□기술해설□

소프트웨어 공학을 위한 시각 환경

건국대학교 김지인*

목 차

1. 서 론

소프트웨어 공학에서 궁극적으로 성취하고자 하는 바를 한 마디로 요약하면, 가장 경제적인 방법으로 양질의 (즉, 사용자가 원하는 대로 작동하는) 소프트웨어 시스템을 생산하는 것이다[1]. 그러나, 1960년대 후반에 "소프트웨어 위기"를 타개하고자 소프트웨어 공학이 소개된 이래, 소프트웨어 공학자들의 30년 이상에 걸친 연구에도 불구하고, 아직까지 소프트웨어 위기를 극복할 수 있는 명쾌한 해결책이 제시되지 못한 실정이다. 특히 대형 소프트웨어 프로젝트의 경우, 시스템의 개발비는 예상을 훨씬 뛰어넘게 되고, 개발기간은 길어지고, 생산된 시스템은 사용자의 요구를 제대로 충족시키지 못하여, 우수한 소프트웨어의 경제적인 개발에 차질이 생기는 경우가 빈번하다.

대형 소프트웨어 시스템 개발에서 발생하는 문제점들을 살펴 보면, 사용자 요구 사항의 작성이나 분석이 제대로 이루어지지 못하여, 사용자가 원하는 대로 시스템이 설계되지 못해서 문제가 발생하는 경우가 많다. 좀더 구체적으로 말하면, 시스템이 갖추어야 할 기능이 제대로 정의되지 못하거나, 시스템의 각 기능이 수행되는데 필요한 조건이 제대로 파악되지 못하거나, 필요한 기능을 파악하지 못하여 시스템 설계에서 누락되거나, 각 기능 간의 입출력 의존도 (I/O Dependency)가 제대로 정의되지 않는 등의 경우가 많다.

본 논문에서 소개될 연구에서는, 이러한 문제점을 해결하기 위한 방법으로,

가) 명세서의 정확한 작성과 철저한 분석
나) 소프트웨어 시스템 설계의 최적화[1]
다) 코드 생성의 자동화

에 초점을 맞춘 소프트웨어 개발 방법을 제안한다. 또한, 제안된 방법을 구현하기 위하여, 각종 소프트웨어 도구들을 효과적으로 결합하고 적용한 "시각 소프트웨어 환경 (Visual Software Environment)"을 제안한다. 이 환경에서는 내부적으로 여러 가지 도구들간에 사용될 수 있는, 하나의 통일된 표현 방식을 사용하고, 명세서 작성, 명세서의 일관성 검사 (consistency checking), 설계 구조화 (design structuring), 설계의 평가 및 분석 등을 실시한다. 그리하여, 소프트웨어 기술자들의 생산성을 높히고, 그들이 생산하는 소프트웨어 시스템의 품질을 극대

*종신회원

[1] 사용자의 요구 사항을 최대한 만족시키고 시스템의 성능을 극대화하는 작업.

화시킬 수 있도록 도와주는 환경을 구축하고자 한다. 이 환경은 고성능 워크스테이션을 사용하여, 그래픽 기술에 바탕을 둔 시스템 표현 방식과 사용자 인터페이스를 채택한다.

제안되는 소프트웨어 개발 방법과 소프트웨어 시각 환경은 다음과 같은 연구 결과를 바탕으로 하여 시작되었다.

가) 사용자의 요구 사항 작성 및 분석, 명세서 작성, 그리고 시스템 설계 단계에서 시스템을 분석, 평가 및 검증하여 에러를 찾아내고 사용자의 요구 사항에 맞추어 시스템을 수정할 수 있다면, (코딩 작업이나 시스템 인테그레이션(system integration)이 끝난 후에 시스템의 에러를 발견하여 고치는 것 보다) 소프트웨어 시스템의 개발, 보수 및 유지 비용이 현저히 절감된다[3, 4, 5, 6, 7]. 이러한 검증 과정을 거친 시스템 명세서나 설계도를 바탕으로 코드를 자동 생성하면 보다 성능이 우수한 소프트웨어 시스템을 개발할 수 있다.

나) 소프트웨어 시스템의 개발, 보수 및 유지를 원활하게 해 주는 소프트웨어 도구들을 (즉, 문서 편집기, 코드 뷰어, 컴파일러, 디버거, 시뮬레이션 도구, 테스팅 도구, 등) 유기적으로 결합시켜 놓은 소프트웨어 개발 환경을 구축한다. 이 환경을 통하여 소프트웨어 개발, 보수 및 유지 과정을 자동화 내지는 반자동화하면 소프트웨어 기술자들의 생산성이 높아지고 소프트웨어 시스템의 성능이 향상된다[1, 2, 4, 5].

다) 소프트웨어 시스템의 개발, 보수 및 유지에 있어서 시스템의 이해는 필수적이다. 트리, 리스트, 그래프나 표 같은 시각 표현 방법을 이용하여 시스템을 나타내면 사용자나 소프트웨어 기술자들이 소프트웨어 시스템을 이해하기가 쉬워 진다[2, 5, 6, 9, 10].

본 논문에서는, 제안되는 소프트웨어 개발 방법을 간단히 소개하고, 그 방법론의 핵심 중의 하나인 설계 최적화를 통한 소프트웨어 개발 방식에 대하여 토론하고자 한다. 2절에서는 이 연구와 관련된 다른 연구들에 대하여 알아본다. 3절에서는 요구 사항 분석과 설계 최적화에 초점을 맞춘 소프트웨어 개발 체제를 설명한다. 4절에서는 설계 최적화 과정의 하나인 설계 구조화에 대해 설명한다. 마지막으로 5절에서 앞으로의 연구 방향에 대하여 살펴 보고, 이 논문을 맺는다.

2. 관련 연구

소프트웨어 개발에 있어서, 프로그램의 코딩 이전 단계인 시스템 설계 과정에서 시스템의 분석, 평가 및 검증을 할 수 있다면, 사용자의 요구 사항에 맞는 소프트웨어 시스템을 보다 경제적으로 생산해 낼 수 있을 것이다. 그러기 위해서는, 사용자의 요구 사항을 제대로 분석하여 정확한 명세서를 작성해야 한다. 명세서는 이해하기 쉽고 명료한 수단으로 작성되어야 한다. 또한, 명세서가 실행 가능하게 작성되었다면, 명세서의 분석 및 평가가 효과적으로 수행될 수 있다. 완성된 명세서를 가지고 시스템을 설계한다. 시스템의 기능을 세분화하고 구조화하여 효율적인 설계가 되도록 한다. 마지막으로, 완성된 설계를 가지고 자동으로 코드를 생성할 수 있는 도구가 필요하게 된다. 이에 관련하여 지금 진행 중이거나 완료된 연구들을 살펴보면 다음과 같다.

[2]에서는 MODEL이라는 방정식 언어를 사용하여 명세서를 작성하고, 번역, 점검, 테스팅 및 검증을 수행하고 있다. 사용자의 소프트웨어 이해도를 높히기 위하여 어레이 그래프라는 자료 흐름 그래프를 이용하여 명세서를 작성하고 실행하는 시각 프로그래밍 환경을 제공하고 있다. 작성된 시각 명세서로 부터 C, Ada, PL/1 같은 프로그래밍 언어로 작성된 프로그램을 자동 생성할 수 있다.

ACSR (Algebra for Communicating Shared Resources)라는 프로세스 대수 (Process Algebra)에 기초한 형식 언어를 가지고 정의한 시각 명세서 언어인 GCSR (Graphical Language for Communicating Shared Resources)를 사용하여 명세서를 작성하고, 명세서 수준에서 시스템의 시뮬레이션 및 수학적 검증을 시행하려는 연구도 있다[5]. 수학적인 방법

으로 시스템의 옳고 그름을 증명하려는 수학적인 프로그램 증명법은 너무 어렵고 비용이 많이 필요하므로 실시간 시스템 같이 시스템의 정확도가 반드시 보장되어야 하는 분야에 주로 쓰이고 있다.

Graspin [7]은 각종 방법론과 시각 언어를 다루는 CASE (Computer Aided Software Engineering) 도구들을 함께 제공하는 프로그래밍 환경으로, 시스템의 분석과 설계 과정에 초점을 맞추고 있다.

설계로 부터 코드를 생성해 주는 방법론에는 실시간 시스템을 객체 지향 방법으로 설계한 후, 해당되는 Ada 코드를 만들어 주는 ADA-RTS(Ada based Design Approach for Real Time Systems)라는 방법론도 있다[8].

시스템 공학 분야에서는 미 해군 연구소가 주관하여 연구 중인 Destination (Design Structuring and Allocation Optimization) 이라는 방법론은 시스템 기술자들에게 설계 최적화와 교환 분석 (trade-off analysis) 을 할 수 있는 체제를 제공하고 있다[4].

Alford가 개발한 RDD-100이라는 시스템 공학 환경에서는, 직접 실행이 가능한, 행위도 (Behavior Diagram)라는 표현법을 사용하여 시스템의 기능을 시각적으로 표현하고 일반 문자열 정보를 구조적으로 나타내고 있다[6]. 여기서도 대형 시스템 개발에 있어서 설계 과정의 중요성을 강조하고 있다. RDD-100를 구성하는 도구들은, 행위도를 통해 시각화 된 시스템 설계를 공유하게 되어, 사용자의 이해도를 높히고 개발 작업의 효율성을 향상시킨다.

3. 소프트웨어 개발 체제

일반적으로 소프트웨어 시스템의 개발 방법은 사용자의 요구 사항 작성에서 출발하여, 명세서를 작성하고, 세부 설계를 한 후, 코딩 작업을 하게 된다. 그리고, 시스템의 테스팅 또는 검증 작업을 실시한다. 본 논문에서는 기존의 시스템 개발 방법을 다음과 같이 확장하여 그림 1과 같이 제안한다.

가) 명세서 작성, 분석 및 점검 : 실행 가능한 명세서 작성 언어를 사용하여 명세서를 작성한다. 사용자의 요구 사항에 맞는지를 일관성 점검, 요구 사항 누락 여부 점검, 시뮬레이션 등의 방법을 통하여 확인한다.

나) 설계 구조화 : 완성된 명세서를 바탕으로 시스템 설계를 한다. 시스템 설계는 시각적으로 표현되며 환경 내부의 도구들에 의해 공유된다. 설계 구조화 작업을 통하여 시스템 설계의 최적화를 추구한다.

다) 코딩의 자동화 : 작성된 설계도로 부터 코드를 자동으로 생성해 내면 소프트웨어 기술자들의 시간과 노력을 절약하여 줌으로써 생산성을 제고할 수 있다.

라) 시스템 개발의 반복 : 시스템이 사용자의 요구 사항을 만족시킬 때까지, 설계, 평가, 수정 작업을 반복하여 실시한다.

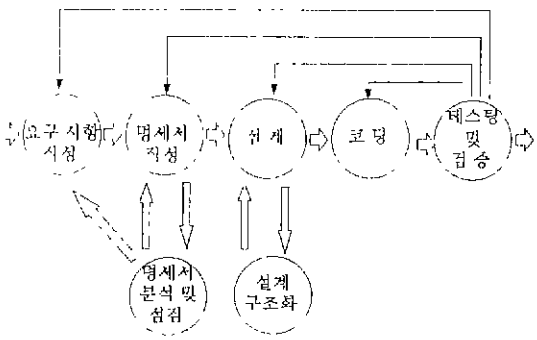


그림 1 소프트웨어 개발 체제

소프트웨어 도구들을 이용하여 이러한 방법론을 구현하고 하나의 소프트웨어 개발 환경을 구축하는 작업은 많은 연구와 시간을 필요로 한다. 이 연구는 다음과 같이 세분화될 수 있을 것이다.

가) 시스템 명세서 분석을 위한 실행이 가능한 명세서 언어의 개발 : 간결하면서도 이해하기 편하여 사용자 요구 사항으로 부터의 변환이 용이한 명세서 언어를 개발한다.

나) 시스템 명세서 분석 기법과 도구의 개발 : 사용자의 요구 사항에 맞추어 명세서의 분석을 실시할 수 있도록, 일관성 점검,

요구 사항 누락 점검, 시뮬레이션 등의 기법과 이에 필요한 도구를 개발한다.

다) 시스템 설계 자동화 및 표현법 연구 : 명세서로 부터 시스템 설계를 자동으로 해주는 방법을 연구하고 구현한다. 시스템 설계를 상태도 (state diagram), 자료 흐름도 (data flow diagram), 개체 관계도 (entity relationship diagram) 등 소프트웨어 기술자들에게 친숙한 그림을 사용하여 시각화 표현하고 각종 도구들에서 잘 사용될 수 있도록 연구한다.

라) 소프트웨어 설계 최적화 방법의 연구와 이에 필요한 소프트웨어 도구의 개발 : 설계 구조화를 통하여 시스템 기능을 세분화하고 각 기능 간의 관계를 정의하여 시스템의 효율성을 극대화하는 방법을 연구한다.

마) 설계도로 부터 프로그램을 생성해 주는 번역 도구의 개발 : 최적화된 설계도를 가지고 소프트웨어 시스템을 자동 생성한다.

바) 소프트웨어 개발 환경의 사용자 인터페이스 개발 : 각종 소프트웨어 도구를 사용할 때, 하나의 통일된 표현 방식을 이용하여 시스템을 표현하고, 하나의 동일한 입출력 및 상호작용 체제를 통하여 각종 도구를 사용하기 위한, 사용자 인터페이스 방식을 개발한다.

4. 설계 구조화

4.1 개 요

설계 구조화 (DS, design structuring)란 시스템 설계를 그래프로 나타내어 시각적으로 표현하기 위한 공학적 행위를 말한다[3, 4]. 설계 구조화의 목표는 시스템 사용자의 요구 사항에 맞추어 시스템 설계를 최적화하는 것이다. 그림 2에 시스템 설계 구조화 과정과 필요한 자료들이 표현되어 있다.

설계 구조화 과정을 수행하려면 사용자의 요구 사항, 설계 구조화의 목적과 시스템 설계도가 입력되어야 한다. 그 다음에, 설계 구조화를 위한 전략이 수립된다. 설계 구조화 전략은 여러 가지의 설계 구조화 기법의 조합으로 구성

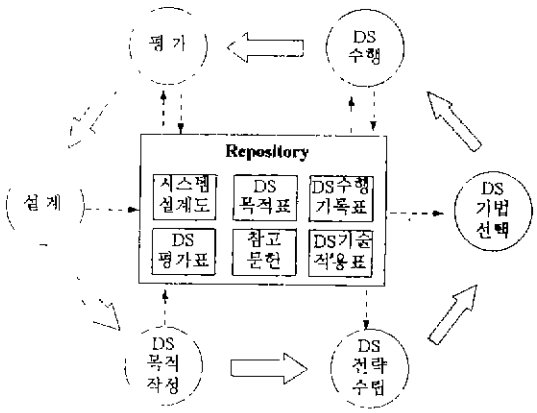


그림 2 설계 구조화 과정 및 자료 흐름

된다. 설계 구조화가 실시되면 기존의 시스템 설계가 설계 구조화 목적에 맞게 변형된다. 그러면, 변형된 설계도가 앞서 정의된 설계 구조화 목적을 얼마나 달성했는지가 평가되어야 한다. 이 평가의 결과에 따라, 시스템 설계를 수정하기도 하고 시스템 설계를 완성하기도 한다.

설계 구조화 작업 과정, 수행 결과, 평가 결과, 등은 모두 기록되어 저장되고 분석되어 다음 번 설계 구조화 작업에 참고자료가 된다.

4.2 기본 연산자

설계 구조화에서는 시스템을 그래프의 형태로 표현하므로, 설계 구조화 작업은 기본적으로 일련의 그래프 변형 작업의 흐름으로 간주될 수 있을 것이다. 따라서 설계 구조화는 분해 (Decomposition) 연산자, 합성 (Recomposition) 연산자, 분열 (Fragmentation) 연산자, 융합 (Defragmentation) 연산자, 치환 (Replacement) 연산자의 다섯 가지의 그래프 변형 기본 연산자로 나뉠 수 있다. 각각의 연산자에 알아보기로 한다.

4.2.1 분해와 합성

분해 연산자는 그래프 표현의 계층 구조 하에서 한 단계 아래의 세부 구조를 만들어 준다. 그림3에서 보듯이, 상위 구조의 노드 A가 하위 구조의 그래프로 분해 된다. 하위 구조의 그래프는 A_1, A_2, A_3, A_4의 4개의 노드와 X_1, X_2, Y, Z의 화살표로 구성된다. 합성은 분해의 역으로

서, 하위 구조의 자세한 그래프를 하나의 노드로 줄여서 표현할 때 사용된다. 그림3의 하부 구조의 그래프가 상위 구조의 노드 A로 간략화되는 것이 합성이다.

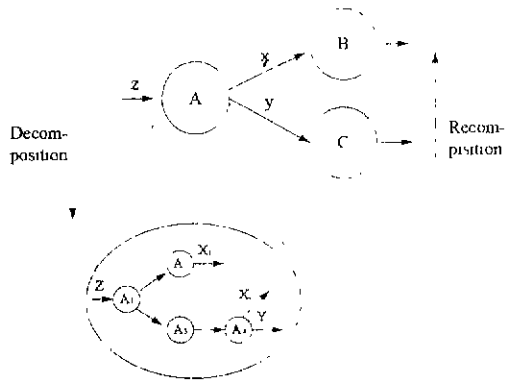


그림 3 분해(Decomposition)와 합성(Recomposition)

분해와 합성 연산자는 시스템 설계도를 계층적으로 구성하여 대형 시스템을 표현하고자 할 때 유용하게 사용된다. 시스템의 전체적인 구조를 보고 싶을 때에는, 자세한 정보를 갖는 그래프들을 합성하여 간략화하여 표현하여 본다. 반대로, 시스템의 세부 사항은 노드를 분해하여 자세히 들여다 보게 된다.

4.2.2 분열과 융합

분열은 설계도 계층 구조에서, 상하위 수준을 바꾸지 않고 동일한 수준에서 시스템의 설계를 보다 복잡한 형태로 바꾸는 연산자이다. 같은 설계 요소의 복제 (Replication)는 분열의 한 특별한 예이다. 그림4에서 나타낸 것 처럼, 노드 A와 화살표 X, Y가 분열하면 A_1, A_2, A_3, A_4로 바뀌어 진다. 융합은 반대로 복잡한 그래프를 같은 수준의 그래프 상에서 간단하게 만들어 주는 연산자이다.

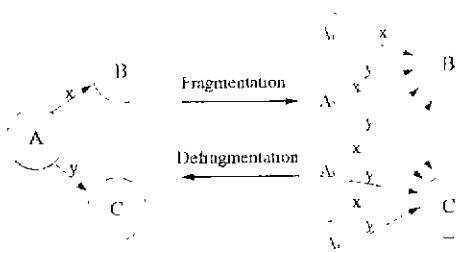


그림 4 분열(Fragmentation)과 융합(Defragmentation)

그림 4에서 표현된 예에서는 노드 A를 분열시켜서 복제함으로써, 전체 시스템의 신뢰도 (reliability)를 향상시켰다. 반대로 융합을 하게 되면, 그래프의 노드와 화살표가 줄어 들어, 시스템의 복잡도 (complexity)를 낮출 수 있다.

4.2.3 치환

치환은 분해, 합성, 분열, 융합을 제외한 모든 설계 구조화 연산을 총칭한다. 그림5에 보이는 치환은, 결정점 (decision point)을 세개 가지고 있는 왼쪽의 그래프를 가지고, 결정점의 갯수를 바꾸지 않은 채, 가운데 그래프와 오른쪽 그래프로 변환시킨 것이다.

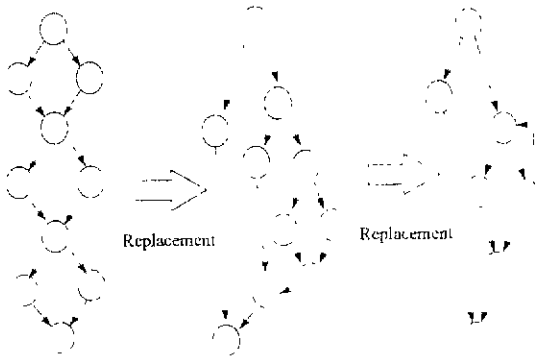


그림 5 치환(Replacement)

5. 결 론

본 논문에서는, 시스템 설계 과정에 (즉, 명세서 작성 및 분석, 설계의 최적화, 코드의 자동생성 등) 초점을 맞춘 소프트웨어 개발 방법론을 제안하고, 그 중에서 설계 구조화 방법에 대하여 간단하게 살펴 보았다. 이미 언급한 대로, 제안된 소프트웨어 개발 방식은 성능이 우수한 소프트웨어 시스템을 경제적으로 제작할 수 있는 방법이다. 이 방식을 바탕으로 CASE 도구의 집합인 소프트웨어 개발 환경이 구축될 것이다. 이 연구를 위하여 명세서 언어 개발, 명세

서 분석 기법 연구, 설계 자동화 기법 연구, 설계 구조화 기법 연구, 설계의 시각 표현 방식 연구, 코드 자동 생성 도구 구현, 사용자 인터페이스 방식 연구 등 다양한 세부 과제를 수행하여야 한다. 우리는 우선 설계 구조화 방법을 연구하고 필요한 도구를 개발할 계획이다. 구체적으로, 시스템의 시각 표현 방법 연구, 설계 구조화 알고리즘의 개발, 설계 평가 방법의 개발 등이 수행될 것이다. 이러한 설계 시각화 도구와 설계 구조화 도구 및 그 시각 인터페이스를 구현하는데는 시각 환경 구축 도구인 Escalante [9]를 이용할 계획이다.

참고문헌

[1] 우치수, *Computer-Aided Software 공학*, 상조사, 1994.

[2] Kim, J-I., "An Environment for visualization, reliability and knowledge acquisition in equational programming," Ph. D. dissertation, University of Pennsylvania, 1993.

[3] Kim, J-I. and Lock, E., "Design structuring for system engineering," *Proceedings of CSESAW*, pp 158-170, July 1993.

[4] Kim, J-I. and Lock, E., "An environment to support design structuring," *Proceedings of CSESAW*, pp 107-113, July 1994.

[5] Choi, J-Y., Lee, I-S., Kim, J-I. and Lock, E., "Software tools for formal specification and verification of distributed real-time systems," *Proceedings of CSESAW*, pp 99-106, July 1994.

[6] Alford, M., "Strengthening the systems engineering process," *Proceedings of NCOSE*, October 1991.

[7] Mannucci S. et al., "Graspin : a structured development environment for analysis and design," *IEEE Software*, pp 35-43, November 1989.

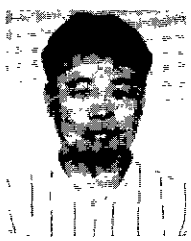
[8] Gomma, H., "ADARTS-an Ada based design approach for real time systems, version 1.0," Software Productivity Consortium Technical Report, SPC-TR-88021, August 1988.

[9] McWhirter, J. D. and Nutt, G. J., "Escalante : an environment for the rapid construction of visual language applications," University of Colorado at Boulder, Technical Report CU-CS-692-93, December 1993.

[10] Lee, M., Prywes, N. and Lee, I-S., "Automation of analysis, simulation and understanding of real time large Ada software," University of Pennsylvania, Manuscript, November 1994.

[11] Glinert, E. P., Visual Programming Environments : Paradigms and Systems, IEEE Computer Society Press, 1990.

[12] Glinert, E. P., Visual Programming Environments : Applications and Issues, IEEE Computer Society Press, 1990.



김 지 인

1982 서울 대학교 컴퓨터 공학과 졸업 (학사)
1984 한국과학기술원 전산학과 졸업 (석사)
1982~1987 금성통신 (주) 주임 연구원
1994 University of Pennsylvania 전산학과 졸업 (박사)
1993~1995 CCCC 전산 과학자
1995~현재 건국 대학교 전자계산학과 조교수

관심 분야 : Human Computer Interaction, Virtual Reality, Medical Image Processing, Software Engineering