Event Storming 規格化需求模型與 AI 自動生成之應用

專題編號：114-2-CSIE-S020
執行期限：114 年第 1 學期至 114 年第 2 學期
指導教授：鄭有進
專題參與人員：112820003 辛政隆
112820018 許景喬
112820025 陳翊揚
112820042 杜詠霖

一、摘要

本研究旨在結合領域驅動設計（Domain Driven Design, DDD）[1] 與整潔架構（Clean Architecture），建立一套由 AI 自動生成應用程式的方法。由於標準事件風暴（Event Storming）[2] 缺乏對領域隱性知識的描述，難以直接支援 AI 程式生成，因此本研究延伸 Event Storming，加入隱性知識建模與領域模型補充機制，產生可供 AI 生成原始碼與測試之中介 JSON 規格，並實作名為 Event2AISpec 的工具，以日曆應用程式驗證方法可行性。

關鍵字： Event Storming、Domain Driven Design、結構化需求規格、AI 輔助軟體工程

二、緣由與目的

Event Storming 為 DDD 常用的協作建模技術，但其便利貼等視覺化產物缺乏欄位型別、驗證規則等隱性知識，需仰賴人工轉譯，容易造成需求遺漏與架構不一致。隨著大型語言模型（Large Language Model, LLM）發展，直接以此類非結構化需求進行程式生成，更易產生幻覺（Hallucination）問題。因此，本研究擴充標準 Event Storming，將隱性知識轉換為機器可讀之中介規格，以支援符合 DDD 與 Clean Architecture 的 AI 程式生成。

三、研究報告內容

（一） 系統整體架構

前端使用 Vue 3 + Konva.js 做拖曳式畫布，後端用 Spring Boot 4.0 處理 REST API 與 WebSocket 協作，資料層用 MySQL 8.0，並內建分析引擎自動將 Event Storming 轉為 JSON。

（二） 延伸事件風暴模型

標準 Event Storming 以指令（Command）、領域事件（Domain Event）、聚合（Aggregate）及讀取模型（Read Model）等描述業務流程，但缺乏程式碼生成所需的技術合約。本研究引入技術性延伸，將各元素明確對應至結構化 JSON 欄位：指令對應使用案例（Use Case）實作類別、讀取模型對應輸入 DTO、聚合包含方法簽名與屬性定義、反應器（Reactor）對應非同步事件監聽器，並以決策（Policy）建模業務規則。

（三） 領域模型元件

為支援強型別聚合定義，本研究引入以 UML 類別框呈現的領域模型元件，關聯四種類別之一：AGGREGATE、ENTITY、VALUE_OBJECT 或 ENUM。每個欄位需指定名稱、資料型別及可選約束（如 orderId: UUID {PK}）。AGGREGATE 名稱在同一看板中必須唯一，以確保規格提取時的明確對應。

（四） JSON 轉換

團隊完成看板建模後，系統自動將圖

形關係解析為以 Use Case 為基本單元的中間規格層 JSON 檔。每份 JSON 規格嚴格包含以下欄位：useCaseName（Use Case 名稱）、input（輸入欄位與型別定義）、actor（觸發角色）、aggregateName 與 aggregateWithAttributes（聚合邊界與實體屬性）、domainEvents（含 eventName、domainEventAttribute、reactor、policy）、method（核心業務邏輯方法）及 comment（設計備注）。

透過此中間規格層，業務需求得以完全結構化，消除自然語言的隨意性，為下游 AI 自動化實作提供精確的上下文，同時保留設計假設與隱含業務規則。

（五） 系統整合與外部參考

1. 規格驅動 AI 程式碼生成

使用者將產生的 JSON 規格輸入 AI 模型，利用開源專案 ai-coding-exercise [3] 的提示詞工程技術，引導 LLM 在清晰的架構上下文中自動生成後端 API、Use Case、領域實體（Domain Entity）與儲存庫（Repository），確保所有生成程式碼皆符合 DDD、事件溯源（Event Sourcing）及整潔架構（Clean Architecture）的分層標準。

2. 架構驗證與靜態分析

生成的程式碼自動導入整合於 ai-coding-exercise 的靜態分析機制，進行架構對齊檢查，嚴格驗證是否違反分層規範，例如：Controller 是否直接注入 Repository、測試檔案的套件（Package）結構是否正確、Value Object 是否錯誤使用了靜態工廠方法等。任何違規結果將回饋至系統進行架構修正。

四、 案例研究與需求落差分析

本研究以日曆系統作為主要實驗案例，對多用戶邀請確認流程與行程管理流程進行建模，並成功匯出對應的 JSON 規格檔案。以 InviteUserToCalendar 這個 Use Case 為例，用戶（OWNER）透過三個強型別輸入欄位發起邀請，聚合執行對應方法後發布領域事件，並由 InAppNotificationService 非同步推送通知，同時附帶限定擁有者邀請、最多 10 位成員等業務規則約束。實驗揭示了兩類 AI 生成的系統性落差：型別降級導致 UUID、LocalDateTime 等強型別被宣告為 String，以及額外欄位生成導致規格未定義的欄位（如 userId、ownerId）被自行添加。這些結果顯示 AI 在缺乏強型別約束的情況下容易產生語義偏差，突顯了結構化規格在品質審查上的潛在價值。

五、 結論與未來工作

本研究提出一套基於延伸 Event Storming 模型的結構化需求規格流程，透過將業務模型轉換為機器可讀的 JSON 中間規格層，有效縮小了從需求到程式碼的語義落差，並提升了 AI 生成程式碼的業務精確度。未來將聚焦於擴展規格的語義豐富度，以及開發自動化規格修正與架構記憶機制，使需求模型能更彈性地對齊不同團隊的技術規範。

參考文獻

[1] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston, MA: Addison-Wesley Professional, 2003.

[2] A. Brandolini, Introducing EventStorming: An Act of Deliberate Collective Learning. Leanpub, 2018.

[3] T. Chen, ai-coding-exercise. (2025). [Source code]. Available: https://gitlab.com/TeddyChen/ai-coding-exercise