

웹 어플리케이션의 모듈위험수준 측정 도구의 구현

김지현*, 박철**

Implementation of the Module Risk Levels Measurement Tools for Web Application

Jee-Hyun Kim*, Chel Park**

요약

웹 어플리케이션의 작성에 많이 사용되는 ASP로 개발된 프로젝트에서 각 모듈의 위험수준을 분석할 수 있는 측정 도구를 개발하고 이 측정 도구를 실무에서 사용 중인 프로젝트에 적용하여 유용성을 보인다. 본 논문에서 개발된 측정 도구는 NASA의 GSFC에서 개발한 소프트웨어 산출물 품질 메트릭인 모듈위험수준 인디케이터(Indicator of the Module Risk Levels)를 사용한 것으로, 웹 어플리케이션의 위험수준을 파악하여 오류 가능성이 있는 위험한 모듈을 조기에 발견함으로써 유지보수성 향상에 기여하고자 한다.

Abstract

This paper implemented the tools to measure the risk of the modules for the web application project written in ASP, using the Indicator of the Module Risk Levels. The Indicator of that is developed by GSFC group in NASA based on structural programming language and gives us software product quality metrics. The implemented tools examined with the ASP projects of the practical business. As the results the data shows the module risk levels easily, and then the data affects the maintenance to improve the application quality.

* 서울대학교 전기전자컴퓨터공학계열 소프트웨어전공 전임강사
** 단국대학교 정보컴퓨터학부 컴퓨터과학전공 강의전임강사

I. 서론

WWW(World Wide Web)이 인터넷의 글로벌 통신 매체로 자리하면서, 끊임없이 증가하고 있는 다양한 어플리케이션의 프론트 엔드(front-end) 프로그램으로 자리 잡게 되었다. 여기에 많은 기업들이 단지 경쟁사와 겨루기 위해 인터넷 기술의 도입이나 새로운 웹 사이트 개발을 너무 빨리 진행하게 되었다.[1]

이렇게 웹은 광고, 영업, 출판, 교육, 여가생활 등 산업 전반의 새로운 매체로 주요한 위치를 차지하게 되었는데, 웹 사이트의 기본적인 구조에 대한 고려 없이 시각적인 측면을 위주로 개발되어 왔으며, 짧은 역사로 숙련되지 않은 개발자들에 의해 만들어지게 되었다.

또한 웹에 대한 시장의 요구가 다양해짐에 따라 웹 어플리케이션의 실패도 늘어나고 있는데, 웹 어플리케이션에 대한 정형화된 테스트 방법론도 없고, 표준화된 프로세스가 없으므로 생산성도 떨어지게 되었다. 웹 사이트는 조잡한 설계나 기초 기능성의 부족과 함께, 사이트 개발 시의 품질 보증활동도 거의 이루어지지 않고 있어 유지보수 문제와 개발 비용 과다라는 심각한 문제를 안게 되었다. 이렇게 웹 어플리케이션에 대한 품질의 중요성이 커짐에 따라 품질에 관한 활동 및 평가는 웹 어플리케이션의 공학적 접근이슈 중 가장 중요한 연구 대상의 하나가 되고 있다.

그러나 품질은 사용자 마음에 있는 것이다. 모든 사람들이 품질의 중요성에는 동의하지만 무엇이 품질이고, 어떻게 그것을 측정할 것 인지에는 이견이 많다. Kitchenham은 품질에 대하여 지적하기를 '품질은 정의하기가 어렵고, 측정이 불가능하며, 인식하기는 쉽다.(quality is hard to define, impossible to measure, easy to recognize)'라고 하였다.[16] 웹 사이트의 품질을 측정하지 않고, 웹 사이트의 발전을 기대하기는 어려운 것이다.

본 연구에서는 소프트웨어 품질 메트릭인 모듈위험수준 인디케이터(Indicator of the Module Risk Levels)[16]를 ASP로 작성된 웹 어플리케이션에 적용하여 모듈위험수준을 평가함으로써, 웹 어플리케이션의 유지보수성 향상

에 기여하고자 한다.

모듈위험수준 인디케이터는 소프트웨어 산출물의 유지보수성을 평가하는 NASA의 GSFC(Goddard Space Flight Center)에서 개발한 템플릿으로 프로그램 규모(size, LOC)와 순환 복잡도(Cyclomatic Complexity)의 상관관계에 의하여 산출된다.

본 논문의 구성은 2장에서 웹 어플리케이션 품질 메트릭으로서의 모듈위험수준 인디케이터에 대한 관련 연구를 하였고, 3장에서 모듈위험수준 인디케이터를 구성하는 LOC와 순환 복잡도를 측정하여 모듈위험수준 측정 도구를 설계 구현하였으며, 4장에서 구현한 도구를 실제 기업에서 사용 중인 ASP 웹 어플리케이션 프로젝트에 적용하여 비교 분석함으로써 모듈위험수준 인디케이터가 웹 어플리케이션의 품질 측정에 유용함을 제시하고자 한다.

II. 관련 연구

1. 순환 복잡도(Cyclomatic Complexity)

본 절에서는 모듈위험수준 인디케이터를 구성하는 순환복잡도에 대해 알아본다.

McCabe는 소프트웨어의 논리적인 복잡도를 정량적으로 측정하는 메트릭으로 순환 복잡도를 제안하였다. 순환복잡도는 프로그램의 논리 흐름을 표현하는 제어흐름 그래프를 기초로 하는 메트릭으로 구조적 프로그램에서 제어흐름의 복잡도를 효과적으로 측정한다.[7]

프로그램의 제어 이동을 나타내는 제어흐름 그래프에서 노드는 제어의 분기점을 의미하고 간선은 연속으로 실행되는 일련의 프로그램코드를 말한다.

선택 경로와 루프가 많다는 것은 그만큼 프로그램이 복잡하다는 것을 의미하므로 순환복잡도 $V(G)$ 는 프로그램의 제어흐름 그래프에 포함된 선형 독립 경로(linearly independent path)의 수로 결정된다. 선형 독립 경로는 프로그램에서 실행 가능한 경우의 수가 된다. [그림 1]에서 선형 독립 경로는 4가지가 있으며 이 프로그램의 순환 복잡도 $V(G)$ 는 4가 된다.

프로그램 구조

```

<1번. 순차코드들>
for(...) {
  <2번. 순차코드들>
  if(...) {
    <3번. 순차코드들>
  }
  else {
    <4번. 순차코드들>
    if(...) {
      <5번. 순차코드들>
    }
    else {
      <6번. 순차코드들>
    }
    <7번. 순차코드들>
  }
  <8번. 순차코드들>
}
<9번. 순차코드들>
    
```

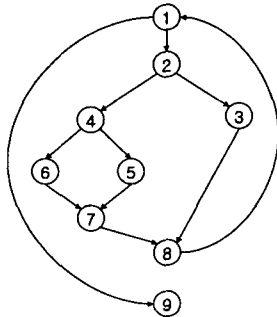


그림 1. 프로그램 구조와 제어흐름 그래프

제어흐름 그래프의 선형 독립 경로의 수는 간선의 수 e, 노드의 수 n, 그리고 연결 요소(connected component)의 수 p를 사용하여 식(1)과 같이 계산될 수 있다.

$$V(G) = e - n + 2p(1)$$

다른 방법으로 제어흐름 그래프 G에 포함된 제어 및 반복 노드들의 개수(DE)를 사용하면

$$V(G) = DE(2)$$

이다. DE는 프로그램의 흐름을 변경시킬 수 있는 프로그램 명령어를 의미하며 IF, SELECT 문등의 분기 명령과 FOR, WHILE, LOOP 등의 반복 명령어를 의미한다.

2. 모듈위험수준 인디케이터

모듈위험수준 인디케이터는 소프트웨어 품질 메트릭으로 유지보수성을 평가한다.

프로그램 유지보수란 오류를 찾아서 수정하거나, 변경 요구에 대하여 기존의 소스 코드를 수정하는 것을 말한다. 두 경우 모두 소스코드를 통해 어떤 작업이 어디에서 행해지고 값의 추적이 어디에서 가능한 지 그 위치를 찾아내는 능력을 요구한다. 규모도 작고, 복잡도도 낮으며 주석이 많은 소스 프로그램이 일반적으로 변경하기 쉽다. 모듈들 사이에 교환되는 변수의 양이나 형태(fan in/fan out)도 유지보수에 영향을 미친다.[16]

[그림2]는 모듈위험수준 인디케이터로서, GSFC에서 개발한 유지보수성 측정에 사용하기 위한 그래프 템플릿

이다. 즉 오류의 위험 가능성이 높은 모듈이 어떤 모듈인지를 알려줌으로서 유지보수를 도와주는 척도가 되는 그래프인 것이다. X축은 수행 가능한 코드 라인의 수를 나타내고 Y축은 모듈의 확장된 순환 복잡도(시험 경로의 수)를 나타낸다.

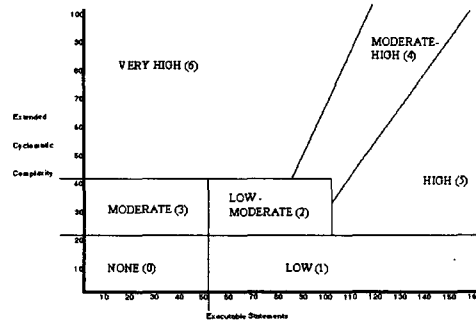


그림 2. 모듈위험수준 인디케이터

위의 그래프를 보면 위험수준을 0에서 6까지 7개의 영역으로 나누고 영역0(없음), 영역1(하), 영역2(하중), 영역3(중), 영역4(중상), 영역5(상), 영역6(최상)으로 명명하였다. 즉, [그림 2]에서 각 영역을 구분하는 가이드라인들 - X축인 LOC의 경우 50과 100, Y축인 순환 복잡도의 경우 20과 40, 그리고 오른쪽 위로 증가하는 두개의 사선(영역4, 영역5, 영역6을 구분) - 은 NASA와 여러 기업의 소스 프로그램으로부터 GSFC에 의해 여러와의 상관관계에서 추출한 가이드 라인에 근거한 것이다.

프로그램 규모와 복잡도 값이 큰 영역4(중상), 영역5(상), 영역6(최상)에 분포한 모듈은 위험수준이 높은 모듈로서, 심각한 에러가 발생할 확률이 높고 향후 유지보수에 문제가 생길 수 있다는 것을 통계에 의하여 추출하였다. 그러므로 이 영역에 분포하는 모듈은 특별 관리가 필요하며, 개발자들은 이 영역에 분포한 모듈의 팬 인/팬 아웃, 주석의 퍼센티지, 에러의 수에 대해 더욱 조사해야 한다고 분석하였다.

또한 영역0(없음), 영역1(하), 영역2(하중), 영역3(중)에 분포한 모듈은 소스 코드의 위험요소가 크지 않아 안정성이 좋고 따라서 수용 가능한 위험수준 영역에 있다고 하였다. SATC 자료에 의하면 C나 C++로 개발한 프로그램이 FORTRAN으로 개발한 프로그램보다 규모나 복잡도에서 비교적 낮은 수치를 보여, 위험 영역인

영역4(중상), 영역5(상), 영역6(최상)에 상대적으로 적은 수의 모듈이 존재한다고 분석하고 있다.

3. 웹 어플리케이션의 품질 메트릭

본 논문은 웹 어플리케이션 언어로서 쉽게 접할 수 있고, Microsoft Windows 환경에서 가장 많이 사용되는 언어 중 하나인 ASP로 작성된 웹 어플리케이션 프로젝트를 실험 대상으로 채택하였다. ASP(Active Server Page)는 동적인 웹 서버 어플리케이션을 생성하는데 사용할 수 있는 서버측 스크립팅 언어로 ASP의 서버 스크립팅은 웹 서비스를 제공하는 웹 서버 컴퓨터 상의 스크립트 엔진에 의하여 스크립트 코드가 해석되고 실행되는 것을 말한다. ASP는 3세대 언어의 모듈 구조를 사용하고, 그 형태가 기존의 구조적 언어의 모듈 형태와 비슷하며 파라미터 형태와 전달 방식도 같다.[17]

웹 사이트의 품질은 전 세계 수많은 동시 사용자를 가질 수 있기 때문에 특히 중요하다. 이러한 웹 사이트의 특수성 때문에 웹 어플리케이션은 소프트웨어 품질의 세계에서는 매우 새로운 것이다. 또 웹의 즉시성은 빠른 어플리케이션의 개발과 즉각적인 품질의 기대를 요구하지만 웹 어플리케이션의 복잡성은 품질 관리를 더욱 어렵게 만든다.[14]

즉, 웹 어플리케이션의 즉시성, 동시성, 광역성, 복잡성 등으로 인해 웹 어플리케이션 품질 관리의 중요성이 가중되지만, 동시에 그러한 특성이 관리를 어렵게 하고 있다. 이에 Powell은 다음과 같은 공학적 해결책을 지적했는데, '소프트웨어 산출물의 품질 속성은 어플리케이션의 구조(structure or architecture)를 분석하여 사용성과 유지보수성의 문제를 파악하고, 그에 따른 오류 모듈이 정의되어야 한다'[19]고 하였다.

물론 웹 사이트의 구조 분석에도 여러 가지 관점이 있다. Ivory는 웹 사이트 설계의 관점을 Information Design, Graphic Design, Navigation Design 등으로 나누고, 구조를 Text Elements와 Formatting, Link Elements와 Formatting, Graphic Elements와 Formatting 및 Page formatting으로 보았다.[13] 또한 Miller는 웹 사이트 구조를 Browser, Display Technology, Navigation, Object Mode, Server Response, Interaction & Feedback 및 동시 사용자 등으로 분석하였다.[14] 황철현은 웹 에플리케이션을 클라이언트, 서버, HTML의 세 가지로 분류하였다.[20]

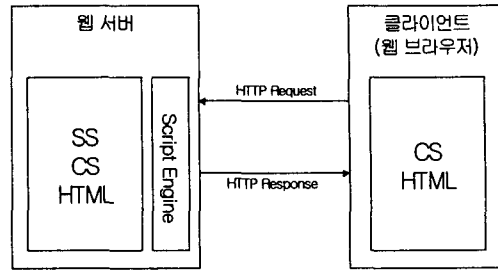


그림 3. 웹 어플리케이션 아키텍처[20]

이렇게 웹 사이트 구조에 대한 이론이 정립되어 있지 않고, 테스트 방법론이나 평가 메트릭 및 웹 사이트에 적합한 새로운 품질 메트릭의 개발은 더 더욱 이루어지지 못하고 있는 실정이다. 이에 본 연구에서는 기존에 이미 구조적 언어와 객체지향 언어의 실험 분석에 의하여 검증된 모듈위험수준 인디케이터를 웹 어플리케이션에 적용하여 실험, 평가함으로써 웹 사이트의 유지보수성을 평가하고자 한다.

III. ASP 모듈위험수준 측정도구

1. 측정 도구의 구성

본 절에서는 모듈위험수준 인디케이터를 사용하여, 그 구성요소인 LOC와 순환복잡도를 ASP 어플리케이션에 적용하여 측정하고, 모듈위험수준을 결정하는 측정도구의 구현을 제시한다. 모듈위험수준 인디케이터는 기존의 구조적 언어로 개발된 프로젝트에서 그 유용성이 검증된 메트릭으로, 새로운 주류인 ASP로 개발된 웹 어플리케이션에 적용해 보는 것은 의미있는 일이다.

또한, 순환복잡도가 논리적인 흐름의 복잡도를 측정하는 것이기 때문에 프로그래밍 언어에 매우 종속적이지는 않고, ASP에서 사용하는 스크립트 언어가 GSFC에서 모듈위험수준 인디케이터 실험에 사용한 FORTRAN, C/C++과 마찬가지로 구조적 언어의 특징들을 가지기 때문에 GSFC의 가이드라인을 그대로 적용하였다.

구현한 측정 도구 시스템의 구성은 [그림 4]와 같이 ASP 소스 파일을 입력하여 파싱 처리 후 순환복잡도를 측정하여 그 결과를 레퍼지토리에 저장한다. 레퍼지토리는

에 저장된 정보를 사용하여 주어진 질의를 통해 추출된 결과를 출력한다.

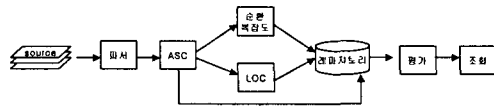


그림 4. 측정도구시스템 구성도

다음의 <표 1>에는 측정도구시스템의 각 구성요소의 역할에 대하여 설명하고 있다.

표 1. 시스템에 사용된 모듈들의 역할

모듈	설명
파서	소스에서 사용되는 토큰을 분리한다. ASP 소스를 구성하는 VBScript, JavaScript, HTML 토큰을 모두 인식하여 분리한다.
ASC	Application Statistics Collector 전체 웹 어플리케이션에 대한 통계 (LOC, 순환복잡도 등)
리포지토리	수집된 정보를 저장하여 평가와 조치 모듈에서 사용한다.
평가	리포지토리의 정보를 사용하여 복잡도를 계산한다.
조치	전체 모듈의 복잡도 목록과 복잡도가 높은 모듈의 목록을 조회한다.

2. 측정 도구의 구현

본 논문에서 사용한 모듈위험수준 측정 도구는 MS Windows 2000에서 MS Visual C++ 6.0을 사용하여 구축하였다. 웹 어플리케이션에서 추출된 데이터를 저장하는 레파지토리는 MS SQL Server 2000을 사용하였다

[그림 5]는 순환복잡도와 LOC를 측정하기 위하여 개발된 도구의 초기 화면을 보여 주고 있는데, 경로 입력 버튼을 클릭하여 웹 어플리케이션이 저장되어 있는 루트 디렉토리의 경로를 선택하여 입력한 후 파싱 버튼을 클릭하면 소스 프로그램에 대한 복잡도와 LOC 측정 결과가 레파지토리에 저장된다.

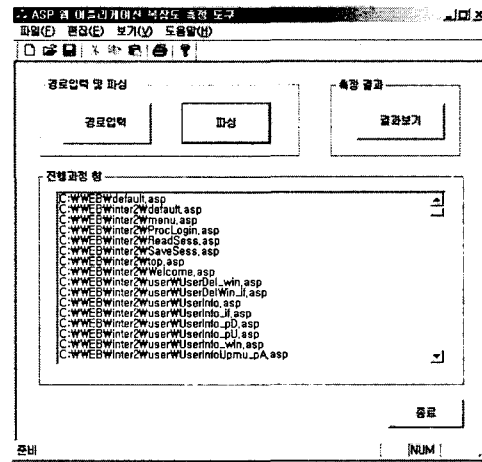


그림 5. 품질 메트릭 측정 도구의 초기화면

파싱 수행 후 결과보기 버튼을 클릭하면 [그림 6]과 같이 ASP 서버 어플리케이션의 해당 경로명과 함께 복잡도 측정결과를 보여준다.

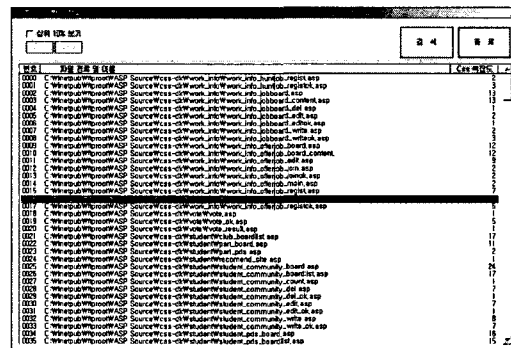


그림 6. 복잡도 측정 도구의 결과 화면

IV. 실험

1. 실험 대상

실험에 사용된 세 개의 프로젝트를 각각 A, B, C라 하면 A는 웹을 사용하여 DBMS를 통합 관리하는 DB 자원관리 시스템으로 사내 인트라넷이며, B는 간단한 상거래 웹 사이트 프로그램, C는 공공목적 웹사이트인 대학의

홈페이지 관리 시스템 등으로 다양한 웹 사이트 시스템이 적용되었다.

먼저 3개의 프로젝트 전체를 통합하여 각 모듈의 LOC와 복잡도를 측정한 후 관련 연구에 제시한 [그림 2]의 모듈위험수준 인디케이터에 적용하여 그 결과를 비교하였고, 다음 단계로 각 서버의 단위 프로젝트 적용 결과를 관찰하여 각 프로젝트의 모듈위험수준을 분석하였다.

다음의 <표 2>는 실험에 사용된 A, B, C 세 프로젝트의 소스 파일(*.asp)의 수를 나타내고 있다.

표 2. 실험에 사용된 웹 어플리케이션 파일 수

	A	B	C
웹 어플리케이션 성격	사내 인트라넷 시스템	상업적 웹 사이트	공공목적 웹 사이트
파일 수	243개	222개	340개

2. 실험 결과

A, B, C 세 프로젝트를 통합한 전체 ASP 파일(805개)에 대해 LOC와 복잡도를 측정하여 모듈위험수준 인디케이터를 적용한 결과 [그림 7]과 같은 분포를 보이고 있다. 즉 관련연구에서 특별관리가 필요하다고 언급되었던 영역4(중상), 영역5(상), 영역6(최상)에 분포한 모듈이 모두 22개로 2.7%의 비율을 보임으로서 ASP 어플리케이션은 복잡도와 LOC에 의한 위험수준비율은 크지 않다고 볼 수 있다.

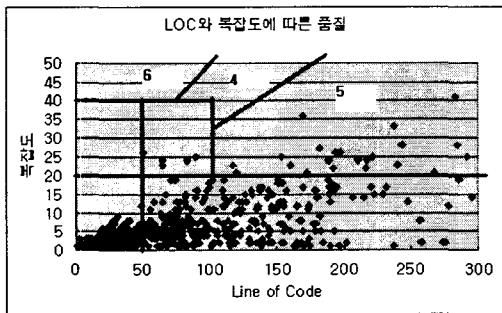


그림 7. A, B, C 세 프로젝트 통합 측정 분포도

다음 단계로 세 프로젝트 각각에 LOC와 복잡도를 계산하여 매트릭을 적용한 결과 다음과 같은 결과를 보였다.

A, B, C 세 프로젝트에서 측정된 LOC와 순환 복잡도에 의한 모듈위험수준은 각각 [그림 8], [그림 9], [그림 10]과 같다.

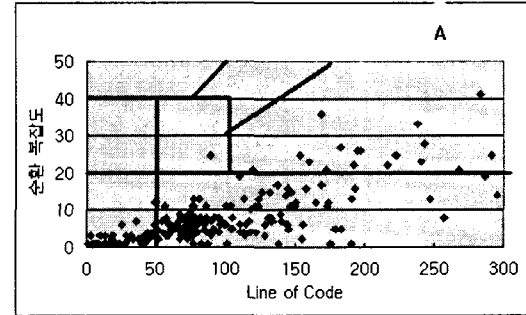


그림 8. A 프로젝트 측정 분포도

[그림 8]에 의하면 A 프로젝트는 특별 관리가 필요한 영역4(중상), 영역5(상), 영역6(최상)중 영역5(상)에 20개의 파일이 있으므로 리스크가 높은 파일이 약 8.2%가 된다.

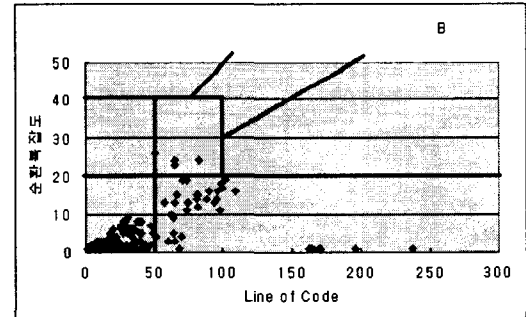


그림 9. B 프로젝트 측정 분포도

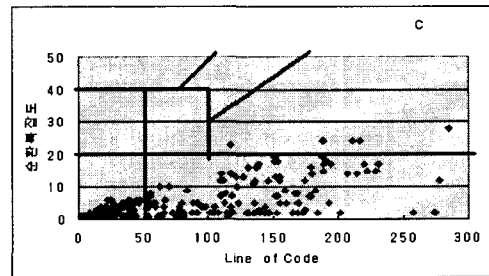


그림 10. C 프로젝트 측정 분포도

[그림 9]에 의한 B프로젝트는 위험수준이 높은 파일이 발견되지 않았으며, [그림10]에서처럼 C프로젝트는 위험수준이 높은 파일이 영역5(상)에 8개로 약 2.3%의

파일이 분포되어 있어, 위험수준은 그리 높지 않다는 것을 알 수 있었다.

즉 A, B, C 세 프로젝트 중에서 인트라넷 시스템이 영역5(상)에 비교적 많은 파일이 있어 모듈 위험 수준이 비교적 높았고, 가장 위험할 것으로 보였던 상거래 시스템이 위험수준이 낮은 것으로 평가되었다.

다음의 <표 3>은 특히 상위 위험수준에 있는 A프로젝트의 5개 파일과 C프로젝트의 2개 파일에 대한 LOC와 복잡도를 보이고 있는데, LOC가 다른 파일보다 현저히 크거나 복잡도가 현저히 큰 파일이 여러 위험 확률이 높았고 품질 및 유지보수성의 저하를 초래할 수 있다는 것이 발견되었다.

표 3. 상위 위험수준 모듈 리스트

파일	LOC	복잡도	리스크 레벨
A1	596	56	5
A2	283	41	5
A3	377	38	5
A4	169	36	5
A5	238	33	5
C1	305	28	5
C2	285	28	5

V. 결론

웹 어플리케이션이 소프트웨어가 가지는 유지보수의 어려움과 비용과다 문제를 고스란히 전수하면서 품질에 심각한 문제점을 드러내게 되었으나, 아직까지 복잡한 웹 사이트의 품질을 측정할 수 있는 새로운 메트릭이 개발되고 있지 못하다. 이에 기존의 소프트웨어 품질 측정 메트릭 중 NASA에서 개발한 모듈위험수준 인디케이터를 웹 서버 어플리케이션인 ASP에 적용하여 유지보수성을 측정하고자 하였다.

모듈위험수준 측정도구를 구현하여 ASP 웹 어플리케이션에 적용한 결과 위험수준 가이드라인에 의한 품질에 이상이 있는 파일을 쉽게 발견하여 모듈위험수준 인디케이터가 ASP 웹 어플리케이션에도 비슷한 결과를 보여줌을 알 수 있었다.

그러나 본 논문에서의 실험은 세 가지 프로젝트를 샘플로 하여 실험한 것이므로 한계가 있다. 향후 연구시 실험 대상 프로젝트를 좀 더 확대하여 보다 많은 프로젝트에 적용함으로써 실험 결과를 일반화 할 필요가 있으며, 향후 모듈위험수준 인디케이터의 가이드 라인이 웹 어플리케이션에서 다른 값을 취할 수도 있다는 것이 제시되어야 할 것이다. 가이드 라인의 지표에 대한 문제는 향후 보다 많은 웹 어플리케이션 프로젝트에 대한 분석을 통해 얻을 수 있을 것으로 본다.

참고문헌

- [1] Boldyreff, Cornelia, Warren, Paul, Gakell, Craig, and Marshall, Angus, "Web-SEM Project: Establishing Effect Web Site Evaluation Metrics", Proceedings of 2nd International Workshop on Web Site Evaluation WSE'2000', p. WSE17, 2000
- [2] Dennis Kafura, Geereddy. R. Reddy, "The Use of Software Complexity Metrics in Software Engineering", IEEE Trans. on Software eng., Vol. SE-13, No. 3, 1987
- [3] A. E. Hatzimanikatis, C. T. Tsalidis and D. Christodoulakis, "Measuring the Readability and Maintainability of Hyperdocuments", Software Maintenance Research and Practice, Vol. 7, 1995
- [4] M. Halstead, "Elements of Software Science", Elsevier North Holland, New York, 1977
- [5] L. J. Artur, "Measuring Programmer Productivity and Software Quality", Jon Wiley ^ Sons, Inc. 1985
- [6] S. Henny and D. Kafura "Software Structure Metrics Based on Information Flow", IEEE Trans. on Software eng., Vol SE-7, No. 5, 1981
- [7] T. J. McCabe, "A Complexity Measure",

IEEE Trans. on Software eng., Vol. 2, No. 4, 1976

[8] Stwphen R. Schach, "Classical and Object-Oriented Software Engineering", McGraw-Hill, 1999

[9] Roger Fournier, "A Methodology for Client/Server and Web Application Development", Yourdon Press Prentice Hall Building, 1998

[10] Linda H. Rosenberg and Lawrence E. Hyatt, "Software Quality Metrics for Object-Oriented Environments", Crosstalk Journal, 1997

[11] W.Harrison, K.Magel, R.Kluczny and A.Decock, "Applying Software Complexity Metrics to Program Maintenance", IEEE Computer, Vol.15, No.9, Sep.1982, pp.65-79

[12] S.D.Conte, H.E.Dunsmore, V.Y.Shen, "Software Engineering Metrics and Model",The Benjamin/ Cumming Publishing Company,Inc., 1995

[13] Melody Y. Ivory, Marti A. Hearst,"Towards Quality Checkers for Web Site Designs", IEEE Internet Computing, March/April 2002

[14] Edward Miller,"The WebSite Quality Challenge", <http://www.soft.com/eValid/Technology/White.Papers/website.quality.html>, Feb 2002

[15] Harry M. Sneed, "Applying size complexity and quality metrics to an object-oriented application", Project Control for Software Quality, Shaker Publishing, 1999

[16] Linda Rosenberg, Ph.D., Lawrence Hyatt, "Developing a successful metrics program", International Conference On Software Engineering(IASTED) SanFrancisco CA, November 1997

[17] 강규욱, "ASP 어플리케이션의 유지보수를 지원 하는 모듈리티 측정도구의 설계와 구현", 단국대학교 석사학위 청구논문, 2000

[18] Giorgio Brajnik, "Quality Models based on

Automatic Webtesting", <http://www.dimi.uniud.it/~giorgio/papers/quality-models.html>, April 2002

[19] Thomas A. Powell , "WebSite Engineering", Prentice Hall PTR, 1998

[20] 황철현 , "웹 어플리케이션 복잡도 메트릭의 설계와 측정도구의 구현", 단국대학교 석사 학위 청구 논문, 2001

저자 소개



김 지 현
 1978년 이화여자 대학교 수학과 학사
 1994년 단국대학교 전자 정보 전공 석사
 2001년 단국대학교 전산통계학과 박사과정수료
 1998년~현재: 서일대학 전기전자컴퓨터공학계열 소프트웨어 전공 전임강사, 정보관리 기술사
 <주관심 분야> 소프트웨어 공학, 프로젝트 관리, 품질 관리, 웹 공학



박 철(Chel Park)
 1994년: 단국대학교 전산통계학과 졸업(학사)
 1998년: 단국대학교 전산통계학과 졸업(석사)
 2001년: 단국대학교 전산통계학과 박사수료
 2001년 9월 ~ 현재: 단국대학교 정보컴퓨터학부 컴퓨터과학전공 강의 전임강사
 <주관심 분야> 웹 어플리케이션, 소프트웨어 공학, 분산 시스템 등