웹 소프트웨어 규모 예측에 관한 연구

김 지 현[†] · 유 해 영[††]

요 약

소프트웨어 개발 패러다임이 21세기에 들어서며 웹 기반으로 빠르게 전환되고 있으나 웹 환경에 적합한 품질 및 예측 매트릭에 대한 연구는 매우 미흡한 실정이다. 본 연구는 웹 소프트웨어의 규모와 객체 속성의 상관관계를 분석하여 실 업무에서 사용되고 있는 ASP 기반의 3개 프로젝트를 대상으로 결함 가능성이 높은 프로그램을 추출하고 프로그램 규모와 클래스 수나 메소드 수에 대한 선형회귀분석을 통하여 웹 소프트웨어의 규모 예측에 적합한 모델을 제안한다. 서버, 클라이언트, HTML의 복합구조를 가지는 웹 소프트웨어 중 자바스크립트 form 파일 유형의 높은 상관관계와 규모 예측에 적합한 메소드 수 매트릭을 제시한다.

키워드 : 규모 예측, 웹 소프트웨어, 클래스 수, 메소드 수

A Study of Estimation for Web Software Size

JeeHyun KIM[†] · HaeYoung YOO[††]

ABSTRACT

Even though development paradigm of software has been changing very fast at the beginning of 21st Centuries, there are just few studies of quality and estimation metrics appropriate for Web environment. So in this study after analyzing the correlation between the size of the final code and property of objects, three industrial real world projects written in ASP have been used for deriving programs with high possibilities of faults. And then the size of programs was analyzed to correlate with the number of classes or the number of methods through linear regression. Among the Web software with the complex architecture of Server, Client and HTML, type of form file written in Javascript for client has the high correlation and the number of methods is well correlated with the size of final code.

Key Words : Estimation of Size, Web Software, Number of Classes, Number of Methods

1. 서 론

소프트웨어 개발과정의 효과적인 관리를 위해 산출물 개발에 요구되는 시간과 노력의 정확한 예측은 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 이후 객체-지향 프로그래밍이 주요 개발 환경으로 자리하면서 객체-지향 매트릭에 대한 연구가 시작되었으며 1990년대 중반에 들어서야 비로소 유지보수에 의해 변경한 라인 수, 결함의 수, 생산성, 설계노력 및 규모 등의 예측 매트릭에 관한 연구가 이루어졌다.

이렇듯 객체-지향 소프트웨어 개발에 요구되는 노력이나 시간을 예측하기 위한 다양한 연구가 진행되어 왔으나 웹 환경으로 급격히 전환되면서 웹 소프트웨어의 구조에 적합한 매트릭은 물론 개발 노력과 시간의 예측에 대한 연구는 이루어지지 않고 있는 실정이다. 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 복합적인 구조를 띄고 있어 결함 가능성이 높을 수 있으며, 전통적 특성과 객체-지향 특성이 혼재하므로 개발 노력이나 규모를 미리 예측하기 위하여 이러한 특성이 동시에 고려되어야 한다.

연구의 궁극적인 목표는 웹 소프트웨어의 개발 시간 및 노력과 밀접한 관련이 있는 객체 속성을 추출하여 관리자나 개발자가 초기에 노력을 미리 예측하고자 하는 것이다.

본 연구는 웹 사이트에서 사용되고 있는 중형 이상의 3개 웹 프로젝트에 위험분석 매트릭을 적용하여 결함 가능성이 높은 프로그램을 추출하고, 프로그램의 규모와 클래스 수(Number of Classes, NOC)나 메소드 수(Number of Methods,

† 정 회 원 : 서일대학 소프트웨어전공 조교수
†† 정 회 원 : 단국대학교 정보컴퓨터학부 교수
논문접수 : 2005년 2월 17일, 심사완료 : 2005년 3월 30일

NOM)와의 선형 회귀 분석을 통하여 규모 예측에 적합한 모델을 제안한다.

제안된 모델은 다음의 두 가지 측면에서 한계가 있는데, 첫째는 분석 대상이 웹 소프트웨어 전체가 아닌 위험도가 높은 프로그램을 추출하여 분석한 것으로 그 이유는 이 프로그램들의 결함 가능성이 높으므로 개발 노력과 시간의 집중이 이루어질 것이라는 가정에 의한 것이다. 둘째는 웹 소프트웨어의 개발 노력과 시간 대신 규모에 초점을 맞추어 분석했다는 것인데, 규모 정보는 쉽게 얻을 수 있는 이점이 있기 때문이다.

이러한 한계에도 불구하고 본 연구가 가지는 유일성은 복합 구조의 웹 소프트웨어 규모와 관련 있는 객체 속성의 추출을 시도하였다는 것이며, 웹 소프트웨어의 일부 파일 유형에 적합한 예측모델을 제시했다는 것에 있다.

논문의 구성은 2절에서 전통적 매트릭과 객체-지향 매트릭 및 기존의 예측 매트릭을 조사하고, 3절에서 웹 소프트웨어의 구조적 특성 및 위험분석 매트릭에 의한 결함 가능성이 높은 웹 소프트웨어의 추출 방법을 제시한다. 4절에서는 분석 대상 자료 및 환경에 대한 설명을 하고 선형 회귀 분석식과 결정계수를 제시하며 파일 유형에 따른 분석결과 모델을 제안한다. 5절에서 결론과 향후 계획에 대하여 기술한다.

2. 관련 연구

전통적 프로그래밍과 객체-지향 프로그래밍은 근본적으로 다르므로 프로그램 평가에 서로 다른 매트릭이 필요하다는 견해도 있으나 전통적 매트릭과 객체-지향 매트릭이 객체-지향 시스템의 전반적 품질 분석에 최선의 결과를 준다는 의견이 지배적이다[4].

2.1 전통적 매트릭

전통적 매트릭은 1970년대 이후 구조적 시스템의 복잡도 측정에 이용되어 왔으며, 여기에는 McCabe의 순환복잡도, 코드 라인 수 등이 있다.

(1) McCabe의 순환복잡도

그래프 이론에 근거한 모듈의 흐름복잡도를 측정하기 위한 매트릭이다. 모듈의 순환복잡도는 선택점의 수와 같으며 모든 수행경로를 커버할 테스트 케이스의 최소 수를 측정하는 것이다. 높은 값의 순환 복잡도는 낮은 품질을 나타내며 시험과 유지보수를 어렵게 한다.

(2) 코드 라인 수(Lines Of Code, LOC)

수행코드의 물리적 라인 수를 측정하는 것이다. 모듈의 LOC 값이 클수록 이해성과 유지보수성은 낮아진다. 비판적인 견해도 존재하나 프로그램 규모 측정의 가장 쉬운 방법으로 널리 사용되고 있다.

본 연구에서는 규모측정에 일반적으로 많이 사용되는 코드 라인수(이하 LOC)를 사용했으며, 복잡도에는 순환복잡도가 사용되었다.

2.2 객체-지향 매트릭

객체-지향 시스템에 널리 사용되는 것으로 Chidamber와 Kemerer는 6가지 측정 매트릭을 제시하였고 V. Misic은 추상화 수준에 따른 측정 매트릭을 분류하였다.

(1) CK 매트릭

Chidamber와 Kemerer가 제안한 것으로 가중메소드(Weighted Methods Per Class, WMC)는 개별 클래스의 복잡도를 측정하는 것이다. 측정방법에는 두 가지가 존재하는데, 하나는 클래스 내 각 메소드의 복잡도 합이고 다른 하나는 메소드 복잡도를 1로 보았을 때 메소드 수의 합, 즉 클래스 당 메소드의 수가 된다. 메소드의 수와 복잡도는 클래스 개발과 유지보수에 드는 시간과 노력의 직접적 예측자이다. 자식의 수(Number of Chidren, NoC)는 특정 클래스의 하위 클래스 수를 나타낸다. 클래스 결합도(Coupling Between Objects, CBO)는 특정 클래스에 연결된 다른 클래스의 수로 정의된다. 클래스 응답(Response for a Class, RFC)은 클래스에 속한 객체가 받은 메시지에 대한 응답으로 수행 가능한 메소드의 수로 정의된다[4].

이 외에 상속트리의 깊이(Depth of Inheritance Tree of a Class, DIT), 응집도의 결여(Lack of Cohesion in Methods, LCOM)등 6가지 매트릭은 클래스 수와 메소드 수가 근간이 되어 객체-지향 품질을 결정하는 중요한 매트릭으로 활용되고 있다.

(2) 추상화 수준에 따른 측정 매트릭

추상화 수준에 따른 측정 매트릭으로 다음의 다섯 가지 방식이 존재한다.

첫째, 메소드나 서비스의 복잡도를 모델화하는 메소드 수준에서의 측정이 있다. 지역 변수와 같이 참조된 실행문의 집단으로 '함수', '프로시저', '모듈' 등이 있다. McCabe의 순환 복잡도나 팬인-팬아웃 같은 것이 복잡도 측정에 사용된다. 둘째, 클래스 수준에서의 측정은 전체적으로 클래스의 특성을 모델화한다. 이것은 메소드와 속성의 수, 개별 또는 공동 메소드나 변수의 비율, 클래스 응집도등을 포함한다. 셋째, 상속 계층의 깊이와 높이, 상위 클래스에서 상속받은 메소드 수, 하위 클래스에 재 정의된 메소드 수나 정의된 메소드 수 등을 측정한다. 넷째, 클래스들 사이의 연결 정보가 있다. 클래스 연결은 메소드 호출에 따라 정적 또는 동적일 수 있다. 다섯째, 프로젝트 개발에 필요한 시간과 노력은 시스템 자체의 규모에 매우 의존적이며, 클래스 수, 메소드 수, 상속 계층의 수와 규모, 클래스 연결의 수 등으로 평가될 수 있다[3].

2.3 기존의 예측 매트릭

객체-지향 환경에서의 소프트웨어 예측 매트릭에는 다음

과 같은 연구가 존재한다.

Li & Henry는 '유지보수 예측 객체-지향 매트릭'에서 유지보수 노력대신 수정 변경한 라인 수를 측정하여 결합도(CBO)를 제외한 CK 매트릭과 관련이 높다고 기술하였다[7].

Basili, Brian & Melo는 '품질 지시자 로서의 객체-지향 설계 매트릭의 검증'에서 결함의 수는 메소드 수(NOM), 상속 트리 깊이(DIT), 결합도(CBO), 클래스 응답(RFC)등과 관련이 있다는 것을 증명하였다[10].

Chidamber, Darcy & Kemerer는 '객체-지향 소프트웨어의 관리적 사용 : 설명적 분석'에서 생산성, 재작업 노력, 설계 노력은 결합도(CBO)와 응집도 부족(LCOM)의 높은 값에 의해 영향 받는다는 것을 제시하였다[11].

Ronchetti 등은 '객체-지향 환경에서의 소프트웨어 규모 조기 예측'에 관한 연구에서 소프트웨어 개발 노력 대신 규모를 측정하였고 매트릭으로는 LOC가 적용되었다. 분석에 사용된 매트릭으로는 외부복잡도, 내부복잡도, 상속트리의 깊이(DIT), 메소드 수(NOM), 자식의 수(NoC), 속성의 수(Number of Attributes)등이었으며, CMM level 3 소프트웨어 회사에서 개발된 2개 프로젝트를 실험 대상으로 하였다. 6개의 매트릭 중 메소드 수(NOM)가 규모와 상당한 관련이 있음을 통계자료를 통해 제시하였다[1].

또한 V. Misic과 D. Tesic은 '노력과 복잡도 예측 : 객체-지향 케이스 스터디'에서 노력과 총 클래스 수(NOC) 및 총 메소드 수(NOM)가 밀접한 관련이 있다는 것을 선형회귀분석을 통해 제시하였고 복잡도와 함수 수와의 상관관계를 실험적 탐구를 통해 정의함으로써 이를 설계 단계에서 예측할 수 있다고 기술하였다[3].

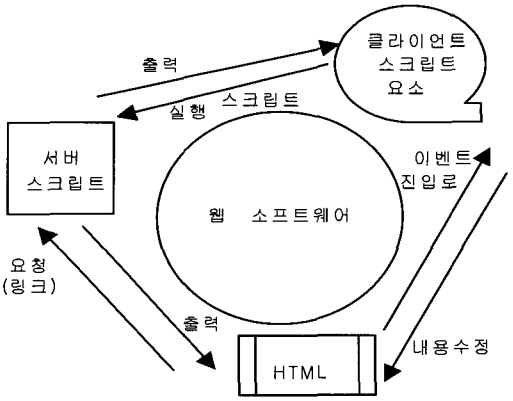
이러한 연구는 대부분 객체-지향 언어인 C++ (Li & Henry의 경우만 Ada 사용)로 작성한 프로그램에 대하여 이루어졌으나 본 연구는 웹 프로그래밍 언어인 ASP로 작성된 프로그램을 대상으로 하였다. 또한 노력의 간접적 측정 매트릭으로 LOC를 적용하였는데 이것은 Li & Henry의 유지보수 노력의 측정 매트릭으로 변경한 라인 수를 사용한 것이나 Ronchetti의 개발노력 대신 규모를 사용한 것과 같은 견해이다.

3. 웹 소프트웨어 구조와 위험도

본 절에서는 웹 소프트웨어의 구조적 특성 및 규모와 복잡도의 계산 방법, 그리고 ASP 프로그램 중 결함 가능성이 높은 프로그램을 추출하기 위한 위험분석 매트릭을 제시한다.

3.1 웹 소프트웨어의 구조

웹 소프트웨어는 보통 웹 사이트로 불리며 구조를 설계 관점(정보, 그래픽, 네비게이션등)이나 페이지 형식적(formatting)관점, 또는 기술적(technical) 관점의 구분등 여러 가지가 있다[12]. 본 연구에서는 소프트웨어 자체의 프로그램 요소를 중심으로 분류한다.



(그림 1) 웹 소프트웨어 구성요소

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[13]. 따라서 웹 소프트웨어의 규모와 복잡도를 측정하려면 서버 스크립트 요소, 클라이언트 스크립트 요소, HTML 요소들의 규모와 복잡도를 동시에 고려하여 각각의 합으로 정의되어야 한다[14].

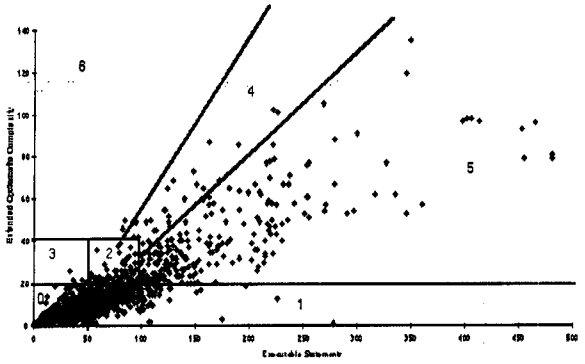
이를 식으로 표현하여 웹 소프트웨어 규모를 Lweb이라 하면, Lweb=Ls+Lc+Lh이 된다. 여기서 Ls는 서버스크립트 LOC, Lc는 클라이언트스크립트 LOC, Lh는 HTML의 LOC를 말한다. 또한 웹 소프트웨어 복잡도를 Cweb이라 하면, Cweb=Cs+Cc+Ch 로 나타낼 수 있다. Cs와 Cc는 서버 스크립트 요소와 클라이언트 스크립트 요소의 순환복잡도로 전통적인 순환복잡도와 같은 방법으로 측정되며 Ch는 HTML 웹 문서의 복잡도로 링크의수 + 1 로 측정한다.

3.2 위험분석 매트릭

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[8].

위험분석 매트릭은 NASA의 SATC(Software Assurance Technical Center)에서 개발한 유지보수성 측정에 사용하기 위한 그레픽 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



(그림 2) 위험분석 매트릭

결함과의 상관관계에서 추출한 통계 분석에 근거한 것이다. 규모(LOC)와 순환복잡도 값이 큰 영역4(중상), 영역5(상), 영역6(최상)에 분포한 프로그램은 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].

본 연구에서는 기존에 이미 C나 C++언어에서 검증된 위험분석 매트릭을 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 개발 노력과 시간의 집중이 이루어 질 것이라는 가정이 성립하며, 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 규모와의 선형회귀 분석을 통한 예측 모델을 제안한다.

4. 통계 분석

통계 분석을 위한 접근 과정은 첫째, 분석 대상 파일을 선정하기 위해 실무에서 사용되고 있는 웹 소프트웨어를 위험분석 매트릭으로 측정하고 결함 가능성이 높은 프로그램들을 추출한다. 둘째, 선정된 파일의 LOC와 NOC, NOM과의 선형회귀 분석 결과를 비교 제시하고, 셋째, 각 프로젝트별 프로그램 유형을 분류하여 유형에 따른 적정한 상관 모델을 제안한다.

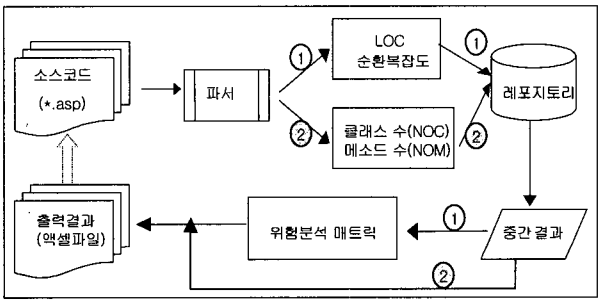
4.1 분석 대상 및 환경

본 연구는 웹 소프트웨어 언어로서 쉽게 접할 수 있고, MS Windows 환경에서 가장 많이 사용되는 언어 중 하나인 ASP로 작성된 웹 소프트웨어 프로젝트를 분석 대상으로 채택하였다. 연구에 사용된 웹 소프트웨어는 서버 스크립트로 VBScript를 사용하고 클라이언트 스크립트로 JavaScript를 사용하는 소프트웨어를 대상으로 하였다.

분석 대상은 실제 산업 현장에서 사용되고 있는 3가지 웹 사이트 프로젝트로 중고차 매매를 하는 인터넷 경매 사이트(이하 P1), 게시판 기능 중심의 대학 홈페이지(이하 P2), 기업의 수발주 단위 업무를 처리하는 프로젝트(이하 P3) 등이 포함되어 있다. 프로젝트의 총 파일 수는 1,768개이며 여기에서 분석에 사용된 ASP 파일 수는 877개, 이 중 결함 가능성이 높은 프로그램으로 객체속성 분석 파일은 39개이다. 이에 관한 프로젝트별 파일수가 <표 1>에 나타나있다.

〈표 1〉 분석 대상 프로젝트

	대상 프로젝트	총 파일수	규모분석 소스파일 수(*.asp)	객체속성분석 파일수
P1	인터넷 경매	263	222	12
P2	대학 홈페이지	757	340	4
P3	수주/발주 업무	748	315	23
	프로젝트 합계	1,768	877	39



(그림 3) 규모 예측 매트릭 분석 구성도

통계 분석을 위해 구현한 측정 도구 시스템의 구성은 (그림 3)과 같이 ASP 소스 파일을 입력하여 파싱 처리 후 LOC와 순환복잡도를 측정하여 그 결과를 레파지토리에 저장한다. 레파지토리에 저장된 정보를 사용하여 주어진 질의를 통해 추출된 결과를 출력한다. 이 결과 값에 위험분석 매트릭을 적용하여 웹 소프트웨어의 구조상 웹 소프트웨어 위험도가 서버 위험도와 5이상의 차이를 보이는 프로그램을 추출한다(그림 3의 ①과정). 이렇게 추출된 프로그램은 다시 파싱 처리하여 NOC, NOM을 측정한다. 마찬가지로 그 결과를 레파지토리에 저장하고 질의를 통해 결과를 출력한다. 이 측정 결과를 쉽게 읽고 시각적 통계처리가 가능하도록 액셀 쉬트에 옮겨 작업한다(그림 3의 ②과정).

여기서 ASP 프로그램의 클래스 수와 메소드 수의 추출에 대한 간단한 설명을 위하여 예제 프로그램이 (그림 4)에 표시되었다. 'document.write()', 'today.getdate()'와 같이 "." 를 기준으로 전반부에 있는 것은 클래스로 클래스 수(NOC)

```
<!-- ASP 예제 파일 sample.asp
.............................
-->

<html>
<head><title>Cookie 사용</title>
<script language="javascript">
 <!--
   document.write("<h3>*** 오늘의 날짜 ***</h3>");
   today = new Date()
   document.write("<b>오늘 날짜 : </b>",today.getMonth()+1, "/" ,
                  today.getDate(), "/", today.getYear());
 -->
</script>
</head>
```

(그림 4) ASP 예제 프로그램

는 2, "."후반부의 함수에 해당하는 것은 메소드로 메소드 수(NOM)는 4로 정의하였다. 각 프로젝트별 LOC와 NOC, LOC와 NOM의 선형회귀 분석과정이 다음 절에 제시된다.

4.2 선형 회귀 분석

3개 프로젝트의 객체속성 분석 파일인 39개 파일의 기술적 통계치에 대한 분석이 <표 2>에 나타나 있다. LOC는 웹 소프트웨어의 총 라인 수, CC는 순환 복잡도를 나타낸다. 표에 나타난 통계치는 액셀쉬트를 이용하여 계산된 것으로 min은 최소값, max는 최대값, mean은 평균, std.dev는 표준편차로 $\sqrt{(n\sum x^2-(\sum x)^2)/n(n-1)}$ 의 계산식에 의해 산출된 것이다.

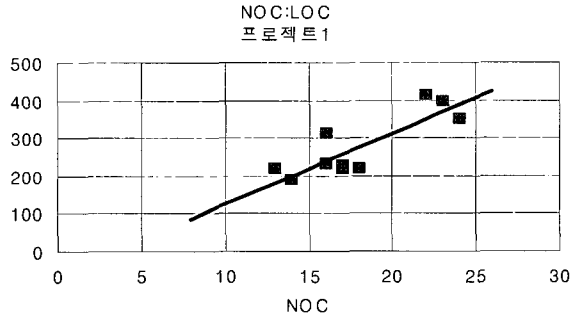
〈표 2〉 3프로젝트의 기술적 통계표

		min	max	mean	std.Dev
P1	LOC	192	414	270.8	77.9
	CC	33	83	44.9	15.6
	NOC	13	24	17.8	3.4
	NOM	25	53	36.4	8.8
P2	LOC	170	303	254.0	57.9
	CC	28	53	44.5	11.4
	NOC	4	14	10.3	4.8
	NOM	4	45	26.0	22.1
P3	LOC	124	284	172.1	36.4
	CC	21	40	29.3	5.1
	NOC	6	13	9.8	2.2
	NOM	6	21	14.6	4.3

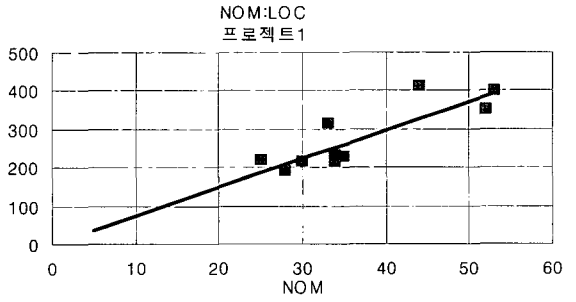
각 프로젝트 별 LOC와 NOC, LOC와 NOM과의 상관관계를 선형 회귀식으로 표현하면 다음과 같다. 프로젝트1은 LOC_{1C}=18.767NOC−63.934, 결정계수 R^2=0.6845이며, LOC_{1M}=7.4511NOM−0.5936, 결정계수 R^2=0.7081로 그래프가 (그림 5), (그림 6)에 나타나있다.

프로젝트2는 LOC_{2C}=3.9607NOC+218.02, 결정계수 R^2=0.1035이며, LOC_{2M}=1.5062NOM+214.84, 결정계수 R^2=0.3294로 그래프는 생략한다.

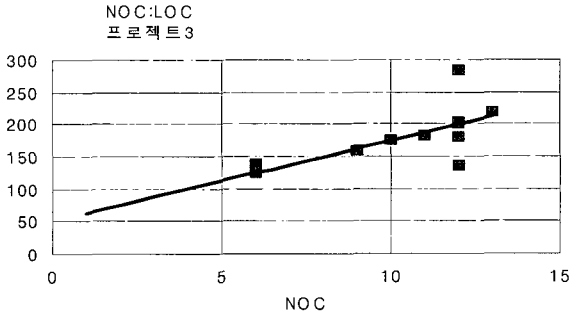
프로젝트3은 LOC_{3C}=12.561NOC+49.248, 결정계수 R^2=0.5957이고, LOC_{3M}=7.1253NOM+68.038, 결정계수 R^2=0.7211로 그래프가 (그림 7), (그림 8)에 나타나있다.



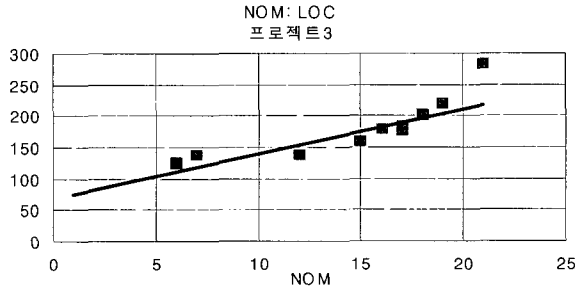
(그림 5) 프로젝트1의 NOC와 LOC의 선형관계



(그림 6) 프로젝트1의 NOM과 LOC의 선형관계



(그림 7) 프로젝트3의 NOC와 LOC의 선형관계



(그림 8) 프로젝트3의 NOM과 LOC의 선형관계

프로젝트1은 결정계수 R^2이 0.7정도로 NOC와 NOM이 LOC와 상당한 관련이 있는 것으로 나타나며, 프로젝트2는 R^2 이 0.4이하의 값을 가지므로 관련이 적은 것으로 분석된다. 또한 프로젝트3은 NOM은 R^2이 0.7이상으로 관련이 있으나 NOC는 관련이 적은 것으로 나타났다.

4.3 파일 유형에 따른 모델

각 프로젝트별 LOC와의 상관정도를 나타내는 결정 계수 값이 다르게 나타남에 따라 위험도를 증가시키는 프로그램 유형을 분석하였더니 다음의 세 가지 유형으로 나타났다. 링크 요소가 많은 홈페이지나 사이트 맵을 구성하는 HTML 소스로서의 menu 파일, 주로 입력 양식을 구성하는 클라이언트 자바 스크립트 소스로서의 form 파일, 복잡한 사용자 인터페이스를 위한 클라이언트 소스로서의 control 파일등이다. 각 프로젝트별 파일 유형은 <표 3>과 같이 구성되었는데 프로젝트1은 form 파일로 이루어졌으며, 프로젝트2는 form 파일과 menu 파일, 프로젝트3은 form 파일과 Control 파일이 존재하였다. form 파일로만 구성된 프로젝트1이 높은 상관관계를 나타내어 프로젝트 2, 3도 menu파일과 control파일을 제외하고 회귀분석을 하였더니 높은 상관정도를 나타냈다.

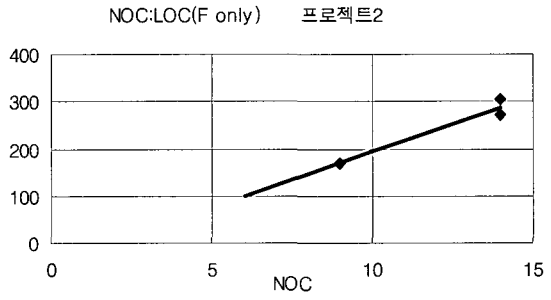
〈표 3〉 프로젝트별 파일 유형

	프로젝트 1	프로젝트 2	프로젝트 3
Menu 파일	0	1	0
Form 파일	12	3	20
Control 파일	0	0	3
합 계	12	4	23

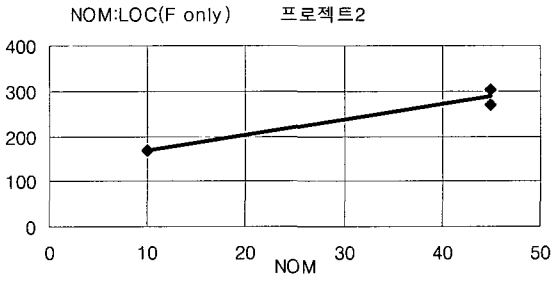
전체 파일과 menu파일이나 control 파일을 제외한 form 파일의 선형회귀식의 비교가 <표 4>에 나타나 있다. 프로젝트2는 NOC, NOM과의 결정계수 R^2=0.9469의 동일한 값으로 매우 높은 상관관계를 보였고, 프로젝트3은 NOC와의 결정계수 R^2=0.7558, NOM과의 결정계수 R^2=0.8139로 통합 유형보다 훨씬 높은 상관관계를 나타냈다.

프로젝트2의 form파일 선형회귀 그래프가 (그림 9), (그림 10)에 나타나 있고, 프로젝트3은 (그림 11), (그림 12)에 표시되어 있다.

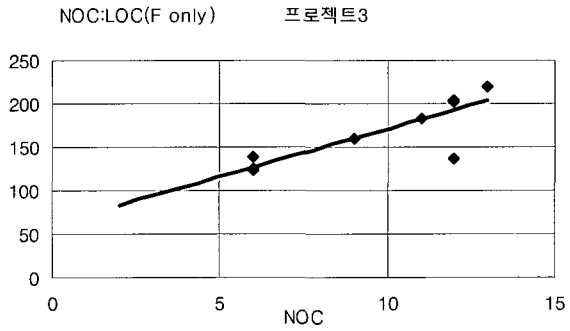
이와 같이 웹 소프트웨어의 구조는 서버, 클라이언트, HTML등의 파일이 통합 구성되어 있으므로 LOC와 NOC, LOC와 NOM의 관련성을 찾기가 쉽지 않았다. 이것은 웹 소프트웨어 구조상 HTML로 작성된 menu 파일 유형과 복



(그림 9) 폼파일의 NOC:LOC선형관계 (P2)



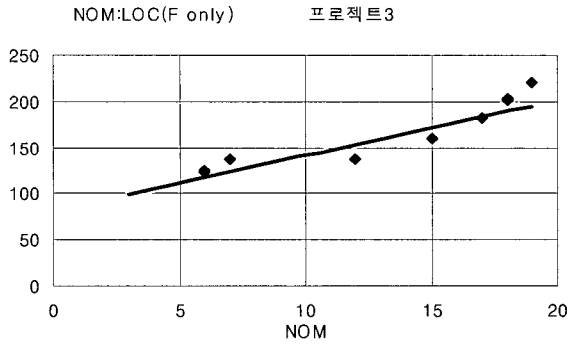
(그림 10) 폼파일의 NOM:LOC 선형관계(P2)



(그림 11) 폼파일의 NOC:LOC 선형관계(P3)

〈표 2〉 3프로젝트의 기술적 통계표

	y \ x	NOC	NOM	NOC(F)	NOM(F)
P1	LOC	y=18.767×-63.934	y=7.4511×-0.5936	y=18.767×-63.934	y=7.4511×-0.5936
	R^2	0.6845	0.7081	0.6845	0.7081
P2	LOC	y=3.9607×+218.02	y=1.5062×+214.84	y=23.4×-40.6	y=3.3429×+136.57
	R^2	0.1035	0.3294	0.9469	0.9469
P3	LOC	y=12.561×+49.248	y=7.1253×+68.038	y=10.971×+61.18	y=5.9703×+81.769
	R^2	0.5957	0.7211	0.7558	0.8139



(그림 12) 폼파일의 NOM:LOC 선형관계(P3)

잡한 인터페이스의 클라이언트 control 파일 유형 등이 방해 요소로 작용한 것으로 보이고 위험도 차이가 높은 웹 소프트웨어 구성의 90%이상을 차지하는 자바스크립트로 작성된 입력 중심의 form파일 유형이 매우 높은 관련성을 나타낸 것으로 분석되었다. 즉 웹 소프트웨어의 자바스크립트 form 파일 유형은 NOC나 NOM으로 규모 예측이 가능하며 그 중에서 특히 NOM이 더 밀접한 관련을 보이므로 예측모델에 적합함을 제안한다.

5. 결 론

본 연구는 웹 소프트웨어의 규모를 예측하기 위하여 ASP로 작성된 3개 웹 프로젝트에 대하여 분석단계에서 파악할 수 있는 클래스 수나 메소드 수가 상관관계가 있는지 파악하여 파일 유형에 따른 적정 모델을 제안하였다.

분석 대상 자료로 웹 소프트웨어와 서버 스크립트 복잡도의 차이가 5이상인 결함가능성이 높은 프로그램이 선택되었고 복합구조를 가진 웹 소프트웨어의 클라이언트나 HTML 파일 유형에 따라 상관 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. 그러므로 웹 소프트웨어의 규모를 예측하기 위해서는 분석대상 프로그램 파일 유형 중 90% 이상을 차지하는 form 파일 유형의 LOC를 예측하는 것이 정확도를 높일 수 있으며 메소드 수(NOM)가 클래스 수(NOC)보다 더 높은 상관관계를 가짐을 알 수 있었다.

분석대상 자료의 제한으로 제안 모델의 일반적인 적용이 어렵다는 한계가 존재하나, 웹 소프트웨어에 대한 규모 예측에 객체 속성의 사용을 시도했다는 것과 웹 소프트웨어의 특정 파일 유형에 적합한 예측모델을 제시했다는 것에 의의가 있다.

향후 웹 소프트웨어의 모든 파일 유형에 적합한 예측모델에 대한 연구가 진행되어야 하며 규모 뿐 아니라 복잡도를 예측할 수 있는 객체 속성을 추출하여 개발 노력과 시간의 예측을 좀 더 정확하게 하고 분석 대상 자료를 확대하여 일반적인 적용이 가능하도록 보다 많은 연구가 이루어져야 할 것이다.

참 고 문 헌

[1] Marco Ronchetti, Giancarlo Succi, Witold Pedrycz, Barbara Russo, "Early estimation of software size in object-oriented environments:a case study in a CMM level3 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] V.B.Misic, D.N.Tesic, "Estimation of effort and complexity: An object-oriented case study", Journal of Systems and Software, Jan., 1999.

[4] Victor Laing and Charles Coleman, "Principal Components of Orthogonal Object-Oriented Metrics", www.gsfc.nasa.gov/support/OSMASASMSEP01/, 2001. 10.

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

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

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

[10] 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.

[11] 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.

[12] 김지현, 박철, "웹 어플리케이션의 모듈위험수준 측정도구의 구현", 한국컴퓨터정보학회논문지 제7권 제2호, 2002. 6.

[13] 안종근, 유해영, "웹 어플리케이션의 순환복잡도 매트릭에 관한 연구", 정보처리학회논문지D, Vol.9-D, No.3, pp.447-456, 2002. 06.

[14] 박철, 유해영, "웹 어플리케이션의 순환복잡도 분석", 정보처리학회논문지D, Vol.11-D, No.4, pp.865-872, 2004. 08.

김 지 현

e-mail : jhkim@seoil.ac.kr

1978년 이화여자대학교 수학과(이학사)

1994년 단국대학교 전자정보학과 (경영학석사)

2001년 단국대학교 정보통신대학원박사 과정수료

1997년~현재 정보관리기술사

1998년~현재 서일대학 소프트웨어전공 조교수

관심분야 : 소프트웨어 공학, 웹 공학, 프로젝트관리, 품질 평가

유 해 영

e-mail : yoohy@dankook.ac.kr

1979년 단국대학교 수학과업(이학사)

1982년 단국대학교 대학원수학과수료 (이학석사)

1994년 아주대학교 대학원컴퓨터공학과 수료(공학박사)

1983년~현재 단국대학교 정보컴퓨터학부 교수

관심분야 : 소프트웨어 공학, 시스템 프로그램