生成式AI輔助之軟體開發指引

軟體技術研究院
Software Technology Institute

中華民國112年11月

目錄

前言

生成式AI已引起全球廣泛的關注和想像力，它有助於企業改善客戶體驗、提高員工生產力、優化業務效率和促進內容製作創造力。展望未來，生成式AI在多個領域有廣泛的應用前景，包括軟體開發、行銷業務、金融服務、零售服務、健康醫療、生產製造等。

其中，特別在軟體開發領域，資策會相信是生成式AI最具前景與影響力的應用之一，也是企業數位轉型必須掌握的一個關鍵要素。

生成式AI已能夠在軟體開發生命週期的各個階段提供協助，甚至取代部分人力工作。2023年初，Google測試ChatGPT之程式撰寫能力已達到三級工程師（初階工程師）的水準。也因此，許多軟體開發者對生成式AI發展感到擔憂，擔心未來工作職可能會減少或被取代的風險。然而，若能積極掌握生成式AI的能力，善用生成式AI的輔助，將會大幅提升軟體開發效率，為軟體開發者增強核心競爭力，帶來更多發展機會。

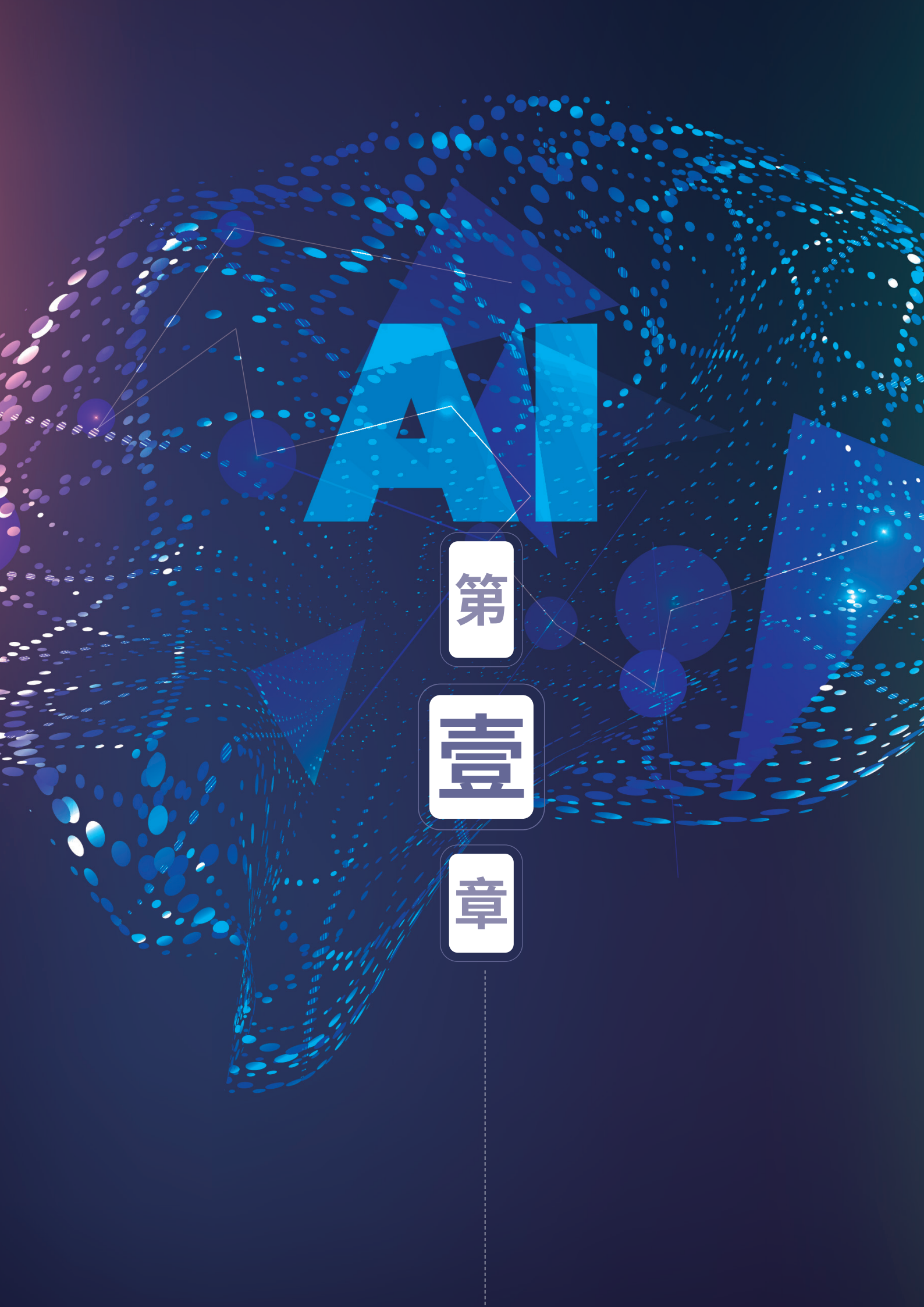
以生成式AI方式開發軟體，為企業帶來效益的同時，亦衍生一些新的風險問題及治理挑戰。企業應當如何建立負責任的軟體開發機

制，是現今軟體開發至關重要的議題。因應於此，本指引提出六大關鍵面向，從問責性、系統透明度、AI生成項目、資料治理、系統公平性、系統可靠度，協助發展建立負責任的軟體開發機制。

本指引定位在探討生成式AI方式如何輔助軟體開發，在軟體開發方法論方面，我們採用資策會新一代軟體開發方法「RAISE」：需求挖掘（Requirement）、敏捷開發（Agile）、整合驗測（Integrated）、軟體安全（Secure）、環境建構（Environment），再加上軟體開發完成上線營運後之維運監控階段（Operations），總共六大關鍵面向。本指引就生成式AI如何在這六大面向輔助軟體開發，對其各細項目標及執行原則進行概要性的闡述。

本指引彙整資策會軟體院團隊實際運用生成式AI軟體開發實踐經驗為基礎，並參考與融合國內外相關文獻與資料（詳見附錄），經過資策會各院所及產學專家之悉心指導與協助下，精心編撰共同完成。

然而，鑑於產業及AI技術快速演進，我們深知要撰寫一本面面俱到的軟體開發指引實屬艱鉅，因此，我們先提出生成式AI輔助軟體開發指引初版，為與時俱進，以拋磚引玉的心態誠摯邀請各界提供寶貴反饋意見，資策會團隊秉持敏捷精神，將持續不斷地更新與完善內容，為產業發展貢獻心力。



第壹章

生成式AI對於軟體開發之影響

生成式AI之應用的類別隨著技術發展推進逐漸廣泛，根據市場應用成熟順序大致包括下列4類型：

類型1 文本：生成式AI可生成自然語言之文本，如：文本摘要、對話系統、自動翻譯和寫作助手等。

類型2 程式碼：生成式AI可透過自然語言描述生成程式碼（Code），根據已知的資料和模型生成新的程式碼，如：Python、Java、C++、JavaScript等。

類型3 圖像：生成式AI可生成擬真的圖像，如：圖像修復、風格轉換、圖像合成和生成、圖像標註和辨識等。

類型4 影片/3D/遊戲：生成式AI可生成擬真的影片（Video）、3D物件內容、及遊戲。如：影片修復、風格轉換、影片合成和模擬、影片標註和辨識；根據主題描述或2D圖像，並由已知的資料和模型生成新的3D物件；遊戲內容和關卡，如：電子遊戲、桌遊等。

在軟體程式開發領域中，使用生成式AI可以帶來許多優勢和好處，包括：提高開發效率、產生創新解決方案、解決開發挑戰等，分別說明如下：

1. 加速軟體開發過程

傳統上，程式開發是一項複雜而費時的任務，需要開發人員具備深厚的專業知識和經驗。然而，生成式AI方法可以通過模型的學習和生成能力，自動生成需求分析、程式碼、測試模組等各項目，從而大大縮短開發時間。開發人員可以利用這些自動生成項目作為基礎，進一步修改和優化，以符合實際的需求。這種方式不僅節省了大量的時間和精力，還提高了開發效率，使開發人員能夠更快速地交付產品和服務。

2. 產生更加創新和多樣化的程式解決方案

生成式AI模型通常通過學習龐大的資料模式和結構，從而獲得對程式碼的深入理解。這使得模型能夠生成符合特定需求的程式碼，同時保持一致性和可讀性。生成式AI方法可以突破傳統的程式設計思維，提供新穎的解決方案，並激發開發人員的創造力。這對於開發複雜系統、處理大量數據和應對快速變化的需求非常有價值。

3. 幫助開發人員克服各種困難和挑戰

有時候開發人員可能面臨著複雜的問題，需要在有限的時間內找到有效的解決方案。生成式AI方法可以通過提供需求功能規劃、程式碼示例、測試案例自動完成、錯誤修復等功能，為開發人員提供寶貴的輔助工具。這不僅可以節省時間和精力，還可以減少錯誤和疏漏，提高程式的穩定性和可維護性。

儘管生成式AI方法在程式開發中具有許多優勢，但也存在一些挑戰和限制。首先，生成式AI模型可能產生不完整或不正確的程式碼，需要開發

人員進一步修正和調整，因此，開發人員仍然需要具備相關的專業知識和技能，並且對生成的程式碼進行仔細的檢查和驗證。此外，生成式AI可能存在隱私風險和潛在的安全問題，例如：生成不安全的程式碼或違反隱私保護的內容。因此，在使用生成式AI時，開發人員需要謹慎評估和監控模型的性能和結果，建立負責任的開發機制，以確保生成的項目符合需求並且安全可靠。

第一節 軟體開發生命週期與軟體工程方法之演進

軟體開發生命週期（Software Development Life Cycle，簡稱SDLC）是指軟體開發過程中的各個階段，從需求分析到系統部署。這些階段有著特定的目標和活動，旨在確保軟體開發順利進行並產出高品質的軟體。以下是軟體開發生命週期的各個階段說明：

圖 1-1：軟體開發生命週期說明

1. 需求

在這個階段，軟體開發團隊與客戶合作，收集和分析使用者的需求。包括：討論功能、性能要求、系統限制等。需求分析階段的產物是需求規格書或需求文件，用於明確定義系統的目標和範圍。

2. 設計

在設計階段，軟體工程師根據需求分析階段的需求規格書，設計系統的架構和功能。架構設計關注整體系統結構和各模組之間的關係，而功能設計則更具體地定義各模組的細部說明。

3. 開發

在這階段，開發團隊根據設計階段的規範和需求開始編寫程式碼。開發團隊通常會使用統一的程式碼編寫規範和測試方法來確保程式碼的品質。

4. 測試

在這個階段，軟體測試人員使用測試計畫和測試案例來驗證系統的功能和品質。測試可以包括：單元測試、整合測試、系統測試和驗收測試等不同層次和類型的測試。測試的目的是發現和修復潛在的錯誤，確保軟體符合預期的需求。

5. 部署

在軟體開發的部署階段，已經測試和驗證過的軟體版本被部署到目標環境中。這可能涉及安裝、設置和配置軟體，並將數據轉移或整合到新系統中。部署過程應該是受控且可重複的，以確保軟體在生產環境中的正確運行。

「軟體工程方法」是一個包括：軟體開發生命週期的學科和實踐領域，其目標是以系統性和結構性的方法來設計、開發、測試和維護高品質的軟體系統。軟體工程也強調團隊合作和有效的專案管理，團隊成員通常

按照特定的角色和責任分工，並遵循協作和溝通的最佳實踐。

常用的軟體工程方法包括：「瀑布式」（Waterfall）及「敏捷式」（Agile），此外，隨著生成式AI技術與方法快速演進，新一代的軟體工程方法將是「生成式AI式」，以下將對於此3種軟體工程方法分別描述。

1. 瀑布式（Waterfall）

方法 瀑布式方法是一種序列化的開發方法，依次完成需求分析、設計、開發、測試和部署等階段。每個階段需要完整完成後，才進入下一個階段。

特色 瀑布式方法強調確定性和嚴格的控制，但缺乏快速適應和迭代開發的能力。

適用情境 瀑布式方法適用於需求相對非常穩定且明確的專案。當需求在開發過程中很少發生變更時，瀑布式方法能夠提供結構化的流程和明確的里程碑，使得開發團隊可以按照預先定義的計畫進行開發。

2. 敏捷式（Agile）

方法 敏捷式方法是一種迭代的開發方法，將開發過程拆分為短期的迭代週期（通常是2到4週）。在每個迭代週期中，團隊會完成一個功能子集的開發，進行測試和審查，然後根據反饋進行調整。

特色 敏捷式方法能夠更快地產生可工作的原型，並且可以根據用戶和利益相關者的反饋進行調整，這種方法鼓勵靈活性和自組織團隊彼此配合度，才能有效發揮其成效。

適用情境 敏捷式方法適用於需求可能會變動且需要快速反應的專案。當需求可能在開發過程中不斷演進時，敏捷式方法能夠提供靈活性和迭代開發的方式，讓團隊能夠快速適應變化並優化產品。

3. 生成式AI

方法 生成式AI是結合提示工程（Prompt engineering）與低程式碼/無程式碼（Low Code/No code）之新一代軟體開發方法，軟體開發的各階段（需求、設計、開發、測試、部署）皆可以結合此方法共同完成。

特色 生成式AI是以提示工程為主的開發方法，因此如何產生精準且有效率的提示是其重點，此外生成產生的需求文件、設計架構、程式碼、測試案例等仍需要相關人員進行確認及調整為確保其正確性及品質。

適用情境 生成式AI目前屬初期發展階段，因短期內較適合開發功能較簡易的軟體，但隨著技術快速進步，中期預期將可生成非常複雜的軟體應用。

第二節 提示工程方法

如前所述，生成式AI將對軟體開發生命週期與軟體工程方法造成許多面向的改變，其中因此而衍生出之新興工程方法即為本節所要論述之「提示工程」方法。此「提示工程」將可有效協助使用者於軟體開發過程中之

各個生命週期。

所謂「提示工程」（Prompt Engineering）旨在針對給予生成式AI系統（如：ChatGPT、大型語言模型Large Language Model等）關鍵之文字或詞句作為提示（Prompt），使得生成式AI系統可更準確地回應出對使用者有意義的產出物（內容）。

圖 1-2：提示工程概念

生成式AI系統的產出是否切合使用者的需求，跟使用者所給予提示之文字與詞句直接相關，使用者所使用的提示技巧越熟練，系統所產出的產出物越能切合使用者的需求。以ChatGPT為例，一些「提示工程」常見的提示方式如下，當然使用者可依其不同複雜程度的需求反覆連續地給予提示，進而完成最終的成果。

1. 基礎提示（Basic Prompt）

提示文字或詞句須包含明確的要素：A.動作指令、B.必要資訊和C.產出物樣態。主要概念如下：

圖 1-3：基礎提示概念

A. 動作指令建議是明確的動作要求，例如：「建立」、「產生」、「撰寫」或「解釋」等明確提示，甚至是「避免」、「排除」之類的迴避動作均可；所給予的動作指令越明確，避免模糊，產出物越能切合需求。

B. 必要資訊是給予明確的問題或期望得到的內容，以軟體開發為例，可以是一個系統的需求文件、程式碼或是系統流程等，例如：「需求規格」、「系統架構」、或「測試方式」等。

C. 產出物樣態是給予明確的產出物應呈現的樣態，可以是所需的語言類別，產出物內容、文字限制或格式等，亦或是程式碼；綜上，基礎提示之形式如：「請以300字以內中文撰寫XXX系統的需求規格」或「請以XXX語言撰寫YYY演算法」等等。適用於簡單明確且容易以文字描述的需求。

2. 鏈式提示（Chained Prompt）

鏈式提示是基於基礎提示，針對較為複雜的需求，而與基礎提示最大的差異在於，鏈式提示的技巧為利用前次提示的產出物納入後續提示的內涵中。其概念如下圖：

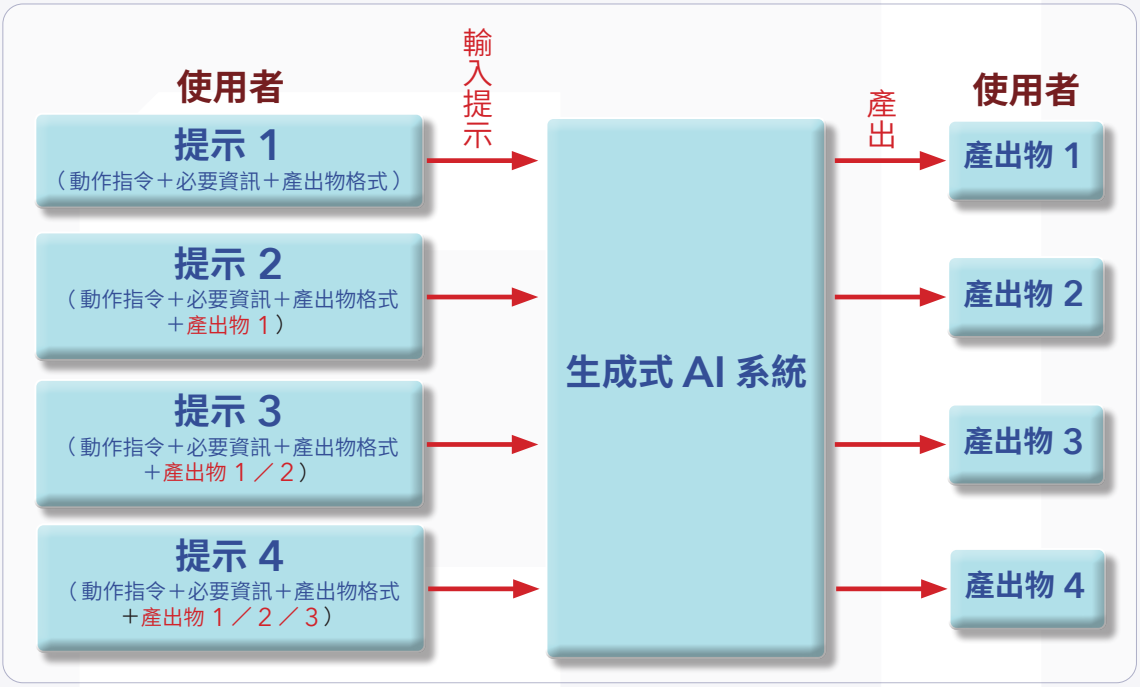


圖 1-4：鏈式提示概念

以軟體開發為例，在利用鏈式提示的技巧時，使用者可以迭代地將部分或全部前次提示的產出物作為下一次提示的內容，持續迭代地將產出物反饋至提示，逐漸導引生成式AI系統針對需求進行分析，亦或是修正前幾次的產出物、增加或微調尚未滿意的產出物等。例如：「請說明產出物1的更細項的內容」、「請說明產出物1與產出物2的關聯性」等等。

3. 思維樹提示（Tree of Thoughts Prompt）

對於軟體開發而言，另一項重要的提示工程為思維樹提示技巧；所謂思維樹為使用者預先將標的系統進行拆解，將大項系統分析為多個子系統、模組等，如下圖，使用者將分析拆解完後的項目，作為給予生成式AI系統的提示依據。

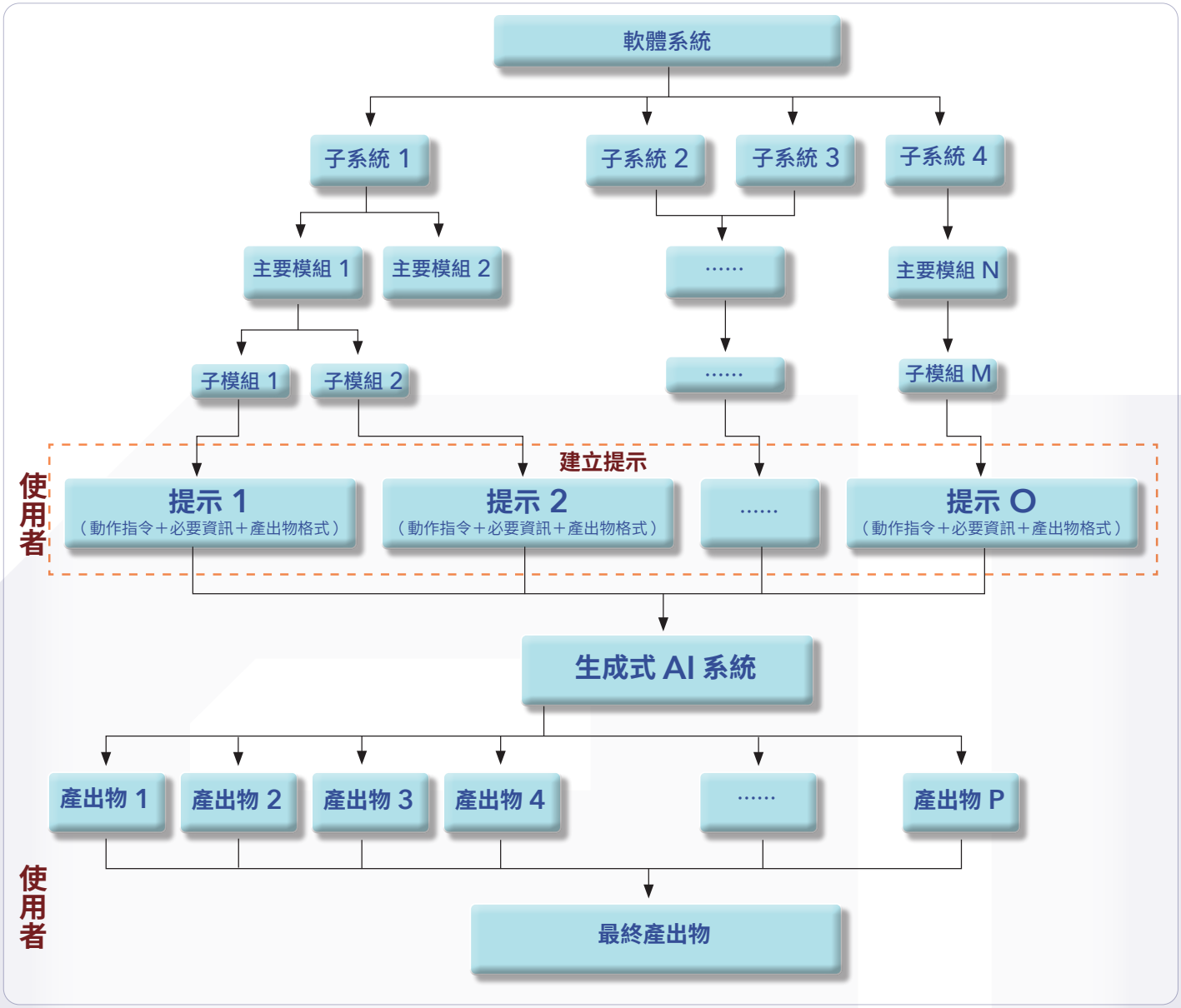
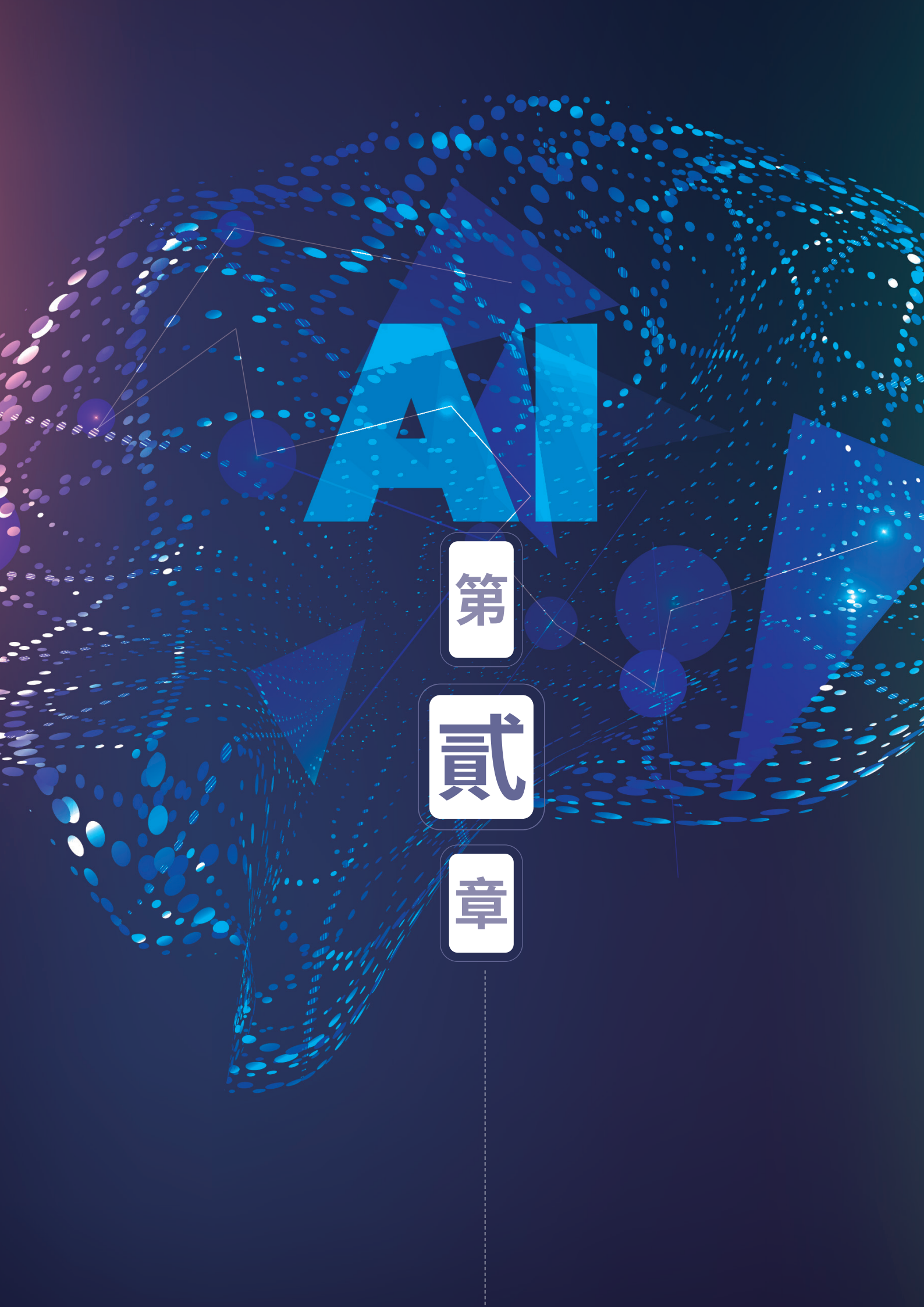


圖 1-5：思維樹提示概念

而系統所產生出之產物，最後再由使用者綜合整理為最終產出物，如圖1-5：於使用者已將系統拆解為多個細部項目後，再根據各細部項目的內涵建立提示資訊，如：需求規格、程式碼、測試案例或程式碼優化等，均可透過此一方式來提升工作效率。

而本思維樹提示技巧中，生成式AI系統的多項產出物，仍須仰賴使用者進行最後的綜整，如：文案上整合、程式碼整合、使用者介面整合或驗證測試的整合等。

上述三項提示工程可根據使用者需求與提示技巧的熟悉度交互使用，方能發揮生成式AI在軟體開發上最大的助益。現階段提示工程方法亦在不斷的演進中，但應用在協助軟體開發的做法上，本指引已建立出可以實踐的基礎，在生成式AI的發展下，未來新的提示工程方法也可納入評估參考，以逐步完善本指引。



第貳章

建立負責任之軟體開發機制

以生成式AI方式開發軟體時，如何制定有原則和可操作的規範以確保組織負責任地開發和部署AI，是一項重要的議題。

第一節 問責性機制建立

一、目標

對於整體系統可能產生之各個重要影響項目，進行評估、審核、與記錄，以提升其可追溯性及可問責性。建議機制如下：

- 在系統開發的初期（通常是在定義產品目標和需求時），確認所需評估及審核的各個重要影響項目。
- 在系統開發階段發布之前，組織內品保人員對於此重要影響項目評估結果進行審核，並且進行批准確認。
- 記錄其評估階段與審核階段各項文件資料，並對相關人員發布。
- 在系統增加新的功能以及進入新的發布階段之前，更新和審查其影響評估。

二、重要影響項目

- 系統透明度。
- AI生成項目。
- 資料治理。
- 系統公平性。
- 系統可靠性。

第二節 系統透明度

一、目標

系統之使用者介面、各項功能、決策產生的方法與邏輯等項目，應盡可能地提供相關文件及記錄，以提供未來系統使用者更完整的理解，包括：

- 使用者介面的直覺性。
- 各項功能的可使用性。
- 系統決策的可理解性。

二、執行原則

- 應盡可能地提供系統使用者介面、功能說明、決策產生的方法與產出等文件，進行分類整理及記錄。
- 與系統使用者相關人員進行溝通，以評估此系統是否能提供使用者

直覺的操作互動、是否能提供使用者理解系統的預期功能、是否能提供使用者理解相關系統的決策回應。

- 如有部分評估結果未能通過審查標準，系統應進行調整其規劃設計，以進行重新評估。

第三節 AI生成項目

一、目標

軟體系統開發各階段（需求、設計、開發、測試、部署），應記錄使用全自動化生成或半自動化的各個項目，以期達成下列目標：

- 提供有效率地專案管理、效率評估、及溯源追蹤。
- 隨著生成式AI技術持續演進，提供未來開發時可持續強化的項目等依據。

二、執行原則

- 記錄使用AI生成方法的所有原始提供單位、原始提供系統、原始提供者之授權規範等說明文件。
- 定義及記錄由AI方法「全自動化」生成的需求模組、設計架構、程式模組、測試模組、測試數據、部署模組等項目。
- 記錄使用提示（Prompt）之內容與過程，重點項目並加以註解說明。
- 定義及記錄由AI方法「半自動化」生成的需求模組、設計架構、程式模組、測試模組、測試數據、部署模組等項目，並且記錄每項由

人工調整之原因、方式、成果等說明。

■ 定義及記錄由AI方法生成的圖像、音訊、視訊、或其他等產出之說明文件，並且確保任何打算在系統外部使用的圖像、音訊、視訊等產出都已經被「標記」為由 AI 生成。

第四節 資料治理

一、目標

軟體開發階段，應記錄使用AI生成與提示所需之各類型資料種類、來源、品質及使用授權等說明，以期達成下列目標：

■ 建立所用資料之可追溯性及可問責性。

■ 提供軟體後續改版精進之依據。

二、執行原則

■ 定義並記錄資料來源，及使用授權等說明。

■ 定義並記錄資料收集和處理的程序，包括：相關的註解、標記、清理、多元融合。

■ 如果預計使用這些資料集來訓練系統或提示，請根據上述中定義的資料要求評估系統所需的可用資料集的數量和適用性。在影響評估中記錄此評估。

■ 定義並記錄評估系統使用資料的方法。

■ 依據上列方法評估所有需要的資料集。

第五節
系統公平性

一、目標

系統預期功能及系統預計部署，應盡可能地減少對於不同地理區域或不同人口群體（包括：邊緣化群體），系統可能產生之功能差異、品質差異、偏見、或詆毀。這些差異可能是由系統開發者或生成式AI所產生，包括：

- 減少對於不同地理區域或不同人口群體（包括：邊緣化群體），其提供功能的差異。
- 減少對於不同地理區域或不同人口群體（包括：邊緣化群體），其服務品質的差異。
- 減少對於不同地理區域或不同人口群體（包括：邊緣化群體）之偏見或詆毀。
- 對於系統可能產生之上述差異提供相關文件，說明其差異評估以及任何可解釋這些差異的合理因素。

二、執行原則

- 識別及排序可能面臨績效下降（包括：提供功能減少、服務品質下降、偏見或詆毀）的風險的地理區域或人口群體（包括：邊緣化群體）。
- 蒐集所有數據，定義其各項績效評估方案中，應包括之地理區域或人口群體的個別模組、評估模組、和整個系統的指標。

- 對於上述已識別之地理區域或人口群體定義並記錄各項績效評估指標結果，對於每個指標，記錄其（i）所有達到最低績效水準的所有群體；（ii）達到最高績效水準的群體；（iii）最低績效水準與最高績效水準的差距。
- 上列指標評估結果超過可接受範圍時，重新評估系統設計，包括：訓練數據、特徵、目標函數、及訓練演算法的選擇，以期達成（i）提高任何未達到最低績效水準的已確定地理區域或人口群體的績效；（ii）減少已識別的地理區域或人口群體之間的績效水準差異；（iii）最大程度地減少已確定地理區域或人口群體之偏見或詆毀。
- 識別並記錄任何合理「無法滿足任何的最低績效水準」的因素，例如：環境因素和其他操作因素（例如：語音識別系統在「吵雜背景噪聲環境」下的績效水準下降）。
- 記錄其評估結果，並且發布以下資訊：
  - ► 確定績效可能達不到某項最低績效水準的地理區域或人口群體。
  - ► 任何解釋這些績效水準差異的合理因素。

第六節 系統可靠性

一、目標

系統預期功能及系統預計部署下，應記錄結合AI開發方法後整體系統的可靠度指標，以及發生錯誤時之補救、記錄、及後續監控評估方法。其目標包括：

- 定義、評估、及記錄此「系統性能」可接受的錯誤率是多少。
- 定義及說明發生錯誤時之補救措施，以減少對使用者的不利影響及抱怨。
- 建立系統健康監測方法，並記錄系統實際運作情況，以掌握系統可靠度及做為後續精進之依據。

二、執行原則

- 確定並記錄包括系統輸入、使用、和操作環境等品質參數。
- 定義系統可靠性評估計畫，此評估計畫需包括評估系統的環境。
- 定義並記錄每個操作因素的可接受範圍，定義並記錄系統在這些範圍內運行時可接受的錯誤率。
- 定義可預測的錯誤，包括整個系統的誤報和誤報結果，以及它們將如何影響每個預期功能。
- 建立系統健康監測的項目，包括：
  - ► 系統產生的數據記錄。
  - ► 使用者可以提供有關錯誤和疑慮資訊的程序。
  - ► 使用者可以提供回饋建議的程序。
- 建立系統健康監控行動計畫，包括那些項目將受到監控、檢視監測項目的頻率、使用者回饋記錄檢視、那些項目事件會進行優先回應和解決等。

第參章

實踐生成式AI輔助之軟體開發方法

生成式AI輔助之新一代軟體開發方法，包括：需求挖掘（Requirement）、敏捷開發（Agile）、整合驗測（Integrated）、軟體安全（Secure）、環境建構（Environment）共五個面向，取其各項英文名稱第一個字母後簡稱為「RAISE軟體開發方法」，對應到軟體開發生命週期之五個步驟如下圖所示，其各面向說明如下：

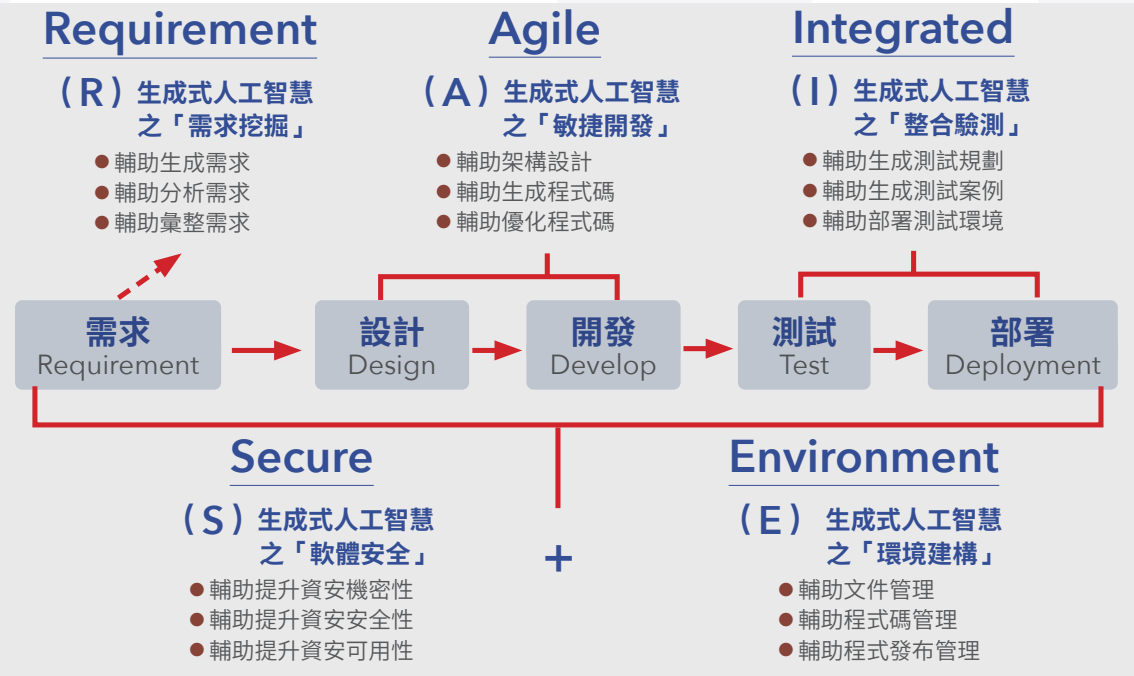


圖 3-1：生成式AI輔助之新一代軟體開發方法

1.（R）需求挖掘

在軟體開發生命週期之「需求」階段，可以使用生成式AI輔助生成需求、輔助分析需求、及輔助彙整需求等。其各細項目標及執行原則，將在本章第一節描述。

2.（A）敏捷開發

在軟體開發生命週期之「設計」及「開發」階段，可以使用生成式AI輔助生成軟體架構、輔助生成程式碼、及輔助優化程式碼等。其各細項目標及執行原則，將在本章第二節描述。

3.（I）整合驗測

在軟體開發生命週期之「測試」及「部署」階段，可以使用生成式AI輔助生成測試規劃、輔助生成測試案例、及輔助部署測試環境等。其各細項目標及執行原則，將在本章第三節描述。

4.（S）軟體安全

在軟體開發生命週期之全階段（需求+設計+開發+測試+部署），可以使用生成式AI輔助提升資安機密性、完整性、可用性等。其各細項目標及執行原則，將在本章第四節描述。

5.（E）環境建構

在軟體開發生命週期之全階段（需求+設計+開發+測試+部署），可以使用生成式AI輔助文件管理、輔助程式碼管理、輔助程式發布管理等。其各細項目標及執行原則，將在本章第五節描述。

此外，軟體開發完成正式上線營運後之維運監控，「維運監控階段」(Operations)，透過生成式AI亦可以輔助完善網路設備管理、提升網路資安事件處理時效、提升軟體資訊安全、以及強化效能監控能力。其各細項目標及執行原則，將在本章第六節描述。

第一節 需求挖掘

軟體開發生命週期的第一段階段「需求」，乃係針對預計所需發展之軟體系統進行系統分析後釐清所要發展之各項需求，包含各項功能、使用者介面、資料、甚至包含資訊安全管理以及系統發布後的運轉監控等；傳統之需求挖掘方式大多基於與客戶的訪談，並根據規劃者的經驗逐步逐項地完成各式需求。

本階段的目標包括R1：需求內容引出、R2：需求概念塑模、R3：需求規格彙整。透過這些目標的指引，說明如何利用前述之提示工程方法，提升需求挖掘的效率，加速需求挖掘的進程。

一、目標R1：需求內容引出

目的：以生成式AI協助取得需求，透過提示工程方法，挖掘出企業所要的需求，藉以展開對所需之軟體建構的程序。

執行原則：

- ▶ 反覆引導：提示中所給予的資訊越詳細，所得到的生成資訊越有價值，反之，越是不明確的資訊，所得到的資訊可能越不精準。

- 結構化引導：對於目標軟體系統了解深入的分析者而言，可以鏈式提示與思維樹提示之方式混合應用。

二、目標R2：需求概念塑模

目的： 以生成式AI協助進行需求概念塑模，透過提示工程方法，將需求以使用案例形式描述出企業所要的需求，藉以描繪出需求基礎。

執行原則：

- 需求建議：透過提示工程告知AI系統使用場景與情境，由AI建議可能的功能需求、非功能需求，以及運轉監控需求，以協助後續需求分析。
- 塑模案例：由所引出需求之內容進行分析，並將需求抽象化為使用案例（Use Case），並加入參與者角色進行塑模，產出使用案例模型（Use Case Model）。
- 描述案例：就所引塑模出的使用案例進行內容的描述定義，包含使用案例名稱、參與者、使用案例簡述、前置狀態、基本流程、選擇性流程、事後狀態之內容。

三、目標R3：需求規格彙整

目的： 以生成式AI協助進行需求規格彙整，透過提示工程方法，將描述需求之內容以律定之規格描述，藉以建立需求基準。

執行原則：

- 彙整製作：由需求概念塑模所產出之結果，整理出客戶的需求、期望、限制及介面，記錄並產製於需求文件當中。

- 分析驗證：檢視與分析需求，檢驗是否滿足客戶業務需要與目標、是否滿足時效性功能的順序與關鍵人員需求，並進行評估需求與限制是否達成平衡。
- 建立規格：系統分析者對於生成式AI所產生之內容需再加以統合整理，最終產出軟體需求規格。

第二節 敏捷開發

軟體開發生命週期的第二階段「設計」及第三階段「開發」，乃係實現需求所發展之軟體系統進行系統設計與程式開發。

在敏捷開發中，以生成式AI協助軟體設計與開發，可以運用在設計模型架構、開發程式碼、開發測試、改良測試、開發文件撰寫等，還可以協助語言轉換、函式庫查詢、程式碼理解、開發文件撰寫等功能，由使用者來判斷在開發過程中遭遇甚麼問題，並以指引的方式來誘導生成式AI產出成果。使用者必須辨識並確認最終產出是否合乎使用，並由不斷互動指引來修改最終產出，以此實現系統設計與開發。

本階段的目標分為A1：需求分析與架構設計優化、A2：資料庫設計、A3：程式碼開發、A4：模組測試與除錯、A5：文件化與程式碼品質、以及A6：程式碼重構。透過這些目標的指引，說明如何利用前述之提示工程方法，提升設計與開發效率的方法。

一、目標A1：系統分析與架構設計優化

目的：以生成式AI協助系統設計及開發程式，透過提示工程方法來引導從需求歸納出的領域模型（Domain Model）轉為系統設計模型（Design Model），並且在系統架構模組設計、模組介接設計過程中，提供協助優化，於增加後續的說明來取得設計與方向調整，以便逐步逼近成果產出。

執行原則：

- 生成式AI工具的應用：使用生成式AI工具可以規劃需求、分析領域模型和使用情境，特別是針對非功能需求如安全性、性能和可靠性，以及上述非功能需求的監控與預警方式，以建立完整的需求範圍和了解。
- 系統架構優化：在開發前期，透過生成式AI描述系統的方法、工具、環境等資訊，建議合適的系統架構，包括：前後端技術、雲端／移動裝置限制、資料庫選擇、系統效能監控架構與做法等，以優化現有的系統架構。
- 使用AI建議與人為審查：應該將AI生成的規劃內容進行人為審查，確保其合理性並進行必要的修改。同時，可使用Markdown或Mermaid繪製關係圖、UML等來協助開發，幫助釐清模組之間的關聯，並遵守單職責原則（Single-responsibility principle, SRP）。
- 對話內容從大題目拆成多個小題目，並減少小題目之間的耦合性。
- 設計模型應遵循單職責原則。

二、目標A2：資料庫設計

目的： 在資料庫設計上，針對關聯式資料庫部分，透過使用生成式AI的幫助，可以加快基本的資料表結構的設計。以協助敏捷設計過程中的新增與修改。透過不斷的互動和指引，我們能夠修改先前的資料表結構結果，以實現系統符合當前團隊的資料庫結構。

執行原則：

- ▶ 應確保生成的內容符合用戶或團隊的需求：透過生成式AI產生數據表和SQL語法，包括：資料表的欄位和約束條件、數據完整性、一致性以及結構化的方式來創建和組織數據表。
- ▶ 敏捷開發與持續反饋：使用生成式AI進行快速原型開發，可指示生成SQL語法以滿足檢索條件。然後測試並確保查詢語法滿足業務需求，持續在開發過程中進行修改和反饋，確保對話上下文連續性。
- ▶ 不完全依賴AI並查證生成內容：生成式AI為輔助工具，應謹慎使用並查證其生成的內容。持續反饋、驗證和測試是必要的，以改進生成的查詢，確保滿足不斷變化的需求和優化目標。
- ▶ 建立資料庫效能相關KPI，以提供O4目標所需的KPI輸入。

三、目標A3：程式碼開發

目的： 當系統設計模型已建立，實現該系統時，利用生成式AI協助建構程式碼層級的細節，包括：程式碼命名、資料結構、控制結構、變數、可見性以及修飾詞（例如：static、abstract）。

執行原則：

- 基於類別圖指引程式碼：從生成式AI中獲得類別名稱、方法和屬性，但需人工檢查並補充互動關聯，確保程式碼能夠完整運作。
- 命名和註解重要性：對話式生成式AI的命名需要特別留意語意，並加上可理解的註解，幫助模型和開發者閱讀和理解程式碼。完成的程式碼也可要求AI產生註解或文件化，協助維護和開發工作。
- 利用生成式AI產生測試資料：可以使用生成式AI生成測試資料，幫助驗證後續處理程式是否能正確處理資料。
- 使用AI檢測程式碼中隱含的系統效能問題，並提供建議。

四、目標A4：模組測試與除錯

目的：在修改程式功能或偵錯時，需要開發人員對測試結果回覆的例外訊息做判斷，並簡單判斷可能導致錯誤的模組片段，作為目前狀態，並與設計目標對生成式AI做進一步的指引詢問，以取得更新程式碼或修改方式。

執行原則：

- 人工輸入改善效果：手動找到錯誤模組並輸入提示對話以獲得更好的改善結果，因為生成式AI的記憶力有限。
- 整合開發工具運用生成式AI的限制：整合開發工具使用生成式AI在除錯方面較薄弱，生成結果受使用者的設計和命名語意影響。因此，應特別注意程式命名、參數及註解語意之正確性。
- 對話型生成式AI的限制：基於對話的生成式AI因版本和記憶限制，可能無法解決邏輯性或整合性問題，需要抽象引導或自行判斷

處理整合型問題。

五、目標A5：文件化與程式碼品質

目的：產生程式碼後，一致的撰寫風格，以及提供對應的註解，能夠幫助團隊在開發的過程中能夠互相支援，以及後續的維護。在程式撰寫的過程中，可以使用生成式AI來協助檢查程式碼是否有不符合團隊程式風格的部分，以及對於函式與方法做說明註解。

執行原則：

- 保持一致的程式碼風格：確保團隊和生成式AI在程式碼撰寫風格上保持一致，進行程式碼檢視（Code Review）以防止成果凌亂。
- 增加註解和解釋：對話式生成式AI產生的程式通常會有註解或解釋，建議在每次引導生成時，將生成目的寫入程式註解，以增加程式碼的可讀性和理解性。

六、目標A6：程式碼重構

目的：程式碼重構（code refactoring）為軟體系統的重要階段之一，尤其是需要長久維護的系統，當生成式AI加入後更需要時常重構程式碼，由於生成式AI存在一定程度的變異，配合開發團隊完成階段性程式版本的過程中，將程式編碼風格或整合後能產生進一步優化的結構，使程式碼更簡潔、更具可讀性、一致性，並且更易於測試。因此配合模組設計藍圖與生成式AI對模組的判別，可以協助部分的程式碼重構。

執行原則：

- ▶ 生成式AI的限制：對話式生成式AI在給定程式範圍時能提供重構建議，但在較大範圍或複雜系統的控制方面可能較為薄弱，使用上可先以小範圍為主，最後再予組合。
- ▶ 團隊共識和重構準則：使用生成式AI進行重構需要團隊有共同的重構標準，保存參考文件確定重構方向正確。重構應該在重大開發斷點時進行，並需要團隊和審查人員確認所有更動是正確且可行的。

第三節 整合驗測

軟體開發生命週期的第四階段「測試」及第五階段「部署」，為系統完成開發後所進行系統成品的驗證與上線準備。

在整合驗測當中，包含整合測試及部署，以生成式AI協助系統測試與系統部署，可以運用在測試階段的測試計畫書規劃、產生測試資料、測試案例、自動化測試腳本及彙總測試報告之撰寫；以及運用在部署階段的上線文件生成及部署作業、上線系統測試和驗證與監控及管理系統。

指引的方式來誘導生成式AI產出成果，使用者必須辨識並確認最終產出是否合乎使用，並由不斷互動指引來修改最終產出，以此實現整合測試及部署。

本階段的目標分為I1：測試規劃、I2：單元測試、I3：整合測試、I4：系統測試，以及I5：部署運行。透過這些目標的指引，說明如何利用

前述之提示工程方法，提升測試與部署的效率。

一、目標 I 1：測試規劃

目的：以生成式AI協助系統測試，透過提示工程方法來引導測試負責人根據需求規格書、設計規格書的內容撰寫其測試計畫書（包含測試案例），依此分配測試案例提供測試人員進行自動化測試腳本之撰寫，以完善系統之驗證。

執行原則：

- 測試計畫書之撰寫：測試計畫書基本的架構包含系統簡介、測試目標、測試軟硬體環境及工具需求、測試案例等。
- 找出潛在弱點測試：軟體測試的規劃應該從需求過程的早期階段開始，並且隨著軟體開發的進行，測試計畫和程序應該系統化地及持續地開發，並且被改進。這些測試計畫和測試設計活動為軟體設計人員提供了有用的輸入，並有助於發現潛在的弱點，例如：設計疏忽/矛盾，或文件中的遺漏/含糊不清。
- 使用生成式AI產出效能相關的測試情境與測試個案並於系統開發階段的單元/整合/系統測試與運轉維護階段使用，以提高系統效能監控能力。

二、目標 I 2：單元測試

目的：單元測試驗證可單獨測試的軟體單元的功能。以生成式AI協助單元測試時，透過提示工程方法根據上下文來引導單元測試之測試個案及測試程式生成。

執行原則：

- 生成單元測試個案設計：透過思考正向、反向以及例外三種情境來引導生成式AI協助產生單元測試案例，包括：正向、負向和邊界值等測試。
- 生成單元測試自動化測試腳本：依據受測物的程式碼、資料結構與細部設計引導生成式AI獲得更適合的測試程式，同時，也可引導生成式AI進行測試程式之除錯，但因其在除錯方面較薄弱，必要時還是須由人為介入。

三、目標 I 3：整合測試

目的：已通過單元測試的軟體單元，組合後未必能順利執行，因此以生成式AI協助整合測試時，透過提示工程方法針對「內部介面」與「外部介面」來引導生成整合測試之測試個案及測試程式，以便根據測試個案執行測試。

執行原則：

- 生成整合測試個案設計：一般的整合測試策略是由上而下，或從底層往上，通常適用在階層式結構的軟體。透過軟體單元間的互動與介面、工作流（workflows）及系統與其它系統之外部介面等情境來引導生成式AI協助產生整合測試案例，包括：內部介面、外部介面和介面之效能等測試案例。
- 生成整合測試自動化測試腳本：整合的範圍越大越難將問題孤立出來，因此整合策略應採遞增式，並依據介面規格書引導生成式AI，

以獲得更適合的測試程式，同時，也可引導生成式AI進行測試程式之除錯，但因其在除錯方面較薄弱，必要時還是須由人為介入。

四、目標 I 4：系統測試

目的：系統測試著重整體系統的功能與行為，確認系統完整且符合使用者需求，系統測試包括：功能性測試、非功能性測試如：效能、負載及壓力測試等。

以生成式AI協助系統測試時，透過提示工程方法根據規劃之系統功能及效能描述，來引導生成系統測試之測試個案及測試程式以進行測試。

執行原則：

- ▶ 生成系統測試個案設計：可從功能、非功能以及例外三種情境來引導生成式AI協助產生系統測試案例，包括：功能、非功能等測試。
- ▶ 生成系統測試自動化測試腳本：可運用使用者需求、使用案例及使用者手冊引導生成式AI，以獲得適合的測試程式。
- ▶ 以例外案例引導補充：由於真實的運作環境通常包含各種可能性，往往有許多開發時無法預期的錯誤，大多數超出真實發生過的案例，需要特別引導或人為補充。
- ▶ 對話型生成式AI的限制：單元測試及整合測試屬於白箱測試，可將程式碼複製於對話型生成式AI的提示工程中，產生的測試腳本差異較小，系統測試屬於黑箱測試，提示工程只進行文字描述，受語意影響所產生的測試腳本有很大差異，應注意需人為引導或自行判斷進行修改。

五、目標 I 5：部署運行

目的：以生成式AI協助系統部署，透過提示工程方法來引導部署人員根據部署規劃內容，產生系統上線所需的文件（如：安裝指南、使用手冊、配置文件等）並進行部署作業；以及於系統上線後進行系統測試和驗證與效能監控，確保系統正常運行；並透過監控和管理系統運行的狀態和效能，執行日常維護及故障排除，及備份與恢復系統資料之演練。

執行原則：

- ▶ 上線文件生成及部署作業：系統上線所需的文件包括：安裝指南、使用手冊、配置文件等，皆可由生成式AI協助自動生成，直至文件備齊，部署人員則可進行部署作業。
- ▶ 上線系統測試和驗證：部署階段上線系統測試案例是評估系統是否符合交付標準以及能被使用者接受，能發現的缺陷包括：系統運作流程不符合業務或使用者需求、系統不符合合約或法規要求、非功能性失效（如：安全漏洞與性能效率不足）。
- ▶ 軟體效能監控及管理：於監測系統的運行狀態並生成警報，藉由生成式AI分析包含CPU使用率、網路流量、磁碟空間等系統指標，監控系統日誌、安全日誌等日誌數據，以及監測系統運行狀態、效能、配置設定、服務可用性等，生成分析這些數據，識別潛在的問題，並生成相關的報告和警報。但其建議和輸出應該經過人工驗證和審查，以確保其準確性和合規性。

第四節
軟體安全

軟體安全涵蓋軟體開發生命週期的所有階段，應於各階段應用生成式AI協助，運用在資安威脅與弱點的管理、資安威脅與弱點分析，以及持續優化資安威脅與弱點管理上。

整體資安能力涵蓋機密性、完整性、可用性（稱為CIA Triad），是一種安全模型，用於描述和保護資料和資訊系統，以確保數據的安全性並幫助組織確保其資訊資源不受未經授權的存取、損壞或阻斷。生成式AI可依據系統實際的使用環境（如：雲端、行動裝置、網路連線安全性等）建議合適的加解密演算法、身分驗證機制等。

以下說明CIA Triad的定義與範例，以協助開發團隊選擇合適的內容透過生成式AI工具協助提升其資安能力。以下分別說明CIA以及實際執行時可考慮的重點，這些重點均可以借助生成式AI來提供開發團隊相關技術的建議。

本階段的目標分為S1：機密性、S2：完整性、S3：可用性。透過這些目標的指引，說明如何利用前述之提示工程方法，發展軟體資訊安全。

一、目標S1：機密性

目的：以生成式AI協助，確保數據只能被授權的人或實體所訪問和查看的性質。這意味著數據需要得到保護，防止未經授權的存取、洩露或盜竊。

執行原則：

- 使用者密碼：當使用者在登錄網站或應用程式時，他們的密碼應該是保密的，只有授權的使用者能夠訪問該資源。
- 保密文件：對於敏感文件或機密資訊，可以使用加密技術將其保護起來，只有授權的人員才能解密和訪問。
- 網路通信加密：使用加密協議（例如：TLS）來加密網路通信，以防止未經授權的人能夠截取和閱讀數據。

二、目標S2：完整性

目的：以生成式AI協助，確保數據在傳輸、儲存或處理過程中不被意外或惡意地修改、竄改或損壞的性質。保護數據完整性的措施旨在防止數據被非法修改，並確保數據的準確性和完整性。

執行原則：

- 數位簽章：使用數位簽章技術，將數據與其相關的數位簽名進行綁定，以驗證數據的完整性和真實性。
- 文件校驗和完整性檢查：使用校驗和完整性檢查工具，例如：哈希函數，對文件進行計算散列值，以檢測文件是否在傳輸或存儲過程中被竄改。
- 安全性漏洞掃描：對系統和應用程式進行定期漏洞掃描，以確保系統的完整性，並修補發現的漏洞和弱點。

三、目標S3：可用性

目的：以生成式AI協助，確保數據和資訊系統在需要時能夠正常使用和訪問的性質。這意味著數據和系統需要可靠、穩定，並且能夠抵抗各種可能導致服務中斷或無法訪問的威脅和風險。

執行原則：

- ▶ 備份系統和故障轉移：設計具有備份/併行組件的系統，以確保在某個組件失效時，系統仍能繼續提供服務。
- ▶ 網路負載平衡：通過使用負載平衡器，將流量分配到多個伺服器上，以提高系統的可用性和性能。
- ▶ 災難復原計畫：制定災難復原計畫（DRP），包括：數據備份、備份站點和應急程序，以確保系統在災難事件後能夠快速恢復運行。

第五節 環境建構

環境建構軟體亦涵蓋軟體開發生命週期的所有階段，應於各階段應用生成式AI協助，運用在管理技術文件儲存與變更、建立程式碼（包括使用的提示）版本管理策略、以及程式碼提交與發布管理上。

有效保存系統開發各個階段的資料，包含生成式AI的建議與團隊產出的重要技術資料與經驗教訓，並進行分享復用、管理，監控的建構管理與版本管理、程式碼管理。

本階段的目標分為E1：管理技術文件儲存與變更、E2：建立程式碼版本管理策略、E3：程式碼提交與發布管理。透過這些目標的指引，說明如何提升生成式AI整體開發環境建構與管理的效率。

一、目標E1：管理技術文件儲存與變更

目的：在專案開發過程中的技術資料以生成式AI相關內容（由工程師Copy/Paste AI建議，或是透過AI生成的技術文件檔案（如：Mockup、HTML），均需要有合適的保管與分享機制來有效保存與共享。透過建構管理系統，可以追蹤和管理項目的變更，確保變更的可追蹤性和可控性。以有助於確保變更的準確性和一致性，並同時減少潛在的風險和問題。

執行原則：

- ▶ 由管理單位統一建立文件的目錄結構，以統一存放各種技術文件，如：架構設計、需求規格、測試個案、各類計畫等。
- ▶ 配合敏捷迭代開發每個迭代的最後一天（如：第二週週五下午）重構活動，生成必要的技術文件或修改現有的技術文件。
- ▶ 配合管理單位稽核活動定期（如：半年）稽核是否納入相關文件，以協助專案執行。

二、目標E2：建立程式碼版本管理策略

目的：透過這些程式碼（包括使用的提示）版本管理記錄，可以提供準確的資訊和指引，以支持項目的開發和維護過程，以及多人與跨團隊開發協作。

執行原則：

- 建立分支、主幹的目的或不建立分支目的、打包與發布方式、操作場景如開發人員數量/發布頻率提交通過準則等，以建立程式碼管理的策略，並以此策略透過生成式AI提供版本協作建議。
- 透過團隊於生成式AI互動，提供生成式AI上述版本管理策略，由生成式AI建議合適的分支/合併/打標籤/發布策略（類似GitFlow概念），以建立適合團隊需求的程式碼管理策略，協作多人與跨團隊的開發協作。

三、目標E3：程式碼提交與發布管理

目的：檢查建構過程中是否符合建構管理和變更管理的要求，以及內容是否正確、完整和符合標準。有助於確保生成式AI環境的可靠性和一致性，並維護建構基準的有效性。

執行原則：

- 分析現有程式碼的Code Base，透過提供生成式AI現有程式碼的Repository、常見的版本管理問題（如：重複出現的程式碼或分支、不明分支、上錯版本、可讀性低、不易維護等）、可由生成式AI建議如何改善，以便優化Repository提高可用性。
- 透過生成式AI強化程式碼註釋與提高可讀性（如：參數命名原則、撰寫風格、code smell、冗餘程式碼等），並建立必要的技術文件（如：class diagram, sequence diagram等）與程式碼說明文件及API使用文件（如：透過GitHub Copilot產出相關文件）等。

► 將欲提交的程式碼、提交目的地（分支/主幹）、成功的條件（如：通過審查），由生成式AI建議搭配如：Jenkins進行整合性工作流管理，以強化DevOps工作能力。

第六節 維運監控

一個企業IT環境的正常運作，企業IT維運擔任了相當重要的角色，而在傳統IT維運的過程中，往往需要投入大量IT專業人力進行繁複的系統監控與維運作業，但因企業IT運作環境日趨複雜，過程中生成的維運資料日益增加，單靠人工已無法有效並完整的分析與判讀。

近來生成式AI技術因應大數據、機器學習及硬體運算能力日新月異的發展，而有大量各式的應用，同時在系統監控與維運作業中，生成式AI更是足以擔任重要的輔助角色，協助企業更有效率地進行系統監控與維運，促使企業IT環境運作更加穩定及安全。

本階段的目標分為O1：完善網路設備管理、O2：提升網路資安事件處理時效、O3：提升軟體資訊安全、O4：強化軟體效能監控能力。透過這些目標的指引，說明如何透過生成式AI輔助企業完善系統監控與維運的執行品質。

一、目標O1：完善網路設備管理

目的：透過生成式AI協助完善網路設備管理，降低遭駭客入侵風險。

執行原則：

- 為了解現有設備組態及運作現況，透過Syslog或SNMP等方式收集網路設備資訊（如：版本、系統運作Log），同時進行資產盤點，了解管理範圍內有多少網路設備。
- 生成式AI收集設備原廠KB資訊，建立弱點資料庫，定期比對資產組態是否出現重大系統弱點或運作異常，並自動通知以及產生建議處理方式。

二、目標O2：提升網路資安事件處理時效

目的：透過生成式AI協助增強網路異常行為偵測效率，提升網路資安事件處理時效。

執行原則：

- 首先利用網路工具收集網路封包傳輸狀況，了解網路運作狀況並進行特徵識別，另外亦收集各大資安論壇資訊，獲得當下資安論壇正在討論的情資。
- 透過生成式AI進行特徵分析與情資比對是否出現資安論壇正在談論的攻擊事件，並自動通知以及產生建議處理方式，以求盡快發現網路異常事件，提升資安事件處理時效。

三、目標O3：提升軟體資訊安全

目的：強化軟體資訊安全管理，以提供業務持續運作環境，並符合相關法規之要求，使其免於遭受內、外部的蓄意或意外之威脅。

執行原則：

- 系統開發日誌（Log）不管是在實驗階段或者產品階段，日誌的記錄可以經由生成式AI協助分析過去成功、失敗的參數組合。也有助於查找系統錯誤及異常的原因，或知道引入新版本的模型或系統之後帶來的成效。
- 應施行日誌存取控管，避免未經授權使用者惡意讀取、竄改或刪除日誌。
- 針對使用者帳號進行行為分析，依據使用者帳號登入的時間、登入方式、來源IP與使用設備種類等資訊，登入後的使用功能頻次進行分析，若發現使用者帳號使用行為改變或異常，即自動發出告警且鎖定異常帳號。

四、目標O4：強化軟體效能監控能力

目的：提高系統部署後的運轉效能監控能力，透過生成式AI協助即時與準確的監控系統運轉效能，以提高系統可用度。

執行原則：

- 識別系統效能相關KPI（如：記憶體/CPU使用率、能耗、I/O等），以作為後續監控的依據。
- 將上述KPI的運轉資訊寫入系統日誌。
- 使用生成式AI來分析以檢測潛在的錯誤、異常行為或效能問題，如：預測性分析由AI分析過去的效能數據，並預測未來的效能趨勢；或自動化故障檢測，由生成式AI自動檢測應用程序中的故障或效能下降，提供性能優化建議與相關警示資訊。

【附錄】

參考資料

（1）「生成式AI導入指引-企業應具備的AI素養」（2023年7月），財團法人資訊工業策進會。

（2）Microsoft Responsible AI Standard, v2, June 2022.

（3）ISO/ICE/IEEE 12207:2017 Systems and Software Engineering: Software life cycle processes.

（4）Sunil Ramlochan, “Tree of Thought Prompting - Walking the Path of Unique Approach to Problem-Solving” , Prompt Engineering Institute, June 7, 2023.

（5）Ipek Ozkaya, “Application of large language models to software engineering tasks: Opportunities, risks, and implications” , IEEE Software, vol. 40, no. 3, pp. 4–8, 2023.

發行單位：財團法人資訊工業策進會

發行人：卓政宏

總編輯：楊仁達

編輯執行單位：軟體技術研究院

編輯主任委員：蒙以亨

編撰群：劉培森、王義智、林　謙、黃芳蘭、陳彥希
劉文楷、江彩雲、戴宜萍、曾裕勝、游展鑑
李宏儒、林威志、張耿益、吳怡欣、張淑慧
張凱筌、楊堯順、賴傳尉

顧問群：蔡澤銘、陳立群、洪春暉、張育誠、戴偉峻
顧振豪、林敬文、何文楨、徐志浩、劉文山

地址：10622台北市大安區和平東路二段106號11樓

電話：02-6607-3133

傳真：02-6607-3507

網址：https://www.iii.org.tw

Email：aigcassistguide@iii.org.tw


軟體技術研究院
Software Technology Institute