A Concurrent Model for Software Development
軟體開發流程同步化的探討

侯永昌
Hou, Young-Chang

國立中央大學資訊管理系
ychou@im.mgt.ncu.edu.tw

胡光輝
Kuang-Huei Hu

私立南亞工商專校應用外語科
khhu@nanya.edu.tw

摘 要

本文提出一個軟體系統發展架構的模式，解決軟體系統同步開發所遭遇的問題。所提出的模式稱之爲雙螺旋模式(double spiral model)，此模式分爲外螺旋模式(outside spiral model)及內螺旋模式(inside spiral model)。

在外螺旋模式方面，其關鍵概念是將系統開發引入版本演進的概念，透過模式中的風險分析，它能夠適當的過濾外界的變動需求，必要時可將某些需求變更延遲至新版本再加入，以降低需求變動對於系統開發的時程影響。

內螺旋模式係指在每一個外螺旋模式週期內，運用多個團隊共同開發產品時的反覆性模式，經由物件團隊、功能團隊及品管團隊間的密切合作，軟體系統之一致性、整合性的品質問題，可獲得一相當的保證。

關鍵詞：軟體系統開發、雙螺旋模式、同步化開發、物件導向

Abstract

In this paper, we present a software development model, the double spiral model, to solve the problem encountered at the time of developing software concurrently.

The outside spiral model involves the concepts of version enhancement. The development process will continue periodically to incorporate the new requirements and the better functions. Hence, after the cost-benefit analysis and the technical reviews, some functions may be decided to put into the next version in order to reduce the risk of development and meet the time constraint for market.

At each development cycle, the object design team, the function design team and the quality assurance team work together closely. Through the interaction among these teams, the inside spiral, the software quality can be improved.

Keyword: Software development, Double spiral model, Concurrent development, Object-oriented

壹、前言

大多數軟體工程的研究，多半集中在軟體品質的改善或降低軟體發展成本的議題上，往往忽略了如何減少軟體開發的時間週期(cycle time)。尤其是在快速改變及高度競爭的市場裡，一個短期內就能夠上市(short time-to-market)的產品，往往佔有相當大的優勢[9]。

就最近 10 年軟體開發流程的改變來看，目前軟體開發流程的模式已漸漸朝：由循序到同步，由集中到分散的趨勢[3]。但是在追求軟體開發同步化、縮短開發週期的過程中，軟體品質的確保卻也更具挑戰難度。尤其在同步發展大型軟體系統方面，往往是以人力資源的量來謀求開發時間的減少，但是根據 Brooks 的經驗，單純以多少人、多少月(Man-Month)的工作量計算方法並無法確保軟體的品質及開發時程的縮短[5]。

有相當多的期刊論文基於不同的觀點提出各種軟體流程管理的典範(paradigms)，如以 net-based 觀點的動態任務網路(DYNAMIc Task nEts, DYNAMITE)[7]、派屈網路(Petri Net)觀點的流程模式

化(process modeling)[6]、物件導向觀點的流程模式化[2]等。雖然這些模式化工具都支援同步團隊的軟體發展，但它們的主要目的是用來輔助管理者有效監控專案的進度，以及傳達可能的危機警示。亦即，這些輔助工具僅僅只是一個流程配置的框架，其預設流程仍是從一個單一流程的觀點出發，雖然有所謂模組分解的同步設計和實作，但仍將同步流程設計的工作交由開發者本身去負責。

本文試圖提出一個軟體系統發展架構的模式，解決軟體系統同步開發所遭遇的上述問題。所提出的模式稱之爲雙螺旋模式(double spiral model)，此模式分爲外螺旋模式(outside spiral model)及內螺旋模式(inside spiral model)。其關係如圖 1 所示。

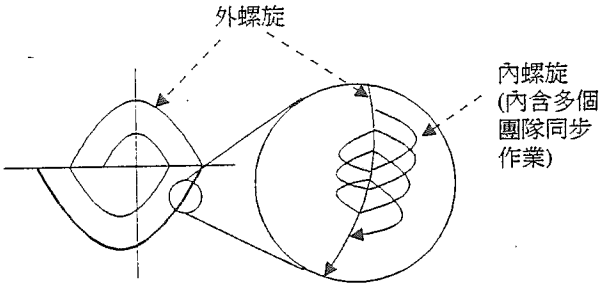

圖 1 雙螺旋模式的內、外螺旋關係圖

在外螺旋模式方面，它的概念架構基礎來自 Boehm 於 1988 年提出的螺旋式模式(Spiral Model)[4]。其關鍵概念是將系統開發引入版本演進的概念，亦即系統的開發不須一次就將所有完整的功能完美的實現，必要時可將某些需求變更延遲至新版本再加入，以降低需求變動對於系統開發的時程影響。

內螺旋模式係指在每一個外螺旋模式週期內，運用多個團隊共同開發產品時的反覆性模式。在此模式中，我們將需求分析之間及子系統之間的一致性檢核，皆轉換成物件的層次來檢視；並將一致性檢核及流程干擾之間的取捨(trade off)，透過產品形成的演進結構，找出同步化系統發展的演進路徑(evolution path)。

由上述的內、外螺旋模式的關鍵概念得知，降低軟體需求變動、檢核子系統間的一致性概念是發展同步化軟體開發的必要條件，而這些條件的目標設定就是期望在軟體品質的優先考量下，試圖達到軟體開發時程的有效縮減。

貳、雙螺旋模式的特性

外螺旋模式主要集中在版本演進的觀念上，每一個版本都是完整可執行的系統(如同我們已熟悉的各類套裝軟體版本)。其每個循環週期是以版本的上市/上線爲主要分際，而以客戶的評估作爲主要的訊息回饋。因此，該模式對於系統外界環境的敏感度相當高，它在於強調系統發展的模式必須隨時因應外在環境面的挑戰。

至於內螺旋模式方面，觀念著重在逐步細緻化(refinement)的演進上。從團隊運作面來看，它代表多個同步團隊協力發展軟體系統(在本模式中，即是物件團隊、功能團隊、品管團隊)。這些協力團隊在系統發展過程中，藉著不斷的訊息傳遞，有效的將系統逐步由概念落實爲真正的系統。從方法運用面來看，內螺旋模式綜合多個物件導向的技術和觀念，使之融合在一個同步化系統開發的規範中。

由上述對於內、外螺旋模式的特性說明中，我們可大致歸結出雙螺旋模式的功能(如表 1)。

表 1 外螺旋模式與內螺旋模式的主要功能

	外螺旋模式	內螺旋模式
主要功能	(1) 因應、過濾外界的需求變動。 (2) 提供內螺旋模式平順的開發環境。	(1) 處理剩餘的變動需求。 (2) 克服同步協調的技術細節。

參、外螺旋模式

圖 2 爲外螺旋某個循環的流程圖。由圖中可以看到，整個循環是由三個環境因素作爲輸入的起始。經由這樣的輸入，會初步的得到欲開發系統的目標、定義、範圍及問題描述，再透過風險分析的策略擬定後，最後會獲得系統開發的正式需求架構。

一旦軟體系統的需求架構出來後，便會交由同步發展團隊進行系統的實際開發，而這也是內螺旋模式

運作的核心部份。首先，將需求架構中每個子功能視爲一個使用個案(USE CASE)[8]，開始進行使用者觀點與系統間的互動模擬，在此階段，整個系統的使用個案皆可同步進行。使用個案代表的是未來使用者眼中的「真實系統」，所有開發團隊都應以實現這樣的系統外觀爲首要任務，所以熟悉使用個案的運作情況，是每一個團隊的共同要求。同步發展團隊是軟體系統開發的生產群體，也是下一節內螺旋模式的論述對象，它包括一個物件團隊、多個功能團隊、及一個品管團隊。

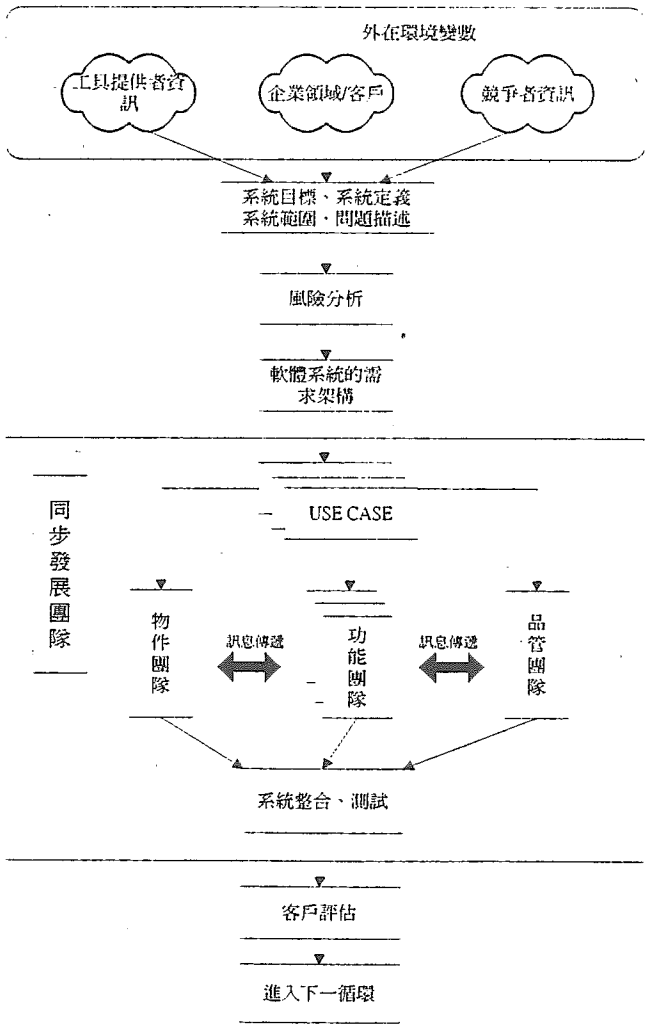

圖 2 外螺旋流程圖

在物件團隊方面，它負責將每個使用個案中的「物件成份」識別出來，並運用物件導向技術加以綜合、組織。其示意圖如 3 。

在功能團隊方面，主要的任務是利用物件團隊所識別出的物件，將使用個案實作成可執行的軟體系統。每一個功能團隊可能負責一或多個使用個案，其目標是能夠在最緊湊的排程下，獲致最快速的開發時間。

在品管團隊部份，其任務是將功能團隊所開發出來的子系統保持在最穩定的狀態中，它仿效 JIT 的及時化觀念[1]，隨時將各個階段性成果進行品質稽核，一旦發現有誤，立即回饋要求修正。

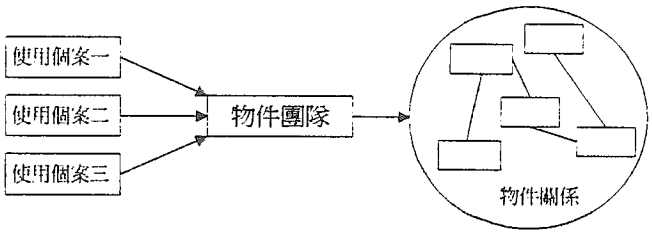

圖 3 物件團隊將使用個案轉換成物件關係

圖 4 顯示出上述團隊間的回饋流程。當物件團隊自使用個案篩析出各類物件後，功能團隊即利用這些物件，並以使用個案爲藍本，「組裝」出使用個案所模擬的系統功能。

當功能團隊產生出階段成果後，品管團隊隨即進行功能的檢核、測試，在檢核方面，著重於系統功能是否完全吻合使用個案所表現出的語意空間；在測試方面，則是檢驗系統執行過程的穩定度。一旦發現有誤，立即透過此回饋流程進行系統修正。

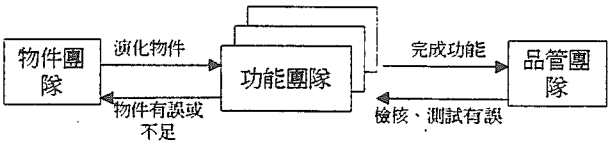

圖 4 團隊間回饋流程

由於物件團隊所建構的各類物件爲所有的功能團隊共享，是以當物件有任何改變時，亦應通知其他團隊。

由上述可知，同步發展團隊中的各團隊之間，具有非常緊密的合作關係。物件團隊居於此關係的樞紐地位，將使用個案中個別呈現的系統觀點以物件的角度做整合；而各個功能團隊則圍繞在物件團隊四周，藉著取用物件團隊所生產的物件進行系統組裝；品管團隊則扮演後勤支援的角色，爲功能團隊提供一個成

果累積的背書保證(圖 5)。

當功能團隊將其所負責的子系統完成後，即需要將其組合成一個完整的系統。系統一旦整合完成後，便須遞交給客戶使用。在使用過程中，必定會發現許多待改進的部份，而此類訊息的回饋，將成爲進入下一個外螺旋循環的起點。

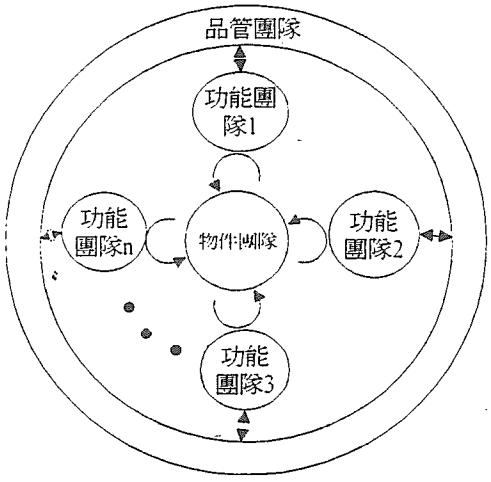

圖 5 同步發展團隊內的團隊關係

肆、內螺旋模式

之前我們在圖 1 顯示外螺旋及內螺旋關係的示意圖，並在論述外螺旋模式的同步發展團隊中，將團隊間的合作關係描繪在圖 5 。現在，我們將兩者間的關係說明如下：

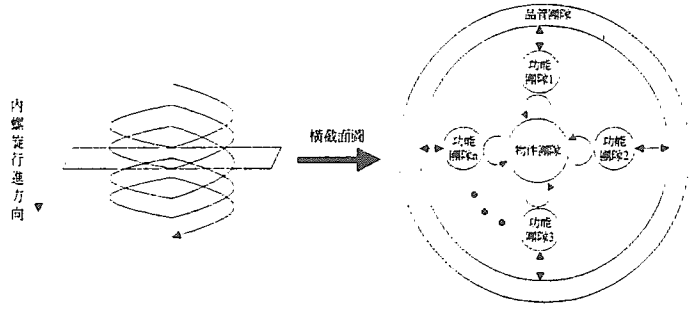

圖 6 內螺旋與同步發展團隊關係圖

在圖 6 中可以看到，同步發展團隊可以從內螺旋模式盤旋行進時的截面圖觀察出來。在同步發展團隊中，團隊間經由密集的訊息溝通及分工合作，一步步的將系統由概念落實爲真正的系統。而當我們將這些團隊間的互動以動態的方式看待時，其系統演進的軌跡即是圖 6 左方的螺旋路徑。

以下分成兩個小節，分別敘述內螺旋模式的核心觀念及運作細節。

4-1 、核心觀念--演進動力

在之前闡述同步發展團隊的時候，我們得悉軟體系統的開發主力來自於物件團隊及功能團隊的密切合作，在合作間的重要傳遞物即是由使用個案中所識別出的物件成份；事實上，物件除了擔負模塑企業功能的任務外，更扮演了子系統間的一致性確認的積極角色。

在我們所建構的同步發展模式中，強調的是自需求分析開始，即可由功能團隊獨立的開發各自的子系統，爲免相互干擾，在機制的設計中，我們不允許功能團隊之間從事訊息的傳遞；而由此相對衍生的一致性問題，則由物件團隊所維護的物件來負責(圖 7)。

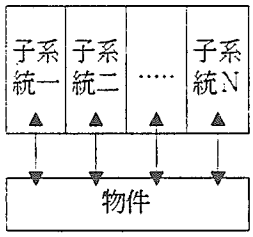

圖 7 子系統間的一致性可由物件層次確認

由上圖可知，所有功能團隊在發展各自的子系統時，皆必須取用來自物件團隊所篩析出的物件。是以只要在物件層次上作好物件演化的維護工作，那麼對於日後整合階段的一致性衝突問題，應可降至最低。

另外，由歷來軟體系統的發展經驗得知，系統需求的獲取並不是一次就能夠完整的，它必須不斷的由使用者處加以確認與修正，是以系統需求的發展形態是演進式的。而在我們的模式中，使用個案是系統需求的表現方式，它演進的目的自然必須反映真實世界的需求；再則，物件間的關係由於是由使用個案所提供的，是以使用個案的演進會連帶的影響物件的靜、動態結構(亦即物件模式、動態模式)的演進。

從物件團隊及功能團隊的任務導向中，可以清楚的觀察到使用個案及物件關係演進的動力來源。

就物件團隊來講，其任務是將每一個使用個案中的物件尋找出來，並建立起其間的關係。它必須持續監控使用個案的最新演進狀況，並適時的反映在物件模式及動態模式中。有關此任務的特性我稱之爲平衡導向(balance-oriented)(如圖 8)。

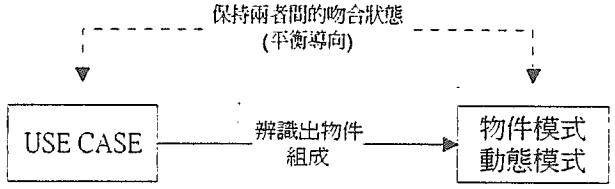

圖 8 物件團隊的任務示意圖

就功能團隊來講，它是一個求解導向(solution-oriented)特性的任務團隊。它的進行方式是以使用個案作爲系統建構的藍圖，而以物件作爲實現的工具，其目標是將子系統真正的實作(implement)出來，所以它在整個思惟模式方面，皆圍繞在實作的可行性上。若發現不可行時(如需求不明確)，即透過制式的機制回饋訊息(如要求進一步的明確化需求)。

如果該回饋訊息的請求被接受，則會進一步衍生出更爲細緻的使用個案。如此將會破壞物件團隊先前建立的使用個案/物件間的平衡狀態，而迫使物件團隊進一步的細化物件模式及動態模式。

就功能作用來看，功能團隊實質上扮演著瓦解使用個案和物件之間的平衡關係(如圖 9)。如此兩個團隊一來一往，即會使得每個子系統朝向真正的實際系統邁進。

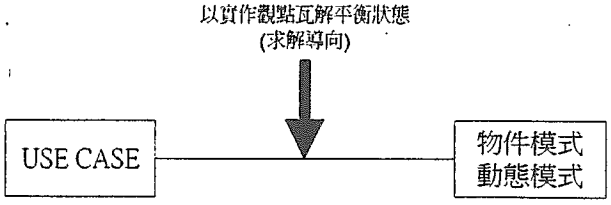

圖 9 功能團隊的任務示意圖

4-2 、運作細節--反覆路徑

圖 10 顯示出內螺旋模式運作細節的流程圖。由圖中得知，軟體系統經由外螺旋模式的風險分析確定出需求架構後，即可依此從事使用個案的建置工作。如此即能啓動以同步發展團隊爲核心的內螺旋開發活動。

在使用個案左側，標示出物件團隊進行物件模式建構的工作流程，其「 A 、 B 、 C 、 D 」流程記號者，代表的是流程進行的順序。而此流程的最後目的，則是建立起使用個案所對應的物件模式[10]及動態模式。

由上一節的圖 3 可以得知，物件團隊必須將每一個使用個案所篩析出的物件共同融合於同一個物件模式及動態模式中，所以從每個使用個案所獲得的物件關係不一定是最終的確定關係。

在使用個案右側，標示出功能團隊在實作子系統時的工作流程，其「 1 、 2 、 3 、 4 、 5 、 6 」流程記號者，代表的是流程進行的順序。

功能團隊基於實作的觀點，綜合「 1 、 2 、 3 」的資料來源，進行系統的實際開發，如果三方向的資訊及工具齊全，即可順利開發出可執行的系統；反之，若不足時，則透過「 4 、 5 、 6 」的工作流程，進行需求再細化的回饋流程。如此亦間接啓動了物件團隊對於物件關係的再建構工程。

伍、結論

本文針對同步化的軟體系統開發流程，提出一個實質性的運作模式。該模式稱之爲雙螺旋模式(double spiral model)，區分爲外螺旋模式(outside spiral model)及內螺旋模式(inside spiral model)(參考圖 1)。

外螺旋模式著重於因應環境對於系統規格的變動上。透過模式中的風險分析，它能夠適當的過濾外界的變動需求，並提供內螺旋模式中的同步發展團隊一個平順的開發環境。

內螺旋模式則是處理某些剩餘的變動需求，並描述如何確保品質、克服協調的運作細節。其間包括一致性的檢核方式、同步開發時的運作機制等等。特別就物件團隊與功能團隊各自所扮演的角色來看，它們之間存在一種既對立又合作的吊詭(paradox)關係。

就對立關係來講，物件團隊著力於平衡使用個案與物件靜、動態模式間的對應關係(圖 8)；而功能團隊則是站在實作的觀點上，企圖瓦解使用個案與物件靜、動態模式間的平衡狀態(圖 9)。兩個團隊各依本

身的任務特性反覆運作。

就合作關係來講，兩個團隊雖然反覆的在平衡/瓦解的慣性動作中一來一往，但每一次的往返週期卻也產生出更爲精緻的使用個案與物件靜、動態模式。站在系統發展的角度來看，可視爲一種良性的合作關係。

經由物件團隊及功能團隊間的密切合作，軟體系統之一致性、整合性的品質問題，可獲得一相當的保證，而此保證進而能夠使得多個功能團隊之間真正的進行同步化的系統開發，並在此過程中可望衍生出縮短產品發展週期的擴大效益。

誠如 Brooks[5]認爲，人力盲目的大量投入，並無法保證系統能夠如期或提早完工，更重要的是，必須對此人力建構出更爲精巧的協調機制，方能在確保軟體品質的前提下，發揮出人力資源數量與系統發展週期之間的交換利益。本文之雙螺旋模式即是依上述理念，將多團隊的軟體開發流程置於一個重覆性的運作架構中。

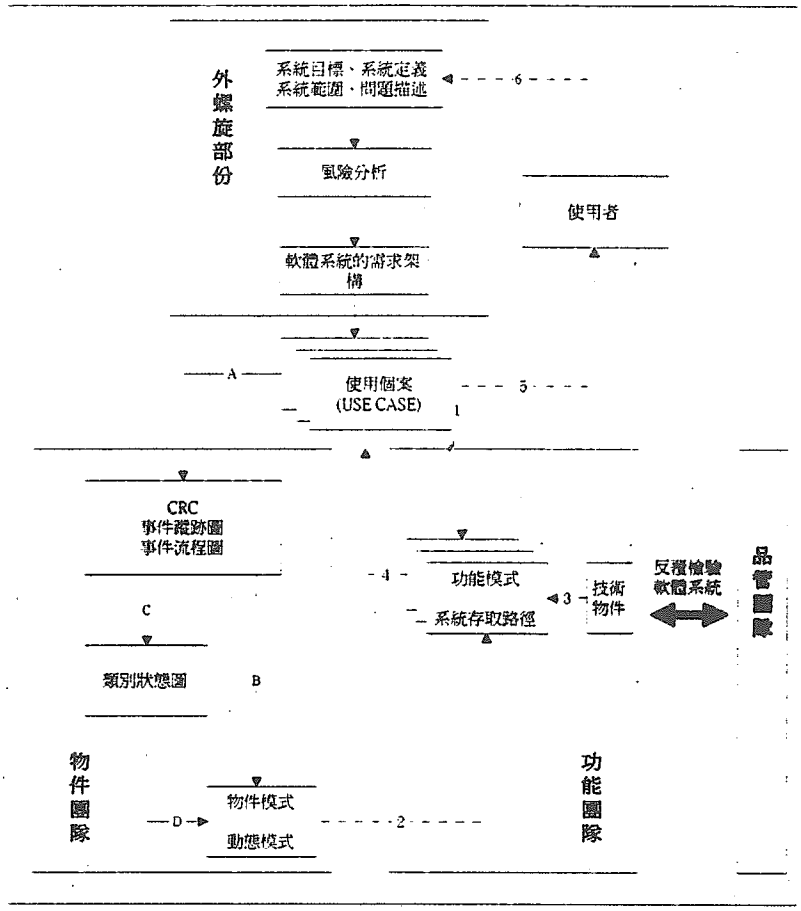

圖 10 內螺旋模式流程圖

參 考 文 獻

[1] 賴士葆, 生產/作業管理---理論與實務, *華泰書局*, 1993.

[2] Aoyama Mikio, Distributed Concurrent Development of Software Systems: an Object-Oriented Process Model, *IEEE Comput. Soc. Press*, 1990.

[3] Aoyama Mikio, Management of Distributed Concurrent Development for Large-Scale Software Systems, *IEEE Comput. Soc. Press*, 1995.

[4] Boehm B. A Spiral Model for Software Development and Enhancement, *Computer*, Vol. 21, No. 5, May 1988, pp. 61-72.

[5] Brooks F., The Mythical Man-Month. *Reading, MA: Addison-Wesley*, 1975.

[6] J-Y Chen, C-P Lai, An enactable software process modeling approach, *Information and Software Technology*, 1993.

[7] Heimann Peter, etc. DYNAMITE: Dynamic Task Nets for Software Process Management, *IEEE Comput. Soc. Press*, 1996.

[8] Jacobson I., Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1992.

[9] Perry D.E., A. Porter, L.G. Votta Evaluating Workflow and Process Automation in Wide-Area Software Development, *http://lsdis.cs.uga.edu/activities/NSF-workflow/s.html#ieee* 1989, 1996.

[10] Rumbaugh J., etc., Object-Oriented Modeling and Design, *Prentice-Hall, Englewood Cliffs, New Jersey*, 1991.