異質性企業架構整合器應用在資訊系統平台整合規劃
Applications to Information System Platform Integration by Using Heterogeneous Enterprise Architecture Integrator Platform

蕭國裕 碩士研究生
德明財經科技大學資訊科技與管理研究所
Kuo-yu Hsiao
Graduate Institute of Information Technology and Management,
Takming University of Science and Technology
Email: G9763118@cc.takming.edu.tw

韓孟麒 副教授
德明財經科技大學資訊科技系
Meng-chyi Harn
Department of Information Technology,
Takming University of Science and Technology
Email: harn@takming.edu.tw

陳文賢 教授
德明財經科技大學資訊科技與管理研究所
Wen-hsien Chen
Graduate Institute of Information Technology and Management,
Takming University of Science and Technology
Email: wchen@im.ntu.edu.tw

摘 要

企業資訊系統是一個幫助企業以追求最小成本和最大利潤為基礎的經營平台，企業依靠著資訊系統支配企業架構下各種不同的資源。這些資源有些是靜態可重複利用的，也有些是動態建立不能重複利用的，有些採用分散管理，也有些是集中管理，有些甚至是行動裝置可隨處移動的。因此，企業資訊系統勢必擁有非常大的擴充彈性，將企業架構下的資源做最有效的發揮。當下各界在企業資訊系統規劃的領域有眾多的方法論和設計工具，令人遺憾的是至今仍然沒有任何一種模式能用簡單的方法來敘述複雜的企業架構。目前流行的各種方法傾向使用多種不同構面的視圖來敘述企業資訊系統。而每一個構面都需要花費時間相互測試，一旦面對問題需要進行評估或修正的時候，必須仔細研究每一張視圖才能從複雜的構面中取得相關資訊。這個過程使得企業增加了有形及無形的成本，相對造成經營上的損失。由於上述問題的不斷產生，因此本研究利用過去軟體工程領域中的一種雛型理論，搭配其衍生的圖型化介面理論為基礎，對異質性平台整合提出一種新的解決方法。此方法實作過程中包含兩項工作：(1)分析企業組織架構(2)分析組織工作流程。是故，本研究係以前述兩項工作為基礎，對企業資訊系統做進一步規劃。

關鍵字：企業架構、電腦輔助雛型系統、雛型系統描述語言、可擴展標記語言

本研究為國科會所補助，專題研究計畫編號為：98-2623-E-147-001-D

一、緒論

為了因應企業架構下的協同作業中所產生異質性資訊系統整合議題，本研究的目的乃在設計一個「異質性企業架構整合器」(Heterogeneous Enterprise Architecture Integrator, HEAI)使其可提供企業分析工作流程及組織架構。並且為異質性企業架構進行細部分解，藉此協助企業的流程再設計(Business Process Redesign, BPR)。研究中探討的 HEAI 具備能夠描述企業組織功能和運作流程，同時也能夠分析各項作業過程中的條件和參數細節。HEAI 還提供了外部組織的功能加入，能夠從處理企業內部的流程控制到企業外部的流程並給予整合。為了方便讓企業內部非資訊相關人員操作，本研究強調以簡單的操作介面來處理複雜的敘述。企業若善加利用 HEAI 則能夠避免長期採購複雜而昂貴的輔助軟體，也能有效防止委外設計的資訊系統造成後續不可維護的問題。

二、技術背景

針對研究目的所設計的 HEAI 是利用可擴展標記語言 (eXtensible Markup Language, XML) [12]技術做為資料交換的標準，並且採用雛型系統描述語言 (Prototype System Description Language, PSDL) [9]理論做為分析工具，同時配合電腦輔助雛型系統 (Computer-Aided Prototyping System, CAPS) [7]為視覺輔助介面。HEAI 綜合以上技術獨特的功能性，幫助企業建立物件存取協定(Simple Object Access Protocol, SOAP)。

(一) 雛型系統描述語言 PSDL

根據 PSDL 的理論，它是以作業中的層次結

構為設計基礎[6]，使用高階程式語言[15]將任意一層的作業元件做更詳細的分解。每項動作的描述都包含內部控制流程以及外部依賴關係；每一個作業元件都可以是高層次抽象描述，也可以是詳細實作的方法[4] 。以 PSDL 定義的 2 項作業如圖 1 所示，簡單的描述出 2 項作業的依賴關係。因此能夠得知 PSDL 的設計概念對於反應事件需求以及流程再造相關工作的方便性[6]，經由過去的相關研究發現，PSDL 有效的增加了設計的可靠性[10]。

```
OPERATOR Order
    SPECIFICATION
        OUTPUT Send_message : Order_info
    END

OPERATOR Confirm
        INPUT Send_message : Order_info
        OUTPUT export_commodity_inspection : Order
    END
```

圖 1 以 PSDL 敘述的 2 項作業

(二) 電腦輔助雛型系統 CAPS

CAPS 是以標準化的規範運用抽象方式將使用者提出的需求快速展現，透過一個可重用的方法元件庫直接產生可運作的系統架構[7]。這種開發模式在設計的概念層級做關鍵整合，再以分解的方式逐一詳述每項作業。在過去的相關研究資料顯示，這樣一個高度自動化的環境是一個能夠確實滿足需求的有效方法。不但縮短了整體架構分析的人力跟時間，並且能明顯的提高了系統完成後的可信度和整體效率[9]。

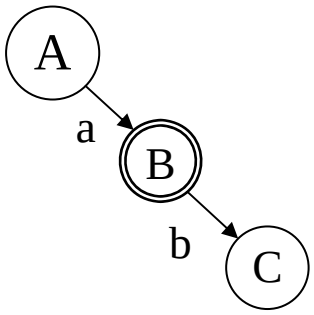

圖 2 CAPS 描述一個處理流程

如圖 2 所示，透過資訊流「→」的方向表示發送與接收端間的關係，當中包含資料格式的定義和資料本身的內容，是各作業元件之間連結的重要符號。圖中的「◎」符號代表流程內包含細部流程，而「○」符號則是代表流程目前沒有提供進一部的細部描述。

(三) 可擴展標記語言 XML

XML 分離各顯示區塊和內容描述，可以改善內容讀取的效能並且減少解析錯誤的機會。區塊清析並且使得解析程式能夠更加容易判斷，對於經常變換內容的需求能夠更加流暢的適應[12]。XML 是從 1995 年開始有其雛形，並向 W3C(全球資訊網聯盟)提案，在 1998 年 2 月發佈為 W3C 的標準。

(四) 簡單物件存取協定 SOAP

SOAP 由國際商業機器股份有限公司(International Business Machines Corporation, IBM)在 1998 年發表，於 2000 年提供給全球資訊網聯盟(World Wide Web Consortium, W3C)目前為各界所認同的標準。SOAP 的產生是為了讓資料在遠端存取的時候，明定雙方或多方共通的標準規格[1]。SOAP 是以 XML 格式的檔案互相交換彼此的資料，與企業所使用的資訊技術均無直接關連。因此異質性的企業平台能夠輕易的利用 SOAP 建立合作標準。

三、問題探討

當下的分析企業架構的理論及輔助軟體已經為企業製訂了多種構面[11]，並依照各個構面產生了數種至數十種概念完全不同的視圖[16]。過度地對作業上不同的構面所建立的分析，反而會妨礙對企業流程的瞭解。如同「義大利麵文字(Spaghetti Code)」[18]，即是一種複雜難懂的文字，雖然得到了分析結果，卻損失了分析效率。所以我們開始找尋過去的相關研究，發現了推出至今近 30 年的歷史的 PSDL 理論，其主要用途是快速展現即時系統的需求[4]。PSDL 語法簡單易懂，除了能夠輔助開發，也可以持續更新並擴充。使用 CAPS 輔助 PSDL 做為軟體工程用途近乎完美，但若將其理論應用在 HEAI 架構上仍有少數問題有待解決。這些問題有些是早在設計之初就存在的缺憾，有些則是為了符合現代化需求所必須建立的新規格。

PSDL 的語法沒有明確的將動作指令與動作參數分離，因此在解析時容易造成輔助解析工具的編譯錯誤。比較明顯的例子除了一般的語法容易出現編譯錯誤之外，編譯過程中無法解析萬國碼(Unicode Transformation Format, UTF)。這是一個嚴重的缺陷，此缺陷使得 PSDL 無法建立以萬國碼編碼的文字。如圖 3 所示的 PSDL 的內容如果加入了如圖 4 所示的中文或是萬國碼，將使得其內容無法解析。

```
OPERATOR Order
    SPECIFICATION
        OUTPUT Send_message : Order_info
    END

OPERATOR Confirm
        INPUT Send_message : Order_info
        OUTPUT export_commodity_inspection : Order
    END
```

圖 3 以 PSDL 敘述的 2 項作業

```
OPERATOR 訂購
    SPECIFICATION
        OUTPUT  訊息傳送 : 訂單資料
    END

OPERATOR \u78bc\u7dad\u8b77
        INPUT \u52a0\u9078 : \u78ba\u5b9a
        OUTPUT \u8acb\u52fe\u9078 : \u67E5\u8A62
    END
```

圖 4　PSDL 敘述嵌入中文編碼將無法解析

PSDL 易學易用，但是 PSDL 對於元件的敘述方式受限於 PSDL 本身所規範的格式無法進一步擴充。如圖 5 所示的內容中新增「COST」屬性後便無法解析。為了描述協同作業的企業架構或新加入的工作流程，PSDL 在企業應用上必需擴充方法。

```
OPERATOR Order
    SPECIFICATION
        OUTPUT Send_message : Order_info
        COST : 10
    END

OPERATOR Confirm
        INPUT Send_message : Order_info
        OUTPUT export_commodity_inspection : Order
        COST : 70
    END
```

圖 5 PSDL 無法對應此新屬性「COST」

企業內的組織架構、各種活動、不同狀況產生的不同流程與複雜元件若以 PSDL 描述，會使得 PSDL 文件的內容將日漸增長，同時文件的搜尋效率必定和文件長度成反比。當檔案資料成長到硬體處理效能的極限時，將會導致分析工作難以進行。

綜合上述問題，本研究提出利用 XML 輔助 PSDL。經過 XML 的重新編排，對於上述問題，能提供顯助的改善。首先，XML 能夠明確的將文件中的區塊標示與區塊內容分開，因此輔助解析工具對於內容中的資料編碼不需以特例的方式處理，因此 XML 被廣泛用來作為跨平台之間交互數據的形式。

```
<operator>
    <name>訂單受理作業</name>
    <output>訂單資料</output>
</operator>

<operator>
    <name>貨運物流作業</name>
    <input>訂單資料</input>
    <output>出貨資料</output>
    <fax>02-2345-6789</fax>
</operator>
```

圖 6 將圖 1 的動作以 XML 敘述

XML 格式的檔案可用來傳送自定格式的資料，XML 能夠同時可以表現資料內容，也能說明資料架構格式，甚至能任意擴充並使其更豐富[13]；如圖 6 中的第 2 項作業以標籤所顯示新屬性「fax」。

```
<operator>
    <name>接單作業</name>
    <output>訂單資料</output>
    <cost>
        <fax>6</fax>
        <print>4</print>
    </cost>
</operator>
<operator>
    <name>出貨作業</name>
    <input>訂單資料</input>
    <output>出貨資料</output>
    <cost>70</cost>
</operator>
```

圖 7 運用 XML 為流程中增加屬性

XML 格式的文件可同時接受按照循序排列，或是隨機排列的格式讀取，並且允許任意插入資料。除此之外還能夠對於複雜的 XML 加上文件類型定義(Document Type Definition, DTD)XML 能夠提供索引，增加辨識的效率[12]。

四、需求分析

組織活動的單位集合可稱為架構(A-architecture)，架構技術至今無論在企業界、學術界尚未形成一個統一的模式，而不同方法所採用的觀察角度也會造成不同理解。本研究之需求分析從組織及其活動進行細部分析，依據分析結果進一步針對原本的作業模式做功能架構設計及整合。為使本文方便閱讀，接下來的例子中每項作業元件或組織單位，只針對本研究相關的屬性提出並加以分析。

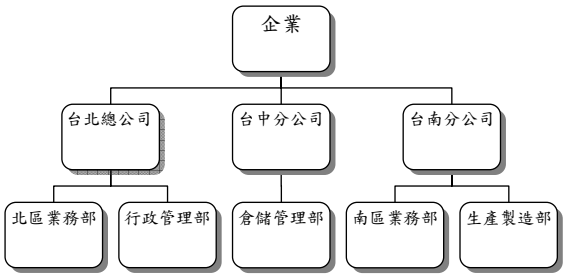

圖 8 企業組織架構圖

某個企業的組織架構如圖 8 所示，其架構下有 3 間子公司，而各子公司都擁有 2 個部門及若干單位。在這個階段尚未使用任何資訊技術或科技輔助，只使用抽象的概念把圖形加以連結用來表示彼此間的關係。為了整合企業資源的前置工作，首先有需要先將這些抽象的最上層節點加以分析，產生標準格式的 XML。

(一) 架構分析

如圖 9 所示的一段 XML 敘述，當中每筆資料是運用「parent」屬性的關連性建立樹狀清單的結構，其屬性資料若為「NULL」代表為根元素，反之則表示該資料擁有上層元素。

```
<enterprise-config>
    <unit-beans>
        <unit-bean name="台北總公司" parent="NULL"/>
        <unit-bean name="台中分公司" parent="NULL"/>
        <unit-bean name="台南分公司" parent="NULL"/>
        <unit-bean name="北區業務部" parent="台北總公司"/>
        <unit-bean name="行政管理部" parent="台北總公司"/>
        <unit-bean name="倉儲物流部" parent="台中分公司"/>
        <unit-bean name="生產製造部" parent="台南分公司"/>
        <unit-bean name="南區業務部" parent="台南分公司"/>
    </unit-beans>
</enterprise-config>
```

圖 9 以 XML 表示組織架構

(二) 資源分析

如圖 10 所示，以細部分析檢視組織中的一個單位「倉儲管理組」，該單位擁有 2 個「parent」標籤，分別為「行政管理部」、「倉儲物流部」。這表示由台北總公司的「行政管理部」及台中分公司的「倉儲物流部」2 個上層組織的架構下都有配置「倉儲管理組」的相關業務。因此「倉儲管理組」可以獲得其上層組織架構的所有屬性參數，無需再做多次的重複分析。如果有擴充的需求，可以在屬性標籤內增加欄位。

```
<enterprise-config>
  <unit-beans>
    <unit-bean name="台中分公司" parent="NULL"/>
<unit-bean name="台北總公司" parent="NULL"/>
<unit-bean name="行政管理部">
      <parent>台北總公司</parent>
      <telphone>02-2345-6789</telphone>
      <fax>02-6789-5432</fax>
    </unit-bean>
    <unit-bean name="倉儲物流部" />
      <parent>台中分公司</parent>
      <telphone>06-789-1234</telphone>
      <fax>06-789-2468</fax>
    </unit-bean>
    <unit-bean>
      <name>倉儲管理組</name>
      <parent ext="222">行政管理部</parent>
      <parent>
        <name>倉儲物流部</name>
        <ext name=”林組長”>333</ext>
        <ext name=”陳組員”>666</ext>
      </parent>
    </unit-bean>
  </unit-beans>
</enterprise-config>
```

圖 10 針對一個單位進行細部描述並擴充

圖 10 的例子中顯示：「倉儲管理組」內標記的 2 個上層單位「parent」標籤均增加了辦公室分機「ext」屬性。因此追溯上層組織的屬性可以就能夠清楚顯示：撥了「台中分公司」的電話就可以利用分機 333、666 都可以找到「倉儲管理組」的相關人員。以這個模式將整個系統所描述輪廓更加簡單嚴謹，前因後果清楚明白地表達。即使未來的狀況有所變動，只需要在根節點做修正即可。有效避免整體規劃的結果中發生前後矛盾的情形。

(三) 作業分析

經過分析後以 XML 表示的作業內容如圖 11 所示。其內容主旨為：北區業務部傳送一筆訂單資料給行政管理部，行政管理部得到訂單資料後將生產命令下達給生產製造部。

```
<operator>
  <name>訂單受理作業</name>
  <unit>北區業務部</unit>
  <output>
    <name>訂單資料</name>
    <receiver>行政管理部</receiver>
    <receiver>生產製造部</receiver>
  </output>
</operator>

<operator>
  <name>訂單審核作業</name>
  <unit>行政管理部</unit>
  <input>
    <name>訂單資料</name>
  </input>
  <output>
    <name>生產指令</name>
    <unit>生產製造部</unit>
  </output>
</operator>
```

圖 11 作業的細部描述同時與組織形成關連

五、實作驗證

企業架構下的組織各司其職並有效率的分工合作，而合作過程中，資料交換是一項重要的工作。如圖 12 所示的一項典型的資料交換作業「南區業務部」將「訂單」傳送給「行政管理部」，而

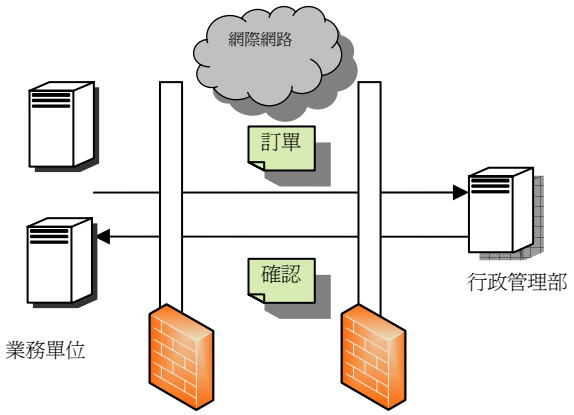

圖 12 既存的資料交換架構

「行政管理部」將「確認」傳回給「南區業務部」。

雙方得到相關資料後，就能夠順利進行後續的工作 。資料交換的工作內容非常簡單，因此也很有效率的運作。但是隨著資訊科技的提昇，本研究對此資料交換流程提出了 3 項簡單的需求： (1)數位簽章(Digtal Signture)、(2)傳輸過程加密、(3)即時性的資料備份。

這 3 項新加入的作業流程若使用了目前主流的各種新技術或是昂貴的輔助軟體以多種構面和視圖展現。比較好的狀況是企業內部資訊人管理單位閱數張至數十張不等的視圖後，經過很長的時間仔細分析各種構面，就能夠開始設計新流程。比較糟糕的狀況則是企業內部資訊人員無法分析這數張至數十張不等的視圖，必須以極高的代價聘請企業外的顧問公司協助分析。委外設計雖然能夠快速得到分析結果，但是顧問公司無法於短時間分析企業的完整架構，在佈署後的新作業流程則存在極高的風險。

使用 HEAI 為輔助工具分析如圖 12 所示的資料交換架構圖，產生的 HEAI 圖形介面如圖 13 所示。圖中顯示有 2 項可分解的工作分別為「訂單受理作業」和「訂單確認」，這 2 項作業各自在完成內部工作後之後產生 1 條資訊流。其中包含資料本身、各種屬性以及控制參數。

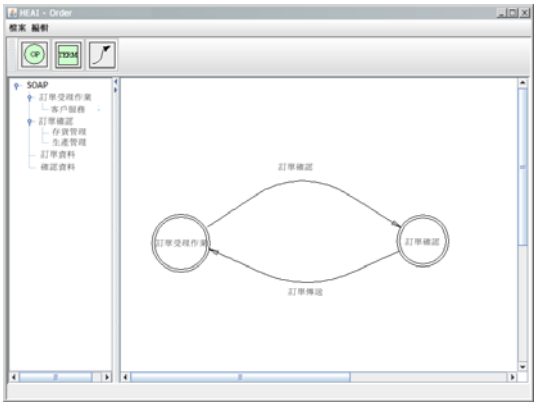

圖 13 以 HEAI 描繪的資料交換架構

分解後的 XML 檔案如圖 14 所示，圖中同樣有 2 項作業標籤「operator」，第 1 項作業標籤包含 2 個「unit」標籤，分別為「北區業務部」及「南區業務部」，因此能夠得知這項作業目前適用於這 2 個單位。接下來的「output」標籤中只包 1 個「unit」為「行政管理部」，表示資訊流只有傳給行政管理部 1 個單位。第 2 項作業擁有第 1 項作業所沒有的「input」標籤，這是為了承接第 1 項作業而產生的，因此第 2 項作業能夠辨識資訊中包含的發送單位回傳「確認資料」給該單位，雙方完成作業。圖 14 的內容顯示，作業中若由 2 個組織共同合作的情況下，產生的「order-by」標籤是為了指明觸發條件。條件為「some」時，表示何事件觸發則立即動作。條件為「all」時，則表示全部事件完成才能進行下一動作。如果有多項動作產生時則以「order-by」標籤產生陣列，依先後順序排列表示。

```
<operator>
  <name>訂單受理作業</name>
  <unit>北區業務部</unit>
  <unit>南區業務部</unit>
  <output>
    <name>訂單資料</name>
    <unit>行政管理部</unit>
    <order-by>some</order-by>
  </output>
</operator>
<operator>
  <name>訂單確認作業</name>
  <unit>行政管理部</unit>
  <input>
    <name>訂單資料</name>
  </input>
  <output>
    <name>生產排程資料</name>
    <order-by>生產製造部</order-by>
    <order-by>北區業務部</order-by>
    <order-by>南區業務部</order-by>
  </output>
</operator>
```

圖 14 以 XML 分解的作業內容

為了加入本研究為資料交換架構提出的 3 項新需求，因此以 HEAI 介面為流程提出修改。利用圖形化的操作介面在過程中加入了新的作業元件，幾乎對整份設計毫無變更的情況下嵌入這些流程。實作上無論是企業內部的資訊人員開發，或是委外開發均能確實達到需求，不會造成任何風險。

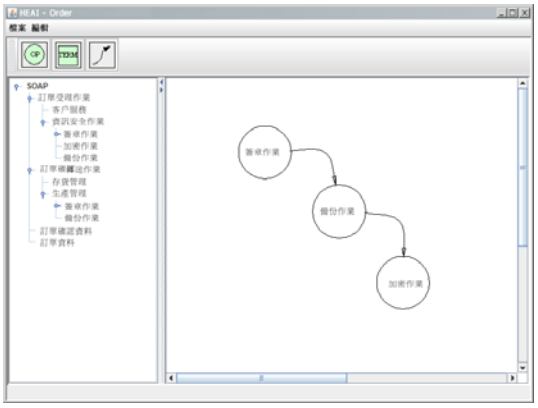

圖 15 使用 HEAI 對原先的作業加入了 3 項新需求

如圖 15 所示的 HEAI 運用原本的「訂單受理作業」分解並加入了 3 項需求，產生了新的資料交換架構。由於根元素的重複利用，因此在加入新的流程設計時，將會連動相關的各項作業，不需逐一修改調整。為了快速符合新的需求，甚至能夠隨時給予調整。圖 14 的作業階層顯示，在訂單確認的作業中，順利地在適當的時機加入了新規劃的 3 項新需求，如圖 16 所示。

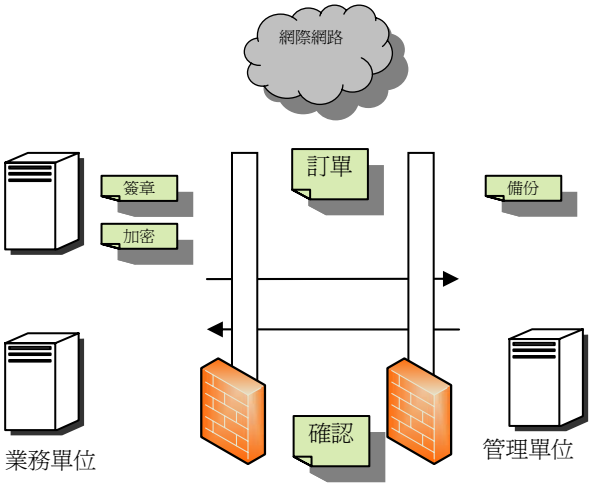

圖 16 經過重新設計後的新架構

六、結論

現代軟體工程的需求已經朝向大量且快速建置的開發模式，為了提高軟體可信度及後續發展，新的流程必須以適應需求變化為目的[3]。企業流程的週期性與軟體工程的週期性有著顯著的相同之處。改善後的 HEAI 將為軟體工程與企業流程建立一座溝通的橋樑，能夠更加靈活而且更加正規化的高效率的適應新需求。HEAI 能將需求經過簡單的操作分解流程中的元件，當流程運行了一段時間後需要擴充或修改時，HEAI 則再次運用相同的操作模式快速的反應新需求。目前 HEAI 已可達到分析組織架構並和工作流程的要求，未來將配合非資訊人員設計更人性化的操作介面。為了建立新一代的 CAPS 工具，我們同時會在未來的研究中增加與軟體工程相關的研究項目。

參考文獻

[1] T. Chester, "Cross-platform integration with XML and SOAP," *IT Professional* Volume 3, Issue 5, September-October 2001, pp.26-34.

[2] E. Guerrieri, "Software Document Reuse with XML," *IEEE Software*, June 1998, pp.246-254.

[3] M. Harn, V. Berzins, and Luqi, "Software Evolution Process via a Relational Hypergraph Model," *Proceedings of IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems, Tokyo, Japan*, October 5-8, 1999, pp. 599-604.

[4] B. Kramer, Luqi, and V. Berzins, "Compositional Semantics of a Real-Time Prototyping Language," *IEEE Transactions on software engineering*, Volume 19, No. 5, May 1993, pp.453-477.

[5] P. Louridas, "Version control," *IEEE Software*, Volume 23, Issue 1, January-February 2006, pp.104-107.

[6] Luqi, V. Berzins and T. Yeh, "A Prototyping Language for Real-Time Software," *IEEE Software*, October 1998, pp.1409-1423.

[7] Luqi and M. Ketabchi, "A Computer-Aided Prototyping System," *IEEE Software*, March 1988, pp.66-72.

[8] Luqi and V. Berzins, "Rapidly Prototyping Real-Time Systems," *IEEE Software*, September 1988, pp.25-36.

[9] Luqi, "Computer-Aided Prototyping for a Command-And-Control System Using CAPS," *IEEE Software*, January 1992, pp.56-67.

[10] Luqi, Z. Lin, V. Berzins and Q. Ying, "Documentation Driven Development for Complex Real-Time Systems," *IEEE Transactions on Software Engineering*, Volume. 30, No. 12, December 2004, pp.936-952.

[11] R. Lee and W. Tepfenhart, *Practical Object-Oriented Development with UML and Java*, Pearson, 2004, pp.A1-A32.

[12] J. Roy and A. Ramanujan, "XML: data's universal language," *IT Professional*, Volume 2, Issue 3, May-June 2000, pp.32- 36.

[13] C. Suvendi, "An XML Model for Use Across Heterogeneous Client-Server Applications," *IEEE Transactions on instrumentation and measurement*, Volume 57, No. 10, October 2008, pp.2128-2135.

[14] C. Suvendi and G. Hancke, "A New Model for Secure Dissemination of XML Content," *IEEE Transactions on Systems, Man, and Cybernetics part C: applications and reviews*, Volume 38, No. 3, May 2008, pp.292 - 301.

[15] H. Strong, "High level languages of maximum power," *Switching and Automata Theory (1971) 12th Annual Symposium*, October 1971, pp.1-4.

[16] G. Shelly, T. Cashman and H. Rosenblatt, *Systems Analysis and Design*, Thomson, 2007, pp.190-252.

[17] W. Wolf, "Object programming for CAD," *Design & Test of Computers, IEEE*, Volume 8, Issue 1, March 1991, pp. 35-42.