第2章 資訊系統開發模式

2.1 導論

一、軟體開發模式

這是描述軟體開發一系列的步驟或階段，可分為生命週期模式（Life Cycle Models）和程序模式（Process Models）。

(一)生命週期模式將軟體開發的階段及各階段的關係以概念性的模式表示，隱含開發程序的時間順序。例如瀑布模式、雛型法、漸進模式、演化雛型模式、螺旋模式和同步模式，每一種方法代表一種系統開發的策略。生命週期模式的最終結果是軟體系統。

(二)程序模式則是為了某一特定目的而設計一系列的活動。例如軟體開發程序、品質改善程序、專案管理程序。程序模式的最終結果則是某一管理目標，其實生命週期模式也可視為程序模式的一種，當程序模式所指為開發程序時，兩者所表示的是相同的概念。

二、軟體專案依循某一開發模式的優點

(一)統一的名詞及概念有助於溝通、規劃及管理。

(二)有利於標準、規範與政策的建立及推行。

(三)可提供評估、檢核及里程碑的參考時點。

(四)簡要描繪重要的功能、活動及特性。

(五)開發過程更結構化、更容易管理。

(六)提供一個不斷改善的基礎。

2.2 瀑布模式（waterfall）

◎、概念

(一)瀑布模式把系統開發的過程分成明確的循序執行階段，清楚定義工作及交付文件。各階段產出均須驗證或確認，更改、錯誤或爭議必須回溯修正。

(二)當問題較小或較單純時，可劃分成僅有分析、設計、實施等三個階段;若面對較大或較複雜之問題時，階段可再被細分。

表2.1 大略與詳細之資訊系統開發階段

分　析	設　計	實　施
1. 可行性分析 2. 需求分析 3. 系統分析	4. 概念性設計 5. 細部設計	6. 程式編輯與單元測試 7. 整合測試 8. 安裝與系統測試 9. 教育訓練 10. 操作與維護

(三)瀑布模式的管理意義：它鼓勵依生命週期階段來進行規劃，各階段的產出都必須經過確認或驗證。確認是用來檢驗產出是否符合真實世界顧客的需求。驗證是檢驗系統是否依規格正確地執行。如此開發程序變得更結構化且更容易管理。

(四)瀑布模式也提供兩個主要的加強項目：

1. 若在各階段發現錯誤，可允許階段間之回饋，如此能儘早修正以減少系統修改或重做之成本。
2. 各階段明確定義應做之工作及須交付之文件，使系統開發之工作更明確及容易掌握。

(五)瀑布模式的問題：

1. 在專案開始時，需求須完全且清楚地描述。
2. 所有需求在各階段均需同時考量，且系統開發須在一個週期

內完成。

3. 在程式編輯前過於強調完整的分析與設計文件，故一旦需求變更，文件將需大幅修改。
2. 系統開發週期較長且過程中使用者參與不足。
5. 程式編輯於系統開發週期較後階段才開始，故風險較高，且失敗之成本亦高。

2.3 雛形模式（Prototyping model）

一、基本概念

(一)雛型模式先針對使用者需求較清楚的部分或資訊人員較能掌握之部分，依分析、設計與實施等步驟快速開發雛型。開發過程中，強調儘早以雛型作為使用者與資訊人員需求溝通與學習之工具，雙方透過雛型之操作與回饋，釐清、修改及擴充需求，並藉以修改與擴充雛型。上述步驟反覆進行，直到系統符合雙方約定為止。

(二)雛型法主要是基於需求變更無可避免的構想而設計的。一個可看得到、可操作的雛型是開發者與使用者溝通的良好工具。雛型的建造及修改應該要非常快速，以因應顧客的要求。提高使用者參與的意願，進而提高顧客滿意度。

表2.2 雛型模式之系統開發程序及參與人員

開發程序	主要參與人員
需求擷取／分析	雙方
雛型開發	資訊人員
操作與檢討雛型／需求	雙方
評估是否符合雙方約定	雙方

(三)主要特性與原則：

1. 強調雛型快速開發及使用者高度參與。
2. 強調以雛型作為使用者及系統開發者之需求溝通與學習機制。
3. 從需求最清楚的部分著手開發雛型，並透過使用者對雛型之操作與回饋，反覆修改與擴充，每次反覆時間間隔（週期）要盡可能縮短。

(四)潛在問題：

1. 系統文件較不完備，程式亦可能較難維護。短期而言，可能較能滿足使用者需求；但對長期而言，系統較易失敗。
2. 因缺乏整體之規劃、分析與設計，故較不適用於大型及多人參與之系統開發專案。

(五)常見應用策略：

1. 演進式雛型策略（Evolutionary prototyping）：演進式雛型策略主要係將所有需求看成一個整體，從需求最清楚的部分先快速經歷一系統開發週期，以完成初版雛型系統開發。再利用該雛型與使用者溝通，以確定、修改和擴充需求，並藉以作為下一週期雛型演進之依據。該週期不斷地反覆進行，直到雛型系統符合雙方約定為止。
2. 快速用後丟棄式雛型策略（Rapid throwaway prototyping）：是以一種快而粗糙的方式建立雛型，讓使用者能夠盡快藉由與雛型互動來決定需求項目，或允許資訊人員藉以研發問題之解決方法與資訊科技之應用等。這種雛型因為用後即丟，所以不需要考慮雛型系統之運用效率與可維護性，也不需要考慮容錯的能力。快速用後丟棄式雛型策略若用於具高困難度之技術或設計的專案，可以藉由快速的雛型開發與檢討，探索出問題之解決方法或資訊科技應用的可行性。由於成本較高，快速用後丟棄雛型策略通常僅在風險程度最高的地方實施，而其他情況則盡可能地採用演進式雛型策略。