C 소스코드를 이용한 PAD 생성 시스템 구현

천준석*, 임진수**, 우균*
*부산대학교 컴퓨터공학과
**LG 전자 HA 제어 연구소
e-mail: {rance18, woogyun}@pusan.ac.kr*, jinsu.lim@lge.com**

Implementation of the PAD Generating System Based on C Source Code

Joonseok Chun*, Jin-Su Lim**, Gyun Woo*
*Dept of Computer Engineering, Pusan National University
**HA Control R&D Lab LG Electronics Inc.

요 약

컴퓨터 하드웨어의 발달로 인해 하드웨어의 성능을 충분히 발휘할 수 있는 소프트웨어의 개발이 요구되고 있다. 소프트웨어가 복잡해질수록 개발 시 인적 비용, 물적 비용, 시간 비용이 증가하는데, 실제 개발 비용보다는 개발 후 유지보수에 사용되는 비용이 훨씬 크다. 유지보수는 코드 수정을 통해 이루어지는데, 다른 사람이 작성한 코드를 수정할 경우에는 코드의 이해가 요구된다. 코드는 흐름도를 이용하게 되면 좀 더 쉽게 이해할 수 있는데, 복잡한 코드를 흐름도로 작성할 경우 흐름도가 자체가 복잡해져서 코드 이해가 어려워지는 경우가 많다. 이러한 단점을 보완하기 위한 방법의 하나로 1979년 PAD가 개발되었다. 이 논문은 C 소스코드를 입력받아 PAD를 자동으로 생성하는 시스템을 제안한다. 이 시스템을 이용하면 유지보수 비용에 소비되는 시간과 노력을 절약할 수 있을 것으로 생각된다.

1. 서론

어떤 연구에 따르면 소프트웨어 유지보수는 소프트웨어 프로젝트 비용의 70~90%를 차지한다고 한다[1]. 이는 하드웨어 성능의 발달로 인해 소프트웨어에 대한 요구사항이 비약적으로 증가하여 개발 난이도가 상승하고, 시스템 크기 증가를 야기하였기 때문이다. 즉, 소프트웨어 개발 후 소프트웨어의 생명주기 동안에는 끊임없는 버그 수정과 기능 변경 등의 유지보수를 수행해야 한다.

소프트웨어 유지보수는 코드 수정을 통해 이루어지는데, 이를 위해서는 코드의 이해가 필요하다. 코드 수정은 보통 유지보수 담당 프로그래머가 개발자로부터 프로그램을 넘겨받아서 수정하는 경우가 많다. 유지 보수 담당 프로그래머는 프로그램에 대한 이해도가 개발자보다 떨어지기 때문에 소스코드의 분석 작업을 통해 프로그램의 전체적인 구조를 파악해야 한다.

소스 코드를 분석하기 위해 흐름도(flowchart)를 생성하는 방법이 있다. 흐름도는 1921년 Gilbreth가 Process Chart[2]를 최초로 고안한 이후 90년 이상 사용되어 온 프로그램 구조 표시 수단으로, 노드와 에지의 조합으로 누구나 쉽게 이해할 수 있는 표현이다. 하지만 복잡한 코드의 경우 노드 간 에지 연결이 스파게티처럼 얽혀버려서 코드를 이해하기 힘들어지는 단점이 있다.

흐름도의 단점을 보완하기 위하여 1979년 일본 히타치 제작소의 후타무라 요시히코(二村良彦)가 PAD(Problem Analysis Diagram)를 개발하였다[7]. PAD는 구조적 프로그래밍 언어의 제어구조를 순차구조, 반복구조, 선택구조의 노드 조합을 통해 표현한다. 노드와 에지의 배치에 제약을 두어서, 누가 그리더라도 같은 구조의 PAD가 생성된다.

흐름도를 작성하는 방법은 개발자가 수동으로 작성하는 방법과 자동 생성 도구를 이용하는 방법이 있다. 수동 작성의 경우 시간이 오래 걸리고, 수정 작업이 힘들다. 자동 생성 도구의 경우에는 생성된 흐름도 정보만으로 코드 흐름을 파악하기 어렵다.

이 논문에서는 C 소스코드를 입력받아 PAD를 자동으로 생성하는 도구를 제안한다. 주석 형태의 메타 태그를 사용하여 개발자가 원하는 정보를 PAD에 삽입할 수 있도록 하였다. 2장에서는 기존의 흐름도 자동 생성 도구의 종류와 PAD의 사용 방법에 대해 설명하였고, 3장에서는 제안하는 시스템의 구조를 설계하였다. 4장에서는 3장에서 제안한 PAD를 이용한 흐름도 자동 생성 도구를 구현하였으며, 5장에서 결론을 맺는다.

2. 관련연구

현존하는 흐름도 자동 생성 도구는 Visustin v7 Flow chart generator[3], Eclipse Control Flow Graph Generator[4], Code Visualizer 5.06[5] 등 여러 종류가 있다. 이러한 자동 생성 도구의 특징을 나열해보면, Visustin v7 Flow chart generator는 함수 단위로 흐름도를 생성해주며, UML 출력이 가능하다. Eclipse Control Flow Graph Generator는 Eclipse에 플러그인 형식으로 사용하

는 도구로, 색을 통해 노드의 역할이 구분되는 특징이 있다. Code Visualizer 5.06의 경우 흐름도 뿐만 아니라 시퀀스 다이어그램, 나시-슈나이더만 그림 등 다양한 방법으로 코드를 시각화한다. 하지만 이러한 도구들은 흐름도만으로 개발자가 원하는 정보를 전달하지 못하고, 공간 배치가 효율적이지 못한 단점이 있다.

PAD는 흐름도와 마찬가지로 노드와 에지로 구성된다. 표 1은 PAD에서 사용하는 노드를 나타낸 것이다. PAD는 시작과 종료 노드 사이에 순차, 반복, 선택 노드 등을 이용하여 코드를 함수 단위로 추상화하여 나타낸다. 경우에 따라서 시작 노드와 종료 노드는 생략될 수 있다.

<표 1> PAD의 프로그램 구조 표기법[8]. PAD는 시작과 종료 단자 사이에 순차구조, 반복구조, 선택구조를 표현하는 요소의 조합을 통해 구조적 프로그래밍 언어의 제어구조를 표현한다.

요 소		표 기 법
기 본 요 소		절차
순 차 요 소		절차1 절차2
반복 요소	반복내용 앞에 반복조건 위치 (WHILE)	조건 — 절차
	반복내용 뒤에 반복조건 위치 (DO WHILE)	조건 — 절차
선택 요소	한 갈래 선택 (IF)	조건 — 절차 조건 — 절차
	양 갈래 선택 (IF)	조건 — 절차1 / 절차2
	여러 갈래 선택 (CASE)	조건 — 값1 절차1 / 값2 절차2 / … / 값n 절차n
서브루틴		외부 콜 / 내부 콜
단 자		
결 합 자		n / n

그림 2는 선택정렬 코드(그림 1)를 PAD로 표현한 것이다. 순차처리의 흐름은 위에서 아래 방향으로만 가능하며 흐름도와 같이 화살표 등을 이용하여 위로 올라가는 것이 불가능하다. 반복처리와 조건처리의 흐름은 조건을 기입하는 노드 우측에 트리 형태로 기입한다.

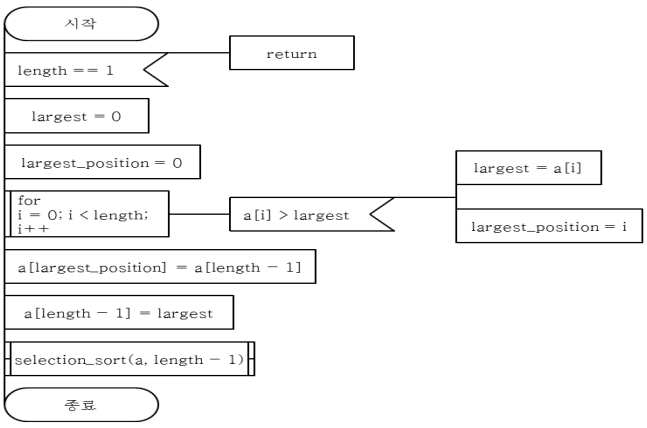
그림 3은 선택정렬 코드(그림 1)를 Visustin v7 Flow chart generator를 이용하여 흐름도로 자동 생성한 것이다. PAD와 달리 순차요소가 하나의 사각형 노드에 표현되어 있다. 또한 반복요소와 선택요소가 비슷하게 표현되어 있어서 흐름도 만으로는 반복문인지 선택문인지 판단이 어렵다. 이에 반해 PAD는 노드의 모양만으로 반복요소인지 선택요소인지 쉽게 판단할 수 있다.

PAD 표현은 흐름도를 개량한 기법이지만 (1) 시작과 종료 노드에 정보가 없고, (2) 반복문과 조건문의 트리 구조가 복잡해질수록 레이아웃의 공간 활용도가 떨어진다는 단점이 있다. 이 논문에서는 PAD의 이러한 단점을 보완하여 C 소스 코드를 입력받아 자동으로 PAD를 생성해 주는 시스템을 제안하고자 한다.

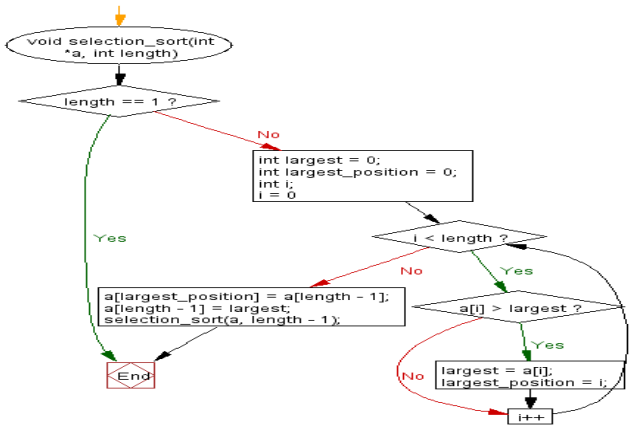
```
void selection_sort(int *a, int length)
{
    if(length == 1)
    {
        return;
    }
    int largest = 0;
    int largest_position = 0;
    int i;

    for(i = 0; i < length; i++)
    {
        if(a[i] > largest)
        {
            largest = a[i];
            largest_position = i;
        }
    }
    a[largest_position] = a[length - 1];
    a[length - 1] = largest;
    selection_sort(a, length - 1);
}
```

(그림 1) C 언어로 작성한 선택정렬 코드. 배열 포인터 a와 배열 길이를 입력받아 선택정렬을 수행한다.



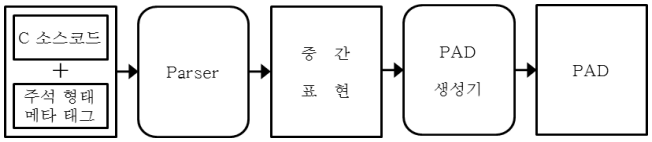
(그림 2) PAD를 이용한 선택정렬 표현. 순차 처리는 위에서 아래로, 반복과 조건 처리는 왼쪽에서 오른쪽으로 표현된다.



(그림 3) Visustin v7 Flow chart generator[3]를 통해 자동 생성된 선택정렬 코드. 순차요소가 하나의 사각형 노드에 표현되며, 반복요소와 선택요소를 구분하기 어렵다.

3. PAD 생성 시스템 설계

제안하는 시스템의 전체 시스템 구성은 그림 4와 같다. 우선 C 소스 코드에 주석 형태의 메타 태그를 추가한다. 이 출력물을 파싱하게 되면 PAD를 출력하기 위한 정보가 저장되어 있는 중간 형태의 파일이 생성된다. 이를 PAD 생성기를 통하여 PAD를 자동 생성한다.



(그림 4) 시스템 구성도. C 소스 코드에 주석 형태의 메타 태그를 추가하여 파싱을 통해 LISP 형태의 중간표현을 생성한다. 이를 PAD 생성기에 입력하면 PAD가 출력된다.

메타 태그는 개발자가 원하는 정보를 PAD 노드에 출력하기 위해 사용하는 것으로 외줄 주석인 // 뒤에 `(back quote) 두 개를 단 후, 그 사이에 노드에 들어갈 내용을 기입하는 형식으로 구성된다. 메타 태그는 설명을 넣고자 하는 코드의 바로 앞줄에 추가해야 하며, 메타 태그를 추가하지 않은 코드는 실제 코드가 출력된다. 예를 들어, length++; 라는 코드 앞에 //`length 증가`와 같은 메타 태그를 추가하게 되면 PAD 출력 시 length 증가로 출력된다. 그림 1의 선택정렬 코드에 메타 태그를 추가한 예는 그림 5와 같다.

```
//`선택 정렬(배열, 배열길이)`
void selection_sort(int *a, int length)
{
    //`배열 길이가 1과 같으면 true`
    if(length == 1)
    {
        return;
    }
    int largest = 0;
    int largest_position = 0;
    int i;
    //`배열 길이만큼 반복`
    for(i = 0; i < length; i++)
    {
        //`가장 큰 수 찾기
        if(a[i] > largest)
        {
            //`배열 값을 largest 변수에 저장`
            largest = a[i];
            //`배열 위치를 largest_position 변수에 저장`
            largest_position = i;
        }
    }
    //`배열 마지막 요소를 큰 수 위치에 저장`
    a[largest_position] = a[length - 1];
    //`가장 마지막 요소에 큰수를 저장`
    a[length - 1] = largest;
    //`선택 정렬(배열, 배열 길이 - 1)`
    selection_sort(a, length - 1);
}
```

(그림 5) 메타 태그가 포함된 선택정렬 코드. 설명을 넣고자 하는 코드의 바로 앞줄에 주석 형태의 메타 태그를 삽입한다.

파서의 경우 PAD 자동 생성에 필요한 결과를 처리할 수 있도록 OpenC++[9] 파서를 변경하여 구현하였다. 우선 메타 정보를 인식하기 위해 어휘 분석기를 수정하였고, 각 AST 노드에 메타 정보 분석을 위한 자료 구조를 추가하였다. 또한 파싱 이후 AST 노드를 순회하여 구문에 맞는 중간 표현과 PAD 생성기에서 사용할 함수 리스트 파일을 생성하도록 수정하였다.

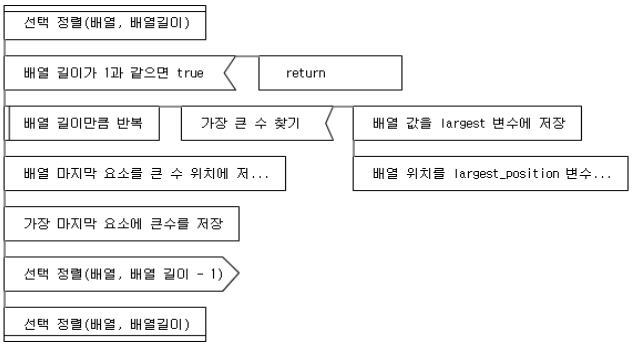
그림 6은 파싱을 통해 생성된 중간 표현이다. 함수형 프로그래밍 언어인 LISP 구조를 차용하여 작성하였다[10]. 괄호를 통해 노드 간의 관계를 나타내며, 괄호는 노드의 종류를 지정하는 식별자와 노드에 출력되는 문자열 정보로 구성된다. `(backquote) 안의 내용이 PAD의 노드에 출력되어야 하기 때문에, 소스 코드에 추가하였던 메타 태그 내용이 중간 표현에 추가된다. 메타 태그를 추가하지 않은 소스 코드의 경우에는 소스 코드의 내용이 들어가게 된다.

```
( function `선택 정렬(배열, 배열길이)` ()
    (( if `배열 길이가 1과 같으면 true`
        (( return ` return ` )) ())
    ( preloop `배열 길이만큼 반복`
        (( if `가장 큰 수 찾기`
            (( stmt `배열 값을 largest 변수에 저장` )
             ( stmt `배열 위치를 largest_position 변수에 저장` )) ))))
    ( stmt `배열 마지막 요소를 큰 수 위치에 저장` )
    ( stmt `가장 마지막 요소에 큰수를 저장` )
    ( call `selection_sort` `선택 정렬(배열, 배열 길이 - 1)` )))
```

(그림 6) 파싱을 통해 생성된 중간 표현. 괄호를 통해 노드 간의 관계를 나타내며, 괄호 안에는 노드의 종류를 지정하는 식별자와 PAD에 출력되는 문자열 정보가 저장된다.

PAD 생성기는 중간 표현을 입력받아 UI 상에 PAD를 자동으로 생성해주는 역할을 한다. PAD 생성기는 UI 상에 출력할 때 노드의 좌표, 노드의 모양, 노드의 크기, 노드 간 관계 등의 정보를 저장하는 XML 파일을 생성한다. PAD 저장 및 로드 시에도 XML 형식으로 관리되도록 설계하였다. XML 파일과 함수 리스트 파일을 입력받아서 자동 생성된 PAD가 출력된다.

기존 PAD의 시작과 종료 노드에 아무런 정보가 없는 것을 개선하여 그림 7과 같이 메타 태그 정보가 출력되도록 수정하였다. 또한 노드 모양을 타원 모양에서 선이 있는 사각형으로 바꾸어서 직관적인 코드 흐름 파악이 가능하도록 수정하였다. 함수 호출도 사각형에서 오각형으로 수정하였다.

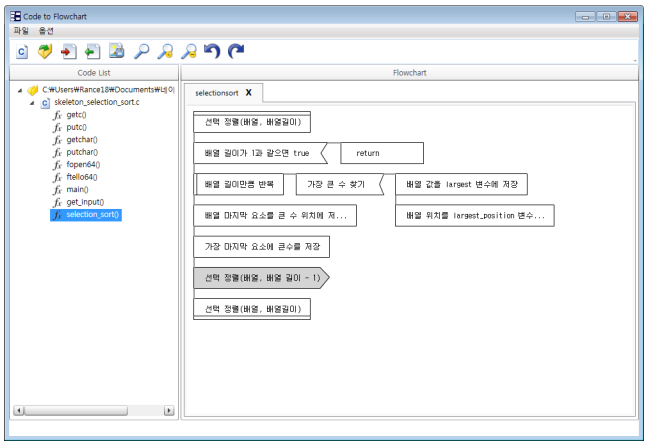


(그림 7) 제안하는 PAD 모양을 이용한 선택정렬 표현. 시작과 종료 노드에 정보를 추가하였으며, 반복문과 선택문의 자식 노드의 레이아웃을 수정하였다.

또한 공간 최적화 문제를 해결하기 위한 방편으로 노드의 최대 크기를 지정하여 노드 내용 때문에 최대 크기를 초과하는 노드의 경우 최대 크기 이상 늘어나지 못하도록 수정하였다. 또한 노드의 문자열이 노드 최대 크기보다 클 경우 말줄임표(...)를 사용하여 내용이 더 있다는 것을 표시하였다. 기존 PAD에서 반복문과 조건문의 트리 구조가 복잡해질수록 노드 배치 상 레이아웃이 깔끔하게 나오지 못하는 문제를 해결하기 위해서 자식 노드가 부모 노드보다 위로 올라가지 못하도록 수정하였다.

4. PAD 생성 시스템 구현

이 논문에서 제안하는 PAD 자동 생성 도구 구현을 위해서 .NET Framework 4.0 환경에서 개발을 수행하였다. 개발 언어는 C#이며, UI는 WPF(Windows Presentation Foundation)[11]을 이용하여 구현하였다. 동작 테스트는 Microsoft Windows XP 이상의 운영체제에서 수행하였다.



(그림 8) PAD 자동 생성 도구 실행 화면. 소스 코드를 로드하면 좌측에 소스 코드에 포함된 함수 리스트가 출력된다. 이 중 하나를 선택하면 우측에 PAD가 출력된다.

시스템은 그림 8과 같이 상단의 메뉴 바와 아이콘 바, 좌측 하단의 함수 리스트를 출력하는 Code List, 우측 하단의 PAD를 출력하는 Flowchart 메뉴로 구성된다. UI를 통해서 C 소스 파일을 로드하면 Code List 항목에 C 소스 파일에 있는 함수 목록이 출력된다. 이 중 하나를 클릭하면 우측의 Flowchart 항목에 해당 함수의 PAD가 생성된다. 노드 내용이 말줄임표로 생략된 경우에는 마우스 커서를 오버랩하면 해당 내용이 말풍선으로 팝업된다. 소스 코드 파일 하나만 로드할 수 있으며, 특정 폴더에 있는 모든 C 파일을 불러올 수 있다.

노드를 드래그하여 노드 배치를 변경할 수 있도록 하였으며, 함수 호출의 경우 더블클릭하여 새 탭에 호출된 함수가 출력되도록 하였다. 특정 노드를 우클릭 시 팝업되는 소스 보기 메뉴를 통해 코드를 수정할 수 있다. 또한 저장과 불러오기 기능을 통해 PAD를 관리할 수 있으며, .bmp와 .jpg 등의 이미지 출력 기능을 지원한다.

5. 결론

이 논문은 이전 연구[6]에서 제안하였던 흐름도 자동 생성 도구를 응용하여 자동으로 PAD를 생성하는 시스템을 구성하는 것에 목표를 두었다. 흐름도의 문제점을 해결하기 위해 PAD를 사용하였다. 소스 코드가 아닌 개발자가 전달하고자 하는 정보를 PAD에 출력하기 위해 주석 형태의 메타 태그를 사용하였으며, 메타 태그를 달지 않으면 소스 코드가 출력되도록 하였다. 생성된 PAD는 수정이 가능하며, 저장 및 불러오기가 가능하다.

이 논문에서 제안한 시스템은 완전한 소스 코드가 아닌 스켈레톤 코드에 메타 태그를 추가한 것만으로도 PAD 생성을 할 수 있다. 따라서 유지보수 단계뿐만 아니라 개발 단계에서도 PAD 생성 시스템을 활용한다면 소프트웨어 품질 향상 및 비용 절약에 기여할 것으로 생각된다.

참고문헌

[1] T. M. Pigoski, “Practical Software Maintenance: Best Practices for Managing Your Software Investment,” Wiley & Sons Inc, 1996.

[2] Frank Bunker Gilbreth and Lillian Moller Gilbreth, “Process Charts,” American Society of Mechanical Engineers, 1921.

[3] Aivosto Oy, “Visustin v7 Flow chart generator,” http://www.aivosto.com/visustin.html, 2013년 3월 20일 방문.

[4] Aldi Alimucaj, “Eclipse Control Flow Graph Generator,” http://eclipsefcg.sourceforge.net/, 2013년 3월 20일 방문.

[5] CodeDrawer Co., “Code Visualizer,” http://www.codedrawer.com/, 2013년 3월 20일 방문.

[6] 천준석, 이기화, 우균, “메타 정보와 Graphviz를 이용한 흐름도 자동 생성 도구 구현”, 제38회 정보처리학회 추계학술발표대회 논문집 제19권 제2호, 2012.

[7] 二村良彦, 川合敏雄, 堀越彌, 堤正義, “PAD(Problem Analysis Diagram)によるプログラムの設計および作成”, 情報處理學會論文誌21卷4号, 1980.

[8] 日立製作所, “プログラム構造表記法(PAD)”, http://www.hitachi.co.jp/Prod/comp/soft1/manual/pc/d378270/LANG0027.HTM, 2013년 3월 20일 방문.

[9] Shigeru Chiba, “OpenC++,” http://opencxx.sourceforge.net/, 2013년 3월 20일 방문.

[10] Harold Abelson, Gerald Jay Sussman, “Structure and Interpretation of Computer Programs - 2nd Edition,” The MIT Press, 1996.

[11] Microsoft, “Windows Presentation Foundation,” http://msdn.microsoft.com/ko-kr/library/ms754130.aspx, 2013년 3월 20일 방문.