웹 어플리케이션의 순환복잡도 분석

박 철[†] · 유 해 영[††]

요 약

웹 어플리케이션은 기존의 구조적 언어, 객체지향 언어 또는 4세대 언어로 개발된 기존의 어플리케이션과는 달리 웹 서버에서 실행되는 서버측 스크립트 요소와 웹 브라우저에서 실행되는 클라이언트 스크립트 요소, 그리고 문서의 내용을 표현하는 HTML요소들이 결합된 구조를 가지고 있다. 그렇기 때문에 웹 어플리케이션 개발자들은 동시에 3가지 이상의 개발 언어를 사용하여야 한다. 웹 어플리케이션의 순환복잡도(CCWA : Cyclomatic Complexity for Web Application) 매트릭은 웹 어플리케이션을 구성하고 있는 각 구성 요소들의 복잡도를 복합적으로 측정하도록 고안된 매트릭이다. 본 연구에서는 웹 어플리케이션의 순환복잡도 매트릭을 복잡도 수준 지시자와 함께 사용하여 웹 어플리케이션에 적용한다. 실무에서 개발된 10개의 중 대형 규모의 웹 어플리케이션에 이를 적용하여 기존의 어플리케이션과는 구별되는 복잡한 웹 어플리케이션의 유형을 MENU, FORM, CTRL, GEN의 4가지로 분석하였다. 이러한 분류는 웹 어플리케이션에 대한 공학적 접근에 다양하게 활용될 수 있다.

Analysis of Cyclomatic Complexity for Web Application

Chel Park[†] · Hae-Young Yoo[††]

ABSTRACT

Web applications have different structural characteristics from conventional applications with the structural language or object-oriented language or 4GL. A web application typically consists of server-side script elements which run on web servers, client-side script elements which run on the client web-browser, HTML elements that contains context. Therefore web applications developer concurrently uses 3 or more development language. Cyclomatic Complexity for Web Application(CCWA) metrics reflected composite complexity of each element. In this paper, we applied cyclomatic complexity for web application metrics with Complexity level indicator to web application. We applied it to 10 web applications that were developed in practical business. High complexity web applications classify into four type(MENU, FORM, CTRL, GEN). This paper has contributed to practical use of engineering approach for web application.

키워드 : 웹 어플리케이션(Web Application), 순환 복잡도(Cyclomatic Complexity), 복잡도 수준 지시자(Complexity Level Indicator)

1. 서 론

1990년대 후반에 대중화 되기 시작한 인터넷 기술들이 현재는 사회 변화의 중심축을 이루고 있다. 이러한 인터넷 환경의 도래로 과거에 C/S 환경에서 개발되어 사용되던 기업의 어플리케이션들이 이제는 웹 브라우저 환경에서 실행되도록 변화하고 있다.

초기의 웹은 웹 사이트로 불렸다. 웹 사이트는 텍스트를 기반으로 이미지와 결합된 단방향의 하이퍼 텍스트 문서로 구성되었었다. 그러나 인터넷 기술의 발달과 확산으로 현재의 웹 사이트는 단순한 문서의 수준을 넘어 기업의 업무처리, 학교의 학사관리 업무 등을 담당하고 있다. 이러한 웹 사이트들은 다양한 기능과 복잡한 정보를 실시간으로 제공하기 위하여 복잡하고 대형화된 어플리케이션의 형태를 가지고 있다. 이렇게 양방향으로 서비스를 제공하는 복잡한 웹 사이트를 웹 어플리케이션이라고 부른다.

웹 어플리케이션의 발전과 함께 웹 어플리케이션에 대한 여러 가지 연구들이 진행되어 왔고, 1997년 WebISM(Web-based Information Systems and Methodologies)의 연구 그룹에서 웹 공학(WebE : WebEngineering)이라는 이름으로 웹 어플리케이션에 대한 공학적인 접근을 시작하였다[15, 8, 7]. 웹 공학은 "웹 기반의 시스템과 어플리케이션의 개발 예산 초과, 개발 일정의 지연, 저조한 생산성, 미흡한 품질 등과 같은 문제점을 해결하기 위하여 표준 방법론을 모색하고, 생산성과 품질을 향상시키기 위해 소프트웨어 공학을 웹 어플리케이션 개발에 적용시키는 것"으로 정의할 수 있다[15].

그러나 실제로 많은 웹 어플리케이션들이 짧은 역사로

† 종신회원 : 단국대학교 정보컴퓨터학부 교수
†† 정 회 원 : 단국대학교 정보컴퓨터학부 교수
논문접수 : 2004년 2월 11일, 심사완료 : 2004년 3월 25일

인하여 숙련되지 않은 개발자들에 의해 만들어지고 있으며, 아직까지도 웹 어플리케이션 개발에 적용할 수 있는 공학적인 표준이 부족한 상태이다[8, 12]. 복잡도(Complexity)는 소프트웨어의 유지보수성과 품질 평가를 위한 기본적인 매트릭 중의 하나이다[9, 16]. 웹 공학 분야에서 웹 어플리케이션의 복잡도는 웹 문서가 가지는 구조적인 복잡도로 보는 연구가 진행되어 왔다[6, 10, 11, 13]. 그러나 웹 어플리케이션을 웹 문서로 보는 접근 방법은 현재의 복잡하고 다양한 웹 어플리케이션의 복잡도를 표현하기에는 부족하다[3, 1, 4].

웹 어플리케이션의 순환복잡도(CCWA : Cyclomatic Complexity for Web Application) 매트릭은 웹 어플리케이션을 웹 문서로 보지 않고, 여러 가지 형태의 코드가 결합되어 있으며 각각이 서로 유기적인 연관관계를 가지고 있는 프로그램으로 간주하고 복잡도를 측정하는 안종근이 제안한 매트릭이다[3].

본 논문에서는 CCWA 매트릭이 실무에서 유용하게 활용될 수 있는지를 확인하기 위하여, CCWA 매트릭을 복잡도 수준 지시자와 함께 사용하여 웹 어플리케이션의 복잡도를 분석한다. 2장에서는 웹 어플리케이션의 복잡도를 설명하고, 3장에서는 실무에서 개발된 10개의 중 대형 규모의 웹 어플리케이션에 대하여 복잡도 위험 수준을 평가하고, 4장에서는 높은 복잡도 수준을 보이는 웹 어플리케이션을 분석하여 기존의 어플리케이션과 구별되는 복잡한 웹 어플리케이션의 유형을 분석한다.

2. 웹 어플리케이션과 복잡도 매트릭

2.1 하이퍼텍스트 관점의 웹 어플리케이션 복잡도

웹 문서에 대한 복잡도에 대한 연구는 1990년대 후반부터 진행되어 왔다. 1995년 "Measuring the Readability and Maintainability of Hyperdocuments"에서는 순환복잡도 매트릭을 하이퍼텍스트 문서에 적합하도록 재해석하였다[6]. 이 연구에서는 사용자가 하이퍼링크를 클릭하면 새로운 문서로 이어지는 흐름이 생긴다는 점을 이용하여, 하이퍼링크를 프로그램 언어에서 제어의 흐름을 변경하는 제어문과 유사한 것으로 보고 제어 그래프를 작성하고, 이 제어 그래프에 순환복잡도를 적용하였다. 이 연구에서 웹 문서의 복잡도 C는 링크의 수 +1로 정의되었다.

2003년, Emilia Mendes, Nile Mosley은 웹 설계와 저작을 위한 노력 예측 모델을 개발하기 위하여 GQM 모델을 사용하였다[10]. 이 연구에서 GOAL은 "개발자 관점에서 웹 설계와 저작의 노력을 측정"하는 것이고 QUESTION은 "웹 어플리케이션의 크기를 특징 짓는 속성은 무엇인가?"였다. 이 QUESTION에 대한 의견을 듣기 위해 오클랜드 대학의 컴퓨터과학 전공 학생들로부터 GQM 설문지를 받아 <표 1>의 매트릭을 수집하였다.

<표 1> GQM 매트릭

METRIC	설 명
Page Count	웹 어플리케이션에서 사용된 html이나 shtml등의 파일의 수
Media Count	어플리케이션에 사용된 미디어 파일의 수
Program Count	어플리케이션에서 사용된 cgi script, Javascript files, Java applets의 수
Total Page Allocation	어플리케이션에 사용된 html이나 shtml등의 파일의 총 용량(Mbytes)
Total Media Allocation	어플리케이션에 사용된 미디어 파일의 총 용량(Mbytes)
Total Code Length	어플리케이션에 사용된 총 코드의 줄 수(LOC)
Reused Media Count	재사용되거나 수정되어 사용된 파일의 수
Reused Program Count	재사용되거나 수정되어 사용된 프로그램의 수
Total Reused Media Allocation	어플리케이션에서 재사용된 미디어 파일의 총 용량(Mbytes)
Total Reused Code Length	어플리케이션에서 재사용된 모든 프로그램원 줄 수(LOC)
Connectivity	내부 링크의 총 수
Connectivity Density	Connectivity/Page Count
Total Page Complexity	$\frac{\sum_{PageCount} Numb.diff.type\hat{}media\hat{}in\hat{}a\hat{}page}{Page\ Count}$
Cyclomatic Complexity	(Connectitity Page Count) + 2

수집된 매트릭을 이용하여 중복적인 요소를 제거하고 TPA (Total Page Allocation)과 TRCL(Total Reused Code Length)를 사용하여 선형 회귀를 이용한 모델과 단계적 다중 회귀를 이용한 2가지 노력 예측 모델을 제안하였다.

[선형 회귀 모델]

Effort = 40.558 + 0.097TPA + 0.103TRCL

[단계적 다중 회귀 모델]

Effort = 28.681 + 0.034TRCL + 0.051TPA

TPA : Total Page Allocation, TRCL : Total Reused Code Length

이 연구에서 GQM 설문지를 통해 조사된 <표 1>의 매트릭을 보면 페이지의 수, 링크의 수를 기본적인 측정의 대상으로 하고 있으며, 최종적으로 제안된 모델도 비즈니스 로직이나 알고리즘과 무관한 TPA와 TRCL의 공식으로 제안되어 있다.

웹 어플리케이션의 복잡도에 대한 하이퍼텍스트 관점의 접근법을 웹 어플리케이션에 그대로 적용할 경우, 논리적으로 단순한 업무와 복잡한 업무가 동일한 노력이 필요한 것으로 예측될 수 있는 문제점을 가지고 있다. 따라서 이러한

문제점을 개선하기 위해서는 개발자 관점의 접근법이 필요하다.

2.2 개발자 관점의 웹 어플리케이션 복잡도

최근에는 웹 어플리케이션을 개발자 관점에서 보는 여러 연구들이 있었다[3, 1, 4]. 그 중에서 문헌 [4]의 "웹 어플리케이션의 복잡도 척도" 연구에서는 웹 어플리케이션을 정적인 페이지, 클라이언트 스크립트 페이지, 서버 스크립트 페이지, 클라이언트 컴포넌트 페이지, 서버 컴포넌트 페이지의 5가지의 구성요소로 이루어진 것으로 간주하고 웹 어플리케이션의 복잡도를 각 구성요소의 복잡도의 결합된 식으로 표현하였다[4].

이 연구에서 제안된 복잡도 척도 CM은 (그림 1)과 같다.

CM1 = k1×SP+k2×CSP+k3×SSP+k4×CCP+k5×SCP
SP : 정적인 페이지의 수
CSP : 클라이언트 스크립트 페이지의 수
SSP : 서버 스크립트 페이지의 수
CCP : 클라이언트 컴포넌트 페이지의 수
SCP : 서버 컴포넌트 페이지의 수

(그림 1) 복잡도 척도 CM

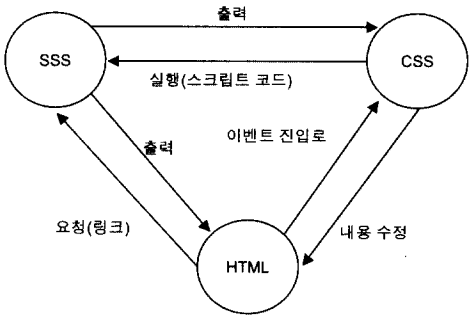
이 연구는 복잡도 척도가 여러 구성 요소들의 결합으로 표현할 것을 제안하였으나 개발될 척도의 모양만을 제시하였고 k1~k5의 가중치를 제안하지는 못했다. 그러나 웹 어플리케이션을 여러 구성요소의 결합체로 보고 그 요소들을 복합적으로 측정하려는 시도를 하였다는데 의미가 있다.

2.3 웹 어플리케이션의 순환복잡도(CCWA)

웹 어플리케이션은 기존의 어플리케이션과 달리 여러 프로그램 요소들이 결합되어 있다. 웹 어플리케이션을 구성하는 구성요소와 각 구성요소들 간의 관계는 (그림 2)에 나타나 있다. 웹 서버에 존재하는 웹 어플리케이션 소스에는 서버에서 스크립트 엔진에 의해서 실행되는 서버 스크립트 요소(SSS : Server side Script)와 클라이언트에 그대로 전송되어지는 클라이언트 스크립트 요소(CSS : Client side Script)와 HTML 태그로 구성된 HTML 요소들이 동시에 포함되어 있다. 서버 측에 존재하던 소스코드는 클라이언트의 요청에 의해 클라이언트로 전송되기 전에 서버에서 SSS 요소가 실행되고 실행된 결과만이 웹 브라우저로 전송된다. 클라이언트로 전송된 결과에는 CSS와 HTML 요소만이 남아있게 된다. 따라서 웹 어플리케이션의 복잡도를 측정하려면 SSS 요소, CSS 요소, HTML 요소들의 복잡도를 각각 고려하여야 하고 또한 동시에 고려하여야 한다.

CCWA 매트릭은 안종근이 제안한 웹 어플리케이션의 순환복잡도 매트릭으로 웹 어플리케이션을 구성하는 요소는 SSS, CSS, HTML 요소들의 복잡도의 합으로 다음과 같이 정의 된다[3]. 식에서 C(SSS)는 서버 스크립트 요소의 순환복잡도로 전통적인 순환복잡도와 같은 방법으로 측정된다. C(CSS)는 클라이언트 스크립트 요소의 순환복잡도로 C(SSS)와 마찬가지로 전통적인 순환복잡도와 같은 방법으로 측정된다. C(HTML)은 웹 문서의 복잡도로 링크의수 +1로 측정한다.

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



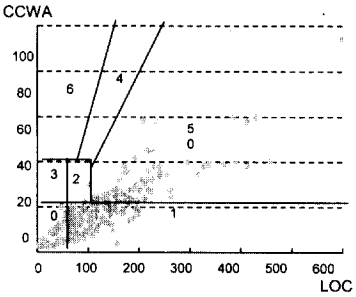
(그림 2) 웹 어플리케이션 구성요소와 상호관계도

2.4 복잡도 수준 지시자

복잡도 수준 지시자(Complexity Level Indicator)는 1997년 NASA의 SATC(Software Assurance Technology Center)에서 개발한 것으로 모듈에 문제가 발생할 가능성을 0~6의 7단계로 표현하는 위험 정보에 대한 지표이다[14]. SATC에서는 NASA에서 C, C++, FORTRAN, ADA등의 다양한 언어로 개발된 프로젝트로부터 순환복잡도와 LOC를 측정하고 그 결과를 사용하여 복잡도 수준 지시자를 개발하였다.

복잡도 수준 지시자는 (그림 3)에서처럼 LOC와 순환복잡도를 이용하여 수준 0에서 수준 6까지 7단계로 복잡도 수준을 결정할 수 있도록 도와준다. 본 연구에서는 SATC에서 개발한 복잡도 수준 지시자를 채택하여 웹 어플리케이션의 위험 수준을 평가하는데 사용한다.

수준	설 명	빈 도
0	MININUM	46
1	LOW	131
2	LOW-MODERATE	0
3	MODERATE	0
4	MODERATE-HIGH	0
5	HIGH	66
6	VERY-HIGH	0



(그림 3) 프로젝트 P01의 CCWA 복잡도 수준 분포

3. 웹 어플리케이션의 순환복잡도 분석

3.1 실 험

본 연구에서는 실무에서 개발된 10개의 중 대형 웹 어플리케이션 프로젝트에 CCWA 매트릭과 복잡도 수준 지시자를 적용하여 복잡도를 분석한다. CCWA 매트릭을 적용한 경우와 기존의 순환복잡도 매트릭을 적용한 경우를 비교하기 위하여 CCWA 매트릭과 서버 스크립트 요소만의 순환복잡도인 C(SSS)를 비교한다. 개발자 관점에서 기존의 순환복잡도 매트릭을 웹 어플리케이션에 적용한다면 서버 스크립트 요소가 선택될 수 있기 때문에 C(SSS)를 비교 대상으로 하였다.

각 파일의 복잡도 수준을 측정하기 위하여 전체 웹 어플리케이션의 순환복잡도인 CCWA 매트릭을 적용한 경우를 "CCWA 복잡도 수준"으로 부르고, 서버 스크립트 요소만의 복잡도인 C(SSS) 매트릭을적용한 경우를 "C(SSS) 복잡도 수준"으로 부르기로 한다.

분석은 다음의 단계로 수행되었다.

분석 1. C(SSS) 복잡도 수준과 CCWA 복잡도 수준을 측정하고 복잡도 수준의 분포를 관찰한다.
분석 2. 각 파일의 복잡도 수준 변화를 관찰한다.
분석 3. 복잡도 수준의 변화가 큰 파일들을 조사하여 그 원인을 분석한다.

3.2 실험 대상 웹 어플리케이션

본 연구에서 실험 대상으로 한 웹 어플리케이션은 서버측 스크립트로 VBScript를 사용하고 클라이언트측 스크립트로 JavaScript를 사용하는 것들을 대상으로 하였다. 실무에서는 이 이외에 몇가지 조합이 웹 어플리케이션 개발에 사용되고 있으나 이 중 대표적인 조합 중 하나를 선택하였다.

〈표 2〉 웹 어플리케이션 프로젝트

프로젝트 코드	총 파일수	소스파일수 (*.asp, *.inc등)	설 명
P01	831	243	대기업의 전산 자원 관리 시스템
P02	263	222	인터넷 경매 시스템
P03	757	340	게시판 중심의 대학 홈페이지
P04	1,658	658	기업 생산성 향상을 위한 상업용 패키지 (판매용)
P05	1,232	407	기업의 업무처리 시스템
P06	748	315	기업의 업무처리 시스템
P07	980	142	기업의 업무처리 시스템
P08	6,106	1,456	기업의 고객센터 시스템, 4개의 서브 시스템으로 구성
P09	622	269	품질관리 시스템
P10	1,747	185	프로젝트 관리 시스템

실험은 총 10개의 웹 어플리케이션을 대상으로 하였다. 이중에서는 게시판 기능을 중심으로 하는 대학의 홈페이지에서부터 기업의 단위 업무를 처리하는 시스템, 개발회사에서 품질관리와 프로젝트 관리를 위해 사용하는 시스템 등이 포함되어 있다. 실험에 사용된 프로젝트를 구성하는 총 파일수와 소스 파일수 등을 <표 2>에 설명하였다.

3.3 C(SSS) 복잡도 수준과 CCWA 복잡도 수준 비교

C(SSS)를 기준으로 각 파일의 복잡도 수준을 측정한 결과가 <표 3>이고, CCWA를 기준으로 각 파일의 복잡도 수준을 측정한 결과가 <표 4>이다.

〈표 3〉 C(SSS)복잡도 수준 분포(비율)

프로젝트 / 복잡도수준	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10
L0	29%	78%	55%	54%	34%	25%	30%	36%	19%	25%
L1	58%	19%	43%	39%	61%	40%	51%	57%	32%	52%
L2	0%	2%	0%	0%	0%	0%	0%	0%	0%	0%
L3	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
L4	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
L5	12%	0%	2%	7%	4%	34%	19%	7%	49%	22%
L6	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

〈표 4〉 CCWA 복잡도 수준 분포(비율)

프로젝트 / 복잡도수준	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10
L0	19%	41%	41%	15%	10%	5%	6%	21%	5%	5%
L1	54%	44%	39%	29%	64%	37%	34%	50%	25%	33%
L2	0%	1%	1%	5%	1%	0%	0%	1%	0%	0%
L3	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
L4	0%	0%	0%	1%	0%	0%	0%	0%	0%	0%
L5	27%	15%	19%	50%	25%	58%	60%	28%	70%	62%
L6	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

<표 5>는 <표 3>과 <표 4>에서 복잡도 수준이 4 이상으로 복잡도가 높은 파일의 수를 비교한 것으로CCWA복잡도 수준을 적용함으로써 복잡도가 높은 것으로 판명되는 파일이 얼마만큼 증가하는지를 보여준다. 복잡도 수준이 4 이상인 파일의 수는 프로젝트에 따라 C(CSS) 복잡도 수준을 측정했을 때는 0%~49%의 분포를 보이고 있고, CCWA 복잡도 수준을 측정했을 때는 15%~70%를 차지한다. 각 프로젝트 별로 복잡도 수준이 4이상인 파일의 비율이 큰 차이를 보이는 것은 그 프로젝트의 특성에 의한 것으로 본 연구의 관심 대상은 아니다.

그러나 증가율을 살펴보면 CCWA 복잡도를 측정한 경우, 복잡도가 4이상인 파일의 수가 각 프로젝트에서 최소 15% 이상으로 증가하였다. CCWA 매트릭이 C(SSS)를 포함하고 C(CSS), C(HTML) 복잡도를 합한 것이기 때문에

어느 정도 복잡도의 증가는 불가피하다. 그러나 그 증가율이15%~41%에 이르는 것은 C(SSS)만으로는 설명되지 않는 복잡도 요인이 존재할 가능성을 보여준다.

〈표 5〉 복잡도 수준이 높은 파일 비교

프로젝트	C(CSS) 복잡도 수준		CCWA 복잡도 수준		증가율
	파일수	비 율	파일수	비 율	
P01	30	12%	66	27%	15%
P02	0	0%	33	15%	15%
P03	8	2%	64	19%	17%
P04	43	7%	332	51%	44%
P05	18	4%	104	25%	21%
P06	108	34%	184	58%	24%
P07	27	19%	85	60%	41%
P08	102	7%	410	28%	21%
P09	131	49%	189	70%	21%
P10	41	22%	114	62%	40%

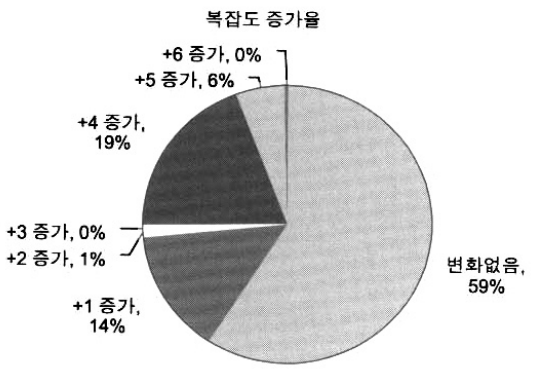
3.4 복잡도 수준의 증가 크기 분석

앞의 실험으로 웹 어플리케이션에는 C(SSS) 복잡도 수준으로는 설명할 수 없는 복잡도 요인이 존재함을 예상할 수 있었다. 이번 실험에서는 이를 보다 구체적으로 확인하기 위하여 각 파일별로 복잡도가 어느 정도 증가하는지를 비교해 보았다. <표 6>은 각 파일에 대하여 CCWA 복잡도 수준과 C(SSS) 복잡도 수준을 비교하여 그 증가하는 크기를 표시한 것이다. 예를 들어 P01의 경우 CCWA 복잡도 수준을 측정하더라도 C(SSS) 복잡도 수준과 같은 수준인즉, CCWA 매트릭을 채택하더라도 전혀 영향을 받지 않는 파일이 79%이고, CCWA 복잡도 수준을 측정하면 C(SSS) 복잡도 수준보다 +5이상 수준이 높아지는 파일(수준 0→수준 5 또는 수준 1→수준 6으로 변하는 파일)이 3%를 차지하고 있다.

〈표 6〉 복잡도 변화율

프로젝트 / 복잡도 증가	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	전체
+0	79%	53%	71%	30%	57%	62%	39%	70%	68%	44%	58.8%
+1	7%	32%	11%	22%	21%	14%	20%	9%	11%	17%	14.4%
+2	0%	1%	1%	4%	1%	0%	0%	1%	0%	0%	1.1%
+3	0%	2%	0%	2%	0%	0%	0%	0%	0%	0%	0.5%
+4	11%	7%	15%	30%	18%	17%	38%	16%	18%	36%	19.4%
+5	3%	5%	1%	13%	2%	7%	3%	5%	3%	3%	5.6%
+6	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0.2%

10개 프로젝트에서 측정된 결과를 평균한 결과가 (그림 4)이다. 전체적으로 볼 때 복잡도가 변하지 않는 파일은 약 59% 정도이고 41%의 파일이 서버측 스크립트 요소가 아닌 다른 요소의 복잡도로 인하여 복잡도 수준이 증가하였다. 특히 복잡도 수준이 +4 이상으로 크게 증가한 파일이 전체의 25%로 매우 큰 비율을 차지하고 있고 특히 복잡도 수준이 +4만큼 증가한 것이 19.4%로 매우 높았다. 이것으로 서버측 스크립트 요소 이외의 것이 웹 어플리케이션의 복잡도에 큰 영향을 주고 있음을 확인할 수 있다.



(그림 4) 복잡도 변화율

위의 결과를 다시 정리하면 다음과 같다.

- 웹 어플리케이션 중 59%는 서버 스크립트 요소에 의하여 복잡도가 결정된다.
- 웹 어플리케이션 중 41%는 서버 스크립트 요소 이외의 것들에 의하여 복잡도에 영향을 받는다.
- 웹 어플리케이션 중 25%는 서버 스크립트 요소 이외의 것들로 인하여 복잡도가 매우 크게(복잡도 수준이 +4이상) 증가한다.

4. 복잡도 증가의 유형 분석

4.1 복잡도가 증가하는 웹 어플리케이션의 유형

CCWA 복잡도 수준을 측정했을 때 복잡도 수준이 크게 변화하는 원인을 분석하기 위하여 복잡도 수준의 변화가 +5이상 증가하는 234개의 웹 어플리케이션을 조사한 결과, 웹 어플리케이션의 유형을 <표 7>와 같이 4가지로 구분할 수 있었다.

〈표 7〉 웹 어플리케이션 유형

유 형	설 명
MENU	웹 어플리케이션의 상단 또는 측면의 메뉴를 구성하거나 홈페이지, 사이트 맵을 구성하는 소스 파일들이 이 유형에 속한다.
CTRL	복잡한 사용자 인터페이스를 구성하는 소스 파일들이 이 유형에 속한다.
FORM	CTRL 유형에 속하지만 주로 입력 양식을 구성하는 소스 파일들이 이 유형에 속한다.
GEN	웹 문서 편집도구의 자동 코드 생성기로 생성된 코드에 의하여 높은 복잡도가 측정된 소스 파일을 따로 분류하였다.

MENU 유형은 C(HTML) 복잡도가 높은 소스 파일을 의미한다. 대부분의 웹 어플리케이션에는 상단 또는 측면에 메뉴 시스템을 가지고 있다. 주로 이런 메뉴를 구성하는 소스들은 많이 하이퍼 링크와 OnMouseOver 같은 이벤트에 의해 구현되어 있다. 메뉴 소스 파일 이외에도 주로 HTML 요소의 복잡도가 전체 복잡도의 대부분을 차지하는 소스 파일들이 이 유형으로 분류될 수 있다. (그림 5)는 MENU 유형의 소스 코드 중 일부이다. 이 코드를 보면 메뉴를 구성하기 위하여 onmousemover, onmouseout 같은 이벤트 속성들이 많이 존재한다.

```
<DIV ID='usermanageMenu'>
<TABLE BORDER = 0 CELLPADDING = 0
CELLSPACING = 0><TR>
<TD>
<A HREF = "onmouseover = 'hideAll()' onmouseout ='hideAll()'>
<IMG SRC = /dbrms/images/menu/blank.gif WIDTH = 15
HEIGHT = 14 BORDER = 0></A></TD>
<TD  width=158 height = 14>
<A HREF = onmouseover = 'imgUsermanage(1,1)' onmouseout =
imgUsermanage(1,0)'>
<IMG SRC = /dbrms/images/menu/drop_menu/euserm_a0.gif
NAME = 'euserm_a' BORDER = 0></A>
<A HREF = "onmouseover = 'imgUsermanage(2,1)'
onmouseout = 'imgUsermanage(2,0)'>
<IMG SRC = /dbrms/images/menu/drop_menu/euserm_b0.gif
NAME = 'euserm_b' BORDER = 0></A></TD>
<TD>
<A HREF = "onmouseover = 'hideAll()' onmouseout = 'hideAll()'>
<IMG SRC = /dbrms/images/menu/blank.gif BORDER = 0
WIDTH = 15 HEIGHT = 14></A></TD>
```

(그림 5) MENU 유형 예

CTRL 유형은 C(CSS) 복잡도가 높은 소스 파일을 의미한다. 클라이언트 스크립트 요소가 많이 포함되어 있어서 복잡한 사용자 인터페이스를 구성하는 소스 파일들이 이 유행에 포함된다. 웹 브라우저 상에서 팝업 창을 실행하거나 사용자의 선택에 따라 다양한 동작을 하는 복잡한 사용자 인터페이스를 구성하는 소스 파일들이 이 유형에 속한다. 많은 웹 사이트에 주소를 입력할 때 우편번호를 검색하는 창이 새로운 창으로 실행되고, 그 창에서 주소를 선택하면 원래의 화면에 선택한 우편번호와 주소가 입력되는 기능의 소스 코드가 CTRL 유형으로 분류될 수 있다. 단 C(SSS) 복잡도가 높은 파일 중 일부는 FORM 유형으로 분류하였다.

(그림 6)은 CTRL 유형의 소스 코드의 일부이다. 이 코드는 웹 브라우저 상에서 사용자가 활용할 수 있는 동적인 달력을 구현하기 위하여 7개의 클라이언트 스크립트 함수를 사용하고 있다(코드 길이가 길어 내부 코드는 생략하였다).

FORM 유형은 입력 양식을 구성하는 소스 파일로 C(CSS) 복잡도는 높지만 비교적 단순한 입력 필드의 유효성 검사

```
<script language="JavaScript">
function day_title(day_name) {
        ...
}
function fill_table(Y,M,D) {
        ...
}
function valDate(M, D, Y) {
        ...
}
function calendarbtn(cYear,cMonth,cDay) {
        ...
}
function dayteselect(nyear,nmonth,nday,nweek) {
        ...
}
function init() {
        ...
}
</script>
```

(그림 6) CTRL 유형 예

를 수행하는 클라이언트 스크립트로 구성 유형이다. 웹 어플리케이션의 입력 양식은 입력한 필드의 길이와 유효성을 검증하기 위하여 클라이언트 스크립트를 사용하는 것이 일반적이다. 이런 경우 submit을 수행하기 전에 클라이언트 스크립트를 호출하여 각 필드에 적절한 값이 있는지 확인하고 부적절한 필드가 있는 경우 오류 메세지를 출력하게 된다. CTRL과 같이 C(SSS) 복잡도가 전체 복잡도의 대부분을 차지하지만 클라이언트 스크립트의 구조가 CTRL 유형에 비해 단순하기 때문에 별로의 유형으로 분류하였다.

(그림 7)은 전형적인 FORM 유형의 소스 코드 중 일부이다. sujeong_form() 함수는 사용자가 수정버튼을 클릭했을 때 수행되는 함수로 입력 내용이 적절한지 확인하고 최종 확인을 거친후 실제 submit()동작을 수행한다.

```
<script language = "javascript">
function sujeong_form(){
        if(document.siljeog.geumweol_naeyong.value.length>500){
                alert("금월내용을 한글 500자 이내로 적어주십시
                요.") ;
                document.siljeog.geumweol_naeyong.focus() ;
                return ;
        }

        if(document.siljeog.caweol_naeyong.value.length>500){
                alert("차월내용을 한글 500자 이내로 적어주십시
                요.") ;
                document.siljeog.caweol_naeyong.focus() ;
                return ;
        }

        if(confirm("정말로 수정하시겠습니까?")){
                document.siljeog.jageob_gb.value = "sujeong" ;
                document.siljeog.action = "../pj/pj_pjsj01_00.asp" ;
                document.siljeog.submit() ;
        }
}
</script>
```

(그림 7) FORM 유형 예

GEN 유형은 웹 문서 편집도구의 자동 코드 생성기에 의하여 생성되는 코드에 의하여 높은 복잡도가 측정된 경우이다. 분석 대상 프로젝트 중 프로젝트 P04에서 코드 생성기로 생성된 코드가 발견되어 별로의 유형으로 구분하였다. (그림 8)은 드림위버라는 개발 툴에 의하여 생성된 소스 코드이다.

```
<Script Language = "JavaScript">
// MM으로 시작하는 펑션은 즉 드림위버에서 추가된 펑션 이 펑션의 역할은 롤오버 이미지
function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i = 0 ; a&&i<a.length&&
  (x = a[i])&&x.oSrc ; i++) x.src = x.oSrc ;
}

function MM_preloadImages() { //v3.0
  var d = document ; if(d.images){ if( ! d.MM_p) d.MM_p = new
  Array() ;
        var i,j = d.MM_p.length,a = MM_preloadImages.
        arguments ; for(i = 0 ; i<a.length ; i++)
        if (a[i].indexOf("#") ! = 0){ d.MM_p[j] = new Image ;
        d.MM_p[j++].src = a[i] ; }}
}

function MM_findObj(n, d) { //v4.0
  var p,i,x ; if( ! d) d = document ; if((p=n.indexOf("?"))>
  0&&parent.frames.length) {
        d = parent.frames[n.substring(p+1)].document ;
        n=n.substring(0,p) ; }
  if( ! (x=d[n])&&d.all) x = d.all[n] ; for (i = 0 ; ! x&&i<
  d.forms.length ; i++) x = d.forms[i][n] ;
  for(i = 0 ; ! x&&d.layers&&i<d.layers.length ; i++)
  x=MM_findObj(n,d.layers[i].document) ;
  if( ! x && document.getElementById) x = document.
  getElementById(n) ; return x ;
}
```

(그림 8) GEN 유형 예

4.2 유형 분석

10개의 프로젝트에서 복잡도의 증가가 +5이상인 총 234개의 파일을 분석하여 MENU, CTRL, FORM. GEN의 네 가지 유형으로 분류하였다. <표 8>을 보면 총 234개 파일 중 40%인 93개 파일이 FORM 유형에 해당되고 29%가 MENU 유형에 해당되고 22%가 CTRL 유형에 해당되었다.

그런데 <표 8>의 파일수를 보면 다른 프로젝트에 비하여 프로젝트 P04, P08에서 복잡도의 변화가 높은 파일들이 83개, 76개로 많이 발견되었다. 그 이유를 살표보면, 프로젝트 P04는 대부분의 파일에 자동으로 생성된 코드가 포함되어 있다. 때문에 복잡도 변화가 +5 이상인 파일들 중에서 자동 생성된 코드가 주를 이루는 파일만을 GEN으로 구분하였으나, MENU, CTRL, FORM으로 구분된 소스 코드에도 자동 생성된 코드가 포함되어 있어 복잡도의 증가가 큰 것으로 나타났다. P06은 특이하게 MENU 유형의 분포가 매우 높다. P06은 다른 프로젝트와 달리 메뉴를 별로의 프레임으로 구성하지 않고 각 소스파일에 포함하는 형태로 구현되어 있어 MENU 유형의 소스 파일이 많이 발견되었다. 이는 구조 설계 단계의 문제로 볼 수 있다. 이러한 이유로 특이하게 복잡도가 높은 파일이 많이 나타난 프로젝트 P04와P06을 제외하고 <표 9>를 다시 작성하였다.

그 결과 복잡도가 +5이상 증가하는 파일의 68%가 FORM 유형이고, 24%가 CTRL 유형이며, 8%가 MENU 유형으로 나타났다. MENU, CTRL, FORM 유형들은 모두 C(SSS) 복잡도 수준에서는 발견되지 않는 유형들이다.

〈표 8〉 복잡도 유형 분석(비율)

	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	합계
파일수	8	12	4	83	10	23	4	76	8	6	234
MENU	13%	0%	25%	2%	10%	0%	50%	78%	13%	0%	29%
CTRL	50%	0%	0%	31%	70%	13%	25%	9%	25%	17%	22%
FORM	38%	100%	75%	39%	20%	87%	25%	13%	63%	83%	40%
GEN	0%	0%	0%	28%	0%	0%	0%	0%	0%	0%	10%

〈표 9〉 P04, P08을 제외한 복잡도 유형 분석(비율)

	P01	P02	P03	P05	P06	P07	P09	P10	합계
MENU	13%	0%	25%	10%	0%	50%	13%	0%	8%
CTRL	50%	0%	0%	70%	13%	25%	25%	17%	24%
FORM	38%	100%	75%	20%	87%	25%	63%	83%	68%

복잡도 유형 분석의 결과를 정리하면 다음과 같다.

- 웹 어플리케이션에는 MENU, CTRL, FORM 유형의 소스 코드가 많이 존재한다. 따라서 웹 어플리케이션의 복잡도를 낮추려면 이러한 유형에 대한 고려를 충분히 하여야 한다.
- 클라이언트 스크립트에 의하여 복잡도가 크게 증가한다. FROM 유형과 CTRL 유형은 모드 클라이언트 스크립트 요소에 의해 복잡도가 증가하는 유형이다. HTML 요소에 의해 복잡도가 증가하는 경우는 8%에 불과하다.
- FORM 유형을 자동화하면 위험 요인을 크게 줄일 수 있다. 전체의 68%에 달하는 FROM 유형을 구성하는 코드들은 대부분 입력 필드의 유효성을 검사하는 코드로 코드의 패턴이 정형화 되어 있어 자동화가 용이한 구조이다. 이 부분을 자동화 또는 모듈화 함으로써 전체 프로젝트의 복잡도를 크게 줄일 수 있다.

5. 결 론

웹 어플리케이션의 순환복잡도를 측정하기 위해 개발된 CCWA 매트릭을 복잡도 수준 지지자와 함께 사용하여 실무의 웹 어플리케이션에 적용하여 보았다. CCWA 매트릭을 채택한 CCWA 복잡도 수준을 웹 어플리케이션에 증가하면 복잡도가 4이상으로 높게 나타나는 파일의 비율이

15% 이상 증가하였다. 이러한 증가의 원인은 MENU, FORM, CTRL으로 분류되는 기존의 어플리케이션에서는 볼 수 없는 웹 어플리케이션의 유형으로 인한 것이다. 실무에서 개발된 웹 어플리케이션에서는 FORM 유형이 68%로 대부분을 구성하고 CTRL 유형이 24%, MENU 유형이 8% 발견되었다. 이러한 유형 분석의 결과는 웹 어플리케이션의 복잡도를 낮추기 위한 전략 구축에 활용될 수 있다.

웹 어플리케이션은 기존의 어플리케이션과는 확연히 다른 구조적인 특징을 가지고 있다. 웹 어플리케이션에 대한 공학적인 접근을 수행하기 위해서는 본 연구에서와 같은 개발자 관점의 접근법이 필요하며 개발 비용과 예측, 일정 계획 등의 분야에서도 본 연구에서 조사된 웹 어플리케이션의 유형을 고려한 연구가 필요하다.

참 고 문 헌

[1] 강규욱, "ASP 어플리케이션의 유지보수를 지원하는 모듈 리티 측정도구의 설계와 구현", 석사학위 청구논문, 2000.

[2] 강병도, 이미경, "웹 어플리케이션의 효율적인 개발 환경 구축에 관한 연구", 정보처리학회논문지D, Vol.10-D, No.3, pp.489-500, June, 2003.

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

[4] 이기열, 이병정, 이숙희, 우치수, "웹 애플리케이션을 위한 복잡도 척도", 정보과학회 학술대회논문집, 2001.

[5] 홍의석, 김태균, "혼성 메트릭을 이용한 소프트웨어 개체 복잡도 정량화 기법", 정보처리학회논문지D, Vol.8-D, No.3, pp.233-240, June, 2001.

[6] A. E. Hatzimanikatis, C. T. Tsalidis and D. Christodoulakis, "Measuring the Readability and Maintainability of Hyperdocuments," Software Maintenance Research and Practice, Vol.7, 1995.

[7] Athula Ginige, "Workshop on web engineering : Web engineering : managing the complexity of web systems development," Proceedings of the 14th international conference on Software engineering and knowledge engineering, July, 2002.

[8] 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.

[9] 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.

[10] Emilia Mendes, Nile Mosley, "Comparing Effort Prediction Models for Web Design and Authoring using Boxplots," IEEE, 2001.

[11] Devanshu Dhyani, Wee Keong Ng, Sourav S. Bhowmick, A Survey of Web Metrics, "ACM Computing Surveys (CSUR)," Volume 34, Issue 4(December 2002), pp.469-503, Year of Publication : 2002, ACM Press.

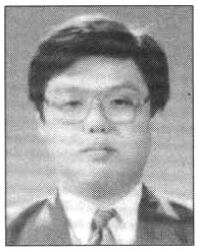
[12] Diego Bonura, Rosario Culmone, Emanuela Merelli, "Patterns for web applications," Proceedings of the 14th international conference on Software engineering and knowledge engineering, July, 2002.

[13] D. Lowe, and W., Hall, "Hypertext and the WebAn Engineering approach," John Wiley & Sons Ltd., eds., 1998.

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

[15] WebE(Web Engineering) Home : http://aeims.uws.edu.au/WebEhome/.

[16] M. Halstead, "Elements of Software Science," Elsevier North Holland, New York, 1977.



박 철

e-mail : pcman@dankook.ac.kr

1994년 단국대학교 전산통계학과(이학사)

1998년 단국대학교 대학원 전산통계학과 수료(이학석사)

2001년 단국대학교 대학원 전산통계학과 박사과정 수료

2001년~현재 단국대학교 정보컴퓨터학부 강의전임강사

관심분야 : 웹 어플리케션, 소프트웨어 공학, 분산 시스템 등

유 해 영

e-mail : yoohy@dankook.ac.kr

1979년 단국대학교 수학과(이학사)

1982년 단국대학교 대학원 수학과 수료(이학석사)

1994년 아주대학교 대학원 컴퓨터공학과 수료(공학박사)

1983년~현재 단국대학교 정보컴퓨터학부 교수

관심분야 : 소프트웨어 공학, 시스템 프로그램 등