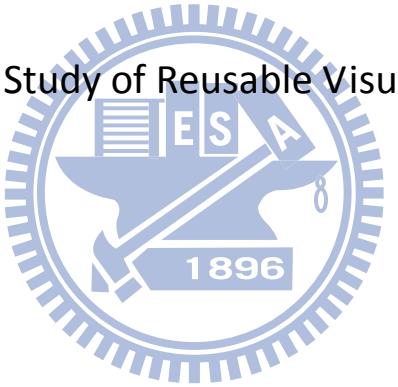
# 國 立 交 通 大 學

## 資訊學院　資訊學程

## 碩 士 論 文

可重用的視覺化劇情研究

On the Study of Reusable Visual Scenario



研 究 生：王毓維

指導教授：陳登吉　教授

中 華 民 國 一 百 年 一 月

# 可重用的視覺化劇情研究

# On the Study of Reusable Visual Scenario


研 究 生：王毓維　　Student：Yu-Wei Wang

指導教授：陳登吉　　Advisor：Deng-Jyi Chen


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

January 2011

Hsinchu, Taiwan, Republic of China

中 華 民 國 一 百 年 一 月

# 可重用的視覺化劇情研究

學生：王毓維　　指導教授：陳登吉　博士

國立交通大學　資訊學院　資訊學程碩士班

## 摘要

隨著資訊科技的進步，程式語言不斷的在演進與發展，如何在現有的程式語言上建立更快速便捷的開發環境，或是發展更方便更直覺的程式語言，就成為程式語言研究上重要的議題。有別於傳統以文字為基礎的程式語言，視覺化程式語言利用圖像、聲音或動畫，透過圖像化的開發界面讓程式開發更為直覺與容易，成為近年來程式語言發展的方向之一。本研究是專注在視覺化程式語言編輯完成的視覺化劇情，利用重用的概念，讓開發視覺化程式更為快速簡便。

在程式語言的發展過程中，重用一直是個很重要的觀念。從組合語言中實現程式碼重用，結構化程式語言中利用子函式實現程式碼重用，或是物件導向程式語言中的物件重用，到程式設計的概念上，利用設計模式達到設計概念的重用，這些重用不但加快程式開發速度，同時減少重複撰寫程式碼或是程式設計上可能犯錯的機會及時間，增加程式可讀性。相同的，在視覺化程式語言中提供了物件重用，讓使用者在編輯視覺化劇情時可以快速重用物件，但辛苦編輯完成的劇情卻無法重用，若是有大量類似劇情的需求時，只能夠一再重複編輯劇情的動作，不但浪費時間且容易出錯。

因此本研究提出視覺化劇情重用，藉由套用機制將已完成的視覺化劇情重複使用，並透過實例來說明視覺化劇情重用在視覺化程式語言中對於開發程式的助益，確認此研究可適用於視覺化程式語言的未來發展上。

關鍵字：視覺化程式語言、視覺化劇情、重用

# On the Study of Reusable Visual Scenario

Student：Yu-Wei Wang　　Advisor：Dr. Deng-Jyi Chen

Degree Program of Computer Science
National Chiao Tung University

## ABSTRACT

By development of information technology, the programming language is constant evolution and development. How to create quickly and easily environment at existed programming language, or develop more convenient and intuitive programming language has become an important research issue. Unlike traditional textual programming language, visual programming language that makes develop more easily by using image, sound, animation and graphical interface, has become one of direction of development recent years. We focus on the study of visual scenario that made by visual programming language, using the concept of reusing, make the visual programming more easily and quickly.

Reuse is an important concept in the process of development of programming language. From reuse code at assembly, reuse sub-routine at structured programming language, reuse object at object-oriented programming language, or reuse design pattern when design architecture, these reuses not only speed up the program development, but also reduce duplicated code, opportunities of making mistake ,save time, and increase the readability of the code. Similarly, the visual programming language also provides object reuse that allowing users to reuse objects quickly when develop visual scenario. But the visual scenario can't be reused, you have to re-create scenario if you need lots of similar scenario, it waste time and easy to make mistake.

This research proposed a reusable visual scenario, reuse visual scenario by applying mechanism, and through example to illustrate the reusable visual scenario is helpful for the development program, confirm that this research is suitable for future development of visual programming language.

Keywords: Visual Programming Language, Visual Scenario, Reuse

# 誌謝

衷心感謝陳登吉教授耐心的指導，在研究的過程中，時時給予建議與方向，才能順利完成本論文。

同時也要特別感謝孔崇旭教授給予的指導與建議，讓我瞭解論文研究的方法與注意事項。

而在研究的過程中，則要感謝學長信江、仲智、學姐詩雯的指導幫助，以及學弟興郎的互相鼓勵與一同進行研究，透過彼此的分享與討論，使我可以更清楚相關的研究方向。

最後，要感謝我的家人與同事在求學的這一段期間內給予的支持與鼓勵，特別要感謝我的母親，兄長，大嫂，兩位可愛的姪子以及在我身邊陪伴我的晴慧，有您們的支持才能成就我的理想，感謝您們！！

# 目錄

# 表目錄



# 圖目錄

# 一、緒論

## 1.1 研究動機及背景

隨著資訊科技的進步，程式語言不斷的在演進與發展，如何在現有的程式語言上建立更快速便捷的開發環境，或是發展更方便更直覺的程式語言，就成為程式語言研究上重要的議題。有別於傳統以文字為基礎的程式語言，視覺化程式語言利用圖像、聲音或動畫，透過圖像化的開發界面讓程式開發更為直覺與容易，成為近年來程式語言發展的方向之一。本研究是專注在視覺化程式語言編輯完成的視覺化劇情，利用重用的概念，讓開發視覺化程式更為快速簡便。

傳統程式語言是透過文字化編輯環境，利用程式語言提供的指令來撰寫，來實踐一段有意義的程式碼，在這個時期就已經開始有重用的概念，但僅止於程式碼的重用。程式語言後來演進到結構化的程式語言，仍然是透過文字化編輯介面，但結構化的程式語言除了利用語言本身提供的指令以外，另外也允許將常用的程式碼作成子函式供重複使用，此時重用的單位從原本的程式碼重用，抽象化成為結構重用。大型系統的出現讓程式語言有了新的趨勢，這種複雜又容易出錯的軟體，出錯的代價可能是昂貴的，因此有了新的創新[1]，這就是物件導向程式語言，而開發環境也從原本的文字化編輯介面，開始出現整合式開發介面（Integrated Development Environment），讓物件的開發與重用更為便捷，也加速了程式的開發速度，重用單位也從結構重用，抽象化成為物件重用。

在物件導向程式語言發展出來之後，程式語言的演進就沒有較大的創新，發展方向改為加速程式設計師的生產力，改善開發環境，或是擴充程式語言的適用環境，直到近年來發展出視覺化程式語言（Visual Programming Language，簡稱為 VPL），利用圖形與動畫當作表現工具，藉由對話表單，視窗與圖示等比喻，和使用者利用滑鼠與鍵盤來互動[2]，同時藉由視覺化程式語言所提供的圖像化編輯環境，讓使用者可以利用簡單的步驟來完成應用程式。在視覺化程式語言當中，每個圖示或動畫就代表一個物件，而紀錄這這些物件的位置，播放次數或是移動路徑等屬性，我們就稱為是視覺化的劇情。在這個階段，重用單位一樣為物件重用，但重用的是視覺化 程式語言的物件，而非傳統程式語言的物件。程式語言的發展歷程與重用的單位，大致上就可以用圖 1 來代表。


圖 1　程式語言發展歷程圖

雖然視覺化程式語言有著較友善的介面，但是若是需要大量類似或相同的視覺化程式，會有以下的缺點產生：

1. 編輯成果無法重用

   編輯成果是由重用物件與視覺化劇情組成，無法重新使用。若是需要大量類似或相同的視覺化劇情，則必須重複進行編輯動作。

2. 成果品質有可能不穩定

   若需要類似或相同的劇情，必須一再重複編輯，有可能因為程式設計師的一時粗心或手誤，造成編輯成果的錯誤。

3. 難以用來編輯大型程式

   因為視覺化程式語言利用圖形、動畫等視覺化物件來表達，因此使用上比起文字的程式語言更為耗費空間，建構大型程式就需要更多的空間或頁面來表達，對於閱讀或維護上都非常困難，難以用來編輯大型程式。

## 1.2　研究目的

程式語言的發展從傳統程式語言，結構化程式語言，物件導向程式語言到視覺化程式語言，重用單位也隨著不斷的演進。從指令重用、結構重用、發展到物件重用與視覺化物件重用，目的就是讓程式設計師能夠在更短的時間內開發完成。

物件導向對於程式設計人員開發程式，除了效率上的助益，另外對於軟體工程上，在軟體品質項目中的其中三個項目也有所幫助[3]，並沒有因為縮短開發時程而犧牲軟體品質。這三個項目分別是：

1. 可重用性（Reusability）

   物件導向的繼承性（Inheritance）與多型性（Polymorphism），讓軟體的重用程度提高。

2. 可維護性（Maintainability）

   物件導向程式語言較容易新增或移除部份程式碼，對於軟體的維護與加強也較容易。

3. 可靠性（Reliability）

   物件導向的封裝性（Encapsulation），讓資料的修改只能透過特定方式，較為安全可靠。

重用的概念隨著程式語言發展而不斷抽象化，隨著物件導向程式語言的發展，重用概念也抽象化到了物件重用及共用元件的階段，但程式語言後續發展走向加速程式設計師的生產力，改善開發環境，或是擴充程式語言的適用環境，發展方向是在現有的程式語言中做改善，而重用概念並無大量的更新，後來發展出視覺化程式語言，擁有物件導向程式語言的特性，品質方面也有相同的優點。但是在重用的概念上仍然是物件重用，並沒有更抽象化的概念產生。

因此本研究的目的就是提出一個更抽象的重用概念，在視覺化程式語言所產生的視覺化劇情上，藉由套用機制來達成更抽象的重用，讓視覺化程式開發不再是使用可重用的視覺化物件，而是使用可重用的視覺化劇情來完成程式開發。另外，利用可重用的視覺化劇情開發同時，軟體品質可以穩定並沒有隨之降低。

## 1.3 研究方法及步驟

程式語言演進的目的是為了讓開發程式更為快速便捷，為了達成這個目的，方法不外乎：

1. 加強指令的能力：

   利用一個指令就可以執行多項動作，較少的程式碼能完成相同的功能，因

此開發時間就會縮短。

2. 大量使用重用元件組裝：

   不需要每行程式碼都自己撰寫，利用已經完成的元件來組合出相同的功能，也有助於開發程式的速度。

3. 提供更方便的使用介面：

   透過方便的介面，不需要記憶指令或查詢使用方法，就可編輯程式，減少查詢或使用錯誤浪費的時間。

因此，首先我們探討了程式語言的發展歷程與重用概念的演進，並探討了物件導向程式語言與視覺化程式語言的特性。接下來我們將依照下列步驟進行研究：

1. 探討程式語言的演進及發展趨勢，並說明視覺化程式語言的特性，藉由這個步驟說明視覺化程式語言的發展目的。
2. 比較程式語言與視覺化程式語言的差異性，包括在撰寫程式會使用的函式或元件，以及撰寫程式的方法和步驟，藉由這個步驟來說明視覺化程式語言的優缺點。
3. 針對重用機制來進行探討，藉由這個步驟來說明重用機制的精神和應用結果。
4. 介紹可重用的視覺化劇情。
5. 利用實做上的範例，來說明可重用的視覺化劇情在設計上的精神。
6. 進行優缺點的分析及其適用性。

## 1.4 章節概要

本論文共分為六個章節，各章節之內容依序摘要如下：

第一章，說明程式語言的發展歷程與重用概念的演進，並提出本論文的研究動機、目的與研究方法。

第二章，探討程式語言的演進及發展趨勢。

第三章，比較程式語言與視覺化程式語言的差異性。

第四章，針對重用機制來進行探討。

第五章，介紹可重用的視覺化劇情及其範例。

第六章，提出本論文的結論與未來可以繼續發展的方向。

# 二、探討程式語言的演進及發展趨勢

程式語言從 1940 年代開始發展，至今已經發展了約 70 年，在這段期間程式語言的架構及特性有了很大的變化，程式設計師開發程式的方法及概念也不斷的變化。所以本章節會探討程式語言的演進及發展趨勢，以及在這樣的發展趨勢下，對於程式設計師有著什麼樣的影響。

## 2.1 程式語言的演進與發展趨勢

如何發展出一套程式語言，對於開發程式有著更快速及簡單的方法，一直是軟體領域的重要議題之一。在探討是否能夠提出更新的觀念之前，我們必須先從以往的演進及發展趨勢來看起，藉由歷史的發展軌跡，來找尋在程式語言的發展中，什麼樣的改變對於程式設計師開發程式是有幫助的，或是什麼樣的觀念更有助於軟體品質的提昇，由歸納的結果來找出未來發展的方向。

### 2.1.1 程式語言的演進

程式語言是一組用來撰寫程式的符號與規則，藉由組合字元與運用規則，在電腦上執行時可以產生特定功用[4]，在演進上可以分成四代：

1. 第一代：機械語言（Machine Language）

   由 0 與 1 構成，電腦能夠直接執行不需要翻譯的語言。

2. 第二代：組合語言（Assembly Language）

   一種符號化的程式語言，利用簡單、有意義且容易記憶（mnemonic）的英文縮寫字取代機械語言、可讀性較高，需要經過組譯程式（Assembler）轉換成機械語言才能執行。

3. 第三代：高階語言（High Level Language）

   利用人類易用的文字符號來表達的程式語言，撰寫完成的程式語言還需要經過編譯程式轉換成機械語言才能執行，從 FORTRAN 開始，到 C/C++，Java 等等都是屬於高階語言。

4. 第四代語言（Fourth Generation Language）

要使用機械語言、組合語言或是高階語言並非簡單的事，通常都需要經過程式撰寫的專業訓練。為了讓使用者更容易使用，因此發明第四代語言，通常以句子或圖像選擇所要的功能，目的是讓使用者與電腦溝通更為容易，例如像 SQL（Structured Query Language）或是 SAS（Statistical Analysis Software）。

### 2.1.2 程式開發演進歷程

透過程式語言開發應用程式，也會隨著程式語言特性的不同，而會有著不同的設計概念。在編寫組合語言程式時，使用指令來針對暫存器（register）進行操作，或是利用流程控制指令來控制程式流程，此時是使用組合語言的內建指令或函式來完成程式。

子函式（subroutine）是非常強大的程式撰寫工具[5]，結構化程式語言發展出來後，程式語言可由程式設計師自行建立子函式，因此程式開發除了使用內建函式外，還會建立子函式減少重複的程式碼。物件導向程式語言的出現，讓物件更容易重用[3]，因此程式設計師也可以自行建立類別（Class）來達到物件重用的目的。為了減少軟體開發的成本與開銷，許多軟體專案已經不再是自行開發所有的元件，而是透過整合性的軟體元件來開發[6]，因此開發程式不僅只有內建指令、子函式或是類別，更包含了使用共用元件。

### 2.1.3 程式語言發展趨勢

程式語言的演進從機械語言到第四代程式語言，程式開發從使用內建指令到共用元件，我們可以從前面的討論歸納出以下幾點特性：

1. 基本指令的複雜程度提高

   從組合語言發展到第四代語言，單一指令從簡單的暫存器操作或流程控制，提升到數學運算或檔案存取，第四代語言一個指令就完成了資料的排序的動作，單一指令的複雜度有明顯的提昇。

2. 程式語言的使用範圍受限制

   為了滿足特定目標而設計的程式語言稱為特殊用途語言（special-purpose language）[4]，第四代語言有特殊用途語言的特性，設計上是為了特定範圍的使用，提供資料庫操作，網路連結，文件撰寫的功能。

3. 硬體需求提昇

   指令的複雜度提高，因此為了維持執行效能，對於硬體的需求也相對提昇。

4. 程式語言提供的功能增加

組合語言只提供了暫存器操作或流程控制，所有程式的邏輯都需要程式設計師自行開發實做。高階語言及第四代語言歸納出經常使用的功能及物件，在基本函式、物件及共用元件中提供，節省重複開發常用程式或物件的時間。

5. 開發程式時程縮短

基本指令的複雜程度提高及提供的功能增加，第四代程式語言在開發上比較起組合語言或高階語言，在達成相同的功能上可以利用較少指令實做完成，開發時程縮短。

6. 程式撰寫能力需求降低

第四代語言發展目的是讓使用者與電腦溝通更為容易，加上程式語言提供功能多，因此程式撰寫能力比較起組合語言或高階語言需求較低。

## 2.2 視覺化程式語言

傳統程式語言很少注重對於使用者是否容易使用，一般都假設使用者都擁有訓練有素的程式技能，隨著計算成本的降低，個人電腦的普及，讓使用電腦的人口大量增加。對許多人來說使用電腦會受限於應用程式，想做額外的事情，就必須要自行撰寫程式，但是撰寫程式並不容易[7]，針對這樣的問題，其中一個解決方法就是視覺化程式語言。

### 2.2.1 視覺化程式語言定義

視覺化程式語言已經發展多年，研究上也有許多不同的定義[2]，其中一種最常被提到的定義，就是由張系國博士所提出的『使用者利用視覺化程式工具來定義基本元件，並且將元件組合成一個有意義的視覺化程式，最後透過視覺化程式系統，來編譯並且執行程式』[2]。

要稱為是視覺化程式語言有以下幾點前提[7]：

1. 圖片可以簡潔的傳達更多的表示式
2. 圖片可以幫助記憶和理解
3. 圖片可以讓撰寫程式更有趣
4. 若是設計得當，無論使用者的語言為何，圖片所表達的意義都可以被理解

在這些前提之下，近年來有許多的視覺化程式語言被發展出來，但不同的視覺化程式語言仍然會有不同的特性。本研究針對視覺化程式語言使用以下的定義[2]：

1. 用來表示視覺化語法的元件包括圖片、圖示、動畫、其他可視的媒體以及直線
2. 利用基本元件來描繪出彼此之間的關係，就是視覺化語句
3. 視覺化程式是由許多視覺化語句所構成
4. 必須經由視覺化編輯工具來完成，編譯且執行視覺化程式。

經由這幾個定義所發展出來的視覺化工具程式，使用者由圖示、動畫及媒體瞭解元件的意義，並且藉由元件組合圖形來理解元件之間的關係，利用視覺化編輯工具來完成視覺化程式，不需要像傳統程式語言透過文字化的編輯介面，還必須記憶程式語言的指令與文法，以達到使用者容易使用的目的。

### 2.2.2 視覺化程式工具範例

市面上有眾多的視覺化程式語言及其相對應的編輯工具，每種工具都有不同的特點和訴求，根據前面章節所描述的視覺化程式語言定義，視覺化程式工具要能夠達到容易使用的目的，需包含以下特點[2][ 10]：

1. 容易編輯以及學習使用
2. 支援多樣化的媒體元件與互動功能
3. 透過引導式的使用方法，不需要撰寫程式碼就可以完成編輯
4. 提供簡單的管理方法，來設定媒體元件的屬性

歸納以上特點，發現由智勝科技所發展的互動式多媒體編輯工具編輯手，擁有人性化的介面，直覺式的操作行為，不用撰寫文字的程式碼就可以完成編輯，是符合本研究對於視覺化程式工具的定義與要求，可以作為視覺化程式工具的範例。

編輯手的操作介面如下所示：


圖 2　編輯手編輯內容頁面的操作介面

## 2.3　探討結論

程式語言歷經了約 70 年的發展，從機械語言、組合語言、高階語言直到到四代語言，發展趨勢走向使用者容易使用且快速開發，但是都是透過文字化的開發介面。視覺化程式語言的出現，開啟了人機溝通的新境界[8]，提供了圖像化的介面，讓使用者不需撰寫程式碼，透過視覺化編輯工具就可以完成程式開發。

因此，從程式語言的發展對使用者的影響，我們歸納出以下幾點結論：

1. 指令功能趨於強大

組合語言只提供了暫存器操作或流程控制，高階語言及第四代語言歸納出經常使用的功能及物件，在基本函式、物件及共用元件中提供，視覺化程式語言雖然無法讓使用者撰寫程式碼，但是透過視覺化編輯工具的每個編輯動作，都如同傳統語言呼叫指令一般。對使用者的影響有：

一. 易於開發程式

二. 縮短開發時程

三. 指令抽象程度提高

2. 開發介面圖像化

在高階語言及第四代語言，就已經有出現透過整合式開發介面來撰寫程式

碼的工具，這些工具就提供了部份的圖像來代表常用的元件，方便快速的產生程式碼。而視覺化程式語言透過視覺化編輯工具，利用圖示、動畫及媒體瞭解元件的意義，以及元件組合圖形來理解元件之間的關係，也是圖像化的介面。對使用者的影響有：

一. 對使用者友善程度增加

二. 不需要記憶程式指令

三. 程式技能需求降低

3. 適用範圍縮小

由於第四代語言有特殊用途語言的特性，設計上是為了特定範圍的使用，視覺化程式語言也容易受限於語言對於圖示、動畫所賦予的意義限制了使用範圍，因此對使用者的影響就是適用範圍縮小，但在開發適用範圍內的軟體會非常的方便及快速。

# 三、程式語言與視覺化程式語言比較

在本章節中將會比較程式語言與視覺化語言的特性，程式結構與開發流程，程式碼或物件的重用，並依照比較的結果來瞭解從程式語言延伸出來的視覺化程式語言會有什麼相異與相同性。

## 3.1 比較目標

程式語言與視覺化程式語言種類繁多，語言特性及開發工具也各有不同，因此在比較前必須選擇符合研究項目的語言。

### 3.1.1 程式語言的比較目標

高階程式語言發展至今，設計與實做的已經超過了上百種程式語言[4]，每種程式語言的目的不同，因此特性也會不同，我們無法針對所有程式語言來做比較，因此在比較程式語言與視覺化程式語言的差異之前，選擇的程式語言必須符合以下幾點：

1. 必須要是高階語言或是第四代語言
2. 程式語言應用方面廣泛，具可攜性
3. 不僅僅只有高階語言的特性，還擁有能夠處理低階位元的能力
4. 具有強而有力的結構化流程控制
5. 擁有物件導向的特性，方便重複使用與擴充

符合這些條件的程式語言，有能力開發小型的應用程式，也可以開發大型系統，應用範圍廣泛，不僅限於特定領域的使用，使用者儘管需要程式技能，但透過人類常用的文字符號來表達，有助於使用上的易於瞭解。歸納以上特點，程式語言選用 C/C++來做為比較目標，另外 C++還提供了標準樣板庫（STL – Standard Template Library）[9]，在使用上更為便利。

### 3.1.2 視覺化程式語言的比較目標

視覺化程式語言的定義是由圖示、動畫及媒體來瞭解元件的意義，藉由元件組合圖形來理解元件之間的關係，並且利用視覺化編輯工具來編輯。但是符合以上定義的並非就屬於視覺化程式語言，以下提出兩種視覺化編輯工具的類型符合以上定義，來探討是否能夠視為視覺化程式語言的比較目標。

1. Visual C++

由微軟所開發的程式語言開發軟體，提供整合式開發介面（IDE – Integrated Development Environment），透過引導來建立應用程式。Visual C++允許使用者利用內建的元件來產生 Windows 的應用程式，並且透過 Wizard 的引導來建立函式與接受訊息。Visual C++的開發介面如下所示：

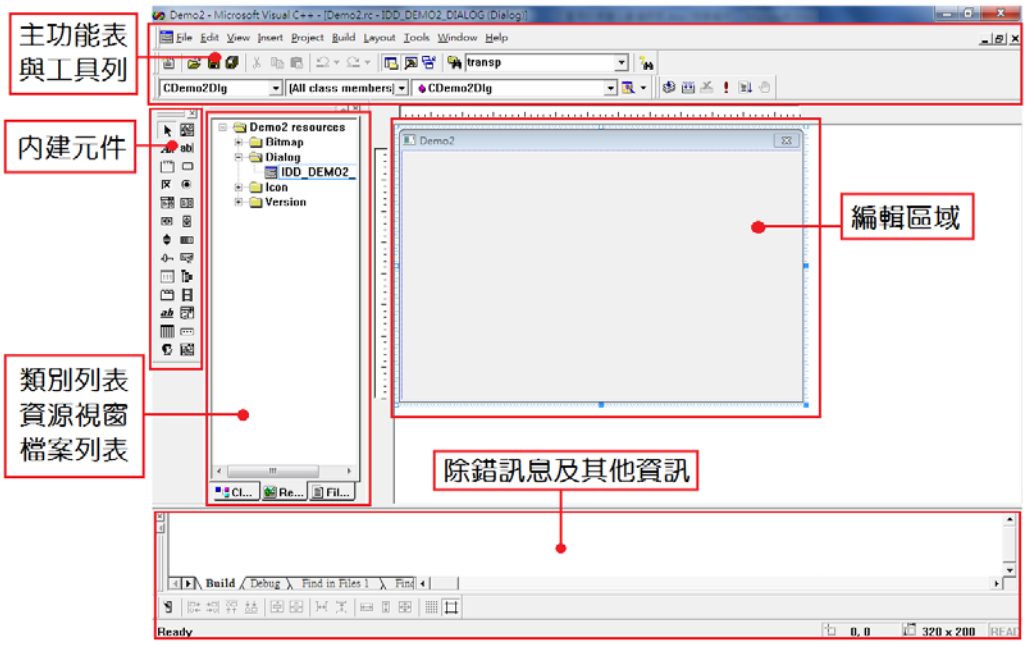



圖 3　Visual C++編輯的操作介面

另外還有提供引導式的 Wizard 來協助開發，操作介面如下所示：

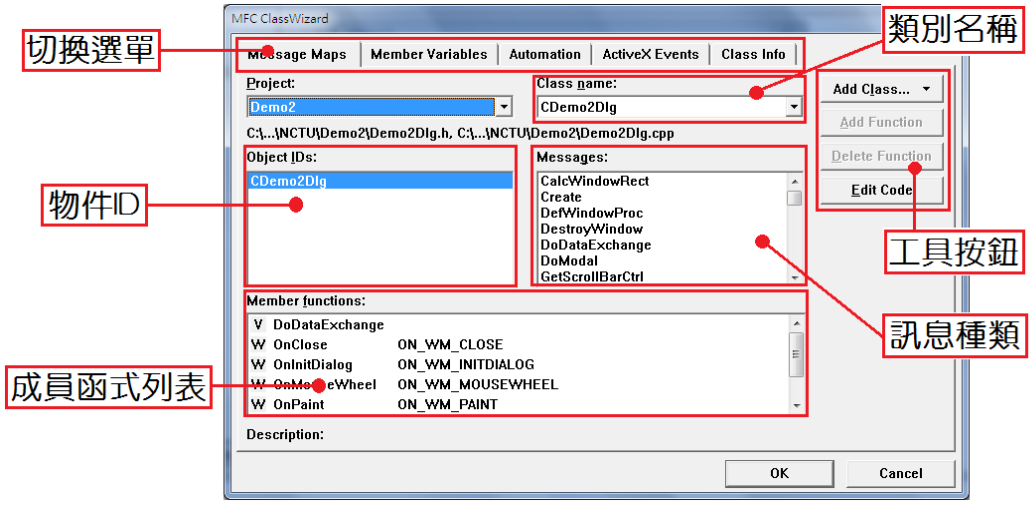



圖 4　Visual C++所提供 Wizard 的操作介面

透過開發介面及 Wizard 最終會產生程式碼，而可以在 C++所提供的程式碼

編輯環境中進行編輯除錯及執行，程式碼編輯介面如下：

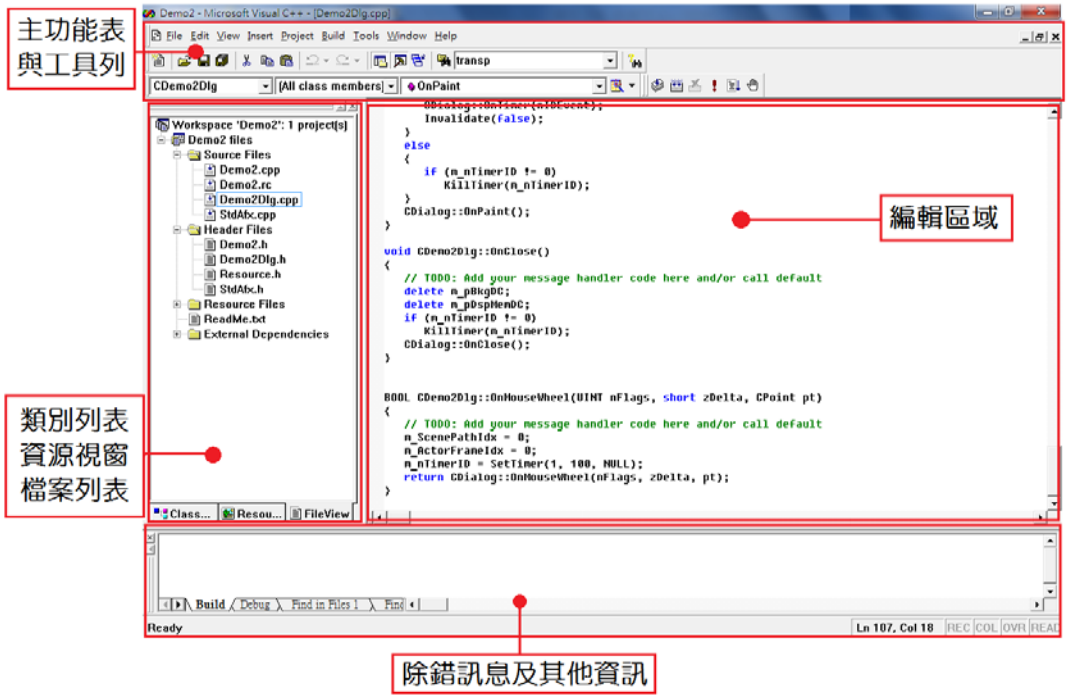



圖 5　Visual C++所提供的程式碼編輯介面

2. 編輯手、命題手

智勝科技所發展的互動式多媒體編輯工具，是一套具有劇情式與動畫式的互動式多媒體編輯工具[10]，在編輯過程中不需要撰寫程式碼，就可以完成編輯。編輯手的開發介面如下所示：


圖 6　編輯手的編輯介面

命題手是延伸自編輯手，專門用來編輯多媒體互動試題的編輯工具，除了有編輯手不需撰寫程式的特性，還提供了樣板套用的功能。命題手的開發介面如下所示：


圖 7　命題手的編輯介面

這兩類的工具都符合視覺化程式語言中對於編輯工具的需求，但是第一類的 Visual C++無法直接編輯產生結果，必須要撰寫程式碼以及經過編譯後才能執行，不符合視覺化程式語言僅需要透過引導式的使用方法，不需要撰寫程式碼就可以完成編輯的特性，因此只能稱為視覺化開發環境，並不能稱為是使用視覺化程式語言的編輯工具[11]。因此，視覺化程式語言選用編輯手與講解手來做為比較目標

## 3.2　比較項目

視覺化程式語言是透過編輯工具來撰寫程式，使用者接觸到的是視覺化程式語言中的物件、內建函式、共用元件等，因此比較項目就由內建函式開始，到子函式、類別、物件與共用元件，藉由比較來分析與瞭解兩種語言在不同範圍的關係。

### 3.2.1 內建函式

由高階程式語言所提供，經由簡單的賦予參數和呼叫就可以被執行，這就是內建函式。C 語言提供了內建的標準函式庫，包含了 ANSI 和 ISO 兩種規格[12]，針對 I/O 輸入輸出、字串處理，常用數學運算等功能提供函式，使用者編寫程式可以直接應用。C++的標準函式庫提供了 I/O、字串、容器（Containers）、演算法（Algorithms，例如 sort、

search、merge）、支援數值運算（Numeric computation）以及國際化（Internationalization，例如支援不同字元集）[13]，除了支援原本 C 語言的函式庫以外，還針對物件導向的特性，提供了物件方面的函式庫。

編輯手與命題手無法讓使用者編輯程式碼，內建函式的呼叫是經由對話表單，視窗與圖示來和使用者溝通，使用者利用鍵盤與滑鼠等設備來編輯所要表達的程式[10]，內建函式包含了物件屬性設定、劇情設定以及播放流程。

### 3.2.2 子函式

在程式語言中，子函式是完整程式的一部分，是一段特定功能的程式碼，相對於其餘的程式碼是較為獨立的。在 C/C++，使用者可以建立子函式使用，讓程式結構化，同時減少重複的程式碼。

編輯手與命題手無法讓使用者編輯程式碼，因此無法自行建立子函式。

### 3.2.3 類別

在物件導向程式語言中，類別就像是物件的藍圖，規劃出物件的屬性及狀態，物件導向程式語言的特性有封裝性（Encapsulation）、繼承性（Inheritance）、多型性（Polymorphism）、資訊隱藏（Information Hiding）與資料抽象化（Data Abstraction）[14]，使用物件導向設計可以有以下優點：

1. 以真實世界的概念來架構軟體物件
2. 重用程式碼及互通性[14]

在 C++程式語言中，允許使用者自行定義類別來產生物件，因此可以設計實做物件，來達到使用物件導向的優點，而在某些開發環境中（例如微軟的 Visual C++提供的 MFC），甚至會內建特定類別供使用者使用。

編輯手與命題手的開發環境本身就提供了類別，在編輯程式就使用了物件導向的概念，但無法自行撰寫程式來擴充使用。

### 3.2.4 共用元件

為了減少軟體開發的成本與開銷，許多軟體專案已經不再是自行開發所有的元件，

而是透過整合性的軟體元件來開發[6]，透過共用元件的產生，可以讓撰寫程式更為獨立且模組化，可以在不同的程式呼叫共用元件，將程式的共用性大大的提昇。C++允許使用者自行開發共用元件，相同的在某些開發環境中（例如微軟的 Visual C++）也會內建共用元件。

編輯手並沒有共用元件的設計，但是在命題手中，利用樣板套用，將內建的試題樣板重複使用，就是共用元件的設計。

### 3.2.5 程式開發步驟

C/C++透過文字化的編輯介面，利用程式語言的內建函式、子函式、類別物件或是共用元件，來將程式編輯完成。文字的程式碼無法直接執行，必須經過編譯器（Compiler）轉成機械語言，再由連結器（Linker）轉成執行檔，最後才能得到結果，若是修正程式則必須重複以上步驟才能看到修正後的成果。

編輯手與命題手提供圖像化的介面，在編輯工具中利用滑鼠和鍵盤透過表單和視窗來撰寫程式，有著所見即所得的優點，編輯完成的結果可以透過編輯工具的執行器看到結果，對使用者而言較為直覺且方便。

## 3.3 比較結果

視覺化程式語言的發展對傳統程式語言有著重大的影響[15]，讓使用者不需要瞭解程式語言的架構，指令的邏輯或是編譯器的使用方式，只需要透過簡單的介面就可以完成編輯，達成和使用傳統程式語言來編輯程式相同的目的。視覺化程式語言會隨著編輯工具應用範圍不同，而產生不同的特性，但還是會有一些共通的發展目標[15]：

1. 讓編輯程式更普及化，不只有侷限程式設計師
2. 提高編輯程式時的正確性，也就是提昇程式的品質
3. 提昇編輯程式的效率

因此視覺化程式語言會有下列的屬性[15]：

1. 具體性：抽象性的相反，具體的利用圖示或箭頭等方法來表達一段程式
2. 直接性：減少使用者為了達成目標所需要的動作
3. 清晰性：直接表達程式的含意，不需要使用者去猜測
4. 立即性：在視覺化編輯工具中，可以對於編輯內容得到立即的回饋

透過多方面比較程式語言和視覺化程式語言，結果如下表所示：

表 1　程式語言與視覺化程式語言的比較

| | 程式語言 | 視覺化程式語言 |
|---|---|---|
| 內建函式 | 提供內建函式，支援一般檔案存取，字串處理或使用物件 | 提供內建函式，針對視覺化物件及劇情邏輯的操作 |
| 子函式 | 使用者可自行建立子函式 | 無法自行建立子函式 |
| 類別 | 使用者可自行建立類別，或是開發環境提供內建類別 | 僅能使用開發環境提供的類別，無法自行定義 |
| 共用元件 | 使用者可自行建立共用元件，或是開發環境提供內建共用元件 | 僅能使用開發環境提供的共用元件，無法自行定義 |
| 程式開發步驟 | 需經過編譯、連結、執行後才能看到編輯結果 | 開發環境中編輯完成後就可以利用執行器看到結果 |

可以看到視覺化程式語言擁有諸多程式語言的特性，重用方面程式語言包含程式碼重用與物件重用，視覺化程式語言只有物件重用。但是因為無法讓使用者自行編輯程式，因此在自行擴充方面較為欠缺，但是透過視覺化的編輯工具對於使用上的友善程度是較佳的。

# 四、重用機制的探討

本章節首先介紹重用抽象程度不同的 Programming-in-the-small 與 Programming-in-the-large，再來探討重用機制對於程式語言與視覺化程式語言的影響，包括設計上的重用，設計及實做上的重用，以及應用上的重用，並利用實際舉例－如何實做出可和使用者互動且類型多樣化的試題程式，來探討不同方面的重用各有哪些優缺點。

## 4.1 Programming-in-the-small versus Programming-in-the-large

在軟體開發中，Programming-in-the-small 與 Programming-in-the-large 代表兩種實做程式的方法[16]，根據程式規模不同選用不同的方式，對於軟體重用的抽象程度也不同。

Programming-in-the-small 代表實做一個小型程式的行為，用較少的人力實做少量的程式碼，程式行為簡單，生命週期短暫，重用程度只在於程式碼的重用。
Programming-in-the-large 代表實做一個大型程式，可以是較多的人用較少的時間完成，也可以是較少的人用較長的時間完成，需要利用模組互連語言（Module Interconnection Language）來完成，程式的撰寫模組化，重用程度為模組重用。另外一種意義，是利用程式撰寫系統的狀態，包括訊息的傳遞和狀態的轉換，而非實做所有狀態邏輯部份的程式碼。

總結前面所述，Programming-in-the-large 代表的是較為抽象的程式撰寫，利用模組來構成程式，或利用程式碼撰寫系統的狀態改變，而 Programming-in-the-small 則是撰寫小型程式，或是模組內的程式碼部份。

## 4.2 重用機制的演進

軟體重用可以讓使用者充分利用已開發的成果，並且促進軟體開發的效率和品質[17]，從程式語言的發展，組合語言到第四代語言，程式撰寫從使用內建函式，到使用物件導向和共用元件，重用單位從指令發展到物件，抽象化的重用概念也在不斷的提昇。

視覺化程式語言也有重用的概念，在視覺化編輯工具中，每一個圖示或動畫都代表一個可重用的物件，透過滑鼠或鍵盤就可以產生重用物件，因此視覺化程式語言一樣有效率和品質的優點，但是目前為止仍只限於在 Programming-in-the-small 的領域中。視覺

化語言在 Programming in the small 已經取得重大進展，但難以建構大型應用程式，因此未來的發展須走向 Programming in the large[15]，讓視覺化程式語言不僅有物件重用，而是能夠更抽象化到模組的重用。

## 4.3 設計上的重用 – Design Pattern

Design Pattern 是基本的設計工具，用來改善現有的程式碼，用來提高效率的工具，可以改善設計技巧、提昇專案品質 [18]。Design Pattern 的精神在於提供解決的方案，而不直接用來完成程式碼的寫作[19]，應用在軟體設計的過成中，使用前人累積下來的經驗，避免在設計時犯下相同的錯誤，或是讓設計架構更有彈性及擴充性，都是在設計時導入 Design Pattern 的原因。

應用在實際的例子中，要實做出可和使用者互動且類型多樣化的試題程式，在設計時就必須考慮到多樣化的試題類型，程式必須能夠靈活的擴充試題類型，因此在設計時選用了樣板模式（Template Pattern），樣板模式的使用條件如下：

1. 使用條件在於某些類別的演算法中，實做了相同的方法，造成程式碼的重複
2. 控制次類別必須遵守的一些事項

在試題程式中，會面臨到相同類型的試題重複產生，每題試題都實做試題邏輯和顯示部份的程式碼，容易造成程式碼的重複，因此在設計時就將試題邏輯與顯示部份提出寫成樣板，將不同的資料套用在相同的樣板上，就可以快速產生類型相同的試題。

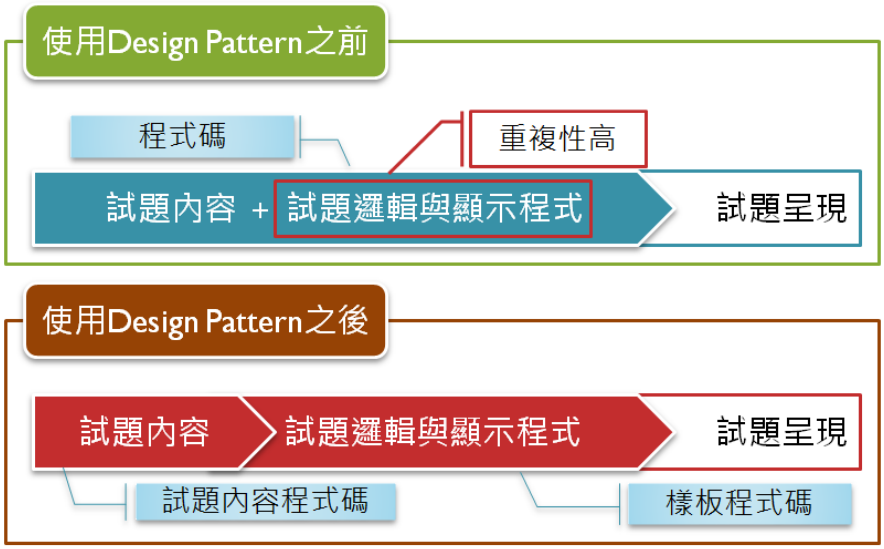



圖 8　試題程式使用 Design Pattern 的改變

題目中另外還有需求，就是多樣化的試題類型，設計上使用 Design Pattern 將試題邏輯與顯示提出成為樣板的另一個好處，就是相同的試題內容，套用到不同的樣板上，就可以產生不同類型的試題。


圖 9　試題程式套用不同的樣板

設計上使用 Design Pattern 中的樣板套用，優點在於重用程式碼，縮短程式開發時程，確保共用部份的一致性，並且在設計架構上也保有擴充性。缺點是整個程式的實做仍然需要撰寫程式碼，並且透過文字的編輯介面，而且修正程式碼後需要經過編譯、連結，最後執行程式才能看到編輯結果。

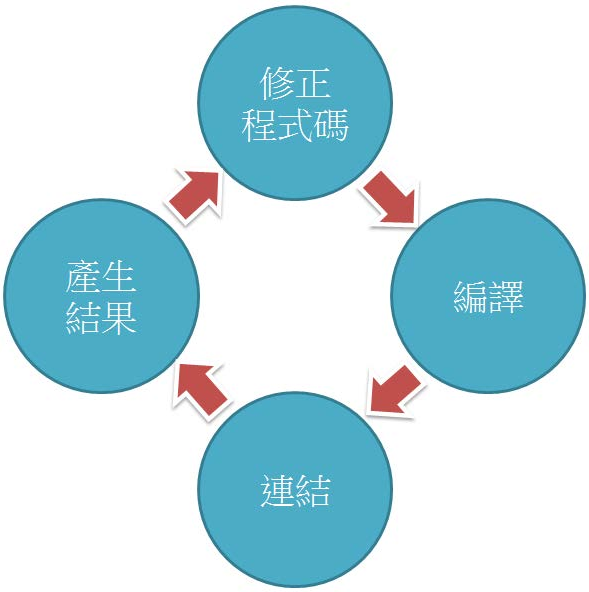



圖 10　試題程式使用 Design Pattern 的修正程序

## 4.4　設計與實做上的重用 - UML

統一塑模語言（UML –Unified Modeling Language）用來溝通系統、是一種先進的、普遍的、應用範圍廣泛的、有工具支援、並且在業界通用的一種語言[20]。在軟體開發中，利用 UML 的優點在於透過視覺化的 UML 開發環境來建立軟體架構清晰易懂，無論實做上採用什麼程式語言，透過 UML 都容易與人溝通，並且容易修改與建立文件。

應用在實際範例中，要實做出可和使用者互動且類型多樣化的試題程式，設計概念上仍然是使用樣板，但在規劃架構時利用 UML 建立物件類別，經由工具轉成程式碼，實做邏輯部份的程式碼即可完成。UML 就是物件導向的概念，可利用重複產生物件來建立類型相同的試題，多樣化的試題可以利用物件導向中繼承（Inheritance）或多型（Polymorphism）的概念，來產生不同類型的試題。

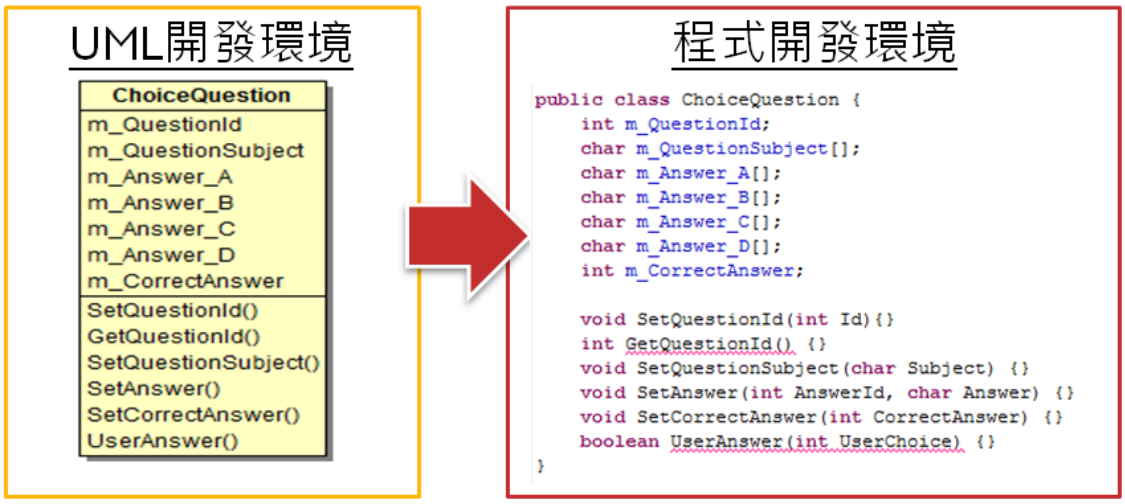



圖 11　試題程式使用 UML

實做上使用 UML，優點在於透過視覺化的 UML 開發環境建構程式，圖像化介面方便使用且架構明顯易懂，設計時就有物件導向的特性，產生文件也很方便。缺點是轉換成程式碼之後，仍然需要程式設計師實做細節的程式碼，開發環境是文字介面，並且在修改時，若是僅有程式碼修正，需要經過編譯、連結，最後執行程式才能看到編輯結果，若是架構上的修正，需要重新將 UML 轉換成程式碼，更加複雜。

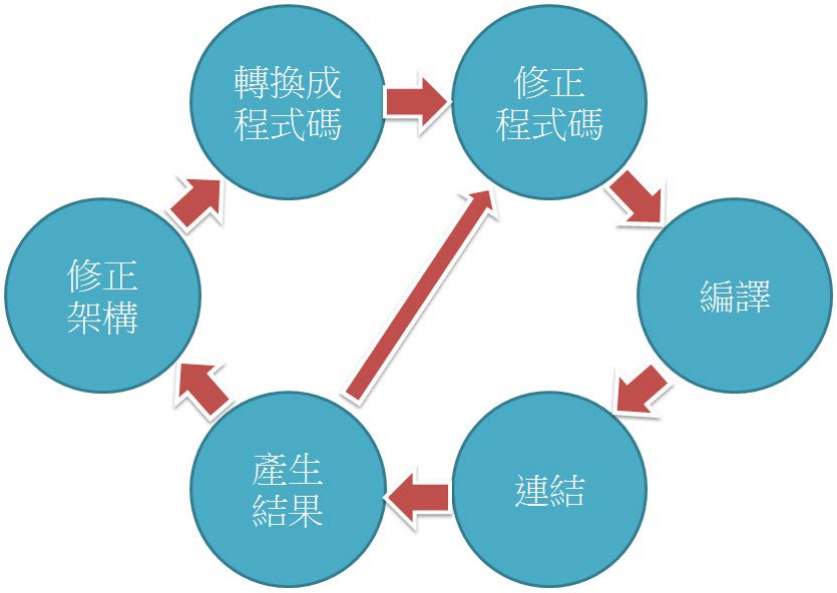



圖 12　試題程式使用 UML 的修正程序

## 4.5　應用上的重用 - 多媒體樣板套用

多媒體樣板套用，是利用樣板模式的精神，在視覺化程式語言中，對於重複性高的互動式多媒體，為了簡化使用者在編輯時的一些重複性，因此提出樣板套用[10]，將資料套用到樣板內，就可以快速產生多個互動式多媒體的應用程式，避免編輯重複的動作。

應用在實際範例中，要實做出可和使用者互動且類型多樣化的試題程式，利用多媒體樣板套用機制，將資料套用到樣板內即可完成，套用時會檢查資料類型，避免錯誤發生，多樣化的試題可以利用套用不同的樣板來產生。多媒體樣板套用不需要撰寫程式碼就可以編輯出互動式的多媒體應用程式。

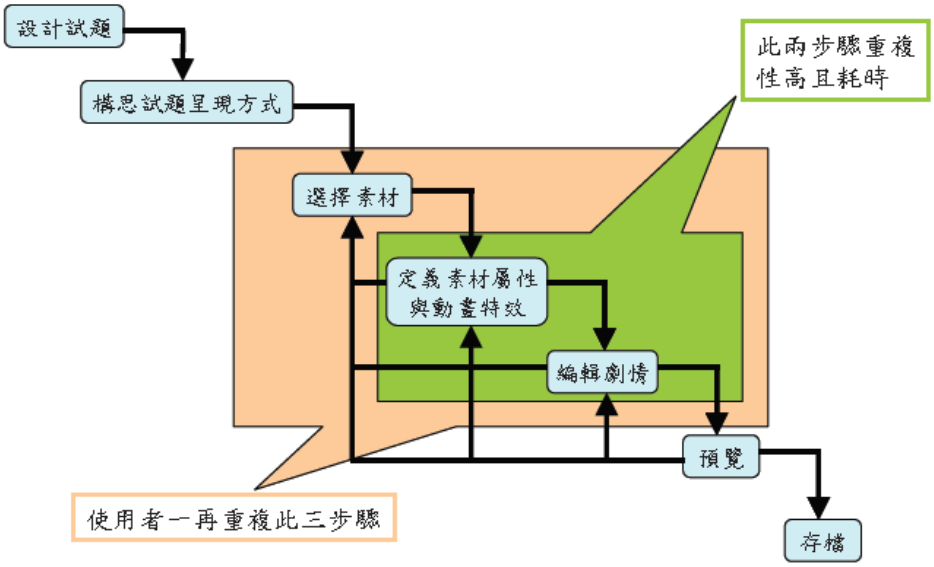



圖 13　多媒體試題的編輯流程

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

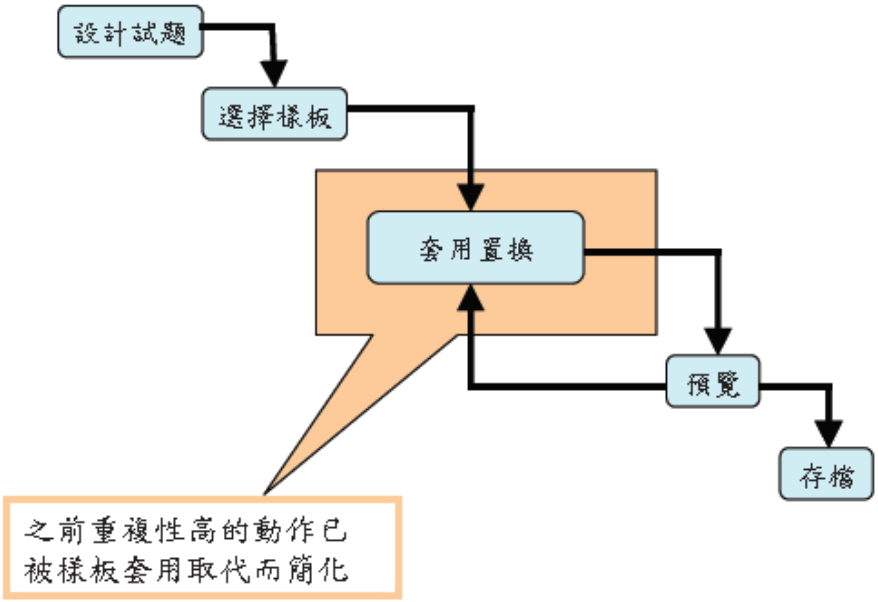



圖 14　試題程式使用多媒體樣板套用的編輯流程

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

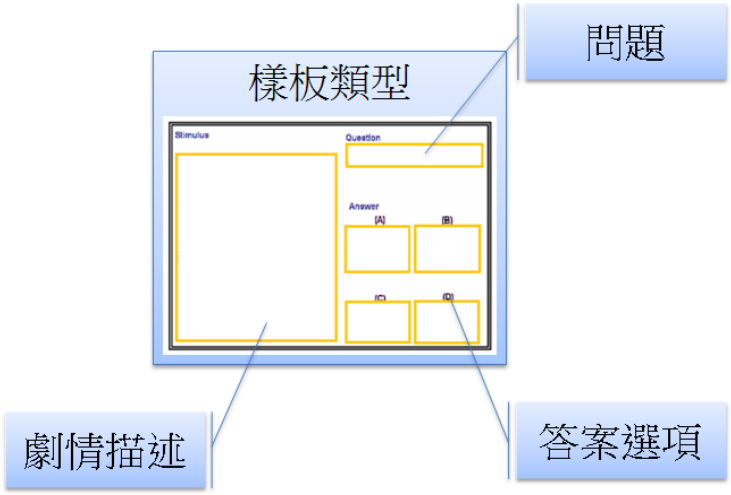



圖 15　多媒體樣板屬性說明

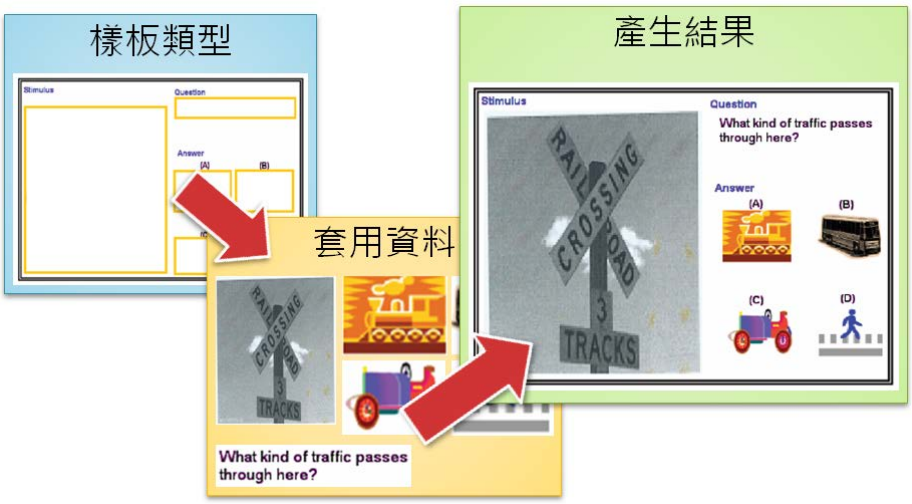



圖 16　多媒體樣板套用範例一

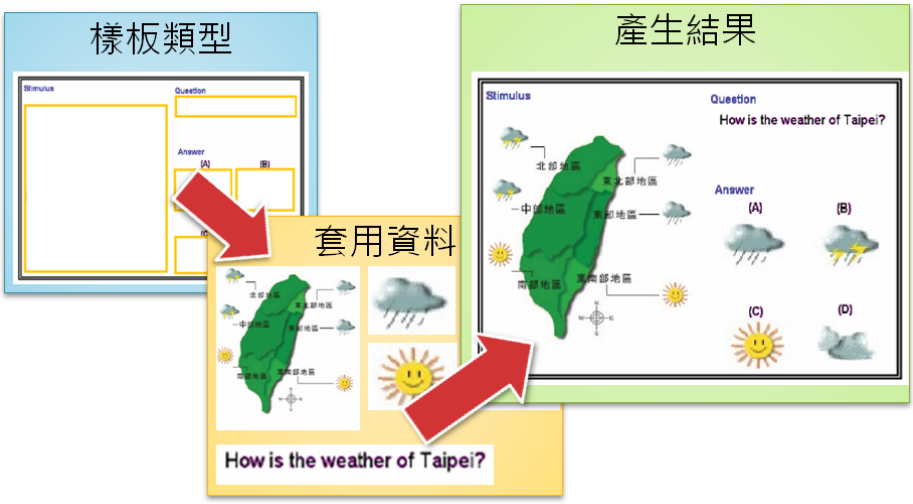



圖 17　多媒體樣板套用範例二

實做上使用多媒體樣板套用，優點在於透過視覺化的多媒體樣板套用編輯程式，圖像化介面方便使用，設計時就有物件導向的特性，使用者不需要有程式技能即可使用，不需要依靠程式設計師，而且產生的程式品質穩定，修改時只需要重新選擇套用資料，就可以看到成果。缺點是樣板的種類受限制，要擴充必須依賴多媒體樣板套用編輯程式的提供者，實際應用範圍也會受到樣板類型的限制。

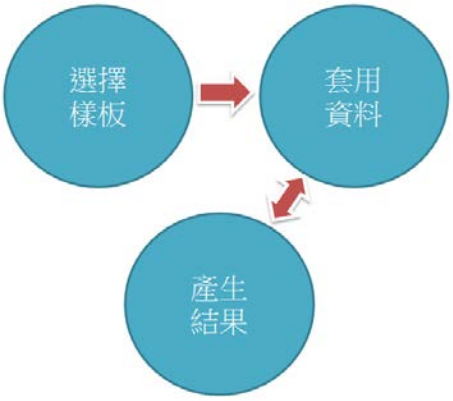



圖 18　試題程式使用多媒體樣板套用的修正程序

# 五、視覺化劇情重用

本章節將說明視覺化劇情重用的精神和意義，並且舉例說明視覺化劇情重用的應用。

## 5.1 說明

前面討論了程式語言的發展趨勢與視覺化程式語言，並且比較程式語言與視覺化程式語言在內建函式、子函式、類別、共用元件及程式撰寫的相同相異性，也討論到 Design Pattern、UML 與多媒體樣板套用三種重用方式的優缺點。從前面的討論中，看到了程式語言的發展，從 Programming-in-the-small 發展到 Programming-in-the-large，重用機制一直在隨著語言發展而發展，從程式碼重用、物件重用到共用元件，並沒有停歇。視覺化程式語言發展時，就具備了高階語言中物件重用的機制，將視覺化程式語言中的圖示視為一個重用元件，在編輯視覺化程式中加以重用，但是在視覺化程式語言的重用機制近年來並沒有突破性的發展。

本文提出了可重用的視覺化劇情，就是希望在視覺化程式語言的領域，將重用機制提昇更抽象化，把重用單位從物件提升到劇情，透過編輯工具編輯視覺化程式時，每個圖示代表的是一段有意義的劇情，藉由套用機制將劇情內需要的演員置換，以達到相同的劇情由不同的演員來扮演而一再重複使用，也就是視覺化程式語言中的 Programming-in-the-large。

### 5.1.1 什麼是視覺化劇情

視覺化程式語言是使用廣義的圖示來代表物件，呈現出邏輯部份（程式的含意）與物理部份（圖像）[22]，編輯結果就是視覺化劇情。因此視覺化劇情包含了一組演員，並且描述演員所需要出現的順序以及彼此的關係[2]，這些關係可以分成空間與時間上的關係。空間上的關係描述了演員的位置、大小、移動路徑以及所在層數，時間上的關係描述演員及演員間出場的順序。

### 5.1.2 如何規劃出重用的視覺化劇情

要將視覺化劇情重用，首先要考慮的就是怎麼定義重用劇情，哪些劇情要成為重用的單位。在這裡我們參考 Design Pattern 的作法，對於 pattern 的定義是『每個 pattern 描述了一個在我們的環境中一再重複發生的問題，然後描述了核心的解決方案，你可以

不需要重複做相同的事就可以使用這個解決方案上百萬次』[19]，因此對於可重用視覺化劇情的定義，要符合以下原則：

1. 重用劇情需要符合應用範圍內的使用

   視覺化程式語言的定義是廣泛通用的，被研究很多年，並且應用在不同領域[2][23][24][25]。但是在實際使用上，應用範圍會受到視覺化編輯工具中提供物件的影響而受到限制，就如同第四代語言一樣。因此要提出可重用的視覺化劇情，必須也符合實際上的應用範圍，才能夠在編輯時可以一再的被使用。

2. 重用劇情在應用範圍內有機會被使用數次

   在應用範圍內的劇情若是重用機率小，直接透過視覺化編輯工具自行編輯即可，只有重用機率高的劇情才適合成為可重用的視覺化劇情，原理就和 pattern 的應用是相同的。

3. 重用劇情長度要適當

   劇情長度會決定是否方便重用，若是劇情太短則每次重用都需要拼湊數個劇情，效用和直接用物件編輯差異不大。若是劇情太長又會難以重用，或是重用後需要再編輯劇情內容，因此長度必須適當。

符合以上三個原則的劇情，就可以成為可重用視覺化劇情，應用在視覺化程式的編輯上。一個完整的 design pattern 必須包含四個要素，分別是名稱，問題描述，解決方案以及效果[19]，一個完整的可重用視覺化劇情，除了劇情內容符合以上的定義外，還應該包括了劇情名稱，劇情描述，演員描述，與應用範圍，這樣的可重用視覺化劇情才容易讓人理解及使用。

### 5.1.3 如何使用重用的視覺化劇情

使用視覺化程式語言編輯時，每段視覺化程式碼必須重複設定重用元件的素材及屬性，所有元件設定結束後編輯劇情，預覽時有錯誤必須重新設定素材或編輯劇情，最後才能產生一段視覺化程式碼，若是規模較大的視覺化程式，就必須重複以上步驟數次，浪費時間且沒有效率。編輯流程如下圖：

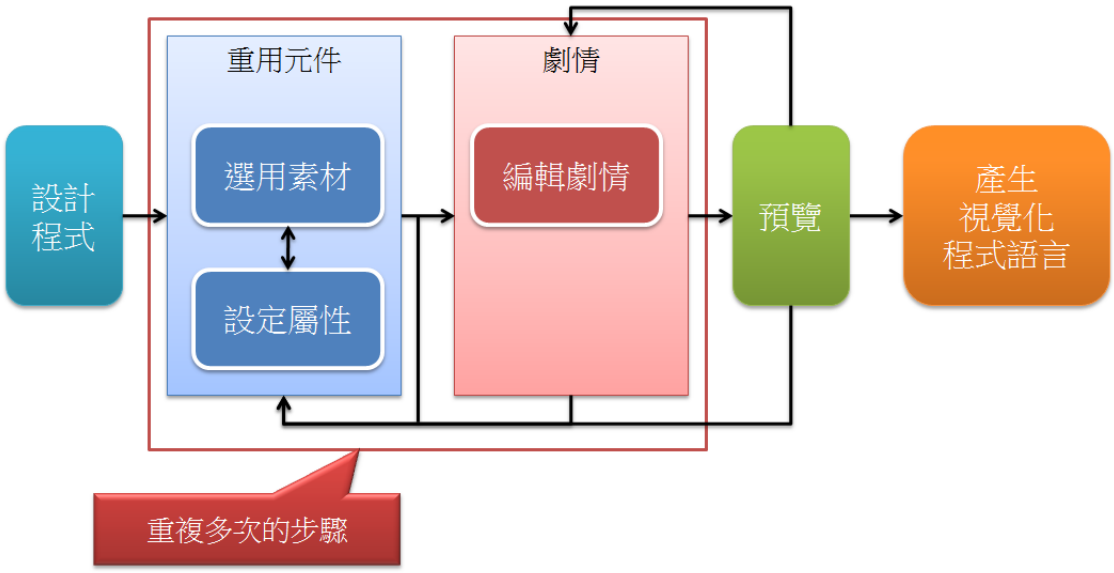



圖 19　無可重用的視覺化劇情時，編輯視覺化程式的步驟

若能將常用的視覺化劇情擷取出來當作重用單位，每段視覺化程式碼的編輯步驟就會改成為先選擇重用的視覺化劇情，再將重用元件的素材帶入套用即可完成，當劇情數量眾多時，就可以節省可觀的時間與人力，並且編輯成果品質穩定。編輯流程如下圖：

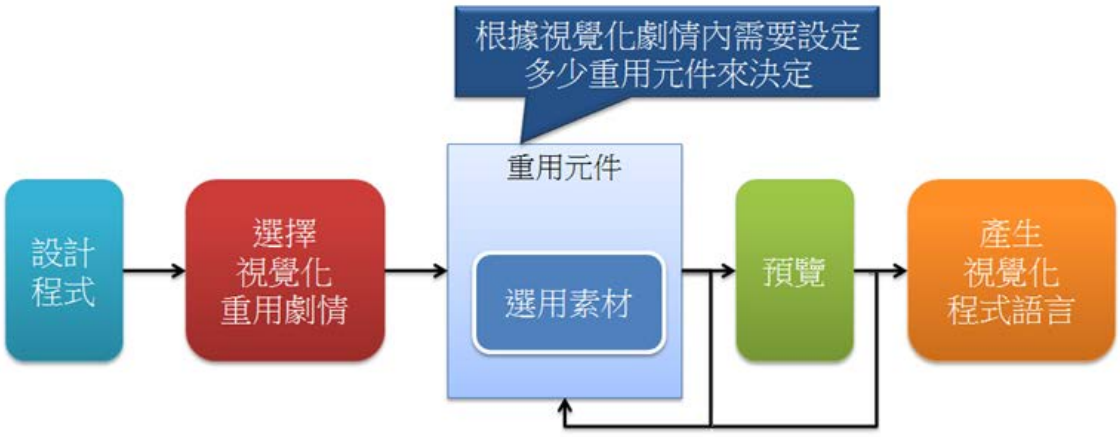



圖 20　利用可重用的視覺化劇情時，編輯視覺化程式的步驟

## 5.2　應用上的範例

這一節將針對可重用的視覺化劇情舉例，並且介紹兩個範例來具體說明，第一個是互動式多媒體試題的劇情重用，第二個是手機人機介面的劇情重用。

### 5.2.1 可重用的視覺化劇情範例

一個完整的可重用視覺化劇情，包括了劇情名稱，劇情描述，演員描述，應用範圍，與視覺化劇情，下面就利用一個互動式多媒體選擇題的劇情來說明：

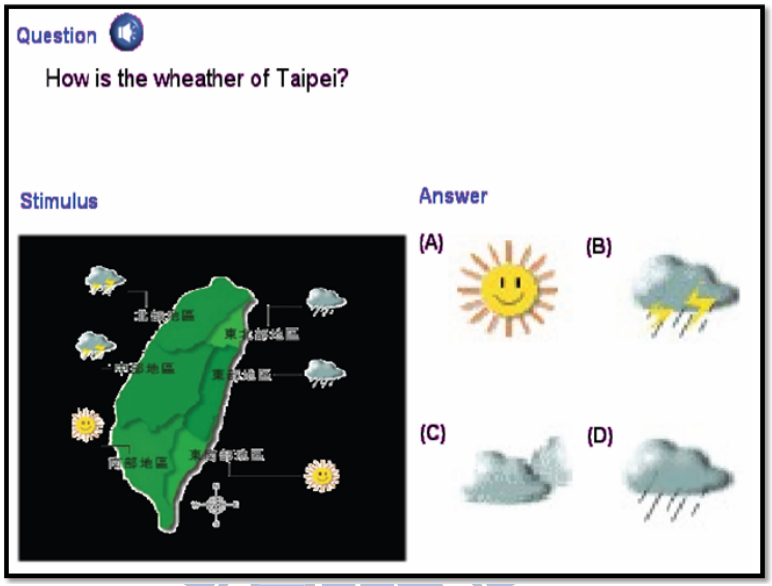



圖 21　互動式多媒體選擇題範例

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

- 劇情名稱：互動式多媒體選擇題
- 劇情描述：利用動畫、聲音、圖型與文字來與使用者溝通的互動式多媒體選擇題
- 演員描述：利用三個文字演員來代表以下項目，分別是劇情描述、問題與答案選項。每個項目的演員描述如下：
  - 劇情描述：利用播放影像來描述劇情，包含一個影像演員
  - 問題：利用文字與聲音來提出問題，包含了一個文字演員與聲音演員
  - 答案選項：利用圖形來表現答案選項，讓使用者直接點選，包含了四個圖形演員與四個文字演員
- 應用範圍：應用在互動式多媒體教學測驗中
- 視覺化劇情：根據應用上使用的視覺化程式語言，來放置不同內容，但相同邏輯的視覺化劇情。

### 5.2.2 互動式多媒體試題的劇情重用

在互動式多媒體試題的劇情重用範例中，重用的單位是完整的試題劇情，透過套用的方式，取代原本劇情中的多媒體物件，並藉由每次取代時的檔案型態檢查，來避免套

用錯誤的檔案造成編輯失敗，確保最後產生試題的品質。以下的範例將會執行套用劇情描述，問題與答案選項，產生內容完全不同的試題。以下步驟說明重用試題劇情的套用機制，套用完成就可以產生新試題。

第一步：開啟可重用視覺化劇情

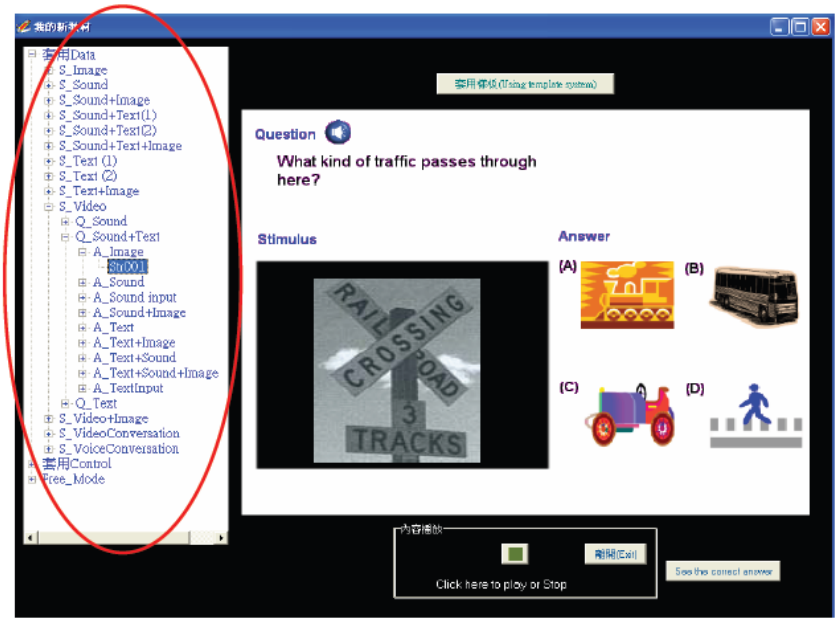



圖 22　互動式多媒體試題的物件列表

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

第二步：套用劇情描述：選擇檔案會先進行檔案型態的過濾，適合的才能選擇

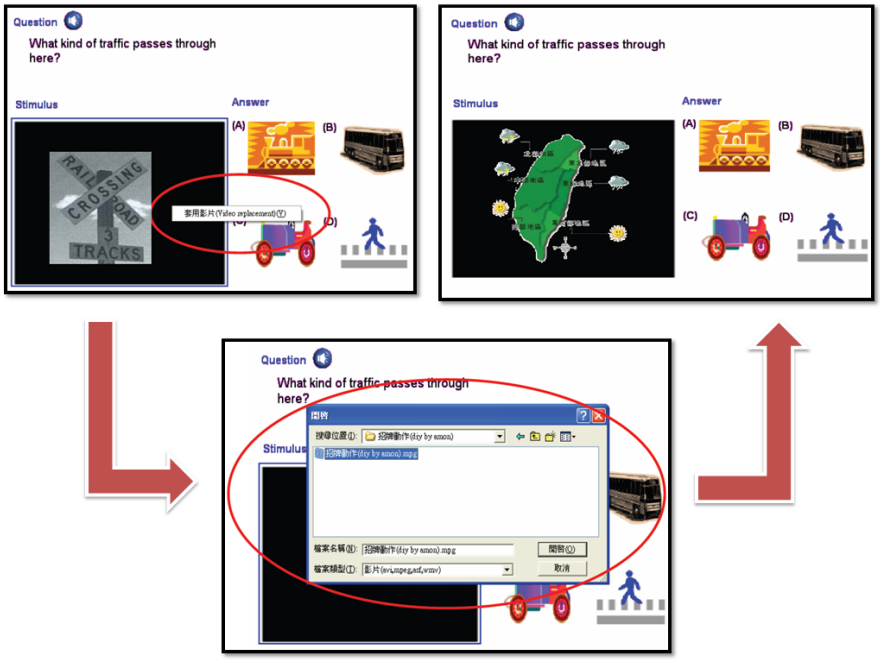



圖 23　互動式多媒體試題套用劇情描述

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

第三步：套用問題的語音部份：套用時會先列出系統內建素材，選擇檔案也會進行檔案型態的過濾


圖 24　互動式多媒體試題套用問題的語音部份

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

第四步：套用問題的文字部份：套用時會顯示文字編輯區，選擇檔案也會進行檔案型態的過濾

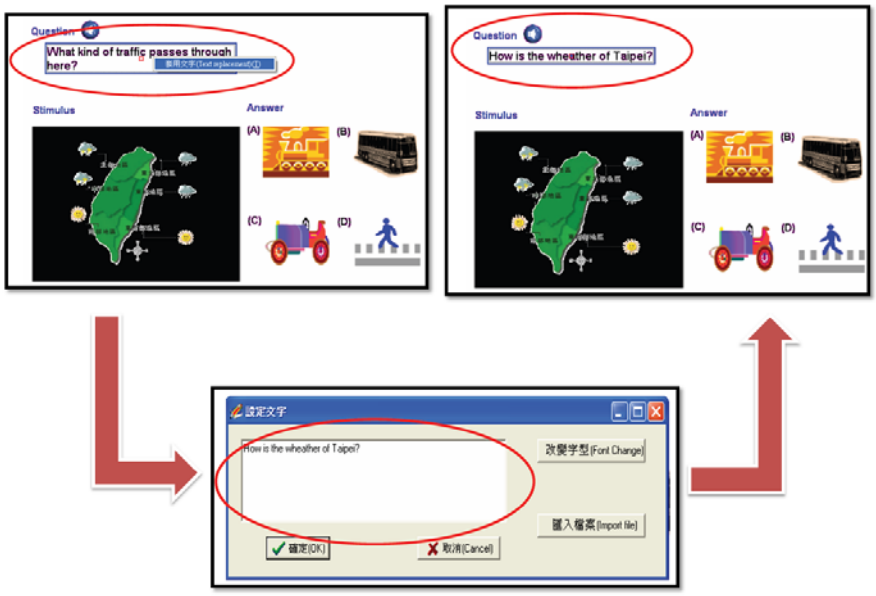


圖 25　互動式多媒體試題套用問題的文字部份

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

第五步：套用答案：套用時會先列出系統內建素材，選擇檔案也會進行檔案型態的過濾

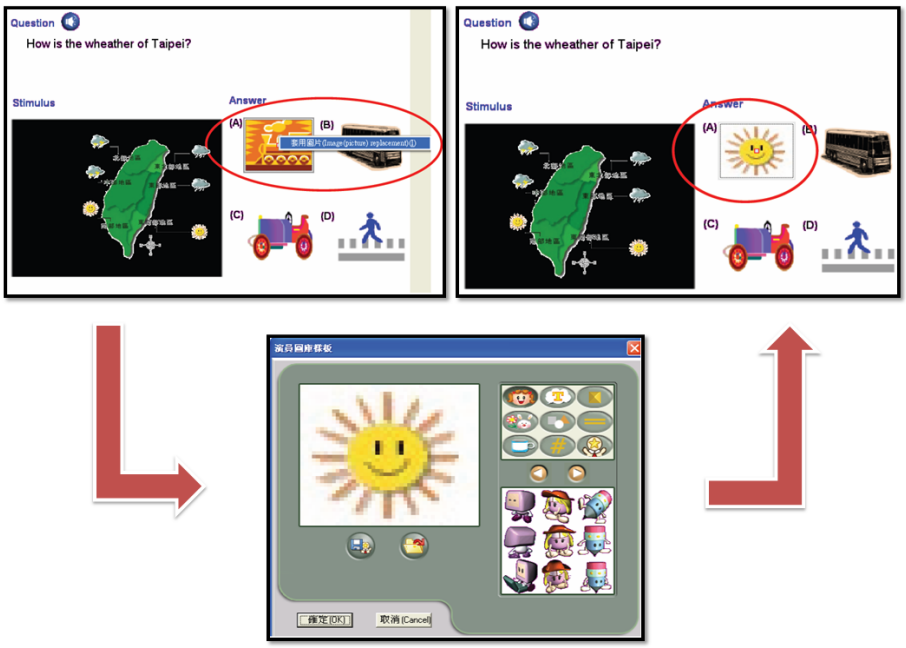



圖 26　互動式多媒體試題套用答案

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

第六步：套用完成，產生新試題

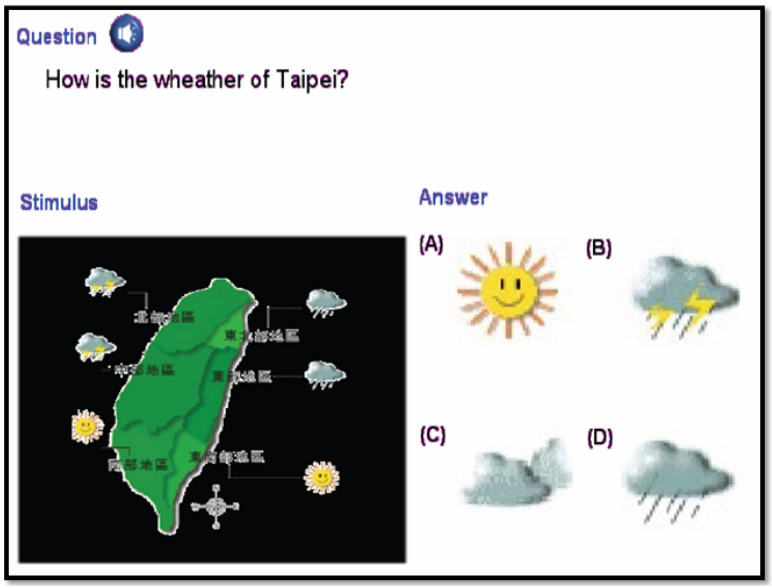


圖 27　互動式多媒體試題套用完成

資料來源：互動式多媒體的視覺化劇情編輯機制[10]

### 5.2.3 手機人機介面的劇情重用

在手機人機介面的劇情重用範例中，重用的單位是一個畫面的操作劇情。手機人機介面系統中，每個畫面的顯示內容，按鍵及觸控的行為，由客戶制定，交給專案經理寫成文件，再由程式設計師實做，容易因解讀文件認知不同或編碼時犯錯造成產品和客戶期望有落差，也會產生重複的程式碼。若是將手機中經常出現的劇情歸納，並抽象化成為可重用的視覺化劇情，客戶或專案經理就可以直接編輯內容及使用流程，藉由簡單的步驟完成另外一套人機介面。以下說明將手機人機介面提昇成為可重用視覺化劇情的概念。

1. 客戶提出需求，由專案經理訂立文件，利用文字與示意圖詳細說明畫面規格及行為

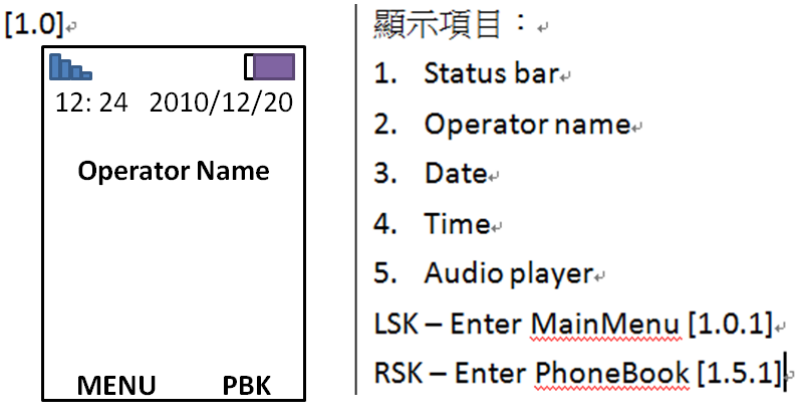



圖 28　人機介面的文件示意圖

2. 程式設計師用程式碼將畫面實做出來，並且用程式碼編寫使用流程

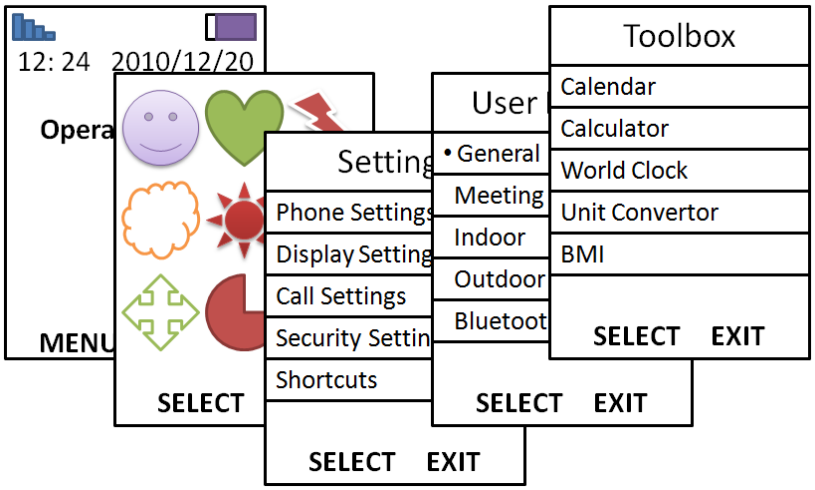



圖 29　人機介面利用程式碼實做

3. 許多畫面樣式相近，若個別開發容易產生重複的程式碼，因此歸納出相似處

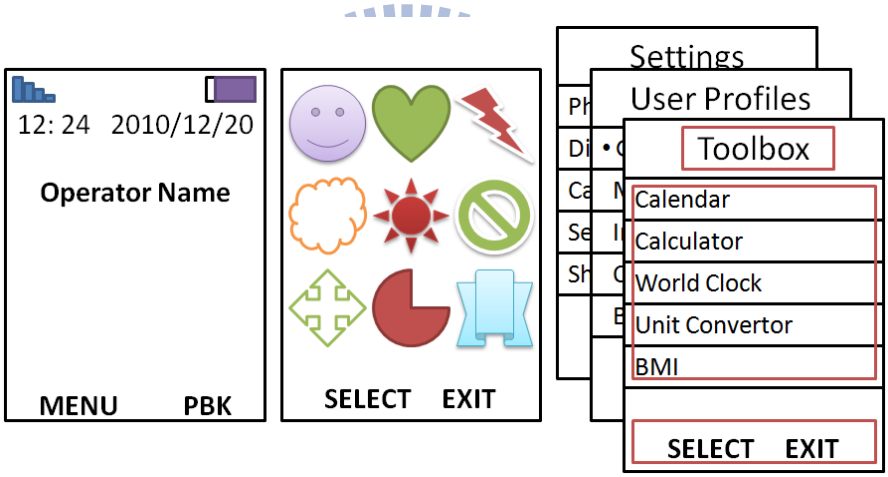



圖 30　人機介面歸納出相似的畫面

4. 將相似的畫面作成樣板，在編輯程式時將資料帶入，產生畫面

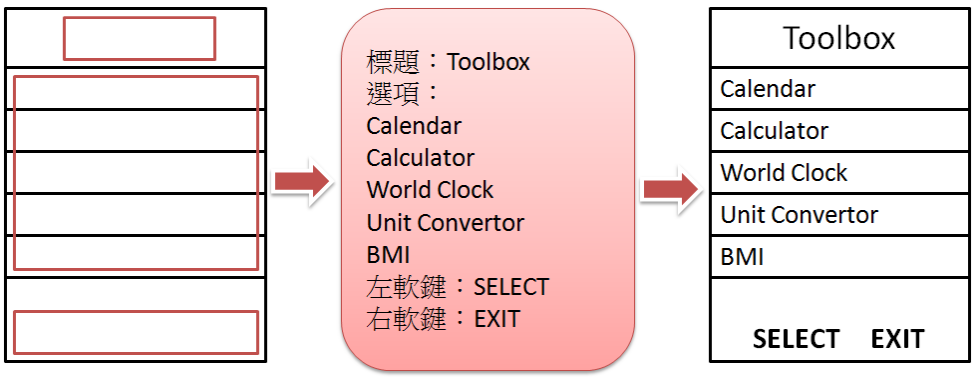


圖 31　人機介面利用樣板產生類似畫面

以上步驟是利用程式語言的開發，缺點是需要程式技能，而且利用程式碼編寫使用流程，難以閱讀且只有程式設計師可以理解，客戶和專案經理需要看到結果才能知道是否有誤。

若是利用視覺化程式語言來開發，可以不用寫程式，利用現有的物件組合成畫面，並賦與邏輯上的行為，就可以將完成程式的開發，但相同劇情就必須一再重複撰寫，耗時且可能會出錯。

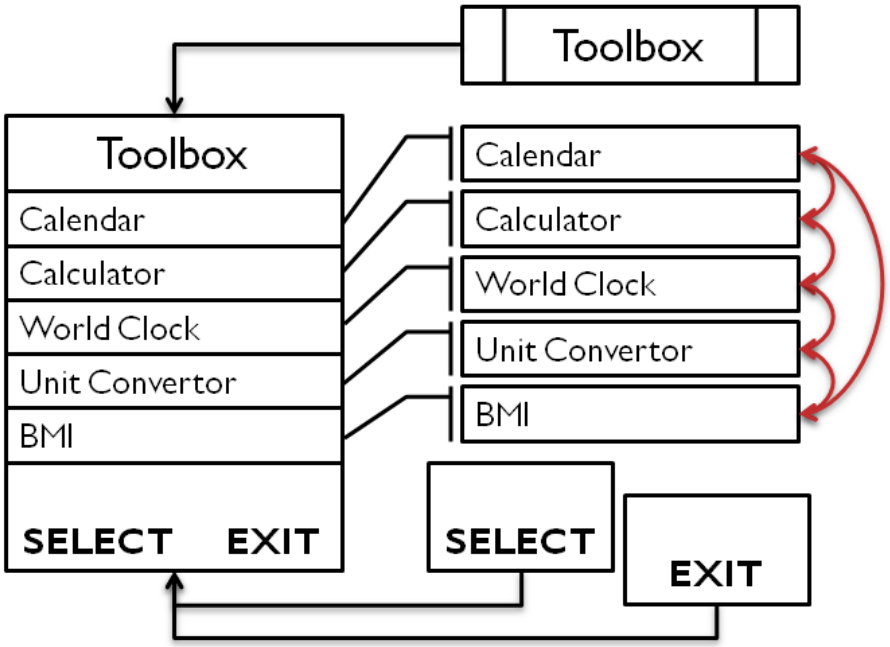



圖 32　利用視覺化程式語言實作人機介面範例

將畫面與行為抽象化成為可重用視覺化劇情，利用視覺化編輯環境與套用機制，就可以完成新的人機介面流程。

5. 將畫面與行為抽象化成為可重用視覺化劇情，利用套用機制產生新的畫面

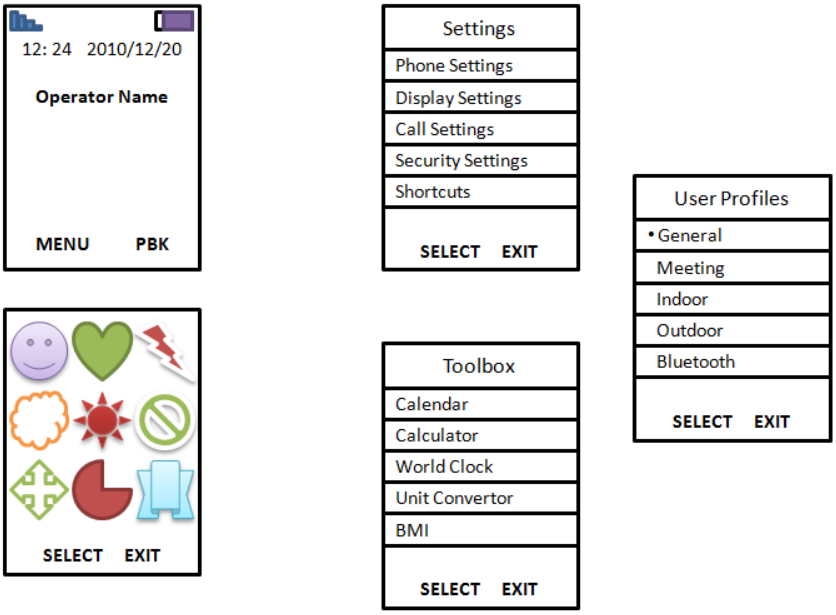



圖 33　人機介面將畫面與行為抽象化成為可重用視覺化劇情

6. 利用圖示與線條來規劃出使用流程，不需要會寫程式，簡單易懂

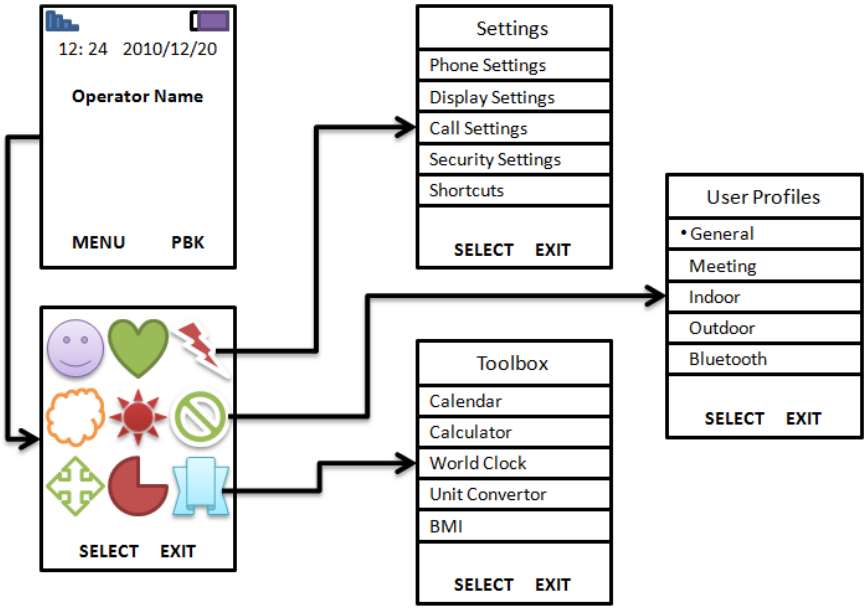



圖 34　人機介面利用可重用視覺化劇情編輯

## 5.3　可重用視覺化劇情的優缺點

程式語言的物件重用與共用元件，讓程式開發速度與品質上皆有所提昇。可重用視

覺化劇情導入後，除了保有視覺化程式語言不需撰寫程式與圖像化介面的優點之外，還有以下優點：

1. 可重複使用

   在不同的程式中都可以套用相同的劇情，而且沒有次數限制。

2. 開發流程簡化

   直接套用現有的劇情，省略重複編輯劇情的動作。

3. 程式品質穩定

   軟體重用一直是提升工作效率與軟體品質最有效的方法[21]，此外要成為可重用的劇情，就像制定共用元件一樣，經過審核驗證後才能提供出來使用，因此程式若是用可重用劇情來編輯，程式品質上就有一定程度的保障。

缺點則在於：

1. 應用範圍受限制

   這個缺點與程式語言發展所產生的缺點相同，編輯程式的方向會受到實際應用上的編輯工具提供的重用劇情內容而受限制，無法編輯其他範圍的應用程式。

2. 使用者無法自行擴充重用劇情

   由於可重用視覺化劇情發展的目標，就是希望包括沒有程式技能的人都能方便使用，因此若是沒有適當可重用的劇情，就必須自行編輯劇情來使用，而無法自行擴充可重用的視覺化劇情

# 六、結論

## 6.1 總結

為提升視覺化程式語言的開發速度，本研究針對視覺化程式語言的重用，提出一個以視覺化劇情為重用單位，利用套用機制，加速開發視覺化程式的概念，將視覺化程式語言的物件重用，提昇抽象程度到劇情重用。

由於視覺化程式語言與程式語言有密不可分的關係，因此我們首先探討了程式語言的發展趨勢與瞭解視覺化程式語言，接著從多方面比對程式語言與視覺化程式語言在內建函式、子函式、類別、共用元件及編寫程式等各方面的差異，再來比較 Design Pattern、UML 和多媒體樣板套用三種重用方式的優缺點，

最後介紹可重用視覺化劇情的概念與精神，並且舉出多互動式多媒體的視覺化劇情編輯機制以及手機人機介面的劇情重用，確認本研究的設計成果可以提昇視覺化程式語言的重用。

因此本論文在協助提升視覺化程式語言的重用方面，主要的貢獻如下：

(1) 視覺化程式開發能以較快的速度完成

運用視覺化劇情重用與套用機制，將素材套用到重用劇情就完成編輯，不需要每個素材都重新編輯屬性並且編輯劇情，對於開發大型視覺化程式，或是大量建立類似的視覺化劇情，在速度的提昇上非常的顯著。

(2) 抽象程度比原有重用機制更高

程式語言重用機制從程式碼重用、函式重用、物件重用到共用元件重用，不斷的在提昇重用機制的抽象程度。視覺化程式語言已經有擁有物件重用的機制，但仍缺乏更高程度的重用，因此提出視覺化劇情重用，更大範圍的重用，也可以讓視覺化程式語言的重用提升到更高的抽象程度。

(3) 重用劇情可以維持程式品質

透過視覺化編輯工具編輯劇情儘管很方便，但是如果大量的重複編輯相同劇情，還是有可能因為粗心犯錯造成程式品質不穩定。透過重用劇情，就算重用次數多，但仍能確保程式品質的穩定。

(4) 內容與品質的提昇

視覺化劇情重用與視覺化編輯環境，讓使用者利用簡單的步驟可以編輯，不需要依賴程式設計師就能完成劇情，也可以節省與程式設計師溝通的時間，因此可以專注在內容與品質的提昇

## 6.2 未來發展方向

最後，我們提出以下可以繼續發展的方向，讓可重用的視覺化劇情可以在其他領域應用：

(1) 發展另一個方向的重用

視覺化程式是由重用元件與劇情構成，本文已經針對視覺化劇情的重用提出探討，將重用元件用套用的方式更新。而另外一個方向的重用，代表的就是將多個重用元件結合，套用上不同的劇情，就可以讓相同的元件表現出不同的程式結果。

(2) 將需求利用視覺化劇情表現

目前軟體開發的需求表現，是利用文字或是圖形動畫來表現，最終還是需要程式設計師理解後實做程式，才能夠將需求真正轉換成為程式，在這個過程中就有可能產生錯誤。若是提供足夠的可重用視覺化劇情，利用視覺化劇情來編輯需求，等需求編輯完成，經過轉換後就可以產生程式，減少可能因為軟體工程師理解錯誤產生的誤差。

(3) 程式邏輯訓練工具

透過本研究討論的可重用視覺化劇情，將程式的基本邏輯用劇情表達，在教學時利用訓練工具輔助，讓學生對於邏輯概念更為具體，增進學習的成果。

# 參考文獻


[1] Lafore, Robert., Object-oriented Programming in C++, Indianapolis, Ind Pearson Education, Inc., 2002.

[2] Chorng-Shiuh Koong, “A Component-based Visual Scenario Construction Environment for Non-Programming User to Create Interactive Electronic Books”, PHD Thesis, N.C.T.U. Taiwan, October, 2000.

[3] Low, G., Huan, S, “Impact of object oriented development on software quality”, Software Technology and Engineering Practice, 1999

[4] Jean E. Sammet, David Hemmendinger, “Programming Languages”, Addison-Wesley, 2003

[5] Donald E. Knuth, Art of Computer Programming, Addison-Wesley, 2005

[6] Miao Huaikou, Yu Chuanjiang, Li Li, “A Formalized Abstract Component Object Model - Z-COM”, Technology of Object-Oriented Languages and Systems, 2000

[7] N. C. Shu, “Visual programming: Perspectives and approaches”, IBM Systems Journal, Volume 28 Issue 4, 1989

[8] Shi-Kuo Chang , “Visual languages: a tutorial and survey”, IBM Systems Journal, 1999

[9] Stephen P Levitt, “C++: an evolving language”, 7th AFRICON Conference in Africa, 2004

[10] Shu-Ying Chiang, “The Design and Implementation of Multimedia Test Question Template System Based on an Enhanced Visual Scenario Authoring Tool”, Master Thesis, N.C.T.U. Taiwan, October, 2005.

[11] Visual programming language, “Wikipedia, the free encyclopedia”, http://en.wikipedia.org/wiki/Visual_programming_language

[12] P.J. Plauger, The Standard C Library, Englewood Cliffs, N.J. : Prentice Hall, 1992

[13] Nicolai M. Josuttis , The C++ Standard Library: A Tutorial and Reference, Addison-Wesley, 1999

[14] Klump, R., “Understanding object-oriented programming concepts”, Power Engineering Society Summer Meeting, 2001. IEEE, 2001

[15] Chang, S.K, et al., ”The future of visual languages”, Visual Languages, 1999. Proceedings. 1999 IEEE Symposium on, 1999

[16] Gorlick, M., Quilici, A., “Visual programming-in-the-large versus visual programming-in-the-small”, Visual Languages, 1994. Proceedings., IEEE Symposium on, 1994

[17] Selby, R.W., “Enabling reuse-based software development of large-scale

systems", Software Engineering, IEEE Transactions on, 2005

[18] Lasater, Christopher G., Design Patterns, Plano, Tex Wordware Publishing, 2007

[19] Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995

[17] Sinan Si Alhir, Learning UML, O'Reilly Media, 2003

[18] Llorens, J., Amescua, A., Velasco, M., Software Thesaurus: a tool for reusing software objects , Assessment of Software Tools, 1996

[19] Guercio, A., Shi-Kuo Chang, "A Tree Systems Inference Algorithm for an Iconic Environment", IBM Workshop, 1998

[20] Kang Zhang, Da-Qian Zhang, Yi Deng, "A visual approach to XML document design and transformation", Proceedings IEEE Symposia on, 2001

[21] Zhang, K., Zhang, D.-Q., Cao, J., "Design, construction, and application of a generic visual language generation environment", IEEE Transaction on, 2001

[22] Yacoub, S.M., Xue, H., Ammar, H.H., "Automating the development of pattern-oriented designs for application specific software systems", Proceedings IEEE Symposia, 2000