《特 輯》

소프트웨어의 자동생산

이 경 환
(중앙대 전산학과 교수)

■ 차 례 ■

[1] 소프트웨어의 자동생산의 의미

제품의 대량생산을 하는 다른 산업에서처럼 소프트웨어의 대량생산을 위해 1968년도에 R.W.Bemer 는 도구(tool)의 표준화와 생산, 품질의 엄격한 관리를 들었고, M.D.Mcklroy는 NATO과학 학회에서 새로운 프로그램을 만들때 코드의 체계적인 재사용을 역설하였다.

이러한 주장들은 소프트웨어 산업계에서 관심을 가져 1969년 Hitachi를 필두로 NEC, Toshiba, Fujitsu, NTT, Mitsubishi 등 일본 소프트웨어 업계와 미국의 System Development Corp. (Unisys의 전신) 등에서 추진해 오고 있다.

초기의 기술단계에서 점차 체계화되어 자동생산공장(S/W factory)으로 발전하게 되었는데, 기술적인 면에서 이러한 자동 생산공장들의 방향은 (1) 도구의 표준화, 생산과 품질의 엄격한 관리와 사용자 인터페이스의 체계화 등을 통한 소프트웨어 개발환경의 구축과, (2) 코딩(혹은 설계) 수준에서 기존에 만든 모듈을 재사용 함으로써 생산성의 향상과 품질의 향상을 시키려는 노력으로 나아가고 있다. 재사용 기법을 근간으로 자동 생산 공장 건설을 추진하는 회사들 중 대표적인 곳이 일본의 Toshiba다.

실시간 소프트웨어 제품들을 주로 만드는 Toshiba 는 그림 1에서 보여주는 것처럼 모든 프로젝트가 표준화된 생명주기 모형을 따르며, 기준선(baseline)을 두어 중요한 관리적계획의 기준점으로 삼고 있다.

생산 공장내에서 재사용되는 대상은 모듈화된 문서(modularized document 혹은 specification)나 프로그램이며, 재사용 부품(reusable parts)을 만드는 부서와, 부품 센터 그리고 소프트웨어 부품의 관리위원회와의 관계를 보여주는 것이 (그림 2)이다.

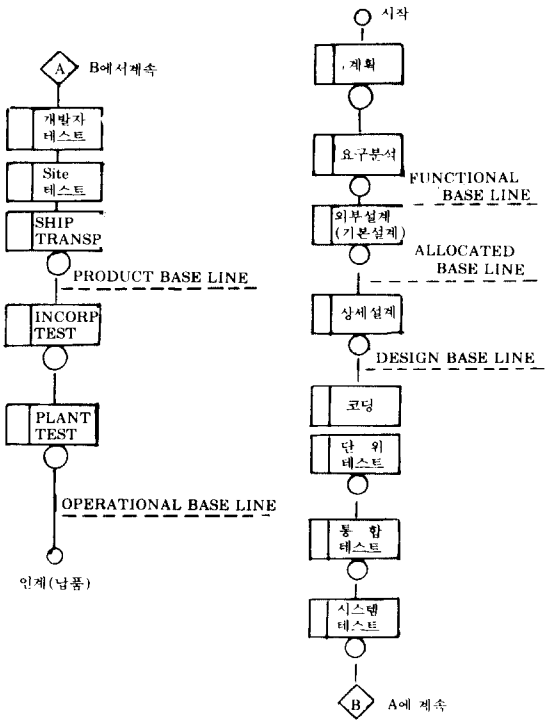


그림 1. 자동생산 공정

○ 부품관리위원회

재사용 부품을 모으고, 선택하여 어떤 부품들을 새로 만들며, 또한 수정하며 제거할지를 결정한다.

○ 부품의 제조부서

부품을 디자인하고 구현하여 부품으로써 일정한 품질을 만족하는지를 체크하여 재사용 소프트웨어 데이타 베이스에 등록한다.

○ 부품센터

재사용 소프트웨어 Data Base의 부품 등록 관리, 프로젝트를 위한 디자이너와 프로그램을 위한 인터페이스 기능(검색과 인도)을 제공한다.

소프트웨어 데이타 베이스에 등록된 부품들의 49%가 1-10K EASL(Equivalent Assembler Source Lines), 35%가 10-100K EASL인데 반해 재사용된 부품들 중 1-10K EASL이 55%, 10-100K EASL이 36%를 기록하고 있다. 또한 재사용 부품의 추출방식은 현재 단순한 키워드를 바탕으로 하고 있지만 앞으로 자체 고안한 OKBL (Object-Oriented Knowledge based

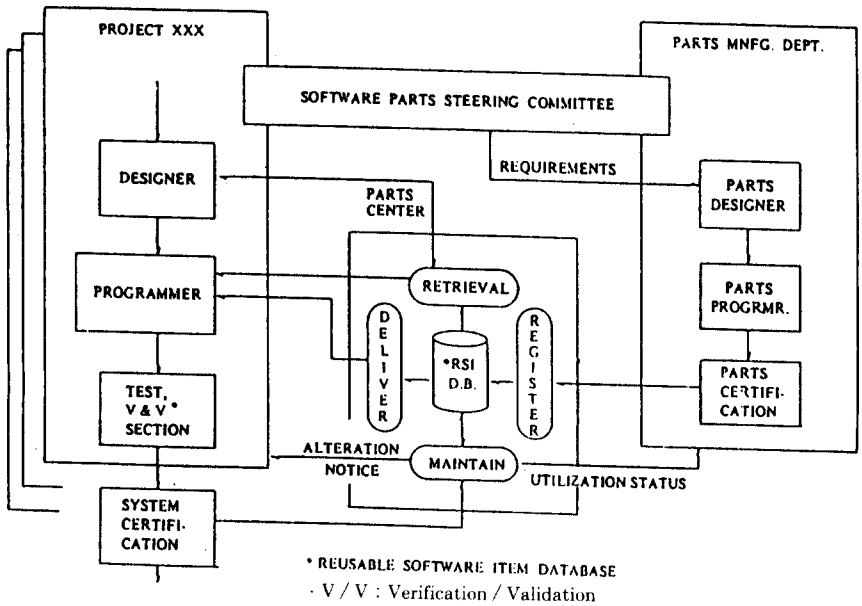


그림 2. 재사용 촉진 위원회의 조직

language)로 구현하는 객체지향적 시스템을 형성하려 하고 있다.

위에서 설명한 재사용 환경과 잘 디자인된 워크 스페이스(work space), 적절한 도구, 비용관리 시스템, 생산성 관리 시스템, 교육 프로그램, 표준화된 기술적 방법론 등과 혼합한 환경을 구축한 Toshiba의 자동생산 공장은 매년 14%의 생산성을 향상하게 되었다.

2. APSE에 의한 자동화

프로젝트 Stoneman은 APSE(Ada Programming Support Environment)을 구축하여 소프트웨어 개발의 생산성을 향상시키고자 추진된 프로젝트이다. APSE는 커널 APSE로서 KAPSE, 최소환경인 Minimal APSE, 그리고 전체적인 APSE의 세가지 레벨로 이루어진다.

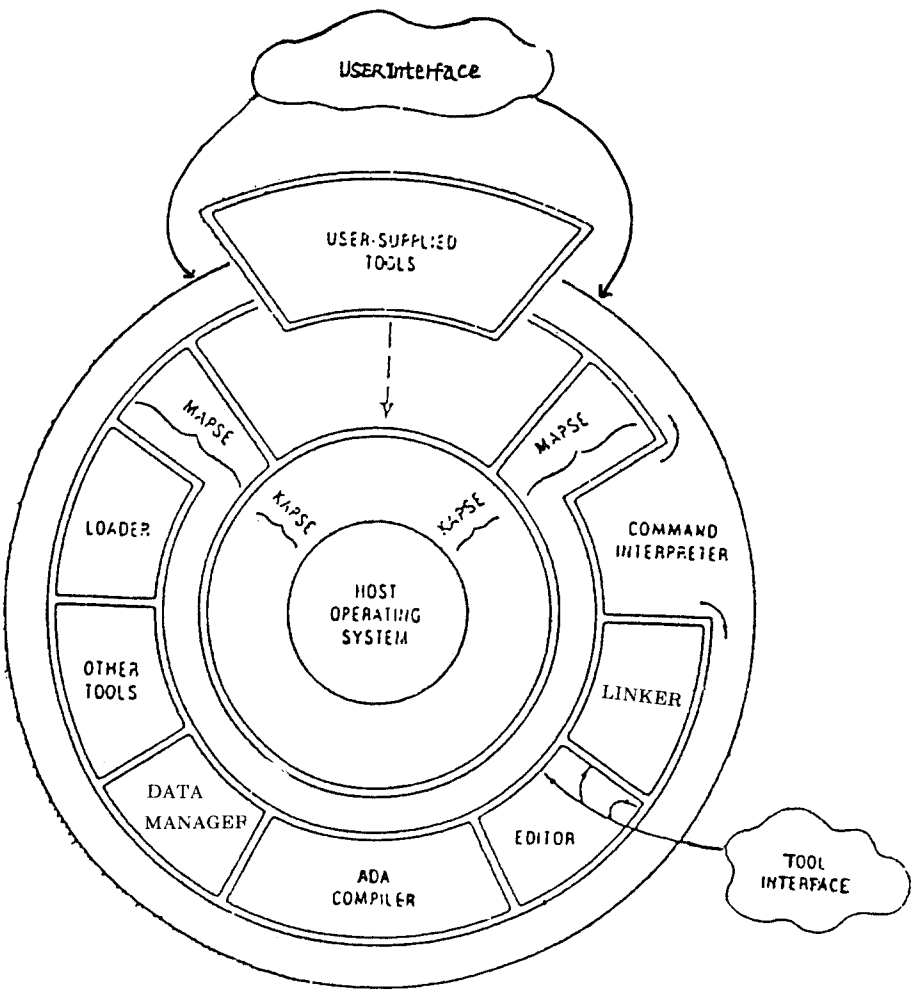


그림 3. APSE의 구성

KAPSE는 OS와 연결하여 Ada 프로그램을 실행하고 호스트 OS하에서 단순하고 간단한 다음과 같은 서비스기능을 수행한다.

- DB기능(질의 및 조작)
- Communication 기능
- 실행시간 지원
- TOOL Composition
- Debugging Facility 제공
- DB질의 및 조작을 위한 low-level 서비스
- Ada package I/O를 위한 low-level 서비스

MAPSE는 Ada 프로그램의 개발과 계속적인 지원을 위한 툴을 제공하고, 모든 프로젝트 팀들이 MAPSE 환경하에서 일할 수 있도록 지원하며, 생명주기 단계별로 프로젝트 활동을 지원한다. 이러한 MASPE의 툴 셋트는 다음과 같은 서브 시스템들을 포함한다.

- COMMAND LANG INTERPRETER
- LINKER
- EDITOR
- ADA COMPILER
- DATA MANAGER
- LOADER
- CONFIGURATION MANAGEMENT TOOLS
- ASSEMBLER
- STUBGENERATOR
- FILE ADMINISTRATION TOOLS
- TIMING ANALYZER
- FREQUENCY ANALYZER
- SYMBOLIC DYNAMIC DEBUGGER
- PRETTYPRINTER

전체적인 APSE는 데이타의 완전성 유지와 형상관리(configuration management)조절, 그리고 응용 라이브러리를 지원하는 기능을 지니며, 그 기능은 다음과 같다.

- TOOL의 추가와 합병
- TOOL의 추가가 없으면 APSE는 MASPE 자신이 된다.
- MAPSE는 일반적인 툴을 제공하고, APSE는 새롭고 특수한 툴을 제공한다.
- 새로운 툴을 제공하고 구식의 툴은 제거한다.

[3] PCTE 프로젝트에 의한 자동화

- PCTE(Portable Common Tool Environment)는 소프트웨어 개발을 위한 관리, 행정, 기술정보, 시스템향상 등을 지원하고 여러가지 정보를 소프트웨어 상품의 품질 및 생산성 향상을 위해서 통합적인 툴을 사용할 수 있도록 지원한다.
- CEC(Commision of the European Community) 가 ESPRIT 프로젝트하에서 추진하고 있는 PCTE의 특성은 다음과 같다.
  (1) 일종의 OS
  (2) 일종의 DBMS
  (3) 분산처리 기능 : 데이타 툴을 네트워크를 통해서 W/S끼리 연결시켜 사용자는 단일 기계를 사용한 것처럼 활용
  (4) PCTE의 사용자 인터페이스는 그래픽에 의한 Toolkit 형식

이상과 같은 특성을 가지고 추진되는 PCTE프로젝트는 다음과 같은 장점을 갖는다.

- 서로다른 vender가 지원하는 툴을 통합사용
- 모든 툴을 자동화시켜서 사용하여 생산성 향상
- 새로운 툴과 방법론의 개발과 추가
- 산업적인 표준화 추구

[4]. 객체지향 방법에 의한 자동화

4.1 재사용과 객체지향 방법

70년대 가장 많이 쓰고 있는 Top-down기능적 방법(Top-down Functional Approach)에 대응하는 소프트웨어 개발방법으로 80년대 붐을 이루고 있는 것이 객체 지향적 소프트웨어 개발방법(Object Oriented Design/programming)이다.

소프트웨어 개발과정에서 디자인과 코딩부분을 지원하는 "부분적 생명주기 소프트웨어 개발방법"으로서 객체 지향적 개발은 객체(object)의 개념을 중심으로 시스템을 구성해 나가는 방법이다.

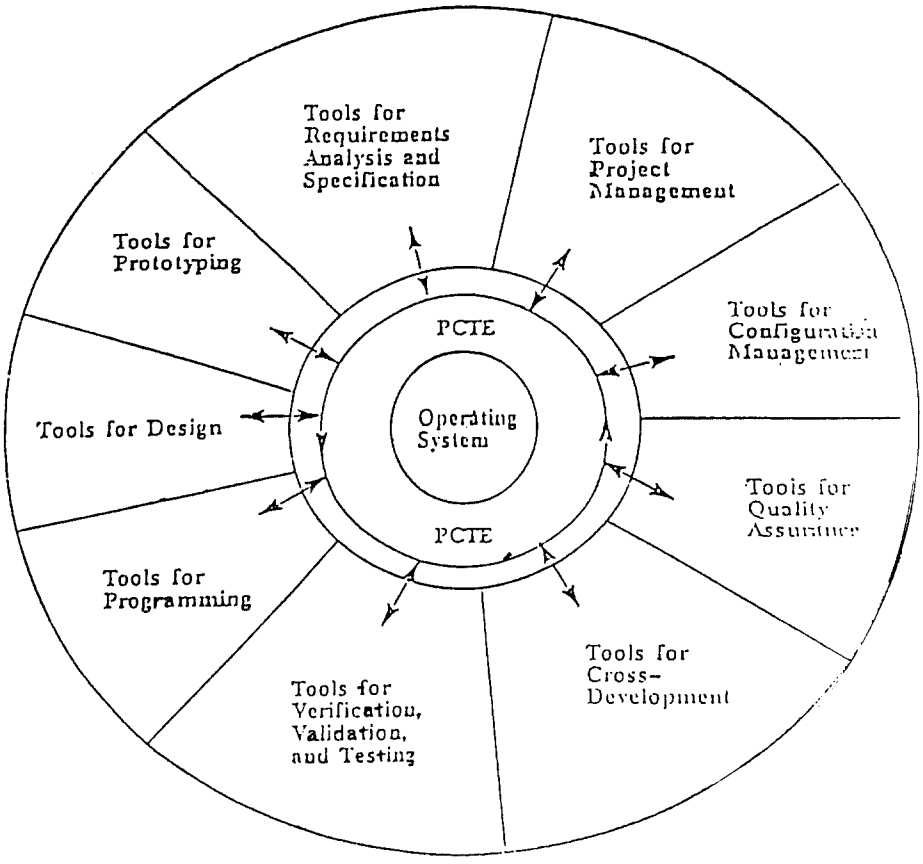


그림 4. PCTE의 구성

이러한 객체 지향적 개발이 재사용과 연관하여 중요한 의미를 갖는 것은 재사용 모듈(reusable modules 또는 reusable components)들을 구축하여 모듈 뱅크(module bank) 혹은 소프트웨어 Base(S/W Base)를 형성하는데 기존의 Top-down 기능적 방법보다 용이하다는 것에 있다.

객체 지향적 개발방식이 재사용 모듈을 형성하는데 얼마나 용이한가를 자세히 살펴보기 전에 대략적으로 "Bertrand Meyer"가 보여준, 재사용 컴포넌트(reusable-component)가 갖추어야 할 모듈구조(module structures)의 특성을 살펴보자.

(1) 여러 타입에 대응하는 모듈구조(variation in types)

모듈의 파라미터(passing parameter)에 사용요구에 따라서 여러가지 타입으로 변환될 수 있는 구조로서, 그 예로는 Parameterized Programming 이 있다.

(2) 여러 데이타 구조와 알고리즘에 대응하는 모듈구조(Variation in data structures and algorithms)

검색(search)모듈 구조를 생각할 때 유일한 하나의 데이타 구조와 알고리즘 뿐만아니라 여러 데이타 구조와 그와 연관된 검색 알고리듬, 즉, Sequential tables, arrays, binary search trees, B-trees 등과 같은 모듈 구조.

(3) 관련이 있는 루틴들을 한 모듈로한 구조 (Not just one operation but several related operations)

하나의 테이블을 검색하는 방법을 알기 위해서

는 데이블이 어떻게 만들어지며, 인자들이 어떻게 삽입되고 제거되는지를 알아야 한다.

그래서 검색루틴 자체 만으로는 충분치 않고 그 밖의 연관된 루틴들(테이블의 생성, 인자의 삽입, 제거 등)에 관한 모듈을 형성할 수록 좋다.

(4) Run-time시 변화하는 변수 형태에 따라 자동적으로 모듈의 알고리듬이 선택되는 구조(Requesting an operation with out knowing its implementation)로서, 다음과 같은 보기에서 볼 수 있다.

Present : =search(x, t) : 테이블 t에서 x를 검색한다.

Run-time시 t의 형태가 변화하면, 시스템의 어디선가는 여러 가능한 검색 알고리즘들 중에서 한 알고리즘을 선택하는 결정을 요한다. 이러한 결정을 모듈이나 모듈의 사용자가 수행한다. 이러한 결정을 하려면 한 모듈이 모든 가능한 테이블들의 구현에 관한 정보를 갖고 있어야 함으로 이는 모듈들을 관리하는데 많은 어려움을 야기시킨다. 또한 사용자가 결정한다는 관점은 더욱 나쁜것이 사용자가 t의 형태가 어떻게 변할지 모르기 때문이다. 이러한 결정을 시스템 내부에서 자동적으로 할 수 있는 동적바인딩 매키니즘을 가지면 좋다.

(5) 유사 모듈사이의 공통모듈(Commonality with subgoup)

모듈 뱅크를 형성하는데 있어 유사한 모듈들 사이의 공통인 것을 한 모듈로 형성한다. 이는 정형화된 모듈들의 집합을 형성하기 위한 것으로 재사용 모듈의 설계에 관련된 사항이다. 공통모듈을 형성한 후 유사한 모듈들은 공통모듈을 수정하여 모듈을 형성한다.

Top-down, 함수적방식(Functional approach)에 의해 구성되는 루틴(함수, 프로시져, 서브루틴, 서브 프로그램의 통칭)은 과학적 계산분야의 문제를 해결하는데 아주 성공적이었다.

그러나 앞에서 제시한 다섯가지의 재사용 모듈 구성 요건으로 볼때 많은 제약성을 갖고 있지만 그중에서도 가장 문제가 되는 것은 (3)으로써 검색루틴이 검색방법을 알기 위해 필요한 테이블 생성, 인자의 삽입, 제거에 관한 아무런 정보를 갖지 못해, 다른 루틴들이 추가식으로 필요하다.

이런 배경하에 객체 지향적 방식에서 제공하는 Package, Overloading과 Genericity가 재사용 모듈을 구성하는데 어떻게 효과적으로 연관되는지를 살펴보자.

○ Package

Ada와 Modular-2와 같은 Modular · 언어들이 제공하는 Package는 Package를 사용하는 프로그램과 연결을 위한 인터페이스 부분인 명세와 인터페이스에서 선언한 함수들의 구현을 시킨 몸체로 구성된다. 한 객체와 연관된 함수들을 한곳에 모아 둠으로써 검색 함수가 검색방법에 관한 정보를 충분히 갖게 된다. Package를 사용함으로써 함수적방식에서 루틴들이 갖고 있는 (3)의 한계점을 극복한다.

○ Overloading

여러 객체 지향적 언어에서 제공해주는 Overloading 은 프로그램 속에 나오는 이름에 한가지 의미 이상을 붙여주는 것으로 전형적인 예로써는 함수의 이름(Names of operations)을 들 수 있다. Overloading은 한 모듈이 여러 데이타 구조에 대응하고, 어떤 알고리즘을 사용할 것인가를 스스로 결정해 준다. 이런 점에서 Overloading은 (2)번과 (4)의 문제점을 해결할 수 있는 능력을 준다.

○ Genericity

Parameterized모듈을 정의할 수 있는 능력을 주는 Genericity는 직접 사용할 수 있는 모듈은 아니지만 핵심모듈(Generic module)을 구성할 수 있게 한다. Genericity는 (1)번의 문제를 해결해 준다.

Package, Overloading과 Genericity가 해결해

주지못한 (5)번의 문제는 모듈들을 구성하는 디자인 상의 문제로 객체 지향적 방식에서는 상속(Inheritance) 메커니즘들을 사용함으로써 해결을 하고 있다.

위에서 살펴본 객체 지향적 개발방식의 특성을 이용하면 기존의 기능 지향적 개발방식에 비해 코드수준(혹은 디자인수준)의 재사용 모듈을 효과적으로 구성할 수 있다. 객체지향 방법은 설계와 프로그래밍으로 나누어 진다. 객체지향 설계는 능동적인 재사용 기법으로서

· Information hiding을 위한 설계
· Module의 encapsulation

등의 기법이 채용되고, 객체지향 프로그래밍에서는 수동적인 재사용 기법으로서 콤포넌트에 추상 자료형(Abstract Data Type)의 표현에 의해서 소프트웨어 부품을 만들고, 이들 부품에 승계(Inheritance)기능을 부여하여 새로운 부품을 생성(Generic)하여 부품의 갯수를 확장시켜 간다.

객체지향 프로그래밍 언어로는 현재 다음과 같은 툴들이 사용되고 있다.

· C++
· objective C
· objective Pascal
· Flavour
· Actors
· Eiffel
· Smalltalk 80

4.2 부품화의 방법

1) 부품의 정의

소프트웨어 개발의 생산성 및 소프트웨어의 품질을 비약적으로 향상시켜야 하는 과제에 대한 하나의 해결책이 소프트웨어 부품의 재사용에 있다. 즉 소프트웨어를 '재사용 가능한 정보(코드 수준에서)'로서 취급될 수 있도록 작성, 축적해두고 이후의 소프트웨어 개발에 있어서는 새로운 소프트웨어를 작성하기 보다는 적극적으로 그것을 재사용한다면 빠른 시간내에 쉽게 소프트웨어를 작성할 수 있다.

이와같은 '재사용 가능한 정보(코드 수준에서)'를 소프트웨어의 부품이라고 하며, 이것을 작성, 이용하기 위한 기술을 '소프트웨어의 부품화기술'이라 부른다.

2) 부품의 특성

재사용 가능한 부품으로서 라이브러리와 같은 등록, 검색 기능등을 갖춘 저장매체에 저장된 부품은 그 조직의 중요한 자산(Asset)이다. 이러한 부품은 초보자에서 숙련된 프로그래머에 이르기까지 매우 다양한 사람들에 의하여 사용이 되며 재사용되는 영역도 매우 넓어지게 된다. 그러므로 부품은 소프트웨어 공학이 추구하는 여러 특성들을 갖춘 양질의 부품이어야 한다. 일반적으로 재사용 가능한 부품이 가져야될 특성들을 나열해 보면 다음과 같다.

가. 부품은 환경에 간섭을 받지 아니한다.
나. 부품은 객체 지향적 방식으로 설계되고, 또 그 데이타 상에서 연산되는 프로시듀어 및 함수(function)들과 함께 형태 지워진 데이터(typed data)로써 패키징된다.
다. 부품은 building단계동안 사용된 발판으로 작용한다.
라. 부품은 높은 응집도(cohesion)와 낮은 결합도(coupling)의 특성을 가져야 한다.
마. 부품은 개발자가 아닌 다른 사람에게도 읽기 쉽게 작성되어져야 한다.
바. 부품은 일반성(generality)과 특수성(specificity) 사이의 조화가 잘 이루어지게 작성되어야 한다.
사. 부품은 찾기 쉽도록(findable)하게 하기 위해 충분한 문서(documentation)를 동반한다.
아. 부품은 변경없이 사용되거나 혹은 최소한의 수정만을 거쳐서 사용될 수 있어야 한다.
자. 부품은 그의 환경이나 환경에 의존적인 사항으로부터 독립적이며, 사용되지 않는 정보 형태나 내용으로부터 분리되어야 한다.

차. 부품에 대한 호출, 제어 및 종료 등이 표준화 되어야 한다.

카. 부품은 해당영역에 잘 응용되어야 하며, 올바른 추상화와 모듈화(modularity)로 작성되어야 한다.

재사용 가능한 부품이 이와같은 특성들을 많이 가질수록 양질의 부품이라 할 수 있다.

부품들을 라이브러리와 같은 저장매체에 모아 놓고 사용자가 원하는 부품을 찾아 재사용할 수 있도록 해주는 재사용 시스템 역시 초보자에게 숙련된 사람에 이르기까지 매우 다양한 사람들이 사용하게 되며 사용되는 영역도 매우 넓다. 일반적으로 재사용 시스템이 갖추어야 할 특성들을 살펴보면 다음과 같다. 이러한 특성들을 기본 특성이라 부른다.

- 재사용될 가능성이 있는 부품을 찾아 내기가 용이해야 한다.(findable)
- 한번 찾아내어진 부품은 재사용 하기에 충분할 만큼 이해가 되어야 한다.(understandable)
- 찾아내고 이해된 후, 그 부품을 재사용하기에 적정한 타당성이 있어야 한다. 즉 재사용되기 위해 적정 제한 조건으로 건조(Build)되어야 하고, 재사용 되는데 꼭 맞아야(fit)한다.

부품을 작성하는 경우와 부품을 기반으로한 재사용 시스템을 구축할 때 위와같은 특성들을 갖추도록 노력해야 할 것이다. 물론 위에서 열거한 특성들에 새로운 특성들이 더 추가되어야 하며, 앞으로 부품 특성에 대한 기준이나 재사용 시스템이 갖추어야할 기준들을 더 연구하여야 할 것이다.

부품이 갖추어야할 특성과 재사용 시스템이 갖추어야할 특성들간에는 서로 상관관계가 있다. 이들의 관계가 〈표 2〉에 나타나 있다.

〈표 2〉는 부품의 각 특성과 이들 특성으로 증진되는 기본 특성을 관련지워 표현한 것이다. 예를 들어 〈다〉에서와 같이 부품이 builing 단계 동안 사용된 발판으로 작용되어야 한다는 특성을 만족한다면 기본 특성중 built를 직접 지원할 것이며, 〈사〉에서와 같이 충분한 문서화가 된 부품이라면 부품을 찾기 쉬워야 한다는 특성을 잘 만족하며 이런 문서로 이해도가 증진되며 건조시의 효율도 증진하게 된다.

표 2. 부품 성격과 재사용에 필수적인 특성들과의 관계

성격	재사용에 필수적인 기본 특성들				
	Findable	Understanable	Built	Fit	Conceptually sound
가		○		•	
나	○	○	○		•
다			•		
라		○		•	
마		•		○	
바		○	•		•
사	•	○	○		
아			○	•	
자		○	○	•	
차			○	•	
카		○			•

(key : • = 긴밀한 관계, ○ = 부수적인 관계)

3) 부품화 기술

가. 부품의 작성

소프트웨어 부품을 만드는 방법에는 '신규로 부품을 작성'하는 경우와 '기존에 존재하는 소프트웨어 자산을 부품화'하는 경우의 2가지로 나눌 수 있으며, 2가지 경우에 대한 일반적인 사항만을 아래에서 기술한다.

(1) 신규작성

본래 부품을 신규로 작성하는 경우에는 가능한 한 규칙, 기준에 따라서 개발해야 한다. 그러나 현재에는 규격과 기준을 모색하는 상태에 있다. 여기서는 신규로 부품을 작성하는 경우 준수해야할 점을 몇가지 소개한다.

(가) 부품 재사용을 위한 코드 설계방식

- 요구 사항을 보다 적은 부분으로 분해
- 블럭 작성
- 새로운 코드를 작성하는 경우에도,

명확한 인터페이스를 명시
- 부품은 단 하나의 기능만을 수행하도록 한정한다.

(나) 부품의 변화에 대한 독립성
부품의 변경시 사용자에 영향을 주지않는 인터페이스를 확립해야 하며 시스템의 제약을 받지않고 용이하게 부품을 조합할 수 있어야 한다. 또한, 다음중 어느 하나를 변경시키더라도 다른 두개에 영향을 주지 않도록 부품을 작성해야 한다.
- 부품의 내부구조(데이타 구조와 처리 수 순)
- 부품의 인터페이스
- 부품을 호출하는 측의 처리내용

(다) 부품간 인터페이스
UNIX의 파이프(PiPe)와 같이, 메세지의 일로서 정보전달을 하는 방법이 넓은 범위의 응용에 대해 쉽게 사용되는 것이다.

(2) 기존의 소프트웨어를 부품화
이미 개발되어져 있는 소프트웨어 중에서 부품을 만드는 경우는 크게 2가지로 나누어 볼 수 있다.

첫째, 동질(homogeneous)환경이다. 이 경우는 기존에 작성되어 있는 소프트웨어중 부품으로 적합한 부품만을 골라내는 방식이다.

둘째, 이질(heterogeneous)환경이다. 이 경우는 기존의 소프트웨어중 부품에 적합한 부품만을 골라내었다고 해서 그대로 재사용할 수 없다. 환경(물론, 여기서 말하는 환경은 사용하고 있는 언어 차원의 환경이다)이 다르기 때문에 골라낸 부품을 사용하고자 하는 환경으로 변환시켜 주어야 한다.

따라서 기존에 개발되어진 많은 소프트웨어를 부품화 하는데 있어서 드는 노력은 어떠한 부분이 부품으로서 적합한지를 찾아내는 노력과 환경이 다른 경우 변환기(transformer) 개발의 노력이 들게된다.

나. 부품의 관리

부품을 관리하기 위해서는 먼저 이들을 저장하기 위한 부품 데이타 베이스가 있다. 그리고 최적한 부품의 데이타 베이스를 구축하기 위한 기술이 필요하다. 부품의 관리에 필요한 기능들을 나열해 보면 다음과 같다.
- 부품 등록
- 부품 삭제
- 부품 평가
- 부품 분류
- 부품 검색
- 부품 변경, 교환

이들 각 항목에 대한 자세한 사항과 관리기능의 구현방법은 참고문헌을 참고하길 바란다. 현재 개발 자원의 시스템이 갖고있는 부품관리 기능은 부품을 단순한 하나의 데이타로 본 데이타 베이스 관리 기능 그 자체에 있고, 아래에 나타난 것과 같은 특유의 기능에 대해서는 거의 갖고 있지 않다. 아래의 기능을 갖출수록 보다 나은 부품관리를 행할 수 있게 된다.
- 유사 부품의 취급
등록, 검색(대강 검색, 최적 부품의 검색), 부품으로의 비젼 관리 등.
- 부품 변경시의 다른 부품으로의 파급효과 파악 등.

4.3 재사용 시스템 CARS

CARS(Computer Aided Reuse System)을 중앙대학교 연구개발중인 재사용시스템으로 그 시스템의 구성은 〈그림 5〉와 같다. CARS는 앞에서 지적한 RSL의 단점을 보완해서 연구한 재사용 시스템이다.

MS / DOS 및 UNIX / X-Window / Motif 환경하에서 개발중인 CARS는 MS / DOS용으로는 C로 작성된 클래스를 라이브러리에 저장하고 4가지의 Facet(object, function, domain, language) 에 의해서 검색할 수 있도록 설계하였다. User Interface중에서 Finding 서브시스템과 Building 서브시스템을 지원하고 Knowledge Base를 별도로 구축하여 지원하도록 하고있다.

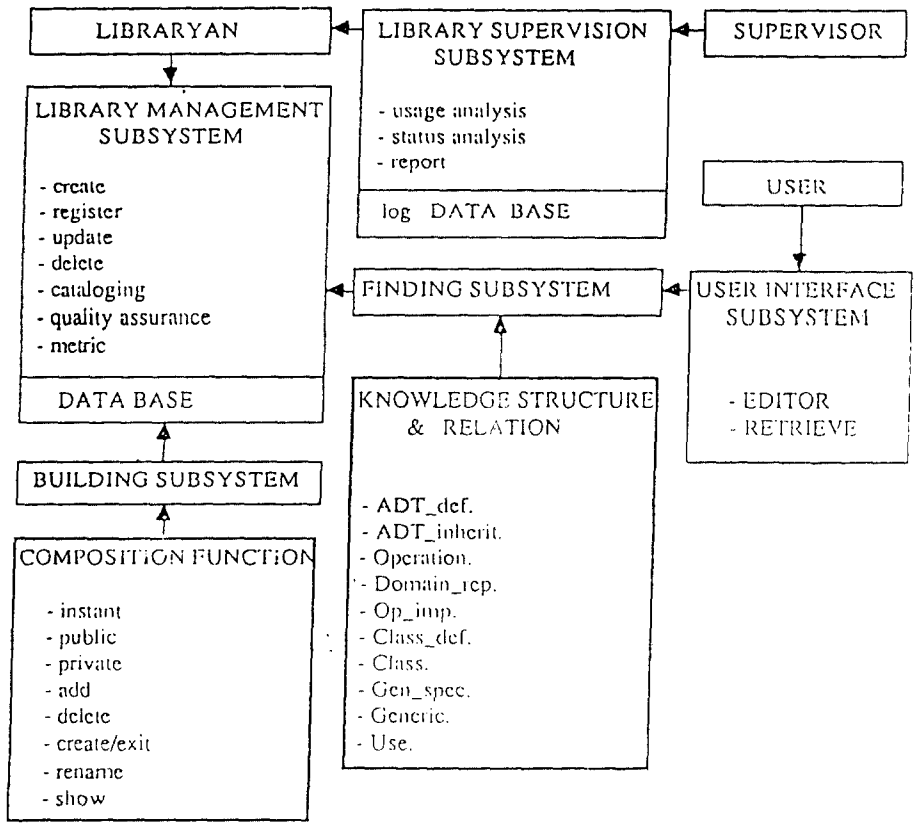


그림 5. CARS의 구성

현재, 자료구조의 기본 알고리즘과 그래픽 primitive들은 클래스를 작성하여 라이브러리를 구축하였다.

그러나 MS / DOS하에서는 객체지향파라다임을 완전하게 도입하기가 어려워서 적용영역의 한계가 있고, 역시 넓지 않을 것으로 생각한다.

현재 개발중인 UNIX / X-Window / Motif 환경에서의 CARS는 C와 C++를 기본 툴로 해서 모델링과 객체지향 분석 및 설계방법을 표준화 시켜서 classification을 하고 라이브러리를 구축하고 있다. CARS와 병행해서 객체지향 방법론과 C++에 의한 재사용의 지침서의 표준화를 연구하고 있다.

객체지향 방법론으로서는,

1) information modelling과 요구분석
2) 영역분석과 시스템 요구분석
3) 객체지향 분석과 설계
4) classification 방법

을 중심으로 개발하고 있다. 또 C++에 의한 재사용 지침서는,

1) 설계 지침 : 재사용과 parameterization을 위한 설계
2) 코딩 지침 : CARS의 User Interface를 이용하여 이식성, 컴파일 문제 등을 지원하는 방법
3) 문서작성 지침 : 읽기 쉽게하고, 콤멘트, mnemonics 및 스타일 지원
4) 관리 지침 : 재사용 부품을 위한 동기부여,

콘트롤 방법, 하청개발과 유지보수 지원

5) 품질관리 지침 : 객체의 encapsulation 유지와 coupling 및 cohesion 체크를 할 수 있도록 지원

등의 기능을 부여하도록 연구하고 있다.

5 결 론

소프트웨어의 자동생산은 CASE 개발과 함께 많은 전문가들이 관심을 가지고 연구개발 중에 있다. 지금까지 연구결과에서 문제점으로 제기된 소프트웨어 분류방법과 추상자료형과 계승방법들을 위한 연구가 이루어 지면 소프트웨어 개발의 생산성에 획기적인 공헌을 기대할 수 있을 것이다.

Paramterized Specification, Category Theory 및 Formal Specification 등이 이러한 문제점을 해결할 수 있을것으로 기대한다. 그러면 끝으로 지금까지 개발된 소프트웨어 자동생산에서 연구개발된 부품화에 대한 효과를 평가하기로 하자.

가. 부품의 평가척도

부품 혹은 부품 라이브러리의 이용률에 대한 평가 척도에 관해서 소개한다. 여기서는 평가 척도로서 다음 4가지를 설명하고 있다.

- 잔존율(r)
- 재사용률(u)
- 효율성(e)
- 융통성(f)

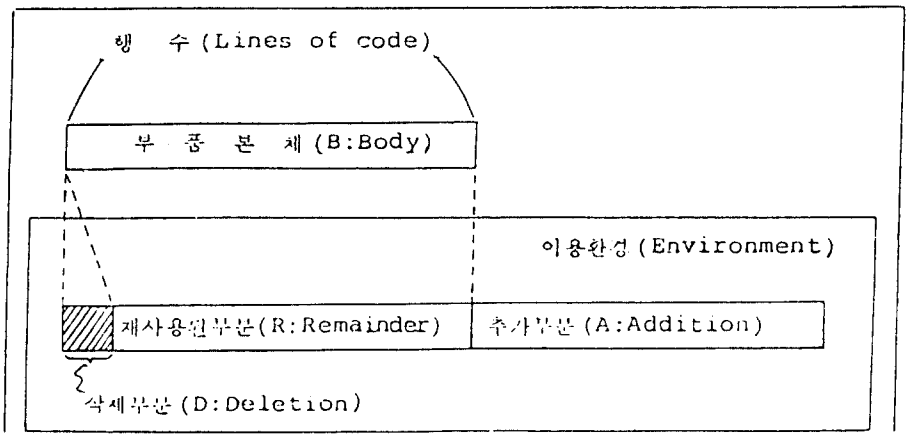
각 평가척도를 산정하는 방식이 〈표 3〉에 나타나 있다.

부품에 대한 추가와 삭제의 경우는, 부품의 성공율을 나타내는 척도로서 이용할 수 있고, '잔존율'과 '재사용율'의 형태로 정식화하고 있다. 더우기 부품을 재단(Customize : 부품을 이용하는 환경에 맞도록 편집하는 일련의 행위)하는 작업량이 얼마나 적은가는 '융통성'을 척도로서 정의하고 있다. 그리고 이들의 척도의 정의로부터 다음과 같은 몇가지 사항을 끄집어 낼 수 있다.

- 부품의 효율성은 재사용 및 잔존율과 완전한 상관관계가 있다.
- 잔존율은 주로 부품 개개의 성공율을 나타내는 척도이다.
- 재사용율은 주로 환경에 영향을 받지 않는 부품들을 준비하고 있는가를 나타내는 부품 라이브러리의 성공율을 나타내는 척도이다.

나. 품질등에 미치는 효과

부품화 및 재사용에 의한 효과라고 하는 것은, 부품 단독만의 효과뿐만 아니라,



〈표 3〉 부품 이용률에 대한 평가척도

잔존율(Remainder) : $r=\frac{\text{재사용된 부분의 행수(R)}}{\text{재사용한것의 총 행수(B)}}$ $(0\leq r\leq 1)$
재사용율(Reusablilty) : $u=\frac{\text{재사용된 부분의 행수(R)}}{\text{재사용환경에 있어서의 달성 프로그램의 총 행수(R+A)}}$ $(0\leq u\leq 1)$
효율성(Effectiveness) : $e=\frac{\text{사용환경에 있어서의 완성 프로그램의 총 행수(R+A)}}{\text{사용환경에 있어서의 삭제된 행수(D)+추가된 행수(A)}}$
융통성(Flexibility) : f=주어진 재사용 유니트B가 얼마만큼 다른 환경에서 재사용될 수 있는지 여부
(주) 행수는 소스 프로그램의 행수이다.

부품 재사용에 의해 파생된 여러 종류의 효과로 확대 해석할 수 있다. 그리고 이 효과의 내용은 크게 다음 3가지로 집약할 수 있다고 생각한다.

- 소프트웨어 개발 생산성의 향상
- 소프트웨어 제품의 질 향상
- 소프트웨어 개발환경의 향상(하드웨어, 소프트웨어, 조직, 인력 등)

참 고 문 헌

1. 이경환 소프트웨어 재이용에 관한 연구, 과기처보고서, 1990.
2. 이경환 CASE 환경에서 재사용도구의 고찰. 정보과학회지, 1991. 6, 한국정보과학회.
3. 이경환 소프트웨어 재사용을 위한 모델링, 소프트웨어 공학 연구회지 3권 4호, 1990. 12, 정보과학회.
4. Ian Thomas, PCTE Interface : Supporting Tools in Software Engineering Environment, Nov. 1989, IEEE Software.
5. Kien-phng Vo, IFS : A Tools to Build Application System, July 1990, IEEE Software.

이 경 환

저자약력

1964 중앙대학교 수학과 졸업
1980 중앙대학교 대학원 수학과(이학박사)
1970 중앙대학교 전자계산학과 교수
1982-3 미국 Auburn 대학 객원교수(소프트웨어 공학 연구)
1986 서독 Bonn 대학 객원교수(프로그래밍 언어 연구)
1991 중앙대학교 전산센터 및 슈퍼컴연구소 소장
1990 한국정보과학회 부회장
소프트웨어 공학 연구회 위원장