통합 객체지향 관리기 중점을 둔 F77/J++ 생성기 설계

선수균*, 송영재**
*동원대학 사무자동화과
**경희대학교 전자계산공학과

A Study on Design of F77/J++ Auto-Generator for Based Integration Object-oriented Manager

Su-Kyun Sun* , Yong-Jea Song**
*Dept. of Office Automation. DongWon Collage, Korea
**Dept. of Computer Engineering, KyungHee University, Korea

요 약

웹 환경하에서 소프트웨어 개발을 통합하는 방법과 기존의 Legacy code를 새로운 소프트웨어로 자동 생성하는 연구가 집중되고 있고, 새로운 소프트웨어의 단 시간에 습득하려는 노력이 가중되고 있다. 따라서, 본 논문에서는 Legacy code를 통합하고, 호환성, 완전성이고, 적응성이 있는 효율적인 유지보수를 관리해 주는 기능을 담당하는 통합 객체지향 관리기를 제안하고, 제안한 통합 객체지향 관리기를 중점을 둔 F77/J++ 코드 생성기(FORTRAN-77/Java Code generator)를 설계한다. 이는 향후 시스템의 통합에 잇점인 소프트웨어의 재사용성을 극대화하여 생산성을 향상 시키는 프로토타이핑을 지원할 것으로 기대된다.

1 서론

객체 지향 패러다임의 확산으로 인하여 소프트웨어 개발을 위한 객체 모델의 사용이 일반화되고 있으며 아울러 객체 지향 패러다임을 지원하는 자동화 도구의 사용이 확산되고 있다[1]. 기존의 시스템을 통합시키고, 소프트웨어의 생산성을 높이기 위해서 표준화시키며, 효율적인 유지보수의 전반적인 것을 관리해주는 기능이 매우 필요하게 되었다. 따라서 웹 환경하에서 기존 legacy system code를 재 사용하여[2] 호환성, 완전성, 적응성이 있는 유지보수를 효율적으로 할 수 있게 자동 생성하는 시스템을 설계함으로써 시스템의 코드를 재사용할 수 있는 것과 코드를 생성하는 생성기를 설계방향을 제시하려고 한다. 본 논문에서는 관리기를 제안하고 소프트웨어 개발 시간단축, 효율적인 유지 보수, 소프트웨어 재 사용하여 생산성을 향상시켜 주는 F77/J++ 생성기(FORTRAN-77/Java Code generator)를 제시하여 생성기를 설계한다. 기존의 Legacy code를 웹 환경에 적합한 Java 코드로 자동 생성하여 시스템을 통합하고, 재 사용하여 소프트웨어의 생산성 향상을 추구하는 것이다. 이로 인한 소프트웨어의 개발을 쉽게 해 주고 다음에 재사용 할 수 있게 부품을 클래스별로 저장하고 소프트웨어 개발시간을 단축 할 수 있을 것으로 기대된다. 따라서 Legacy code를 재공학의 기반으로 효율적인 유지보수 등을 관리해주고 전반적인 기능등을 조절해주는 통합 객체지향 관리기를 제안하여 F77/J++ 코드 생성기를 설계함 소프트웨어 개발 시간을 단축하여, 소프트웨어의 공학의 주목표인 소프트웨어 생산성을 높이는 일이 본 논문의 목적이다. 재공학의 한 분야인 코드 자동 생성으로 인한 재사용성을 극대화하여 소프트웨어 재공학으로 인한 생산성향상, 소프트웨어 개발의 통합화로 인한 개발시간 단축과 유지 보수기능을 함상시킬 수 있는 것이다

2. 관련 연구

J.Ewer와 B.Knight의 논문에서는 Case study를 집중적으로 연구하고 있으며 과거의 코드 (legacy code)를 변환하는 단계를 아홉 단계로 구분하여 설계 구현하였고 연구 평가한 결과도 상당한 효과가 있었다[2]. 여기에서는 재공학 기법을 적용했으며 FORTRAN CFD(Computational Fluid Dynamic) code를 C++로 변환하는 단계를 구현하였다.

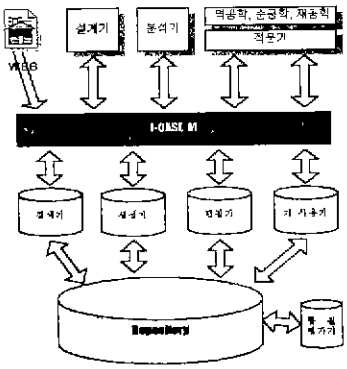
최근 연구는 프로그램 변환을 지원하기 위해 자동화된 소스코오드로 재구성(re-structuring)하는 툴로 발전하고 있다. 이러한 툴이 Sage++이다[4]. Sage++는 객체지향된 툴키인데 재구성을 실현하는 툴이다. 여전히 계속 연구가 진행되고 있으며 자동 코드를 재구성하는데 가능하게될 것이다.

Angus 와 *Curtis*의 논문에서는 수치적인(numerical code) 코드 FORTRAN-77를 C++ 코드로 재공학 기법으로 구현하였다. 여기서는 재공학의 여러 단계의 장점과 단점을 자세하게 묘사하고 있다[5]. 또한 Byrne's의 논문에서는 FORTRAN 시뮬레이션 코드를 재구현하여 Ada을 위한 재공학 기법으로

구현 하였다[6]

3. 통합 객체지향 관리기 제안

기존의 시스템은 개발자에게 비시각적 개발환경하에서 많은 단점을 내포하고 있으며[9] 요구 변경시 신속한 대처를 못한다. 이에 본 논문에서는 소프트웨어 개발자에게 시각적 개발환경을 제공하고, 개발과정의 부분적 자동화를 이루고, 객체지향 방법론 설계를 정착시키고 커뮤니케이션 수단을 제공하여 소프트웨어의 생산성을 높이고, 완전성, 호환성 있는 효율적인 유지보수를 전반적으로 관리해주는 관리기를 제안한 것이 통합 객체지향 관리기(Integrated Object-oriented Manager. I-CASE M)이다



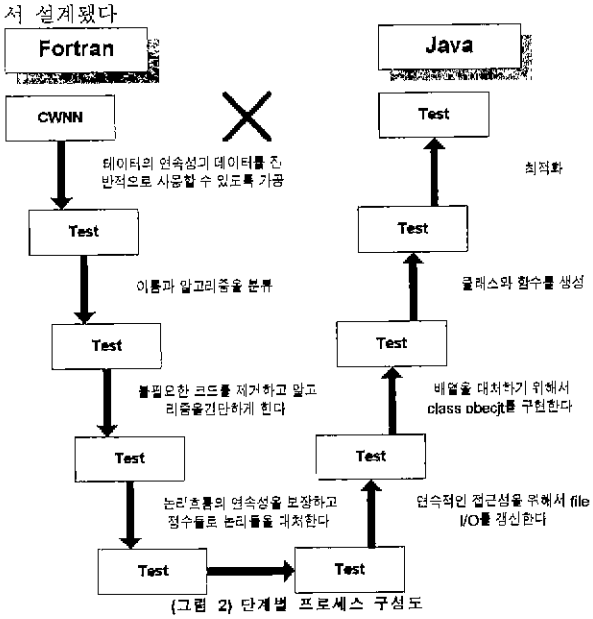
(그림 1) 통합 객체지향 관리기 전체 구성도

적용기는 순공학, 역공학, 재공학을 통합 관리해 주고 커뮤니케이션 수단 제공하는 관리기 이다. 즉, I-CASE M는 소프트웨어 개발자에게 시각적 개발환경을 제공하여 개발과정의 부분적 자동화하여 요구변경에 신속한 대처로 생산성을 높이는 것이 주 목적이다. 또한 통합하여 역공학, 순공학,재공학의 정보를 적용 할 수 있도록 관리하는 기능이며 정보를 검색하여 재사용 할 수 있게 추출기를 통해서 재사용 생성코드를 관리하는 기능이다. 이 제안한 것 중에 하나인 부분적 자동화 생성기는 본 논문에서 적용하여 F77/J++ 생성기를 설계하려고 한다. 정보 저장소는 Oracle DB 를 사용하여 부품을 클래스별로 저장하고, 모듈별로 분류하여 상속개념을 추구해서 저장 검색한다.

4. F77/J++ 생성기 설계

본 논문에서는 재공학의 한 분야인 코드 자동 생성으로 인한 재사용성 극대화하여 생산성향상과 완전성이고 효율적인 유지보수를 관리해 주는 생성기를 만들기 위한 프레임 워크의 설계방법을 제시한다. F77/J++ 부분적 자동 생성기의 전체 시스템 구조는 다음 (그림2)과 같으며 아홉단계로 구성되고 있다. 본 논문에서는 legacy code인 F77에서 웹 환경에서 구현할 수 있는 Java언어로 자동 생성하는 생성기를 설계함으로써 자동 변환하여 재구성하는 단계이며 멀티 단계 프로세서로 설계하는 과정을 다루기로 한다.

여기서 멀티 단계에서 네 번째 단계는 FORTRAN code가 완전한 알고리즘적인 상세설계로 기록될 것이고 호환성(compatibility)이 있다. F77/J++ 자동 생성기 설계단계는 아홉가지의 재공학 전략으로 이루어지며 ,변형과정으로 나눌 수 있다 설계 방법은 소프트웨어 공학 설계 원리로 적용했으며 적응성 있고 호환성이 있는 유지보수와 완전성을 고려해서 설계됐다



(그림 2) 단계별 프로세스 구성도

첫 번째 단계는 데이터 일관성을 확실하게 하고, 포괄적인 데이터를 만든다 동질성이 없는 원시화일은 다른 파일로 결합된다. 열린 파일(Open file)의 숫자를 허용하는데 제한이 있기 때문에 이 단계가 필요하다. 생성기에서는 이 단계가 필요하며 철저하게 사용된다

```
Legacy FORTRAN code
CALL BUOYAN (RMETHD, ELEMAT,     ELEVOL, TEMPER,  )
SUBROUTIME BUOYAN( RMETHD,       ELEMAT, VOLUME, T,  )
INTEGER ELEMAT (TOTELE)
REAL   VOLUME(TOTELE), T(TOTELE)
B = T(I)   * ...
```

(리스트 1) COMMON과 파일로 된 Passing data

(리스트 1)는 COMMON 블록안에 데이터 파싱을 하는 예제를 나타낸 것이다

```
becomes
CALL HBOUND ( ICELL, . )
SUBROUTINE HBOUND ( ICELL,  )
INCLUDE 'DATABASE  INC'
INTEGER    ICELL
H (ICELL) = H(ICELL) *
```

(리스트 2) 아규먼트 파싱하는 것(Revised argument passing for COMMON data)

서브루틴을 포함한 더미 아규먼트(argument) 이름은 새롭게 정의된 COMMON 배열 변수들로 직접 대신 할 수 있다. (리스트 2)는 code fragment로 지적된 것을 어떻게 배열 인덱스 값이 배열 요소들로 대신에 패스(pass)되는 지를 나타내고 있다.

두 번째 단계로는 이름과 알고리즘을 설명한다.

FORTRAN-77에서 여섯 가지 표준 문자와 루틴이름(routine names)은 롱거(longer),와 하층 케이스(lower case),와 이름(names)으로 대체 할 수 있다 이것은 기능적인 의미(functional meaning)와 사용을 전달한다

Legacy FORTRAN	*New naming convention*
MCSOLV	solve_momentum
CALGEN	calc_generation_rate
RDINFF	read_inform_file

```
H               read_inform_file
TEMPER          enthalpy
U               temperature
KINET           Cu_velocity    kinetic_energy
DISSIP          dissipation_rate
```

(리스트 3) 코오드 분류를 위한 이름 변환

세 번째 단계는 괴다한 코오드를 제거하는 것과 단순화(simplification)시키는 것이다 현 프로젝트를 위한 요구가 없는 code path와 변수는 시스템으로부터 제거 될 수가 없다

Legacy code fragment *Equivalent code abstracted*

```
MITERS = MAXITR(4)      NTEGER VAR_W_VELOCITY
CALL SORSCH(    )       PARAMETER( VAR_W_VELOCITY = 4)
SERROR(4) = RESIDU      CALL SORSCH( VAR_W_VFLOCITY,  )

RELAXA = VRELAX(4)      CALL LINRLX( VAR_W_VELOCITY,  )
CALL I NRLX(    )
VARERR(4) = RESIDU
```

(리스트 4) 재배치하는 간단한 할당 문장 (추상화 시키는 것)

네 번째와 다섯단계는 모든 논리적인(logical) 변수를 변환시키고 제어의 견실한 사용이 확실하게 한다
실행할 수 있는 문장보다 오히려 CONTINUE 문장을 사용 하게 한다. 이것은 J++로 변환하는데 도움이 되고 다른 loop 루틴을 만드는데 쉽게 해 준다. 예로써 "IF (<expr>) <statement>"가 "IF(<expr>) THEN <statement> ENDIF"으로 대신함으로써 다음의 J++로 변형이 쉽게 만들어질 것이다.
여섯단계에서는 호환성(compatibility)을 위하여 모든 입,출력 파일을 수정하고 재구성한다

```
Original legacy code                    New Java   code using a vector class
    Do 1 I = 1, NOCELL                      int i, j;
    Do 2 J = 1, 3                           for ( I=1 , I<=cell.length, I++)
    CENTRE( I, J ) = 0 0                    cell [i]  mid.set( 0 0, 0 0, 0 0 ),
2   CONTINUE                                    for ( j=1,   j<=cell[i]  getNum_of_pts(), j++ ){
      DO 3 J = 1, 3                               cell[i]  mid = cell[i]  mid +
      DO 4 K = 1, NPTCTY( CELTYP( I ))                  point[ cell[i]  pt_num[j]   ],
        CENTRE( I, J ) = CENTRE( I, J )             }
+       + XYZCRD( CELPTS( I, K ), J )           cell[i]  mid = cell[i]  mid /
4     CONTINUE                                      (float)   cell[i]  getTnum_of_pts(),
      CENTRE ( I, J ) = CENTRE ( I, J )
+       / REAL ( NPTCTY ( CELTYP (J) ))
    3       CONTINUE                    B  This code does not loop for the three
    1 CONTINUE                          directions because the overloaded vector
                                            operators "+" and "/" had
```

(리스트 5) vector algebra를 위한 클래스 사용하는 예

일곱 단계에서는 프로시저 루틴(procedural routines)를 위하여 클래스 메쏘드(class method)를 생성한다.

Legacy FORTRAN *Macro replacement*

```
ELSEIF (   ) THEN           } else if (   ) {
IF (   ) THEN               if (   ) {
ELSE                          } else {
ENDIF                         }
CALL                          /*  CALL REMOVED */
DO I = a, b, c              for ( I=a, I>=mun(a,b) && I<=max(a,b), I++C ) {
DO I = a, b                   for ( I=a, I<=b, I++ ) {
ENDDO                         }
SUBROUTINE   (   )          void   (   ) {
END                           }
RETURN                        return,
PRINT*,                     System out println(       ),
nnn CONTINUE                Label_nnn.
GOTO nnn                    goto Label_nnn.
```

(리스트 6) 매크로 대치(Macro replacement)

```
Enhanced debug facilities for code development via class methods
float access( int mode, int var)
{

   if ( ( mode == NEWEST ) && ( var == TEMPERATURE ) ) {
      if ( data[mode]  (var) < 0 0 ) {
/ / Error negative temperature detected in cell data access
```

(리스트 7) New data access function

여덟 번째 및 아홉 단계에서는 최적화(Optimisation)와 enhancement로 이루어진다 최초로 class 객체 C++버전은 통계적으로 객체들의 선언된 배열로 사용된다. 개인객체가 요구에 생성되어지고 소멸되어지는 것을 위해서 객체에서 포인터의 배열이 구현하기가 가능하다 초기의 재공학 시스템은 cell 객체를 위해서 직접 데이타 접근을 사용했으나 그러한 접근은 쉽게 제어와 모니터 되도록 할 수가 없었다

5. 결 론

기존에 존재하는 대부분의 CASE 도구들은 플랫폼에 의존적인 구현 언어로 작성되어 있기 때문에 이식성이 매우 부족하고, J++은 추상적이고 정형적으로 작성할 수 있다 반면, 명세로부터 구현을 직접 도출하기는 쉽지 않다 이에 본 논문에서는 이식성과 관련한 부수적인 문제점의 해결을 위해서 표준으로 통합의 기반을 이루고 객체지향 개념을 도입하여 역공학을 이용하기 쉽게 포트란언어로 작성된 명세로부터 Java코드를 자동 생성하는것을 설계했다. 또한 통합 객체지향 관리기를 제안 했다. 이것을 중심으로 F77/J++ 코드 생성기(Code generator)를 설계한다. 이것은 과거의 코오드를 재사용 할 수 있는 legacy code를 FOR77에서 J++로 생성 시키는 코드 생성기를 설계한 것이다 향후 연구과제로는 이 설계를 바탕으로 자동 생성기를 구현하는 일이다.

참 고 문 헌

[1] Rumbaugh, J et , "Object-Oriented Modeling and Design ", Prentice-Hall, 1991

[2] J.Ewer, B Knight & D.Cowell, Case study:an incremental approach to re-engineering a legacy FORTRAN Computational Fluid Dynamic code in C++, May 1995.

[3] Boehm, B., Software engineering, IEEE Trans. Comput., Chap. 25, NO. 12, 1976, pp. 1226-41.

[4] Bodin, F. et al., Sage++:An object-oriented toolkit and class library for building Fortran and C++ restructuring tools OON-SKI '94 Conf Proc., USA., PP. 122-36.

[5] Angus, I & Curtis, W., From Fortran to object orientation: experience with a production flutter analysis code OON-SKI'94 Conf. Proc., Oregon, USA., PP 174-80

[6] Byrne, E., Software reverse engineering. a case study. software-practice Experience, 1991, 21(12), 1349-64.

[7] Leschinzer, M., Modeling turbulent recirculating flows by finite-volume methods – current status and future directions. Int. J. Heat Fluid Flow, 1989, 10(3), 186-202.

[8] Patankar, S. V., Numerical Heat Transfer and Fluid Flow, Hemisphere, 1980

[9] 선수균, 송영재,"Java기반 ObjectWeb을 이용한 통합시스템 설계", 98년 추계 학술발표집 한국정보과학회, 1998

[10] Kernigan, B. & Wilson, B., lex - a lexical analysis tool, The C Programming Language, Prentice-Hall, 1988

[11] Kernigan, B. & Wilson, B , yacc - yet another compiler ,The C Programming Language, Prentice-Hall, 1988.