韓國 컴퓨터情報學會 論文誌
第11卷 第3號, 2006. 7.

2006-11-3-4-2

웹 소프트웨어의 위험분석 모델에 관한 연구

김 지 현*, 오 성 균**

A Study of Risk Analysis Model on Web Software

JeeHyun Kim*, SungKyun Oh**

요 약

소프트웨어 개발환경이 웹 기반으로 급격히 전환되고 있으나 웹 소프트웨어 품질 측정 메트릭이나 추정 모델에 대한 연구는 매우 미흡한 실정이다. 본 연구는 웹 소프트웨어의 위험도가 객체 속성과 상관관계가 있는지 선형회귀 방법을 사용하여 분석하였고, 실무에서 사용되고 있는 중형이상의 6개 시스템을 대상으로 규모와 클래스 수(NOC), 규모와 메소드 수(NOM) 및 복잡도와 클래스 수(NOC), 복잡도와 메소드 수(NOM)에 관한 적정 모델을 제안하였다. 실험에 사용한 6 시스템 중 5 시스템(S06 제외)의 규모(LOC)와 NOM이 높은 관련성을 보였고 4 시스템(S04 & S06 제외)의 복잡도와 NOM, 복잡도와 NOC가 높은 관련성을 보였다. 여기서 웹 소프트웨어 구조를 이루는 서버, 클라이언트, HTML 세 요소 각각의 복잡도를 비교하였는데, 두 시스템(S04, S06)은 각 요소의 복잡도 차이가 비교적 높았으며 1시스템(S06)은 HTML 복잡도가 크게 치우쳐 있었다. 즉 위험도를 미리 추정하여 유지보수성을 향상시키기 위해서는 NOM으로 추정가능 하도록 세 요소의 복잡도를 균일하게 유지해야 함을 제시한다.

Abstract

Even though software developing environment has been changing to Web basis very fast, there are just few studies of quality metric or estimation model for Web software. In this study after analyzing the correlation between the risk level and property of objects using linear regression, six middle sized industrial system has been used to propose the correlation model of size and Number of Classes(NOC), size and Number of Methods(NOM), complexity and NOC, and complexity and NOM. Among of six systems 5 systems(except S06) have high correlation between size(LOC) and NOM, and four systems(except S04 & S06) have high correlation between complexity and NOC / NOM. As Web software architecture with three sides of Server, Client and HTML, complexity of each sides has been compared, two system(S04, S06) has big differences of each sides

• 제1저자 : 김지현
• 접수일 : 2006.05.22, 심사일 : 2006.06.23, 심사완료일 : 2006.07.16
* 서일대학 IT계열 소프트웨어 전공 조교수 ** 서일대학 IT계열 소프트웨어 전공 교수
※ 이 논문은 서일대학 학술연구 지원 논문입니다.

complexity values and one system(S06) has very higher complexity value of HTML. So the risk level could be estimated through NOM to improve maintenance in case of that the system has no big differences of each sides complexity.

▸ Keyword : 위험 분석(Risk Analysis), 웹 소프트웨어(Web Software), 클래스 수(Number of Classes), 메소드 수(Number of Methods)

I. 서 론

인터넷의 발달과 더불어 소프트웨어 개발 환경도 웹 기반으로 급속히 전환되고 있으나 아직도 소프트웨어 측정 기술과 메트릭에 대한 연구는 객체지향 개발 환경 및 기술 수준을 벗어나지 못하고 있다.

소프트웨어 개발과정의 효과적인 관리를 위해 산출물 개발에 요구되는 시간과 노력의 정확한 추정은 선행되어야 하는 필수요소이다. 시간과 노력의 측정기술에는 여러 가지가 있는데 이러한 기술의 대부분은 산출물의 외부 기능과 개발 시간 및 노력, 규모와의 연결 또는 규모와 개발 시간 및 노력 사이의 관계발견에 초점이 맞춰져 있다[1].

1980년대 객체-지향 프로그래밍이 등장하면서 15년이 지난 1990년대 중반에 들어서야 비로소 객체지향 측정 메트릭에 대한 관심이 고조되며 생산성, 설계노력 및 규모 등을 추정할 수 있는 추정 메트릭에 관한 연구가 이루어졌다.

개발환경이 웹 기반으로 전환되면서 이미 작성된 많은 프로그램들이 웹 기반으로 재 작성되고 있으며, 새로운 프로그램들도 다투어 웹 기반으로 작성되고 있는데 그에 비해 결함에 의한 위험은 높고 사용자 만족도는 극히 낮은 것으로 판명되고 있다[2].

웹 소프트웨어는 서버, 클라이언트, HTML등의 복합 구조를 띄고 있어 결함 가능성이 높을 수 있으며, 전통적 특성과 객체-지향 특성이 혼재하므로 위험가능성을 미리 추정하기 위하여 이러한 특성이 동시에 고려되어야 한다[13].

연구의 궁극적인 목표는 웹 소프트웨어 개발시 위험도가 높은 파일의 유형을 분석하여 미리 대비함으로써 효율적인 유지보수와 사용자 만족도를 높이고자 하는 것이다.

본 연구는 웹 사이트에서 사용되고 있는 중형 이상의 6개 시스템에 위험도 메트릭을 적용하여 결함 가능성이 높은 프로그램을 추출하고, 이 프로그램의 규모와 클래스 수(Number of Classes, NOC), 규모와 메소드 수(Number of Methods, NOM)와의 선형 회귀 분석을 통한 기존의 규모 추정 모델을 검증하고 복잡도(Cyclomatic Complexity, CC)의 추정을 위하여 복잡도(CC)와 클래스 수(NOC), 복잡도(CC)와 메소드 수(NOM)와의 상관관계를 규모 추정 모델과 같은 방법으로 분석한다. 이 때 복잡도와 NOC, 복잡도와 NOM과의 상관관계가 높게 나타난다면 미리 위험도가 높은 프로그램을 추정하여 유지보수 향상을 기대할 수 있을 것이다.

그러나 본 위험분석모델은 한계가 있는데 분석 대상이 웹 소프트웨어 전체가 아닌 위험도가 높은 프로그램을 추출하여 분석한 것으로 그 이유는 이 프로그램들의 결함 가능성이 높으므로 유지보수 관리의 집중이 이루어질 것이라는 가정에 의한 것이다.

논문의 구성은 2절에서 웹 소프트웨어의 구조적 특성 및 측정 메트릭과 문제점을 제시하고 3절에서 위험분석 모델을 설계하기 위한 이론적 배경으로 위험도 메트릭과 추정모델 동향 및 규모 추정 모델을 파악한다. 4절에서는 모델 설계를 위한 분석 대상 시스템의 설명과 설계 절차 및 복잡도 추정 과정을 보이고 선형 회귀 분석을 통하여 상관계수를 제시하며 복잡도 분석 파일에 따른 위험분석모델을 제안한다. 5절에서 결론과 향후 연구에 대하여 기술한다.

II. 웹 소프트웨어의 측정

2.1 웹 소프트웨어 구조

웹사이트를 구축하는 프로그램을 웹 소프트웨어라 하는데, 웹사이트 구조는 설계 관점(정보, 그래픽, 네비게이션 등), 페이지 포맷 관점, 또는 기술적(technical) 관점 등 여러 가지로 분류할 수 있다[12].

웹 소프트웨어는 기존의 소프트웨어와 달리 여러 프로그램 요소들이 결합되어 있다. 웹 소프트웨어 구성요소는 〈그림 1〉에 나타나 있는 것처럼 웹 서버에 존재하는 웹 소프트웨어 소스에는 서버에서 스크립트 엔진에 의해서 실행되는 서버 스크립트 요소와 클라이언트에 전송되는 클라이언트

스크립트 요소 및 HTML 태그로 구성된 HTML 요소들이 포함되어 있다[12]. 따라서 웹 소프트웨어의 규모와 복잡도를 측정하려면 서버 스크립트 요소, 클라이언트 스크립트 요소, HTML 요소들의 규모와 복잡도를 동시에 고려하여 각각의 합으로 정의되어야 한다[14].

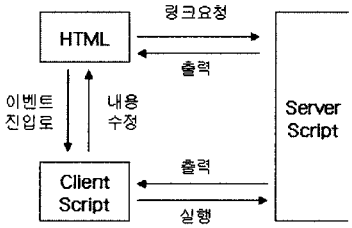

그림 1. 웹 소프트웨어의 구조
Fig.1 Web software architecture

이를 식으로 표현하여 웹 소프트웨어 규모를 Lweb이라 하면, Lweb = Ls + Lc + Lh 이 된다. 여기서 Ls는 서버스크립트 LOC, Lc는 클라이언트스크립트 LOC, Lh는 HTML의 LOC를 말한다. 또한 웹 소프트웨어 복잡도를 CCweb이라 하면, CCweb = CCs + CCc + CCh 로 나타낼 수 있다. CCs와 CCc는 서버 스크립트 요소와 클라이언트 스크립트 요소의 순환복잡도로 전통적인 순환복잡도와 같은 방법으로 측정되며 CCh는 HTML 웹 문서의 복잡도로 링크의수 + 1 로 측정한다[15].

2.2 웹 소프트웨어 측정과 문제점

"측정할 수 없는 것은 관리할 수 없다"라는 말에서 알 수 있듯이 웹 소프트웨어의 품질을 적정하게 유지하기 위해 측정은 필수적인 요소이다. 특히 웹 소프트웨어의 품질은 전 세계 수많은 사용자를 동시에 가질 수 있기 때문에 더욱 중요하다. 이러한 웹 사이트의 특수성 때문에 웹 소프트웨어는 소프트웨어 품질의 세계에서는 매우 새로운 것이며 웹의 즉시성은 빠른 소프트웨어의 개발과 즉각적인 품질의 기대를 요구하지만 웹 소프트웨어의 복잡성은 관리를 더욱 어렵게 만든다[3].

웹 소프트웨어 개발은 시스템화와 품질보증 과정의 부족으로 위기를 맞고 있는데, 이러한 위기를 극복하기 위하여 웹 소프트웨어의 개발과 유지보수에 웹 공학적 접근이 요구되고 있다. 소프트웨어의 품질수준, 시장요구 등은 증가 하지만 문서화 부족, 예산 초과, 품질 저하로 공급된 시스템이 비즈니스 요구를 충족시키지 못했다. 더구나 웹 소프트웨어의 품질은 상식이나 직관, 개발자의 경험에 의한 특별한 방법으로 접근되어 왔다. 웹 산출물의 품질과 과정에 대한연구는 매우 최근에 이루어 진 것으로 아직 널리 알려진 평가방법이나 기술이 부족한 실정이다. 복잡한 웹 소프트웨어 구조는 유지보수 비용을 증가시킨다는 것은 경험에 의해 얻어진 결론이다[4].

웹이 광고, 영업, 출판을 위한 중요한 새로운 매체라 할지라도, 기존의 웹 사이트 대부분이 엉성한 설계와 기초적인 기능 부족의 문제를 안고 있다. 이러한 문제들의 주된 이유는 웹 통신의 기초적인 기술이 실시간 응답을 제공하도록 설계되지 않았다거나, 웹 사이트 개발자들은 사이트를 가장 최근의 기술을 적용하는 것을 목표로 하는 경향이 있으며 많은 웹 사이트는 아마튜어에 의해 개발되어 대부분 공학적 표준을 적용하지 않는다는 것이다. 또한 웹사이트의 구현이 초기 요구를 충족시켜 설치되었는지 평가될 수 있는 추적성이 부족하다[2].

III. 모델설계의 이론적 배경

3.1 위험도 메트릭

웹 소프트웨어는 즉시성, 동시성, 광역성, 복잡성 등으로 인해 관리의 중요성이 가중되지만, 동시에 그러한 특성이 관리를 어렵게 하고 있다. 이에 Powell은 다음과 같은 공학적 해결책을 제시하였는데, '소프트웨어 산출물의 품질 속성은 소프트웨어의 구조를 분석하여 사용성과 유지보수성의 문제를 파악하고, 그에 따른 오류 모듈이 정의되어야 한다'고 기술하였다[5].

위험도 메트릭은 NASA의 SATC(Software Assurance Technology Center)에서 개발한 유지보수성 측정에 사용하기 위한 그래픽 템플릿이다. X축은 코드 라인의 수를 나타내고 Y축은 순환 복잡도를 나타낸다.

〈그림 2〉를 보면 위험도를 0에서 6까지 영역0(없음), 영역1(하), 영역2(하중), 영역3(중), 영역4(중상), 영역5(상), 영역6(최상)의 7개 영역으로 구분하였다. 각 영역을 구분하는 가이드라인은 NASA와 여러 기업의 소스 프로그램으로부터 결함과의 상관관계에서 추출한 통계 분석에 근거한 것이다. 규모(LOC)와 순환복잡도 값이 큰 영역4(중상), 영역5(상), 영역6(최상)에 분포한프로그램은 위험률이 높은 프로그램으로서, 심각한 결함이 발생할 확률이 높고 향후 유지보수에 문제가 생길 수 있다는 것을 통계분석에 의하여 추출하였다. 그러므로 이 영역에 분포하는 프로그램은 특별 관리가 필요하며, 개발자들은 이 영역에 분포한 프로그램의 결함에 대해 더욱 조사해야 한다고 분석하였다[6].

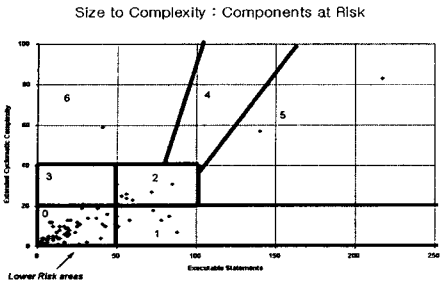

그림 2.. 위험도 메트릭
Fig.2 Risk Level Metric

본 연구에서는 기존에 이미 C나 C++언어에서 검증된 위험도 메트릭을 ASP로 작성된 웹 소프트웨어에 적용하였다. 여기서 서버, 클라이언트, HTML등 복합구조의 웹 소프트웨어 위험도와 서버 스크립트의 위험도 차이가 5이상인 프로그램을 추출하였는데 그 이유는 웹 소프트웨어 구조로 전환되면서 기존의 서버 스크립트 위험도를 크게 증가시킨 웹 소프트웨어는 문제가 되므로 자동화해서 위험도를 줄여야 한다는 연구 결과에 의한 것이다[14]. 즉 위험도를 증가시킨다는 것은 결함 가능성을 증가시켜 유지보수 효율을 저하시킬 것이며, 이 영역의 프로그램을 특별 관리하는 것은 전체 웹 프로그램 개발 과정의 효율성에 영향을 미치게 될 것이다. 이렇게 추출된 프로그램의 클래스 수(NOC), 메소드 수(NOM)를 측정하여 위험도와의 선형회귀 분석을 통한 위험분석 모델을 제안한다.

3.2 추정 모델 동향

최근에 발표된 소프트웨어 추정 메트릭에는 다음과 같은 연구가 존재한다.

1993년 Li & Henry는 '유지보수 추정 객체-지향 메트릭'에서 유지보수 노력대신 변경한 라인 수를 측정하여 결합도(CBO)를 제외한 CK(Chidamber & Kemerer) 메트릭과 관련이 높다고 기술하였다[7].

1996년 Basili, Brian & Melo는 '품질 지시자 로서의 객체-지향 설계 메트릭의 검증'에서 결함의 수는 메소드 수(NOM), 상속 트리 깊이(DIT), 결합도(CBO), 클래스 응답(RFC)등과 관련이 있다는 것을 증명하였다[8].

1998년 Chidamber, Darcy & Kemerer는 '객체-지향 소프트웨어의 관리자적 사용: 기술적 분석'에서 생산성, 재작업 노력, 설계 노력은 결합도(CBO)와 응집도 부족(LCOM)의 높은 값에 의해 영향 받는다고 하였다[9].

1999년 V. Misic과 D. Tesic은 '노력과 복잡도 추정 : 객체-지향 케이스 스터디'에서 노력과 총 클래스 수(NOC) 및 총 메소드 수(NOM)가 밀접한 관련이 있다는 것을 선형회귀분석을 통해 제시하였다. 복잡도와 함수 수와의 상관관계를 실험적 탐구를 통해 정의함으로써 이를 설계 단계에서 추정하였다10].

또한 2003년 Ronchetti 등은 '객체-지향 환경에서의 소프트웨어 규모 조기 추정'에 관한 연구에서 소프트웨어 개발 노력 대신 규모를 측정하여 LOC 메트릭을 사용하였다. 실험대상으로는 CMM level 3 소프트웨어 회사에서 개발된 2개 프로젝트가 이용되었다. 실험을 위하여 사용된 총 6개의 메트릭 중 메소드 수(NOM)가 규모(LOC)와 상당한 관련이 있음을 통계적 분석을 통해 제시하였다[1].

이러한 연구는 대부분 객체-지향 언어인 C++ (Li & Henry의 경우만 Ada 사용)로 작성한 프로그램에 대하여 이루어졌으나 본 연구는 웹 프로그래밍 언어인 ASP로 작성된 프로그램을 대상으로 하였다. 또한 결함 가능성을 보이는 위험도는 규모와 복잡도에 의해 정의되는데 선행 연구에서 제안된 규모추정 모델의 검증과 본 논문에서 제안한 복잡도 추정 메트릭으로 위험분석 모델의 설계를 제안하고자 한다.

3.3 규모추정 모델

위험도 영역은 규모와 복잡도에 의해 결정되므로 규모(Lines of Code, LOC)와 NOC, NOM과 상관정도가 높고 복잡도도 NOC, NOM과의 상관정도가 높다면 위험도 또한 NOC, NOM에 의해 추정이 가능하다. 이에 본 연구에서는 기존의 LOC와 NOC, LOC와 NOM의 상관관계에 따른 규모 추정 모델을 적용하여 검증절차를 거치고자 한다.

이에 규모 추정 모델로써 웹 소프트웨어의 자바스크립트 form 파일 유형은 NOC나 NOM으로 규모 추정이 가능하며 그 중에서 특히 NOM이 더 밀접한 관련을 보이므로 규모추정모델에 적합함을 제안하였다[15].

이 실험에는 분석 대상 자료로 3개의 웹 사이트 시스템이 사용되었고 웹 소프트웨어와 서버 스크립트 복잡도의 차이가 5이상인 결함가능성이 높은 프로그램이 선택되었으며 복합구조를 가진 웹 소프트웨어의 클라이언트나 HTML 파일 유형에 따라 상관관계를 나타내는 결정계수가 매우 다른 양상을 보였다. 즉, 클라이언트 자바 스크립트로 작성된 form 파일로 구성된 프로젝트는 가장 높은 상관관계를 보였으나 HTML menu 파일이 혼합된 프로젝트는 가장 낮은 관련성을 나타냈고, 클라이언트 control 파일이 혼합된 프로젝트는 중간 정도의 관련성을 보였다. 이에 따라 후자의 두 프로젝트에서 menu 파일과 control 파일을 제거하고

form 파일 유형만 상관관계를 분석하였더니 0.7이상의 매우 높은 결정계수를 보였다. 즉 웹 소프트웨어의 규모를 추정하기 위해서는 분석대상 프로그램 파일 유형 중 90% 이상을 차지하는 form 파일 유형의 LOC를 추정하는 것이 정확도를 높일 수 있으며 메소드 수(NOM)가 클래스 수(NOC)보다 더 높은 상관관계를 가짐을 알 수 있었다[15].

IV. 위험분석모델의 설계

4.1 대상 시스템 및 설계 절차

실험 대상 시스템은 웹 사이트 구축 언어로서 쉽게 접할 수 있고, MS Windows 환경에서 가장 많이 사용되는 언어 중 하나인 ASP로 작성된 웹 소프트웨어를 실험 대상으로 채택하였다. 연구에 사용된 웹 소프트웨어는 서버측 스크립트로 일반적으로 많이 사용하는 VBScript를 사용하고 클라이언트측 스크립트로 JavaScript를 사용하는 것들을 대상으로 하였다.

실험은 총 6개의 웹 사이트 시스템을 대상으로 하였다. 공기업의 전산자원관리에서부터 중고차 매매사이트, 기업의 단위 업무를 처리하는 시스템, 개발회사에서 품질관리와 프로젝트 관리를 위해 사용하는 시스템 등이 포함되어 있다. 6개 시스템의 총 파일 수는 4,968이며 여기에서 실험에 사용된 ASP 파일 수는 1,574이다. 실험에 사용된 시스템별 총 파일수, LOC와 복잡도 추출 소스 파일수, 위험도차이 5 이상인 파일 수 등을 〈표 1〉에 설명하였다.

표 1. 실험 대상 시스템
Table 1. Description of Source System

	실험대상 프로젝트	총 파일수	위험도추출파일수 (*.asp)	위험도차이 5 이상인 파일 수
S01	공기업 DBRMS	831	243	8
S02	중고차 매매사이트	263	222	12
S03	대학 웹사이트	757	340	4
S04	수발주 관리	748	315	23
S05	품질관리	622	269	8
S06	프로젝트 관리	1,747	185	6
	프로젝트 합계	4,968	1,574	61

위험분석모델을 설계하기 위하여 구현한 설계절차는 〈그림 3〉과 같으며 검증을 위한 상세 절차는 다음과 같다.

1. 먼저 ASP 소스 파일을 입력하여 파싱 처리 후 LOC와 순환복잡도를 측정하여 그 결과를 레파지토리에 저장한다(〈그림 3〉의 ①과정).
2. 레파지토리에 저장된 정보를 사용하여 다시 ASP 소스 파일을 파싱처리하여 클래스 수(NOC)와 메소드 수(NOM)를 측정하여 결과를 레파지토리에 저장하고 중간 결과를 산출한다(〈그림 3〉의 ②과정).
3. 위험도 메트릭을 적용하여 웹 소프트웨어 위험도가 서버 스크립트의 위험도와 5이상의 차이를 보이는 프로그램을 추출하여 각 시스템별 LOC와 NOC, LOC와 NOM의 상관관계 및 CCweb와 NOC, CCweb와 NOM의 상관관계를 분석한다.
4. 이 측정 결과를 쉽게 읽고 시각적 통계처리가 가능하도록 액셀 쉬트에 옮겨 통계 분석 작업을 한다.
5. LOC 와 NOC, LOC와 NOM, CC와 NOC, CC와 NOM의 시스템별 선형회귀 분석을 하여 결정계수 R2이 0.6 이상의 값을 취하는지 알아본다. 결정계수 값이 0.6 미만일 경우 웹 소프트웨어 파일 유형을 분석하고 유형별 선형회귀분석을 한다. 파일 유형별 적정 결정계수가 보이지 않으면 클라이언트 복잡도(CCc)와 HTML 복잡도(CCh)를 비교분석하여 적정 모델을 제안한다.

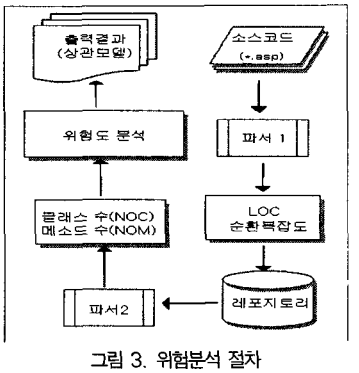

그림 3. 위험분석 절차
Fig.3 Risk Analysis Procedure

여기서 ASP 프로그램의 클래스 수(NOC)와 메소드 수(NOM)의 추출은 'document.write()', 'today.getDate()'와 같이 "."를 기준으로 왼쪽에 있는 것은 클래스로 프로그램에 나타난 총 갯수를 클래스 수(NOC)로, 오른쪽의 함수에 해당하는 것은 메소드로 그 총 갯수를 메소드 수(NOM)로 정의하였다.

다음 절에서 각 시스템별 웹 소프트웨어 복잡도인 CCweb와 NOC, CCweb와 NOM의 상관관계를 파악하기 위해 선형회귀분석 과정을 제시한다.

다음은 본 연구에서 사용한 선형회귀식으로 LOC와 NOC의 관계식은 LOC = a x NOC + b 로 나타내며 LOC와 NOM의 관계식은 LOC=c x NOM + d 로 나타

낸다. 또한 CC와 NOC의 관계식은 CC = a x NOC + b 로 나타내며 CC와 NOM의 관계식은 CC=c x NOM + d 로 나타낸다. 모델의 적정성(R2)은 모델이 얼마나 자료와 잘 맞는지를 나타내는 지시자이다. R2 은 0~1사이의 값을 가지며, 값이 1에 가까울수록 모델의 정확성은 높아진다.

4.2 복잡도 추정

복잡도를 추정하기 위하여 규모 추정 방법과 같은 방법으로 복잡도(CCweb)와 NOC, 복잡도와 NOM의 선형회귀 분석을 한 결과 〈그림 4〉와 같이 6개 시스템 중 3개 시스템(S01, S02, S05)에서 결정계수(R2)가 0.6이상의 높은 값을 나타내고 있다.

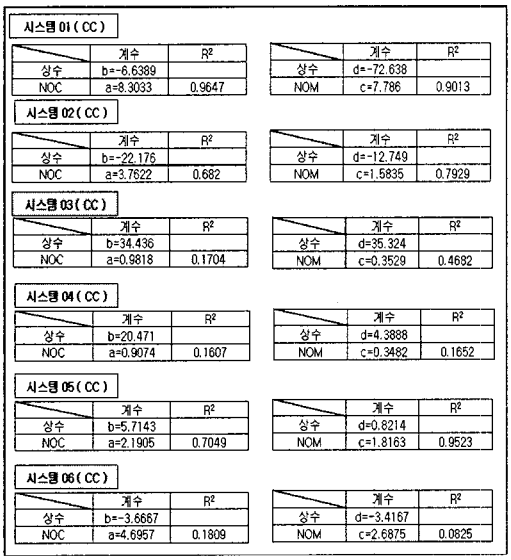

그림 4. 복잡도와 NOC / NOM의 선형회귀분석
Fig.4 Linear regression analysis of CC & NOC, CC & NOM

그러나 S03, S04, S06의 결정계수(R2)는 0.3이하의 낮은 값을 나타내고 있다. 이에 규모추정모델이 메뉴파일 유형이나 control파일 유형을 제외한 폼 파일 유형이 메소드 수(NOM)와 상관관계가 높다는 연구결과에 따라 6 시스템의 파일유형 분포를 파악하여 〈표 2〉에 제시하였다.

표 2. 각 시스템의 파일유형 분포
Table 2. File pattern of source system

	S01	S02	S03	S04	S05	S06
Menu 파일	1	0	1	0	1	0
Form 파일	3	12	3	20	5	5
Control 파일	4	0	0	3	2	1
합계	8	12	4	23	8	6

복잡도와 NOC/ NOM의 관계를 분석하기 위해 메뉴파일유형을 제외하고 회귀분석한 결과가 〈그림 5〉에 나타나 있다. S04, S06 모두 결정계수에 별 차이가 없었다.

그림 5. 메뉴파일 제거후 복잡도의 선형회귀 분석
Fig.5 Linear regression analysis of Complexity after elimination of menu file

여기서 각 시스템의 복잡도 분포를 비교하기 위하여 결정계수가 높은 S02를 대표로 제시하고 결정계수가 낮은 S04, S06의 그래프를 〈그림 6〉에 제시하였다. 웹 복잡도(CCweb)와 클라이언트 스크립트 복잡도(CCc), HTML 복잡도(CCh)의 차이를 비교하기 위한 누적그래프이다.

S04의 웹 복잡도(CCweb)는 40이하로 타 시스템과 큰 차이가 없었으나 HTML 복잡도(CCh)값이 클라이언트 스크립트 복잡도(CCc) 값에 비해 80%이상의 높은 값을 보이고 있었다.

S06의 경우 6개중 4개의 웹 복잡도(CCweb)가 50 이상으로 높았으며 HTML 복잡도(CCh)도 타 시스템에 비해 높은 값을 나타냈고 클라이언트 스크립트 복잡도(CCc) 도 5개가 3이상의 높은 값을 보였다. S04, S06 시스템의 공통점은 HTML 복잡도(CCh)값이 클라이언트 스크립트 복잡도(CCc) 값에 비해 차이가 크다는 것이다. 이러한 현상은 타 4개의 시스템에서는 발견할 수 없는 현상이다.

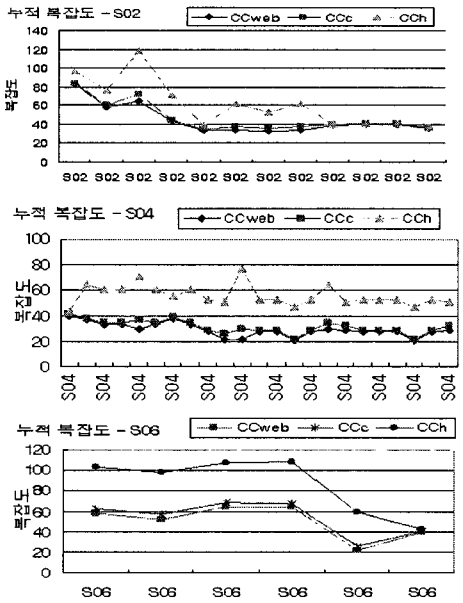

그림 6. 복잡도 비교-S02 대 S04 & S06
Fig.6 Complexity comparison-S02 vs S04 & S06

4.3 위험분석모델

위험도 차이가 5 이상인 파일들의 규모가 NOC, NOM과 높은 관련성을 보임에 따라 실험 대상 시스템의 개수를 배로 확장하여 검증하였다.

위험도는 규모(LOC)와 복잡도(CC)에 의해 결정되는데, 규모추정은 NOC, NOM의 수가 커지면 따라서 증가하므로 NOC, NOM의 값이 증가하면 규모가 커질 것을 예상할 수 있고 특히 NOM의 값이 더 관련이 높으며 웹 소프트웨어를 구성하는 네 가지 파일 중에 메뉴파일유형을 제거한 form 파일 유형이 관련이 높다는 연구 결과가 제시되었다.

본 연구에서 6개 시스템으로 확장하여 메뉴파일을 제거하고 실험한 결과 〈그림 7〉에서 보이는 것과 같이 S01, S02, S03, S04, S05 5개 시스템에서 결정계수(R2) 값이 0.6이상의 높은 값을 나타냈고 단지 S06 시스템에서 LOC와 NOM의 결정계수(R2) 값이 매우 낮은 현상을 보였다.

또한 복잡도(CCweb)와 NOC, 복잡도(CCweb)와 NOM과의 결정계수를 선형회귀분석으로 검증하였더니 위의 결과에서처럼 4개 시스템에서 비교적 높은 관련성을 보였고 2개 시스템인 S04, S06에서 낮은 관련성을 나타냈다.

시스템 01 (LOC) (F,C)

	계수	R^2		계수	R^2
상수	b=4.5141		상수	D0.4726	
NOC	a=-0.0036	0.0064	NOM	c=0.0887	0.6229

시스템 02 (LOC) (F)

	계수	R^2		계수	R^2
상수	b=-63.934		상수	d=-0.5936	
NOC	a=18.767	0.6845	NOM	c=7.4511	0.7081

시스템 03 (LOC) (F)

	계수	R^2		계수	R^2
상수	b=-40.6		상수	d=136.57	
NOC	a=23.4	0.9469	NOM	c=3.3429	0.9469

시스템 04 (LOC) (F)

	계수	R^2		계수	R^2
상수	b=61.18		상수	d=81.769	
NOC	a=10.971	0.7558	NOM	c=5.9703	0.8139

시스템 05 (LOC) (F,C)

	계수	R^2		계수	R^2
상수	b=5.3556		상수	d=4.8824	
NOC	a=0.0267	0.2869	NOM	c=0.0596	0.7289

시스템 06 (LOC) (F)

	계수	R^2		계수	R^2
상수	b=5.7639		상수	d=17.51	
NOC	a=0.0256	0.3338	NOM	c=0.0104	0.0417

그림 7. 메뉴파일 제거후 규모의 선형회귀 분석
Fig.7 Linear regression analysis of Size after elimination of menu file

S06 시스템은 규모와 NOC/ NOM, 복잡도와 NOC/ NOM의 결정계수가 낮은 값을 보였는데 복잡도 추정에서 조사한 바와 같이 HTML 복잡도(CCh)가 타 시스템에 비해 높은 값을 나타냈고 클라이언트 스크립트 복잡도(CCc)도 높은 값을 보였으며 HTML 복잡도(CCh)와 클라이언트 스크립트 복잡도(CCc)의 차이가 타 시스템에 비해 매우 큰 것으로 나타났다. 또한 S04 시스템에서는 클라이언트 스크립트 복잡도(CCc)는 낮은 값을 보이나 HTML 복잡도(CCh) 는 상대적으로 높은 값을 보였다. 이와같이 HTML 복잡도(CCh)가 클라이언트 스크립트 복잡도(CCc)보다 상대적으로 높아 차이가 클 때는 NOC, NOM으로 미리 추정 가능하지 않다는 것이다.

그러므로 위험도가 높은 웹 소프트웨어의 복잡도를 미리 추정하기 위해서는 서버, 클라이언트, HTML 세 가지 요소의 복잡도가 고르게 분포되어 있어야 가능하다.

결론

웹 소프트웨어는 서버, 클라이언트, HTML의 세가지 요소가 복합적으로 구성되어 있어 위험도를 미리 추정하거나 유지보수가 쉽지 않은 속성을 내재하고 있으며 웹 소프트웨어의 위험도가 높은 소프트웨어를 중심으로 규모를 추정하는데 메소드 수(NOM)가 밀접한 관련이 있다는 연구가 진행되었다.

이에 규모(LOC)와 복잡도(CC)에 의해 결정되는 위험도의 NOC, NOM과의 관련성을 조사하여 만약 선형관계가 파악된다면 NOC, NOM 값이 큰 파일의 결함률을 줄이도록 미리 주의를 기울여 유지보수성을 높일 수 있을 것이다.

실험에 사용한 6개 시스템 중 5개의 시스템은 규모(LOC)와 NOM의 상관계수(R2)가 0.6이상으로 관련이 있음을 보였으며 복잡도와 NOC, 복잡도와 NOM의 결정계수(R2)는 4개 시스템에서 0.7이상의 높은 관련성을 보였다. 복잡도와 NOC/NOM과의 결정계수(R2)가 낮은 S04 시스템은 HTML복잡도(CCh)가 클라이언트스크립트 복잡도(CCc)에 비해 매우 큰 경우이었고, 규모와 복잡도 모두에서 NOC/NOM과의 결정계수(R2)가 낮은 S06 시스템은 HTML복잡도(CCh)와 클라이언트스크립트 복잡도(CCc)가 타 시스템에 비해 현저히 클 뿐만 아니라 둘의 차이도 매우 큰 값을 보였다.

따라서 웹 소프트웨어의 위험도를 미리 추정하고 대비하기 위해서는 서버, 클라이언트, HTML 세가지 요소의 복잡도가 고르게 구성되어 있어야 클래스 수(NOC), 메소드 수(NOM)로 추정 가능하며, 만약 HTML 복잡도(CCh)가 클라이언트 스크립트 복잡도(CCc)보다 상대적으로 높아 차이가 클 때는 NOC, NOM으로 미리 추정 가능하지 않다는 것이다.

본 연구에서 사용된 시스템은 모두 다른 종류의 웹시스템으로 규모와 복잡도를 추정하기 위한 정확한 회귀식을 제시하지는 못했으나, 향후 같은 업무 분야의 사이트 들을 검증 할 수 있다면 NOC, NOM 또는 다른 속성을 통해 규모와 복잡도 추정식의 제시가 가능하리라 보며 위험도를 효율적으로 개선하여 웹 사이트의 만족도 향상에 기여할 수 있을 것으로 기대한다.

참고문헌

[1] Marco Ronchetti, Giancarlo Succi, Witold Pedrycz, Barbara Russo, "Early estimation of software size in object-oriented environments : a case study in a CMM level 3 software firm", www.unibz.it/web4archiv/objects/pdf/cs_library/1/, 2003

[2] 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

[3] Edward Miller,"The WebSite Quality Challenge", http://www.soft.com/eValid/Technology/White.Papers/website.quality.html, 2002.2

[4] Silvia Abrahao, Luis Olsina, Oacar Pastor, "A Methodology for Evaluating Quality and Functional Size of Operative WebApps", www.dsic.upv.es/~west/, 2002

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

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

[7] Li, W., and S. Henry, "Object Oriented Metrics That Predict Maintainability," Journal of Systems and Software, 23(2), 1993

[8] Basili, V.R., L.C. Briand, and W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," IEEE Transactions on Software Engineering, 22(10),1996

[9] Chidamber, S.R., D.P. Darcy, and C.F. Kemerer, "Managerial Use of Object-Oriented Software : An Explanatory Analysis," IEEE Transactions on Software Engineering, 24(8), 1998

[10] V.B.Misic, D.N.Tesic, "Estimation of effort and complexity: An object-oriented case study", Journal of Systems and Software, Jan 1999

[11] Victor Laing and Charles Coleman, "Principal Components of Orthogonal Object-Oriented Metrics", www.gsfc.nasa..gov/support/OSMASASMSEP01/, 2001.10

[12] 김 지현, 박철, "웹 어플리케이션의 모듈위험수준 측정 도구의 구현", 한국컴퓨터정보학회논문지 제7권 제2호, 2002.6.

[13] 오성균, 김미진, "웹어플리케이션의 복잡도 예측에 관한 연구", 한국컴퓨터정보학회논문지 제9권 제3호, 2004.9.

〔14〕 박철, 유해영, "웹 어플리케이션의 순환복잡도 분석", 정보처리학회논문지D, v.11-D, n.4, pp.865-872, 2004.08
〔15〕 김지현, 유해영, "웹 소프트웨어 규모 예측에 관한 연구", 정보처리학회논문지D, 제12-D권 제3호, 2005.05

저 자 소 개

김 지 현

1978년 이화여자대학교 문리대학 수학과
1994년 단국대학교 전자정보전공 석사
1997년 정보관리 기술사 취득
2005년 단국대학교 전산통계학과 컴퓨터학전공(이학박사)
1998년~현재 : 서일대학 IT계열 소프트웨어 전공 조교수
관심 분야 : 웹 공학, 프로젝트 관리, 품질 관리

오 성 균

1981 홍익대학교 이공대학 전자계산학과 (이학사)
1984 연세대학교 산업대학원 전자계산학과(공학석사)
1999 홍익대학교 이공대학 전자계산학과 (이학박사)
1987년 ~ 현재 : 서일대학 IT계열 소프트웨어 전공 교수
관심분야 : 능동 및 멀티미디어 DB, XML 모델링