□사례발표□

소프트웨어 조립·분해 자동화의 구현사례

유 홍 준†

◈ 목 차 ◈

1. 소프트웨어 조립·분해 기술의 중요성
2 소프트웨어 조립·분해의 배경기술
3. 소프트웨어 조립·분해 구현기술의 핵심
4. 부품조립을 통한 소프트웨어제작의 기본방법
5. 소프트웨어 조립·분해의 원리
6. 소프트웨어 유지보소에의 적용 실험 사례
7 결론 및 향후 과제

1. 소프트웨어 조립·분해 기술의 중요성

최근들어 소프트웨어 공정에서 유지보수가 차지하는 비율이 90%이상으로 증대됨에 따라 이를 원활하게 수행할 수 있도록 하는 소프트웨어 조립·분해 기술의 필요성이 더욱 강조되고 있다.

그 이유는 명쾌하다. 소프트웨어의 개발생산 인력을 10%, 유지보수 인력을 90%라고 보았을 때, 개발 생산성을 2배 향상시켰을 때 절감되는 인력은 전체적으로 5%에 불과하지만, 유지보수성을 2배 향상시켰을 때에는 전체적으로 무려 45%이상의 인력이 절감될 수 있기 때문이다. 그렇게 되면, 절감되는 인력의 재배치를 통해 신규개발업무를 보다 활성화시킬 수 있게 되며, 이것은 조직의 업무 리엔지니어링과 접목되어 회사전체의 체질개선과 대외적인 경쟁력 강화로 이어지게 되는 것이다.

소프트웨어의 유지보수성을 향상시키기 위해서는 현재와 같은 수작업으로는 한계가 있기때문에 적은 인력으로 최대의 유지보수업무량을 감당해낼 수 있도록 이제까지 수작업으로 하던 일들을 자동화시켜주어야 한다. 하지만, 자동화를 위해서는 먼저 해결해야 할 사항이 있다. 소프트웨어 개발 및 유지보수 기술을 예술의 차원에서 공학의 차원으로 체계화해야 하는데, 이를 위한 선결사항이 바로 소프트웨어 부품의 표준화·규격화이다 소프트웨어 부품이 표준화·규격화되면 소프트웨어를 공장에서 하드웨어를 조립·분해하는 것처럼 정밀하게 다룰 수 있게 되며, 궁극적으로 무인공장에서 로봇에 의해 하드웨어가 자동조립되는 것처럼 소프트웨어 무인공장에서 자동으로 유지보수될 수 있게 된다.

이처럼, 소프트웨어의 자동화 기술에 표준화·규격화된 소프트웨어 부품의 조립·분해 및 추상화 기술은 아주 중요한 비중을 차지한다.

소프트웨어의 조립·분해는 그 대상이 소프트웨어이기때문에 그것을 조립·분해하는 도구도 어떠한 물리적인 기계가 아니라, 소프트웨어 프로그램이 행해주게 된다.

2. 소프트웨어 조립·분해의 배경기술

소프트웨어 조립·분해를 자동화하기 위해서는

† 정회원 : 홍준 정보처리학원 원장

소프트웨어 부품이 조립되었을 때 추상화되어 조립된 여러개의 부품들이 전체적으로 하나의 모습으로 보이도록 하고, 분해된 부품들은 구체화되어 여러개의 개별적인 모습으로 보이도록 하는 것이 가능한 추상화사다리(ladder of abstraction) 단계를 작업영역으로 선택해 주어야 한다.

우리가 소프트웨어 부품을 조립·분해하는 단계로서 분석단계, 설계단계 및 코딩단계의 3단계를 작업영역의 선택단계로서 생각해 볼 수 있는데, 분석단계는 시스템분석가의 주관적인 시각이 개입되는 단계이므로 추상도가 너무 높고, 코딩단계는 프로그램 코드 자체가 마치 영어의 단어처럼 나눠져서 나열되는 형식으로 문장을 구성하기 때문에 이것을 조립하여 전체적으로 하나의 모습으로 보이게 하는 것이 불가능하다.

따라서, 소프트웨어 IC를 구현하여 추상화사다리의 상하를 자유자재로 오르내리기 위해서는 분석과 코딩의 중간에 위치한 설계단계를 부품조립·분해의 자동화를 위한 소프트웨어 가공공장의 핵심영역으로 선택해주는 것이 바람직하다.

또한 부품은 패턴형태로 시각화하는 것이 조립·분해를 자동화하는 데에 꼭 필요하기 때문에 소프트웨어 부품은 설계도 형태로 만들어져야만 하며, 설계도 중에서 가장 조립·분해에 적합한 것을 선정하여 자동화에 이용하는 것이 중요하다.

그렇다면, 기존의 설계도는 과연 소프트웨어 IC를 만드는 데에 있어서 어떠한 장단점을 가지고 있을까?

제1세대 설계도인 순서도(flow chart)의 설계처리는, 처리라든가, 판단이라든가, 초기화라든가와 같은 기능별로 약속된 도형을 그 속에 써넣는 문자수에 따라 크기를 임의로 정하여 비규격적으로 그리고, 그 비규격적으로 그려진 기능도형들을 비정형화된 화살표로 연결해가는 방법이었다. 따라서, 순서도를 소프트웨어 IC의 재료로 사용하는 것은 불가능하다.

제2세대 설계도로서 등장한 것이 NS챠트이다. 이것은 순서도를 상자형태로 변형하여 연접시켜 그려나감으로써 기술밀도를 높히고 goto사용을 금지시킨 형태의 설계도로 제1세대보다 진전된 형태를 이루었다. 하지만, 이것도 폐쇄된 형태의 상자내에 요소를 기입해넣어야 하는 등의 결정적인 단점으로 인해, 선택요소 등의 내포단계에 따라 도형의 크기가 비정상적으로 작아지거나 커지는 등의 비규격성을 초래하여 소프트웨어 IC의 재료로 사용하는 것이 불가능하다.

제3세대의 설계도는 제1세대와 제2세대 설계도의 단점을 상당부분 보완한 PAD, HCP, SPD, YAC 등과 같은 규격화된 제어구조중심의 작도형태를 가지는 설계도이다. 이것들은 제2세대 설계도와 합하여 통칭적으로 구조화도(structured diagram)라고 호칭되고있으며, 순차(sequence)·선택(selection)·반복(iteration) 등과 같은 제어구조별로 약속된 도형들을 접착 또는 선으로 연결하여 구현해 가는 형태의 설계도이다. 제3세대 설계도는 PAD 등과 같은 극히 일부의 설계도를 제외하고는 대부분이 개방식으로 표기를 해나가는 진전을 이룸으로써, 설계처리장치에 의한 소프트웨어 설계를 위한 프로그램의 제어구조의 인식의 문제점을 해결할 수가 있었으며, 자동화도구에 의해 프로그램의 제어구조를 인식하는 기술을 구현하는 것이 수월해짐으로써, 소프트웨어 개발의 생산성을 향상시킬 수 있게 되었다.

하지만, 제3세대의 설계도는 설계표기요소 자체가 추상화 및 구체화의 추상화사다리를 오르내릴 수 없도록 고정되어 자동 조립·분해의 과정에서 부품의 조립삽입·분해제거·가공 등의 유지보수 작업능률의 향상에는 적합하지 않아서 역시 양질의 소프트웨어 IC로서 사용되기에는 역부족이다.

제4세대 설계도는 설계표기요소 자체를 조립해 나가는 형태의 SEC(구조화능률도: Structured Efficiency Chart)이다. 이것은 제어구조를 세계최초로

정상상황과 비정상상황으로 나누어 환경의 변화에 대응하는 형태로 체계화하고, 소프트웨어 설계를 워드프로세서를 이용해서도 할 수 있도록 보다 체계화 함으로서 설계기술의 진보를 꾀하였다. 하지만, 이것도 소프트웨어의 개발 생산성의 향상에는기여하였으나, 유지보수를 감안한 양질의 소프트웨어 IC로서 사용됨에 있어서는 단점을 노출하였다.

설계도가 제4세대에 이르기까지의 과정에도, 급속한 컴퓨터의 하드웨어기술의 발전으로 인한 가격하락과 기능향상으로 일반보급이 급진전되었다. 범세계적으로 구축된 정보화 환경하에서 일반사용자도 컴퓨터에 관해 상당히 풍부한 지식을 갖도록 되어, 소프트웨어에 대한 사용자의 만족도의 수준도 더욱 높아지게 되었으며, 요구사항도 한층 복잡다양하게 되었다.

이 때문에, 소프트웨어 개발의 비중보다 소프트웨어 유지보수의 비중이 급속히 높아지게 되었지만, 제1세대에서 제4세대에 이르기까지의 종래의 설계도는 소프트웨어의 조립·분해 기술에 꼭 필요한 소프트웨어 설계도 구조의 추상화, 구체화 및 부분적인 변경이라든가 확장이라든가 보수 등의 유지보수성의 향상에의 대처가 어려웠다

3. 소프트웨어 조립·분해 구현기술의 핵심

상기와 같은 소프트웨어 IC를 만들어 냄에 있어서 노출되었던 기술적인 문제점은 제5세대의 설계기반인 SOC(Structured Object Component)이 등장함으로써 완전히 해결되게 된다.

SOC은 제1세대 내지 제4세대와는 달리 설계도(design diagram)의 개념에서부터 벗어나 설계부품(design component)의 개념으로 진보하였다.

이 기술은 조립식소프트웨어 설계처리장치에 의한 소프트웨어 설계처리작업에 있어서 소프트웨어 설계도를 종래의 순서도라든가 구조화도의 설계처리방법과 같이 일일이 설계도형을 연결시켜 그리는 것이 아니라, 소프트웨어 설계의 용도별로 사전에 기본부품(basic component), 덩이부품(block component) 및 구조부품(structure component)의 3종류의 패턴으로 나누어 조립식 소프트웨어 설계처리장치의 내부에 있는 별도의 기억수단에 있는 설계부품기억부에 저장시켜둔 표준화·규격화된 조립식 소프트웨어패턴 설계부품을 마치 공장에서 산업용로보트에 의해 하드웨어 칩을 자동화된 공정으로 조립 또는 분해하는 것과 동일하게 언제라도 원하는 위치에 신속정확하게 조립해 넣을 수 있으며, 불필요하게 된 소프트웨어 패턴부품은 기존의 설계도면에 손상을 가하지않고 언제라도 깨끗하게 원하는 부분만 간단히 분해하여 제거할 수 있도록 한 기술이다. 또한, 구현되어 있는 설계조립품중에서 가독성이라든가 처리기능의 구분 등의 필요에 따라서 인접된 기본부품이라든가 덩이부품이라든가 구조부품 등과 같은 패턴 설계부품들을 묶어서 추상화시키는 것을 가능하게 하는 기술이다

이 기술을 적용하면, 개발 및 유지보수에 있어서 하드웨어를 취급할 때와 같은 접근방법의 적용이 수월해진다. 소프트웨어를 하드웨어 칩(chip)을 조립하는 식으로 제작하는 경우에는 표준화·규격화시켜 제작해 두었던 가장 작은 부품들을 서로 결합시켜 조금씩 큰 부품으로 추상화시켜나가면서 조립하면 된다. 또, 유지보수할 때는 필요한 크기·범위만큼의 덩이를 분해하여 필요한 만큼의 크기와 모양의 덩이로 대체하여 컴퓨터의 스롯에 카드를 꽂았다가 뺐다가 하는 것처럼 재조립해주면 된다.

이처럼, 소프트웨어의 조립·분해를 위한 부품을 추상화의 정도에 따라, 기본부품, 덩이부품, 구조부품의 세가지 형태로 나누어서 하드웨어의 부품처럼 표준화·규격화시켜주면 불과 몇 개의 부품만으로 어떠한 구조의 소프트웨어 뼈대(software frame)도 조립·분해가 될 수 있게 되는 것이다. 따라서, 제5세대 설계기반인 SOC(Structured

Object Component)의 단계에 이르러서는 소프트웨어 IC 조립·분해 공정을 자동화시키는 것이 수월해짐으로써 개발생산성 및 유지보수성의 획기적 증진이라는 두가지 해결 과제를 동시에 해결할 수 있게 되었다.

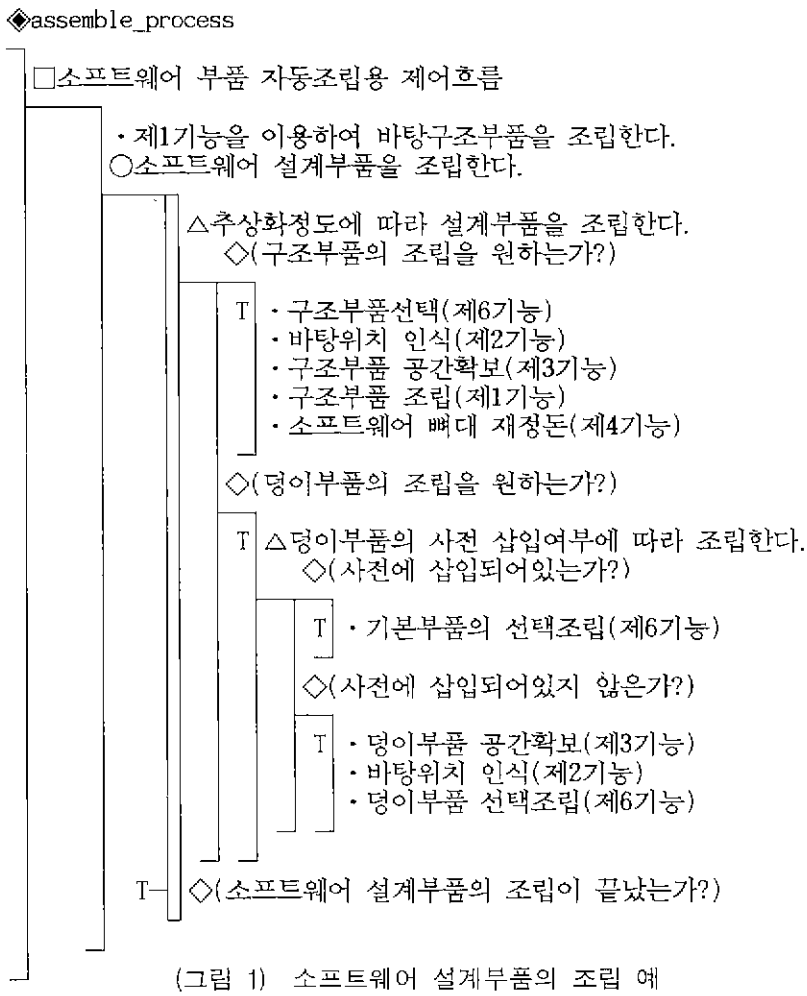
4. 부품조립을 통한 소프트웨어 제작의 기본방법

자동화된 방법으로 부품을 조립하기 위해서 필요한 소프트웨어의 기능은 다음과 같다.

제1기능 부품제작기능: 소프트웨어 설계부품의 모양을 자동적으로 만들어 주는 기능

제2기능 위치인식기능: 작성된 소프트웨어 뼈대 속에 새롭게 삽입해 넣고자 하는 설계부품이 있을 때, 이 부품의 삽입위치를 자동적으로 인식하여 지정해주는 기능

제3기능 부품조립기능: 제2기능에 의해 인식된 삽입위치에 새로운 소프



(그림 1) 소프트웨어 설계부품의 조립 예

제3기능 부품조립기능: 제2기능에 의해 인식된 삽입위치에 새로운 소프트웨어 설계부품을 자동적으로 삽입하여 정확하게 조립해 넣을 수 있는 기능

제4기능 새판짜기기능: 새로운 소프트웨어 부품이 삽입됨과 동시에 소프트웨어 뼈대의 전체 모양을 자동적으로 정돈하여 재구성해주는 기능

제5기능 영역확장기능: 새로운 기능을 추가할 수 있도록 소프트웨어 뼈대 사이를 확장할 수 있는 기능

제6기능 부품가공기능: 기본적인 소프트웨어 뼈대를 짜집기(부품의 선택, 복사, 이동, 삭제 등)하는 기능

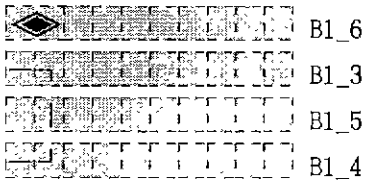
위의 6가지 기능을 이용하여 소프트웨어 부품을 조립하는 예를 SOC(Structured Object Component)을 조립하여 나타낸 것이 (그림 1)이다.

이와 같은 자동화 기술을 적용하여 만들어진 소프트웨어 설계부품 조립용 소프트웨어를 가동시키면 놀랄만큼 빠른 속도로 설계부품을 자동조립할 수가 있다.

5. 소프트웨어 조립 · 분해의 원리

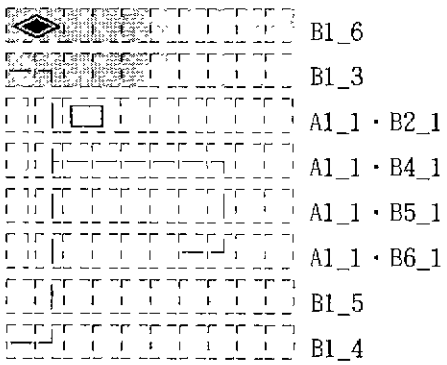
5.1 소프트웨어 설계부품 조립의 원리

(그림 2)는 거듦바탕구조부품(부품번호: C1_2)의 조립예를 표현한 것이다.
이 때의 조립공식은 B1_6 · B1_3 · B1_5 · B1_4 가 되며 조립된 후에는 C1_2부품으로 추상화된다.



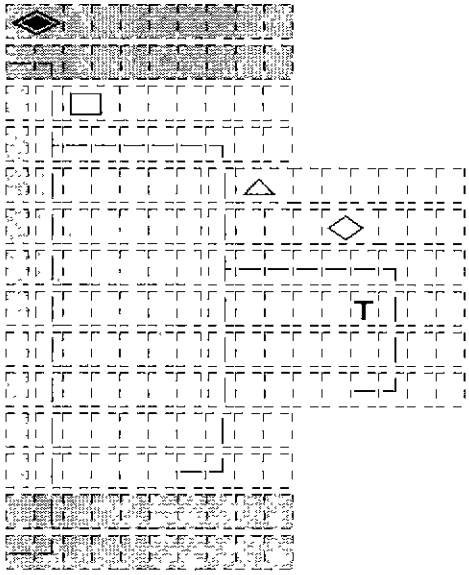
(그림 2) 거듦바탕구조부품의 조립예

(그림 3)은 (그림 2)의 C1_2의 구조부품에다가 한잇따름구조부품(부품번호: C2_1)을 조립해 넣은 예를 표현한 것이다. 이 때의 조립공식은 한잇따름구조부품의 조립공식이 B2_1 · B4_1 · B5_1 · B6_1 인 동시에 한잇따름구조부품의 추상화 부품번호가 C2_1이므로, 거듦바탕구조부품(부품번호: C1_2)속에 한잇따름구조부품(부품번호: C2_1)이 조립될 때의 공식은 B1_6 · B1_3 · (A1_1 · B2_1) · (A1_1 · B4_1) · (A1_1 · B5_1) · (A1_1 · B6_1) · B1_5 · B1_4 가 됨과 함께, 조립후에는 C1_2 -> C2_1 부품으로 추상화된다.



(그림 3) 한잇따름구조 부품의 추가 조립예

(그림 4)는 (그림 3)의 구조패턴에다 한가림구조부품(부품번호: C2_1)을 조립해 넣은 예를 표현한 것이며, 조립후에는 C1_2 -> C2_1 -> C2_2 로 추상화된다.



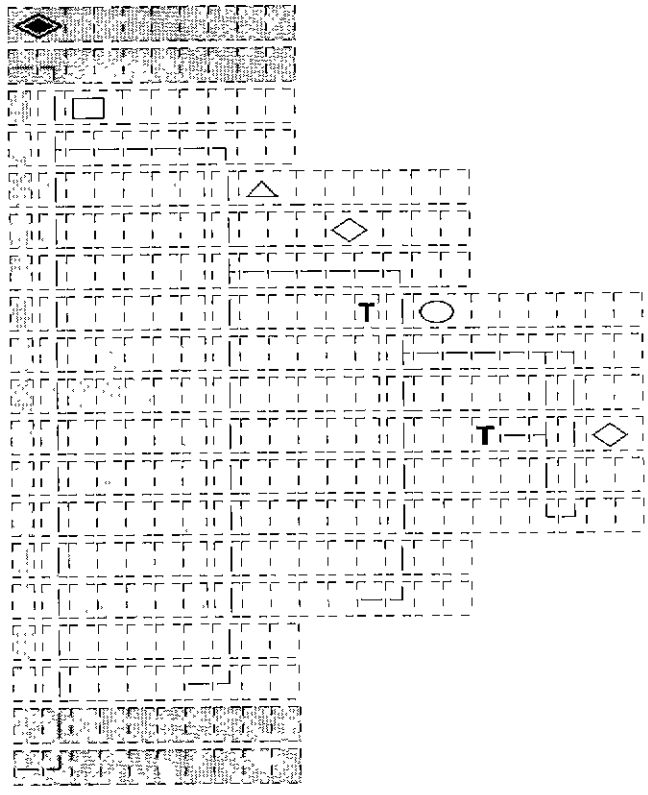
(그림 4) 한가림구조 부품의 추가 조립에

(그림 5)는 (그림 4)의 구조패턴에다 사이끝되풀이구조부품(부품번호: C2_6)을 조립해 넣은 예를 표현한 것이며, 조립후에는 C1_2 -> C2_1 -> C2_2 -> C2_6 으로 추상화된다.

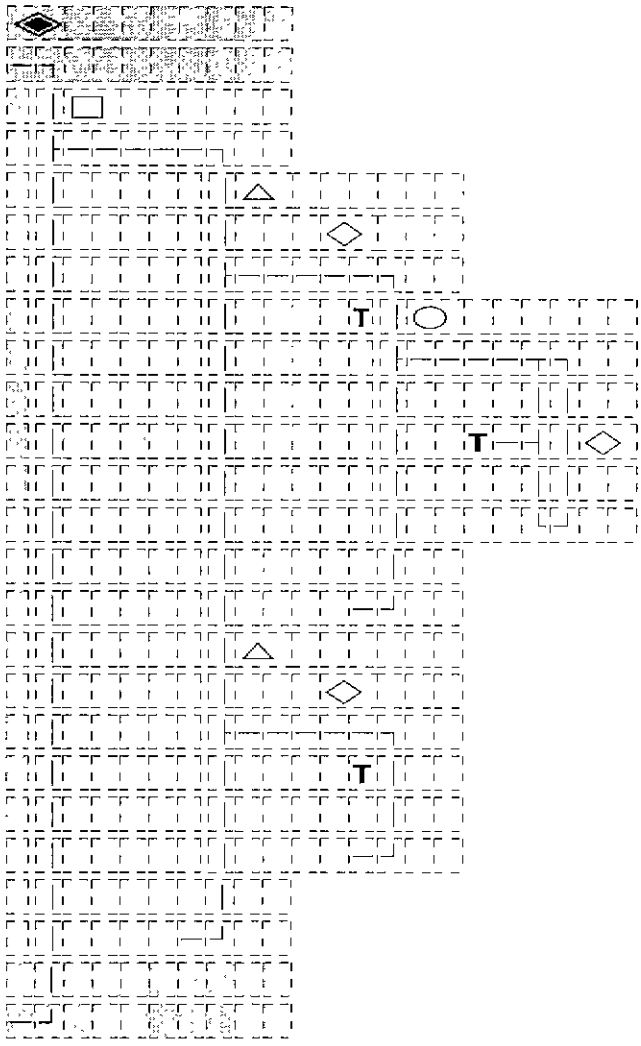
(그림6)은 (그림5)의 구조패턴에다 C2_2의 한가림구조부품(부품번호: C2_2)을 추가로 조립해 넣은 예를 표현한 것이다. 이러한 식으로 조립하여 완성된 소프트웨어 설계모듈 패턴부품의 뼈대는 (C1_2 -> C2_1 -> ((C2_2 -> C2_6) · C2_2)) 와 같이 추상화된다.

조립공식 중에서 '·'표시는 연접(concatenation)을 의미하며, '->'는 내포를 의미한다.

소프트웨어 설계부품을 이와 같은 조립공식에 의거하여 계속 조립해 나감으로써 소프트웨어 설계 패턴을 공장자동화된 방법으로 구현할 수 있게 되는 것이다.



(그림 5) 사이끝되풀이 구조 부품의 추가 조립 예



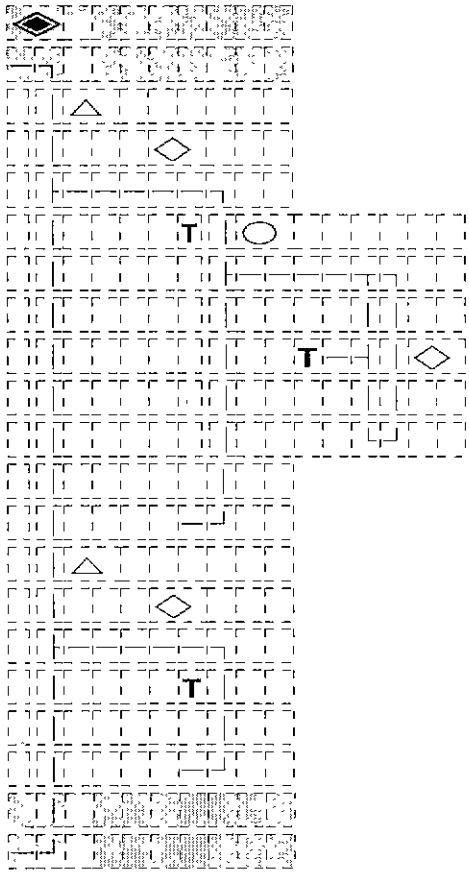
(그림 6) 한가립구조 부품의 추가 조립 예

6. 소프트웨어 유지보수에의 적용 실험 사례

소프트웨어 패턴 설계부품의 규격화·표준화를 통한 일련의 가공(加工)작업의 자동화가 소프트웨어의 유지보수 능력에 미치는 효과는 발음 그대로 가공(可恐)할 정도이다.

소프트웨어 조립·분해의 자동화를 도모하기 위해 제5세대 설계부품인 SOC(Structured Object Component)을 이용하도록 만들어진 소프트웨어인 SETL(Structured Efficiency TooL)을 사용하여 유지보수를 할 경우와 인간이 프로그램 편집기 등을 이용하여 기존 소스 프로그램을 유지보수할 경우의 효율성을 비교하기 위한 실험이 1996년 11월부터 12월까지 2개월 동안 4회 행해졌다.

(그림8)은 실험에 사용된 소프트웨어 조립·분해 자동화 도구인 SETL의 모습이다.

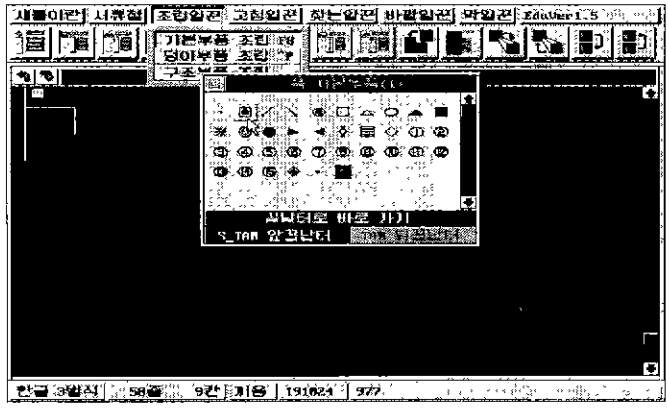


(그림 7) 한잇따름구조부품의 분해 · 제거 예

주어진 문제는 비구조적으로 작성된 400라인 이내의 스파게티 C언어 소스프로그램을 유지보수하여 구조화시키는 것으로 제한시간은 1시간(3600초)이었다. 실험방법은 자동화도구인 새틀 1개를 상대로 C언어 과정을 이수한 진문대학의 전자계산과학생 그룹 또는 전산실무에서 종사하는 각 정부부처의 전산담당 공무원 다수가 합세하여 문제를 푸는 형식으로 진행되었다. 각 그룹 내에는 상당한 C언어 프로그래밍 실무 이력을 가진 사람들이 포함되어 있었다. 실험의 공정성을 도모하기 위해, 주어진 문제를 풀기전에 실험현장의 어느누구나 주어진 C언어 소스프로그램 문제에 대해 원하는 구조나 명령을 삽입 또는 변경할 수 있는 기회를 준 뒤에 실험이 진행되었다. 주요 실험결과는 <표 1>과 같다.

7. 결론 및 향후 과제

위와 같은 주요 실험이외에도 1996년 한 해동안 크고 작은 실험들을 통해 소프트웨어를 수작업으로 유지보수할 경우와 자동화도구를 사용하여 유지보수할 경우의 생산성의 차이는 비교할 수 없을 정도로 큰 차이가 난다는 사실을 확인하였다.



(그림 8) 소프트웨어 자동화 도구 SETL 모습

실험일자 (1996년도)	실 험 장 소	실 험 참 가 인	실 험 결 과	
			인 간 측	도구측(SETL)
11월 30일	대림전문대학	대림전문대학 C언어과 정 이수자 전자계산과 1학년 주간 80명	1시간(해결못함)	1초(해결함)
12월 4일	대림전문대학	대림전문대학 C언어과 정 이수자 전자계산과 1학년 야간 80명	1시간(해결못함)	2초(해결함)
12월 6일	서울시 전자계산소	서울시 전자계산소 전산담당공무원 28명	1시간(해결못함)	1초(해결함)
12월 13일	총무처 정부전자계산소 전자계산교육원	정부 각부처 5급~9급 전산담당공무원 30명	1시간(해결못함)	1초(해결함)

<표 1> 자동화도구 대 인간의 유지보수 능력 실험결과

인간의 특징은 도구를 상대로 대항하는 것이 아니라, 도구를 잘 활용할 줄 안다는 데에 있다. 이제 우리는 소프트웨어 조립·분해 자동화도구를 어느 시점에서 받아들일 것인가를 결정해야 하는 단계에 와 있다.

앞으로 소프트웨어 조립·분해의 자동화 기술의 소프트웨어의 개발 및 유지보수 실무에의 파급효과는 그 크기를 가늠하기 어려울 정도로 클 것이며, 소프트웨어의 개발 및 유지보수에 있어서 지금처럼 코드중심의 환경에서 벗어나 설계중심으로 패러다임전이가 곧 일어나게 될 것이다.

그렇게 되면 어느 정도의 기간동안 기존의 코딩 환경을 고수하여 신기술을 거부하는 측과 새로운 설계 조립·분해 자동화 기술을 적극적으로 도입하여 생산성을 올리는 쪽과의 물결충돌이 예상된다. 향후 이러한 갈등을 어떻게 조정하여 충격을 극소화하면서 설계자동화의 단계로 진입할 것인가에 대한 보다 심층적인 연구가 진행된다면, 조직의 업무리엔지니어링과 소프트웨어 프로세스 리엔지니어링이 자연스럽게 접목되어 회사의 체질을 강화시켜 전략 및 전술의 양면적 측면에서 대외경쟁력을 높힐 수 있게 됨은 물론, 우리의 소프트웨어 개발 및 유지보수를 자동화하기 위한 조립·분해 기술 환경은 보다 성숙된 단계로 진입할 수 있게 될 것이다.

참고문헌

[1] 平井利明·山崎弘, フロ-チャ-ト 徹底研究, HBJ PUBLISHING, 1989.

[2] CARMA McCLURE, THE THREE Rs OF SOFTWARE AUTOMATION, Prentice Hall, 1992

[3] M.A. JACKSON, 構造的プログラム設計の原理 (PRINCIPLES OF PROGRAM DESIGN), 日本コンピュ-タ協會, 1980.

[4] James Martin, RECOMMENDED DIAGRAMMING STANDARDS for ANALYSTS and PROGRAMMERS - A BASIS for AUTOMATION, 近代科學社, 1991.

[5] 竹內元一 外, 圖解する技術, 中經出版, 1994.

[6] 山下伸之・高橋守満, 流れ圖の見方・書き方, オ-ム社, 1990.

[7] 鯉沼 章, フロ-チャ-トによる事務分析, 日刊工業新聞社, 1991

[8] 酒井博敬 外, オブジェクト指向設計, オ-ム社, 1993.

[9] 江村潤朗・野津昭, プログラム流れ圖の作成技法, オ-ム社, 1989.

[10] 馬場 勇, ソフトウェア開發實踐技法, 技術評論社, 1989.

[11] 大塚純一 , 仕事を圖解する技術 フロ-チャ-ト入門, 日本能率協會, 1990.

[12] James Martin & Carma McClure, Diagramming Techniques for Analysts and Programmers, 近代科學社, 1991.

[13] 上野一郎, 能率の考え方, 産能大學, 1990.

[14] 竹下 亨, ソフトウェアの保守・再開發と再利用, 共立出版株式會社, 1992.

[15] 佐藤允一, 問題構造學入門, ダイヤモンド社, 1989.

[16] 原田賢一, 構造エディタ, 共立出版株式會社, 1988

[17] Washida Koyata, 思考の技術發想のヒント, 日本實業出版社, 1996.

[18] ジャストシステム, オブジェクト モデリング, 落水浩一郎 著, 1995.

[19] 木下 しゅん, ソフトウェアの法則, 中公新書, 1995.

[20] 土居範久, オブジェクト指向の おはなし, 日本規格協會, 1995.

[21] 유홍준, 한국형 순서도 쎄크, 도서출판 홍은, 1992.

[22] 유홍준, 인간지향 무른모 설계 방법론, 도서출판 홍은, 1996.

유 홍 준

1992년 일본 산능단대 능률학과 졸업(정보처리 전공)
1983년 1급통신사(무선통신 분야)
1994년 정보관리 기술사(정보처리 분야)
1994년 기수지도사(정보처리 분야)

1977년-1981년 한국특수선(주) 통신장
1981년-1985년 범양상선(주) 통신장
1988년 제5회 전국PC경진대회 공모부문 응용소프트웨어 분야 1위(상공부장관상 수상)
1985년-현재 홍준 정보처리 학원 원장
1989년-현재 도서출판 홍은 대표
1996년-현재 힘스미디어 대표
관심분야 : 소프트웨어 공학(특히, 순공학, 역공학, 재구조화, 재공학, 소프트웨어 공장자동화, 소프트트웨어 프로세스 리엔지니어링)