基於大型語言模型與文字向量比對之微服務識別方法

謝知祐 [1]，黃子毓 [2]，馬尚彬 [3]
國立臺灣海洋大學資訊工程學系
E-mail: [1]4114W016@mail.ntou.edu.tw, [2]11257018@mail.ntou.edu.tw,
[3]albert@ntou.edu.tw

摘要

在遷移單體架構至微服務架構的過程中，切割與識別微服務是最重要的關鍵步驟，為因應此需求，本研究提出一種新的微服務識別方法，稱為 MICDA (Microservice Identification based on Code Semantics and Data Access)，MICDA 主要運用程式碼分析工具與大型語言模型 (Large Language Model, LLM) 提取或生成原始碼語意、資料存取模式與文字向量等資訊，並進行群聚分析與排序，以此產出多組建議的微服務分群組合。依據對六個單體系統實施 MICDA 方法的實驗結果證實，該方法能有效識別符合使用者預期的微服務切割方案。

關鍵字：微服務、自然語言處理、大語言模型、資料存取、文字向量

1. 緒論

微服務架構因其高可擴展性、元件隔離能力以及圍繞業務核心的架構，獲得了廣泛關注，許多組織因此規劃將其單體式系統遷移至微服務架構，在遷移的過程中，切割與識別微服務是最重要的關鍵步驟。近年來，已有許多針對微服務的切割 (Cutting) 或識別 (Identification) 的學術方法被提出，其中包含基於業務流程分析[2]、資料存取模式策略[3]、基於功能模組[1]、根據不同使用者角色的操作需求等多種方式來進行識別[4]等。這些方法提供從多個視角進行分析的策略，有助於更全面地理解系統結構，而在選擇架構遷移策略時，仍然需要考慮具體目標應用程式之特性及需求。

在許多既有單體式應用程式中，文件與相關規格文本會因為多年應用程式更新，變得過時或不完整，因此原始碼為唯一能夠準確反映應用程式實際功能的資源，根據微服務架構之 Database Per Service 原則，服務存取之資料來源亦是辨識微服務的重要依據，能有效降低服務間的耦合性 (Coupling)。基於上述分析，本研究規劃運用大型語言模型 (Large Language Model, LLM)，分析原始碼語意 (Code Semantics) 及其文字向量 (Text Embeddings)，並考量存取資料來源 (Data Source)，以能進行微服務之識別與切割。本研究希望能為系統架構現代化(Architecture Modernization)提供可行且有效的方法，能順利將系統從單體遷移至微服務架構，進而提升整體系統的可延展性和可維護性。

2. 研究方法

本研究提出了基於原始碼文字向量與資料存取分析的微服務識別方法－MICDA (Microservice Identification based on Code semantics and Data Access)。MICDA 方法流程如圖 1 所示，首先 MICDA 透過 JavaParser 識別服務端點，並收集相關資訊。接著，MICDA 使用 OpenAI API 生成文本摘要及多種分群建議。最後，MICDA 將文本轉換為向量，計算本研究提出平均群內相似度指標 (AIS)，以根據 AIS 值排序產出最終建議。MICDA 方法在進行微服務辨識時，除了使用文字向量外，還整合了資料存取模式分析與微服務設計原則，主要因為單純依賴向量分群無法反映資料依賴關係和不同的服務分群考量。

MICDA 以 Java Spring Boot 實作，並使用 Swagger UI 生成互動式網頁介面，透過 JavaParser 工具進行原始碼靜態分析，以識別專案中的服務端點，並使用 GPT 生成摘要與文字向量技術。

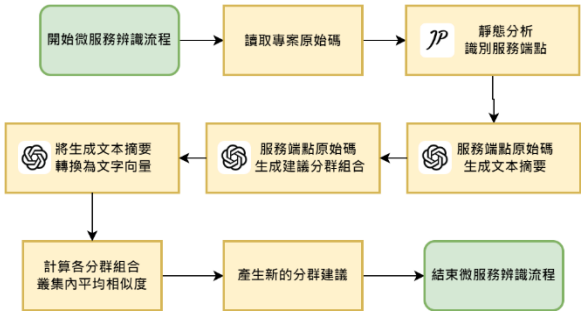

圖 1. MICDA 識別方法流程

2.1. 識別服務端點

服務端點通常定義於控制器層，且包含註解。本研究使用 JavaParser 解析 Java 原始碼並生成抽象語法樹 (AST: abstract syntax tree)，從巡訪語法樹的過程中，識別帶有 HTTP 註解的方法。

2.2. 收集服務端點資訊

成功識別服務端點後，接著分析每個服務端點所調用的服務層方法，本階段同樣使用 JavaParser 巡訪控制器層與服務層的原始碼，結合 Visitor 模式收集與紀錄各服務端點的名稱、HTTP 路徑、HTTP 方法、參數、回傳型態、調用服務名稱、方法名稱及資料來源(data source)等，這些原始資料會先儲存下來，用於後續分析（本研究目前尚未使用 HTTP 路徑、HTTP 方法與回傳型態）。其中，端點名稱與資料來源會做為後續微服務識別的特徵資訊。由於本研究目前主要支援 Spring Boot 框架，因此資料來源即為 Repository 名稱，是提供各服務端點之 Controller 所調用之 Service 其連接的 Repository。

2.3. 服務端點原始碼，進行文本摘要生成

MICDA 使用 OpenAI 的 Chat Completions API[15]生成文本摘要，透過自然語言處理技術解析服務端點的原始碼，提取關鍵的語意資訊，接著透過 OpenAI 的 Prompt Template 將收集到的服務端點資訊生成其他兩個特徵資訊：關鍵字 (Suggested keywords) 與業務邏輯 (Business logic)。

在 OpenAI 的 Chat Completions API 使用 Prompt Template 時，會以 **role** 區分對話中的不同角色，此設定能指引 GPT 模型生成更為準確且相關的回應。MICDA 主要使用 **System** 與 **User** 角色，實際應用如表 1 所示，首先以 User 提供單一服務端點原始碼片段，並於 System 設定模型將以軟體工程研究員身分來理解與回應後續指令，接著由 User 輸入明確指令，要求生成格式化的摘要。其中生成式 AI 可根據程式碼摘要出業務邏輯，可作為後續的識別流程重要的分群依據。

表 1. 單一服務端點生成文本的提示工程

Role	Content
User	"### Code: [endpointSourceCode] ###"
System	"You are a software engineering researcher."
User	"Please respond with the following formatted summary, including only the answer content without repeating the question description."
User	"1-Endpoint name: 2-Suggested keywords (Word limit: 2-3 words or fewer): 3-Business logic (Word limit: 10-30 words or fewer):"

2.4. 服務端點原始碼，生成建議分群組合

MICDA 會將該專案的所有服務端點的四個特徵資訊(端點名稱、資料來源、關鍵字、業務邏輯)提供給 GPT 模型，並基於下列五種微服務分群概念生成分群建議[22]，進而優化微服務識別、設計流程與運行效率。MICDA 採用分群概念包含：

- 基於功能模組 (Functionality-Based)：如[5]之研究建議，可依據應用的功能模組劃分微服務，並將相關功能邏輯集中在一個服務中。此方式可提高內聚性與可維護性，適用於明確劃分功能模組的系統。
- 基於領域驅動設計 (Domain-Driven Design, DDD)：如[2, 9]之研究建議，從業務領域的角度設計系統，可將系統劃分為不同的領域模型，每個模型代表系統的部分業務邏輯。此方法強調設計與業務需求的高度一致性，領域邊界明確。
- 基於資料所有權 (Data Partitioning)：如[8]之研究建議，可按照資料分區劃分微服務，每個服務負責特定資料的儲存和管理。該方法提高了資料存取效率與系統效能，適合需要處理大規模資料數據的系統。

- 基於功能介面 (Interface-Based)：如[6]之研究建議，可依據系統內不同功能模組之間的介面劃分微服務，確保每個服務皆能透過明確定義的介面進行溝通與合作。該方法提高了系統的模組化與可測試性，適合需要強調服務之間鬆散耦合的系統。
- 基於使用者角色 (User Role-Based)：如[4, 10]之研究建議，若可依據系統中的不同使用者角色來劃分微服務，亦是一個合理的切分方式，可確保每個服務針對特定的使用者角色提供適切的功能。此方法適合擁有多種使用者角色的系統。

2.5. 將服務端點資訊，轉換為文字向量

MICDA 經由 OpenAI 的 Embeddings API 進行文字向量轉換。首先會將每個服務端點的文本摘要作為輸入，接著調用 OpenAI Embedding API [15]獲取對應的文字向量並將其儲存，作為後續的相似度計算與微服務識別分析使用。

2.6. 計算各分群組合，平均群內相似度

本研究提出了群內相似度指標 (AIS, Average intra-cluster Similarity)，如公式(1)，用來衡量同一分群服務端點彼此間的平均相似度。找出各個分群的所有向量後，開始計算各服務端點與其他服務端點的平均值，並作為該分群的平均相似度。接著計算所有平均群內相似度，作為該分群方式的總體平均相似度 AIS。AIS 值高低表示該分群的服務端點相似程度，從而評估分群品質與提供分群建議排序。

$$Average\ Intra-Cluster\ Similarity = \frac{1}{|C| \times (|C|-1)} \sum_{i,j \in C, i \neq j} sim(i,j) \quad (1)$$

2.7. 產生新的分群建議

藉由計算各個分群組合的 AIS 值並以此指標進行排序，我們能夠有效評估每個建議分群組合的效能，並依據提出的五種不同的分群概念，提供最適當的分群建議。

3. 實驗與分析

本研究於 GitHub 取得 6 個開源專案與相關論文中所提及的微服務識別案例，使用 MICDA。首先取得各專案其建議分群組合排序，及 AIS 指數資料，再結合調整蘭德指數 (Adjusted Rand Index, ARI) [11]以及肯德爾相關係數 (Kendall's tau) [12]，用以衡量與驗證微服務分群方法的有效性。ARI 與 Kendall's tau 指標之核心概念分述如下：

- 調整蘭德指數 (Adjusted Rand Index, ARI)：ARI [11]是一種群聚外部評估指標，用於對蘭德指數進行調整，以消除隨機群聚的影響。ARI 的值範圍介於-1 到 1 之間，數值愈高表示群聚結果愈好。該指標用於衡量群聚結果與真實標籤之間的一致性，能夠有效地評估群聚方法的準確性和穩定性。
- 肯德爾相關係數(Kendall's tau)：Kendall's tau [12]是一種用於衡量兩個變數間相關性的統計量。主要用於評估排序資料 (Ordinal Data) 或名次資料 (Ranked Data) 之間的相關性。Kendall's tau 基於排列對之間的一致性來計算，與 Spearman's rho 相關係數相似，但計算方法有所不同。該指標可以幫助我們評估 AIS 指數排序與 ARI 指數排序之間的相關性。

3.1. 實驗設計

為評估本研究提出的微服務識別方法，並觀察其在不同專案中的適用性。實驗步驟如下：

1. 收集資料：收集基於 Java 與 Spring Boot 框架開發的 GitHub 專案，共選取 6 個專案應用 MICDA 進行分析。
2. 生成分群建議：對每個專案使用 MICDA，各別生成前章節提及的五種分群組合建議，並計算每個分群的 AIS 指數，作為推薦指標依據。此外，為能進行本方法的有效性評估，我們亦設計單純的 GPT 識別方案：直接請 GPT 讀取專案程式碼去生成分群。
3. 人工分群：我們會對每個專案進行人工分群，作為基準分群結果。人工分群考慮的準則包括資料來源、領域的區隔、服務大小的合適程度、功能內聚性、接口定義的清晰性及效能和擴展性需求等綜合因素判斷。

4. 計算 ARI 指數：使用調整蘭德指數評估每個專案的推薦分群與基準分群之間的一致性。
5. 觀察肯德爾相關係數：將 AIS 指數與 ARI 指數排序，計算兩者之間的肯德爾相關係數 (Kendall's tau)，並分析其相關性。
6. 綜合權重計算：在進行微服務分群時，我們將對應四種不同資料來源的文字向量(如表 2)，利用[0.25, 0.25, 0.25, 0.25]作為綜合權重計算最終結果。
7. 比較 GPT 直接分群效果：為進一步驗證 MICDA 方法，本研究使用 GPT 模型直接生成的分群結果作為比較對象。將未經預處理的程式碼直接提交給 GPT，比較 GPT 分群結果、人工分群與 MICDA 之間的差異，並評估 GPT 於微服務架構分析中的實際應用效用。

表 2. 權重計算資料

EN	對應 Endpoint Name 資料
SK	對應 Suggested Keywords 資料
BL	對應 Business Logic 資料
RI	對應 Data Source-Repository Info 資料
AWE	對應 Aggregated Text Embeddings 資料

3.2. 實驗案例探討與分析

本實驗將使用 MICDA 於六個 GitHub 專案 E-Commerce [16]、QuizZz [17]、Blogs [18]、five-six [19]、E-Book [20]、SportShop [21]進行微服務識別分析。因篇幅關係，底下僅詳細說明兩個代表性案例：E-Commerce 與 QuizZz 之實驗結果。

3.2.1. 案例 1：E-Commerce

E-Commerce 是一個電子商務系統，主要提供使用者身份驗證、產品管理、訂單處理等功能，總共有 32 個服務端點。

實驗結果如表 3，可得到以下推薦優先順序：

1. Functionality-Based、DDD 與 Interface-Based (AIS = 0.848)
2. Data Partitioning (AIS = 0.8277)
3. User Role-Based (AIS = 0.82)

根據實驗結果，我們建議優先採用 AIS 與 ARI 表現較好的三種方法，表現較差的兩種方法相較之下可能不適用於電子商務系統的微服務分群。

表 3. E-Commerce 各項群聚評估指標

A_E-Commerce		True Cluster	Functionality Based	DDD	Data Partitioning	Interface Based	User Role Based	GPT
ED	AS	0.8566	0.8365	0.8365	0.8092	0.8365	0.8028	0.8341
	ARI	1	0.6702	0.6702	0.4593	0.6702	0.2579	0.5451
	Kendall's tau (Excl. GPT)/(Incl. GPT)	1	1					
	P-value (Excl. GPT)/(Incl. GPT)	0.0028	0.0004					
SK	AS	0.8718	0.8558	0.8558	0.8352	0.8558	0.8248	0.8528
	ARI	1	0.6702	0.6702	0.4593	0.6702	0.2579	0.5451
	Kendall's tau (Excl. GPT)/(Incl. GPT)	1	1					
	P-value (Excl. GPT)/(Incl. GPT)	0.0028	0.0004					
BL	AS	0.7846	0.7635	0.7635	0.7487	0.7635	0.7315	0.762
	ARI	1	0.6702	0.6702	0.4593	0.6702	0.2579	0.5451
	Kendall's tau (Excl. GPT)/(Incl. GPT)	1	1					
	P-value (Excl. GPT)/(Incl. GPT)	0.0028	0.0004					
RI	AS	0.9551	0.9361	0.9361	0.9177	0.9361	0.9209	0.9399
	ARI	1	0.6702	0.6702	0.4593	0.6702	0.2579	0.5451
	Kendall's tau (Excl. GPT)/(Incl. GPT)	0.8667	0.619					
	P-value (Excl. GPT)/(Incl. GPT)	0.0167	0.069					
AWE	AS	0.867	0.848	0.848	0.8277	0.848	0.82	0.8472
	ARI	1	0.6702	0.6702	0.4593	0.6702	0.2579	0.5451
	Kendall's tau (Excl. GPT)/(Incl. GPT)	1	1					
	P-value (Excl. GPT)/(Incl. GPT)	0.0028	0.0004					

3.2.2. 案例 2：QuizZz

QuizZz 是一個測驗應用程式，使用者可以在其中建立與發布測驗。此專案共有 54 個服務端點，實驗結果如表 4，可得出以下推薦優先順序：

1. User Role-Based (AIS = 0.8515)
2. Functionality-Based (AIS = 0.8472)
3. DDD (AIS = 0.8464)
4. Data Partitioning, Interface-Based (AIS= 0.8406)

根據實驗結果，我們建議優先採用 User Role-Based 分群方法，其次為 Functionality-Based 和 DDD，這三種方法在 AIS 與 ARI 指數表現較好。表現較差的兩種方法相較之下可能不適用此測驗應用程式的微服務分群。

表 4. QuizZz 各項群聚評估指標

B_QuizZz		True Cluster	Functionality Based	DDD	Data Partitioning	Interface Based	User Role Based	GPT
ED	AS	0.8313	0.8313	0.8342	0.8161	0.8161	0.8349	0.8143
	ARI	1	1	0.7854	0.6558	0.6558	0.8798	0.0921
	Kendall's tau (Excl. GPT)/(Incl. GPT)	0.4667	0.619					
	P-value (Excl. GPT)/(Incl. GPT)	0.2722	0.069					
SK	AS	0.8571	0.8571	0.8472	0.8551	0.8551	0.8573	0.8387
	ARI	1	1	0.7854	0.6558	0.6558	0.8798	0.0921
	Kendall's tau (Excl. GPT)/(Incl. GPT)	0.4667	0.619					
	P-value (Excl. GPT)/(Incl. GPT)	0.2722	0.069					
BL	AS	0.7914	0.7914	0.7768	0.7897	0.7897	0.791	0.7676
	ARI	1	1	0.7854	0.6558	0.6558	0.8798	0.0921
	Kendall's tau (Excl. GPT)/(Incl. GPT)	0.7333	0.8095					
	P-value (Excl. GPT)/(Incl. GPT)	0.0556	0.0107					
RI	AS	0.9089	0.9089	0.9273	0.9017	0.9017	0.9227	0.8966
	ARI	1	1	0.7854	0.6558	0.6558	0.8798	0.0921
	Kendall's tau (Excl. GPT)/(Incl. GPT)	0.3333	0.5238					
	P-value (Excl. GPT)/(Incl. GPT)	0.4694	0.1361					
AWE	AS	0.8472	0.8472	0.8464	0.8406	0.8406	0.8515	0.8293
	ARI	1	1	0.7854	0.6558	0.6558	0.8798	0.0921
	Kendall's tau (Excl. GPT)/(Incl. GPT)	0.7333	0.8095					
	P-value (Excl. GPT)/(Incl. GPT)	0.0556	0.0107					

3.3. 整體實驗結果分析

本研究將 MICDA 應用於六個 GitHub 上的專案，並透過具體的評估指標來驗證方法是否有效。

實驗結果顯示，將 AIS 指數與 ARI 指數進行排序後，計算兩者之間的 Kendall's tau 係數，結果呈現正相關，以此證明本研究所提出的 MICDA 在微服務識別中的成效。

1. AIS 指數
   - Functionality-Based、DDD 與 Interface-Based：三種方法的平均相似度在所有案例均相對較高，表示其在保持服務內部一致性的優勢。
   - Data Partitioning 與 User Role-Based：兩種方法的平均相似度相對較低，可顯示其在內聚性方面有一定程度的不足。
2. ARI 指數
   - Functionality-Based、DDD 與 Interface-Based：三種方法的 ARI 分數表現同樣出色，顯示了在準確劃分服務方面的可靠性。
   - Data Partitioning 與 User Role-Based：兩種方法的 ARI 分數較低，尤其是 User Role-Based 方法，顯示其在準確劃分服務方面存在較大挑戰。

此外，觀察 GPT 直接分群結果，雖然 AIS 指數表現與人工分群有一定程度相同，就結果而言，GPT 在未經特別訓練的情形下仍需進一步優化，以此提高在微服務識別中的準確性。

綜上所述，MICDA 能夠有效識別微服務，結果也顯示在部分情況下，Functionality-Based、DDD 與 Interface-Based 方法是較合適的分群策略。

3.4. 與既有方法之比較

CoCoME (Common Component Modeling Example)是一個連鎖超市的銷售交易系統，主要提供銷售與進貨商品管理，該系統之核心功能包括 9 個服務端點，涵蓋從產品銷售到庫存管理等功能。在文獻[7, 13]中，分別提出了不同的微服務識別方法，以下為此兩種方法之概要：

- 基於資料庫存取分析的微服務遷移方法(AMIADA, Microservices Identification using Analysis for Database Access)：該方法透過監控每個服務端點對資料庫的存取情況，並利用階層式分群對收集到的資料進行分析，從而確定微服務的最佳分群方式。
- 功能分解(Functional Decomposition)：透過分析系統功能需求，將系統分解為高內聚性和低耦合度的微服務。此方法能在系統演化過程保持高質量的設計結構，減少維護和擴展的複雜性。

上述兩種方法各有優勢，Functional Decomposition 適合系統設計初期進行微服務識別，而 AMIADA 適合現有系統的遷移過程，上述兩篇文獻中均採用了 CoCoME 作為案例，為進一步分析，我們將 CoCoME 應用於 MICDA，並將結果與兩個文獻提出的方法進行比較，如表 5。從實驗結果可以看到，MICDA 方法於 AIS 與 ARI 指標都能取得較好的表現，可印證 MICDA 的確具有良好的微服務識別效果。

表 5. CoCoME 系統微服務識別方法比較

AS	ARI	識別方法	分群數量	serviceID	endpoints
0.8899	0.7692	MICDA	5	1	Create Delivery Report Update Delivery Report
				2	Order Product Receive Ordered Product
				3	Buy Product Change Product Price
				4	Sell Product
				5	Create Stock Report Update Stock Report
0.8808	0.6	AMIADA	4	1	Create Delivery Report Update Delivery Report
				2	Buy Product Change Product Price
				3	Receive Ordered Product Order Product
				4	Sell Product Create Stock Report Update Stock Report
0.8765	0.4706	Functional Decomposition	4	1	Create Delivery Report Update Delivery Report
				2	Create Stock Report Update Stock Report Order Product Receive Ordered Product
				3	Sell Product Buy Product
				4	Change Product Price

4. 結論

本研究提出基於原始碼文字向量與資料存取分析的微服務識別方法，成功提取語意摘要與資料，並實現自動化識別微服務分群，為使用者提供微服務分群參考，而實驗結果顯示，本研究提出之 MICDA 方法相較於先前方法，的確有較好的識別成效。在研究限制上，本方法目前僅支援以 Spring Boot 專案做為分析目標，且文本需遵守有意義的程式碼命名規則方可有好的識別效果；此外，大型語言模型在處理複雜或模棱兩可的語境時，可能會出錯而導致結果不準確，因此仍需要在人工監控和校驗下使用這些技術。

本研究未來的規劃如下：(1) 原始碼分析：提

高原始碼其他資訊應用率，探索如介面 (Interface) 與實體 (Entity) 等資訊的利用，以增強方法的領域調適性和服務辨識效果。(2) 多語言開發：目前僅能分析 Spring Boot 專案，未來應擴展至其他程式語言與框架，探索如何將本方法應用於不同語言與架構。(3) 模型訓練：針對不同系統的差異性，進行相似專案的訓練，研究是否能透過此方式提高分群的精準度和效果。(4) 動態分析技術：結合動態分析技術，以捕捉運行時的資料傳遞與依賴關係，提供更全面與準確的微服務識別結果。

致謝

本研究接受科技部編號：112-2221-E-019-021-MY3 研究計畫經費補助，特此感謝。

參考文獻

[1] S. Newman, Building microservices. " O'Reilly Media, Inc.", 2021.

[2] E. Evans, Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional, 2004.

[3] J. Lewis and M. Fowler, "Microservices: a definition of this new architectural term (2014)," *URL:http://martinfowler.com/articles/microservices. html (cit. on p. 26),* 2014.

[4] C. Richardson, *Microservices patterns: with examples in Java*. Simon and Schuster, 2018.

[5] G. Mazlami, J. Cito, and P. Leitner, "Extraction of microservices from monolithic software architectures," in *2017 IEEE International Conference on Web Services (ICWS)*, 2017: IEEE, pp. 524-531.

[6] M. Gysel, L. Kölbener, W. Giersche, and O. Zimmermann, "Service cutter: A systematic approach to service decomposition," in *Service-Oriented and Cloud Computing: 5th IFIP WG 2.14 European Conference, ESOCC 2016, Vienna, Austria, September 5-7, 2016, Proceedings 5*, 2016: Springer, pp. 185-200.

[7] S.-P. Ma, T.-W. Lu, and C.-C. Li, "Migrating monoliths to microservices based on the analysis of database access requests," in *2022 IEEE international conference on service-oriented system engineering (SOSE)*, 2022: IEEE, pp. 11-18.

[8] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Software,* vol. 35, no. 3, pp. 24-35, 2018.

[9] N. Dragoni *et al.*, "Microservices: yesterday, today, and tomorrow," *Present and ulterior software engineering,* pp. 195-216, 2017.

[10] J. Fritzsch, J. Bogner, A. Zimmermann, and S. Wagner, "From monolith to microservices: A classification of refactoring approaches," in *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers 1*, 2019: Springer, pp. 128-141.

[11] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association,* vol. 66, no. 336, pp. 846-850, 1971.

[12] H. Abdi, "The Kendall rank correlation coefficient," *Encyclopedia of measurement and statistics,* vol. 2, pp. 508-510, 2007.

[13] S. Tyszberowicz, R. Heinrich, B. Liu, and Z. Liu, "Identifying microservices using functional decomposition," in *Dependable Software Engineering. Theories, Tools, and Applications: 4th International Symposium, SETTA 2018, Beijing, China, September 4-6, 2018, Proceedings 4*, 2018: Springer, pp. 50-65.

[14] OpenAI. (2024). *Chat completions API* (Version 1.0). https://platform.openai.com/docs/api-reference/chat

[15] OpenAI. (2024). *Embedding API* (Version 1.0). https://platform.openai.com/docs/api-reference/chat

[16] Sirajuddin. (2022). *E-commerce application. GitHub.* https://github.com/Sirajuddin135/E-Commerce-Application

[17] J. Rodriguez (2017). *QuizZz*. GitHub. https://github.com/oojorgeoo89/QuizZz

[18] Osopromadze. (2018). *Spring Boot Blog REST API.*GitHub.https://github.com/osopromadze/Spring-Boot-Blog-REST-API

[19] xiaott-ahh. (2020). *five-six*. GitHub. https://github.com/xiaott-ahh/five-six

[20] F. Mahmud and H. Santika (2020). *E-BookShop—Spring Boot* [Computer software]. GitHub. https://github.com/foysal-mahmud/E-BookShop----Spring-boot

[21] Bavenka. (2016). *RESTful API SportShop*. GitHub. https://github.com/bavenka/RESTful-API-SportShop

[22] Hsieh, Chih-Yu (2024). MICDA-code.GitHub. https://github.com/fcebk17/MICDA-code