# 專利撰寫專案說明

## 專案概述

這是一個**基於 AI Agent 的自動化專利申請文件撰寫系統**，採用**多 Agent 協作架構**，將複雜的專利撰寫任務分解為多個專業子任務，由不同的專業 Agent 按順序完成。

## 核心運作原理

### 1. **架構設計：多 Agent 流水線**

使用者輸入(技術交底書.docx)  
 ↓  
[主 Agent 專案經理]  
 ↓  
建立工作目錄結構 (output/temp\_[uuid]/)  
 ↓  
按順序呼叫 8 個專業子 Agent  
 ↓  
輸出完整專利文件

### 2. **子 Agent 流水線（8 個階段）**

| 階段 | 子 Agent | 核心任務 | 輸入 | 輸出 |
| --- | --- | --- | --- | --- |
| 1 | input-parser | 解析技術交底書 | raw\_document.docx | parsed\_info.json |
| 2 | patent-searcher | 檢索相似專利 | parsed\_info.json | similar\_patents.json |
| 3 | outline-generator | 生成專利大綱 | parsed + similar | patent\_outline.md |
| 4 | abstract-writer | 撰寫摘要 | outline | abstract.md |
| 5 | claims-writer | 撰寫權利要求 | outline + abstract | claims.md |
| 6 | description-writer | 撰寫說明書主體 | outline + claims | description.md (>10000字) |
| 7 | diagram-generator | 生成 Mermaid 圖表 | description + structure | \*.mmd 檔案 |
| 8 | markdown-merger | 合併最終文件 | 所有 md + 圖表 | complete\_patent.md |

### 3. **資料流轉機制**

* **結構化傳遞**：使用 JSON 檔案在 Agent 間傳遞結構化資料
* **版本控制**：每個階段輸出保留多版本（v01, v02…）
* **依賴管理**：嚴格按順序執行，後一個 Agent 依賴前一個的輸出
* **品質把關**：每個階段完成後驗證輸出格式和內容完整性

## 標準化目錄結構

output/temp\_[uuid]/  
├── 01\_input/ # 原始輸入和解析結果  
│ ├── raw\_document.docx # 原始技術交底書  
│ └── parsed\_info.json # 解析後的結構化資訊  
│  
├── 02\_research/ # 專利檢索和分析  
│ ├── similar\_patents.json # 相似專利搜尋結果  
│ ├── prior\_art\_analysis.md # 現有技術分析  
│ └── writing\_style\_guide.md # 寫作風格參考  
│  
├── 03\_outline/ # 專利大綱和結構  
│ ├── patent\_outline.md # 完整專利大綱  
│ └── structure\_mapping.json # 結構映射檔案  
│  
├── 04\_content/ # 核心撰寫內容  
│ ├── abstract.md # 摘要  
│ ├── claims.md # 權利要求書  
│ ├── description.md # 具體實施方式（>10000字）  
│ └── figures.md # 附圖說明  
│  
├── 05\_diagrams/ # Mermaid 圖表  
│ ├── flowcharts/ # 流程圖  
│ │ ├── system\_architecture.mmd  
│ │ └── method\_flow.mmd  
│ ├── structural\_diagrams/ # 結構圖  
│ │ ├── component\_structure.mmd  
│ │ └── data\_flow.mmd  
│ └── sequence\_diagrams/ # 時序圖  
│ └── operation\_sequence.mmd  
│  
├── 06\_final/ # 最終輸出檔案  
│ ├── complete\_patent.md # 完整專利文件  
│ ├── patent\_application.docx # Word 格式專利申請  
│ └── summary\_report.md # 專案總結報告  
│  
└── metadata/ # 專案元資料和日誌  
 ├── project\_info.json # 專案基本資訊  
 ├── agent\_logs/ # 各代理執行日誌  
 └── quality\_check.json # 品質檢查結果

**設計理念**： - **模組化**：每個子 Agent 只關注自己的目錄 - **可追溯**：保留所有中間檔案便於偵錯 - **可恢復**：透過 checkpoint 機制支援失敗恢復

## 使用方式

### **基本使用流程**

#### **步驟 1：準備輸入檔案**

* 準備技術交底書（.docx 格式）
* 包含：技術背景、技術方案、優點、實施例等

#### **步驟 2：發起請求**

使用者：請幫我撰寫專利申請文件，輸入文件是 xxx.docx，專案 UUID 是 abc123

或者讓系統自動生成 UUID：

使用者：請基於這個技術交底書撰寫專利

#### **步驟 3：系統自動執行**

主 Agent 會： 1. 建立目錄結構 output/temp\_abc123/ 2. 依次呼叫 8 個子 Agent 3. 監控執行進度和錯誤 4. 輸出最終結果路徑

#### **步驟 4：獲取結果**

最終文件位置：

output/temp\_[uuid]/06\_final/complete\_patent.md  
output/temp\_[uuid]/06\_final/summary\_report.md

## 技術實作原理

### **1. Agent 協作機制**

* **主 Agent**：專案經理，負責協調和品質把控
* **子 Agent**：8 個專業 Agent，使用 Task 工具呼叫
* **通訊方式**：透過檔案系統傳遞 JSON 和 Markdown 檔案

### **2. 子 Agent 呼叫方式**

使用 Task 工具，例如：

Task(  
 subagent\_type="input-parser",  
 prompt="解析 output/temp\_abc123/01\_input/raw\_document.docx，輸出到 parsed\_info.json",  
 description="解析輸入文件"  
)

### **3. 品質保證機制**

* **格式驗證**：JSON 格式、Mermaid 語法檢查
* **內容驗證**：字數要求（說明書 >10000 字）
* **一致性檢查**：術語在不同章節保持一致
* **錯誤恢復**：最多重試 3 次，儲存錯誤日誌

### **4. 錯誤處理**

* 子 Agent 失敗 → 記錄日誌 → 重試（最多 3 次）
* 輸入檔案缺失 → 檢查前置 Agent 輸出
* 格式錯誤 → 驗證並提示修復
* 回滾機制 → 恢復到上一個穩定版本

## 核心優勢

1. **專業分工**：每個 Agent 專注單一任務，提高品質
2. **流程標準化**：嚴格遵循中國專利法和審查指南
3. **可維護性**：模組化設計，便於偵錯和改進
4. **可擴充性**：可輕鬆新增新的子 Agent 或階段
5. **可追溯性**：完整保留中間過程，支援品質審查

## 適用場景

* **發明專利申請**：技術方案完整，需要詳細說明書
* **實用新型專利**：結構創新，需要大量附圖
* **批次專利撰寫**：可並行處理多個專案（不同 UUID）
* **專利品質提升**：學習已有專利的優秀寫作風格

## 重要注意事項

⚠️ **重要限制**： - 輸入必須是 .docx 格式 - 技術交底書需包含完整的技術方案 - 系統遵循中國專利法，不適用其他國家 - 具體實施方式章節強制要求 >10000 字 - 需要手動審查最終輸出，AI 無法替代專業專利代理人

## 品質標準

### **文件品質要求**

* 嚴格遵循中國《專利法》和《專利審查指南》規範
* 確保具體實施方式章節 > 10000 字
* 全文術語必須一致
* 各章節邏輯鏈條必須完整
* 所有中間檔案都儲存在暫存目錄中

### **目錄結構品質要求**

* 目錄結構必須 100% 符合規範，不得缺失任何子目錄
* 所有檔案必須放置在正確的目錄中
* 檔案命名必須嚴格遵循命名規範
* JSON 檔案必須格式正確且可解析
* Mermaid 圖表檔案必須以 .mmd 為副檔名

### **資料完整性要求**

* 每個子 Agent 的輸出必須包含版本號和建立時間戳記
* 結構化資料（JSON）必須包含完整的欄位驗證
* 關鍵技術術語在所有檔案中必須保持一致
* 圖表引用必須與實際檔案名稱匹配

### **輸出格式要求**

* Markdown 檔案使用標準語法
* JSON 檔案使用 2 空格縮排
* Mermaid 圖表語法正確性驗證
* 最終文件必須包含完整的目錄和章節編號

## 錯誤處理和恢復機制

### **子代理執行失敗處理**

* 某個子代理失敗時，儲存錯誤日誌到 output/temp\_[uuid]/[agent\_name]\_error.log
* 檢查輸入檔案是否存在且格式正確
* 必要時重新執行前置子代理
* 最多重試 3 次，超過後人工介入

### **資料一致性檢查**

* 每個子代理完成後驗證輸出檔案完整性
* 關鍵欄位在不同檔案間的一致性檢查
* JSON 格式和 Mermaid 語法驗證
* 檔案大小和基本內容合理性檢查

### **回滾機制**

* 保留每個子代理的多個版本輸出
* 發現問題時可回退到上一個穩定版本
* 清理機制：專案完成後可選擇刪除中間檔案
* 重要節點自動建立 checkpoint

### **監控和日誌**

* 每個子代理的執行時間記錄
* 資源使用情況監控
* 關鍵決策點的日誌記錄
* 最終生成專案執行報告

## 快速開始

如果您想實際使用這個系統撰寫專利，可以提供您的技術交底書文件路徑，系統會立即啟動完整的撰寫流程！

**範例指令**：

請基於 [文件路徑]/技術交底書.docx 撰寫專利申請文件

系統會自動完成所有步驟，並在 output/temp\_[uuid]/06\_final/ 目錄下生成完整的專利文件。