Journal of The Korea Society of Computer and Information
Vol. 17 No. 6 , June 2012

2012-17-6-4-4

웹 프로그래밍을 위한 복잡도 한계값의 적정성

김 지 현*

Relevance of the Cyclomatic Complexity Threshold for the Web Programming

JeeHyun Kim*

요 약

본 연구는 웹 환경에서 어플리케이션 복잡도의 빈도분포를 근거로 한계값의 적정성을 분석하기 위한 실험을 하였으며 두 가지 가정을 기준으로 작업하였다. 즉, 절차적 프로그래밍에서 McCabe의 상한값 10과 자바 프로그래밍에서의 Lopez의 상한값 5에 대하여 웹 프로그래밍 구문에 이들 설정값의 적용이 가능한가?에 대한 실험으로 10웹 사이트 프로젝트를 수집하였고 4,000여개의 ASP파일 표본이 측정되었다. 웹 어플리케이션에 대한 복잡도 빈도분포를 파악한 결과 통합된 웹 어플리케이션의 90% 이상이 복잡도 50이하의 값을 가짐으로써 한계값 50이 제안되었다. 서버, 클라이언트, HTML이 통합된 웹 어플리케이션의 구조상 HTML의 복잡도가 35~40의 값을 가지게 되는데 이는 HTML이 주로 홈 페이지나 사이트 맵을 구성하는 메뉴 형태로 되어 있어 높은 복잡도의 적합성이 설명되었다. 향후 웹 어플리케이션의 구조상 복잡도와 관련된 숨어 있는 속성은 없는지 관련성을 찾아보는 노력이 필요하다.

▸ Keyword : 측정, 복잡도, 한계값, 웹 어플리케이션

Abstract

In this empirical study at the Web environment based on the frequency distribution of the cyclomatic complexity number of the application, the relevance of the threshold has been analyzed with the next two assumptions. The upper bound established by McCabe in the procedural programming equals 10 and the upper bound established by Lopez in the Java programming equals

• 제1저자 : 김지현
• 투고일 : 2012. 05. 21, 심사일 : 2012. 06. 11, 게재확정일 : 2012. 06. 28.
* 서일대학 컴퓨터소프트웨어과(Dept. of Computer Software, Seoil University)
※ 본 논문은 2011년도 서일대학 학술연구비에 의해 연구되었음.

5. Which numerical value can be adapted to Web application contexts? In order to answer this 10 web site projects have been collected and a sample of more than 4,000 ASP files has been measured. After analyzing the frequency distribution of the cyclomatic complexity of the Web application, experiment result is that more than 90% of Web application have a complexity less than 50 and also 50 is proposed as threshold of Web application. Web application has the complex architecture with Server, Client and HTML, and the HTML side has the high complexity 35~40. The reason of high complexity is that HTML program is usually made of menu type for home page or site map, and the relevance of that has been explained. In the near future we need to find out if there exist some hidden properties of the Web application architecture related to complexity.

▸ Keyword : measurement, complexity, threshold, Web application

I. 서 론

지난 수 년 동안 소프트웨어 개발 환경이 이미 웹 기반으로 빠르게 전환되어 정착되고 있으나 아직도 소프트웨어 측정 및 관련 메트릭에 대한 연구는 객체지향 개발 환경 및 기술 수준에 머물러 있는 실정이다.

산출물의 속성 측정은 프로젝트 개발 비용이나 신뢰소프트웨어 에 관한 의사 결정에 도움을 주며 프로젝트에 관한 조직의 이해를 증진시킨다. 그러나 측정은 어렵고 복잡하여 측정 데이터 집합만으로는 좋은 결정을 하기에 충분하지 않다는 것이다. 그 이유는 소프트웨어 속성, 즉 신뢰성이나 결합도 등의 숫자 값이 어떤 구체적인 참조 없이 이해가 어려우며 해석 단계에서 수용 가능한 값이 정확히 정의되어야 하기 때문이다.

이러한 측정과정 중 만나는 주요 문제는 한계값(threshold)과 관련이 있다. 한계값은 코드의 취약부분을 정의하는 숫자 경계를 의미하며 측정자가 수용 가능한 영역 밖의 속성을 정의하는데 도움을 준다. 한계값의 범위는 어디까지이고 한계값 사용의 의미를 결정하기 위한 파라미터 정의 방법은 무엇인지에 대한 답은 중대하며 또한 매우 어려운 문제이다. 최대, 최소 한계값은 결과의 개발 도중 의사결정을 돕기 위해 측정의 응용 전에 정확히 정의되어야 한다. 그럼에도 불구하고 한계값이 주어진 측정에 제공될 때 그 값들이 사용되는 조건은 잘 정의되지 않는다. 또한 소프트웨어 개발자들은 프로젝트를 측정하지 않는 경향이 있으며 측정하더라도 기밀로 간주하고 때론 신뢰성 때문에 사용하지 않는데 여기에 한계값 적정성 체크의 어려움이 발생하는 것이다[1].

웹 어플리케이션은 서버, 클라이언트, HTML등의 복합구조를 이루고 있어 복잡도가 높을 수 있으며 그에 의한 결함으로 위험도는 높고 사용자 만족도는 낮은 것으로 판명되고 있다[2].

연구의 궁극적인 목표는 전통적 특성과 객체지향 특성이 혼재하고 있는 웹 어플리케이션의 복잡도 한계값을 제안하고 적정성을 분석함으로써 결함가능성이 큰 프로그램의 예측을 돕고 효율적인 유지보수와 사용자 만족도를 높이고자 한다.

본 연구는 웹 어플리케이션 복잡도에 대한 한계값 설정 작업으로 McCabe의 전통적 프로그램의 복잡도 상한값 10과 Lopez의 객체지향 자바 프로그래밍 복잡도 한계값 5에 대하여 첫째, 웹 어플리케이션 구조에 적용 가능한 한계값은 무엇인가? 둘째, 웹 환경에서 클라이언트 요소인 객체지향 자바 스크립트의 한계값은 무엇인가? 에 대한 가정과 그에 대한 통계치를 얻기 위하여 웹 복잡도 빈도 분포를 분석하고자 한다.

실험을 위해 10개의 웹 사이트 프로젝트에서 ASP어플리케이션의 복잡도 빈도 분포에 따른 한계값을 측정한다. 실험 작업으로는 첫째, 웹 어플리케이션의 대표적 표본에서 클라이언트 요소인 자바 스크립트의 빈도 분포 분석, 둘째, 수집한 웹 어플리케이션 전체의 빈도 분포 분석, 셋째, 웹 어플리케이션과 서버와의 복잡도 영역 레벨 차이가 5이상인 복잡도가 높은 어플리케이션의 빈도 분포 분석을 통해 웹 복잡도 한계값의 적정성을 평가하는 것이다.

웹 어플리케이션의 구조에 따른 복잡도 산출은 서버 스크립트 요소, 클라이언트 스크립트 요소, HTML 세가지 요소들의 복잡도 합으로 구성되어 있다. 서버 복잡도와 클라이언트 복잡도는 메서드나 모듈의 제어 흐름에서 추출한 순환 복잡도 값이 계산되었고 HTML 복잡도는 링크의 수 +1이 적용되었다[13][15].

논문의 구성은 2장에서 측정의 문제점 및 기존의 한계값 연구를 파악하고, 3장에서 순환복잡도, 객체지향 복잡도, 웹 어플리케이션 복잡도 및 복잡도 인디케이터를 제시한다. 4장에서 실험 대상 표본을 추출하고 실험과정을 보이며 실험 결과를 통해 복잡도 빈도 분포에 의한 한계값의 제안 및 적정성에 대하여 분석 평가한다. 5장에서 결론과 향후 과제를 언급한다.

II. 측정과 한계값

1. 측정의 문제점

소프트웨어 공학에서 측정은 힘이 드는 작업으로 남아있는데 그 이유는 측정 생명주기 즉, 측정 설계에서 결과 이용까지 전 단계에 걸쳐 어려움이 발생하기 때문이다. 지시자가 정확한지에 관한 측정을 측정하는 방법을 확실하게 하기 위해 명확하지 않은 설계 단계와 연관된 어려움을 강조하고 있다[3]. 또한 이러한 어려움이 해결된다고 해도 측정 결과의 이용은 논문에 나타나지 않고 있다. 측정의 이용은 측정자에게 관련 값을 분석하여 의사결정하게 하는데 이 작업이 쉽지 않은 것이다. 예를 들어 주어진 객체지향 소프트웨어가 100개의 클래스를 가질 때 크기가 '대'인지 '중'인지 어떻게 결정할 것인가? 이 질문은 그러한 판단의 근거는 무엇이고 그의 경험에 따른 추정의 적정성은 무엇인가이다[1].

측정 이용 단계의 중요한 이슈는 과학 분야에서 별로 탐구되지 않고 있는 한계값의 결정이다. 이슈와 관련된 질문 중에는 '수용 가능한 값의 범위를 정하는 한계값을 어떻게 정하는가?', '한계값은 구문-의존적(context-dependent)인가', '만약 구문-의존적일 때 주어진 한계값의 사용이 의미있을 것인지를 결정하기 위해 구문에 영향을 주는 파라미터를 어떻게 정할 것인가?'등이 있다. 이러한 질문은 중요하나 대답이 쉽지 않다. 질문에 대한 답은 내려진 결정에 대한 신뢰성과 측정 과정 전체의 인상을 결정한다. 주어진 한계값에 대한 신뢰 부족이 측정 결과의 사용을 방해할 수 있으며 실제 소프트웨어 프로젝트에서 발견되지 않은 한계값은 측정자에게 더 좋은 측정을 찾게 하고 측정 결과를 숙고하게 할 수 있다.

한계값을 결정하는 가장 일반적인 방법은 경험에 기초해 간단히 선택하는 것이다. 한계값을 결정하는 다른 방법은 주어진 측정의 실험적 빈도 분포에 대한 개별 값의 비교와 중간으로부터 극적으로 다른 경우에 집중하는 것인데 이러한 접근은 대표적 샘플의 생산이 어렵기 때문에 시간과 노력이 많이 드는 작업이다.

자료 집합에서 한계값의 적정성을 체크하는 두 가지 경험적 어려움이 있는데 첫째, 측정은 소프트웨어 프로젝트에서 드문 작업으로 소프트웨어 엔지니어는 그들의 프로젝트를 측정하지 않는다는 것이다. 둘째, 실제 소프트웨어의 대표적 샘플을 구축하는 것은 곤란한 작업이기 때문에 그러한 자료가 있다하더라도 자주 신뢰성에 대해 고려하고 있다[4].

이와 같은 어려움에도 불구하고 한계값을 결정하는 반복적이고 재 생산적인 방법의 제안은 측정의 이용 증진, 검증 향상 및 잠재적 사용 증진을 도모할 것이다.

2. 기존의 한계값 설정

순환 복잡도 한계값을 정의하는 두가지 방법과 객체지향 프로그램에서의 한계값 결정 방식에 다음과 같은 연구가 존재한다.

첫째 방법으로, 복잡도 한계값은 두 가지 속성과 연계되어 결정되는데, 테스팅 노력과 오류밀도 또는 문제의 양과 관련되어 결정하는 방식이다[12]. 그 과정은 먼저, 프로그램 집합의 순환복잡도를 측정하고 다음은, 시험 중에 만난 테스팅 노력이나 문제의 양이 수집되며 마지막으로, 이 측정들 사이에서 상호 연관성을 계산하는 것이다. 여기서 테스팅 노력이나 문제의 양이 많으면 순환복잡도 값이 복잡으로 정의되어 재작업이 요청되는 방식이다[5]. 그러나 이 방식은 너무 무겁고 복잡하며 개발자의 숙련도에 의해 좌우되는 문제점이 있다. 연구의 주요 문제는 실험이 수행된 상황에 있는데 프로그래밍 언어와 패러다임, 개발자나 시험관의 기술수준 또는 개발 모델과 같은 요인들이 고려되지 않음으로 복잡도 한계값의 적정성은 회의적이다.

두 번째 방법으로, 16개의 산출물 샘플내에서 순환복잡도 빈도분포를 분석해서 한계값의 분포를 기초로 하여 결정하는 실험 연구 방식이 있다[6]. 이 방식은 3가지 언어가 샘플로 주어졌으나 한 가지 타입의 프로그램만 고려되었고 결국 프로그램 언어와는 무관하다. 함수의 50%가 10이하의 복잡도를 가지며 90%는 80이하의 복잡도를 가진다는 실험결과를 제시하였고 자료에 기초해서 세 가지 카테고리(<10, >=10, >80)가 정의되었다. 이 연구에서 문제에 관한 레포트 수와 유지보수 노력의 상호연관성에 대한 탐구를 간단히 설명하였으나 실험 조건이 정의되지 않았다.

세 번째 방법으로 10 이상의 복잡도를 갖는 자바 메서드의 비율을 확인함으로써 제안된 순환복잡도 한계값의 적정성을 평가하는 방식이다. 자바 프로젝트의 694 소프트웨어 샘플이 작업에 사용되었다. 순환복잡도와 연관된 개체로 자바 프로그램에서는 "클래스내의 메서드"로 정의하였으며 대부분의 산출물 샘플은 웹사이트 라벨이 부여되어 성숙(mature)된 것을 대상으로 하였고, 오픈소스 특성이 샘플의 약점으로 간주될 수도 있으나 순환복잡도의 비독립적 변수는 폐쇄된 소스와 오픈소스의 독립변수에 영향을 주지 않는다고 하였다. 이러한 작업 환경에서 순환복잡도의 밀도 분포를 계산하여 90% 이상이 5이하의 복잡도를 가진다는 것을 보임으로써 객체지향

자바 프로그램에서는 5의 한계값이 적정함을 제안하였다. 측정의 한계값은 구문 의존적이며 개발자의 숙련도와는 관계없음을 제시하였다[1].

본 연구는 세 번째 방식인 Lopez의 제안 중 객체지향 언어인 자바 프로그램 대신 웹 어플리케이션에서 복잡도 한계값을 추출하고 적정성을 분석하고자 한다. 실험을 위해 10개의 웹 사이트 프로젝트에서 클라이언트 요소인 자바 스크립트 프로그램의 복잡도 빈도 분포를 분석하고, 웹 어플리케이션 전체의 복잡도 빈도 분포 분석 및 웹 어플리케이션의 구조적 특성상 서버와의 복잡도 영역 레벨 차이가 5이상인 어플리케이션의 복잡도 빈도 분포를 분석하여 웹 복잡도 한계값의 고려를 제안한다.

논문의 목표는 McCabe가 제안한 복잡도 한계값 10과 Lopez가 제안한 한계값 5이상의 복잡도를 가지는 웹 어플리케이션의 비율을 확인함으로써 제안된 순환복잡도 한계값의 적정성을 평가하는 것이다.

III. 웹 프로그래밍의 복잡도

1. 순환 복잡도와 웹 복잡도

본 절에서는 절차적 프로그래밍에서의 순환복잡도와 객체지향 프로그래밍에서의 복잡도 및 웹 환경에서의 복잡도에 대해 알아보고자 한다.

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 경로의 수로 결정된다. 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.

$$V(G) = e - n + 2p \quad \text{.....[13]}$$

이렇듯 McCabe의 순환복잡도는 구조적 프로그래밍의 복잡도 측정에 이용되어 왔으며 제어 흐름의 복잡도를 매우 효과적으로 측정하는 메트릭으로 최대 한계값이 10으로 그 한계에 대한 어떤 제한 조건도 없다는 특징이 있다. 또한 10이라는 한계값의 사용에 의문을 갖기에는 엄청난 양(695,741개의 파일)의 측정 결과에 기초한 연구이다[1].

한편 객체지향 시스템에 널리 사용되는 복잡도 측정에는 CK(Chidamber & Kemerer) 메트릭과 추상화 수준에 따른 측정 메트릭이 있다.

Chidamber와 Kemerer가 제안한 것으로 복잡도 측정방법에 두 가지가 존재하는데, 하나는 클래스 내 각 메서드의 복잡도 합이고 다른 하나는 메서드 복잡도를 1로 보았을 때 메서드 수의 합, 즉 클래스 당 메서드의 수가 된다[7][15].

객체지향 시스템은 여러 단계의 추상화 수준을 가지며 추상화 수준에 따른 측정 메트릭으로 여러 가지가 존재하는데 그 중 하나로 메서드 수준에서의 측정이 있다. 지역변수와 같이 참조된 실행문의 집단으로 '함수', '프로시저', '모듈'등이 그것이다. McCabe의 순환복잡도, 팬인-팬아웃같은 것이 복잡도 측정에 사용된다[8][15].

객체지향 프로그래밍에서의 복잡도는 모듈이나 메서드의 그래프로부터 계산되는 것으로 자바 프로그램을 위한 McCabe의 순환복잡도 개체는 '한 클래스 내의 메서드'로 정의하고 있다. Lopez는 메서드의 90%이상이 5이하이며 숙련도는 복잡도 한계값과 관련이 없다고 하였다[1].

웹 환경에서의 복잡도는 하이퍼링크를 제어문과 유사하게 보고 링크의 수 + 1로 정의하였다. 웹 소프트웨어 복잡도(CCWA)는 서버 스크립트 요소(SSS), 클라이언트 스크립트 요소(CSS), HTML 요소들의 복잡도 합으로 한다. 복잡도 계산식은 CCWA = C(SSS) + C(CSS) + C(HTML) 을 사용한다[14]. 웹 어플리케이션의 구조에 따른 복잡도 산출은 서버 스크립트 요소, 클라이언트 스크립트 요소, HTML 세가지 요소들의 복잡도 합으로 구성되어 있으며 서버 복잡도와 클라이언트 복잡도는 메서드나 모듈의 제어 흐름에서 추출한 순환 복잡도 값이 계산되었으며 HTML복잡도는 링크의 수 +1이 적용되었다.

2. 복잡도 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.

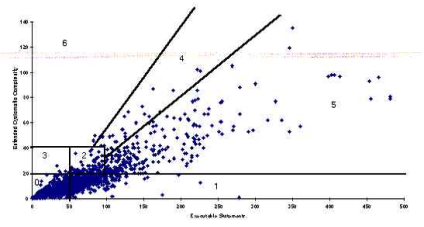

그림 1 복잡도 인디케이터
Fig 1. Complexity Indicator

위의 그래프를 보면 복잡도를 0에서 6까지 7개의 영역으로 나누고 영역0(없음)에서 영역6(최상)까지 구분하여 명명하였다. 즉, [그림 1]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 이 영역에 분포하는 프로그램은 특별 관리가 필요하며 개발자들은 이 영역에 분포한 프로그램의 결함에 대해 더욱 조사해야 한다고 분석하였다[10][13].

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(CCWA)의 차가 5이상인 결함 가능성이 높은 어플리케이션을 추출하여 복잡도 빈도 분포를 파악함으로써 웹 환경에서의 통합된 복잡도 빈도 분포와의 비교를 통해 복잡도 한계값의 설정 및 적합성 분석에 접근하고자 한다.

IV. 실험 및 결과

객관적 자료를 위해 수집한 웹 사이트 프로그램 파일의 실험 및 통계 분석을 하였으며 먼저, 실험대상 표본을 추출하고 둘째 실험 과정을 설명하고 셋째 피벗 테이블과 히스토그램을 통한 통계 분석과 그 결과를 제시하여 웹 환경에서의 복잡도를 비교하고 위험 수준을 넘지 않는 한계값의 적정성에 대해 제안한다.

1. 실험 대상 표본 추출

본 연구는 웹 환경에서 ASP 어플리케이션의 복잡도 빈도 분포를 계산하고 그에 따른 한계값의 적정성을 체크하는 것이 목적으로 10 웹 사이트 프로젝트 14,944개의 파일 중 4,237개의 표본을 추출하여 웹 어플리케이션 구조에 따른 복잡도를 산출하고 복잡도 빈도 분포를 분석한다. 총 10개의 웹 사이트 시스템은 대기업의 DBRMS, 중고차 매매 사이트, 대학 웹 사이트, 상업용 팀 다이어리, 기업의 단위 업무를 처리하는 시스템으로 인사 관리, 수주 발주 관리, 영업 관리, 고객 관리, 품질 관리, 프로젝트 관리등의 시스템이 포함되어 있다.

실험대상 시스템은 ASP로 작성된 웹 어플리케이션으로 서버측은 VBScript, 클라이언트측은 JavaScript로 작성된 시스템이다. <표1>은 실험 대상 시스템의 종류 및 총 파일 수와 복잡도를 추출한 ASP 소스 파일 수를 나타낸 것이다.

표1. 실험대상 시스템 구성 및 파일 수
Table1. Description of source system & no.of files

코드	실험대상 시스템	총파일 수	소스파일수 (*.asp등)
S01	대기업의 전산자원관리 시스템	831	243
S02	인터넷 중고차매매 시스템	263	222
S03	게시판 중심의 대학홈페이지	757	340
S04	상업용 패키지-팀 다이어리	1,658	658
S05	복리후생관리 시스템	1,232	407
S06	수발주관리 시스템	748	315
S07	영업관리 시스템	980	142
S08	고객지원센터 시스템	6,106	1,456
S09	품질관리 시스템	622	269
S10	프로젝트 관리 시스템	1,747	185
	합계	14,944	4,237

실험 대상 시스템은 다시 3종류의 집단으로 나누어 복잡도 빈도 분포를 분석한다. 첫째 클라이언트측의 JavaScript 파일의 복잡도 빈도 분포를 확인함으로써 McCabe의 한계값 10이나 Lopez의 한계값 5의 적합성을 분석한다. 둘째, 서버, 클라이언트, HTML을 통합한 웹 어플리케이션 전체를 모집단으로 하여 복잡도 빈도 분포를 분석한다. 셋째, 웹 어플리케이션과 서버와의 복잡도 영역 레벨 차이가 5이상인 복잡도가 높은 어플리케이션을 실험대상 표본으로 추출하여 복잡도 빈도 분포를 분석한다.

2. 실험 과정

측정을 위한 실험 과정이 [그림2]에 나타나 있으며 상세 절차는 다음과 같다.

1) ASP 소스파일을 파서를 통해 토큰을 분리한다. ASP 소스를 구성하는 VBScript, JavaScript, HTML 토큰을 인식하여 분리한다.

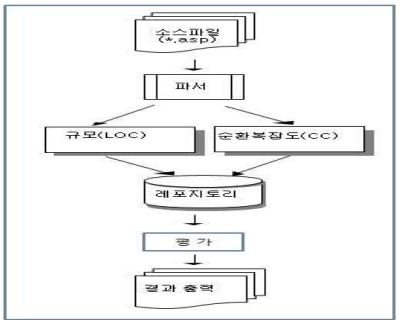

그림2. 측정 실험 과정
Fig2. Measurement experiment process

2) 파싱처리후 LOC(Lines of Code)와 순환복잡도(CCN)를 측정하여 결과를 레파지토리에 저장한다

3) 레포지토리에 저장된 정보를 주어진 질의를 통해 복잡도 빈도 분포를 위해 통계처리한다.

4) 통계 처리한 내용을 분석, 평가하여 결과를 출력한다.

실험 대상 소스 파일을 3종류의 집단으로 나누고 통계처리를 위하여 피봇테이블과 히스토그램을 이용하여 한계값을 측정한다. 먼저 클라이언트측 자바스크립트 복잡도 빈도 분포 측정, 다음 전체 웹 어플리케이션의 복잡도 빈도 분포 측정, 마지막으로 웹 복잡도와 서버측 복잡도 차이가 5이상인 프로그램의 빈도 분포 측정 등이다.

3. 결과 분석

실험의 측정 결과 위의 3종류의 집단에 대한 복잡도 빈도 분포의 결과를 피벗테이블과 히스토그램을 통해 분석한다.

표2. 클라이언트측 복잡도 빈도분포
Table2. frequency distribution of complexity of client

CCN	갯수	비율	누적수	누적비율
1	122	55.3	122	53.3
2	30	13.1	152	66.4
3	21	9.2	173	75.5
4	22	9.6	195	85.2
5	12	5.2	207	90.4
6	4	1.7	211	92.1
7	8	3.5	219	95.6
8	2	0.9	221	96.5
9	4	1.7	225	98.3
11	1	0.4	226	98.7
12	1	0.4	227	99.1
14	1	0.4	228	99.6
18	1	0.4	229	100.0
		0.0	229	100.0
총합	229			

3.1. 클라이언트측 자바스크립트의 복잡도 분포 측정

<표 2>는 클라이언트측 자바스크립트로 작성된 메서드의 복잡도 빈도 분포를 피벗테이블로 나타낸 것으로 1열은 순환복잡도(CCN)를 나타내며 2열은 1열의 순환복잡도(CCN)를 가진 메서드의 개수, 3열은 1열의 CCN을 가진 메서드의 비율, 4열은 1열의 CCN보다 작거나 같은 CCN을 가진 메서드 개수, 5열은 1열의 CCN보다 작거나 같은 CCN을 가진 메서드의 비율을 나타내고 있다.

복잡도 1인 메서드는 전체의 53.3%를 차지하며 5이하의 복잡도를 가지는 메서드는 전체의 90.4%임을 알 수 있다. 또한 복잡도가 1보다 큰 메서드는 46.6%에 해당함을 알 수 있고 복잡도가 11이상인 메서드는 1.3%에 해당함을 알 수 있다. 전체적으로 낮은 복잡도를 보이며 Lopez의 객체지향 프로그램 복잡도 5의 한계값을 가짐을 알 수 있다.

3.2. 전체 웹 어플리케이션의 복잡도 분포 측정

웹 어플리케이션의 복잡도(CCWA)는 총 4071개의 파일에서 3~266의 분포를 보이는데 <표3>은 전체 웹 어플리케이션의 복잡도 빈도 분포를 나타내는 피벗테이블로 지면관계상 복잡도 52를 넘는 90.5%이외의 나머지부분은 생략하였다. 메서드의 31.9%가 복잡도 10이하이며, 61.7%가 20이하의 복잡도를 가지며 메서드의 90.1%가 51이하의 복잡도를 가짐을 알 수 있다.

표3 전체 웹어플리케이션의 복잡도 빈도분포
Table3. frequency dist. of complexity of total web app.

CCWA	요약	비율	누적 수	누적비율
3	267	6.6	267	6.6
4	151	3.7	418	10.3
5	147	3.6	565	13.9
6	122	3.0	687	16.9
7	125	3.1	812	19.9
8	242	5.9	1054	25.9
9	105	2.6	1159	28.5
10	140	3.4	1299	31.9
11	137	3.4	1436	35.3
12	171	4.2	1607	39.5
13	112	2.8	1719	42.2
14	107	2.6	1826	44.9
15	113	2.8	1939	47.6
16	120	2.9	2059	50.6
17	86	2.1	2145	52.7
18	129	3.2	2274	55.9
19	76	1.9	2350	57.7
20	160	3.9	2510	61.7
21	79	1.9	2589	63.6
22	57	1.4	2646	65.0
23	59	1.4	2705	66.4
24	36	0.9	2741	67.3
25	39	1.0	2780	68.3
26	43	1.1	2823	69.3
27	42	1.0	2865	70.4

28	53	1.3	2918	71.7
29	47	1.2	2965	72.8
30	44	1.1	3009	73.9
31	21	0.5	3030	74.4
32	35	0.9	3065	75.3
33	41	1.0	3106	76.3
34	24	0.6	3130	76.9
35	23	0.6	3153	77.5
36	26	0.6	3179	78.1
37	28	0.7	3207	78.8
38	24	0.6	3231	79.4
39	37	0.9	3268	80.3
40	57	1.4	3325	81.7
41	36	0.9	3361	82.6
42	34	0.8	3395	83.4
43	28	0.7	3423	84.1
44	26	0.6	3449	84.7
45	16	0.4	3465	85.1
46	62	1.5	3527	86.6
47	33	0.8	3560	87.4
48	23	0.6	3583	88.0
49	24	0.6	3607	88.6
50	40	1.0	3647	89.6
51	19	0.5	3666	90.1
52	19	0.5	3685	90.5

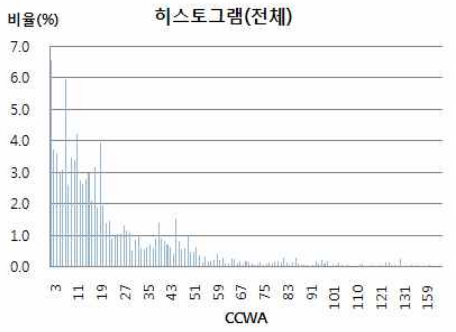

그림3. 전체 웹어플리케이션의 히스토그램
Fig3. histogram of total web application

[그림3]은 전체 웹 어플리케이션의 복잡도 빈도 분포를 나타내는 히스토그램으로 X좌표는 복잡도이며, Y좌표는 같은 복잡도를 가진 메서드의 개수를 전체메서드에 대한 비율로 나타낸 것이다. 복잡도 3이 6.6%로 가장 많은 비율을 차지하여 왼쪽 편향의 스타일을 보인다.

3.3. 웹과 서버측 복잡도 영역 차이가 5이상인 어플리케이션의 빈도 분포 측정

웹어플리케이션과 서버와의 복잡도 영역 차이가 5이상(영역 0에서 5 또는 영역 1에서 6으로 증가)인 어플리케이션의 복잡도를 피벗 테이블로 나타낸 <표4>를 보면 21~173의 높은 빈도분포를 보이고 있다. 메서드의 21.4%가 복잡도 30이하이며 41.9%가 40이하의 복잡도를 가짐을 알 수 있다. 메서드의 63.8%가 50이하의 복잡도, 90.8%가 70이하의 복잡도를 가진다.

실험 측정 결과를 보면 첫째, 클라이언트측 자바 스크립트 프로그램에서는 90.4%가 5이하의 복잡도로 복잡도 5를 한계값으로 설정 가능하다. 이는 Lopez Number와 일치하는 한계값으로 자바 스크립트의 객체지향 특성으로 설명이 가능하다.

둘째, 웹 어플리케이션 전체를 표본으로 했을 때 90.1%가 51이하의 복잡도를 가지며 61.7% 가 20 이하의 복잡도를 보인다. 나머지 40%정도는 21이상의 복잡도를 가짐을 알 수 있다. 즉, 웹 어플리케이션 전체에서 90.1%가 51이하이므로 복잡도 51을 한계값으로 설정할 수 있다. 그러나 여기서 총 4071개의 파일 중 복잡도 140 이상의 돌출된 파일 17개 정도를 제거한다면 90.0%가 50의 한계값을 갖게 되므로 한계값을 50으로 제안한다.

표4. 복잡도영역차이가 5이상인 빈도 분포
Table4. frequency dist.'n of comp. of differ.5

CCWA	요약	비율	누적갯수	누적비율
21	9	3.9	9	3.9
22	10	4.4	19	8.3
23	2	0.9	21	9.2
24	1	0.4	22	9.6
25	2	0.9	24	10.5
26	3	1.3	27	11.8
27	1	0.4	28	12.2
28	13	5.7	41	17.9
29	4	1.7	45	19.7
30	4	1.7	49	21.4
31	1	0.4	50	21.8
32	2	0.9	52	22.7
33	6	2.6	58	25.3
34	4	1.7	62	27.1
36	2	0.9	64	27.9
37	3	1.3	67	29.3
38	2	0.9	69	30.1
39	9	3.9	78	34.1
40	18	7.9	96	41.9
41	1	0.4	97	42.4
42	2	0.9	99	43.2
43	2	0.9	101	44.1
45	2	0.9	103	45.0
46	1	0.4	104	45.4
47	2	0.9	106	46.3
48	9	3.9	115	50.2
49	6	2.6	121	52.8
50	25	10.9	146	63.8
51	10	4.4	156	68.1
52	9	3.9	165	72.1
53	5	2.2	170	74.2
54	3	1.3	173	75.5
55	2	0.9	175	76.4
56	5	2.2	180	78.6
57	3	1.3	183	79.9
58	2	0.9	185	80.8
59	4	1.7	189	82.5
60	1	0.4	190	83.0
62	2	0.9	192	83.8
63	1	0.4	193	84.3
65	7	3.1	200	87.3
66	1	0.4	201	87.8
67	4	1.7	205	89.5
68	1	0.4	206	90.0
69	1	0.4	207	90.4
70	1	0.4	208	90.8
73	1	0.4	209	91.3
74	2	0.9	211	92.1
75	2	0.9	213	93.0
77	2	0.9	215	93.9
79	2	0.9	217	94.8
82	1	0.4	218	95.2
83	1	0.4	219	95.6
95	1	0.4	220	96.1
96	1	0.4	221	96.5
106	1	0.4	222	96.9
109	1	0.4	223	97.4
121	1	0.4	224	97.8
124	1	0.4	225	98.3
131	1	0.4	226	98.7
140	1	0.4	227	99.1
147	1	0.4	228	99.6
173	1	0.4	229	100.0
(비어 있음)		0.0	229	100.0
총합계	229	100.0		

웹 어플리케이션 복잡도는 서버, 클라이언트, HTML 3요소의 복잡도 합으로 다음과 같은 계산이 가능하다. 즉, 첫 번째 실험의 표본에서 추출한 클라이언트 한계값 5와 서버측의 한계값으로 McCabe 또는 Lopez 한계값 10 또는 5를 합하면 15 또는 10이 된다. 웹어플리케이션의 한계값을 50으로 설정하였으므로 HTML 복잡도는 50에서 15~10을 차감한 35~40의 높은 복잡도를 갖게 된다. HTML 복잡도는 링크의 수 +1을 복잡도로 하고 있으며 주로 홈페이지나 사이트 맵을 구성하는 메뉴 형태로 되어 있어 35~40의 높은 복잡도에 대한 설명이 가능하다.

셋째, 웹 어플리케이션과 서버와의 영역 레벨 차가 5이상인 프로그램은 전체의 약 3%에 해당하는데 복잡도가 21~173의 분포를 가진다. 이중 63.8%가 50이하의 복잡도를, 90.8%는 70이하의 복잡도를 가진다.

여기서 누적 비율이 90.8%를 차지하는 복잡도 70을 한계값으로 설정할 수 있다. 이때 전체 웹 어플리케이션의 복잡도 한계값 50과는 20의 차이가 생기는데 이것은 서버와의 영역 레벨의 차이가 5이상(복잡도 20이상)인 복잡도가 높은 어플리케이션의 분포도이기 때문에 복잡도의 최소 한계값 20과 최대 한계값 70은 웹 어플리케이션의 한계값 50을 뒷받침하는 것으로 볼 수 있다. 그러나 이 표본은 복잡도가 현저히 높은 프로그램을 위주로 한 것이기 때문에 일반적인 웹 어플리케이션 보다 고려해야 할 사항이 더 많이 요구된다.

다음 <표5>는 Lopez의 연구와 본 연구를 비교한 것이다. Lopez는 객체지향 패러다임에서 자바 프로그램의 오픈소스를 사용하였고 본 연구는 웹 패러다임에서 세 가지 환경에서 실제 사용되고 있는 소스를 사용하였다. 그 결과 클라이언트 측의 자바 스크립트 프로그램에서는 Lopez와 같은 5의 한계값을 보였으며 웹 통합 구조와 서버와의 영역차이가 5이상인 복잡도가 높은 어플리케이션에서는 50의 한계값을 가짐을 알 수 있었다.

표5. Lopez 연구와 본 연구의 비교
Table5. Comparison of the study of Lopez and this

	Lopez	본 연구		
개발 패러다임	객체 지향	웹		
프로그래밍 언어	자바	자바 스크립트(클라이언트)	웹구조 (ASP+자바 스크립트 + HTML)	서버와의 복잡도 영역 차이 5이상인 프로그램
표본 소스 타입	오픈 소스	산업에서 실 사용되는 소스		
측정	순환 복잡도	순환 복잡도		
통계적 기법	피벗 테이블 사용	피벗 테이블 사용		
복잡도 빈도 분포	복잡도 1 : 66.73%	복잡도 1 : 53.3%	복잡도 20 : 61.7%	복잡도 50 : 53.3%
	복잡도 5 : 93.85%	복잡도 5 : 90.4%	복잡도 51 : 90.1%	복잡도 70 : 90.8%
조정	없음	없음	4071개 중 17개의 돌출값 제거후 복잡도 50 : 90.0%	70-20=50 (영역5=복잡도 20)
한계값의 제안	5	5	50	50

향후 웹 어플리케이션의 복잡도와 관련된 숨어 있는 특성은 없는지 관련성을 찾아보는 노력이 필요할 것이다. 또한 서버측 복잡도와 HTML측 복잡도의 두 요소 각각에 대한 복잡도 상한값 분포도가 이에 대한 설명이 가능하도록 할 수 있을 것이다.

V. 결론

본 연구는 서버, 클라이언트, HTML이 복합된 구조의 웹 환경에서 통합 어플리케이션의 복잡도 빈도 분포를 파악하고 한계값의 적정성을 제안하는 실험을 3종류의 집단에 대하여 측정하였다. 실험의 측정 결과를 비교해 보면 우선 클라이언트 요소인 자바 스크립트 프로그램에서는 메서드의 90.4%가 5이하의 복잡도로 Lopez의 객체지향 프로그램의 한계값인 5가 적정함을 알 수 있었으며, 둘째 통합된 웹 어플리케이션 전체를 대상으로 했을 때는 메서드의 90%가 50이하의 복잡도 분포를 보였다. 또한 웹 어플리케이션과 서버와의 복잡도 영역 레벨차가 5(복잡도 20의 차이)이상인 어플리케이션은 메서드의 90.8%가 70이하의 복잡도를 나타내었다. 복잡도가 매우 높은 위험도가 큰 어플리케이션에서도 전체 웹 어플리케이션의 한계값 50과 20의 차이를 보임으로써 제안된 한계값 50의 적합성을 설명하였다.

즉 세 가지 표본 집단의 실험을 통해 웹 어플리케이션의 복잡도 한계값은 50의 적합성이 제안되었고 HTML측의 높은 복잡도는 홈페이지나 사이트 맵을 구성하는 메뉴 형태에 의해 설명되었다.

객체지향 패러다임과 마찬가지로 웹 패러다임도 복잡도 한계값에 영향을 미치는 요인이라는 것과 복잡도 측정의 한계값은 구문 의존적이라는 것이다.

향후 웹 환경에서 순환 복잡도에 의해 측정되지 않는 감추어진 웹의 속성이 있는지에 대한 분석이 요구되며 복잡도와 웹 어플리케이션 속성사이의 관련성을 분석하는 노력이 필요하다.

참고문헌

[1] Miguel Lopez, Naji Habra, "Relevance of the Cyclomatic Complexity Threshold for the Java Programming Language", In Proceedings of the Software Measurement European Forum(SMEF 2005), 2005.

[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] Habra, N., Abran A., Lopez, M. & Paulus, V. "Towards a framework for Measurement Lifecycle", University of Namur, Technical Report, TR37/04, 2004

[4] Britoe e Abreu, P., Poels, G., Sahraoui H.A., and Zuse, H., "Quantitative Approaches in Object-Oriented Software Engineering", Kogan Page Science, 2004

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

[6] Stark G., Durst, R.C., "Using Metrics in Management Decision Making", Computer(IEEE), 1994

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

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

[9] Abran, A., Lopez, M., and Habra, N., "An Analysis of the McCabe Cyclomatic Complexity Number", in IWSM Proceedings, Berlin, 2004

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

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

[12] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object Oriented Design Metrics as Quality Indicators," IEEE Trans. Software Eng., vol. 22, no. 10, pp. 751-761, Oct. 1996

[13] JeeHyun Kim, Chel Park, "Implementation of the Module Risk Levels Measurement Tools for Web Application", Journal of the Korea Society of Computer and Information vol. 7, no. 2, Jun. 2002

[14] Chel Park, HaeYoung Yoo, "Analysis of Cyclomatic Complexity for Web Application", Journal of the Korea Information Processing Society, v.11-D, n.4, pp.865-872, Aug. 2004

[15] JeeHyun Kim, HaeYoung Yoo, "A Study of Estimation for Web Software Size", Journal of the Korea Information Processing Society, vol. 12-D. no. 3, May 2005

저 자 소 개

김 지 현

1978: 이화여자대학교 수학과 이학사

1994: 단국대학교 전자정보전공 경영학석사

2004: 단국대학교 전산통계학과 이학박사

현 재: 서일대학 컴퓨터소프트웨어과 부교수

관심분야: 웹공학,데이터베이스, 품질 관리

Email : jhkim@seoil.ac.kr