韓國 컴퓨터情報學會 論文誌
第9卷 第3號, 2004. 9.

2004-9-3-1-4

웹 어플리케이션의 복잡도 예측에 관한 연구

오 성 균*, 김 미 진**

A Study of Estimation for Web Application Complexity

Sung-Kyun Oh*, Mi-Jin Kim**

요 약

개발 패러다임이 점차 복잡한 웹 환경으로 전환되면서 복잡도에 대한 연구가 다시 활발해지고 있으나 아직 웹 어플리케이션의 구조나 복잡도 측정 매트릭에 정립된 이론이 부족한 실정이다. 또한 전통적 복잡도를 측정하는 프로그램 규모(LOC)나 순환복잡도 매트릭은 구현 후에나 알 수 있어 소프트웨어 개발주기 초기의 분석 및 설계 단계에는 큰 도움을 주지 못하고 있다. 본 연구에서는 실무에서 사용되는 6개 웹 프로젝트에 복잡도 인디케이터를 적용하여 결함 가능성이 높은 어플리케이션을 추출한다. 추출한 61개의 프로그램을 대상으로 복잡도와 클래스 수 및 메소드 수에 대한 선형적 상관관계를 제안함으로써 웹어플리케이션의 복잡도를 구현 전에 미리 예측 가능하도록 하여 개발 프로세스의 인적 자원 관리나 비용 예측에 기여하고자 한다.

Abstract

As software developing paradigm has been changing to complicate Web environment, study of complexity becomes vigorous. Yet still it seems that general agreement has not to be reached to architecture or complexity measure of Web application. And so traditional complexity metrics - program size(LOC) and Cyclomatic Complexity can be derived from the source code after implementation, it is not helpful to the early phase of software development life cycle - analysis and design phase. In this study 6 Web projects has been used for deriving applications with possible errors suited by Complexity Indicator. Using 61 programs derived, linear correlation between complexity, number of classes and number of methods has been proposed. As Web application complexity could be estimated before implementation, effort and cost management will be processed more effectively.

▸ Keyword : Software Complexity, Software Metrics, Web Estimation

• 제1저자 : 오성균
• 접수일 : 2004.08.13, 심사완료일 : 2004.09.01
본 논문은 서일대학 과학기술 연구소의 연구지원금에 의한 연구실적물임.
* 서일대학 IT계열 소프트웨어전공 부교수
** 서일대학 IT계열 소프트웨어전공 강사

Ⅰ. 서 론

소프트웨어를 측정하는 방법에는 세 가지가 있는데 프로세스 메트릭스, 프로덕트 메트릭스, 자원 메트릭스 등이 그것이다. 프로세스 메트릭스는 개발, 유지보수, 테스팅 프로세스의 측정에 사용되며 투입된 노력(efforts), 비용, 수행된 작업, 시간 등의 측정이 포함된다. 자원 메트릭스는 하드웨어, 소프트웨어, 인력등의 측정에 사용되며 성능, 가용성, 신뢰도, 생산성 특성이 속한다. 프로덕트 메트릭스는 프로그램, 컴포넌트, 프로젝트, 데이터베이스와 같은 소프트웨어 산출물의 특성을 측정하는데 사용되며 규모(size), 복잡도(complexity), 품질(quality)등의 특성을 갖는다.

Fenton은 또한 내부 속성과 외부 속성을 구분했는데 프로덕트, 프로세스, 자원의 내부 속성은 순수하게 그 자체로써 측정될 수 있는 것들이고 외부속성은 프로덕트, 프로세스, 자원이 어떻게 환경과 관계를 갖는지 측정하는데 사용될 수 있다고 하였다[2]. 소프트웨어 프로덕트의 외부속성은 신뢰도, 보안, 사용성, 성능 등이 있고, 내부 속성은 복잡도, 모듈러리티, 시험성(testability), 재사용성(reusability)등이 있는데 이들은 소스 코드 자체를 시험함으로써 측정될 수 있다. 본 논문에서는 노력(efforts)프로세스와 인적 자원 등의 예측에 영향을 미치는 프로덕트 내부 속성인 규모와 복잡도, 품질 등의 특성에 대한 상관관계에 대해 연구하고자 한다.

소프트웨어 메트릭에 관한 많은 연구가 진행되고 있으나, 대부분의 프로그램이 웹 기반으로 작성되고 있으므로 웹 어플리케이션의 구조에 적합한 메트릭에 대해 연구가 필요하다. 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[1].

본 연구에서는 전통적 복잡도로서의 프로그램 규모(LOC, Lines of Code)와 소스 코드의 논리적 복잡도에 기반한 순환복잡도(Cyclomatic Complexity)의 선형 관계에 의한 복잡도인디케이터를 적용하여 결함 가능성이 높은 프로그램을 추출한다. 복잡도 인디케이터는 NASA의 SATC(Software Assurance Technology Center)에서 실무 프로젝트의 결함 수 등을 조사하여 만든 경험적 지시자이다. 그러나 웹 어플리케이션의 구조적 특성상 서버, 클라이언트, HTML등 세 분야 어플리케이션의 복잡도를 동시에 고려해야 하며 웹 어플리케이션의 객체지향 특성 측정을 위해 클래스 수와 메소드 수를 추출한다. 즉 어플리케이션의 크기(LOC)와 순환복잡도에 의해 결정되는 복잡도 인디케이터에 따라 위험도가 높은 어플리케이션, 즉 결함이 가장 많이 발생할 수 있는 어플리케이션을 추출하는데 서버측 복잡도와 서버, 클라이언트, HTML 세 분야를 통합한 복잡도(CCWA)의 차가 5이상인 어플리케이션을 선정한다. 여기서 클래스 수와 메소드 수를 추출하여 LOC, 순환복잡도, 클래스 수, 메소드 수의 상관관계를 파악함으로써 전통적 특성과 객체지향 특성을 함께 가지고 있는 웹 어플리케이션의 복잡도를 연구한다. 즉 규모 및 순환복잡도가 클래스 수, 메소드 수와 어떤 관계가 있는 지 조사함으로써 만약 이들이 선형적 관계에 있다면 웹 어플리케이션의 분석 설계 단계에서 클래스 수(Number Of Classes, NOC)와 메소드 수(Number Of Methods, NOM)를 파악하면 구현 후에나 파악할 수 있었던 프로그램 규모와 복잡도를 미리 예측함으로써 노력 프로세스 및 인적 자원을 효율적으로 관리하고 궁극적으로 품질 개선에 기여하고자 한다.

Ⅱ. 웹 어플리케이션의 구조와 복잡도 인디케이터

2.1 전통적 복잡도

구조적 방법론에서 사용되는 대표적 복잡도 측정 지표에 프로그램 규모와 순환 복잡도가 많이 사용된다.

2.1.1 프로그램 규모(LOC)

프로그램 규모는 소스코드 라인 수로 프로그램 기능에 영향을 미치지 않는 주석을 제외한 실행 가능한 라인 수를 측정한다.

만약 기능에 의해서만 프로젝트의 비교가 이루어진다면 작은 규모의 프로젝트가 우월한 설계와 적은 유지보수를 요구한다. 또한 큰 규모의 프로그램은 이해성(Understandability), 재사용성(Reusability), 유지보수

성(Maintainability) 등에서 높은 위험을 지닐 것이다[4].

2.1.2 순환복잡도

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[5].

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

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

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

$V(G) = DE$ 이다. DE는 프로그램의 흐름을 변경시킬 수 있는 프로그램 명령어를 의미하며 IF, SELECT 문 등의 분기 명령과 FOR, WHILE, LOOP 등의 반복 명령어를 의미한다[12].

2.2 객체지향 복잡도

객체지향 개발 주기는 보통 전통적 개발주기와 비슷하다. 단지 분석, 설계 활동이 객체지향 개발에서는 쉽게 분리되지 않는다. Booch는 개발주기를 다음과 같이 단계별로 정의하였다. 먼저 문제 영역의 핵심 추상화로써의 클래스와 객체를 정의하고, 두 번째, 클래스와 객체의 의미, 즉 구조(structure)와 행위(behaviour)를 정의하고, 셋째, 클래스, 객체 및 그들의 상호작용 정의 및 시스템 전체 행위를 제공하기 위한 공동 작업 시나리오 정의, 상속 구조의 정의, 분석 및 정제를 하고 넷째, 구조와 행위 상세를 포함하는 클래스와 객체의 구현 단계로 구분한다[7].

이와 같은 단계별 작업은 다른 방법론 즉 OOA(Coad & Yourdon, 1991), OMT (Rumbaugh, 1991)뿐 아니라 다른 단계별 작업을 포함하는 use-case 중심 접근방법(Jacobson, 1994))에서도 제안되고 있다[3]. 이러한 단계는 선형적이 아니라 동시적 특성을 가지므로 설계단계에서 클래스의 수를 발견할 수 있다.

객체지향 설계는 여러 단계의 추상화 수준을 가지는데 상위단계에 클래스와 상호작용이 있고 대부분의 클래스는 상속계층에서 엉켜있다. 그러므로 관리하는 추상화 수준에 따라 가능한 측정을 분류할 수 있다.

먼저 메소드나 서비스의 복잡도를 모델링하는 메소드 수준에서의 측정을 들 수 있다. 거의 모든 프로그래밍 언어에서 볼 수 있는 것처럼 지역변수를 공유하고 단위로 참조된 수행가능 구문집단과 같이 McCabe의 순환복잡도와 같은 복잡도를 사용하는 것이 자연스럽게 보인다. 둘째 클래스 전체의 특성을 모델링하는 클래스 수준의 측정이 있다. 이것은 메소드와 속성의 수를 포함하며, 지역 대 전역 메소드/변수의 비율, 메소드에 호출된 외부 메소드의 분포에 따라 측정될 수 있는 응집도등이 있다. 셋째 상속 계층 측정이 있는데, 상속 계층의 깊이와 높이, 상위 클래스로부터 상속된 메소드의 수와 하위 클래스 내에 정의된 메소드 수의 측정을 말한다. 넷째 클래스들 사이의 상속이 아닌 클래스 연결 정보를 들 수 있다. 클래스 연결은 메소드 호출에 따라 정적이거나 동적이 될 수 있다. 다섯째 프로젝트를 완성하기 위한 시간과 노력은 시스템 자체의 규모에 좌우된다. 즉 클래스의 수, 메소드의 수, 상속 계층의 수와 크기, 다른 타입의 내부 클래스 연결의 수 등이 있다[3].

V. B. Misic의 논문에서는 객체지향 사례분석을 통해 어플리케이션의 사이즈와 클래스 수 및 메소드 수의 선형적 상관관계를 분석함으로써 노력을 예측하고 있다.

2.3 웹 어플리케이션의 구조

웹 어플리케이션은 즉시성, 동시성, 광역성, 복잡성 등으로 인해 관리의 중요성이 가중되지만, 동시에 그러한 특성이 관리를 어렵게 하고 있다. 이에 Powell은 다음과 같은 공학적 해결책을 지적했는데, '소프트웨어 산출물의 품질 속성은 어플리케이션의 구조를 분석하여 사용성과 유지보수성의 문제를 파악하고, 그에 따른 오류 모듈이 정의되어야 한다'고 하였다[6].

웹 어플리케이션은 기존의 어플리케이션과 달리 여러 프로그램 요소들이 결합되어 있다. 웹 어플리케이션 구성요소는 (그림 1)에 나타나 있는 것처럼 웹 서버에 존재하는 웹 어플리케이션 소스에는 서버에서 스크립트 엔진에 의해서 실행되는 서버 스크립트 요소(SSS : Server side Script)와 클라이언트에 전송되는 클라이언트 스크립트 요소(CSS :

Client side Script)및 HTML 태그로 구성된 HTML 요소들이 포함되어 있다[11]. 따라서 웹 어플리케이션의 복잡도를 측정하려면 SSS 요소, CSS 요소, HTML 요소들의 복잡도를 동시에 고려하여야 한다.

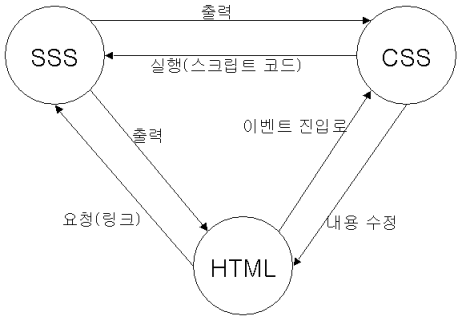

그림 1. 웹 어플리케이션 구성요소
fig 1. Components of Web Application

CCWA 매트릭은 웹 어플리케이션의 순환복잡도 매트릭으로 웹 어플리케이션을 구성하는 SSS, CSS, HTML 요소들의 복잡도의 합으로 다음과 같이 정의 된다. 아래 식에서 C(SSS)와 C(CSS)는 서버 스크립트와 클라이언트 스크립트 요소의 순환복잡도로 전통적인 순환복잡도와 같은 방법으로 측정된다. C(HTML)은 웹 문서의 복잡도로 링크의 수+1 로 측정한다[10].

CCWA = C(SSS) + C(CSS) + C(HTML)
C(SSS) : 서버 스크립트 요소의 순환복잡도
C(CSS) : 클라이언트 스크립트 요소의 순환복잡도
C(HTML) : HTML 요소의 순환복잡도

물론 웹 어플리케이션은 보통 웹사이트로 불리는데 웹 사이트의 구조 분석에도 여러 가지 관점이 있다. Ivory는 웹 사이트 구조를 Text Elements와 Formatting, Link Elements와 Formatting, Graphic Elements와 Formatting 및 Page formatting으로 보았다[8]. 또한 Miller는 웹 사이트 구조를 Browser, Display Technology, Navigation, Object Mode, Server Response, Interaction & Feedback 및 동시 사용자 등으로 분석하였다[9]. 이렇게 웹 사이트 구조에 대한 이론이 정립되어 있지 않고, 테스팅 방법론이나 평가 매트릭 및 웹 사이트에 적합한 복잡도 매트릭의 개발은 더 더욱 이루어지지 못하고 있는 실정이다.

V. B. Misic의 연구에서 사이즈와 클래스 수 및 메소드 수의 상관관계를 분석하였으나 본 논문에서는 복잡도와 클래스 수 및 메소드 수의 상관관계를 웹 어플리케이션의 CCWA 구조[10]에 적용하여 4절에서 언급할 NASA에서 제안한 복잡도 인디케이터에서 특별히 복잡도가 높게 나타나는 결함가능성이 높은 모듈을 추출하여 복잡도와 클래스 수 및 메소드 수의 상관관계를 분석하여 노력의 예측자료로 제안하려한다.

2.4 인디케이터

복잡도 인디케이터는 NASA의 SATC(Software Assurance Technical Center)에서 개발한 유지보수성 측정에 사용하기 위한 그래픽 템플릿이다. 즉 결함 가능성이 높은 프로그램이 어떤 프로그램인지를 알려줌으로서 유지보수를 도와주는 척도가 되는 그래프인 것이다. X축은 코드 라인의 수를 나타내고 Y축은 확장된 순환 복잡도(시험 경로의 수)를 나타낸다.

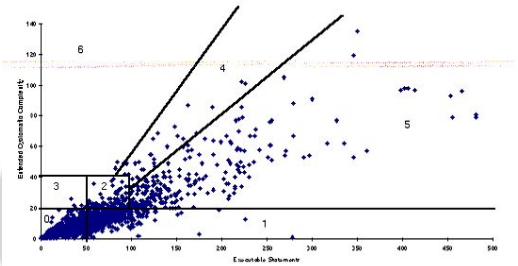

그림 2 복잡도 인디케이터
fig 2. Complexity Indicator

위의 그래프를 보면 복잡도를 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 순환복잡도의 경우 20과 40, 그리고 오른쪽 위로 증가하는 두개의 사선(영역4, 영역5, 영역6을 구분) - 은 NASA와 여러 기업의 소스 프로그램으로부터 SATC에 의해 결함과의 상관관계에서 추출한 가이드 라인에 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 실험적 통계에 의하여 추출하였다. 그러므로 이 영역에 분포하는 프로그램은

특별 관리가 필요하며, 개발자들은 이 영역에 분포한 프로그램의 결함에 대해 더욱 조사해야 한다고 분석하였다[4][12].

본 연구에서는 기존에 이미 구조적 언어(C 언어)와 객체지향 언어(C++ 언어)의 실험 분석에 의하여 검증된 복잡도 인디케이터에 따라 서버측 복잡도 C(SSS)와 웹 어플리케이션의 복잡도 CCWA의 차가 5이상인 결함 가능성이 높은 어플리케이션을 추출하여 클래스 수와 메소드 수를 측정함으로써 순환 복잡도와 클래스의 상관관계를 파악하고 객체지향 개발주기의 첫 번째 단계에서 클래스 수가 결정되므로 개발주기 초기 단계에서 어플리케이션의 복잡도를 예측할 수 있고 따라서 노력 프로세스와 소요 인력 등의 예측에 도움을 줄 수 있다.

III. 웹 어플리케이션의 복잡도 측정

본 절에서는 실무에서 사용되고 있는 ASP 어플리케이션을 복잡도 인디케이터로 측정하여 결함 가능성이 높은 프로그램들을 추출하여 클래스 수(NOC), 메소드 수(NOM)와의 상관관계를 제시한다.

3.1 측정 실험 대상 및 과정

본 논문은 웹 어플리케이션 언어로서 쉽게 접할 수 있고, MS Windows 환경에서 가장 많이 사용되는 언어 중 하나인 ASP로 작성된 웹 어플리케이션 프로젝트를 실험 대상으로 채택하였다. 연구에 사용된 웹 어플리케이션은 서버측 스크립트로 일반적으로 많이 사용하는 VBScript를 사용하고 클라이언트측 스크립트로 JavaScript를 사용하는 것들을 대상으로 하였다.

실험은 총 6개의 웹 프로젝트를 대상으로 하였다. 게시판 기능 중심의 대학 홈페이지에서부터 기업의 단위 업무를 처리하는 프로젝트, 개발회사에서 품질관리와 프로젝트 관리를 위해 사용하는 시스템 등이 포함되어 있다. 6개 프로젝트의 총 파일 수는 4,968이며 여기에서 실험에 사용된 ASP 파일 수는 1,574이다. 실험에 사용된 프로젝트별 총 파일수, LOC와 순환복잡도 추출 소스 파일수, 클래스 추출 파일수 등을 <표 1>에 설명하였다.

표 3. 실험 대상 프로젝트
Table 1. Description of Source Project

	실험대상 프로젝트	총 파일수	복잡도추출파일수(*.asp)	클래스추출파일수
A	대기업의 전산 지원 관리	831	243	8
B	인터넷 경매	263	222	12
C	게시판 중심의 대학 홈페이지	757	340	4
D	일반 업무처리	748	315	23
E	품질관리	622	269	8
F	프로젝트 관리	1,747	185	6
	프로젝트 합계	4,968	1,574	61

구현한 측정 도구 시스템의 구성은 아래 (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 복잡도 인디케이터를 적용하여 웹 어플리케이션의 구조상 CCWA 복잡도가 C(SSS) 서버측 복잡도와 큰 차이를 보이는 프로그램을 추출한다. 이렇게 추출된 프로그램은 다시 파싱 처리하여 클래스 수(Number Of Classes, NOC)와 메소드 수(Number Of Methods, NOM)를 측정한다. 마찬가지로 그 결과를 레파지토리에 저장하고 질의를 통해 결과를 출력한다. 이 측정 결과를 쉽게 읽고 시각적 통계처리가 가능하도록 액셀 쉬트에 옮겨 작업한다. 6개의 프로젝트에서 CCWA 복잡도가 C(SSS) 서버측 복잡도와 5 이상 차이를 보이는 61개 프로그램에서 NOC(Number Of Classes), NOM(Number Of Methods) 값을 추출하여 설정한 가정에 부합하는지 선형 회귀 식을 적용하여 실험한다.

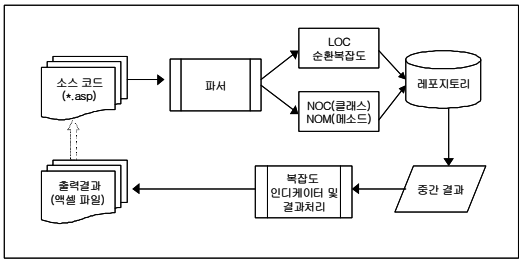

그림 3. 복잡도 예측 매트릭 측정실험 구성도
fig 3. Experimental measure process

3.2 복잡도 인디케이터에 의한 측정

기존의 복잡도 매트릭은 서버측 어플리케이션 복잡도가 선택될 수 있기 때문에 C(SSS) 서버 스크립트 복잡도와

CCWA의 복잡도 레벨 차이가 큰 것을 주 대상으로 하였다. CCWA 복잡도는 서버, 클라이언트, HTML 각각의 순환복잡도를 합산한 것이므로 C(SSS) 서버측 복잡도와 큰 차이가 있는 즉, 레벨 차이가 5이상인 CCWA 복잡도를 미리 예측하고자 한다. 레벨 차이가 5이상이라는 것은 CCWA 복잡도가 5나 6으로 결함 가능성이 높은 위험한 프로그램이며 C(SSS) 서버 스크립트 복잡도는 1이하로 낮다는 의미를 가지므로 C(SSS)를 구성하는 ASP 프로그램으로 CCWA 복잡도를 예측할 수 있다면 나머지 다른 어플리케이션의 복잡도도 비슷한 양상을 보일 것이다. <표 2>는 C(SSS)와 CCWA 복잡도 인디케이터 레벨을 보여주고 있으며 <표 3>은 복잡도 인디케이터 레벨 변화율을 나타내고 있다.

표 4. C(SSS)와 CCWA 복잡도 레벨
Table 2. Complexity Level Comparison

	C(SSS) 복잡도 레벨 분포(%)						CCWA 복잡도 레벨 분포(%)					
	A	B	C	D	E	F	A	B	C	D	E	F
L0	29	78	55	25	19	25	19	41	41	5	5	5
L1	58	19	43	40	32	52	54	44	39	37	25	33
L2	0	2	0	0	0	0	0	1	1	0	0	0
L3	0	0	0	0	0	0	0	0	0	0	0	0
L4	0	0	0	0	0	0	0	0	0	0	0	0
L5	12	0	2	34	49	22	27	15	19	58	70	62
L6	0	0	0	0	0	0	0	0	0	0	0	0

본 연구에서는 <표 3>의 레벨 증가율이 +5이상인 프로그램에 대해 집중 연구한다. 본 연구의 목적은 복잡도를 미리 예측할 수 있는 매트릭을 찾는 것이므로 복잡도 레벨 증가율이 큰 프로그램(즉 복잡도 레벨이 C(SSS)에 비해 +5 이상 증가한 것)을 추출하여 적합한 예측 매트릭을 적용하고자 한다.

표 5. 복잡도 레벨 증가율(단위 %)
Table 3. Increased Rate

	A	B	C	D	E	F	평균
+L0	79	53	71	62	68	44	58.8
+L1	7	32	11	14	11	17	14.4
+L2	0	1	1	0	0	0	1.1
+L3	0	2	0	0	0	0	0.5
+L4	11	7	15	17	18	36	19.4
+L5	3	5	1	7	3	3	5.6
+L6	0	0	0	0	0	0	0.2

<표 3>에 의하면 약 60%의 프로그램이 복잡도에서 변화가 없었으며 약 15%가 복잡도 3이내의 증가를 보였고 나머지 20%가 복잡도 레벨 4이상의 증가를 보였다. 약 5.6%, 61개의 프로그램이 복잡도 5이상의 증가율을 보였고 클래스와 메소드 추출 대상으로 하여 관계를 연구한다.

3.3 복잡도 상위프로그램의 클래스상관관계

복잡도 레벨 증가가 +5이상인 61개 ASP 프로그램의 클래스 수와 메소드 수를 측정하였다.

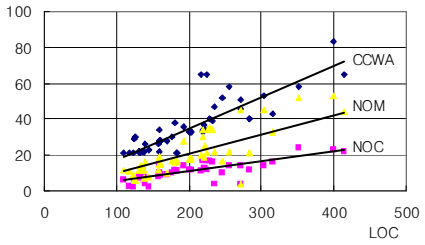

그림 4. CCWA, 메소드 수, 클래스 수의 선형적 관계
fig 4. Linear Relationship

(그림 5)는 3개의 선형 회귀 관계를 나타내고 있는데 프로그램 규모(LOC)에 대한 CCWA, LOC에 대한 메소드 수(NOM), LOC에 대한 클래스 수(NOC)의 선형적 관계를 나타내고 있다. <표 4.>는 프로그램 규모(LOC)에 대한 각각의 관계식(linear correlation)과 결정 계수(determination coefficient R2)이다.

표 4. CCWA, 클래스 수, 메소드 수의 선형회귀식
Table 4. Linear Regression Expression

	관계식	결정계수R2
LOC vs CCWA	CCWA=0.1747L	0.7165
LOC vs 클래스 수	NOC=0.0547L	0.5634
LOC vs 메소드 수	NOM=0.1046L	0.6172

위의 관계식을 보면 세 항목의 결정계수 R2 값이 모두 0.5를 넘으므로 관계가 있다고 볼 수 있다. 프로그램 규모(LOC)와 웹 어플리케이션 복잡도(CCWA)는 결정계수 R2 = 0.7165로 클래스 수나 메소드 수에 비해 높은 상관관계를 갖는다는 것을 알 수 있다. 2.2절에서 '메소드 수준에서의 복잡도 측정은 지역변수를 공유하고 단위로 참조된 수행가능 구문 집단과 같이 McCabe의 순환복잡도와 같은 복잡도를 사용하는 것이 자연스럽게 보인다.'라고 Misic이 언급한 것처럼 규모(LOC)와 메소드 수와의 상관관계(R2=

0.6172)가 규모와 클래스 수의 상관관계(R2= 0.5634)보다 더 밀접한 것을 알 수 있다. 그러나 클래스 수와의 상관관계도 전혀 없다고 볼 수 없으므로 클래스 수와 메소드 수의 합과는 어떤 관계에 있는지 조사하였다.

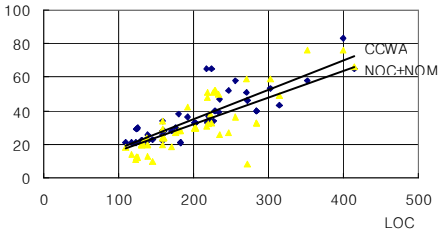

그림 5. 클래스와 메소드 누적분의 선형적 관계
fig 5. Linear Relationship for NOC+NOM

(그림 6)은 LOC에 대한 CCWA와 클래스 수 및 메소드 수의 누적분에 대한 선형적 관계도이다. 규모(LOC)에 대한 클래스 수 및 메소드 수의 합의 관계식을 MC라 하면 MC = 0.1593L 이 된다. 결정계수 R2은 0.6462이다.

이와 같이 위의 클래스와 메소드 수의 합의 관계식 MC의 기울기가 CCWA식(CCWA=0.1747L)의 기울기에 가까워 졌으며 결정계수 R2도 클래스 수나 메소드 수를 따로 따로 고려한 것보다는 증가하였다. 즉, 프로그램 규모(LOC)에 대한 클래스 수와 메소드 수를 함께 고려했을 때 CCWA 복잡도와 매우 유사한 양상을 보이므로 분석 설계 단계 초기에 클래스와 메소드의 정의에 의해 개수를 파악하여 프로그램 복잡도 예측이 가능하다.

input filename	LOC	CCWA	classes	Methods	Class+methods
p01/DataDicManageList-1.asp	117	21	3	11	14
p01/DbmsList-4.asp	145	23	2	8	10
p01/Design-2.asp	109	21	6	12	18
p01/EngineDel-5.asp	132	21	7	14	21
p01/HWManage-6.asp	123	21	2	9	11
p01/searchView-3.asp	139	26	4	16	20
p01/top-7.asp	282	290	35	43	78
p01/UserInsert-8.asp	295	47	4	22	26
p02/clip_req_member-13.asp	222	34	13	25	38
p02/forumtotalmodify-14.asp	219	34	18	30	48
p02/forumtotalwrite_reply-16.asp	219	33	17	34	51
p02/forumtotalwrite-15.asp	226	34	17	34	51
p02/order_quotation-17.asp	233	39	16	34	50
p02/orderclick1-18.asp	229	40	17	35	52
p02/orderclick2-19.asp	228	40	17	35	52
p02/orderclick3-20.asp	192	36	14	28	42
p02/W_SS119_MBR_002_MB_INS_S-9.asp	400	63	23	53	76
p02/W_SS119_MBR_003_MB_INS_S-10.asp	352	58	24	52	76
p02/W_SS119_MBR_013_MB_UPT_S-11.asp	414	65	22	44	66
p02/W_SS119_MBR_016_MB_UPT_S-12.asp	315	43	16	33	49
p03/gradu_regist_form-22.asp	303	53	14	45	59
p03/mem_insert-21.asp	170	28	9	10	19
p03/regist_form-23.asp	271	51	14	45	59
p03/sitemap-24.asp	272	46	4	4	8
p06/ref_calendar-118.asp	284	40	12	21	33
p06/sb_d_grmp_rtr-119.asp	220	37	13	19	32
p06/sb_d_gyss_rtr-120.asp	201	33	12	18	30

그림 6. 어플리케이션별 LOC, CCWA, 클래스, 메소드
fig 6. Source Table of Each Application

단지, (그림 6)에 나타난 것처럼 61개의 프로그램 중 하나의 프로그램이 기울기에서 멀어져 있는 것을 발견할 수 있는데 (그림 7)에서 C프로젝트인 대학 홈페이지의 사이트맵 프로그램이다. 이에 관한 소스 코드의 보다 자세한 연구가 요구되며 여기서는 논문의 범위를 벗어나므로 생략한다.

Ⅳ. 결론

프로그램 규모와 순환복잡도가 전통적으로 복잡도를 나타내는 척도로 사용되고 있으나 이 두 매트릭은 프로그램 구현 후에나 측정할 수가 있다. 또한 최근 소프트웨어 개발 경향이 웹 기반으로 흐르고 있는데 웹 어플리케이션은 서버측, 클라이언트측, HTML 요소가 결합되어 전통적 프로그래밍 기법과 객체지향 프로그래밍 기법을 동시에 고려하여 작성하여야 한다.

실무에서 사용하는 ASP로 작성된 6개의 웹 프로젝트에서 웹 구조상 결함이 가장 많이 발생할 수 있는 위험률이 높은 프로그램으로 서버측 복잡도와 웹 어플리케이션 복잡도의 차이가 큰 프로그램을 추출하였다. 복잡도 인디케이터가 사용되었으며 복잡도 레벨차이가 5이상인 프로그램을 추출하여 복잡도와 클래스 및 메소드 수의 누적분이 선형적 관계에 있다는 것을 연구 제안하였다. 그러므로 웹 어플리케이션의 개발시 분석 설계단계에서 클래스 수와 메소드 수의 추출이 가능하므로 웹 어플리케이션의 복잡도와 클래스 수나 메소드 수의 상관관계를 파악하여 프로그램 복잡도를 미리 예측하고 인적자원 관리나 비용면의 적용으로 개발주기 관리에 도움을 주고자 한다.

본 연구에서는 프로그램 규모에 대한 복잡도 관계에 대해 연구했으나, 연구 대상을 해당 프로젝트의 ASP 어플리케이션 전체로 확대하여 상관 관계를 파악하고 복잡도의 정량적 예측치를 제시할 필요가 있다. 향후 메소드 호출 수나 클래스당 메소드 수를 추출하여 이 메트릭이 복잡도와 어떤 상관관계에 있는지를 발견하고 웹 어플리케이션의 복잡도 예측에 가장 적합한 메트릭의 선정 및 복잡도의 정량적 예측치를 제시함으로써, 인적자원이나 비용과의 정량적 상관관계 설정이 진행된다면 프로세스와 자원 관리의 획기적 전환이 마련될 것으로 기대한다.

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

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

[5] T. J. McCabe, "A Complexity Measure", IEEE Trans. on Software eng., Vol. 2, No. 4, 1976

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

[7] G. Booch, "Object-Oriented Analysis and Design with Applications, 2nd edition.", Benjamin/ Cummings, Menlo Park, CA., 1991

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

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

[10] 안종근, 유해영, "웹 어플리케이션의 순환복잡도 매트릭에 관한 연구", 한국정보처리학회논문지D, v.9-D, n.3, pp.447-456, 2002.06

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

[12] 김 지현, 박철, "웹 어플리케이션의 모듈위험수준 측정도구의 구현", 한국 OA학회논문지 제7권 제2호, 2002.6.

저 자 소 개

오 성 균

1981년 홍익대학교 이공대학 전자계산학과 졸업(이학사)
1984년 연세대학교 산업대학원 전자계산학과 졸업(공학석사)
1999년 홍익대학교 이공대학 전자계산학과 졸업(이학박사)
1987년 ~ 현재 서일대학 IT계열 소프트웨어 전공 부교수
<관심 분야> 능동 데이터베이스, XML 모델링, 소프트웨어 공학

김 미 진

1985년 성균관대학교 수학교육과 이학사
1989년 성균관대학교 교육학석사,
1997년 미국 Northeastern Univ. 이학석사-컴퓨팅전공
1992년 ~ 1996년 농협대학교 전산과 강사
1997년 ~ 1998년 LinkSoft사 소프트웨어 개발분야 인턴
1992년 ~ 현재 서일대학 IT계열 소프트웨어전공 강사