# 企業整合與數位轉型藍圖 統一 Opcenter APS、UiPath 和 SAS Viya 以實現策略性成長

## **一、重整後的章節大綱**

### **版本資訊**

* **文件編號： STRATEGY-FY25-001  
  密等： 內部機密**

#### 版本控制列表

| **版本** | **日期** | **作者** | **修訂說明** |
| --- | --- | --- | --- |
| 1.0 | 2025/07/23 | Gemini | 初版建立：整合開源軟體策略、IDE 策略、數據網格架構分析及合作契約附件。 |
| 1.1 | 2025/07/24 | Vincent | 重製版重製版面 |

### **執行摘要**

* **現狀痛點：**
  + 目前 Opcenter APS、UiPath 和 SAS Viya Studio 各自獨立運作，導致營運效率低下。
  + 人工流程繁瑣，產生大量的「搜尋時間成本」。
  + 碎片化的現狀阻礙了企業的敏捷性與數據驅動決策的能力。
* **核心建議：**
  + **策略性整合：** 建立強健的整合架構，運用 API 與事件驅動模式，將 Opcenter APS、UiPath 和 SAS Viya 無縫連結，形成一個協同運作的生態系統。
  + **AI 驅動自動化：** 將人工智慧（如 GPT/機器學習）融入開發生命週期，從程式碼生成到智慧郵件處理及自動化測試，加速創新並提升營運智慧。
  + **數據中心化與治理：** 建立統一的數據層，並實施嚴謹的數據對齊與版本控制實踐，確保所有平台上的數據完整性與可存取性。
  + **統一管理平台：** 將現有分散的工具與日曆整合，逐步演進為全面的資源、時間與工作管理平台。
  + **績效與問責：** 制定清晰的關鍵績效指標（KPIs），並建立具約束力的問責機制，以確保專案成功執行並達成可衡量的業務成果。
* **預期效益：** 顯著提升生產力、改善準時交貨率、優化庫存管理、加速新功能上市時間、提升客戶滿意度，並形成一個正向的數據驅動工作循環。

### **現況分析與挑戰識別**

本節將深入探討 Opcenter APS、UiPath 和 SAS Viya 各自的現有功能與關鍵痛點，並闡明其獨立運作所導致的效率低下問題。

* **Opcenter APS：生產排程與營運缺口**
  + **功能與優勢：** 先進的排程與規劃軟體，旨在優化生產排程、縮短停機時間並提升客戶服務。支援長期、中期及詳細的順序與排程，提供 Opcenter Scheduling 和 Opcenter Planning 等模組。可偵測瓶頸、減少設定時間、提高機器利用率、降低在製品庫存，並能將準時交貨率提升高達 50%。可與 ERP 和 MES 無縫整合，特別是與 MSSQL 資料庫介接，支援圖形化生產規劃與客製化 KPI。
  + **三大痛點：**
    - **外部演算法缺乏版本控制：** 目前用於 Opcenter APS 的五種外部開發演算法缺乏 Git 版本控制，僅能透過套件更新方式管理，導致技術債務和營運風險。
    - **異常通知仰賴人工郵件：** 現行系統仰賴供應商透過電子郵件發送異常通知，流程被動且低效，增加「搜尋時間成本」和問題解決時間。
    - **自動化範圍受限：** 將 Opcenter APS 的「相同架構」擴展到公司內部會議通知的需求，揭示了內部流程自動化潛力的未被充分利用，導致「時間管理」和「工作管理」碎片化。
* **UiPath：自動化潛力與溝通脫節**
  + **現況用途：** 目前用於公司內部和外部的郵件與系統通知，利用 RPA 執行常見的郵件任務，並能根據郵件內容在任務管理應用程式中建立任務。
  + **三大挑戰：**
    - **郵件智慧化與進階處理不足：** 停留在基礎 RPA 層面，未能充分利用其進階 AI 能力進行智慧內容理解和自動化決策，導致人工分析和處理郵件內容的瓶頸。
    - **「Low Code 卻與 VBA 無異」的觀感：** 非資訊部門同事認為 UiPath 與 VBA 差異不大，阻礙了其更廣泛的採用，並導致溝通碎片化。
    - **Salesforce 整合缺口：** 無法利用自動化的特性將客戶管理系統 Salesforce 的資料進行客戶與場景分類，錯失數據驅動的客戶關係管理機會。
* **SAS Viya：數據分析與擴展性挑戰**
  + **現況用途：** 完整的數據分析工具，支援從 Azure 環境中獲取數據，並能與 Azure Machine Learning/Synapse Machine Learning 整合，支援 Python、R 等多種程式語言進行模型建構與部署。SAS Studio 也支援 Git 版本控制功能。
  + **三大挑戰：**
    - **即時與固定時程串流開發重複：** 對於即時串流與固定時程串流（5 分鐘與 1 小時）的功能，需要額外多次重複開發，導致開發效率低下。
    - **額外演算法開發與報告生成：** 連結不同資料庫、開發其他額外演算法、生成不同特徵報表與經營管理分析報告，都需要額外開發，暗示開箱即用分析能力的不足。
    - **Kubernetes 環境與網路控管複雜性：** 大量的 Kubernetes 環境與網路控管問題，導致 SAS Viya 「並不能與其他軟體合作」，阻礙了其部署和擴展。
* **跨系統共通挑戰：**
  + **數據對齊問題：** 「資料庫的內容對齊」是跨不同系統整合的重大挑戰，導致數據孤島、數據不一致。
  + **SAP 功能整合：** 查詢中提及「SAP 的功能」，但未明確其具體角色，暗示與現有 SAP ERP 系統的整合將是複雜的企業整合挑戰。
  + **統一網路溝通介面缺失：** 缺乏「統一的網路溝通介面」和「現行的工作日曆就有四個以上」的問題，凸顯了公司內部溝通和時間管理的碎片化。
  + **缺乏正向工作循環：** 現有環境未能形成「分析報表是一件事，跟如何思考下一步對策並無法有效的正向工作循環」，表示從數據洞察到實際營運執行之間存在脫節。

### **策略願景與三年營收計畫**

本節將闡述轉型後的策略願景，並提出未來三年的營收目標拆解、關鍵績效指標（KPIs）及具體的兩個月內里程碑，以確保業務目標與技術實施緊密結合。

* **三年營收目標拆解：**
  + **第一年：基礎設施強化與核心流程自動化**
    - **目標：** 營收增長 15%。
    - **拆解：** 主要來自於營運效率提升所帶來的成本節省，以及部分新客戶獲取。
    - **重點：** 完成核心系統（APS、UiPath、SAS Viya）的基礎整合，實現關鍵數據對齊，並將 UiPath 自動化擴展至高頻繁、重複性郵件處理與內部通知。
  + **第二年：數據賦能與智慧決策**
    - **目標：** 營收增長 25%。
    - **拆解：** 透過 SAS Viya 強化數據分析能力，實現更精準的生產預測、供應鏈優化，以及基於 Salesforce 數據的客戶洞察。
    - **重點：** 深度整合 SAS Viya，實現即時數據串流與分析，並將 AI/機器學習模型應用於業務決策。
  + **第三年：平台化與創新服務**
    - **目標：** 營收增長 35%。
    - **拆解：** 建立統一的資源、時間與工作管理平台，實現跨部門、跨系統的無縫協作。透過自動化和數據洞察，開發新的增值服務或優化現有產品。
    - **重點：** 實現全面的平台化願景，探索 AI 驅動的創新應用，並將技術優勢轉化為市場差異化。
* **關鍵績效指標（KPI）：**
  + **營運效率：**
    - 系統可用性與正常運行時間 (System Availability & Uptime)： ≥99.9%
    - 問題解決時間 (Resolution Time)：減少 30%
    - 錯誤率 (Error Rates)：降低 50%
    - 異常事件頻率 (Incident Rates)：降低 40%
    - 技術債務減少率 (Technical Debt Reduction)：減少 25%
  + **開發與創新：**
    - 開發週期時間 (Development Cycle Time)：縮短 20%
    - 自動化測試覆蓋率 (Automated Test Coverage)：提升至 80%
    - 新功能採用率 (Feature Adoption Rates)：提升至 75%
    - 程式碼部署頻率 (Code Deployment Frequency)：增加 50%
  + **數據與分析：**
    - 數據對齊準確性 (Data Alignment Accuracy)：達到 98%
    - 數據查詢效能 (Database Query Performance)：提升 30%
  + **業務影響：**
    - 專案交付率 (Project Delivery Rates)：提升至 90%
    - 客戶滿意度與留存率 (Customer Satisfaction & Retention)：提升 15%
    - 業務影響 (Business Impact)：顯著正向影響
* **問責機制：**
  + **問責框架與 KPI 契約：** 與總經理簽訂正式的 KPI 契約，明確責任機制，建立清晰期望、基於經驗的監督、問責文化，並管理問責挑戰。
  + **KPI 契約條款：** 若未能達成約定的關鍵績效指標，將被視為未履行職責，其後果將被視為自動離職。
* **兩個月內需完成的里程碑：**
  + **第一個月：基礎評估與規劃**
    - 數據資產盤點與對齊初探。
    - APS 外部演算法版本控制導入評估。
    - UiPath 郵件處理需求細化。
    - 整合架構概念設計。
    - 專案團隊組建與職責定義。
  + **第二個月：原型開發與風險識別**
    - APS 異常郵件自動化通知原型。
    - UiPath 郵件分類 AI 模型 POC。
    - SAS Viya 數據連接器測試。
    - 整合前環境檢查清單初稿。
    - 潛在技術風險識別。

### **整合系統架構與開發藍圖**

本節將詳細描述整體系統架構，定義 Opcenter APS、UiPath 和 SAS Viya 各自的功能模組及其串接方式，並闡述如何運用 AI 技術加速開發流程。

* **分層架構圖（概念性描述）：**
  + **操作層 (Operational Layer)：** Opcenter APS (生產排程)、Salesforce (客戶關係管理)、現有工作日曆。
  + **數據層 (Data Layer)：** 統一數據儲存（數據湖/數據倉儲）、數據對齊與轉換模組。
  + **整合層 (Integration Layer)：** API 閘道、訊息佇列/事件匯流排（Apache Kafka）、整合服務/微服務。
  + **自動化與智慧層 (Automation & Intelligence Layer)：** UiPath RPA 平台、AI/機器學習模組、SAS Viya。
  + **應用層 (Application Layer)：** 統一管理平台介面、客製化應用程式。
* **各層模組與串接方式：**
  + **Opcenter APS：**
    - **功能模組：** 生產排程核心、資源管理、訂單處理、排程優化演算法。
    - **串接方式：** 數據庫介接 (MSSQL)、API/事件發佈、外部演算法介面。
  + **UiPath：**
    - **功能模組：** RPA 機器人、郵件自動化活動、Salesforce 連接器、AI Center、Orchestrator。
    - **串接方式：** 郵件系統整合、AI/ML 整合、Salesforce API 整合、內部通知與日曆整合、事件消費。
  + **SAS Viya：**
    - **功能模組：** 數據管理（連接器）、數據探索與建模、模型部署 (MLOps)、即時串流分析、報表與視覺化。
    - **串接方式：** 數據庫連接、即時串流整合、模型部署、報表與視覺化。
* **AI/機器學習加速開發：**
  + **GPT/Codex 於程式碼生成：** 加速整合程式碼開發、客製化演算法輔助開發、自動化測試程式碼生成、文件與註釋生成。
  + **機器學習應用：** 智慧郵件分類與響應、自動化數據映射、自動化測試與缺陷預測。
* **開源工具建議：**
  + **版本控制系統：** Git (搭配 GitLab/GitHub)
  + **整合與自動化流程 (CI/CD)：** Jenkins
  + **資料庫與訊息傳遞：** PostgreSQL & Apache Kafka
  + **容器化技術：** Docker
  + **統一儀表板與視覺化：** Grafana
* **與 SAS Viya 對應策略表：**

| **功能領域 (Functional Area)** | **建議的開源軟體 (Suggested Open Source Software)** | **SAS Viya 內建功能 (SAS Viya's Built-in Capability)** | **整合策略與理由 (Integration Strategy & Rationale)** |
| --- | --- | --- | --- |
| 版本控制 (Version Control) | Git / GitLab / GitHub | SAS Studio 支援與 Git 儲存庫的整合 | **角色分工：** SAS Viya 處理「自身」的程式碼版本控制。Git 將作為通用的版本控制平台，統一管理專案中所有程式碼資產，包括 APS 演算法、UiPath 腳本以及串接各系統的整合程式。 |
| 持續整合與部署 (CI/CD) | Jenkins | SAS Viya 提供 CLI 和 REST API | **協同運作：** SAS Viya 負責「執行」分析任務，而 Jenkins 擔任「總指揮」的角色，串連起整個端到端的自動化工作流。 |
| 數據視覺化與儀表板 (Visualization) | Grafana | SAS Visual Analytics | **不同用途：** SAS Visual Analytics 主要用於「商業分析與決策」。Grafana 則用於建立「技術維運監控儀表板」，即時監控整個整合系統的健康狀態。 |
| 中央數據儲存 (Data Storage) | PostgreSQL | SAS Viya 本身是一個分析平台，會「連接」到各種數據來源 | **架構定位：** 使用 PostgreSQL 建立一個統一的數據倉儲或數據湖，作為乾淨、一致的「單一事實來源」給 SAS Viya 進行分析。 |
| 事件驅動與訊息傳遞 | Apache Kafka | SAS Viya 能夠透過其 Event Stream Processing 引擎連接到 Kafka | **核心骨幹：** Kafka 將作為整個事件驅動架構 (EDA) 的核心骨幹，負責在 APS、UiPath 和 SAS Viya 之間可靠、非同步地傳遞事件訊息。 |
| 應用程式容器化 (Containerization) | Docker | SAS Viya 平台本身就是基於 Kubernetes 部署的 | **生態系一致性：** 用 Docker 來打包為了整合而開發的「客製化輔助元件」，確保其能以和 SAS Viya 同樣標準化的方式運行和管理。 |

### **文件清單與週計畫**

* **第一個月 (第 1-4 週)：規劃、設計與基礎建設**
  + [文件] 專案範疇與目標定義書
  + [文件] 現有系統架構圖
  + [文件] 數據資產盤點與字典
  + [文件] Git 版本控制導入計畫
  + [文件] 整合架構設計草圖
  + [文件] API 與 Atom 元件架構圖
  + [文件] 郵件處理需求規格書
* **第二個月 (第 5-8 週)：開發、測試與交付**
  + [文件] 三系統串接測試計畫與測試案例
  + [文件] 每日自動異常通知與排程執行紀錄範例
  + [文件] 部署手冊 (Deployment Guide)
  + [文件] 內部 Prompt 可複用範本集
  + [文件] 完整技術文件
  + [文件] 甘特圖 (Gantt Chart)
  + [報告] 《三年業績技術架構報告書》
  + [報告] 專案結案報告與 KPI 效益展示
* **八週週進度細項：**
  + **第一週：** 專案啟動與需求盤點（召開專案啟動會議、數據資產盤點、數據對齊初步分析、Git 版本控制方案評估）
  + **第二週：** 架構設計與環境準備（完成 APS VM 演算法模組、MSSQL 結構清單與更新流程文件、繪製整合架構概念圖、細化 UiPath 郵件處理需求、完成跨系統整合 API 與 Atom 元件架構圖）
  + **第三週：** UiPath 原型開發（建立 UiPath 郵件清理、異常通知分類、會議通知模組原型流程、完成 SMTP/IMAP 權限設定、啟動 AI/ML 模型選型與數據準備）
  + **第四週：** SAS Viya 與整合設計（製作 SAS Viya 分析模組、定時排程流程與命名空間設定文件、設計三系統串接詳細流程、提交第一階段 Notion 週報與成果）
  + **第五週：** 三系統串接原型開發（實際開發三系統串接測試流程、開發 UiPath 監控 APS 數據庫異常數據原型、驗證 SAS Viya 與 MSSQL 連接）
  + **第六週：** 自動化與測試（完成每日自動異常通知與排程執行紀錄範例程式、訓練初步 UiPath 郵件分類 AI 模型 POC、撰寫整合測試案例並執行端到端測試）
  + **第七週：** 報告撰寫與文件整理（撰寫並完成《三年業績技術架構報告書》草稿、完成內部 Prompt 可複用範本、整理並提供完整技術文件與甘特圖）
  + **第八週：** 驗收、展示與總結（準備並進行專案結束時成果展示、完成所有專案文件並提交管理層驗收、若交付成果達 80% 準備結案，否則準備檢討會議）

### **整合前待辦與風險緩解**

* **環境與數據準備清單：**
  + **數據對齊與標準化：** 數據審計、定義數據標準、自動化數據映射工具、數據治理框架。
  + **系統準備：** 兼容性驗證、必要的升級與修補、數據清洗與備份。
* **安全態勢控制重點：**
  + 使用安全通訊協定 (HTTPS, TLS, SSL)。
  + 保護 API 和端點 (身份驗證、授權、安全存取令牌)。
  + 定期安全審計與審查。
  + 限制數據暴露 (數據最小化、匿名化、假名化)。
  + 建立錯誤處理機制。
  + 最小權限原則。
  + 員工與終端使用者培訓。
* **版本控制與測試計畫：**
  + **版本控制：** 對客製化演算法實施 Git、統一版本控制策略。
  + **測試計畫：** 定義測試範圍與目標、測試類型與方法 (單元、整合、效能、使用者驗收)、測試案例與場景、資源分配與時間表、成功標準與報告。
* **潛在技術風險與緩解策略：**

| **潛在技術風險** | **具體描述** | **緩解策略** |
| --- | --- | --- |
| 兼容性與整合挑戰 | 不同供應商軟體可能存在數據格式不匹配、API 兼容性問題 | 綜合規劃與策略開發、徹底的兼容性測試、標準化介面 (RESTful API, ODBC) |
| 數據不一致性與重複 | 整合可能導致數據重複、遺失或不一致 | 數據對齊策略 (清洗、標準化、映射)、數據治理 |
| 網路與 Kubernetes 控管複雜性 | 大規模 Kubernetes 環境部署、管理複雜 | 專業知識與培訓、雲端託管服務 (Azure AKS)、網路分段與監控 |
| 客製化演算法管理困難 | Opcenter APS 外部演算法缺乏版本控制，更新困難 | 導入版本控制系統 (Git)、自動化部署流程 |
| RPA 機器人運作風險 | UiPath 機器人可能因底層流程變更或意外錯誤而故障 | 健壯的錯誤處理、持續監控與警報、流程變更管理 |
| 網絡安全風險與數據暴露 | 整合系統增加了數據洩露、未經授權存取風險 | 強化安全政策、特權存取管理 (PAM)、安全審計 |
| 員工抗拒與技能差距 | 員工可能抗拒新技術或缺乏操作新系統技能 | 有效的變革管理、充分的培訓與技能發展 |