

DoDAF 產品資料元素之 CRUD 演算法設計

Design of CRUD algorithm for data elements in the DoDAF products

韓孟麒 曾淼泓 王正航

德明財經科技大學 資訊科技系 C4ISR 研究中心

{harn@takming.edu.tw tmh5735@mail2000.com.tw chwang.chris@gmail.com}

劉中宇

國防大學 理工學院 資訊工程系

cyliu7000@gmail.com

摘 要

本研究的目的，乃是對 DoDAF V1.5 的綜合觀點(All View, AV)、作戰觀點(Operational View, OV)、系統觀點(Systematic View, SV)與技術觀點(Technical View, TV)等所架構出的 26 項產品，律定出相關產資料元素(Data Elements)，並探討資料元素相互間的追溯特性。

我們於資料元素間尋獲相依關係後，若須更動 DoDAF 產品規格相關之資料元素，則必須對 DoDAF 後台資料庫的相關資料元素，作出建立(Create)、擷取(Retrieve)、更新(Update)及刪除(Delete)等動作，我們稱之為 CRUD 操作(CRUD Operation)，俾隨時保持 DoDAF 各資料元素的一致性(Consistency)與完整性(Complete)。

關鍵詞：DoDAF、綜合觀點(AV)、作戰觀點(OV)、系統觀點(SV)、技術觀點(TV)、CRUD 演算法

一、緒論

由於 DoDAF 的各產品間具有關連性(Dependency)，當企業架構師(Enterprise Architect)完成了所屬專案的 DoDAF 產品設計後，往往會因為甲方企業模式(Business Model)的改變，而導致要修改部分的 DoDAF 產品規格。在多變的環境裏，使用者需求(User Requirements)改變的現象相當頻繁，而且是一種常態，這種現象與常態，迫使企業架構師所設計系統必須去適應環境的變化。在修改的過程中，各個 DoDAF 觀點中的元件會受到連動，也會讓企業架構師，搞不清楚哪些觀點的相關資料元素會被修改。另由於 DoDAF 的 26 項產品間具有一致性與完整性，所以企業架構師在製作 DoDAF 產品時，會與甲方不斷地溝通，不斷地訪談，並來回多次的編輯 DoDAF 產品。在修改的過程中，常使各個觀點中的相關資料元素產生不一致與不完整的現象；因此，企業架構師會不斷地根據使用者需求與規格(Specification)，作確認(Validation)及驗證(Verification)的工作。

我們透過系統分析法(System Analysis Method)、正型方法(Formal Method)，在 DoDAF 各項產品製作的活動裏，進行歸納與分析，使製作過程系統化及理論化；我們將研究成果，著重在將

DoDAF 資料庫中的資料元素，隨時保持一致與完整。

當企業架構師面對 DoDAF 某項產品的需求與規格需要變動後，就必須追溯 26 項產品的關連性，這種追溯關係隱含著：作戰觀點(OV)、系統觀點(SV)與技術觀點(TV)各觀點內部產品間的橫向追溯；作戰觀點(OV)、系統觀點(SV)與技術觀點(TV)跨觀點產品間的縱向追溯及跨產品間具遞移性的間接追溯。我們惟有詳盡檢討，並發展一套追溯規則，才能讓後續之建構與維護工作得以持續進行；因此，我們必須透過各資料元素間的對應關係，來檢討各項產品間的追溯關係，並發展出追溯性矩陣來解決相關資料元素間不一致與不完整的問題。本論文聚焦在找到需更動的資料元素後，對 DoDAF 後台資料庫的各資料元素設計出 CRUD 演算法。

依照本研究所設計出來的 CRUD 演算法，將為爾後的 DoDAF 相關研究人員，對 DoDAF 後台資料庫的維護，提供一個效益強大的輔助工具。

二、文獻探討

IEEE 在 1994 年對「追溯性」(Traceability)的定義如下：

(一)「追溯性」係指在二個或更多軟體元件的發展過程中，軟體元件間相互建立連接關係的追溯，尤其該軟體元件相互間，存在著前後順序或主從關係的一種現象；例如，在設計一個軟體元件時，需求、規格與軟體元件間的相互追溯。

(二)「追溯性」係指在軟體元件的發展過程中，各軟體元件會針對其存在，而追溯其設計理由；例如，在追溯氣泡圖(Traceability Bubble Diagram)中，滿足每一軟體元件相互參考的程度，就是軟體元件的「追溯性」[1]。

各軟體元件間的追溯性連接，直到目前為止，仍很少被討論到[2][3]；因為，各軟體元件它們在軟體的發展及維護過程中，必須被確認及維護。軟體元件確認及維護的工作是相當耗時的，且在正常的發展過程中，因時間的壓力及成本的精算下，常被省略掉[4]。一直到最近幾年，「追溯性」在軟體工程上，才廣泛的被認知；而在支援追溯性上的資

料存取，是必需運用工具及專業技巧才能達成的。也就是說，在不同型態的軟體元件間，我們想要得到某種結果，它是必需運用追溯性的連接，及資料存取的技巧，才能獲得[5]。

在軟體元件追溯性上的研究，經文獻考證，已有眾多學者投入此領域，簡述如后。

Bonita 等人[6]，發展一套方法，他們的方法是在不同版本的原始碼間，產生追溯連結的預測，並使用差異辨識的技術去處理各追溯連結。

C. H. Jane 等人[7][8]，使用事件服務(Event Service)的特性來建立連接。而事件服務的各種特性，是透過使用者需求來推動的，當某項使用者需求產生變化時，被連接的軟體元件，就會接收到更改的訊息。事件的維護人員，不斷的對新事件清單的每一事件作研究，並對所發生改變之雙方或多方，作出告知。此方法不需即時對連接之彼此雙方作出告知，連接的確認，可由使用者在任何時間中提出。此為以事件為導向(Event-based)的方法，透過時間來擷取相互間的存在關係，如果需要很快得知其連接關係，那就得透過工具來完成了。

Sherba 等人[9]，在時間的基礎上，對所提供服務的沿革，分析其一系列的改變關係，並發表追溯框架(Traceability Framework)，在不同時間所發展的新舊版本服務中，分析現有的連接關係。

A. V. Knehen[8]，提出一個差異觀念的追蹤模式，並以此決定該被追蹤軟體元件之型態，俾支援使用者需求的改變及衝突分析。

Murta 等人[10]，發展出一套工具，謂為拱門追蹤(ArchTrace)；在架構及執行上，採取以策略為中心的方法，來自動更新軟體元件間連接上的追溯。在每一次使用者需求的改變之後，其軟體元件間連接上的異動關係，即被記錄起來；各軟體元件間的相互連接內容，皆被存入檔案並持續維護。在他們的研究中，因為接觸到夠精細的差異辨識，所以也有連接到其他相關的檔案，以維持研究的繼續進行。

Riebisch 介紹特徵模式(Feature Models) [11]，並由它在各軟體元件間，扮演建構追溯連接上的介質。

Youngki 等人[12]，創造了實用簡單的追溯性工具，來解決自需求到系統各軟體元件間的追溯性連結。Youngki 等人使用相關存取的技术，去處理並儲存不同軟體元件間的連接資訊，然後使用所發展的差異辨識方法去維護它們。

在一個軟體系統開發的計畫中，使用者需求的追溯，在整個開發過程中，提供了一個特定的支援工作。無論其執行過程是如何的複雜，以長期維護的角度來說，其所產生的傳統追溯矩陣，一再證明了追溯性的困難度。而動態存取方法雖然降低了維護上必需的創造及連接工作，並且減少了耗費在傳統追溯矩陣中手動追溯的投資[13]。

需求追溯性矩陣(Requirement Traceability Matrix, RTM)是一種靜態的需求追溯方法，現亦廣

泛地被運用在各項軟體專案中。任何一個廣泛的科學任務計畫，皆須描述該任務的重要性及如何的執行，此可以需求追溯性矩陣來表達；而需求追溯性矩陣，已成為所有 NASA 的任務計畫中，不可或缺的工具[14]。

三、開放性問題探討

對 DoDAF 產品的探討，具有以下幾點目前存在的開放性問題：

1. DoDAF 產品具有一致性(Consistency)與完整性(Complete)：企業架構師在製作 DoDAF 產品時，與甲方不斷地溝通，不斷地訪談；因此，在來回多次的編輯中，常使各個相關觀點中的元件產生不一致(Inconsistency)與不完整(Incomplete)的現象。所以，企業架構師會不斷地根據使用者需求與規格，作確認(Validation)及驗證(Verification)的工作。在探討一致性及完整性的過程中，卻無法去驗證其正確性。
2. DoDAF 產品的製作過程中，缺少了導引地圖(Guidance Map)：目前著名的 DoDAF 製作工具有：IBM 的 SA(System Architecture)、Rhapsody 與 Sparx System 的 EA 等工具，雖然所提供各產品製作方法不同，皆只有扮演工具的提供，沒有對產品發展的全過程順序，提出建議及製作導引。
3. 在目前市面上使用之軟體，如：IBM 的 SA(System Architecture)、Rhapsody 與 Sparx System 的 EA 等工具，已具有追溯之功能。企業架構師可以直接在實體設計上作出增加、刪除、修改之動作。例如：因為 DoDAF 產品之相依性大，所以在 OV-2 直接作出刪除某作戰節點之動作時，系統或許允許企業架構師直接刪除之；但此作戰節點的刪除，在對該產品的相關資料元素「一致性」檢核時，似是無誤，卻會影響到 OV-4 組織關係圖、OV-6c 循序圖的相關資料元素，並使 OV-4 及 OV-6c 失去意義。我們只有透過人工之檢核，才能了解問題之嚴重性，並在邏輯概念上作出修正。

美軍在 DoDAF 的產品發展上，並未強制採用那一種方法論來進行架構發展。為了完成 DoDAF 的產品製作，經考證後，其發展方法主要有：應用結構化分析(Structured Analysis, SA)技術[15][16]、物件導向(Object-Oriented, OO)技術[16]及活動為基礎方法論(Activity Based Methodology, ABM)[17]三種方法。結構化分析技術對架構產品的開發，主要是由作戰概念開始，並由上而下對作戰流程進行分解，來建立架構描述。物件導向技術方法，是利用 UML 進行架構描述，來建立架構開發的程序。活動為基礎方法論(ABM)，是一種半自動化的 DoDAF 架構開發，此方法以作戰活動為基礎，開發出所有的架構產品。

針對軟體發展過程及 DoDAF 產品的追溯性研究，我們體會到如下幾點心得：

1. 在各項軟體系統發展計畫中，需求的管理及彼此間的追溯性是複雜的，尤其是 DoDAF 各產品間的追溯性；惟把握專案內的各項特徵，把資源投資到追溯的過程，將可自不同的需求追溯中得到好處[18]。
2. 探討需求的追溯性矩陣，為軟體開發人員經常使用到的工具，在支援需求追溯性(Requirement Traceability)的過程，一個適當的 CASE 工具是需要被發展出來的[19][20][21][22]。
3. 現有商業化的軟體元件追溯工具，皆無法在各軟體元件間的定義、運用過程中，顯著的建立其追溯性的機制[23]。
4. 透過追溯性機制的探討後，得到 DoDAF 產品發展中，因甲方需求改變而連帶需到 DoDAF 系統的後台資料庫，修改各相依性產品規格。

本論文即針對此現象，透過 DoDAF 產品的追溯性，使用追溯性矩陣，於尋獲相依性需變動的資料元素後，對現存 DoDAF 系統的後台資料庫提出一套資料異動的 CRUD 演算法。

四、DoDAF 各產品資料元素之定義

在對 DoDAF 產品展開追溯性及 CRUD 動作之探討前，我們先對 DoDAF 各產品之定義如下：

Definition 1 (DoD Architecture Framework) A DoD Architecture Framework is a tuple $DoDAF = (AV, OV, SV, TV)$ where

1. AV is a set of all views for architecting a C4ISR system,
2. OV is a set of *operational* views for architecting a C4ISR system,
3. SV is a set of *systematic* views for architecting a C4ISR system,
4. TV is a set of *technical* views for architecting a C4ISR system.

【定義說明】

DoDAF 架構框架係由 DoDAF 的組合 $DoDAF = (AV, OV, SV, TV)$ 所組成，其相關符號說明如下：AV 係由綜合觀點所組成的集合，目的在架構 C4ISR 系統；OV 係由作戰觀點所組成的集合，目的在架構 C4ISR 系統；SV 係由系統觀點所組成的集合，目的在架構 C4ISR 系統；TV 係由技術觀點所組成的集合，目的在架構 C4ISR 系統。

Definition 2 (All View) An all view is a set of architectures $AV = AV-1 \cup AV-2$ where

1. AV-1 is a set of views for architecting the overview and summary information by using *texts* or *tables*, and
2. AV-2 is a set of views for architecting the integrated dictionary by using a structured tool: *data dictionary*, or *tables*.

【定義說明】

綜合觀點 (All View) 係由架構 AV 的集合所

組成， $AV = AV-1 \cup AV-2$ ，其相關符號說明如下：AV-1 係由觀點所成的集合，目的在架構綜合觀點及整合資訊，使用的工具為本文(Text)或表格(Tables)及其屬性；AV-2 係由觀點所成的集合，目的在架構綜合觀點及整合字典，使用的工具為結構化工具(Structured Tool)，包括了資料字典(Data Dictionary)或表格(Tables)及其屬性。

Definition 3 (Operation View) An operation view is a set of architectures $OV = OV-1 \cup OV-2 \cup OV-3 \cup OV-4 \cup OV-5 \cup OV-6a \cup OV-6b \cup OV-6c \cup OV-7$ where

1. OV-1 is a set of views for architecting the high-level operational concept by using *graphics*, *use-case charts* or *deployment charts* with *properties*,
2. OV-2 is a set of views for architecting the operational node connectivity by using *deployment charts* or *data flow diagrams* with *properties*,
3. OV-3 is a set of views for architecting the operational information exchange by using *matrixes* with *properties*,
4. OV-4 is a set of views for architecting the organizational relationships by using *class charts* or *hierarchical trees* with *properties*,
5. OV-5 is a set of views for architecting the operational activity model by using *use-case charts*, *collaboration charts*, *sequence charts*, *activity charts* or *IDEF0* with *properties*,
6. OV-6a is a set of views for architecting the operational rules model by using *decision rules* or *IDEF3* with *properties*,
7. OV-6b is a set of views for architecting the operational state transition by using *state charts* with *properties*,
8. OV-6c is a set of views for architecting the operational event trace by using *sequence charts* with *properties*, and
9. OV-7 is a set of views for architecting the logical data model by using *class charts* or *entity-relationship diagrams* with *properties*.

【定義說明】

作戰觀點 (Operation View) 係由架構 OV 的集合所組成， $OV = OV-1 \cup OV-2 \cup OV-3 \cup OV-4 \cup OV-5 \cup OV-6a \cup OV-6b \cup OV-6c \cup OV-7$ ，其相關符號說明如下：OV-1 係由觀點所成的集合，目的在架構高階概念圖，使用的工具為圖形(Graphics)或案例圖(Use-case Charts)或部署圖(Deployment Charts)及其屬性；OV-2 係由觀點所成的集合，目的在架構作戰節點的連接，使用的工具為部署圖(Deployment Charts)或資料流程圖(Data Flow Diagrams)及其屬性；OV-3 係由觀點所成的集合，目的在架構作戰資訊的交換，使用的工具為矩陣(Matrixes)及其屬性；OV-4 係由觀點所成的集合，目的在架構組織的關係，使用的工具為類別圖(Class Charts)或層級樹(Hierarchical Trees)及其屬

性；OV-5 係由觀點所成的集合，目的在架構作戰活動模式，使用的工具為案例圖(Use-case Charts)或合作圖(Collaboration Charts)或循序圖(Sequence Charts)或活動圖(Activity Charts)或 IDEF0 及其屬性；OV-6a 係由觀點所成的集合，目的在架構作戰規則模式，使用的工具為決策規則(Decision Rules)或 IDEF3 及其屬性；OV-6b 係由觀點所成的集合，目的在架構作戰狀態的轉換，使用的工具為狀態圖(State Charts)及其屬性；OV-6c 係由觀點所成的集合，目的在架構作戰事件的軌跡，使用的工具為循序圖(Sequence Charts)及其屬性；OV-7 係由觀點所成的集合，目的在架構作戰事件的邏輯資料模式，使用的工具為類別圖(Class Charts)或 E-R 圖(Entity-Relationship Diagrams) 及其屬性。

Definition 4 (System View) A system view is a set of architectures $SV = SV-1 \cup SV-2 \cup SV-3 \cup SV-4 \cup SV-5 \cup SV-6 \cup SV-7 \cup SV-8 \cup SV-9 \cup SV-10a \cup SV-10b \cup SV-10c \cup SV-11$ where

1. SV-1 is a set of views for architecting the systems interface by using *deployment charts* or *component charts* with *properties*,
2. SV-2 is a set of views for architecting the systems communications by using *deployment charts* or *component charts* with *properties*,
3. SV-3 is a set of views for architecting the systems-systems matrix by using *matrixes* with *properties*,
4. SV-4 is a set of views for architecting the systems functionality by using *class charts*, *object charts*, *activity charts*, *data flow diagrams* or *hierarchical trees* with *properties*,
5. SV-5 is a set of views for architecting the operational activity to systems function traceability by using *matrixes* with *properties*,
6. SV-6 is a set of views for architecting the systems data exchange by using *matrixes* with *properties*,
7. SV-7 is a set of views for architecting the systems performance parameters by using *matrixes* with *properties*,
8. SV-8 is a set of views for architecting the systems evolution by using *graphics* with *properties*,
9. SV-9 is a set of views for architecting the systems technology forecast by using *tables* with *properties*,
10. SV-10a is a set of views for architecting the systems rules model by using *decision rules* or *IDEF3* with *properties*,
11. SV-10b is a set of views for architecting the system state transition by using *state charts* with *properties*,
12. SV-10c is a set of views for architecting the system event trace by using *sequence chart* with *properties*, and
13. SV-11 is a set of views for architecting the physical schema by using *component charts* or *entity-relationship diagrams* with *properties*.

【定義說明】

系統觀點 (System View)係由架構 SV 的集合所組成， $SV = SV-1 \cup SV-2 \cup SV-3 \cup SV-4 \cup SV-5 \cup SV-6 \cup SV-7 \cup SV-8 \cup SV-9 \cup SV-10a \cup SV-10b \cup SV-10c \cup SV-11$ ，其相關符號說明如下：SV-1 係由觀點所成的集合，目的在架構系統介面，使用的工具為元件圖(Component Charts)或部署圖(Deployment Charts)及其屬性；SV-2 係由觀點所成的集合，目的在架構系統通訊，使用的工具為元件圖(Component Charts)或部署圖(Deployment Charts)及其屬性；SV-3 係由觀點所成的集合，目的在架構系統對系統的關係，使用的工具為矩陣(Matrixes)及其屬性；SV-4 係由觀點所成的集合，目的在架構系統功能，使用的工具為類別圖(Class Charts)或物件圖(Object Charts)或活動圖(Activity Charts)或資料流程圖(Data Flow Diagrams)或層級樹(Hierarchical Trees)及其屬性；SV-5 係由觀點所成的集合，目的在架構作戰活動對系統功能間的追溯性，使用的工具為矩陣(Matrixes)及其屬性；SV-6 係由觀點所成的集合，目的在架構系統資料交換，使用的工具為矩陣(Matrixes)及其屬性；SV-7 係由觀點所成的集合，目的在架構各系統工作的參數，使用的工具為矩陣(Matrixes)及其屬性；SV-8 係由觀點所成的集合，目的在架構各系統進化的歷史，使用的工具為圖形(Graphics)及其屬性；SV-9 係由觀點所成的集合，目的在架構各系統的技術預測，使用的工具為表格(Tables)及其屬性；SV-10a 係由觀點所成的集合，目的在架構系統規則模式，使用的工具為決策規則(Decision Rules)或 IDEF3 及其屬性；SV-10b 係由觀點所成的集合，目的在架構系統狀態轉換，使用的工具為狀態圖(State Charts)及其屬性；SV-10c 係由觀點所成的集合，目的在架構系統事件的軌跡，使用的工具為循序圖(Sequence Charts)及其屬性；SV-11 係由觀點所成的集合，目的在架構系統實體概要，使用的工具為元件圖(Component Charts)或 E-R 圖(Entity-Relationship Diagrams)及其屬性。

Definition 5 (Technical Views) A technical view is a set of architectures $TV = TV-1 \cup TV-2$ where

1. TV-1 is a set of views for architecting the technical architecture profile by using *tables* with *properties*, and
2. TV-2 is a set of views for architecting the standards technology forecast by using *tables* with *properties*.

【定義說明】

技術觀點 (Technical Views) 係由架構 TV 的集合所組成， $TV = TV-1 \cup TV-2$ ，其相關符號說明如下：TV-1 是觀點所成的集合，使用表格(Tables)及其屬性，來架構技術架構輪廓；TV-2 是觀點所成的集合，使用表格(Tables)及其屬性，來架構技術標準預測。

有關 AV、OV、SV、TV 各產品計 26 種，資料元素計 70 個之定義內容如附錄表 1 及 2 所示。

五、解決方法

DoDAF 產品的相關觀點資料，皆存於後台資料庫中，當要作出任何修改規格的情形時，即須對資料庫作出建立(Create)、擷取(Retrieve)、更新(Update)、刪除>Delete)等 CRUD 動作，針對 DoDAF 產品追溯關係空間(Traceability Dependency Space)的 CRUD 演算法，設計如下：

令追溯關係堆疊 N

令追溯關係式為 t

令追溯關係空間為 T

令追溯關係空間 T 中其資料元素所成的集合為 E ，且 E 不為空集合

令欲擷取的資料元素為 e_0

令追溯路徑為 P

令欲修改之資料元素為 u_0

令欲刪除之資料元素為 d_0

(1) 建立(Create)追溯關係空間演算法 Traceability_Dependency_Space(N, T):

Step 1 設定一個追溯關係堆疊 N ，其中有 n 個相依關係式，依序為 $t_0, t_1, t_2, \dots, t_{n-1}$ 其中， $N \neq \emptyset$

Step 2 設定一個追溯關係空間 $T = \emptyset$

Step 3 從追溯關係堆疊 N 中 POP 一個追溯關係式 t_0

Step 4 將 t_0 置放於追溯關係空間 T 中，使得 $T = t_0$

Step 5 Dowhile 追溯關係堆疊 $N \neq \emptyset$

Step 6 從追溯關係堆疊 N 中 POP 一個追溯關係式 t

Step 7 將 t 與 T 結合，使得 $T = T + t$

Step 8 End Dowhile

(2) 在追溯關係空間中，擷取(Retrieve)的演算法 Retrieve(e_0, E, P, T):

Step 1 If $e_0 \in E$ then

Step 2 於追溯關係空間 T 中找到資料元素 e ，使得 $e = e_0$

Step 3 由 e 向前追溯所有路徑 $P1$

Step 4 由 e 向後追溯所有路徑 $P2$

Step 5 令追溯路徑 $P = P1 + e + P2$

Step 6 End If

(3) 在追溯關係空間中，更新(Update)的演算法 Update(u_0, E, P, T):

Step 1 If $u_0 \in E$ then

Step 2 由追溯路徑 P 中找到資料元素 e ，使得 $e = u_0$

Step 3 修改 e

Step 4 修改由 e 向前追溯所有路徑 $P1$ 中的資料

元素

Step 5 修改由 e 向後追溯所有路徑 $P2$ 中的資料元素

Step 6 令追溯路徑 $P = P1 + e + P2$

Step 7 重整 T

Step 8 End If

(4) 在追溯關係空間中，刪除>Delete)的演算法 Delete(d_0, E, P, T):

Step 1 If $d_0 \in E$ then

Step 2 由追溯路徑 P 中找到資料元素 e ，使得 $e = d_0$

Step 3 刪除 d_0

Step 4 刪除由 e 向前追溯所有路徑 $P1$ 中的資料元素

Step 5 刪除由 e 向後追溯所有路徑 $P2$ 中的資料元素

Step 6 設定追溯路徑 $P = \text{nil}$

Step 7 重整 T

Step 8 End If

六、結論

DoDAF 的 26 項產品相依的追溯過程隱含著：作戰觀點(OV)、系統觀點(SV)、技術觀點(TV)，各觀點內的橫向追溯；作戰觀點(OV)、系統觀點(SV)、技術觀點(TV)各產品間，跨觀點的縱向追溯及跨產品間的間接追溯，惟有發展一特定的軟體追溯工具才能解決此追溯性問題。

本研究於尋獲 DoDAF 各資料元素相互間的追溯特性後，對後台資料庫的各該資料元素作出 CRUD 動作，俾隨時保持 DoDAF 各資料元素的一致性與完整性。

參考文獻

- [1] D. Leffingwell and D. Widrig, "The Role of Requirements Traceability in System Development," *The Rational Edge*, 2002, p.2.
- [2] C. H. Jane, B. Berenbach, S. Clark, R. Settini and E. Romanova, "Best practices for automated traceability," *IEEE Computer*, Volume 40, No. 6, 2007, pp.27-35.
- [3] A. D. Lucia *et al.*, "Traceability management for impact analysis," *Proceedings of the 24th IEEE International Conference on Software Maintenance*, Beijing, China, 28 September - 4 October, 2008, pp.21-30.
- [4] O. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," *Proceedings of the 1st International Conference on Requirements Engineering*, Colorado Springs, Colorado, USA: IEEE CS Press, 1994, pp.94-101.
- [5] A. D. Lucia *et al.*, "Improving Source Code Lexicon via Traceability and Information Retrieval," *IEEE Transactions on Software Engineering*, Volume 37, Issue 2, March 2011, p.4.

- [6] B. Sharif *et al.*, "Using Fine-Grained Differencing to Evolve Traceability Links," *The 4th ACM International Workshop on Traceability in Emerging Forms of Software Engineering (GCT/TEFSE'07)*, Lexington, Kentucky, 22-23 March, 2007, pp.76-81.
- [7] C. H. Jane and K. C. Chang, "Supporting Event Based Traceability through High-Level Recognition of Change Events," *Proceedings of the 26th Annual International Computer Software and Applications Conference*, Oxford, England, August 2002, pp.595-600.
- [8] A. V. Knethen, "Change-Oriented Requirements Traceability: Support for Evolution of Embedded Systems," *Proceedings of International Conference on Software Maintenance (ICSM 02)*, 2002.
- [9] S. A. Sherba, K. M. Anderson and M. Faisal, "A Framework for Mapping Traceability Relationships," *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE '03)*, October 2003.
- [10] L. G. P. Murta *et al.*, "ArchTrace: Policy-Based Support for Managing Evolving Architecture-to-Implementation Traceability Links," *Proceedings of the 21st IEEE International Conference on Automated Software Engineering (ASE'06)*, 2006, pp.135-144.
- [11] S. A. Sherba, K. M. Anderson and M. Faisal, "A Framework for Mapping Traceability Relationships," *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE '03)*, October 2003.
- [12] Youngki Hong *et al.*, "Requirements Management Tool with Evolving Traceability for Heterogeneous Artifacts in the Entire Life Cycle," *Proceedings of the 8th ACIS International Conference on Software Engineering Research, Management and Applications*, Montreal, Canada, 24-26 May, 2010.
- [13] C. H. Jane *et al.*, "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability," *The 13th IEEE International Conference on Requirements Engineering*, 2005.
- [14] C. H. Jane *et al.*, "Model-Based Traceability," *ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, Vancouver, Canada, 18 May, 2009.
- [15] M. P. Bienvenu, I. Shin and A. H. Levis, "C4ISR Architectures III: An Object-Oriented Approach for Architecture Design," *Systems Engineering* Volume 3, No. 4, 2000, p.36.
- [16] L. W. Wagenhals, I. Shin, D. Kim and A. H. Levis, "C4ISR Architectures II: Structured Analysis Approach for Architecture Design," *Systems Engineering*, Volume 3, No. 4, 2000, p.2.
- [17] S. J. Ring, D. Nicholson, J. Thilenius and S.

Harris, "An Activity Based Methodology for Development and Analysis of Integrated DoD Architectures," *Command and Control Research and Technology Symposium*, 2004.

- [18] C. H. Jane, G. Zemont and W. Lukasik, "A Heterogeneous Solution for Improving the Return on Investment of Requirements Traceability," *The 12th IEEE International Requirements Engineering Conference*, 2004.
- [19] S. M. Richard, "A Traceable Systems Engineering Methodology," *Proceedings of the 9th Digital Avionics Systems Conference, IEEE/AIAA/NASA*, 15-18 October, 1990.
- [20] A. Jossic, M. D. D. Fabro *et al.*, "Model Integration with Model Weaving: A Case Study in System Architecture," *Proceedings of International Conference on Systems Engineering and Modeling*, 20-23 March, 2007.
- [21] Leilei Kong and Tao Yuan, "Extension Features-Driven Use Case Model for Requirement Traceability," *Proceedings of the 4th International Conference on Computer Science Education*, 2009.
- [22] N. Tamim and I. Exman, "Compact Comparison Of Competing Software Designs," *Proceedings of the 23rd IEEE Convention of Electrical and Electronics Engineers*, Israel, 6-7 September, 2004.
- [23] A. Sousa, U. Kulesza, A. Moreira, V. Amaral, J. Araújo, A. Rummier, N. Anquetil and R. Mitschke, "A Model-Driven Traceability Framework to Software Product Line Development," 2010, <http://www.ample-project.net/>.

附錄

表 1 OV 各產品資料元素定義內容

產品	資料元素	定義之內容	產品	資料元素	定義之內容
OV-1	TargetArea	想定之全部場景	OV-4	IoFlowConnector	作戰行動資訊流連接
	Line	互動關係線		ControlFlowConnector	作戰行動控制項輸入連接
OV-2	AssetIcon	組織等資產標點	OV-6a	Rule	作戰規則
	OpNode	作戰節點		State	作戰狀態
OV-3	NeedLine	需求線	OV-4b	Action	作戰行動
	InfoExchange	作戰資訊交換		Event	作戰事件
OV-4	InfoElement	作戰資訊元素	OV-4c	Transition	作戰狀態轉移
	OrgRelationship	組織關係		Guard	作戰之監控
OV-5	HumanRole	作戰人員	OV-6c	LabelLine	作戰行動時間線
	Organization	作戰組織		EventTime	作戰事件發生時間
OV-6	OrgType	組織型態	OV-7	Event	作戰事件
	OpActivityIoInformation	作戰行動輸入資訊流		RelationshipType	作戰關係型態
OV-7	MechanismFlowConnector	作戰行動輸入資源	OV-7	OpAttributeType	作戰屬性型態
	OpActivity	作戰行動		OpEntityType	作戰實體型態

表 2 SV 與 TV 各產品資料元素定義內容

產品	資料元素	定義之內容	產品	資料元素	定義之內容
SV-1	SystemNode	系統節點	SV-7	SystemEntity	系統實體
	SystemEntity	系統實體		SystemElement	系統元素
SV-2	SystemElement	系統元素	SV-8	SystemComponent	系統組件
	SystemComponent	系統組件		SystemServiceEvolutionPlan	系統演進計畫
SV-3	InterfaceLine	界面線	SV-9	SystemServiceTechForecast	系統技術預測
	SystemNode	系統節點		SystemRule	系統規則
SV-4	SystemEntity	系統實體	SV-10a	SystemState	系統狀態
	SystemElement	系統元素		SystemAction	系統行動
SV-5	SystemInterface	系統界面	SV-10b	SystemEvent	系統事件
	SystemNode	系統節點		SystemTransition	系統狀態轉移
SV-6	SystemEntityComparisonMatrix	系統實體比較表	SV-11	SystemGuard	系統監控
	SystemFunction	系統功能		SystemEvent	系統事件
SV-7	SystemDataFlow	系統數據流	SV-12	SystemEventTime	系統事件發生時間
	SystemFunction	系統功能		SystemLabelLine	系統事件發生時間線
SV-8	SystemActivity	系統行動	SV-13	SystemRelationshipType	系統關係型態
	SystemFunction/OpActivityMatrix	系統功能對作戰行動比較表		SystemAttributeType	系統屬性型態
SV-9	SystemDataExchange	系統資料交換	TV-1	SystemEntityType	系統實體型態
	SystemDataElement	系統資料元素		TechStandardProfile	系統標準專業指標
SV-10	SystemServicePerformanceMatrix	系統服務性能比較表	TV-2	TechStandardForecast	系統技術標準預測
	SystemServiceEvolutionForecastParameters	系統元素演進預測參數表			