「소프트웨어」開發管理를 위한 效率的 方法 硏究

A Study on a Efficient Method for Software Development

趙 敏 元*

Abstract

The development of Software is indispensably needed for the Computer development in the future, that is, a effective method for Software development is required to be studied.

In this paper, a detailed and practical method for the making of the requirements and planning for Software development and efficent ways for Software development and manapement are studied. This paper is considered to be helpful for the efficiency in Software development works.

1. 序 論

現代社會는 情報處理 및 通信技術의 急速한 發展과 함께 高度情報社會로 變貌해 가고 있으며 「컴퓨터」가 그 核心을 이루고 있다. 「컴퓨터」分野는 특히 Hardware 分野의 發展이 그 主軸을 이루고 있으며 利用技術인 Software는 이에 비해 매우 뒤져있는 實情이다. 이같은 現象은 硏究開發된 Hardware 技術은 累積的으로 蓄積되고 이는 새로운 發展을 위해 再活用되고 있으나 「Software」는 旣開發된 技術이 效率的으로 再活用되지 않는데 그 原因이 있다.

따라서 高度情報社會의 要求에 부응하기 위해 「소프트웨어」의 效率的인 開發問題는 매우 重要한 課題로 提起되고 있다. 本硏究에서는 「소프트웨어」의 生産管理 全過程에 걸쳐 主要段階를 구성하는 「소프트웨어」의 要求定義 및 開發計劃의 作成과 그의 標準化와 開發管理 및 保守管理 側面에서 實踐的 方法에 重點을 두어 硏究·檢討하도록 하겠다.

2. 「소프트웨어」의 要求定義

「소프트웨어」의 成功的인 生産은 그의 正確한 要求定義를 必要로 한다. 豫算과 期間이 確定된 狀況下에서 不明確한 要求內容에 따른 「소프트웨어」開發의 着手는 失敗의 主要한 原因이 된다.

要求定義에는 使用者 部門이 主體가 되어 使用者의 必要, Top 關聯部門의 方針等의 配慮下에 「시스템」開發要求가 明確히 記述되어야 하고 EDP 部門 專門家에 의한 妥當性檢討와 그에 따른 「시스템」開發의 可否 및 開發의 優先順位等에 관한 要求分析이 包含되어야 한다. 要求定義는 다음과 같은 特性[1]이 要求된다.

① 完全性 및 一貫性
② 正當性 및 檢査容易性
③ 追加可能性 및 傳達性
④ 設計와의 獨立性
⑤ 「모듈」化 可能性
⑥ 自動處理性 및 實現可能性

2.1 「시스템」化 要求

「시스템」化 要求段階에는 最高責任者의 方針 및 最終使用者의 意見이 反映되어야 하고 文書化에는 다음 內容의 包含을 要한다.

* 명지실업전문대학 전자계산과 조교수.

1) 李東郁外 編譯, 소프트웨어 工學 (서울 : 尙湖社), p.109～110.

2.1.1 「시스템」化 內容

「시스템」化 內容에는 그의 目的, 背景, 必要性, 效果와 함께 運用時期等에 關한 詳細한 說明이 要求된다. 背景에 있어서는 事務 「서비스」의 改善要求, 業務量, 事務員의 增加要因, 業務內容, 流通過程, 最終使用者가 直面하고 있는 背景과 改善方法 및 當面問題點, 「시스템」化에 이르게된 經緯等을 明確히 하고 運用時期에 있어서는 業務의 優先度, 緊急度, 開始時期의 制約(法, 制度) 等과 함께 必要로 하는 希望時期의 設定을 要한다.

2.1.2 適用範圍

① 對象業務範圍 : 「시스템」에서 對象으로 하는 業務內容과 處理範圍를 分明히 하고 他業務와의 關聯 및新規業務에 대한 豫測等을 明確히 한다. 組織 및 制度의 新設 또는 改正에 따른 影響度, 對象業務의 重要度, 優先度, 緊急度等을 明示하고 關聯部門 및 他業務에 대한 影響 및 波及度를 調査檢討하고 특히 關聯部門과의 合意가 요구된다.

② 手作業과의 「인터페이스」: 設定된 對象業務와 事作業과의 境界, 情報의 授受方法 및 그 手段을 明確히 한다. 事務處理過程과 樣式類의 對應關係, 手作業으로 處理된 範圍 또는 改善된 範圍를 事前에 充分히 檢討하고 使用者部門의 統一된 見解를 제시한다.

③ 機能擴張方針 : 將來 「시스템」의 擴張 및 新規業務等에 의한 機能擴張方針을 明確히 한다. 將來 「시스템」 擴張이 豫測되는 경우 그 擴張度를 檢討하고 大幅的인 變更에도 對處 可能하도록 考慮를 하여야 한다.

2.1.3 關聯情報 一覽

① 入出力情報 一覽 : 對象業務에 關聯되는 모든 入出力情報의 屬性(情報名, 作成日字, 發生源, 資料量, 關聯資料, 機密保護의 必要) 等을 明確히 한다. (그림 1 참조)

分類「코드」	入出力情報關聯分析表										페지
「시스템」名	作成者名										作成年月日
情報項目名 / 出力情報名	所屬番號	社員番號	姓名	勤務狀況	勤務時間	生年月日	住所	配偶者名	入社年月日	家族數	職級
社員台張	✕	×	×			×	×	×	×	×	×
給與台張	×	×	×	×	×	×				×	×
給與支給明細	×	×	×								×

〈그림 1〉 入出力情報関聯分析表

② 情報屬性間의 關係 : 入出力情報 書式類의 型式 및 制約을 分明히 하고 業務特性에 따른 各種書式 및 圖表等을 修正하여 必要最少限의 情報項目을 設計하고 相互關聯을 明確히 한다.

③ 制約條件 : 時間的條件(資料作成期間, 應答待期時間의 許容度), 經濟的條件(開發運用經費) 및 組織的條件(地理的制約, 分散管理狀況, 管理體係等)을 明確히 한다. 특히 必要條件 및 制約條件은 最終使用者 各部門으로부터 提出받아 必要하다고 생각되는 모든 것을 記述하여야 한다.

2.2. 要求分析

「시스템」化 要求內容에 대해 事務處理過程, 書式類의 統廢合, 標準化等 全般에 걸쳐 現狀調査分析을 實施하고 妥當性을 綜合的으로 檢討한다.

2.2.1 「시스템」化 要求內容分析

① 對象業務의 範圍分析 : 「시스템」化 對象으로 設定한 業務範圍가 目的 및 必要性을 具體化하는 內容인가를 檢討하고 組織, 法, 制度의 新設, 改正 및 影響度를 確認한다. 效率化, 合理化 및 省力化等의 觀點에서 對象業務內容, 處理過程, 從來의 手作業內容을 必要時 變更하고 使用者가 提起하는 問題點과 改善事項을 滿足하는 業務範圍가 設定되었는가 檢討한다.

② 手作業과의 「인터페이스」 分析 : 對象業務의 作業分担, 情報의 授受方法等 사람과 機械間의 連結關係의 妥當性을 分析 確認하고 事務處理節次와 期限의 妥當性, 사람 機械處理의 對應手順, 情報의 흐름, 授受方法의 一元化, 綜合化의 可能性 및 事務의 員滑한 實行이 可能한 體制인가를 充分히 檢討한다.

③ 機械擴張分析 : 將來 「시스템」에 있어서의 機械擴張의 必要性, 對處方法 및 內容을 分析 確認하고 使用者가 提起하는 「시스템」 및 機能擴張方針을 前提로 그 背後의 必要性, 「데이타」類의 量的膨脹을 綿密히 檢討하여 「시스템」化에 反映될 수 있는 方策을 考慮한다.

2.2.3 關聯情報分析

業務內容과 入出力情報樣式類에 關聯된 事項과 事務處理內容 및 節次를 檢討 確認하고 必要最少限의 情報樣式을 選定한다. 統發合 標準化等의 見地에서 關聯情報를 作成하고 이를 使用者에게 提案한다.

2.2.4 制約條件分析

制約條件에서 規定한 內用 및 背景이 되는 「데이타」 및 環境條件을 分析 確認한다. 必要不可缺條件과 希望的 條件, 制約條件을 만족시키기 위한 付帶事項等을 明確히 하고 所要費用檢討에 重點을 둔다.

2.2.5 「시스템」의 妥當性分析

最終使用者의 要求內容을 分析한 結果에 準해서 「시스템」 開發의 妥當性 및 그 必要性에 관해 結論을 분명히 내린다. 主要記述內容은 「시스템」化 要求案으로서의 可否, 要求內容의 妥當性, 改善內容의 設定이 된다.

2.3. 「시스템」 分析

要求分析에서 행한 調査分析結果를 基礎로 하여 얻어진 類似「시스템」과 最新의 技術動向等을 參考하여 使用者部門의 要求案에 대한 代替案을 作成하고 環境條件을 勘案하여 이들을 比較檢討하여 「시스템」 案을 作成한다.

2.3.1 類似「시스템」 및 技術動向 調査 類似「시스템」의 特質(對象範圍, 機能, 効果, 規模, 經費, 期間, 使用技法等) 및 그의 長短點을 調査分析하고 그 結果를 「시스템」 案에 反映될 수 있도록 明確히 한다. 「시스템」化에 關聯된 各種技術을 調査하고 그의 特徵 및 動向等을 明確히 한다. 특히 先進的 使用者의 事例, 各種 文獻 및 海外資料等을 參考로 하고 廣範圍하게 該當「시스템」 設計에 有効한 技術을 吸收하여 「시스템」에 反映한다.

2.3.2 環境條件分析

環境條件은 「하드웨어」, 「소프트웨어」 및 制約條件에 의한 側面에서의 分析이 要求된다.

① 「하드웨어」: 「시스템」의 目的 및 必要條件의 實現을 위한 機器構成을 明確히 하고 「시스템」 擴張計劃에 對應하고 業務規模 및 處理內容에 適合한 機器를 選擇한다.

② 「소프트웨어」: 「시스템」의 目的과 特質에 따른 「소프트웨어」 選擇方法을 明確히 한다. 使用「프로그램」 言語, 旣存「프로그램」 言語의 利用, 專用 및 汎用「프로그램」 言語 選擇等에 관해 記述한다.

③ 制約條件: 時間的, 技術的 組織的條件으로서의 最終的인 制約條件을 明確히 設定한다.

2.3.3 代替案作成 및 比較檢討

① 機能 및 性能: 要求案과 代替案間의 機能 및 性能을 比較 檢討, 評價하여 그 結果를 明確히 한다. 특히 對象業務의 範圍, 機能 處理結果를 記述한다.

② 經濟性: 開發經費, 運用經費를 中心으로 經濟性을 比較, 檢討한다. 要求內容에 따른 「시스템」의 理想形態, 必要最少限의 構成形態等을 考慮하여 機能과 適用範圍를 設定한다. 經費와 工期見積은 機能別로 分割作成한다.

③ 効果 및 技術分析: 「시스템」化의 効果를 管理, 經費 및 業務面에서 比較, 檢討 評價하고 適用對象技術을 比較檢討한다.

④ 類似「시스템」과의 對比: 類似「시스템」의 調査結果를 如何히 「시스템」에 對應시키고 있는가를 檢討하고 그 結果를 明示한다. 주로 企劃·設計部門에서 提起한 複數의 代替案에 대하여 機能, 性能, 經濟性 効果 및 技能面에서 分析하고 이를 類似「시스템」과 對比 檢討한다. 開發難易度는 充分히 豫測해서 그 對策에 대한 內容을 檢討한다.

2.4. 「시스템」 案 確定

代替案의 比較 및 檢討結果에 의거 綜合的으로 가장 유리한 「시스템」 案을 作成하고 그 目的과 効果를 明確히 한다. 「시스템」의 業務範圍는 對象業務名, 槪要範圍, 手作業과의 「인터페이스」에 「重點을 두어 記述한다. 本 「시스템」 案은 「시스템」化 要求段階, 要求分析段階 및 「시스템」 分析段階의 最終的인 結果가 된다. 「시스템」의 槪要設計는 入出力情報의 基本機能, 構成 및 處理槪要를 「시스템」의 規模와 機能에 따라 記述하고 開發 및 運用에 必要한 經費, 工期 및 所要人力을 明確히 해야 한다.

3. 「소프트웨어」 開發計劃의 樹立

3.1 予算과 計劃

「소프트웨어」 開發事業의 主要失敗原因은[2)]

① 要求仕樣의 途中變更 (約 70%)

② 見積의 잘못 (約 60%)

③ 無理한 計劃(約 40%)에 있는 것으로 1981년 日本情報處理開發協會의 調書에서 밝혀졌으며 美國聯邦政府의 「프로젝트」에 대한 原因調書에서는 計劃上의 不備가 50%이상이라는 報告가 있었다. 「소프트웨어」 開發計劃을 成功시키기 위한 重要한 要件中의 하나는 可能한 한 正確하고 彈力性을 갖는 計劃을 立案하는 일이다. 「소프트웨어」 開發計劃의 類型과 見積作成技法에 관해 檢討하도록 하겠다.

3.1.1 豫 算

豫算은 作業量 産出에 따른 見積에 따라 다음과 같은 費目으로 區分하여 作成한다.

① 直接材料費

② 直接勞務費

③ 「소프트웨어」 購入費

④ 直接經費

⑤ 間接材料費

2) ソフトウエアの開發計劃, ユンピユートピア (1985.10), p.173.

⑥ 間接勞務費

3.1.2 計 劃

「소프트웨어」 開發計劃은 日程計劃, 外注時 (開發의 一部)는 外注計劃, 品質管理計劃, 要員計劃, 移行計劃, 運用計劃 및 修理保守計劃등으로 區分作成하고 그 內容은 具體的이고 詳細해야 한다.

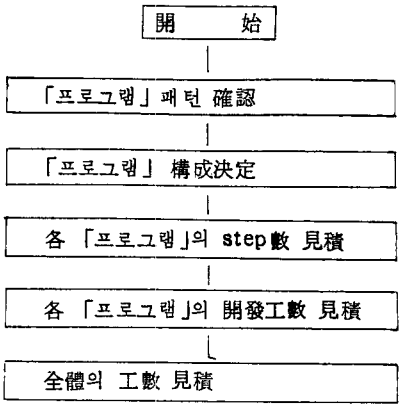
특히 日程計劃은 全體의 工程把握, 遲延不可作業(critical path)의 明確化를 위해 必要不可缺하며 「퍼트」나 「마일스톤챠트」 等의 技法을 適用해서 作成하고 移行計劃은 業務의 一時的인 變化를 적게 하면서 점진적으로 移行되도록 하고 移行을 위한 日程, 教育, 作業計劃等이 要求된다.

現在 全「컴퓨터」 人力中 60% 정도가 維持 保護作業에 使用되고 있을 程度[3]로 「소프트웨어」의 修正作業은 막대한 比重을 占有하고 있다. 따라서 保守計劃은 錯誤에 대한 修理保守와 計算機의 交替等에 따른 適應保守 및 機能의 豫見되는 追加나 性能提高等에 따른 完全化保守計劃의 作成을 要한다.

3.2 見積作成

見積作成은 文献上의 「모델」이나 數値를 있는 그대로 適用해서는 안된다. 開發技術者의 技術水準 및 開發環境에 맞게 作成되어야 하며 「소프트웨어」의 規模, 技術的 難易度, 担當者의 經驗 및 技術水準, 電子計算機의 規模 및 使用形態에 따라 見積內容은 달라지게 된다.

以上의 內容을 定量化하고 見積「모델」을 作成한다. 「모델」 作成者와 利用者間의 方法의 差가 있으므로 他人이 作成한 「모델」을 그대로 適用하지는 말아야 한다. 定量化가 어려운 것은 크게 2～3個의 Group으로 區分시키고 過去 「프로젝트」에 대한 類似한 類型들의 生産性을 見積作成에 活用함이 좋다. 過去의 「데이터」로부터 計算한 生産量을 利用한 見積作成 手順은 다음과 같다.



〈그림 2.〉 見積作成 手順度

① 「프로그램·패턴」 確認 : 該當 「프로그램」이 어느 「패턴」에 屬하는가를 定하고 決定된 「패턴」에 관한 「데이터」가 없을시는 가장 가까운 「패턴」을 擇한다.

② 「프로그램」 構成決定 : 「소프트웨어」가 어떠한 「프로그램」들로 構成되는가를 檢討한다. 「프로그램」들은 同一水準에서 생각하는 것이 아니고 見積이 어려운 部分은 可能한한 細分化시킨다.

③ 「프로그램」 step數 見積 : 3～4名이 다음과 같이 3個의 推定値를 定한다.

a : 可能한 最少의 크기

b : 생각되는 最大의 크기

m : 中央値

各者의 平均을 取하고 다음 計算式에 의해 該當 Program의 step數를 算定한다.

$$\text{「프로그램」 step數} = \frac{a + 4m + b}{6}$$

各者의 a, m, b 値의 差가 큰 경우에는 對話를 통해 各者의 算定理由等 意見을 나누어 可能한한 그 差를 接近시킨다.

④ 「프로그램」 開發工數의 見積 : 各 「프로그램」의 step數를 生産性으로 나누어 「프로그램」 開發工數를 計算한다.

$$C_i = \frac{E_i}{f}$$

C_i : Program의 開發工數

E_i : Program의 step數

f : 生産性

⑤ 全體의 工數見積 : 各 「프로그램」마다의 工數를 加算하여 全體工數의 見積을 얻는다.

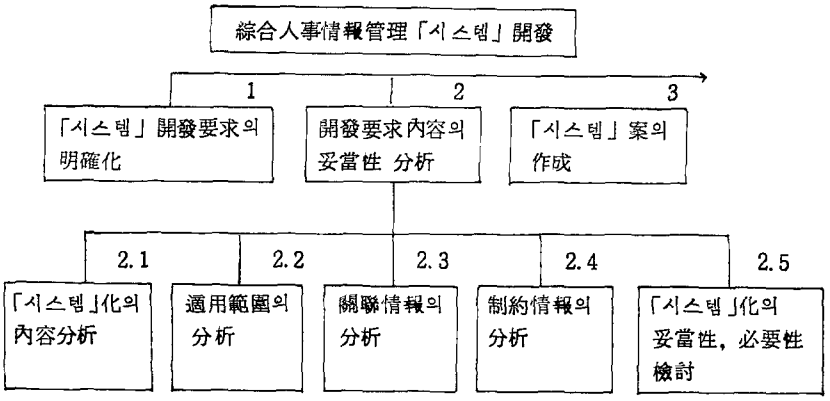
3.3 開発計劃의 樹立技法

3) 李基式外 소프트웨어生産技術 (서울 : 正益社), p.198.

「소프트웨어」의 生産管理 效率을 높이고 計劃完遂를 위해서는 우선 計劃 자체가 잘 되어 있어야 한다. 올바른 計劃入案을 위한 技法 혹은 手段에 대한 理解와 이들의 適用이 要求되며 그 一部를 檢討해 보겠다.

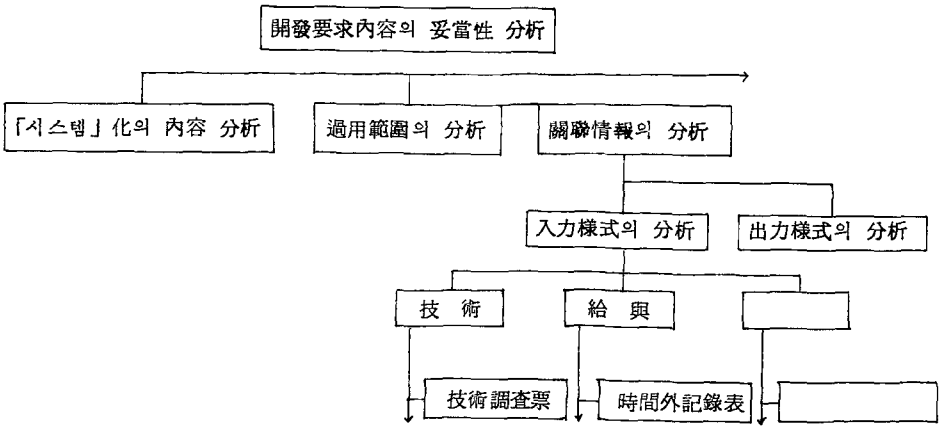
3.3.1 WBS (Work Breakdown Structure)

「소프트웨어」 開發過程에서 全作業段階의 技能을 고려하지 않고 計劃을 임의로 延期시켜 開發計劃의 차질을 초래하는 경우가 많다. 이에 대한 對處方法으로 活用되는 技法이 WBS 인데 이는 作業을 Top-down 方式으로 分析하고 階層的으로 表現한 것이다. 다음 〈그림 3〉은 綜合人事情報管理「시스템」의 開發을 위해 제일 먼저 作成되는 Project WBS 이고 〈그림 4〉는 〈그림 3〉에서 提示된 다음 段階의 두번째 作業의 WBS 이다.



〈그림 3〉 **Project WBS의 例**

WBS의 效果는 作業의 一覽表로서 現在 해야 할 일이 어떠한 作業으로 構成되어 있는가를 알 수 있게 되며 事前에 計劃을 補完할 수 있게 하고 두번째 表現된 個數의 作業은 PERT의 「액티비티」가 되고 다음에 설명되는 TRM에 의해 要員 各者의 責任 또는 役割의 明確化가 可能해지며, 세째 個數의 作業에 따른 工數見積을 하고 그의 正確을 기하기 위한 明細의 作成이 容易하다.



〈그림 4〉 **Phase WBS의 例**

보다 效果를 높이기 위해서는 다음과 같은 點에 留意해서 WBS 를 作成해야 한다.

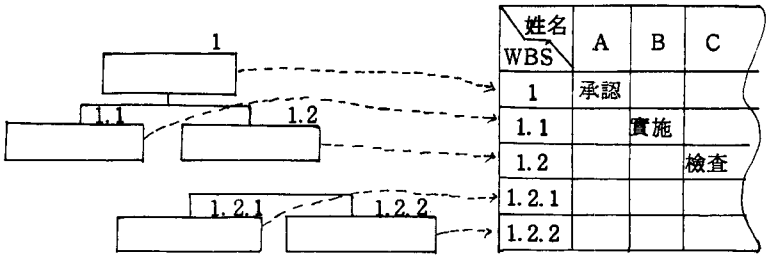
① Project 途中에 作成되는 文書化의 名稱, 構成을 標準化시키고 使用者 參照書, 檢查仕樣書, 檢查結果書, 保守內容內譯書 등을 作成한다.

② 細部的인 作業, 특히 「모듈러·프로그램」 活用時는 每 「모듈」別 테스트, 綜合 테스트 및 運用 테스트를 實施하고 「테스트 데이터」는 充分히 準備한다. 1個 「모듈」에 6個의 1F 文이 있다면 26種의 「테스트 데이터」의 作成이 要求된다.

③ Project에서 使用하는 O.S, 「컴퓨터」 및 DBMS 言語等에 관해 開發要員들이 充分한 知識을 갖고 있지 않을 경우는 敎育과 學習을 위한 工數가 必要하며 計劃 및 評價作業도 包含되어 있도록 하여야 한다.

3.3.2 任務·責任 關聯圖 (TRM : Task Responsibility Matrix)

TRM (그림 5, 그림 6[4] 參照)은 WBS에 나타난 作業의 推進과 關聯되는 사람과 그의 責任 및 權限關係를 Matrix로 表現한 것이다. TRM의 效果는 各者의 役割分担이 明確해지고 Project의 最終使用者, 外注先, 他部門이 關係하고 있을 때 各者 責任의 明示가 可能하고 適材適所的인 作業割當이 可能하다.



〈그림 5〉 WBS 와 TRM과의 관계關係

WBS 番號	作業內容	管理者 金	SE 朴	SE 崔	SE 李	P 文	P 尹
3	開發要求內容分析	承認	檢查				
3.1	關聯情報分析		承認				
3.1.1	入力情報分析			承認			
3.1.1.1	技術情報			實施	實施		
3.1.1.2	給與情報			實施	實施		
3.1.1.3	家族情報			實施	實施		
3.1.2	出力情報分析		承認				
3.1.2.1	部門別社員一覽表		實施			實施	
3.1.2.2	技術別社員一覽表		實施			實施	
3.1.2.3	給與明細表		實施			實施	

〈그림 6〉 TRM의 例

3.3.3 「미일스톤 챠트」 (Milestone Chart)

一般的으로 使用者가 가장 큰 困難을 느끼는 것은 進捗管理이다. 그 理由中의 하나는 單位作業의 마감일자에 대한 解釋의 差異이다. 예를 들어 終了日字가 11月 10日일 때 사람에 따라 11月 10日 끝나면 좋을 것이다. 11月 8日項 끝나도록 하겠다. 이는 어디까지나 計劃上 豫定된 날짜일 뿐이다. 혹은 11月 10日은 管理者의 最後 豫想日字이다 等과 같이 相異할 수 있다. 이같은 自由로운 解釋은 進捗管理를 어렵게 만드는 要因이 된다. 따라서 日字는 다음과 같이 明確히 區分하여 表現하여야 한다.

① 豫想日 : 計劃이 순조로이 實行될 때 豫想되는 日字

② 最早日 : 가장 빠르게 作業이 終了되는 경우의 日字

③ 最遲日 : Project가 計劃대로 終了되기 위해 반드시 作業의 終料가 要求되는 日字

④ 豫定日 : 計劃上의 日字

⑤ 約束日 : 使用者가 最高責任者에 대해 公式的으로 約束한 日字

「미일스톤 챠트」는 〈그림 7〉, 〈그림 8〉과 같이 表現되며 進捗管理上의 主要工程內容과 앞에서 檢討된 마감일자가 表示되어 있어서 日字에 대한 誤解를 없애고 進捗管理의 Check Point 定義가 可能해진다.

4) ソフトウエアの開發計劃, ユンピユートピア (1985. 11號), p. 170.

「프로젝트」名(코드) : A 3140
管理者名 :
細部作業名 : 要求分析
作成年月日 : 87. 3. 2

承認	担當

上位WBS番號	WBS番號	作業內容	主担當者	4月 5 10 15 20 25日
1.1	1.1.1	給與業務	홍길동	⚠ ⚠ 3

미일스톤番號

〈그림 7〉 「마일스톤·챠트」의 例

「프로젝트」名 : A 3140
「마일스톤챠트」코드 : 34215

上位WBS番號	WBS番號	마일스톤番號	마일스톤·이벤트	체크內容	체크担當者	日字 豫想日	最早日	最遲日	豫定日	約束日	實際日
1.1	1.1.1	1	作業開始		○	4/7	4/6	4/8	4/7	4/7	
		2	使用者A의 調査終了	要求精報	○	4/16	4/14	4/17	4/16	4/17	
		3	使用者B의 調査終了	要求「스피드」	○	4/23	4/20	4/23	4/23	4/30	

〈그림 8〉 마일스톤 이벤트 說明書의 例

3.3.4 PERT

經驗이 적은 사람과 熟練된 사람이 있을 때 어려운 일은 熟練된 사람에게 負担시키는 경우가 흔히 있으나 이같은 措置는 잘못된 일이다. 「소프트웨어」 生産管理의 觀點에서 보면 늦어서는 안되는 工程을 能熟한 사람에게 맡기고 工程上 여유가 있는 일은 未熟한 사람에게 맡겨야 한다. 遲延되어서는 안되는 일과 다소 遲延되어도 관계가 없는 일을 區分하고 이를 明確히 圖表上에 表現하기 위해서는 PERT技法을 必要로 한다. PERT 圖上에 크리리컬 패스(Critieal path)는 결코 늦어서는 안될 일이며 그 外의 일들은 餘有工程들이다. PERT는 現在 「소프트웨어」 Tool로서 많이 活用될 수 있다.

4. 「소프트웨어」 開發의 効率的方法

「소프트웨어」의 開發은 既作成「프로그램」의 再活用, 「프로그램」 作成의 標準化와 分業化에 의해

그 效率化를 期할 수 있다.

4.1 「프로그램」의 再活用

旣作成「프로그램」은 잘 管理하여 同一「프로그램」은 再作成함이 없이 再利用할 수 있는 體制를 期하고 旣作成한 類似한 「프로그램」은 積極的으로 活用함으로써 「소프트웨어」 開發의 效率化를 높일 수 있게 된다. 再活用을 쉽게 하기 위해서는 다음과 같은 方法에 따라 作成하면 더욱 좋다.[5]

② Module化 方法을 標準化하고 이에 따라 「모듈」化된 Program으로 作成한다.

② 「프로그램」은 構造化「프로그래밍」 方法을 適用하고 간단한 構造를 갖는 「프로그램」으로 作成한다.

③ 「코딩」 方式을 標準化하고 이에 따라 「코딩」한다.

④ 注釋은 理解가 容易하도록 記述하고 한글로 表記하며 「프로그램」 內容의 50%는 注譯이 되도록 한다.

⑤ 「프로그램」 作成은 意識的으로 第3者에 대한 說明에 目的을 두고 「코딩」한다.

作成된 「프로그램」은 文書化해서 管理하도록 하고 再利用이 可能하도록 다음 事項에 有意해서 文書化 作成을 한다.

① 各種의 文書化를 標準化한다.

② 文書化에 있어 新造語는 避하고 變數名, 파일名, 및 其他 名稱은 알기 쉽고 內容을 表現하는 用語를 사용한다. 同一內容의 說明이 必要時 反復해서 記述한다. 어느 곳을 參照하라는 方式은 避해야 한다.

4.2 標準化

標準化의 重要性은 매우 크며 잘 알려져 있다. 現實的으로 「시스템」의 變更이나 情報處理要員의 빈번한 轉出(프로그래머의 平均 勤續期間은 14～16個月[6]) 等에 對比하기 위해서도 標準化는 매우 堅要하다. 標準化의 實踐的 推進方法을 다음에 檢討해 보도록 한다.

4.2.1 現狀의 問題點

標準化에 있어서 提起되는 主要問題點에는 다음과 같은 事項들이 있다.

① 標準自體가 너무 理想을 추구한 나머지 그 實行에 過多한 勞力과 費用 및 時間을 要한다.

② 標準化 作成에 관한 意義와 그 具體的인 內容에 대해 敎育이 부족하다.

③ 開發하는 「소프트웨어」 內容의 差異에 대한 考慮가 되어 있지 못하다.

④ 現場技術者들의 말을 充分히 把握하지 않고 担當者에 이끌려 標準을 作成한다.

⑤ 新技法의 導入等 現狀變化에 따른 修正措置 不履行으로 現狀과 不一致한다.

4.2.2 標準化의 進行方法

標準化 作業의 推進에 있어서 配慮할 事項은 다음과 같다.

① 開發中인 「소프트웨어」 內容에 관해 實態를 把握한다.(「소프트웨어의 規模, 種類, 對象機種, 言語, O.S 等)

② 開發에 關聯된 技術者들의 技術分布를 明確히 한다.

③ 理想的인 標準案을 만든다.

④ 現場担當者의 代表를 交替해서 標準案을 檢討한다. 이때 ③項에서 作成된 案中에서 必要以上의 部分은 除去한다.

⑤ 作成된 標準案을 現場技術者에게 說明한다.

⑥ 標準의 履行을 強要하지는 말아야 한다.

不滿이나 批判은 定期的으로 수집해서 現實에 맞는 것으로 만든다.

4.2.3 標準化 項目

標準化된 項目別 內容을 具體的으로 例示하면 다음과 같다.

① 用語 : 用語의 標準化는 文書의 傳達性을 높이기 위해 매우 重要하다. 특히 各 部署名等의 「코드」化, 「파일」名, 「프로그램」名, 入出力媒體名, 專門用語等의 標準化는 重要하다.

② 開發工程別 作業內容 : Project 管理를 統一的으로 效率性있게 遂行하기 위해서는 開發工程의 區分方法 및 各 工程內 作業內容을 標準化시켜야 한다.

③ 「코딩」 方法 : 「프로그램」의 文書化는 「코딩」의 標準化가 不可缺하다. 注譯의 記入方法, identation의 方法, 變數名, 불럭名, 定數名等의 賦與方法等에 관한 具體的인 標準化가 만들어져야 한다.

④ 「테스트」 方法 : 테스트는 製品의 最終的인 檢査이므로 그 方法을 標準化해 놓고 品質을 保證할 수 있도록 하여야 한다.

⑤ 保守方法 : 「소프트웨어」를 資産으로 보는 以上 그의 管理인 保守方法은 매우 重要한 業務가 된

5) Robert T. Grauer, Struetured Mothod Through COBOL (Prentice Hall Inc, New Jersey, 1983), p. 2.

6) Maxgray Keither, Documentation Standards (1969), p. 56.

다. 그의 重要內容은 다음과 같다.

4.3 分業化

「소프트웨어」의 效率的인 開發을 위해서는 作業의 分業化가 要求되고 있으며 그 例는 다음과 같다.

① 「시스템」設計作業과 「프로그램」作成作業
② Sub system에 따른 分業
③ 自體와 外注會社와의 分業
④ 開發業務와 檢査業務의 分業

5. 「소프트웨어」開發管理의 實踐的 方法

「소프트웨어」의 開發管理는 進捷管理, 費用管理 및 品質管理가 그 中心이 되고 있다. 開發管理에 있어 注意를 要하는 事項에 重點을 두어 檢討해 보도록 하겠다.

5.1 進捷管理

「소프트웨어」開發現場에서 進捷管理는 매우 어려운 部門이다. 이는 進捷管理上의 方法에 있어서가 아니라 그 前提가 되는 要件이 滿足되어 있지 않는데 그 理由가 있다.

5.1.1 進捷管理의 前提要件

① 工期 및 期間의 正確한 見積 : 「소프트웨어」開發을 主業으로 하는 企業은 흔히 事前에 納期가 定해져 있을 때 納期에 맞추어 「프로젝트」實施計劃을 作成하는 例가 많으며 注問先의 見積을 그대로 自社의 計劃으로 받아드릴 때도 있다. 이 경우는 進捷管理는 제대로 遂行될 수 없으며 納期內 作業完了란 不可能하게 된다. Project 管理에 있어서는 무엇보다도 實現可能한 工數配分을 考慮한 期間의 設定이 重要하다. Project 担當要員의 經驗과 實力에 맞추어 作業人員이 配分되어야 한다.

② 工程別 作業內容의 標準化 : 進捷管理는 定性的이 아니라 定量的으로 이루어지므로 作業內容의 明確한 定義가 要求된다. 이때 進捷率을 K/N(K : 終了된 作業量, N : 全體作業量)라는 計算式으로 表現된다. 作業의 標準化는 여러가지 意味에서 重要性을 지니고 있으며 進捷管理의 경우 進捷率은 全體가 加味된 것이어야 하기 때문에 作業의 標準化는 특히 必要不可缺하다.

5.1.2 進捷管理方法

開發現場에서는 「프로젝트」에 따라 生産되는 「소프트웨어」完成度를 工程期間의 經過日數에 比例해서 計上하거나 前後作業의 難易度가 考慮됨이 없이 表現되는 경우가 흔히 있는데 이는 크게 잘못된 일이다. 올바른 進捷管理는 終了된 作業이 아니라 남은 作業의 完成을 위해 所要되는 期間의 檢討와 期間內 作業完遂가 不可할 때 即時 이에 對處하는 것이 進捷管理의 올바른 方法이다.

5.2. 費用管理

管理者는 費用削減에 큰 比重을 둔다. 아무리 納期를 맞추고 品質이 優秀한 「소프트웨어」를 開發하여도 費用이 受注價格을 上回해서는 안된다. 대부분의 Project 開發担當官은 費用에 대해서는 關心이 적으므로 管理者는 担當官들에게 對話를 통해 機械使用時間, 테프價格, 印刷用紙價格等 費用에 관한 認識을 높일 必要가 있다. 費用管理는 見積에서 나타난 費用 및 所要豫算表에 따라 實績基準으로 豫算實績分析表[7] 〈그림 9〉를 作成해서 費用管理를 해 나가야 한다.

「프로젝트」名 ( )			月 日～ 月 日			
費目	當該期間		現在까지 累計		完了時	
	豫定	實績	豫定	實績	豫定	實績
人件費						
計算機費						
出張費						
消耗品費						
合計						

〈그림 9〉 費用管理表의 例

5.3. 「프로그램」文書化 管理

「소프트웨어」開發에 있어서 Program 文書化管理는 「소프트웨어」의 機能, 性能, 信賴性等의 特徵을 確實히 實現하기 위해서 管理對象을 明確히 하고 「프로그램」作成過程에 따라 항상 文書化와 「프로그램」을 對應시켜 相互關係를 確實히 해 나가도록 해준다. 이 作業은 標準化管理, 變更管理, 資産管理라는 3가지 要素로 構成된다. 自社에서 作成된 Program 等은 貴重한 資産이며 再利用을 위해 잘 管理되고 活用되도록 하여야 한다. Program은 그의 構成圖 및 文書化部分의 目次를 「파일」化해서 再活用이 容易하도록 管理되어야 한다.

5.4. 品質管理

品質向上은 어느 組織에서도 提起되는 問題이다. 「소프트웨어」는 使用者必要를 滿足시키고, 誤謬가 없고, 使用者側 要求變化 또는 誤謬發生時 그에

7) プロジエト上管理研究會, ソフトエア 開發管理の實踐的方法, ユンピユートピア (1986. 3), p. 162.

대한 對處가 容易하도록 누구나 쉽게 理解될 수 있어야 한다는 特性을 지니고 있다.

이같은 品質의 「프로그램」 作成을 위해서는 다음과 같은 作成方法의 導入이 要求된다.

① 「프로그램」의 「모듈」化 및 「모듈」크기의 規格化 (모듈當 50～100 step)

② 「모듈」에 대한 詳細한 注譯文記入 (機能槪要, 入出力 「파라메타」의 意味, 制限事項, 性能槪要, 作成者, 作成日字 등을 한글로 記入).

以上의 保守性外에 品質特性으로는 信賴性과 人間工學 (fluman Engineering) 側面을 갖고 있다.

5.4.1 使用者中心의 設計

「소프트웨어」 生産은 「소프트웨어」를 利用하는 對象이 電子計算機를 充分히 理解하는 技術者가 아닌 一般人으로 假想해야 한다. 各種 「콤파일러」를 비롯해서 最終使用者가 直接 使用하는 「소프트웨어」는 一般的으로 지나치게 專門的이고 水準이 높으며 使用法이나 注譯이 英語로 表記되어 있어서 各種 「콤파일러」가 갖고 있는 機能을 充分히 利用할 수 없게 된다. 이같은 경우는 一般的인 소프트웨어生産에 있어서도 마찬가지인 바 「소프트웨어」 生産은 반드시 使用者 立場에서 設計되고 다음 사항이 實現되어야 한다.

① 「에타메세지」는 한글로 그 內容, 對策 및 再處理方法이 充分히 說明되어져야 하고

② 操作方法 및 出力樣式의 說明도 한글로 理解가 容易하도록 記述되고

③ 處理時間이 길 때에는 使用者에게 豫定 및 終了時間에 관한 中間 메세지를 畵面이나 「콘솔」上에 出力시켜야 한다.

5.4.2 「소프트웨어」의 檢討

「소프트웨어」는 生産이 完了되면 各 作業段階別로 檢討作業을 實施한다. 檢討參加範圍는 該當分野의 專門家들로 構成되어야 하며 管理者에 의한 檢討는 避해야 한다.

① 檢討方法 : 檢討作業을 上司의 強要에 의해 實施해서는 안된다. 檢討作業은 그 過程에서 잘못된 점 問題點, 誤字 등을 찾아 修正하는 會議이므로 本人의 不快感을 되도록 除去시키고 보다 成功的이기 위해서는 本人 自身에 의해 檢討計劃을 設定하고 自身으로부터 檢討委員들에게 問題點의 發見을 依賴토록 하는 것이 最善의 方法이 될 것이다.

② 檢討作業의 役割分担 : 檢討內容은 1주전에 參加者에게 配布하고 上司의 指示에 따라 參席者들은 事前檢討해서 質問票를 作成하도록 해야 한다. 한편 參加者들에게는 各者 能力에 付合되는 役割分担에 의해 該當內容에 대한 責任感을 갖고 檢討하도록 하여야 한다.

③ 質問票作成 : 檢討參加者는 問題點이나 「에라」를 추출해서 個個에 대한 質問票를 作成 提出토록 하고 重複된 指適을 除外시킨후 問題點을 整理하고 그 內容을 個人別로 集計하여 錯誤內容의 類型을 찾아내고 組織으로서의 品質向上對策을 檢討할 수 있게 한다.

6. 「소프트웨어」의 評價

「소프트웨어」의 評價는 生産者에 의한 能力評價가 아니라 「프로젝트」를 옳게 計劃하고 實踐해 나가기 위해 要求되며 그 評價結果를 「프로젝트」 開發에 再活用하는데 目的이 있다.

「소프트웨어」의 開發現場에서는 錯誤 혹은 問題發生狀況, 原因, 措置內容等의 資料가 整理되어 있지 않거나 이들이 組織體 全體에 Feed back되지 않는 例가 많이 있다. 이같은 狀況에서는 生産性이나 品質向上은 主로 個人能力에 左右되고 만다. 「소프트웨어」의 開發은 個人의 集團이 아닌 組織의 活動이어야 하며 「프로그램」의 各 段階에서의 反省 및 評價를 올바르게 實施하는 일은 組織能力을 높이는데 있어 매우 重要하다.

開發過程에서의 評價는 進捷管理와 品質管理 및 反復作業의 除去를 위해 매우 重要하다. 開發終了時의 評價는 다음 開發活動을 效率化시키는데 더욱 必要하다. 具體的인 評價方法은 開發活動의 評價와 「소프트웨어」의 評價로 區分 實施된다.

6.1 開発活動의 評価

開發活動의 評價는 生産性의 測定, 品質管理基準의 整備 및 「프로그램·패러다임」 基準의 整備와 檢討로 이루어진다.

6.1.1 生産性의 測定

生産性의 測定은 다음 「프로젝트」의 見積作成, 生産性向上을 위한 諸活動의 效果的 測定을 위해 必要하다. 生産性은 主要機能 「프로그램」 命令語數를 尺度로 하여 測定되는데 一般的으로는 命令語數가 主로 活用된다. 이때 다음과 같은 「프로그램」 作成環境이나 難易度가 考慮되어야 한다.

① 命令語를 包含하는 「스텝」 數

8) 梁海述, 시스템分析과 設計. (서울 : 常湖社), pp. 340～341.

② 言語의 種類
③ 開發期間
④ 「프로그램」 種別
⑤ 開發要員의 平均水準

6.1.2 品質管理 基準의 整備

生産性이나 品質向上을 위해서는 失敗나 錯誤情報를 蓄積해서 同一한 內容의 失敗를 反復하지 않게 하여야 한다. 이를 위해서는 評價時의 質疑書 內容의 整理가 有效한 方法이 된다. 日常時의 品質管理活動 事例를 一定한 形式으로 整理하고 後日 再利用을 可能하게 하는 基準書가 必要하다.

6.1.3 「프로그램」 基準의 整備

「프로그램」의 再利用은 生産性向上을 위한 最善의 方法이다. 「프로그램·패러다임」은 汎用性이 높은 「모듈」을 수집, 整理한 것이며 이의 整備를 組織 全體的으로 추진하고 開發에 着手하기 전에 이 「패러다임」을 반드시 읽어 보도록 義務化할 必要가 있다. 「프로그램·패러다임」에 登錄될 「모듈」 作成에 包含될 內容은 다음 例와 같다.

- 機能概要 및 制限事項
- 入出力 「패러다임」의 意味
- 制限事項
- 반드시 使用될 上位, 下位 「모듈」 名
- 作成者 및 作成年月日
- 再利用時의 注意點
- 特殊言語 (Cobol, Fortran 等 除外)로 作成된 경우는 「흐름도」 作成登錄

6.2 「소프트웨어」의 評価

完成된 「소프트웨어」는 新「시스템」으로의 轉換을 위해 約 1～2個月間의 現行 「시스템」과의 並行處理가 通常要求된다.[8] 完成된 「프로그램」의 評價는 그 內容을 次期開發에 Feed back시키고 항상 安定된 品質을 保證하기 위해서 必要하며 保守作業量의 削減과 生産法 向上에 도움이 된다. 「프로그램」은 信賴性, 效率性, 使用性, 保守性을 重要 特性으로 하고 있으며 그에 대한 評價에 관해 檢討하도록 하겠다.

6.2.1 信賴性의 評價

「컴퓨터」의 高性能化 및 LAN 등의 On-line 「시스템」 普及으로 開發 「프로그램」의 規模는 매우 커졌으며 「시스템」 障害時 社會的 影響이 매우 큰 「시스템」도 增加一路에 있다. 이같은 狀況下에서 「시스템」에 要求되는 信賴性은 더욱 높아지고 있으며 SE나 「프로그래머」에게 要求되는 能力도 높아가고 있다. 信賴性에서는 故障安全對策이 특히 要求되기 시작하고 있다. 同安全對策은 다음과 같은 事項을 主要檢査內容으로 하고 있다.

① 「매뉴얼」에 記述된 限界値에 의한 檢査
② 限界値를 多少 超過한 「데이터」에 의한 檢査
③ 「하드웨어」 障害를 假想한 檢査
④ 使用者가 범하기 쉬운 誤謬를 假想한 檢査

6.2.2 效率性의 評價

效率性의 評價는 使用者 立場에서 매우 重要한 것이며 納品前에 綜合的으로 評價하고 使用者가 要求하는 性能에 合致하지 않으면 그 原因을 규명하고 解決策을 찾아야 한다.

6.2.3 使用性의 評價

「컴퓨터」 利用者는 專門家가 아니라 一般人들이므로 理解되기 쉽고 使用이 간편한 「시스템」이 要求된다. 使用性의 評價는 利用方法에 관한 說明의 詳細性, 「에라」 內容, 그 原因, 修正方法의 明確性, 操作法 利用法의 使用者 水準에 대한 適合性 等에 관한 檢討를 要한다.

· C.P.U 週邊裝置의 長時間 連續動作時 그 內容의 使用者에 대한 表 方法等에 관한 檢討를 要한다.

6.2.4 保守性의 評價

保守性은 保守의 容易性을 意味하며 그 測定은 「소프트웨어」 工學的인 評價基準도 있으나 基本的인 고려사항으로는

· 注釋의 한글 表記化
· 構造化 「프로그래밍」 技法에 의한 「프로그램」 作成
· 「데이터」 名, 變數名 등의 意味와 一致하는 名稱의 賦與 및 制御變數와 「데이터」 變數의 明確한 區分
· 「프로그램」의 「모듈」化 등을 包含한다.

7. 結　論

앞으로의 「컴퓨터」 開發은 「소프트웨어」 分野의 發展을 不可缺하게 必要로 하고 있으며 특히 「소프트웨어」의 效率的인 開發方法의 硏究가 要求되고 있다. 本 硏究에서는 「소프트웨어」의 效率的인 開發에 前提가 되는 「소프트웨어」의 要求定義 및 開發計劃의 作成과 그의 效率的인 開發 및 管理를 위한 具體的이고도 實踐的인 方法이 硏究되었으며 同硏究는 앞으로 「소프트웨어」 開發의 效率化를 위해 寄與할 것으로 생각된다.

9.) 王昌鍾, 시스템分析과 設計方法 (서울 : 正益社), p. 39.

參 考 文 獻

1. 李東郁外 編譯, 「소프트웨어」 工學 (서울: 尙湖社)
2. 李基式外 「소프트웨어」 生産技術 (서울 :正益社, 1986)
3. 梁海述, 「시스템」 分析과 設計 (서울 : 尙湖社)
4. 王昌鍾, 「시스템」 分析과 設計方法 (서울 : 正益社)
5. Maxgray Keither, Doeumentation Standards (1969)
6. Robert T. Grawer, Structured Method Through COBOL (Prentice Hall Inc. New Jereey, 1983)
7. ソフトウエアの開發計劃, エンピユートピア (1985.11號)
8. プロジクト管理研究會, ソフトウエア開發管理の實踐的方法, ユンピユートピア (1986.3號)