue": 26183    }    }    }    ],
  "Time": [ 2013, 5, 14, 12, 40, 29, 851 ]    },   // 測試結束的時間
{    "End": 1    }    ]    }
```