XML에 기반을 둔 C 원시 코드의 주석 관리 시스템

박 근 옥[†] · 임 종 태[††]

요 약

C 언어가 주로 사용되는 미션 크리티컬 응용 소프트웨어 영역에는 문서화, 간결성, 정확성을 갖는 원시 코드가 요구된다. 본 연구는 XML 기반의 C 원시 코드의 주석관리 체계를 제안한다. 이 주석 관리체계는 주석 사용자 모듈, 검토자 모듈, 주석 추출 모듈, 주석 추적 연계 모듈, 주석 태그 정의 모듈, 저장관리 모듈을 포함하는 6개 모듈로 구성된다. 본 연구에서 정의된 XML 주석 태그는 IEEE 표준 1028과 IEEE 표준 1012를 적용하는 개발공정 활동의 범주를 포함한다. C 원시 코드에 주석을 삽입하고 추출하기 위하여 XML 스키마가 사용되며, 주석 추출 결과의 시각적 표시 처리를 위하여 XSL-FO가 사용된다.

An XML-based Comment Management System for C Source Code

Geun-Ok Park[†] · Jong-Tae Lim[††]

ABSTRACT

Well documented, simplified and clarified source code is required for the mission critical application software area in which C programing language is generally used. We suggest an XML_based comment management system for C source code. The comment management system is composed of 6 modules including comment user module, reviewer module, comment extraction module, comment traceability link module, comment tag definition module and storage management module. The XML comment tags defined in this paper cover categories of the development process activities applying the IEEE standard 1028 and IEEE standard 1012. The XML Schema is used to insert comments into C source code and to extract XML tags from C source code and the XSL-FO is used for the visual display processing of comment extraction results.

키워드 : 주석관리(Comment Management), XML 스키마(XML Schema), C 원시코드(C Source Code), XSL-FO

1. 서 론

좋은 프로그램(원시 코드)이 어떤 것인지를 명확히 정의하기는 어려우나, 일반적으로 단순하고 명확하여 누구나 그 내용을 쉽게 파악할 수 있도록 작성된 원시코드를 의미한다. 원시 코드를 단순 명료하게 작성하기 위한 절대적인 표준과 지침은 없으나 주석의 사용은 원시 코드에 대한 관리를 용이하게 하고 생산성을 높이는데 그 목적이 있다[1]. 오늘날 객체지향 방법론과 객체지향 언어의 사용이 확대되고 보편적인 경향을 보이고 있으나 우주 항공, 원자력 등 실시간 성능제약 요건이 중요한 미션 크리티컬(mission critical) 영역은 코드의 문서화, 간결성, 정확성을 강조하며 있으며 구조적 방법론과 C 언어를 사용한 원시 코드 작성이 아직도 주류를 이루고 있다. 이러한 영역의 C 언어 코드 작성에 사용할 수 있는 일반 지침이 제시되어 있다[2].

좋은 원시 코드의 작성 요건 중 간결성과 정확성은 원시 코드의 작성(구현)과 시험 시점에서의 철저한 확인과 검증, 시험 활동을 통하여 합리적인 수준으로 달성 가능하다. 컴퓨터 산업분야에 많이 적용되는 IEEE 표준 1028은 원시 코드, 소프트웨어 설계명세서, 소프트웨어 요구명세서 간의 연관관계를 체계적으로 검토하는 공정과 활동내역을 제시하고 있으며, 원시 코드의 간결성과 정확성을 객관적인 입장에서 검토할 수 있는 방향을 제시하고 있다[3]. 반면에 원시 코드의 문서화 관점은 다른 요건에 비하여 취약한 측면을 보이고 있다. 원시 코드 문서화의 바람직한 방향 중의 하나는 원시 코드가 작성 완료된 후 설계명세서를 직접 참조하지 않아도 원시 코드가 갖는 기능을 쉽게 이해할 수 있도록 작성하는 것이다. 이를 위하여 원시 코드 작성자는 원시 코드에 사용되는 변수의 이름을 사람이 이해하기 쉬운 형태로 사용하거나, 풍부한 주석(comments)을 추가하기도 한다.

미션 크리티컬 영역의 원시 코드는 개발완료 후 10~30년 동안 운영되는 경우가 일반적이기 때문에 유지보수가 필연적으로 발생한다. 운영주기가 짧은 원시 코드의 경우에는 유지보수가 필요할 경우 코드 개발회사 또는 관련 개발

† 정 회 원 : 공주대학교 대학원 컴퓨터공학과
†† 종신회원 : 공주대학교 컴퓨터공학과 교수
논문접수 : 2004년 3월 10일, 심사완료 : 2004년 5월 13일

자의 협조를 통하여 비교적 쉽게 해결이 가능하다. 반면에 운영주기가 긴 원시 코드의 경우에는 코드 개발회사의 소멸, 관련 개발자의 협조 불가능 등의 이유로 인하여 유지보수가 쉽지 않다. 따라서 본 연구는 문서의 구조와 내용 취급에 강력한 기능을 갖는 XML(eXtensible Markup Language) 관련 기술을 적용하여 C 원시 코드의 주석을 효율적으로 관리할 수 있는 주석 관리체계를 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 C 언어의 주석 표현과 관련 연구사례를 검토한다. 3장에서는 C 언어로 작성된 원시 코드 관리체계의 모델 구성과 모델이 포함하는 각 모듈의 기능, 주석 작성을 위한 XML 태그 정의 내용을 기술한다. 4장에서는 원시 코드 작성자가 정의된 XML 태그를 사용하여 유효한 주석 문서를 작성하였는지를 검증하는 XML 스키마를 설계한다. 5장에서는 prototype 구현에 적용한 XML 관련 기술과 XML 스키마의 구현 결과를 제시하며, 마지막으로 6장에서는 향후의 연구방향을 논한다.

2. 관련 연구

2.1 C 언어의 원시 코드 주석 표현

원시 코드 내부에 주석을 표현하는 방법은 프로그래밍 언어마다 조금씩 다르다. C 언어에서는 키워드나 식별자의 중간에 주석을 사용하는 것을 제외하고, 원시 코드 내의 어느 부분에서든 사용이 가능하다. C 언어에서 주석은 한계자(delimiter) /* 표기와 */ 표기 사이에 있는 임의의 문자열이다. 하나의 원시 코드 파일에서 main() 블록은 하나만 존재하며, main() 블록 외부에는 다수의 함수 블록이 존재하는 형태를 갖는다. 주석은 각 블록의 내부 또는 외부의 어느 위치에서나 사용될 수 있다. (그림 1)은 C 언어로 작성되는 원시 코드의 블록 위치와 주석 사용에 대한 일례이다.

```
/* 시작 시점에 주석을 첨가한다. 저작권, 작성자 신상정보, 작성일자, 수행기능, 형상관리 이력 등 원시 코드에 대한 상위 수준의 내용이 자유스러운 형태로 기입된다. */
#include <stdio.h>
#include <string.h>
/* 작성자 정의한 전역변수의 의미, 사용 용도에 대한 주석이 첨가되기도 한다 */
#define MAXSTRING 10
    ...
void calculation(char *) ; /* 함수 설명은 함수의 전후에 위치할 수 있다. 유지보수 이력이 포함될 수 있다.*/
    ......
void calculation(char *)
{
     /* 지역변수의 대한 주석이 사용될 수 있다. */
    ...
}
    ...
/* main 블록에 대한 주석이 첨가 될 수 있다. */
int main(void)
{
    char s1[MAXSTRING] ;
    /* 함수 사용에 대한 주석이 사용될 수 있다. */
    calculation(s1) ;
        ......
}
/* main() 이후에 main() 이전과 유사한 형태를 갖는다.*/
```

(그림 1) C 언어의 주석 사용 일례

2.2 주석에 대한 연구 사례

Marovac는 절차지향 언어를 대상으로 키워드(keywords), 페이싯(facets), 트레이스백(traceback), 참조(sentences), 서술(description)의 5가지 유형의 플래그 문장(flagged sentense)을 사용하여 문서화 정보를 기술하고 도구에 의하여 원시 코드에 대한 탐색과 추출, 삽입하는 방법을 연구하였다[5]. 이 연구는 주석 정보의 복잡한 구조와 많은 종류의 플래그를 사용하여 원시 코드의 문서화 정보를 표현한다. 이 때문에 원시 코드 문서화 정보를 생성하고 이해하는데 부담이 많다.

C++ 언어로 작성된 프로그램의 유지보수를 지원하기 위하여 C++ MT라 명명한 도구가 제안되었다[6]. C++ MT는 복잡도 측정 도구, 프로그램 시각화 도구, 프로그램 문서화 지원 도구, 객체지향 설계 및 프로그래밍 지원 도구의 4 개 모듈로 구성되어 있다. 특히 프로그램 문서화 도구는 정형화된 문서화 정보 획득을 목적으로 원시 코드를 구성하는 구성 성분(단위 파일, 클래스, 멤버 함수, 일반 함수 등)을 세분화하여 정의하고 있다. 이 연구는 정형화된 문서화 정보를 유지하기 위하여 코딩할 때 프로그래머가 직접 문서화 관련 정보를 삽입해야 하는 부담이 증가되지만 소프트웨어의 유지보수 단계가 개발주기의 전체 비용에서 60% 이상을 차지하기 때문에 감수할 수 있는 대가라고 주장하고 있다. 이 연구의 결과는 C++ 언어로 작성된 원시 코드를 주 대상으로 삼고 있기 때문에 본 연구가 추구하는 C 언어로 작성된 원시 코드의 주석 관리 체계에는 적용하기 어렵다.

Antonioi 등은 원시 코드에 산출물(설계사양, 요구분석사항 등) 관련 정보를 주석으로 삽입하여 원시 코드와 문서 사이의 추적성(traceability)을 가능하게 하는 방안을 제안하고 있다[7]. 이 연구는 원시 코드 자체의 관리보다는 원시 코드와 산출물인 문서간의 연동체계에 목적을 두고 있으며 C++ 언어로 작성된 코드를 대상으로 하고 있다.

Bargeron 등은 주석을 표현하는 방법론으로 XML로 정의된 CAF(Common Annotation Framework)를 통하여 주석을 문서의 특정 위치에 연결하는 방법, 주석의 유형, 연관관계의 표현 등에 대한 골격을 제시하였다[8]. 이 연구는 전자책이라는 형태의 디지털 문서에 주석을 달아 웹 기반의 동기적인 공동작업 수행을 지원하는데 목적을 두고 있다.

최근 국내에서는 개발된 원시 코드를 분석하여 효율적인 유지보수를 지원함과 동시에 재공학 과정에서 재사용 가능

한 소프트웨어 모듈을 제공하기 위한 연구가 수행되었다[9-11]. 조현훈 등은 원시 코드 분석, McCabe와 BP/Win 도구를 이용한 클러스터링, 추출된 산출물의 재변환 및 저장에 대한 연구를 수행하였으며[9], 개발된 원시 코드를 입력으로 하는 유지보수 역공학 산출물 활용 지침을 제시하였다[10]. 조현훈 등의 연구는 역공학을 통한 소프트웨어 유지보수 웍플로우(workflow)의 확립에 주안점을 두고 있다.

장근실 등은 JML(Java Markup Language)을 이용하여 Java 원시 코드로부터 주석을 추출하고 추출된 정보를 XML 형식의 문서로 개발조직 또는 유지보수 조직에 제공하기 위한 연구를 수행하였다[11]. Java 원시 코드를 대상으로 하는 이 연구는 JML 도구의 이용 및 XML DTD에 바탕을 둔 태그 사용이라는 특징을 갖는다.

최근 상용제품으로 출시된 Microsoft의 Visual Studio.Net은 XML과 주석을 사용하여 문서를 만드는 기능을 포함하고 있다[12]. 개발자는 <표 1>과 같은 XML 태그를 사용하여 원시 코드 작성 시에 주석을 삽입한다. Visual Studio.NET은 개발자가 작성한 원시 코드(소스 파일)를 해석하여 XML 주석을 추출하고 이를 HTML 파일로 표시한다. Visual Studio.NET의 주석 생성 방법은 XML을 활용하고 있으나 C# 언어로 작성한 원시 코드에만 적용할 수 있고 주석 추출의 결과를 HTML로 표시 및 관리한다.

〈표 1〉 Visual Studio.NET의 주석 태그

태그 유형	사 용 용 도
<summary>	클래스, 메서드, 속성의 간단한 한 줄 짜리 개요.
<remarks>	보다 길고 자세한 내용 기술. 하이퍼 링크를 만드는 <cref> 뿐만 아니라 <para>와 <list> 포맷팅 태그를 포함.
<value>	속성의 설명
<exception>	메서드나 속성으로부터 발생되는 예외
<param>	메서드 매개변수
사용 일례	///<remarks> ///C#의 주석 사용 예제 ///</remarks>

3. 원시코드의 주석관리 모델

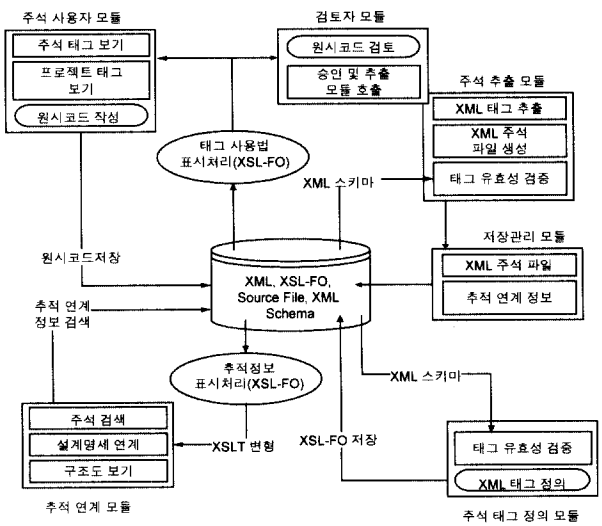
3.1 모델의 구성

주석은 원시 코드를 이해하고 유지보수를 수행하는데 도움을 제공하며, 개발과정에서 개발자와 검토자 간의 대화수단으로도 사용할 수 있다. 반면에 주석을 어느 수준과 어느 위치에 작성해야 하는지에 명확한 지침이 없고, 원시 코드 작성자의 기술수준과 경험에 따라 개인 편차가 크다. 기존의 관련 연구들은 주석이 일정한 형식 없이 자유스럽게 사용되어 왔기 때문에 적절하게 사용되지 못하는 경우 원시 코드를 이해하는데 오히려 방해가 됨을 지적하고 있다. 이는 원시 코드 작성자가 소프트웨어 개발 프로젝트 환경에 적합한 적정 수준의 주석을 사용할 수 있게 지원하는 주석 관리 체계의 필요성을 시사한다.

IEEE 표준 1028을 적용하는 미션 크리티컬 영역의 소프트웨어 개발에서 원시 코드의 검토와 긴밀하게 관련되는 또 다른 공정 활동은 IEEE 표준 1012의 지침에 따라 수행해야 하는 확인 및 검증 활동이다[4]. 이 활동은 원시 코드의 추적성 분석, 평가, 인터페이스 분석, 원시 코드 문서화 평가를 포함하고 있으며, 기술검토(technical review), 검사(inspection), 워크스로우(walk-through) 방법 등이 사용된다. 원시 코드 검토와 확인 및 검증 활동에 가장 직접적으로 사용되는 최소의 소프트웨어 산출물은 원시 코드 자체와 설계명세서라 할 수 있다. 따라서 바람직한 주석관리 체계는 적정 수준의 주석을 쉽게 사용할 수 있고 원시 코드와 직접 관련되는 설계명세서와의 연계성을 갖춘 것이라 할 수 있다.

본 연구는 C 언어로 작성되는 원시 코드의 주석을 효율적으로 작성 및 관리함과 동시에 확인 및 검증 활동을 지원하기 위하여 (그림 2)와 같은 구성을 갖는 주석 관리 모델을 고려하였다. 동 모델은 주석 사용자 모듈, 검토자 모듈, 주석 추출 모듈, 주석 태그 정의 모듈, 추적 연계 모듈, 저장관리 모듈로 구성된다. 기존 사례연구와 구별되는 본 연구의 주요 특징은 다음과 같다.

- IEEE 표준 1028과 IEEE 표준 1012의 공정 활동을 지원하기 위해 요구되는 범주의 XML 태그를 정의한다.
- XML DTD를 사용한 기존 사례연구들의 제약을 극복하기 위하여 XML 스키마[13]를 기반으로 주석 추출 및 관리를 수행한다.
- 주석 추출 처리 결과의 시각적 표시를 위하여 HTML과 CSS(Cascading Style Sheet)를 사용하지 않고 XSL-FO[14]를 사용한다.



(그림 2) 원시 코드 주석관리 모델의 구성

3.2 각 모듈의 기능요건

주석 사용자 모듈은 원시 코드 작성자가 C 언어를 사용하여 원시 코드를 작성하는 과정에서 주석 태그 및 프로젝트 태그 사용법을 참조하면서 주석을 첨가하는 작업을 지원한다. 주석 첨가에 따른 코드 작성자의 작업부담을 경감시키고 사전에 정의된 XML 태그에 따라 올바르게 주석을 사용할 수 있도록 이 모듈은 "태그 보기" 기능을 갖는다. 주석 사용과 관련된 정보는 주석 태그 정의 모듈에 의하여 사전에 생성되어 저장장치에 파일(확장자 : xml, xsl)로 저장되어 있다. 코드 작성자가 "태그 보기" 기능을 호출하면 파일 확장자가 xml과 xsl인 두 개의 파일 사이의 결합이 XSL-FO 기반으로 처리되고, 그 결과로써 태그 사용 방법을 보여주는 파일(확장자 : pdf)이 출력된다. 사용자는 자신이 선호하는 편집기로 주석을 첨가하면서 원시 코드를 작성한다. 작성이 완료된 원시 코드는 컴파일 되며, 오류가 없다면 검토자 모듈에서 사용될 수 있도록 원시 코드가 저장장치에 저장된다.

검토자 모듈은 작성된 원시 코드에 대한 검토와 확인 및 검증 작업 수행을 지원하기 위한 모듈이다. 이 모듈은 주석 사용자가 저장장치에 저장해둔 원시 코드 파일을 판독하여 원시 코드에 대한 검토가 완료된 이후 원시 코드 파일로부터 주석을 추출하기 위하여 주석 추출 모듈을 호출하는 기능을 수행한다. 본 연구에서 주석 사용자 모듈로 주석의 검토를 수행하지 않고 검토자 모듈로 분리하여 수행하는 이유는 미션 크리티컬 영역의 소프트웨어 개발에 요구되는 제3자에 의한 독립검토 요건을 충족하기 위함이다.

주석 추출 모듈은 검토가 완료된 원시 코드 파일을 대상으로 원시 코드로부터 XML 태그로 첨가된 주석을 추출하는 기능을 갖는다. 추출은 원시 코드의 /* 표기와 */ 표기 블록 내부에 삽입된 태그(<tag>와 </tag>의 쌍) 만을 추출하는 방식으로 (그림 3)의 절차로 수행된다. 추출의 최종 결과는 주석 작성을 위하여 정의한 XML 태그들로 구성된 XML 파일이다. 이 파일은 사전에 정의된 XML 스키마 규칙을 준수하는 지에 대한 검증이 수행된 후에 저장관리 모듈에 의하여 저장장치(또는 데이터베이스)에 보관된다. 추출 과정에서 원시 코드를 구성하는 함수들 간의 호출관계, 원시 코드 자신과 다른 원시 코드와의 연계관계, 원시 코드와 설계명세서와의 연계 정보는 추적(tracking)을 위한 별도의 XML 파일로 작성된다. 이는 추후 추적 연계모듈이 사용할 수 있는 자료의 원천이 된다.

추적 연계 모듈은 저장관리 모듈에 의하여 저장된 XML 파일 내의 주석(XML 태그)을 검색하고 검색의 결과를 표시하는 기능을 수행한다. 이를 위하여 이 모델은 XML 파일을 파싱하여 트리 구조로 변환하는 XML 관련 기술인 XSLT (XML Transformation)와 검색의 결과를 시각적 형태로 표시하는 XSL-FO(eXtensible Stylesheet Language - Formatting Objects)을 사용한다. 이 모듈은 주석 추출 모듈이 주석 추출 과정에서 별도로 생산한 추적 파일을 이용하여 원시 코드 또는 원시 코드 내부의 함수에 대한 종속관계를 검사하여 소프트웨어 구조도를 계층구조로 조직하고 표시하는 기능을 갖는다. 또한 원시 코드의 작성 근거인 설계명세서와의 연계(Linking) 정보를 제공한다. 설계명세서 자체가 XML로 작성되어 있다면 이 연계 정보를 이용하여 설계명세서 자체의 내용에 대한 접근도 가능하다.

주석 태그 정의 모듈은 소프트웨어 개발 프로젝트의 실정에 적합한 주석을 XML 태그로 정의하는 기능을 갖는다. 모든 소프트웨어 개발 프로젝트의 환경에 적응할 수 있는 공통되는 주석을 정의하고 원시 코드 작성자에게 이에 맞게 주석을 사용하도록 요구하는 것은 비현실적이다. 프로젝트의 성격에 따라 주석의 사용을 강화하거나 완화시킬 수 있다면 주석 작성에 소요되는 부담의 수준을 조절할 수 있다. 본 연구는 주석 정의와 사용의 지원을 위하여 XML 스키마와 XSL-FO를 사용한다.

```
Input : C source file
Output : XML comment file
open C source file
if (file == empty or EOF) then exit
Begin comment extraction
 call comment tag extraction module(C source file)
 call comment file validation module(comment XML file, XML
      schema)
 call tracking file creation module(comment file)
End comment extraction

Begin comment tag extraction module
 while (source file == EOF)
  if (exists) "/*" and "*/" text block then
    1 : store finding text block to string buffer B
    2 : Begin push operation to find open tag type
       (2-1) scan next letter L from B
       (2-2) if (L == "<") then open_tag = open_tag+L
       (2-3) scan next letter L from B
       (2-4) open_tag = open_tag+L
       (2-5) if (L == ">") then push open_tag to stack
    3 : Begin extracting comment body data
       (3-1) scan next two letters L from B
       (3-2) temp=first letter+second letter
       (3-3) if (temp = "</") then call pop operation
       (3-4) else comment_body=comment_body+temp
       (3-5) repeat step 3-1)~3-4)
    4 : Begin pop operation to close tag type
       (4-1) close_tag=close_tag+temp
       (4-2) scan next letter C from B
       (4-3) if (C is not ">") then close_tag = close_tag+C
             and repeat step (4-2)~(4-3)
       (4-4) if (C == ">") then pop open_tag from stack
       (4-5) extract_result=open_tag+comment_body+close_tag
       (4-6) append extract_result to XML file
  endif
 endwhile
End comment tag extraction module
```

(그림 3) XML 태그 추출 과정

저장관리 모듈은 주석 관리체계를 구성하는 각 모듈이 사용하는 자료(XML 파일, XSL-FO 파일, 원시 코드 자체, XML 스키마 파일 등)를 보관하고 다수의 사용자에 대한 접근통제 기능을 수행한다. XML 기반의 데이터베이스 구축은 본 연구가 추구하는 최종 목표 중의 하나이다.

3.3 XML 태그 정의

좋은 원시 코드가 갖추어야 할 주석의 요건 내용을 축약하면 다음과 같다.

① 원시 코드(또는 모듈)의 목적을 기술하는 목적문
② 인터페이스에 대한 기술(원시 코드 자신을 호출 다른 원시 코드 식별 정보, 모든 인수 에 대한 설명, 원시 코드 자신이 호출하거나 참조하는 모든 종속 모듈 목록 등)
③ 중요 변수들과 이들의 사용한계, 제한 등을 포함하는 중요 정보들
④ 개발역사(모듈 설계자의 이름, 검토자의 이름과 검토 날짜, 수정일자와 수정에 대한 내용 설명 등)

본 연구는 주석 사용에 대한 일반요건과 (그림 1)의 C 언어 원시 코드 주석 사용 유형을 고려하였으며, 원시 코드 작성자, 유지 보수자(원시 코드 수정자)와 접촉할 수 있는 창구 정보를 포함하는 기본 태그의 유형을 <표 2>와 같이 정의하였다. 코드 작성자의 편의를 위하여 취향에 따라 영문 태그 또는 한글 태그를 선택하여 사용할 수 있도록 정의하였다.

<표 3>은 원시 코드 작성자가 프로젝트와 관련된 참여자 정보를 열람하고 해당 프로젝트에 적용되는 주석 작성 규칙 정보에 접근할 수 있는 경로를 제공하기 위하여 정의한 프로젝트 관련 XML 태그 유형이다.

〈표 3〉 프로젝트 관련 XML 태그의 유형

영문 XML 태그 유형 (한글 태그)	주석 태그의 용도
<ProjectTitle> (<프로젝트이름>)	계약서에 등록된 프로젝트 명칭
<StartDate> (<시작일자>)	프로젝트 시작일자
<EndDate> (<종료일자>)	프로젝트 종료일자
<Worker> (<작업자>)	작업자 정보 입력 영역 표시
<Name> (<이름>)	프로젝트 참여자 이름
<Position> (<직위>)	참여자의 직위 명칭
<Task> (<직무>)	프로젝트에서의 역할/임무
<XFileUsage> (<주석파일이름>)	프로젝트에 적용되는 주석 파일 이름
<SuorceFile> (<원시 코드>)	주석관리체계에 등록된 원시 코드 파일들의 이름

〈표 2〉 주석 작성용 기본태그의 유형

영문 XML 태그 유형 (한글 태그)	주석 태그의 용도
<Copyright> (<저작권>)	원시 코드에 대한 저작권 설명
<MainBlock>	main() 블록의 영역 표시
<BeforeBlock>	main() 블록 이전 영역 표시
<AfterBlock>	main() 블록 이후 영역 표시
<Author> (<작성자>)	원시 코드 작성자 또는 책임을 지는 사람의 이름
<Reviewer> (<검토자>)	원시 코드 검토자, 확인 및 검증 수행자
<Name> (<이름>)	작업 수행자의 이름
<Date> (<일자>)	작업 수행 년월일
<Email> (<전자우편>)	작성자와 연락 가능한 전자우편
<Phone> (<전화번호>)	작성자와 연락 가능한 전화번호
<Address> (<주소>)	작성자와 연락 가능한 주소
<FunctionBlock>	함수의 영역을 표시
<FuncionName> (<함수이름>)	원시 코드에 사용된 함수의 이름
<Function> (<기능>)	함수가 수행하는 기능 설명
<ParentName> (<부모이름>)	함수 자신의 부모 함수
<ChildName) (<자식이름>)	함수 자신이 호출하는 다른 함수의 이름
<PassedVar> (<매개변수>)	함수가 전달받는 매개변수 설명
<ReturnVar> (<반환값>)	함수가 반환하는 값 설명
<FreeNote> (<참조사항>)	자유스럽게 내용을 기술함(단, 정의된 태그는 사용 못함)
<RevisionHistory> (<수정이력>)	원시 코드 전체 또는 부분 수정 작업사항 기입
<RevisionContent> (<수정내용>)	원시 코드의 내용 변경 사항을 기입하는 블록
<RevisionNumber> (<수정번호>)	갱신버전의 일련 번호
<DesignSpecID> (<설계명세 식별자>)	원시코드, 함수와 관련되는 설계명세서의 명세부분 식별자 또는 제목

4. XML 스키마 설계

XML 스키마는 다양한 자료 형을 지원하며, 확장 가능한 내용 모델(content model)이고 XML의 요소(element)와 속성(attribute)의 그룹핑(grouping)을 지원한다. 또한 네임스페이스(name space)를 통해 하나의 XML 문서에 다양한 XML 스키마의 적용이 가능하며, 필요할 경우 문서의 구조와 내용의 사용에 대한 적정 수준의 제약을 가할 수 있다[15]. 이러한 XML 스키마의 특징은 너무나 자유스럽게 비정형적으로 작성될 수 있는 원시 코드 주석에 적정 수준의 제약을 부과함으로써 구조적인 형태로 원시 코드의 주석을 관리할 수 있게 한다. XML 스키마 설계를 위하여 (그림 1)의 C 언어로 작성된 원시 코드의 내용을 살펴보면 main() 블록, main() 이전과 이후의 세 개 블록으로 크게 분할할 수 있음을 알 수 있다.

특히 미션 크리티컬 영역에 많이 사용되는 ANCI C에서는 모든 함수는 호출되기 전에 함수형명, 함수명 및 인수를 선언하는 것이 원칙이며, 함수를 구성하는 실제의 코드는 main() 블록 이후에 작성되므로 원시 코드 차체를 세 개의 영역으로의 분할하는 데에는 무리가 없다. 세 개의 블록 중 main() 이전의 블록에서는 저작권, 작성자, 검토자, 수정이력, 변수 설명, 함수 설명, 기타의 설명 등이 나타날 수 있다. C 원시 코드에서 main()은 오직 하나만이 나타날 수 있으며, main() 블록 내부에는 main() 블록의 외부에 선언되고 작성된 함수의 사용(호출)이 주류를 이룬다. 이와 같은 현실세계를 반영하여 본 연구는 XML 스키마를 (그림 4)와 같이 설계하였다.

본 연구는 XML 스키마를 설계함에 있어서 주석 관리의 편의성과 원시 코드 작성자의 부담이라는 상호 배타적인 속성을 조화시키는데 역점을 두었다. 이를 위하여 단순 모델(simple model)을 구성하는 sequence 요소의 최대 및 최소 출현 회수(minOccurs, maxOccurs)를 조정하였고, 원시 코드의 각 블록에서 자주 반복되거나 공통적으로 사용될 수 있는 XML 태그는 복합 모델(complex model)을 사용함과 동시에 그룹으로 정의하여 이들의 최대 및 최소 출현회수를 조정하였다.

```
<xs : element name = "주소" type = "xs:string" minOccurs="0" maxOccurs="1"/>
    </xs : choice> </xs : sequence>
</xs : group>
<xs : complexType name = "ReviewerDetails">
    <xs : sequence> <xs : group ref = "NameDate"/> <xs : group ref =
     "ContactPoint"/> </xs : sequence>
  </xs : complexType>
  <xs : complexType name = "OutsideDetails">
    <xs : sequence minOccurs = "0" maxOccurs = "unbounded">
      <xs : group ref = "FreeNoteDetails" minOccurs = "0" maxOccurs = "unbounded"/>
        <xs : choice><xs : element name = "RevisionHistory"
                type = "RevisionDetails" minOccurs = "0"
maxOccurs = "unbounded"/><xs : element name="수정이력" type="RevisionDetails"
          minOccurs = "0"
maxOccurs = "unbounded"/> </xs : choice>
      <xs : group ref="ContactPoint"/>
      <xs : element name="FunctionBlock" type="FunctionDetails" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs : sequence>
  </xs : complexType>
  <xs : group name = "RevisionDetails">
      <xs : sequence> <xs : group ref = "FreeNoteDetails" minOccurs = "0"
                maxOccurs = "unbounded"/>
        <xs : group ref = "NameDate"/> <xs : choice><xs : element name =
"RevisionNumber" type = "xs : string"/> <xs : element name = "수정번호" type =
"xs : string"/> </xs : choice> <xs : choice> <xs : element name = "RevisionContent"
type = "xs : string"/> <xs : element name = "수정내용" type="xs : string"/>
</xs : choice>
    </xs : sequence>
  </xs : group>
  <xs : group name = "FunctionDetails">
    <xs : sequence>
      <xs : group ref = "FreeNoteDetails" minOccurs = "0" maxOccurs =
"unbounded"/>
      <xs : choice><xs : element name = "FunctionName" type = "xs : string"
minOccurs = "1" maxOccurs = "1"/>
      <xs : element name = "함수이름" type = "xs : string" minOccurs = "1"
maxOccurs = "1"/> </xs : choice>
      <xs : choice> <xs : element name = "DesignSpecID" type = "xs : string"
minOccurs = "1" maxOccurs = "1"/> <xs : element name = "설계명세식별자" type =
"xs : string" minOccurs = "1" maxOccurs = "1"/> </xs : choice>
      <xs : choice> <xs : element name = "Function" type = "xs : string"
minOccurs = "1" maxOccurs = "1"/> <xs : element name="기능" type = "xs : string"
minOccurs = "1" maxOccurs = "1"/> </xs : choice>
      <xs : choice> <xs : element name = "ParentName" type = "xs : string"
minOccurs = "1" maxOccurs = "1"/> <xs : element name = "부모이름" type = "xs :
string" minOccurs="1" maxOccurs="1"/> </xs : choice>
      <xs : choice> <xs : element name = "ChildName" type = "xs : string"
minOccurs = "1" maxOccurs = "unbounded"/> <xs : element name = "자식이름"
type = "xs : string" minOccurs="1" maxOccurs="unbounded"/> </xs : choice>
      <xs : choice> <xs : element name = "PassVar" type = "xs : string"
minOccurs = "0" maxOccurs = "unbounded"/> <xs : element name = "매개변수"
type = "xs : string" minOccurs = "0" maxOccurs = "unbounded"/> </xs : choice>
      <xs : choice> <xs : element name = "ReturnVar" type = "xs : string"
minOccurs = "0" maxOccurs = "unbounded"/> <xs : element name = "반환값"
type = "xs : string" minOccurs = "0" maxOccurs = "unbounded"/> </xs : choice>
    </xs : sequence>
    </xs : group> <xs : complexType name = "MainDetails"> <xs::sequence
minOccurs = "0" maxOccurs = "unbounded"> <xs : group ref = "FunctionDetails"/>
</xs : sequence>
  </xs : complexType>
</xs : schema>
<?xml version = "1.0" encoding = "EUC-KR"?>
<xs : schema xmlns : xs = "http://www.w3.org/2001/XMLSchema">
  <xs : element name = "Comment">
    <xs : complexType>
      <xs : sequence> <xs : element name = "CommentRoot"
        type = "CommentDetails"/> </xs : sequence>
    </xs : complexType>
  </xs : element>
  <xs : complexType name = "CommentDetails">
    <xs : sequence>
      <xs : group ref = "FreeNoteDetails" minOccurs = "0" maxOccurs =
        "unbounded"/>
      <xs : choice>
```

```
            <xs : element name = "Copyright" type = "xs : string" minOccurs = "0"
             maxOccurs = "1"/>
            <xs : element name = "저작권" type = "xs : string" minOccurs = "0"
             maxOccurs = "1"/>
        </xs : choice>
        <xs : group ref = "FreeNoteDetails" minOccurs = "0" maxOccurs ="unbounded"/>
        <xs : choice>
           <xs : element name = "Author" type = "AuthorDetails" minOccurs = "0"
            maxOccurs = "unbounded"/>
           <xs : element name = "작성자" type = "AuthorDetails" minOccurs = "0"
            maxOccurs = "unbounded"/>
        </xs : choice>
        <xs : group ref = "FreeNoteDetails" minOccurs = "0" maxOccurs =
             "unbounded"/>
        <xs : choice>
           <xs : element name = "Reviewer" type = "ReviewerDetails"
            minOccurs = "0" maxOccurs = "unbounded"/>
           <xs : element name = "검토자" type = "ReviewerDetails" minOccurs = "0"
            maxOccurs = "unbounded"/>
        </xs : choice>
           <xs : element name = "BeforeBlock" type = "OutsideDetails" minOccurs = "0"
maxOccurs = "unbounded"/>
           <xs : element name = "MainBlock" type = "MainDetails" minOccurs = "1"
            maxOccurs = "1"/>
           <xs : element name = "AfterBlock" type = "OutsideDetails" minOccurs = "0"
            maxOccurs = "unbounded"/>
         </xs : sequence>
  </xs : complexType>
  <xs : group name = "FreeNoteDetails">
    <xs : sequence>
      <xs : choice>
        <xs : element name = "FreeNote" type = "xs : string"/><xs : element name
          = "참고사항" type = "xs : string"/>
        </xs : choice>
     </xs : sequence>
  </xs : group>
  <xs : complexType name = "AuthorDetails">
      <xs : sequence> <xs : group ref = "NameDate"/> <xs : group ref =
       "ContactPoint"/> </xs : sequence>
  </xs : complexType>
  <xs : group name = "NameDate">
      <xs : sequence>
        <xs : choice>
              <xs : element name = "Name" type = "xs : string"/><xs : element
               name = "이름" type = "xs : string"/>
        </xs : choice>
        <xs : choice>
            <xs : element name = "Date" type = "xs : string"/> <xs : element
             name = "일자" type = "xs : string"/>
          </xs : choice>
      </xs : sequence>
  </xs : group>
  <xs : group name = "ContactPoint">
  <xs : sequence minOccurs = "0" maxOccurs = "1">
  <xs : choice> <xs : element name = "Email" type = "xs;string" minOccurs = "0"
maxOccurs = "1"/> <xs : element name = "전자우편" type = "xs ; string" minOccurs
 = "0" maxOccurs = "1"/> </xs : choice><xs : choice> <xs : element name = "Phone"
type = "xs ; string" minOccurs = "0" maxOccurs = "1"/> <xs : element name = "전화
번호" type = "xs ; string" minOccurs = "0" maxOccurs = "1"/> </xs : choice>
  <xs : choice><xs : element name = "Address" type = "xs ; string" minOccurs = "0"
   maxOccurs = "1"/>
```

(그림 4) XML 스키마 설계

5. Prototype 구현 결과

사용자가 주석 사용자 모듈의 기능인 태그 보기 기능을 사용하여 원시 코드에 <표 2>에서 정의한 XML 태그를 올바르고 쉽게 삽입할 수 있음을 보이기 위하여 (그림 5)와 같은 XSL-FO 프로그램을 작성하였다. 이 프로그램은 주석 사용을 위하여 정의한 XML 태그의 유형과 용도를 원시 코드 작성자에게 제공하기 위한 것이다. XML 문서와 XSL을 사용한 응용 프로그램을 작성하여 주석 사용법을 원시 코드 작성자에게 제공할 수도 있으나 구현의 단순화를 위하여 직접 XSL-FO 프로그램을 작성하였다. 작성된 프로그램은 (그림 4)에서 설계한 XML 스키마를 준수하는지 검증된 후에 저장장치에 저장된다. 주석 사용자가 원시 코드를 작성하는 과정에서 XSL-FO 실행 지원도구를 사용하여 (그림 5)의 프로그램을 실행시키면 태그 사용법 표시처리가 수행되고, 수행의 결과로써 (그림 6)과 같은 PDF 파일 형식으로 주석 사용법 정보를 획득할 수 있다. XSL-FO 실행 지원도구로는 Antenna XSL Formatter V2를 사용하였다[17].

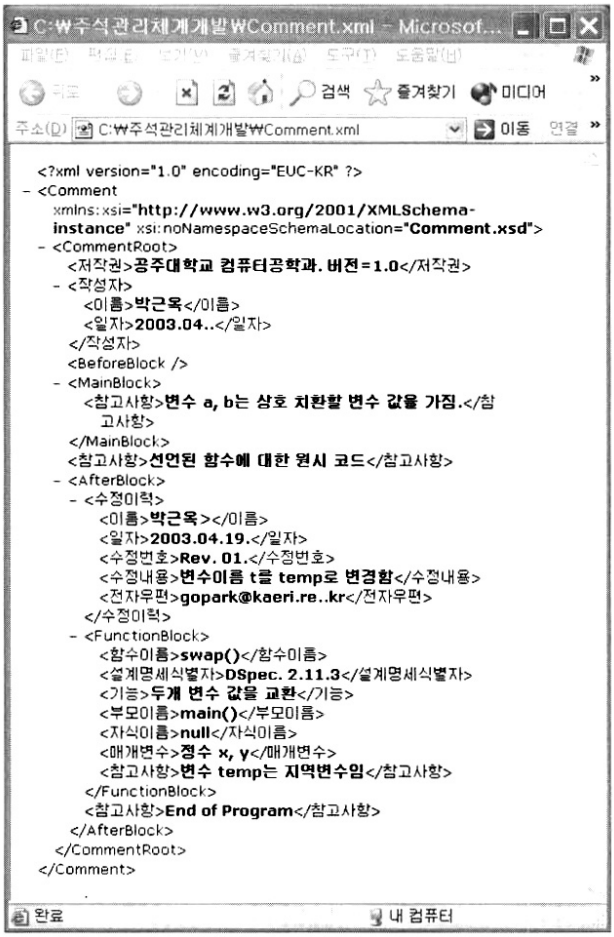
(그림 7)은 원시 코드 작성자가 (그림 6)과 같은 태그 사용법 정보를 활용하여 작성한 C 원시 코드의 일례이다. (그림 8)은 검토자 모듈에서 검토자가 주석 추출 모듈을 호출한 결과로써 생성된 XML 주석 문서이다

(그림 5) XSL-FO 프로그램(일부)

(그림 6) XML 주석 태그 사용법

```
/*<저작권>공주대학교 컴퓨터공학과. 버전=1.0</저작권> <작성자
><이름>박근옥</이름><일자>2003.04..</일자> </작성자>*/
/*<BeforeBlock>*/
#include <stdio.h>
void swap(int x, int y);
/*</BeforeBlock>*/
/*<MainBlock>*/
main()
{ /*<참고사항>변수 a, b는 상호 치환할 변수 값을 가짐.</참고사항
>*/
 int a,b ;
 a=10 ;  b=20 ;
 swap(a, b) ;
 printf("a = %d, b = %d\n", a, b) ;
} /*</MainBlock>
/*<참고사항>선언된 함수에 대한 원시 코드</참고사항>*/
/*<AfterBlock>
/*<수정이력>  <이름>박근옥></이름><일자>2003.04.19.</일자>
<수정번호>Rev. 01.</수정번호> <수정내용>변수이름 t를 temp로
변경함</수정내용>  <전자우편>gopark@kaeri.re..kr</전자우편>
</수정이력>*/
/*<FunctionBlock>
<함수이름>swap()</함수이름> <설계명세식별자>DSpec. 2.11.3</
설계명세식별자> <기능>두개 변수 값을 교환</기능> <부모이름>
main()</부모이름> <자식이름>null</자식이름><매개변수>정수
x, y </매개변수> */
void swap(int x, int y)
{
  /*<참고사항> 변수 temp는 지역변수임</참고사항>*/
  int temp ; temp = x ; x = y ;  y = temp ;
} /*</FunctionBlock>*/
/*<참고사항>End of Program</참고사항></AfterBlock>*/
```

(그림 7) 태그를 사용한 C 원시 코드 작성 일례



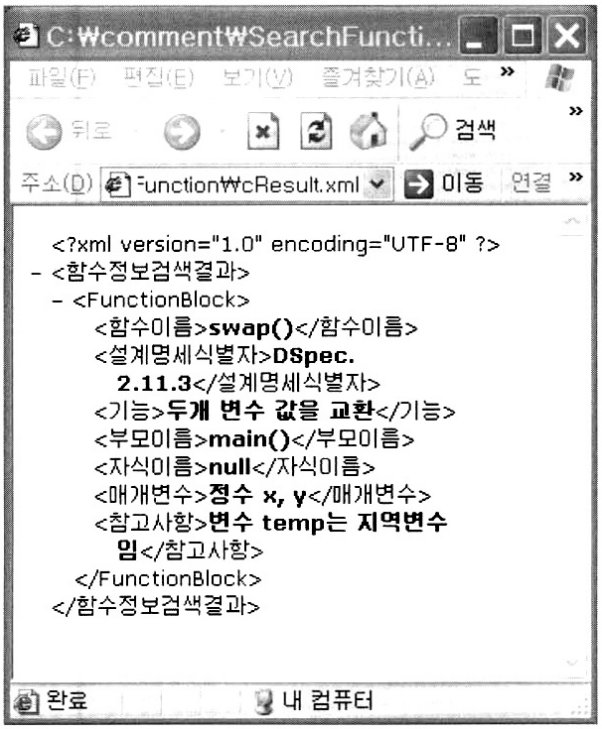
(그림 8) 주석 추출 모듈이 생성한 XML 주석 문서

주석 추출 모듈은 원시 코드로부터 XML 주석 문서를 생성하기 위하여 원시 코드 작성자가 추가하지 않은 몇 가지 요소, 즉 XML의 버전과 인코딩(encoding), XML 네임스페이스 및 XML 스키마, 루트 요소(이름 : CommentRoot) 정보를 추가하여 XML 문서를 생산한다. 생산된 XML 문서는 설계한 (그림 4)의 XML 스키마를 따르는 유효한 문서인지 검사과정을 거친다. (그림 8)의 결과는 주석 관리체계의 각 모듈이 사용할 수 있도록 저장장치에 저장된다.

(그림 9)는 검색자가 입력한 XML 주석 키워드 "FunctionBlock"을 추적 연계 모듈이 입력 받아 처리한 출력 결과이다. 구현의 단순화를 위하여 본 연구는 XSLT가 XML 주석 문서와 미리 작성하여 저장한 Stylesheet 내용을 담은 XSL 문서를 판독하여 변환(transformation)을 수행하도록 프로그램을 작성하였다. 변환 수행을 위한 XSLT로 Xalan-Java2를 사용하였다[18].

주석 추출 모듈 또는 추적 연계 모듈의 처리결과로 얻게 되는 새로운 XML 문서가 비록 브라우저로 그 내용을 확인할 수 있는 (그림 9)와 같은 구조화된 형태라 할지라도 사용자에게 친숙한 형태는 아니다. 이는 XSLT로 처리한 결과를 모양새 있게 브라우저 화면에 출력하기 위하여 HTML과

더불어 CSS(Cascading Style Sheet)가 사용되는 한에는 피할 수 없는 한계이다. 이 한계점은 CSS 자체가 XML 규칙을 따르지 않으며 오직 스타일만을 지정한다는 점에서 유래한다. 이러한 한계는 본 연구에서 XSL-FO를 사용하여 제시하는 태그 사용법 표시 처리와 같은 방법을 사용함으로써 해결될 수 있다.



(그림 9) FunctionBlock 태그 검색결과

6. 결론 및 향후 연구계획

XML은 규칙이 지나치게 복잡하여 전문가가 아니면 취급하기 어려운 SGML(Standard Generalized Markup Language)의 문제점을 해소하고, 단순하여 사용하기 쉬우나 문서의 내용과 구조의 취급에 한계를 갖는 HTML의 문제를 극복하기 위한 노력의 결과로 탄생하였다. 이 XML은 각 산업분야에서의 활용이 급속히 확대되고 있다. XML과 관련된 권고안의 제정과 권고안을 수용하는 구현기술과 도구 또한 성숙도를 더해가고 있다. 이러한 추세를 반영하여 본 연구는 미션 크리티컬 영역에 많이 사용되는 C 언어 원시 코드에 XML 태그를 사용하여 주석을 관리하는 방법을 모색하였다.

주석의 관리를 위하여 본 논문은 총 6개 모듈로 구성되는 관리 모델을 제시하였고 각 모듈이 갖는 기능을 설명하였다. 또한 주석 작성에 사용할 XML 태그를 정의하였고, 정의한 태그가 원시 코드에서 올바르게 사용되었음을 검증하는 XML 스키마를 설계하였다.

본 논문은 주석 관리체계 개발을 위하여 기술적 타당성 확인이 선행되어야 할 일부 기능을 구현하였다. 구현결과, 정의한 XML 태그가 첨가된 C 언어 원시 코드로부터 XML 스키마와의 유효성을 유지하는 XML 주석 문서를 얻을 수 있었다. 원시 코드 작성자가 XML 태그의 사용을 쉽고 정확하게 수행할 수 있도록 지원하기 위한 기능 또한 XSL-FO로 구현하여 확인하였다.

향후 경쟁력을 갖는 주석관리 체계로의 완성을 위하여 본 연구가 중점을 두어야할 방향은 세 가지로 구분할 수 있다. 첫째는 웹 환경에서 주석 관리체계 사용자와의 상호작용성 향상이다. 이를 위하여 웹 사용자로부터의 입력 요구는 현재 권고안이 진행 중인 XForms를 적용할 계획이다. 둘째는 원시 코드로부터 추출된 주석 정보와 소프트웨어 생명주기와의 산출물 연동체계 구축이다. 이를 위하여 현재 본 논문에서 정의한 XML 태그 중 함수이름, 부모이름, 자식이름, 설계명세 식별자 태그를 활용할 것이다. 특히 부모이름과 자식이름의 태그를 적절히 활용하면 프로젝트에서 개발되는 전체 소프트웨어의 계층구조를 생성할 수 있을 것으로 기대한다. 셋째는 총 6개의 모듈이 정보자원을 효율적으로 공유할 수 있는 XML 데이터베이스의 구축이다. 원시코드, XML 주석 문서, XML 스키마, XSF-FO 문서, 소프트웨어 생명주기 산출물, 프로젝트 관리 정보 등 다수 유형의 자원의 효율적 관리와 통합을 위하여 데이터베이스의 존재는 필수적이다.

참 고 문 헌

[1] 윤 청, 성공적인 소프트웨어 개발방법론(하권), 생능출판사, 1998. 6. 30.

[2] R. S. Pressman, Software Engineering - A Practitioner's Approach, McGraw-Hill 4th Edition, 1998.

[3] IEEE Standard 1028-1997, Software Reviews.

[4] IEEE Standard 1012-1986, Software Verification and Validation Plans.

[5] N. Marovac, "Guidelines for Embedded Software Documentation," ACM SIGSOFT, Software Engineering Notes, Vol.19, No.2, pp.22-28, 1994.

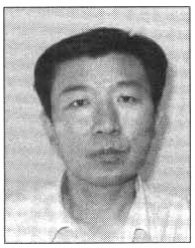
[6] 문양선, 장근실, 유철중, 장옥배, "C++ 프로그램의 유지보수 지원 시스템 개발", 정보처리논문지, 제5권 제7호, pp. 1759-1773, 1998.

[7] G. Antonioi and G. Canfora, "Recovering Traceability Links between Code and Documentation," IEEE Trans. on Software Engineering, Vol.28, No.10, pp.970-983.

[8] D. Bargeron, A. Gupta and A. Brush, "A common annotation framework," MSR-TR-2001-108, Microsoft, 2001.

[9] 조현훈, 최용락, 류성열, "McCabe 및 BP/Win 도구를 이용한 소프트웨어 역공학 사례연구", 한국정보과학회논문지, 제6권 제5호, pp.528-535, 2000.

[10] 조현훈, 류성열, "소프트웨어 재사용을 위한 역공학 기반의 재문서화 방법 및 프로세스", 한국컴퓨터산업교육회지 논문지, Vol.3, No.6, pp.727-738, 2002.
[11] 장근실, 유철중, 장옥배, "JML을 이용한 Java 원시 코드의 역공학/순공학적 접근", 한국정보과학회논문지, 제30권 제1호, pp.19-30, 2003.
[12] J. Sharp and J. Jagger, Microsoft Visual C#.NET, Microsoft Press, 2002.
[13] W3C, XML Recommendation, http://www.w3.org/XML/Schema.
[14] W3C, XSL Recommendation, http://www.w3.org/TR/XSL/.
[15] Jon Duckett, Professional XML Schemas, Wrox Press Ltd., 2002.
[16] Eric, V. D. V., "Comparing XML Schema Languages," XML.com, Dec., 2001.
[17] http://www.antennahouse.com/.
[18] http://xml.apache.org/dist/xalan-j/.
[19] W3C, XML Recommendation, http://www.w3.org/MarkUp/Forms/.



박 근 옥

e-mail : gopark@kougju.ac.kr
1983년 서울산업대학교 전자계산학과 (학사)
1993년 충남대학교 전자계산학과(석사)
2003년~현재 공주대학교 컴퓨터공학과 박사과정
관심분야 : 정보검색, 시뮬레이션

임 종 태

e-mail : jtlim@kongju.ac.kr
1985년 전남대학교 계산통계학과(학사)
1987년 한국과학기술원 전산학과 (공학석사)
1992년 한국과학기술원 전산학과 (공학박사)
1993년~현재 공주대학교 컴퓨터공학과 부교수
관심분야 : 정보검색, 시뮬레이션, 바이오인포매틱스