Requirements分析을 위한 DFD Editor 및 Evaluator의 設計 및 具現

韓相晩·朴京範·金聖曦·宋永基 / S/W工学研究室

〈요 약〉

Requirments 분석 tool은 시스팀의 개발 단계중 한 단계인 requirements 분석의 자동화를 위한 것으로 입력된 requirements의 정보를 데이터베이스에 수록하고 수록된 정보를 이용하여 데이터의 consistency 및 completeness를 조사 분석하는 도구이다. 본 논문은 이 requirements 분석 tool의 설계상의 구성과 입력될 requirements를 Data Flow Diagram(DFD)으로 구현하기 위한 editor와 evaluator에 관한 설계 및 구현에 대해 기술한다.

Ⅰ. 개 요

하나의 시스팀을 개발하기 위해서는 requirements 작성 및 분석, 시스팀 specification 작성 및 분석, 설계, implement, 그리고 시험 및 유지 보수 과정을 거치게 된다. 시스팀의 효율적인 개발을 위해서는 첫단계인 requirements 작성 및 분석을 명확히 해야함은 널리 알려진 사실이다. 명확한 requirements 작성을 위해서는 requirements 분석의 자동화가 필요하며 효과적인 분석을 위해서는 자동화를 위한 tool이 요구된다. Requirements 분석의 자동화를 위해서는 형식화된 입력형태가 필요하다. 이러한 입력형태는 requirements의 내용을 모두 함축할 수 있어야 하며 또한 사용자의 내용파악이 쉬워야 하고 수정 및 확장이 간편해야 한다. 이로 인해 본 논문에서 기술하는 requirements 분석 tool은 개발하고자 하는 시스팀 requirements의 복잡성을 제거해 주고 또한 functional capability의 분석을 용이하게 해주는, 구조화 분석 기법에서 사용하는 계층적이고 구조적인 다이어그램 형태인 Data Flow Diagram(DFD)을 입력으로 사용한다.[6,8] DFD는 개발하고자 하는 시스팀을 망 형태로 표현하는 것으로서 시스팀의 각 component와 그 component들 사이의 모든 인터페이스를 나타내주는 다이어그램이다.[8] 이 DFD는 시스팀내의 데이터의 흐름과 이 데이터를 이용하여 어떤 일을 수행하는 프로세스들을 계층적인 그래픽 형태로 보여준다. 이러한 DFD는 DFD editor에 의해 그래픽 형태로 구현되며 DFD evaluator에 의해 정확성이 검토된다.

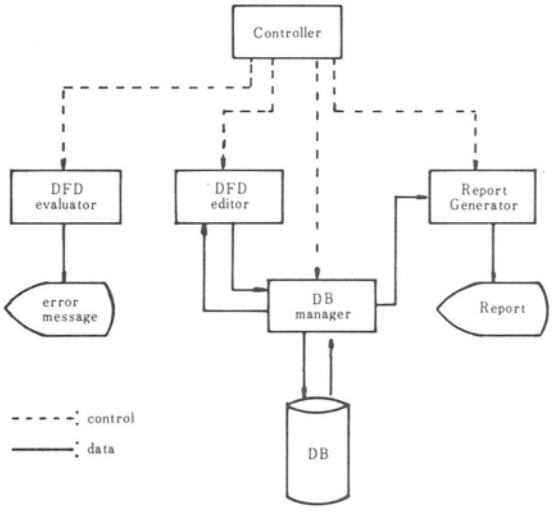
DFD editor를 통해 DFD로 표현된 requirements의 정보는 데이터 베이스에 수록을 하게 되며 수록된 정보는 필요에 따라 수정 및 확장이 이루어 진다. 이로 인해 데이터 베이스의 정보는 항상 최신 정보를 가지게 되며 completeness와 consistency를 만족하는 데이터들로 구

성하게 된다. 또한 데이터베이스에 수록된 시스템의 모든 정보를 여러 관점에서 볼 수 있고 분석 결과를 알 수 있게 해주는 리포트 추출의 자동화도 필요하다.

본 논문은 requirements분석 tool의 전체적인 구성 및 설계 방향, 그리고 DFD를 구현하기 위해 개발이 완료된 DFD editor와 DFD evaluator에 관한 설계 및 구현 방법에 관하여 기술한다.

Ⅱ. Requirements tool의 구성 및 설계방향

Requirements 분석 tool은 전체적으로 사용자 인터페이스, DFD editor 및 evaluator, 데이터베이스 manager, 그리고 report generator로 크게 네부분으로 이루어진다. 사용자 인터페이스 부분은 tool내의 각 기능들을 제어하는 controller로 구성된다. DFD editor는 원하는 DFD를 그래픽 형태로 구현해 주며 DFD evaluator는 DFD editor에서 생성된 DFD의 syntax 및 hierarchy의 정확성을 검사한다. 데이터베이스 manager 부분은 입력된 정보를 데이터베이스에 저장하고 저장된 정보의 확장 및 수정을 위한 DFD editor와의 인터페이스 부분이다.
데이터베이스 manager는 데이터베이스에 DFD 정보를 수록하기 위한 converter 와 수록된 내용의 확장 및 수정을 위해 다시 DFD정보로 변환해주는 deconverter부분으로 나누어 진다.[5] Report generator부분은 여러형태의 format으로 데이터베이스에 저장된 시스템구조에 관한 정보 및 각 정보들간의 consistency, completeness 및 redundancy를 분석한 결과를 리포트형태로 생성해 주는 데이터베이스 manager 와의 인터페이스 부분이다. Requirements분석 tool의 전체적인 구조는 〈그림 1〉에 나타나 있다.



〈그림 1〉 Requirements 분석 tool의 구조

Ⅲ. DFD 소개

DFD는 개요에서 언급한 바와 같이 망 형태의 다이어그램으로서 시스템에서 사용되는 데이터를 기준으로 표현하며 시스템 components 와 각 component 사이의 데이터의 흐름을 계층적으로 보여준다.

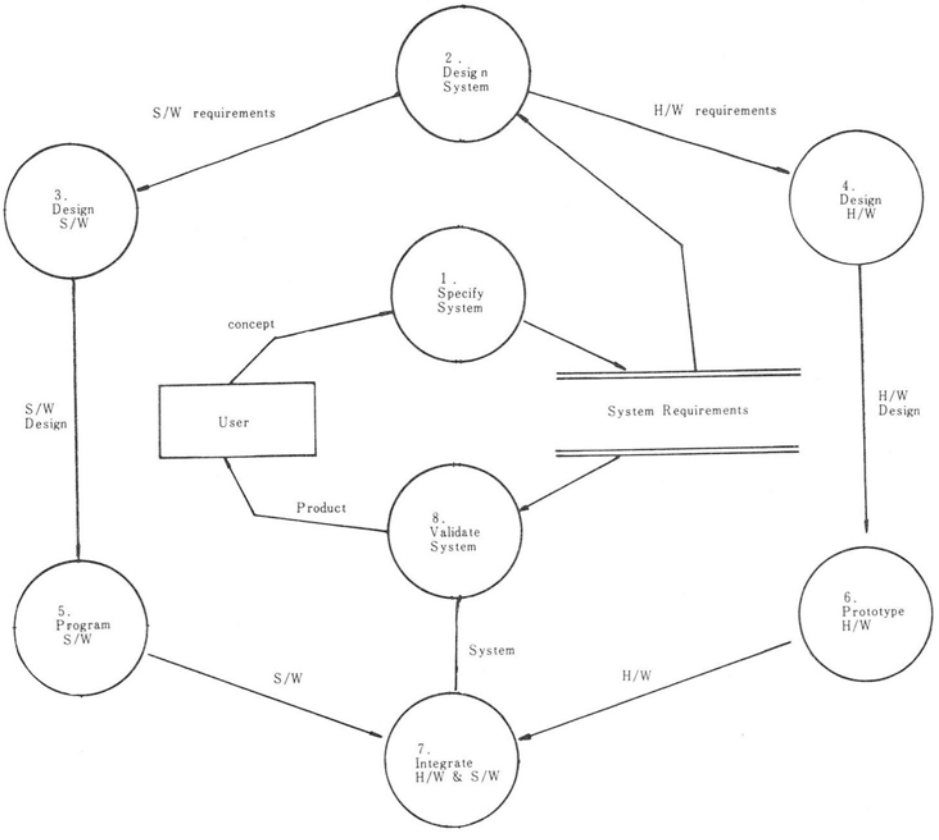
DFD는 데이터의 source 및 sink, process, data store, 그리고 dataflow로 구성된다. Source는 시스템으로 특정 아이템을 제공하며 sink는 시스템이 생성한 데이터를 받아들인다. Process는 시스템이 수행하는 각 기능을 나타내며, data store는 데이터가 저장될 장소를 표현하는 것으로 process에서 사용되지 않는 데이터를 임시적으로 저장하는 장소를 나타낸다. 그리고 dataflow는 데이터와 흐름을 나타낸다. 여기서 하나의 DFD에서 표현된 process는 다시 세부적인 DFD를 구성한다. 〈그림 2〉에서 보는 바와 같이 source 및 sink는 사각형, process는 원, data store는 직선, 그리고 dataflow는 화살표로 표시한다.

〈그림 2〉는 시스템 개발 단계를 DFD로 표현한 것이다.

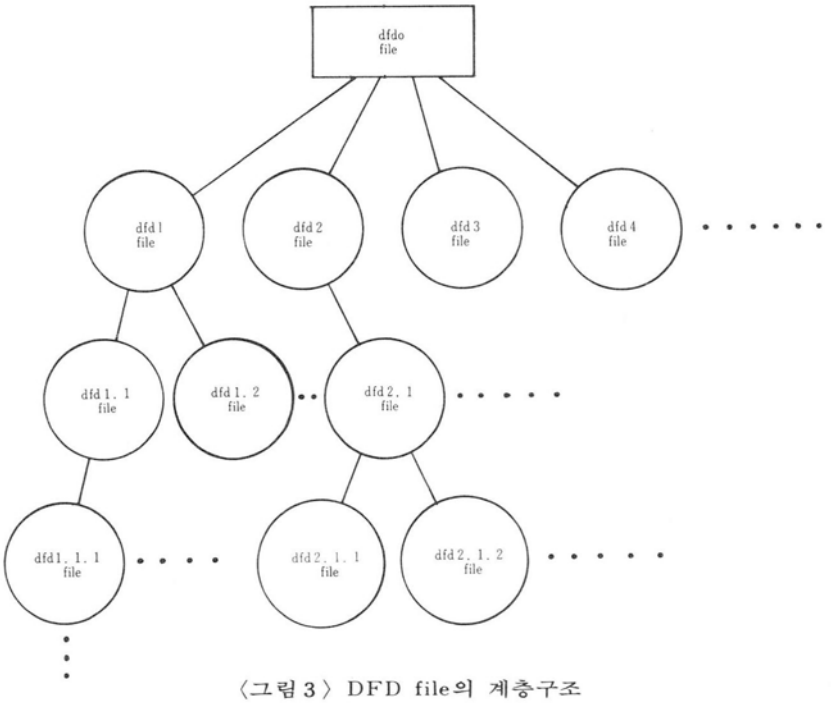
Ⅳ. DFD Editor

1. DFD Editor의 설계

DFD editor는 시스템의 requirements를 구조적인 형태의 다이어그램으로 표현해 주는 requirements분석 tool의 한 기능이다. 각 DFD는 하나의 파일 형태로 존재하며 그 파일의 이름에 따라 DFD의 계층적 구조에서의 위치를 판별하게 된다. 한 DFD내의 각 process가 다



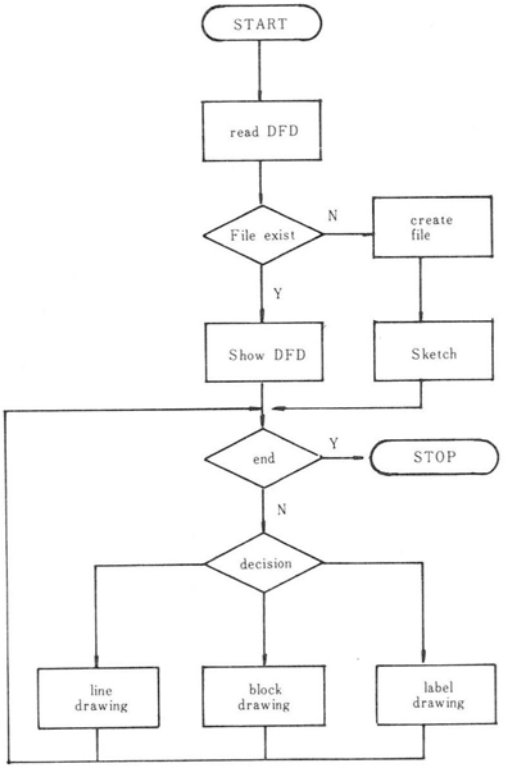
〈그림 2〉 DFD 예제



〈그림 3〉 DFD file의 계층구조

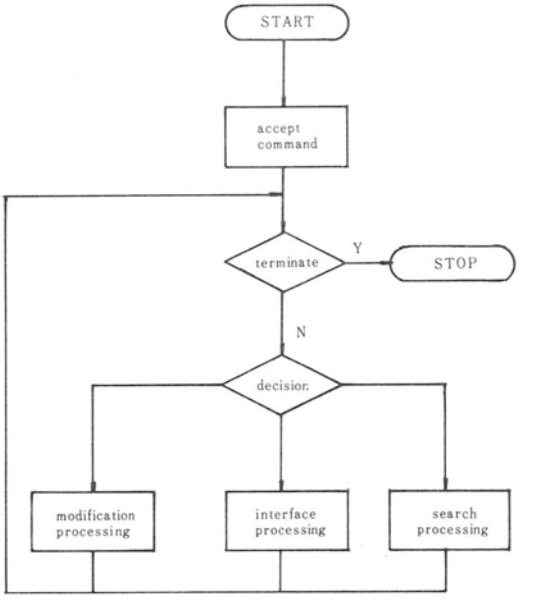
음 단계의 DFD 파일이 되며 프로세스 명은 이 파일의 제목으로서 파일 내부에 존재한다. 〈그림 3〉은 DFD 파일의 계층 구조를 나타낸다.

DFD Editor는 크게 drawing 부분과 handling 부분으로 나누어 진다. Drawing 부분에서는 원하는 DFD의 elements(process, source 및 sink, data store, 화살표, 그리고 label)를 화면상에 그려준다. 원하는 DFD가 존재할 경우에는 그 DFD를 화면에 디스플레이해 주고 editing 모우드로 변환되며 그렇지 않은 경우는 새로운 DFD 파일을 생성하고 editing 모우드로 변환되게 설계를 하였다. 새로이 생성된 DFD 파일은 원하는 DFD의 대략의 구성도를 그리게 해주는 스케칭 단계를 거치게 하였다. 이 스케칭은 원하는 DFD가 존재하는 경우는 생략되며 새로운 DFD를 생성할 경우에도 스킵할 수 있도록 설계하였다. Editing 모우드에서는 default로 블럭(process, source및 sink, 그리고 data store를 편의상 블럭으로 기술한다) drawing 단계로 들어가게 하였으며 이때 부터 블럭, 화살표, 레이블을 drawing하게 하였다. 〈그림 4〉는 drawing 부분의 개략적인 설계 flow를 나타낸다.



〈그림 4〉 Drawing 설계

Handling 부분에서는 DFD elements의 updating, 그리고 editing 모우드와 컴퓨터 시스템과의 제어 전송을 하게 해준다. 입력되는 커맨드에 따라 modification, 인터페이스, 그리고 search processing을 수행하게 설계하였다. 각 processing에 관한 세부사항은 다음 절에 기술한다. Handling 부분의 설계 flow는〈그림 5〉에 나타나 있다.



〈그림 5〉 Handling 설계 흐름도

2. DFD Editor의 기능

DFD editor는 크게 drawing 부분과 handling 부분으로 나누어 진다는 것은 앞 절에서 기술한 바 있다. Drawing 부분은 DFD를 화면상에 구현하는 것으로 다음과 같은 기능을 가진다.

(1) Sketching

새로운 DFD를 editing할 경우에 다이어그램의 대강의 윤곽을 미리 DFD 블럭의 갯수에 맞추어 설정해 주는 기능이다. 이 기능은 생략이 가능하며 기존 DFD의 editing에서는 자동적으로 통과된다.

(2) Block drawing

DFD 블럭들을 화면상에 그려주며 또한 그 블럭내에 포함될 정보 문자들을 써주는 기능이다.

(3) Line drawing

DFD의 각 블럭 사이의 dataflow의 방향을 나타내는 화살표를 화면상에 그려주는 기능이다.

(4) Labelling

DFD의 블럭간의 관계를 표시해주는 data-flow를 화면상에 써주는 기능이다.

Handling 부분은 DFD의 updating 및 제어과정을 위한 것으로 다음과 같은 기능을 가진다.

(1) DFD modification

한 DFD내의 elements를 modify 하는 기능으로 element의 위치를 변동시키는 move, move와 같은 기능이나 원래 위치의 element가 remove되지 않는 copy, 원하는 element를 지우는 delete, 그리고 delete된 element를 다시 복구시키는 undo가 있다.

(2) 시스팀 인터페이스

DFD editing 모우드와 컴퓨터 시스팀과의 인터페이스 기능으로 시스팀을 call하는 shell, shell 모우드에서 다른 DFD를 editing한 후에 원래의 DFD를 화면상에 그려주는 back, editing이 끝난 DFD를 저장하는 write, 그리고 editing중에 현 DFD를 저장하지 않고 editing을 종료하는 quit가 있다.

(3) DFD search

Editing할 DFD를 찾아가는 기능으로 현 DFD의 parent DFD 또는 child DFD를 찾아간다.

(4) Termination

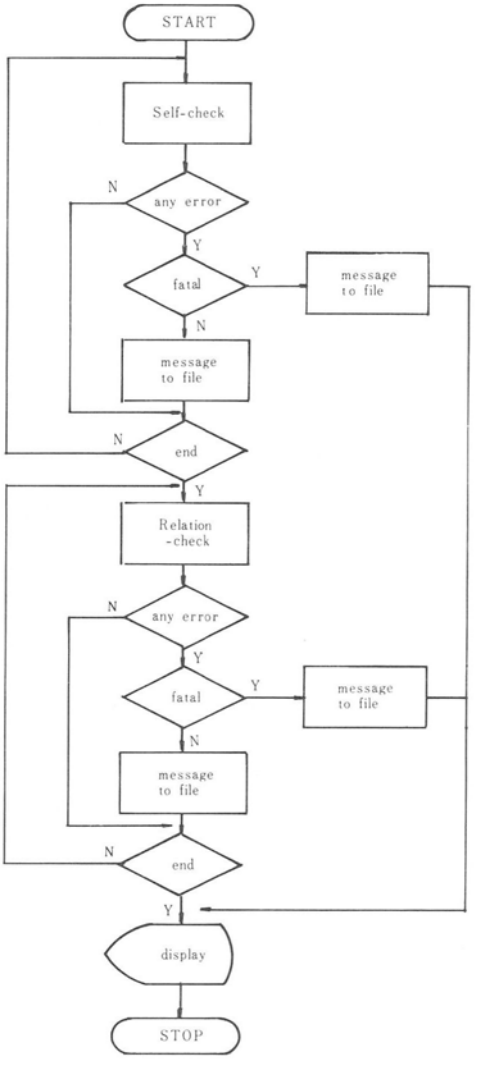
DFD editor의 전 과정을 종료하는 기능이다.

V. DFD Evaluator

1. DFD Evaluator의 설계

DFD evaluator는 DFD editor에 의해 구성된 DFD의 syntax와 각 DFD간의 데이터의 consistency 및 completeness를 검사하기 위한 requirements 분석 tool의 하나의 기능이다.

Evaluation 과정은 크게 self-checking 부분과 relational-checking두 부분으로 나누어진다. Self-checking은 각 DFD의 syntax를 검사한다. Self-checking시에 에러가 발생하면 그 에러가 evaluation 과정을 계속해 나갈 수 없을정도의 중대한 것일 경우 메시지를 지정한 파일에 수록하고 수록된 메시지를 standard output 으로 디스플레이한 후에 evaluation을 끝내게 한다. 일반적인 self-checking 에러일 경우에는 메시지를 파일에 수록한 후에 checking 과정이 완료되었나를 검사하여, 완료되었을 경우 relational-checking 과정으로 제어가 넘어가며, 완



〈그림 6〉 Evaluator 설계 흐름도

료되지 않았을 경우에는 위에 언급한 과정을 반복한다.

Relational-checking 부분에서는 DFD hierarchy를 검사하는 것으로 parent DFD와 child DFD간의 consistency 및 completeness를 evaluate한다. Relational-checking 과정은 self-checking 과정과 동일하며 relational-checking이 완료되면 지정된 파일에 수록된 evaluate 결과 메시지를 디스플레이 하면서 모든 evaluation을 마치게 된다. DFD evaluator의 전체적인 설계 흐름도는 〈그림 6〉과 같다.

2. DFD Evaluator의 기능

Evaluator의 기능이 self-checking과 relational-checking 부분으로 구분되어 있다고 앞에서 언급한 바 있다. DFD의 syntax를 검사하는 self-checking 부분에서 수행하는 기능은 다음과 같다.

(1) Non-existing DFD file access

DFD editor에서 생성되지 않은 DFD를 액세스할 경우와 사용자가 DFD 이름을 잘못 사용하였을 경우를 검사하는 기능이다. 이 에러가 발생하면 evaluator의 수행이 멈춘다.

(2) Non-existing DFD title check

액세스하고자 하는 DFD가 컴퓨터 시스팀상의 에러나 authority가 없는 다른 사용자에 의해 corrupt 되었을 경우를 검사하는 기능이다. 이 에러가 발생하면 evaluator의 수행이 멈춘다.

(3) Consistency between DFD file name and title

액세스하고자 하는 DFD 파일 이름에 속해 있는 hierarchy 번호와 title 번호가 일치하지 않는 경우를 검사하는 기능이다. 이 에러가 발생하면 evaluator의 수행이 멈춘다.

(4) Dataflow name check

Dataflow나 data store의 이름을 잘못 사용하였을 경우를 검사하는 기능이다. 'A-Z','a-z' '0-9', hypen(−), underscore(_)및 period(·)만이 허용된다.

(5) Input-only process check

한 process에 데이터의 입력만 존재하고 출력이 존재하지 않을 경우를 검사하는 기능이다.

(6) Output-only process check

한 process에 데이터의 출력만 존재하고 입력이 존재하지 않을 경우를 검사하는 기능이다.

(7) Disjoint process check

한 process에 데이터의 입력과 출력이 존재하지 않을 경우를 검사하는 기능이다.

(8) Number of dataflows and processes check

한 process에 입력되고 출력되는 dataflow의 갯수와 한 DFD내의 process의 갯수를 검사하는 기능이다. 한 process에 존재하는 dataflow의 갯수는 5개, 한 DFD에 존재하는 process의 갯수는 7개로 제한하였다.

각 DFD간의 데이터의 consistency 및 completeness를 검사하는 relational-checking 부분에서 수행하는 기능은 다음과 같다.

(1) Parent DFD existency check

Evaluate 하고 있는 DFD의 parent가 존재하는가를 검사하는 기능이다. 이때 에러가 발생하면 evaluator의 수행이 멈춘다.

(2) Title agreement with parent

Evaluate 하고 있는 DFD의 title이 parent DFD 내의 process 이름과 같은 지를 검사하는 기능이다. 이때 에러가 발생하면 evaluator 의 수행이 멈춘다.

(3) Data conservation check

Evaluate 하고 있는 DFD와 parent DFD 내의 dataflow 이름이 같은 지를 검사하는 기능이다.

(4) Data store level check

Evaluate 하고 있는 DFD와 parent DFD 내의 data store level이 정확한 지를 검사하는 기능이다.

(5) Data store conservation check

Evaluate하고 있는 DFD와 parent DFD 내의 연관있는 data store의 이름이 같은 지를 검사하는 기능이다.

Ⅵ. 결 론

Requirements 분석 tool은 시스팀 개발을 위한 requirments의 효율적이고 명확한 작성 지침이 되게 하고 분석의 자동화를 효과적으로 수행하게 해주는 도구이다. 본 논문에서는 데이터베이스 environment를 이용한 requirements

분석 tool의 각 기능들의 설계에 대한 방안을 제시하였으며 DFD를 구현하기 위한 editor 및 evaluator의 설계 및 구현에 관하여 기술하였다. DFD editor와 evaluator는 앞장에서 기술한 바와 같이 requirements를 분석하고자 하는 사용자와의 communication을 원활하게 해줄 것이며 evaluator의 수행 결과로서 생성되는 메시지는 DFD 작성과 requirements 분석의 지침이 될 것이다.

본 논문에서 제시한 requirements 분석 tool은 현재 개발중에 있으며 개발이 완료된 DFD editor와 evaluator는 requirements 분석 tool의 개발과정에서 필요한 기능을 확장 및 보완해 나갈 것이다.

〈參 考 文 献〉

1. AT&T Bell Lab., UNIX User's Manual.
2. AT&T Bell Lab., UNIX Progammer Reference Manual.
3. Gane, C. and T. Sarson, Structured Systems Analysis: Tools and Techniques, Prentice-Hall, Englewood Cliffs, 1979.
4. Teichroew, D. and E. A. Hershy III, "PSL/PSA: a Computer Aided Technique for Structured Documentation and Analysis of Information Processing System," IEEE Transactions on Software Engineering, SE-3, 1977, pp 41-48.
5. Ludewing, J., M. Glinz, H. Huser, G. Matheis, H. Matheis, and M. F. Schmidt, "SPADES-A Specification and Design System and Its Graphhical Interface," Proc. of 8 th Int. Con. on Software Engineering, 1985, pp 83-89.
6. Kerth, N. L., "Software Tools Automatic Structured Analysis," Electronic Week, 1984, pp 69-72.
7. Seiko Instrument & Electronics, GCSP-II Programming Manual.
8. DeMarco, T., Structured Analysis and System Specification, Yourdon press, New York, 1978.