Software Manufacturing의 標準化에 関한 研究

(A Study on the Standardization of the Software Manufacturing)

朴 永 培*

Abstract

To secure a good quality computer program, the software manufacturing procedure is carefully studied into 2 categories, manufacturing and developing phase.

The former consists of 8 steps which are divided into 2 main categories, work-contains and documentation, the letter contains work planning, progress management, walk through and quality control.

In this paper, I suggest the general procedure of standardization for software manufacturing.

1. 序 論

高性能 컴퓨터의 出現과 科學技術의 急激한 發展에 의하여 Software의 大型化, 多樣化, 複雜化, 시스템化 됨에 따라 「Software 危機」란 用語가 數年前부터 使用되었고, 1960年度에는 開發과 保守를 合한 Software費用이 시스템 總費用의 約 30%였지만 1985年에는 約 90%까지 增加될 傾向이다.[1]

그런데 Software를 開發함에 있어서의 問題點은 生産性, 信賴性의 低下와 保守性, 擴張性, 管理性, 互換性의 缺如에 있다.

從來 우리나라의 Software 開發管理 姿勢가 지나치게 出力指向性이었기 때문에 管理者나 프로그래머도 「돌아가기만 하면 된다」는 姿勢가 되기 쉬우며 무리한 日程으로 開發하게 되고 에러가 發生해도 그때만의 修正으로 끝내 버리는 式이 되므로 完成된 製品은 곧 누더기가 되고 「그저 돌아가기만 하는 프로그램」의 集合이 되는 事例가 많았다.

특히 全컴퓨터 要員의 約 60% 정도가 Software 維持·保守에 종사하고 있는데[2] Software 開發管理를 出力指向型으로만 치중한다면 維持 保守作業에 대한 부담은 더욱 加重되는 結果가 될 것이므로 維持 保守에 대한 부담이 적은 시스템 開發이라는 要求는 앞으로 더 큰 重要性을 갖게 될 것이다.

따라서 本 研究는 品質좋은 Software生産과 System Life Cycle 全體의 費用減少 工程短縮을 위하여 効果的 프로그램 開發技法(IPT), 最新의 開發道具, 管理技法을 利用하여 Software 生産作業 標準化를 꾀하므로써 生産性, 信賴性이 높고 保守性, 擴張性, 互換性, 管理性이 容易한 Software를 生産하고자 함에 있다.

2. Software Manufacturing의 體系

Software Manufacturing의 구성은 生産工程, 開發管理의 2가지로 나누고 그 構成要素는 그림 2·1과 같다.

Software의 品質向上, 費用削減, 工程短縮을 위해서는 生産의 分業化하는 것이 重要한데 分業에는 空間的分業과 時間的分業으로 나누고 前者는 作業을 同一時點에서 分割하여 同時 並行的으로 進行하는 方法이고 後者는 作業을 時間軸으로 分割하여 時系列的으로 進行하는 方法이다.

生産의 空間的 分業을 容易하게 하기 위하여 그림 2·2와 같이 階層構造로 定義하고 生産의 時間的 分業을 容易하게 하기 위하여 그림 2·3과 같이 生産工程을 明確하게 定義할 必要가 있다.

특히 그림 2·3에는 生産工程別로 17種의 文書化와 効果的인 開發技法들을 소개하여 놓았다.

* 明知大學 電子計算學科 專任講師

1) Bohm, "Software Engineering", IEEE Transactions on Computers, 1976.

2) 李基式, 소프트웨어 生産技術, 서울; 正益社, 1981, p.198.

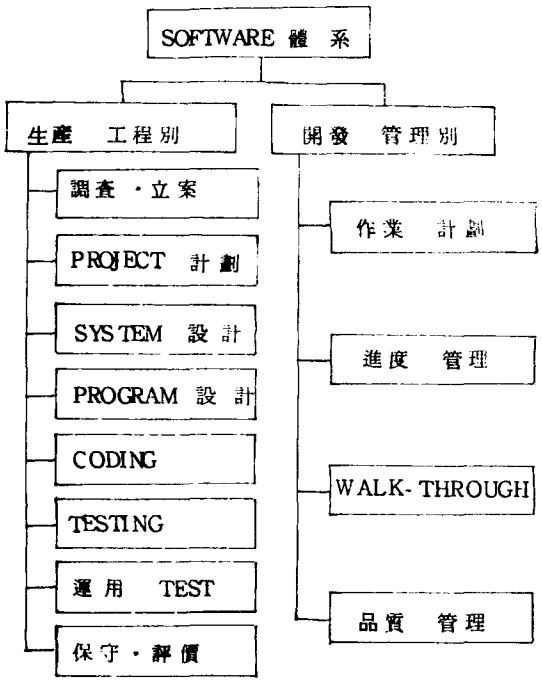


그림 2·1 SOFTWARE 体系

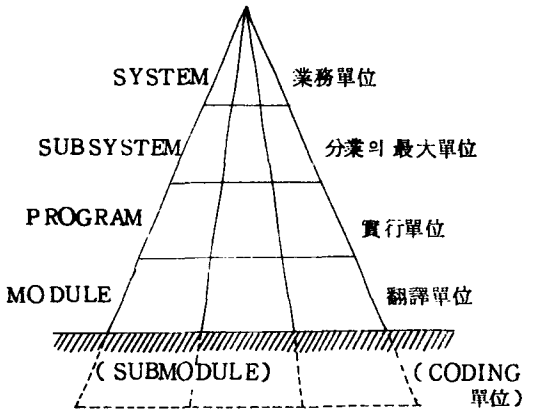


그림 2·2 SOFTWARE 階層構造

3. 生産工程別 作業內容

生産工程에는 그림 2·1 과 같이 8工程으로 나누고 各 工程마다 WHY, WHAT, HOW를 明確하게 하기위해서 重要 作業內容과 Document (文書化)에 대한 標準化 內容을 設定하였다. 또 그림 2·3과 같이 複合設計, HIPO, Structured Programming, Sandwitch Test, Walk-Through 등 最新 Software Engineering의 IPT技法을 適用하였다.

3·1 調査·立案(Survey·Planning)

工程의 目標는 EDP化 對象業務에 關한 使用者 要件(user requirement)을 把握하여 시스템開發의 必要性과 可能性을 檢討, 立案한 다음 最高 經營者의 承認을 얻는 段階이다.

(1) 重要 作業內容은 다음과 같다.

① 國內外의 類似시스템 또는 理論, 알고리즘에 대한 動向 및 技術調査를 한다.

② 使用者 要件과 現行시스템(手作業의 경우 事務흐름)의 現狀 問題點을 調査分析하여 新시스템 構想案을 立案하고 開發의 目的, 必要性, 範圍, 效果를 明確히 한다.

③ 人力, 費用, 期間, 信賴性, 性能 等 制約條件을 감안한 開發計劃案을 立案한다.

④ 新시스템 今後의 課題, 展望, 問題點을 提示한다.

(2) 文書化는 다음과 같이 작성한다.

① 調査·立案書

· 作成對象…… System

· 作成內容…… 動向·技術調査, 使用者 要件, 現狀分析, 시스템構想, 開發計劃(案), 시스템今後의 課題, 展望과 問題點

3·2 PROJECT 計劃(Project planning)

工程의 目標는 對象시스템의 概要設計와 費用效果分析을 하여 資源計劃과 開發計劃을 감안해서 開發의 可否를 決定한 다음, Project의 體制, 日程, 製品目標, 資源을 具體的으로 決定하는 段階이다.

(1) 重要 作業內容은 다음과 같다.

① 使用者要件을 確定하고 製品目標(信賴性, 擴張性, 保守性, 效率性, 互換性, 操作性의 관점에서)를 設定한 다음 投資(開發·運用費用)에 대한 效果를 分析하고 資源 및 開發計劃을 樹立하여 開發의 可否를 決定한다.

② 現行 業務의 데이터量, 處理周期를 分析하여 新시스템의 主要機能, 處理範圍, 處理能力, 處理形態, 規模 등을 設定한 다음 開發實現 可能性을 調査한다.

③ 시스템의 主要入出力 image作成, Master file의 構造 決定, DBMS選定, 데이터흐름, 데이터量, file量을 把握하여 시스템 概要設計를 한다.

④ Test 方針과 基準을 決定하고 Test Tool 選定, 開發言語 選定 및 障害對策을 세운다.

⑤ Project를 圓滑히 運營하기 위한 技術調査(設計技法), 推進體制, 作業標準化(文書化, 用語의 統一), 管理方法을 設定한다.

⑥ 新시스템의 運用(組織)實行計劃, 移行實行計劃을 具體的으로 決定한다.

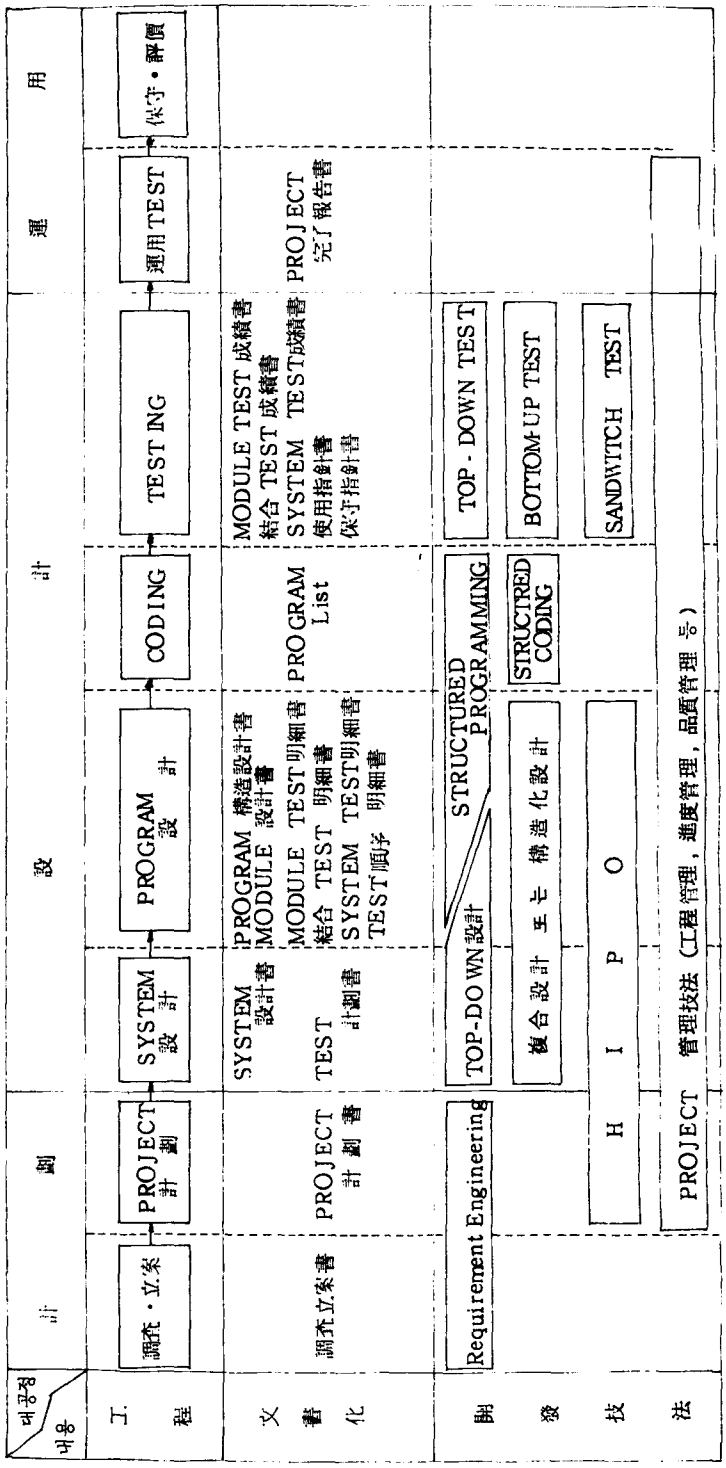


그림 2·3 SOFTWARE 生産工程

⑦ 資源實行計劃, 開發實行計劃, 訓練計劃을 세우고, 進度管理, 豫算實績管理, 明細變更管理 方法을 制定한다.

(2) 文書化는 다음과 같이 作成한다.

① PROJECT 計劃書

· 作成對象…… System

· 作成內容…… 基本方針, 現시스템詳細分析, 시스템化概要, 使用者要件確定, 製品目標, 投資効果分析, 開發實施計劃, 資源實施計劃, Project 管理設定

以上 두 工程을 綜合하면 그림 3·1과 같다.

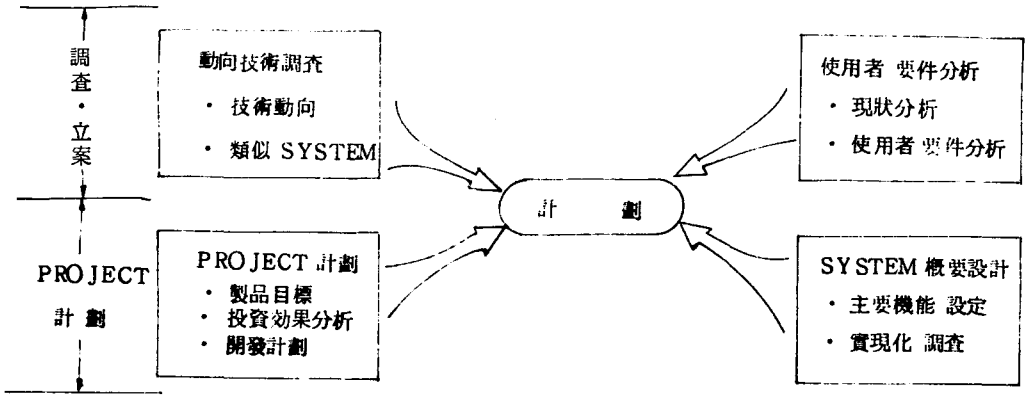


그림 3·1 計畫工程

3·3 SYSTEM 設計(SYSTEM DESIGN)

大規模 System의 경우는 System設計를 基本設計와 論理設計로 分割 設計할 수 있는데, 基本設計는 System을 Subsystem으로 機能別 分割 定義하고 論理設計는 Subsystem別로 System 構築上 必要한 모든 機能을 프로그램으로 分割 定義할 수있다.

工程의 目標는 使用者 要件을 蒐集 分析結果를 基礎로 System機能을 確定하고 이 機能을 實現하기 위하여 Subsystem 과 프로그램으로 分割하여 System外部明細, 프로그램外部明細, 프로그램間 處理흐름을 決定하며 Test를 計劃하는 段階이다.

(1) 重要 作業內容은 다음과 같다.

① 使用者가 본 外部機能을 決定하고 入出力項目 및 形式決定, CODE設計, Message 設計, 알고리즘, Interface 등 System 外部明細를 設計한다.

② System을 Subsystem으로, Subsystem은 프로그램으로 分割한 다음 各 Subsystem의 Process Flow, 데이터 Flow의 決定, 데이터(master, 中間file, DB)의 論理構造設計, Master File의 編成方法과 데이터項目 決定, File의 收容效率과 Access 時間 計算 등 프로그램 外部明細를 設計한다.

③ Test 機器의 構成決定, 機材(Disk, MT) 確保, Test 計劃(體制, 方法, 項目, 環境, 日程)을 立案하고 Test Tool을 設計한다.

④ Center 運用管理, 性能效率測定, 資源管理를 行하는 運用管理 System設計와 移行 System設計를 하고 障害 發生時의 運用順序와 Recovery方式을 設計한다.

⑤ Basic Software의 機能確認 및 敎育, 設計者의 業務敎育, 設計方法論, 設計技法 등을 敎育한다.

⑥ Subsystem單位, Program單位로 詳細工程計劃을 세워서 進度管理, 稼算實積管理, 作業環境整備, 明細變更管理 등을 Testing工程 段階까지 繼續하여 實施한다.

(2) 文書化는 다음과 같이 作成한다.

① 시스템 設計書

• 作成對象 … Subsystem, Program

• 作成內容 … ㉠ Subsystem(시스템構成 外部機能決定, File設計, Code設計, 入出力設計 障害對象). ㉡ Program(System方式, Program 外部明細, 入出力形式決定, 데이터論理設計)

② Test 計劃書

• 作成對象 … System, Subsystem, Program, Module

• 作成內容 … 品質目標 및 管理基準, Test 段階設定, Test의 方法, 體制, 環境, 項目, 日程

3·4 PROGRAM 設計

프로그램機能과 規模가 클 때는 프로그램을 Module(subprogram 또는 Subroutine)로 分割하고 Module은 Submodule로 分割할 수 있다. 여기서 그 槪念을 Step數로 定義해 보면 System은 萬Step以上, Subsystem은 數千~萬 Step, 프로그램은 數百~數千Step, Module은 數拾~數百 Step, Submodule은 約 50 Step정도로 본다. 이 數値는 劃一的으로 말할 수는 없으나 本人이 10餘年間 實務 經驗에서 얻은 結果임을 밝혀 둔다.

工程의 目標는 System設計書에 定義한 各 프로그램의 屬性 決定, Module로 分割, Module 明細, Module間 處理흐름을 決定하고 各種 Test 明細를 設計하는 段階이다.

(1) 重要 作業內容은 다음과 같다.

① System의 能力(CPU, Channel, Device)을 計算하고, Message Buffer 길이와 數를 決定하고, 프로그램屬性(reentrant, task, routine, segment, overlay 등) 決定 等 System 方式의

詳細設計를 한다.

② 프로그램을 Module로 分割하고 各 Module內部 明細設計, Module內 處理順序 詳細設計, Module間 Interface明細設計 全體 制御方式 設計를 한다.

③ 데이터物理構造(master, 中間file, DB) 設計와 Module內 데이터設計를 한다.

④ Module Test 明細, 結合 Test明細, System Test 明細를 設計한다.

⑤ Test를 위한 Machine確保, Tool作成, 實施計劃을 確定한다.

⑥ 運用管理 프로그램(file 및 DB論理 check program, 稼動分析, 資源最適化, 媒體管理 program)과 移行用프로그램(데이터變換프로그램 等)을 設計한다.

⑦ Test實施方法과 管理基準, 制度를 設定하고 Basic Software를 Test한다.

⑧ Coding規約을 決定하고 Program 言語 敎育을 實施한다.

⑨ Program單位, Module單位로 詳細工程計劃을 세우고 管理方法은 3·3工程의 ⑥項과 同一하다.

(2) 文書化는 다음과 같이 作成한다.

① PROGRAM構造 設計書

- 作成對象 … Module間
- 作成內容 … 프로그램內部明細, Module間 Interface, 데이터物理構造設計.

② MODULE 設計書

- 作成對象 … Module間
- 作成內容 … Module內部明細, Coding規約.

③ MODULE TEST 明細書

- 作成對象 … Module間
- 作成內容 … Module內 Test明細, Test項目 Check List

④ 結合 TEST 明細書

- 作成對象 … Module間, Program間, Subsystem間.
- 作成內容 … 結合 Test計劃, 結合 Test明細, Test項目 Check List.

⑤ SYSTEM TEST明細書

- 作成對象 … System
- 作成內容 … System Test計劃, System Test明細, Test項目 Check List.

⑥ TEST順序 計劃書.

- 作成對象 … Module, Program, Subsystem, System
- 作成內容 … Test方法, Test管理方法.

3·5 CODING

COBOL Coding의 경우 첫째 1 Submodule은 1 Section으로 하는 方法. 둘째 SEQUENCE, IF THEN ELSE, DO WHILE, DO UNTIL, CASE의 5가지 制御構造를 使用하는 方法. 세째 데이터名稱을 設定하는 方法. 네째 Comment를 挿入하는 方法 등 標準化 Coding 規約을 設定하여 Coding토록 한다.

工程의 目標는 Coding, Compile하는 段階이다.

(1) 重要 作業內容은 다음과 같다.

① Module을 Coding하고 Compile 하여 文法 에러를 修正한다.

② 各種 File, Tape, Disk의 初期値 設定 作業을 한다.

③ Test時에 使用되는 Terminal Simulator使用方法과 各種裝置의 操作方法을 敎育한다.

④ Test 데이터를 作成하고 프로그램 에러發生狀況을 分析하여 에러對策을 세운다.

⑤ 運用管理프로그램, 移行프로그램을 作成하고 그 機能을 確認한다.

⑥ 開發中인 프로그램 및 Basic Software의 管理方法, 體制 등을 設定하고 System 管理를 實施한다.

(2) 文書化는 다음과 같이 作成한다.

① PROGRAM LIST

- 作成對象 … Program
- 作成內容 … Source List, Linkage List, Object List, 實行 List, 데이터 List.

3·6 TESTING

工程의 目標는 Module Test, 結合 Test, System 綜合 Test 順으로 하고 製品目標項目을 檢證 및 確認하는 段階이다.

3·6·1 Module Test

Module Test明細書에 의거 Module自體를 Test하는 것으로 이것은 信賴性이 높은 Software를 製作하는데 가장 重要한 比重을 차지하므로 個個의 Module信賴性이 높으면 結合 Test作業의 生産性이 높게 되고 따라서 製品의 信賴性도 높게 된다. Desk debug와 Machine debug로 나눈다.

(1) 重要 作業內容은 다음과 같다.

① Desk debug는 Module設計書, Module Test明細書와 文法 에러를 修正한 Source List를 比較하면서 論理(Logic) Check를 한다.

② Machine debug는 Module Test 明細

書에 의거 計算機로 論理 Check를 한다.

(2) 文書化는 다음과 같이 作成한다.

① Module Test 成績書

• 作成對象 … Module內

• 作成內容 … Module 一覽, 實施狀況,品質基準, 에러分析과 評價

3·6·2 結合 TEST(Integration Test)

結合 Test는 從來의 Top-down Test와 Bottom-up Test의 兩者의 短點을 補完한 즉, 兩者를 併用하는 技法인 Sandwitch Test를 使用한다.

(1) 重要 作業內容은 다음과 같다.

① 結合 Test明細書에 의거 Module間 結合 Test, Program單位 Test, Program間 結合 Test, Subsystem間 結合 Test, 裝置間 結合 Test 하여 그 機能을 檢證한다.

② System Test에 대비하여 性能評價, 効率測定을 하기 위해 必要한 하드웨어 Monitor, Simulator를 설치한다.

(2) 文書化는 다음과 같이 作成한다.

① 結合 Test 成績書

• 作成對象 … Module, Program, Subsystem, System

• 作成內容 … Program一覽, 實施狀況, 品質基準, 에러分析과 評價

② 使用 指針書

• 作成對象 … Module, Program, Subsystem, System

• 作成內容 … System概要, 稼動方法, 操作方法과 順序, 實行例, 에러 Message, 障害時의 處理.

3·6·3 SYSTEM TEST

System Test明細書에 의거 System을 綜合的으로 Test하고 製品目標 項目을 確認하는 段階이다.

(1) 重要 作業內容은 다음과 같다.

① System Test明細書에 의거 System의 綜合的인 機能, 性能, 例外處理, 負荷, 構成, 互換性, 操作性, 耐久性 等 Test를 하고 製品目標 項目을 確認한다.

② 文書化와 Software가 一致한 가를 確認한 後, 完了된 開發 프로그램을 運用部署에 移管하고 System의 保守順序와 管理順序를 設定한다.

③ 運用管理 프로그램 機能 確認, 業務 運用日程(machine, punch, job, online), 運用體制等을 最終 確認한다.

④ 移行時의 問題點을 點檢하기 위해서 移行練習을 한다.

⑤ Operator, 實使用者에 대한 System 運用, 端末操作, 業務흐름을 敎育한다.

(2) 文書化는 다음과 같이 作成한다.

① SYSTEM TEST 成績書

• 作成對象 … System

• 作成內容 … 實施狀況, 全般的 品質狀況, 實施評價

② 保守 指針書

• 作成對象 … Program, Subsystem, System

• 作成內容 … 保守對象 File, 保守對象 文書化, 保守順序, 保守發生要因別 對處方法, 保守用 JCL例

3·7 運用 TEST(Operational Test)

工程의 目標는 正常 稼動때와 똑 같은 System 環境에서 新 System을 舊 System과 併行稼動시켜 實使用者(end-user)에게 熟達시키는 동시에 新 System 全體의 稼動狀態를 最終的으로 保證하는 段階이다.

(1) 重要 作業內容은 다음과 같다.

① 同一 System環境에서 新·舊 System을 併行稼動시켜 實稼動狀況을 最終 確認하면서 業務部署와 運用部署에게 開發部署가 運用上의 障害對處法과 修正法을 說明한다.

② 實使用者 및 Center運用部署의 新 System에 대한 熟達度와 安定性을 綜合的으로 判斷하여 新 System의 本格稼動 日字를 決定한다.

③ Project 開發統計情報를 整理 分析, 評價하여 Project完了報告書를 作成한다.

(2) 文書化는 다음과 같이 作成한다.

① PROJECT 完了 報告書

• 作成對象 … System

• 作成內容 … 總括, 開發System의 概要, 作業經過, 開發實績, 評價와 今後展望.

3·8 保守·評價(Maintenance·Evaluation)

工程의 目標는 Project 計劃書와 對比시켜 System을 綜合的으로 短期(本格實施後 1年內) 評價 및 長期評價하여 다음 System構想에 反映하는 마지막 段階이다.

(1) 重要 作業內容은 다음과 같다.

① System의 運用·保守·評價와 Project의 評價를 한다.

② System의 運用에서 System評價用 情報(稼動狀況, 性能·効率情報, 操作性 등)를 蒐集分析하여 製品目標 達成度의 評價 및 Project評價

를 한다.

③ System 障害의 保守 및 實使用者로 부터 改良, 改善要求에 대한 System 保守를 한다.

以上 生産工程의 8工程에 대해서 重要 作業內容과 文書化內容에 대해서 상세히 언급하였는데 이것을 綜合的으로 정리해 보면 計劃·設計·TEST와 文書化와의 關係는 그림 3·2와 같고 各 工程마다 重要 作業內容을 SOFTWARE MANUFACTURING PERT圖로 그려 본 것이 그림 3·3과 같다.

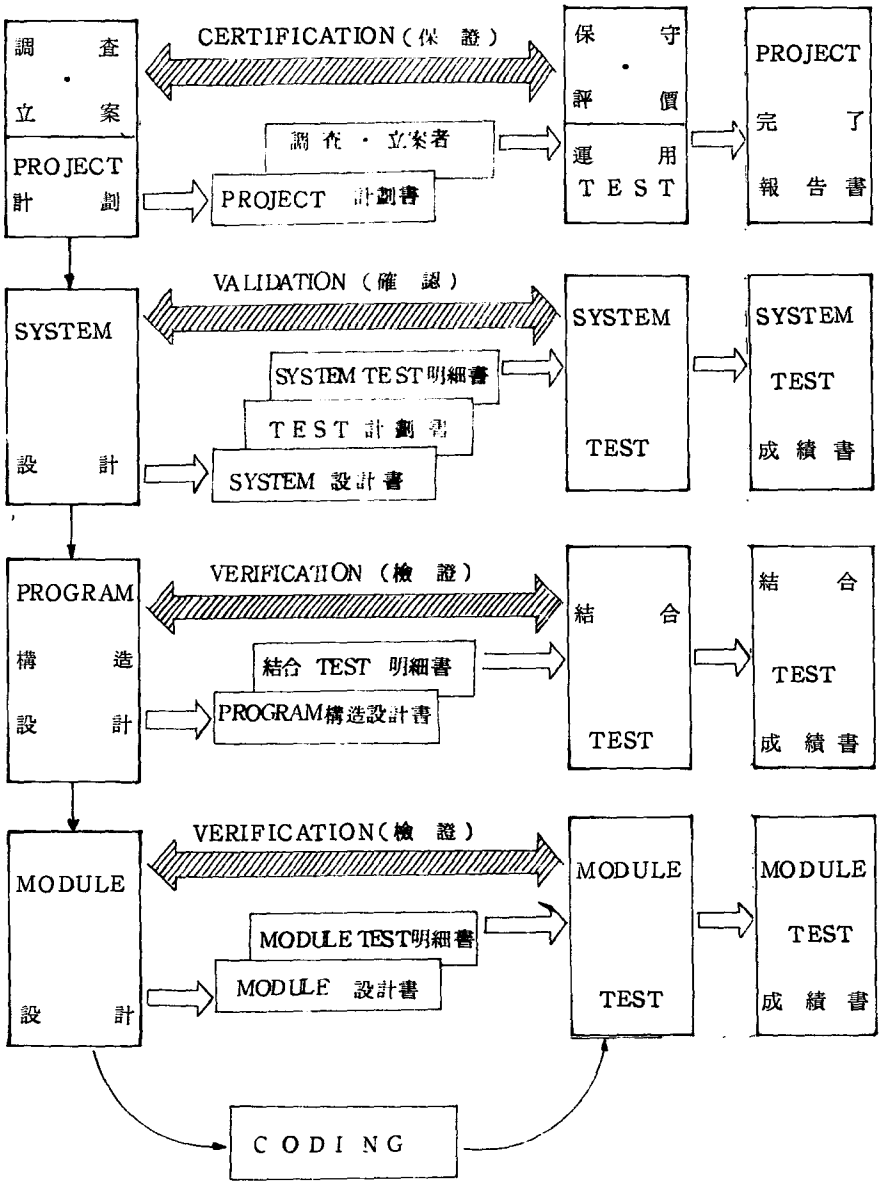


그림 3·2 計畫·設計·TEST와 文書와의 關係

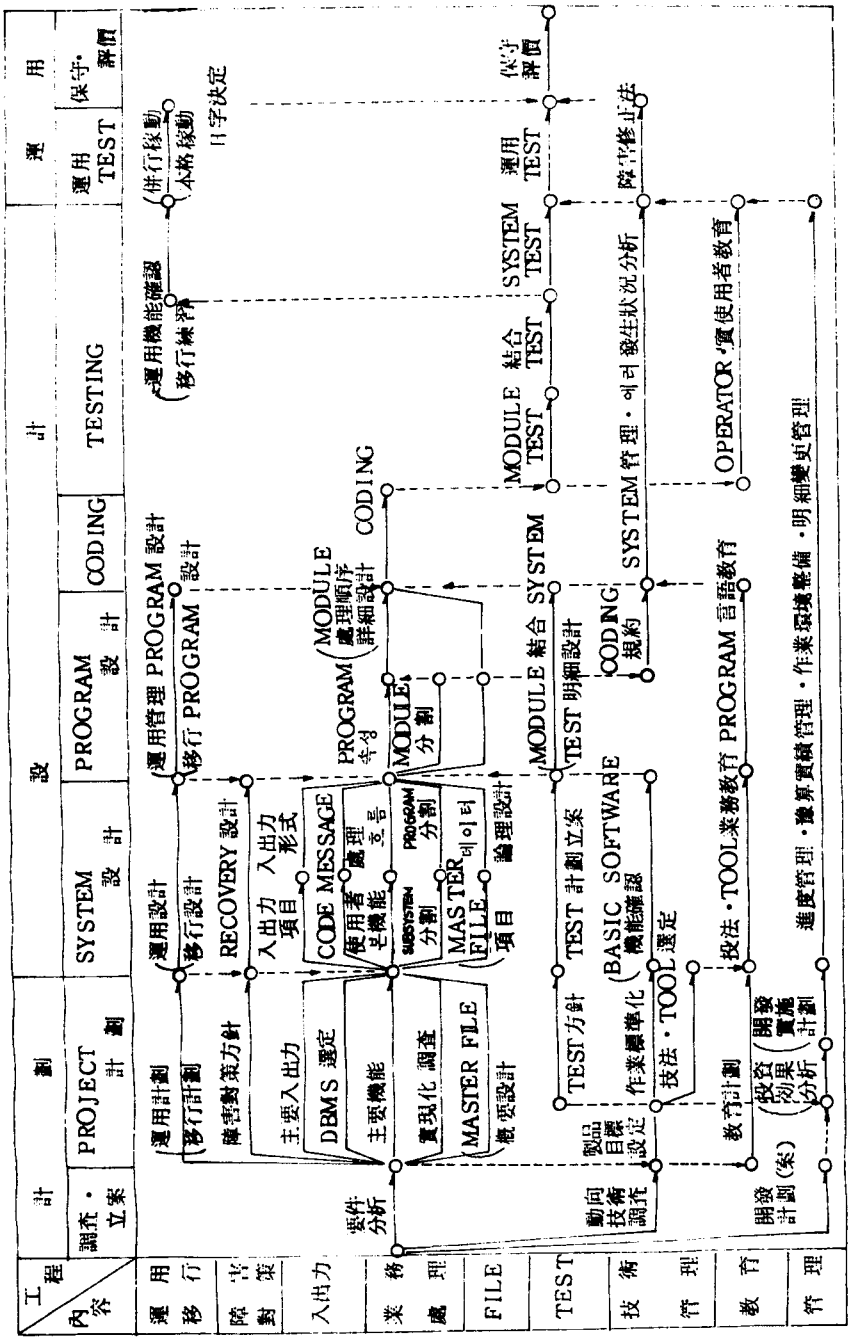


그림 3·3 SOFTWARE MANUFACTURING 의 PERT 圖

4. 開發管理別 作業內容

8段階의 生産工程을 圓滑히 수행하여 品質좋은 Software를 生産하기 위하여 作業計劃, 進度管理, 品質管理, Walk - Through의 4가지 開發管理要領을 設定하였다.

4·1 作業計劃

設計作業을 効率的으로 수행하기 위해서는 設計作業을 獨立性이 높은 Module單位로 分割하는 것이 効果的이다. 이것은 프로그램의 構造化 槪念과 같으며 이 作業을 위한 作業計劃書의 例는 그림 4·1과 같으며 다음과 같은 効果가 있다.

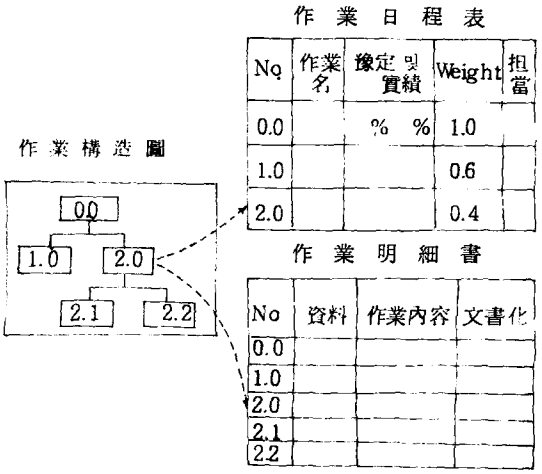


그림 4·1 作業計画書

① 作業者는 自身의 作業位置와 目的을 確實하게 把握할 수 있다.

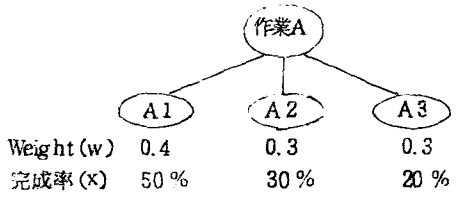
② 作業者는 他人의 作業分担도 쉽게 把握할 수 있기 때문에 作業間의 連絡이 쉽다.

③ 作業者는 特別한 教育을 받지 않고도 日常作業中에서 體系的인 作業方法을 터득할 수 있다.

④ 管理者는 短時間에 作業內容을 理解할 수 있으며, 作業項目의 누락, 進度狀況 check, 要員의 配置 等 實質的인 管理가 容易하다.

4·2 進度管理

이것은 進度狀況을 定量的으로 把握하는데 特徵이 있으며 어떤 作業의 完成率은 그 作業의 下位 作業의 完成率과 各 作業에 주어진 Weight의 積의 合으로 구한다. 여기서 最下位 作業의 完成率은 0~100까지 %로 나타내고 作業의 Weight(加重値)는 Step數나 業務量으로 算出하며, 그 例가 그림 4·2와 같다.



作業A의 完成率
$= \Sigma X_i \times W_i$
$= 0.4 \times 50 + 0.3 \times 30 + 0.3 \times 20$
$= 35\ \%$

그림 4·2 完成率計算方法

이와 같이 最下位 作業에서 最上位 作業까지 完成率을 計算하여 그림 4·1 作業計劃書의 作業日程表에 定期的으로 記入하여 進度管理를 한다.

4·3 WALK-THROUGH(徒步檢査)

대개의 設計作業은 獨立한 部分으로 分割된 要員各自가 各 部分을 分担하여 進行한다. 그러나 個人이 行한 作業에는 많은 缺陷이 있을 수 있는데 이것을 發見하는 데는 그 作業에 關聯하는 여러사람이 모여서 그 作業 結果를 檢査하는 것이 効果的이다. 따라서 이 作業은 檢査 會議의 運營要領을 말하며 進行 順序는 Review段階, Meeting段階, Follow-up段階의 3段階로 나누고 상호관계를 나타낸 것이 그림 4·3과 같다.

여기서는 Review받을 사람(開發者)이 會議 參加者 選定, 製品(檢討資料) 配付, 會議 日程을 計

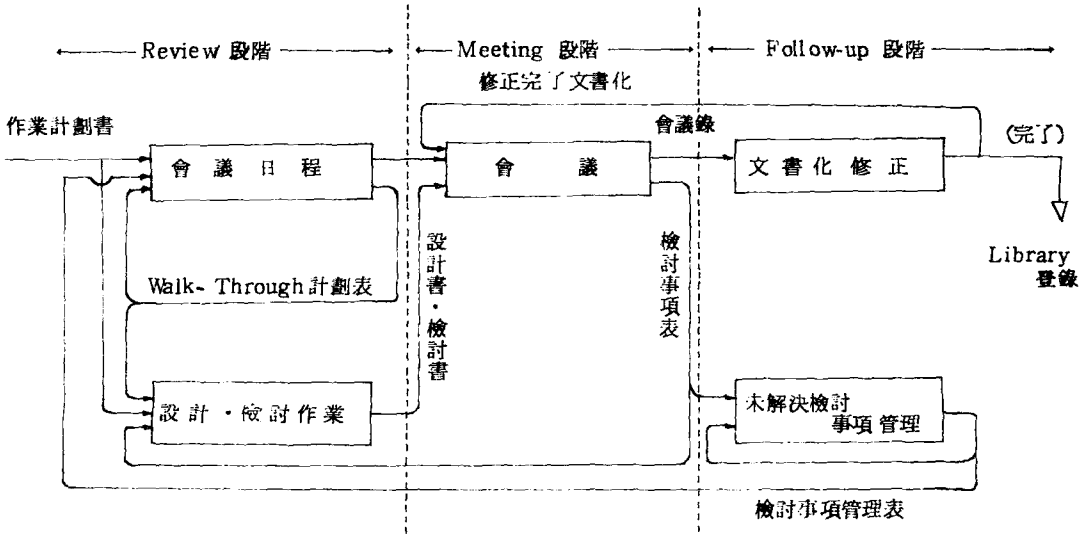


그림 4·3 WALK THROUGH 順序

劃하고 Review할 사람은 製品을 事前 檢討하여 會議(meeting)에 參席하고 Follow - up에서는 開發者가 會議錄에 따라 製品(文書化)을 修正하고 未決事項이 있을 때는 Walk - Through 計劃을 다시 세운다.

Walk - Through는 1 ~2時間內로 하는 것이 가장 效果的이다.

4·4 品質管理

品質, 費用, 納期에 영향을 주는 品質特性(module 크기)을 選定하여 設計工程에서 運用 Test까지 管理하면서 異常有無를 早期 發見하는 것으로 Software의 各 階層別, 單位別로 그림 4·4와 같은 形式의 品質 Check表를 作成하여 管理한다.

換算値를 그림의 Graph上에 그려서 母平均에 대하여 $\pm 2\sigma$ 또는 $\pm 3\sigma$의 범위를 넘는 것을 異常値로 보고 異常値의 Module에 對해서는 原因을 조사하여 對策을 세운다.

品質特性	基礎데이터	換算値	評價 GRAPH -3σ μ $+3\sigma$	母平均	母標準偏差
實積step	412 step	1.34		1	0.3
設計書枚數	28 枚	68枚 ／KS		50	10
TEST項目數	12件	29件／KS		50	15
COMMENT比率	219 step	42 %		30	8
에러發見數	2 件	5 件		10	4.3
⋮		KS : kilo step			

基礎데이터 : Module Test 項目의 直接的인 値

換 算 値 ; 1,000step 當 Test 項目數의 標準的인 値와 比較가 될 수 있도록 換算한 値

그림 4·4 品質 CHECK表

5. 結 論

從來의 Software 開發姿勢가 自己中心的이고, 出力指向型이었기 때문에 生産性과 信賴性이 低下되고 保守性과 擴張性, 管理性, 互換性이 缺如되었다.

이러한 問題는 本論에서 文書化를 中心으로 提示한 標準化를 實務에 適用, 設定하므로서 解決될 수 있고 아울러 다음과 같은 效果도 얻을 수 있다.

1. 新 PROJECT 發足時 標準化作業과 要員敎育이 必要하지 않거나 또는 短期間에 할 수 있다.
2. 新入社員도 短期間에 Software 生産技術을 몸에 익힐 수 있다.
3. 自己中心的인 開發姿勢가 他人과 Team을 위한 姿勢로 된다.
4. Software의 品質이 均質化 된다.

그러므로 모든 컴퓨터 使用者는 Software manufacturing의 標準化 作業을 하루 빨리 推進하고 定着시켜야 할 것이며 또 꾸준히 改善해야 할 줄로 믿는다.

그러나 文書化의 內容은 提示하였으나 形式을 提示하지 못한 點과 長期間에 걸쳐서 標準化作業의 效果를 定量的으로 測定, 分析하지 못한 點은 앞으로 계속 硏究해야 할 것이다.

參 考 文 獻

1) Jensen, R. W., and Tonies, C. C., Software Engineering, Prentice - Hall, 1979.
2) 國友義久, 効果的プログラム開發技法, 近代科學社, 1979.
3) Myers, G.L. (國友義久譯), ソフトウエアの複合／構造化設計, 近代科學社, 1979.
4) FACOM SDEM 解說書, 富士通, 1980.
5) FACOM SDEM 適用, 富士通, 1980.
6) Optimizing Program Quality and Programmer Productivity, IBM, 1980.