웹 어플리케이션의 순환복잡도 메트릭스에 관한 연구

안 종 근† · 유 해 영††

요 약

웹 어플리케이션은 기존의 어플리케이션과는 다른 구조적인 특징을 가지고 있다. 웹 서버 상에서 실행되는 프로그램 코드인 서버측 스크립트 요소, 웹 브라우저에서 수행되는 프로그램 코드인 클라이언트측 스크립트 요소, 웹 브라우저에서 사용자가 선택하는 링크 요소 그리고 사용자의 선택을 클라이언트 스크립트로 연결시켜 주는 이벤트 요소, 이렇게 네 가지 요소들이 상호 연관되어 웹 어플리케이션을 구성한다. 기존의 어플리케이션들보다 다양한 요소들로 구성된 웹 어플리케이션에서 복잡도 측정에 기존의 모듈 또는 클래스 단위로 적용되는 복잡도 측정 방법을 그대로 적용하기 힘들다. 이에 본 논문에서는 웹 어플리케이션의 구조적인 복잡함을 반영하는 웹 어플리케이션의 순환복잡도(CCWA : Cyclomatic Complexity for Web Application) 메트릭스를 제안한다. 제안된 CCWA 도구를 개발하여, 현재 기업에서 사용중인 웹 어플리케이션에 적용하여 평가하였다. 그 결과 제안된 CCWA 도구는 각 요소들에 순환복잡도를 각각 적용했을 때에는 발견할 수 없었던 복잡도가 높은 웹 어플리케이션 파일을 발견하게 해주었다.

A Study of Cyclomatic Complexity for Web Application

Jongkeun Ahn† · Hae-Young Yoo††

ABSTRACT

Web applications have different structural characteristics from conventional applications. 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, link elements that the user clicks, and event elements that connect user-triggered request to the client script elements. These four elements are combined to form a web application. In such environments, direct application of conventional methods for measuring application complexity may not be possible, because they are primarily designed to measure complexity of modules and classes. In this paper, therefore, we propose metrics of Cyclomatic Complexity for Web Application (CCWA). We developed a tool to measure such metrics and applied it to the real-world examples. We found that the proposed CCWA metrics can be used for measuring complexity of highly complex web applications, which is not possible with conventional module and class based measurement techniques.

키워드 : 웹 어플리케이션(Web Application), 복잡도(Complexity)

1. 서 론

1990년대 이후에 인터넷 기술은 교육, 경제 및 사회 전반에 급속도로 퍼져 나가고 있으며 그 중심에는 월드 와이드 웹(World Wide Web)이 자리하고 있다. 이러한 인터넷 환경의 변화로 기존의 C/S 환경에서 운영되던 기업의 어플리케이션들이 웹 기술을 채택하고 있다.

인터넷 환경으로의 급격한 변화는 인터넷 사용인구의 증가와 기술의 발전이라는 커다란 효과를 가져왔지만 급속한 컴퓨팅 환경의 변화로 인한 여러 가지 문제점도 발생시켰다[1]. 초기의 웹 사이트들은 텍스트 기반의 간단한 웹 상의 문서정도였지만 최근의 웹 사이트들은 여러 가지 기능과 정보를 실시간으로 제공하기 위하여 매우 복잡하고 대형화된 어플리케이션의 형태를 가지고 있다. 과거의 웹 사이트가 문서였다면 최근의 웹 사이트는 어플리케이션이라고 할 수 있다. 이러한 웹 사이트를 웹 어플리케이션으로 부른다.

그러나 많은 웹 어플리케이션들이 기본적인 소프트웨어 구조에 대한 고려가 없이 시각적인 측면을 위주로 하여 개발되어 왔다. 실제로 많은 웹 어플리케이션들이 짧은 역사로 숙련되지 않은 개발자들에 의해 만들어지고 있으며, 오늘날 웹은 신문, 방송, 잡지 등의 온라인 정보제공 차원만이 아니라 전자상거래, 금융서비스, 온라인 교육, 인트라넷 MIS, DSS, ERP 시스템 등 모든 분야로 확대됨으로써, 편

† 종신회원 : 순천향대학교 정보기술공학부 부교수
†† 정 회 원 : 단국대학교 정보컴퓨터공학부 교수
논문접수 : 2002년 3월 7일, 심사완료 : 2002년 5월 2일

집 전문가, 시각 디자이너, 프로그래머 등의 다양한 전문가를 필요로 하게 되었고, 인터넷 사용자의 급격한 증가와 이미지나 동영상의 정보 제공은 보다 빠른 통신을 필요로 하고, 대용량의 저장장치도 필요하게 되는 등 급격한 변화가 이루어지고 있다. 그러나 아직까지도 웹 어플리케이션 개발에 적용할 수 있는 공학적인 표준이 부족한 상태이다.

복잡도(Complexity)는 소프트웨어의 유지보수성과 품질을 평가하는데 사용되는 기본적인 메트릭스의 하나이다. 과거의 연구에서는 복잡도 메트릭스를 기존의 COBOL이나 C 언어 등의 어플리케이션에 적용하였으며[2], 웹 어프리케이션에 대한 연구에서는 하이퍼문서의 가독성과 유지보수성을 측정하기 위해 사용하기도 하였다[3]. 그러나 웹 어플리케이션은 과거의 어플리케이션과는 상이함이 있어서 기존의 복잡도 메트릭스와 척도를 그대로 적용하기가 곤란하다.

Harrison, Magel, Kluczny와 Decock[4] 그리고 Conte, Dunsmore와 Shen[5]에 따르면 기존의 소프트웨어에 영향을 주는 복잡도 척도는 크게 나누어 프로그램의 크기를 측정하는 규모 척도, 프로그램 제어흐름의 복잡성을 측정하는 제어 척도, 프로그램 내의 데이터들 사이의 관계를 측정하는 데이터 흐름 척도와 한가지 요인만을 측정함으로써 나타나는 단점을 극복하기 위한 혼합적 척도로 구분했다.

대표적인 크기척도로는 LOC(line of code)와 Halstead[6]의 복잡성 척도가 있으며, 데이터 척도는 Basili의 Segment global usage pair와 Chapin의 Q Measure등이 있다. 제어 척도로는 McCabe[7]의 순환복잡도와 Sunhore의 process cyclomatic complexity등이 있다[8].

혼합적 측정방법에는 프로그램의 복잡도를 제어 논리와 규모의 쌍인 튜플로 나타낸 Hansen에 의한 측정방법과 제어흐름의 복잡도와 데이터 흐름의 복잡도를 가중치를 감안하여 더한 Oviedo에 의한 방법과 규모와 제어흐름 요인을 선형 독립인 경로와 리뷰 상수 R을 이용하여 측정한 방법이 있다[9].

또한, 최은만과 남윤석[10]은 기존의 객체지향 프로그램에 관한 메트릭스는 설계 구조가 중요함에도 불구하고 대부분의 품질 메트릭스는 복잡도나 크기, 단순한 구문 중심의 메트릭스를 사용하거나, 품질에 대한 평가보다 단순한 통계 중심 메트릭스에 불과하고, 부품의 구성과 품질에 대한 기준이 미흡한 문제점이 있다고 판단하여 이를 해결하기 위해 객체지향 부품의 품질을 구문, 스타일, 구조, 패키지화 등 다양한 측면으로 분석하여 평가하는 객체지향 기본 메트릭스를 포함한 객체지향 특유의 품질 메트릭스에 대해 제안했다.

김유경과 박재년[11]은 객체지향 설계의 특징인 크기, 복잡도, 결합도, 응집도 특성으로 나누어 설계의 품질을 평가하기 위한 메트릭 집합을 제시하였으며, 설계의 크기를 나타내는 함수에는 패키지 수, 패키지 당 평균 클래스 수, 패키지 당 평균 메소드의 수를 속성으로 갖고, 복잡도를 나타내는 함수의 인자로는 클래스 당 평균 공용 속성 수, 클래스 당 평균 메소드의 수, 클래스 당 평균 메소드의 호출수, 상속 단위의 수, 다중 상속을 받는 클래스의 수를 사용했다. 또한 결합도를 나타내는 함수의 인자로는 클래스 당 평균 포함하고 있는 클래스의 수, 클래스 당 평균 메시지 패싱의 수, 클래스 당 평균 부 클래스의 수를 사용하며, 응집도를 나타내는 인자는 속성에 대한 메소드의 평균 참조 횟수, 상속 단위에서 평균 상속 깊이, 상속 단위에서 부 클래스에 의하여 재 정의된 메소드의 평균 수, 그리소 상속 단위에서 부 클래스에 의하여 추가된 메소드의 평균 수를 속성으로 갖는 함수를 사용했다.

초기 웹 사이트 들은 대부분 온라인 출판의 형태로 브로셔웨어 기능을 하였고, 규모도 작고, 단방향적이며 정적인 형태였지만, 오늘날에는 웹 사이트에 관한 기술이 발전하고, 방문자와 웹 사이트와의 양방향 커뮤니케이션을 이루게 되었으며, 웹 사이트도 동적으로 변화하였다. 또한 CGI , Java, 데이터베이스 처리 등 전산 기술들이 추가 되면서 웹 사이트는 웹 기반의 전자우편 시스템, 인트라넷 MIS 시스템 등을 도입함으로써 소프트웨어와 구분도 모호해지게 되었다.

소프트웨어에는 여러 개의 ISO 인증 제도가 있다. 그러나 웹 사이트 개발에 있어서는 아직 품질보증 활동이 없는 것으로 알고 있다. 소프트웨어와 온라인 출판 등 어느 분야에도 확실히 속하지 않는 명확하지 않은 영역 때문일 수도 있을 것이다. 그렇지만 앞으로 웹 사이트에 대한 품질의 중요성이 커짐에 따라서 품질 활동은 분명 그 중요성이 커질 것이다[12].

웹 어플리케이션은 어플리케이션과 하이퍼문서의 결합체이기 때문에 한 가지 측정방법만으로는 측정할 수 없다. 웹 어플리케이션은 여러 가지 형태의 코드가 결합되어 있으며 서로 유기적인 연관관계를 가지고 있다. 예를 들면, 웹 서버에서 실행되는 코드는 VBScript로 작성되어 있고 순차적으로 실행된다(procedural program). 웹 브라우저에서 실행되는 클라이언트 스크립트는 주로 JavaScript로 작성하며 이벤트에 의해서 동작한다(event-driven program). 그리고 사용자가 웹 브라우저에서 보게 되는 화면은 문자, 그림, 멀티미디어 등이 결합되어 있으며 HTML(HyperText Markup Language)로 표현된다. HTML은 프로그램이라기보다 문서이며 여러 가지 편집기의 도움을 받아서 작성된다. 이러한 웹 어플리케이션의 복합적인 구성은 기존의 복잡도 메트릭스를 웹에 적용하기 어렵게 한다.

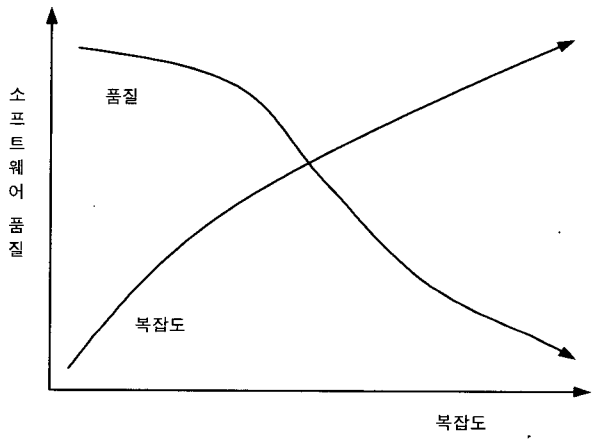
본 논문에서는 기존의 순환복잡도 메트릭스를 웹 어플리케이션에 적용하는 방법을 제시하고, 제시된 방법을 측정할

수 있는 측정 도구를 구현하여 제시된 방법의 타당성을 확인한다. 본 연구에서는 웹 어플리케이션을 SS(Server side Script) 요소, CS(Client side Script) 요소, LINK 요소, 그리고 EVENT요소로 분리하여 각각에 순환복잡도 메트릭스를 적용하고, 그 결과를 종합적으로 반영한 웹 어플리케이션의 순환복잡도(CCWA : Cyclomatic Complexity for Web Application)를 제시한다.

본 논문에서 제시한 CCWA의 타당성을 증명하기 위하여, 구현된 측정도구를 실제 기업에서 사용 중인 웹 어플리케이션에 적용하여, 그 결과를 분석한 결과를 보이고자 한다.

2. 순환복잡도 메트릭스

복잡도란 IEEE 용어집에 따르면 "데이터 구조의 형태, 내포된 정도, 조건 분기(condition branches)의 수, 인터페이스의 수 그리고 다른 시스템 특성(characteristics) 등의 요소로 결정되는 시스템이나 시스템 요소들의 복잡한 정도" 라고 정의되어있다[13]. 또한 일반적으로는 프로그램의 유지보수성, 노력 비용, 시험의 용이성 그리고 품질 등을 나타내기 위해 복잡도라는 용어를 사용한다[8]. 따라서 소프트웨어의 복잡도는 소프트웨어의 품질과 밀접한 관계가 있다고 할 수 있다. (그림 1)은 복잡도와 소프트웨어 품질간의 상관관계를 나타낸다. 그림에서 보면 소프트웨어는 복잡도가 증가할수록 품질이 떨어지는 것을 알 수 있다[14].



(그림 1) 복잡도와 소프트웨어의 품질간의 상호관계

다양한 인자를 사용한 복잡도 메트릭스는 소프트웨어 개발과 유지보수에 다음과 같은 도움을 줄 수 있다.

1. 규모가 큰 시스템의 복잡도를 측정할 수 있으므로 설계의 규모가 큰 시스템을 모듈이나 컴포넌트 단위로 분할할 수 있는 정보를 제공해준다.
2. 시스템의 구조나 모듈간의 연관성이 약간만 변화해도 결합도나 응집도의 변화 측정이 가능하므로 유지보수를 위한 정보를 제공해 줄 수 있다.

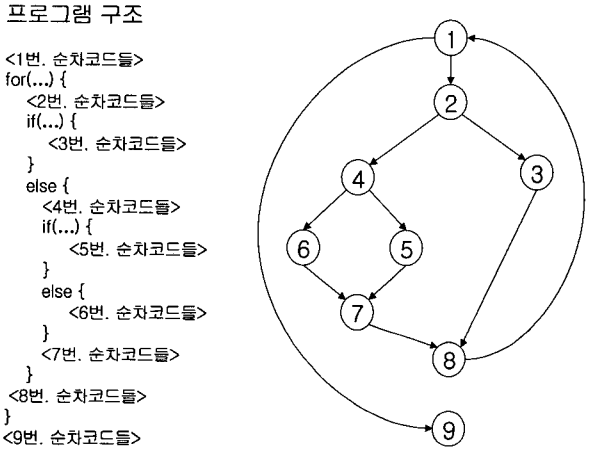
순환복잡도(Cyclomatic Complexity)는 McCabe가 제안한, 소프트웨어의 논리적인 복잡도를 정량적으로 측정하는 메트릭스다[7]. 순환복잡도는 프로그램의 논리 흐름을 표현하는 제어흐름 그래프를 기초로 하는 메트릭스로서 구조적 프로그램에서 제어흐름의 복잡도를 효과적으로 측정한다.

본 논문에서는 McCabe의 순환복잡도를 웹 어플리케이션의 복잡도 측정에 적용한다. 순환복잡도는 프로그램의 제어의 이동을 나타내는 제어흐름 그래프를 기초로 한다. 제어흐름그래프에서 노드는 제어의 분기점을 의미하고 간선은 연속으로 실행되는 일련의 프로그램 코드가 된다.

선택 경로와 루프가 많다는 것은 그만큼 프로그램이 복잡하다는 것을 의미하며, McCabe의 순환복잡도 V(G)는 프로그램의 제어흐름 그래프에 포함된 선형 독립인 경로(linearly independent path) 수로 정의된다. 선형 독립인 경로는 프로그램에서 실행 가능한 경우의 수가된다.

(그림 2)에서 선형 독립인 경로는 다음과 같이 네 가지가 있으며, (그림 2)의 예제와 같은 제어흐름 그래프를 가지는 프로그램의 순환복잡도 V(G)는 4가 된다.

경로 1 : 1-9
경로 2 : 1-2-3-8-1-9
경로 3 : 1-2-4-5-7-8-1-9
경로 4 : 1-2-4-6-7-8-1-9



(그림 2) 프로그램 코드와 제어흐름 그래프

제어흐름 그래프 G의 독립 경로의 수 V(G)는 간선의 수 e, 노드의 수 n, 그리고 연결 요소(connected component)의 수 p를 사용하여 계산될 수 있다.

$$V(G) = e - n + 2p \quad (1)$$

일반적으로 하나의 프로그램은 항상 연결된 그래프(connected graph)이기 때문에 p = 1이 되어

$$V(G) = e - n + 2 \quad (2)$$

로 사용할 수 있다.

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

$$V(G) = DE + 1 \tag{3}$$

을 사용하여 쉽게 순환복잡도를 구할 수 있다. DE는 IF, SELECT문 등의 분기 명령과 FOR, WHILE, LOOP등의 반복 명령어로 프로그램의 흐름을 변경시킬 수 있는 요소를 의미한다.

(그림 2)를 보면 e = 11, n = 9, p = 1이고 DE는 1개의 for 문과 2개의 if문이 있어서 모두 3이 된다. 식 (1)과 식 (3)을 사용하여 V(G)를 계산해 보면

$$V(G) = e - n + 2 = 11 - 9 + 2 = 4$$
$$= DE + 1 = 3 + 1 = 4$$

가 된다.

순환복잡도는 모듈의 복잡도를 측정하는데 사용되며 additive한 속성을 사용하여 전체 시스템의 복잡도를 표현하는데 사용할 수 있다. 전체 모듈러 시스템은 C를 모듈의 수라고 할 때 다음과 같이 각 모듈의 순환복잡도의 합으로 표현된다.

$$V(G) = \sum_{i=1}^{C} V(G_i) = \sum_{i=1}^{C} DE_i + C \tag{4}$$

하이퍼문서에 대한 복잡도에 대한 연구는 기존의 소프트웨어 메트릭스를 토대로 수행되었으며 Tsalidis와 Christodoulakis의 연구에서 정리되었다[3]. 이 중에서 순환복잡도 메트릭스는 하이퍼문서에 적합하도록 재해석되었다.

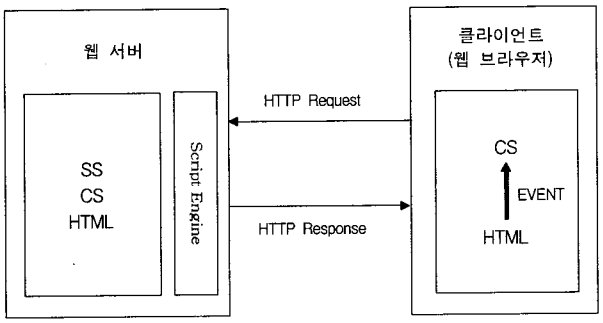
하이퍼문서에 포함된 링크는 사용자가 선택하면 다른 문서로 이동하게 된다. 즉 사용자의 동작에 의해서 경로의 흐름이 변하게 된다. 따라서 하이퍼문서의 링크를 프로그램에서의 제어 또는 반복문에 해당하는 DE로 생각할 수 있다. 따라서 하이퍼문서에서 순환복잡도는 링크의 수에 비례해서 복잡도가 커지게 된다.

3 웹 어플리케이션의 순환복잡도(CCWA)

3.1 웹 어플리케이션의 구조적 특징

웹 어플리케이션은 기존의 어플리케이션과 달리 다양한 프로그램 요소들이 결합되어 있다. 따라서 웹 어플리케이션의 복잡도를 측정하려면 서버 스크립트 코드, 클라이언트 스크립트코드, HTML 태그와 같은 여러 요소들의 복잡도를 각각 고려하여야 하고 또한 동시에 고려하여야 한다.

본 논문에서는 웹 어플리케이션의 개발 환경으로 ASP(Active Server Pages)를 기준으로 하였다. ASP는 Microsoft사에서 개발한 웹 어플리케이션 개발환경으로 현재 사용되고 있는 웹 어플리케이션 환경 중의 하나이다. ASP 환경에서 개발된 웹 어플리케이션 아키텍처는 다른 개발환경에서 개발된 웹 어플리케이션 아키텍처와 기본적으로 유사하다고 볼 수 있다. 다른 개발환경을 사용하는 경우와 측정하는 구체적인 방법에서 약간의 차이가 있겠지만 비슷한 방법으로 적용이 가능하다.



(그림 3) 웹 어플리케이션 아키텍처

(그림 3)은 웹 어플리케이션 아키텍처를 보여준다. 웹 서버에 존재하는 웹 어플리케이션 소스에는 서버에서 스크립트 엔진에 의해서 실행되는 스크립트 코드(SS : Server side Script)와 클라이언트에 그대로 전송되어지는 클라이언트 스크립트 코드(CS : Client side Script)와 HTML 태그들이 포함되어 있다. 서버측에 존재하던 소스코드는 클라이언트의 요청에 의해 클라이언트로 전송되기 전에 서버에서 SS 요소가 실행되고 실행된 결과만이 웹 브라우저로 전송된다. 클라이언트로 전송된 결과에는 CS와 HTML 요소만이 남아있게 된다. HTML 요소에서 발생되는 이벤트에 의해서 CS 요소들이 실행되어진다.

웹 어플리케이션 개발자는 SS, CS, HTML의 세가지 요소를 한 개의 소스 파일에 모두 포함하여 개발하게 된다. ASP 환경인 경우 확장명이 .asp인 파일에 SS 요소, CS 요소, HTML 요소가 모두 포함되게 된다. 그러나 이 요소들은 프로그래밍 언어의 측면에서 각각 다른 특징을 가지고 있다.

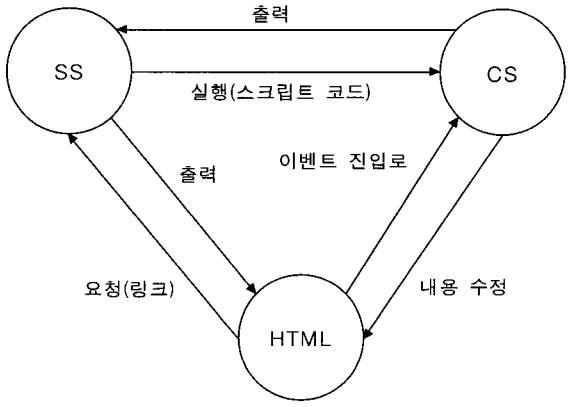
SS 요소는 전통적인 절차적 프로그램으로 만들어진다. 클라이언트의 요청에 의하여 웹 서버가 프로그램을 로딩하여, 시작에서부터 순차적으로 스크립트 명령어들을 수행한다. ASP에서는 VBScript와 JavaScript를 사용할 수 있다. 이 언어들은 모두 구조적 프로그래밍 언어로 모듈을 제공한다. SS 요소가 웹 서버에서 실행되고 나면 HTML 요소와 CS 요소가 클라이언트측의 웹 브라우저로 전송된다.

CS 요소는 전형적인 이벤트-기반 프로그램으로 만들어진다. 이벤트는 HTML 요소인 HTML 태그들에 포함되어 있는 속성인 onClick, onKeyPress 등에 의해서 지정되어진다. 사용자가 HTML 태그에 주어진 이벤트를 발생시키

는 경우 - 그림을 클릭하거나 문자열을 입력하는 등등 - CS 요소가 구동된다. CS 요소는 클라이언트의 웹 브라우저 내에서 사용자에게 메시지를 전달하고, HTML 요소의 내용을 수정하고, SS 요소를 실행하도록 웹 서버에 요청하는 일을 한다.

HTML 요소는 프로그램이기보다는 단순한 하이퍼문서를 구성한다. 그러나 개발자는 HTML 요소를 일반적인 텍스트 문서처럼 작성하지는 않는다. HTML 문서는 SS 요소의 가장 기본적인 출력 결과이다. SS는 HTML 문서를 작성하기 위해 조금은 복잡한 알고리즘을 수행한다. 또한 HTML 요소의 태그들은 CS 요소를 구동시키는 이벤트의 진입로 역할을 한다.

이와 같이 SS, CS, HTML 요소들은 각각 매우 다른 특징들을 가지고 있으면서 서로 밀접한 관계를 가지고 있다. (그림 4)는 각 요소들 간의 관계를 보여준다.



(그림 4) SS, CS, HTML 요소의 관계도

3.2 웹 어플리케이션의 순환복잡도(CCWA)

웹 어플리케이션의 경우, 특히 ASP인 경우 한 개의 소스 파일에는 SS 요소와 CS 요소와 HTML 요소가 모두 포함되어 있다. 이러한 한 개의 ASP 소스는 개발자가 코드를 작성하는 기본단위가 되며 유지보수의 단위가 된다. 이렇게 SS 요소, CS 요소, HTML 요소가 결합된 한 개의 소스 파일의 복잡도를 웹 어플리케이션의 순환복잡도(CCWA : Cyclomatic Complexity for Web Application)로 부른다.

CCWA는 SS 요소의 순환복잡도 C(SS), CS 요소의 순환복잡도 C(CS), 그리고 HTML요소의 순환복잡도 C(HTML)의 결합으로 정의되어야 한다. 지금부터 CCWA를 구성하는 C(SS), C(CS), C(HTML)에 대하여 각각 정의한다.

C(SS)는 소스 파일에 포함된 여러 개의 SS 모듈들이 결합된 복잡도로서 모듈의 수가 m일 때 순환복잡도는 additive한 속성을 가지므로 다음과 같이 표현된다.

$$C(SS) = \sum_{i=1}^{m} C(SSi) \qquad (5)$$

[3]의 연구에서 C(HTML)는 HTML요소들의 링크와 이벤트 속성을 DE로 간주하여 측정된다. HTML은 모듈을 제공하기 않기 때문에 링크의 수와 이벤트 속성의 수를 사용하여 다음과 같이 정의할 수 있다.

$$\begin{aligned} &C(HTML) = DE + 1 = C(EVENT) + C(LINK) - 1 \\ &C(EVENT) = \text{이벤트 속성의 수} + 1 \\ &C(LINK) = \text{링크의 수} + 1 \end{aligned} \qquad (6)$$

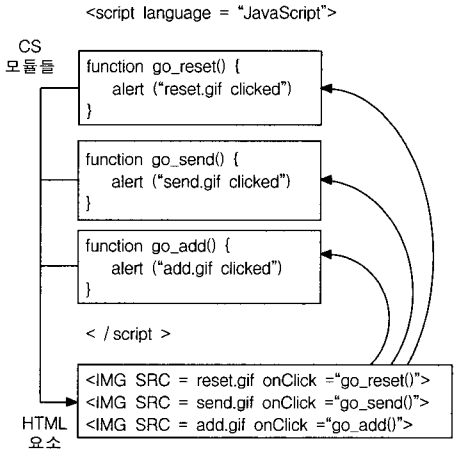
링크와 이벤트 속성은 사용자 입장에서는 모두 링크로 간주되기 때문에 C(EVENT)와 C(LINK)를 구분하는 것은 C(HTML) 요소를 정의하는 데는 필요하지 않다. 두 요소를 구분하는 것은 C(CS)를 설명하기 위해서 필요하다. 여기에서 +1의 의미는 EVENT와 LINK가 없을 경우에도 기본적인 순차처리는 존재 할 것이므로 기본 복잡도로 설정하는 것이다.

C(CS)는 C(SS)와 달리 각각의 CS 요소 모듈들의 복잡도의 결합으로 설명하기에는 부족한 점이 있다. 클라이언트 웹 브라우저에는 HTML 태그에 포함되어 있는 이벤트 속성으로부터 CS 모듈로 진입하는 제어흐름이 존재한다. 제어흐름은 HTML 태그 내에 포함되어 있으면서 전체 클라이언트 스크립트의 제어흐름에 중요한 역할을 한다. (그림 5) HTML 요소에서 이벤트 함수로의 제어흐름을 보여준다.

따라서 C(CS)는 각각의 CS 모듈의 순환복잡도와 (그림 5)에 나타난 제어흐름의 결합으로 표현되어야 한다. (그림 5)의 제어흐름의 순환복잡도는 C(EVENT)가 되므로

$$C(CS) = \sum_{i=0}^{n} C(CS_i) + C(EVENT) \qquad (7)$$

이 된다.



(그림 5) HTML요소에서 CS 모듈로의 제어흐름

웹 어플리케이션을 구성하는 3대 요소인 SS, CS, HTML 요소들의 순환 복잡도를 각각 별개의 요소로 작용하는 additive한 성질을 사용하여 계산하면

$$C(SS)+C(CS)+C(HTML) = \sum_{i=0}^{m} C(SS_i)+\sum_{i=0}^{n} C(CS_i)+C(EVENT) + C(EVENT)+C(LINK)-1 \quad (8)$$

이 된다.

식 (8)은 이벤트 요소의 순환복잡도 C(EVENT)가 두 번 중복되어 계산되어진다. 이벤트 속성은 C(CS)와 C(HTML)을 독립적으로 볼 때 양쪽에 모두 속하지만 웹 어플리케이션을 기준으로 보면 한번만 측정되어야 한다.

따라서 CCWA는

CCWA = C(SS) + C(CS) + C(HTML) - C(EVENT)

(9)로 정의한다.

3.3 모듈 수준의 복잡도 측정

복잡도는 한 개의 모듈 내에서의 제어흐름을 기반으로 하여 측정되어진다. 이러한 측정을 모듈 수준의 측정이라고 한다. 모듈 수준의 측정은 순환복잡도를 측정하는 기본 단위가 된다. 이 절에서는 SS 요소, CS 요소, LINK 요소의 순환복잡도를 측정하는 구체적인 방법을 제시한다.

ASP에서 서버측에서 실행되는 스크립트(SS) 코드의 경우에는 프로그램을 시작하는 C 언어의 main() 같은 표시가 없이 바로 프로그램이 시작된다. 스크립트 소스 파일의 시작부분에서부터 차례로 실행되는 이 프로그램 코드들이 그 소스 파일의 주 모듈이 된다. 이 모듈은 공식적인 이름이 존재하지 않기 때문에 "main"이라는 이름으로 부르기로 한다. ASP로 개발되는 프로젝트의 경우 함수의 활용도가 낮기 때문에 main 모듈이 소스의 유일한 모듈인 경우가 많고 main 모듈의 복잡도가 높은 편이다[15].

클라이언트에서 실행되는 스크립트 코드의 경우에도 역시 프로그램을 시작하는 표시가 없이 시작할 수 있기 때문에 CS의 경우에도 이러한 일련의 코드들을 "main"이라는 이름의 모듈로 정하였다. CS에서 main 모듈은 출현빈도가 비교적 적은 편이다. CS 모듈의 순환복잡도도 SS와 마찬가지로 DE를 측정하여 DE + 1로 측정한다.

HTML 요소의 복잡도는 Tsalidis와 Christodoulakis의 연구 결과[3]를 사용하여 사용자의 동작에 의해 분기가 발생하는 코드의 수(DE)를 세어서 DE + 1로 순환복잡도를 측정한다.

HTML 문서에서 사용자가 클릭하여 다른 문서로 이동할 수 있는 코드에는 두 가지 종류가 있다. 첫째는 <A> 태그를 사용하는 일반적인 하이퍼링크로 LINK요소라고 부른다. 둘째는 태그에 이벤트 속성을 포함하는 경우로서 EVENT 요소라고 부른다.

LINK 요소의 복잡도는 단순히 <A> 태그를 DE로 간주하여 측정하고 EVENT요소의 복잡도는 (그림 6)의 예에 나타나는 onClick과 같은 이벤트 속성을 DE로 간주하여 측정한다.

(그림 6)의 예는 하이퍼문서가 다른 문서로 이동하게 만들지는 못하지만 HTML문서에서의 제어가 CS 모듈로 이동하게 만든다. 즉, HTML 요소에서 발견되는 이벤트 속성은 HTML 문서의 제어흐름이 CS 모듈로 이동하게 하기 때문에 순환복잡도를 측정하는 중요한 측정요소가 된다.

```
<script language = "JavaScript">
function go_url() {
      alert("그림이 클릭되었습니다")
}
</script>
<IMG src = 'button.gif' onClick = "go_url()">
```

(그림 6) HTML에서 이벤트 속성 사용 예

4. 평 가

CCWA의 유효성을 평가하기 위하여 웹 어플리케이션을 구성하는 각 구성요소의 복잡도를 함께 고려하는 새로운 메트릭스가 단순히 서버측에서 수행되는 프로그램 요소에 대해서만 순환복잡도를 측정되었을 때에는 발견할 수 없었던 높은 복잡도의 파일을 찾아낼 수 있는지를 확인한다.

서버측 스크립트는 VBScript, JavaScript등으로 작성되는 전형적인 구조적 프로그램의 형태를 가진다. 따라서 C(SS)는 전형적인 순환복잡도와 같은 성질을 가진다고 간주하고 C(SS)와 CCWA를 비교하였다.

평가를 위하여 실제 실무에서 사용되고 있는 세 개의 프로젝트에서 CCWA와 서버측 스크립트에서만 순환복잡도를 측정한 C(SS)를 측정하고 그 결과를 비교 분석하여 다음을 보인다.

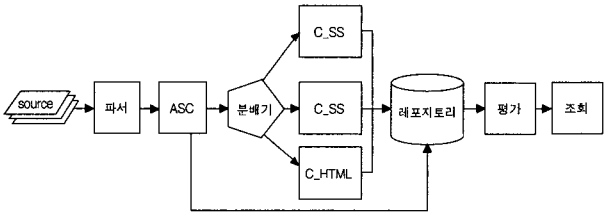
첫째, CCWA가 C(SS)가 가지는 순환 복잡도의 특징을 거의 그대로 반영하고 있음을 보인다.

둘째, CCWA와 C(SS)가 차이가 있음을 알아보기 위하여서 산포도를 통해 CCWA를 사용하여 새롭게 발견되는 복잡도가 높은 파일들이 있음을 확인한다.

셋째, CCWA를 측정했을 때 기존의 순환복잡도 측정만으로는 발견할 수 없었던 새로운 파일들을 분석하여 CCWA가 발견한 파일이 실제로 복잡도가 높았음에도 불구하고 기존의 방법으로 발견될 수 없었던 파일임을 확인한다.

실험에 사용된 세 개의 프로젝트를 각각 A, B, C라고 부른다. 프로젝트는 소스 파일(*.asp)의 수가 200개 이상인 중간 규모의 웹 어플리케이션 들이다.

실험에 사용한 시스템의 구성은 (그림 7)과 같고, <표 1>에 각 요소의 역할에 대한 설명을 나타내었다.



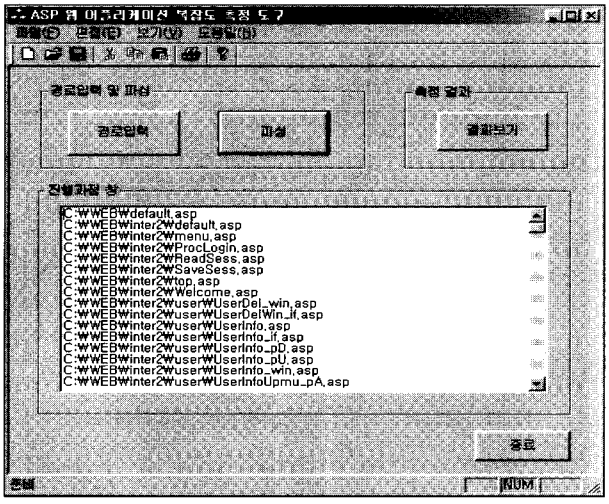
(그림 7) 실험에 사용한 시스템 구성도

〈표 1〉 시스템에 사용된 모듈들의 역할

모 듈	설 명
파 서	소스에서 사용되는 토큰을 분리한다. ASP 소스를 구성하는 VBScript, JavaScript, HTML 토큰을 모두 인식하여 분리한다.
ASC	Application Statistics Collector 전체 웹 어플리케이션에 대한 통계(LOC, 오퍼랜드/오퍼레이션의 수 등)
분배기	ASP소소에서 SS, CS, HTML 요소를 분리하여 각각을 처리하는 모듈로 전달한다.
C_SS	서버측 스크립트의 순환복잡도를 측정한다.
C_CS	클라이언트 스크립트의 순환복잡도를 측정한다.
C_HTML	HTML 요소의 순환복잡도를 측정한다.
레포지토리	수집된 정보를 저장하여 평가와 조회 모듈에서 사용한다.
평 가	레포지토리의 정보를 사용하여 복잡도를 계산한다.
조 회	전체 모듈의 복잡도 목록과 복잡도가 높은 모듈의 목록을 조회한다.

〈표 2〉 실험에 사용된 웹 어플리케이션

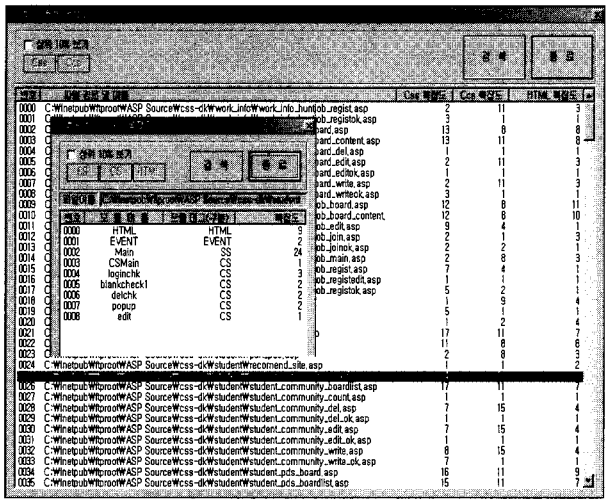
	A	B	C
성 격	업무시스템 인트라넷	상업적 웹 사이트	공공목적 웹 사이트
파일 수	243개	222개	340개



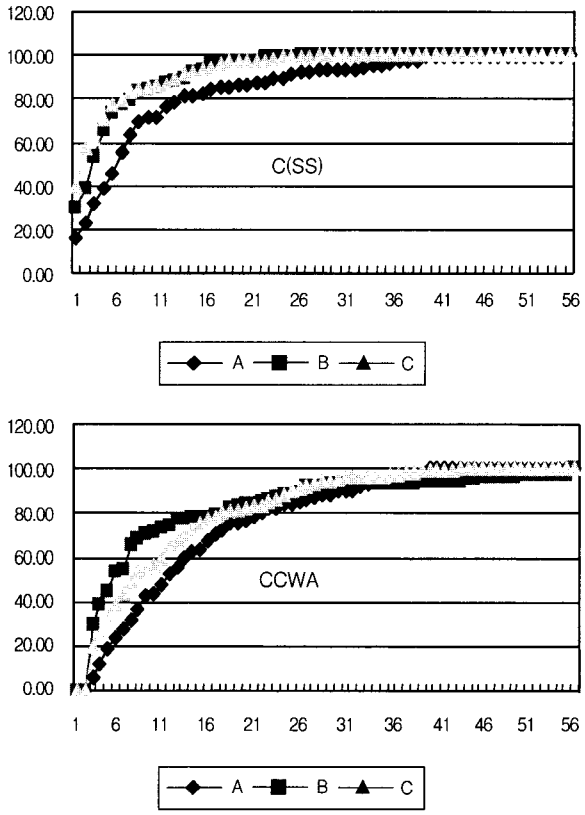
(그림 8) CCWA 측정 도구

CCWA를 측정하기 위하여 개발된 (그림 8)의 도구를 사용하여 C(SS), C(CS), C(HTML)이 측정되었다.

(그림 9)에서 컴포넌트별 복잡도 화면은 각 ASP 소스파일에 포함된 각 요소들의 복잡도를 표시하게 되며 이 중에서 하나의 파일을 선택하여 선택되어진 파일을 클릭하면 컴포넌트 모듈별 복잡도창이 나타나며, 컴포넌트의 복잡도를 구성하는 각 모듈의 복잡도가 출력된다.



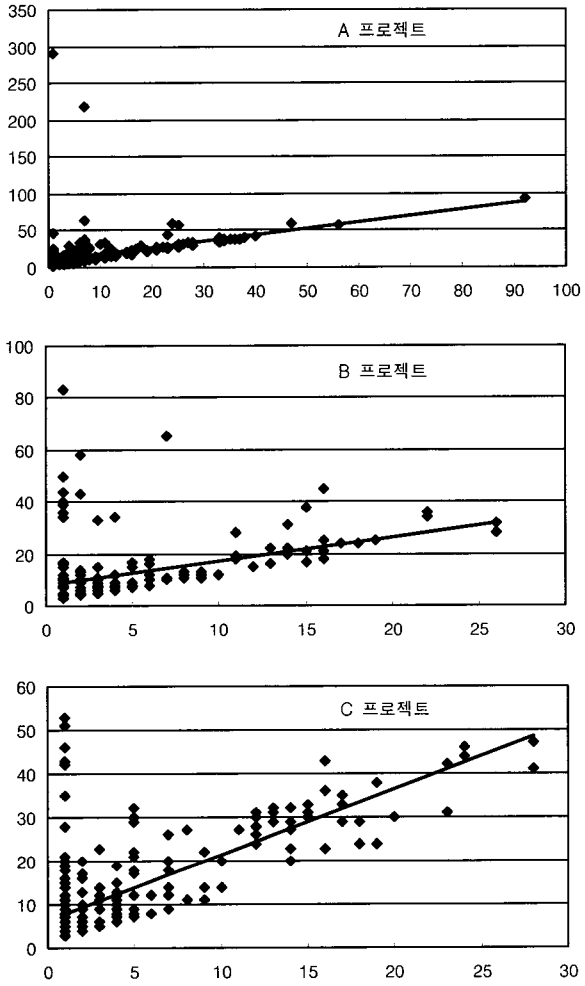
(그림 9) CCWA 실행 화면



(그림 10) C(SS)와 CCWA의 누적분포

측정된 C(SS)와 CCWA는 (그림 10)과 같다. 그림에서 X축은 순환복잡도이고 Y축은 복잡도가 X이하인 소스 파일들의 비율이다.

(그림 10)은 CCWA가 C(SS)와 유사한 경향을 가지고 있음을 보여준다. CCWA는 C(SS)를 포함하기 때문에 전체 값이 C(SS)만 측정한 것에 비해 절대 수치가 2배정도 크기는 하지만 누적된 분포의 전체적인 경향은 유사하다.



(그림 11) C(SS)-CCWA 산포도

(그림 11)은 X축을 C(SS), Y축을 CCWA로 하여 각 웹 어플리케이션을 구성하는 모든 소스 파일의 C(SS)와 CCWA의 순서쌍을 산포도로 그린 것이다. 산포도 위에 그려진 추세선은 C(SS)가 높은 파일은 CCWA도 높은 경향을 가지고 있음을 보여준다. 이것은 CCWA가 기존의 순환복잡도의 특징을 반영함을 설명한다.

그러나 세 그림에서 공통적으로, C(SS)가 낮은 쪽을 보면 추세선의 위쪽으로 길게 분포되어있는 것을 보이고 있으며, 특히 그림의 좌측상단부분에 예외적인 점들이 발견된다. 이 점들은 C(SS)가 낮은 것으로 측정되었던 파일들 중에서 CCWA를 측정했을 때는 상대적으로 복잡도가 높게 평가되는 파일들을 의미한다.

CCWA를 적용했을 때 복잡도가 높다고 새롭게 발견되는 파일들이 실제로 복잡한지 확인하기 위하여 새롭게 발견된 파일의 소스 코드를 분석하였다. 분석 대상은 C(SS)를 측정했을 때는 복잡도가 높은 것으로부터 상위 10%이내에 들지 못했던 파일이 CCWA로 측정하면 상위 10%이내에 들게 되는 파일로 하였다.

CCWA를 측정했을 때 새롭게 상위 10%이내에 들게 되는 파일은 A 프로젝트의 경우 10개, B 프로젝트 14개, C 프로젝트 8개의 소스 파일이다. <표 3>은 이 파일들의 각 요소별 복잡도와 그 소스코드의 특징 유형을 보여준다. 이 소스 파일의 코드 내용을 분석한 결과 각 파일은 대략 세 가지 유형으로 분류할 수 있었다.

첫 번째 유형은 MENU로 웹 어플리케이션의 상단 또는 측면의 메뉴를 구성하거나 홈페이지, 사이트맵을 구성하는 소스 파일로서 많은 링크와 이벤트 속성을 가지는 것이 특징이다. A 프로젝트에서 3개, C 프로젝트에서 2개의 파일이 이 유형에 해당한다.

두 번째 유형은 FORM으로서 주로 입력 양식을 구성하는 화면이다. 대부분의 웹 어플리케이션에서는 입력 폼에서 필드의 길이와 유효성을 검증하기 위하여 클라이언트 스크립트로 JavaScript를 사용한다. 여기에는 주민등록번호 또는 사업자번호 검증, 우편번호 확인 등의 복잡한 코드들이 포함된다. A 프로젝트에서 6개, B 프로젝트에서 13개, C 프로젝트에서 5개가 포함되어 있으며, 새로 발견되는 파일의 2/3가 FORM 유형에 해당한다.

〈표 3〉 CCWA에 의해 복잡도가 발견된 파일

프로젝트 A

파일명	C(SS)	C(CS)	C(HTML)	CCWA	유 형
A1	7	24	7	38	FORM
A2	7	23	6	36	FORM
A3	7	23	8	38	FORM
A4	7	133	78	218	MENU
A5	24	27	9	60	FORM
A6	1	151	138	290	MENU
A7	11	17	7	35	FORM
A8	1	34	12	47	CTRL
A9	1	24	12	47	FORM
A10	7	29	28	64	MENU

프로젝트 B

파일명	C(SS)	C(CS)	C(HTML)	CCWA	유 형
B1	1	33	16	50	CTRL
B2	1	29	4	34	FORM
B3	1	21	22	44	FORM
B4	4	26	4	34	FORM
B5	3	27	3	33	FORM
B6	4	27	3	34	FORM
B7	1	34	4	39	FORM
B8	1	34	5	49	FORM
B9	1	34	5	40	FORM

세 번째 유형은 CTRL로 웹 브라우저 상에서 JavaScript를 사용하여 복잡한 사용자 인터페이스를 구성하는 유형으로서 각 프로젝트에서 한 개씩 발견되었다. 이 유형은 웹 어플리케이션에서 구현이 어렵지만 사용자의 편의를 위하여 구현된 화면들로 구현의 난이도가 매우 높은 파일들이었다.

이상의 분석결과, CCWA는 기존의 서버측 스크립트의 복잡도 측정만으로는 발견될 수 없는 웹 어플리케이션에 실제 존재하는 높은 복잡도를 가지는 소스 파일을 발견하는데 사용될 수 있었다.

5. 결 론

초기 웹사이트들은 대부분 웹 디자이너들이 온라인 출판의 형태로 규모도 작고, 단방향적이며 정적인 형태로 설계하였지만, 오늘날에는 웹 사이트에서 CGI , Java, 데이터베이스 처리 등 전산 기술이 추가되어, 기업의 MRP, DSS, ERP 시스템을 인트라넷에서 수행하려는 요구가 늘어나고 있다. 따라서 이미 개발된 웹 사이트에 대해서 BPR(Business Process Reengineering) 하듯이 다시 기획부터 시작해서 설계, 개발을 하거나 웹 사이트 평가나 진단에 대한 요구가 늘어나고 있다.

앞으로 웹 사이트에 대한 품질의 중요성이 커짐에 따라서 웹 사이트에 대한 품질 활동은 분명 그 중요성이 커질 것이며, 소프트웨어에서처럼 여러 개의 ISO 인증 제도가 웹에서도 적용될 것이다.

기존의 소프트웨어에 영향을 주는 복잡도 척도에 대한 연구는 크게 나누어 프로그램의 크기를 측정하는 규모 척도, 프로그램 제어흐름의 복잡성을 측정하는 제어 척도, 프로그램 내의 데이터들 사이의 관계를 측정하는 데이터 흐름 척도와 한가지 요인만을 측정함으로써 나타나는 단점을 극복하기 위한 혼합적 척도를 기준으로 하여 연구되었다.

웹 어플리케이션은 서버측 스크립트 요소, 클라이언트측 스크립트 요소와 HTML요소로 구성되어 있다. 그 중에서 서버측 스크립트 요소는 전형적인 구조적 프로그래밍 언어의 형태를 가지고 있다.

본 연구에서는 전형적인 순환복잡도 특성을 보이는 서버측 스크립트 요소와 아직 획인되지 않은 클라이언트측 스크립트 요소와 HTML요소의 복잡도를 포함하는 CCWA를 측정했다. 또한 각 요소의 순환복잡도를 복합적으로 표현하는 CCWA 를 측정하는 도구를 개발하여 실제 데이터에 적용하여 그 결과를 분석해봄으로써 CCWA 메트릭스의 유효성을 입증하였다. 또한 CCWA는 서버측 스크립트에만 순환복잡도 메트릭스를 적용하였을 때에는 발견할 수 없는 새로운 복잡한 파일들을 찾아내는데 유용함을 확인하였다.

어떤 메트릭스가 실무에 적용되기 위해서는 적절한 지침(guideline)이 함께 제시되어야 한다. 그러나 본 연구에서는 CCWA가 새로운 복잡도 요인을 발견할 수 있음을 보였지만 적절한 지침을 제시하지는 못하였다. 적절한 지침을 제시하기 위해서는 향후 많은 프로젝트에서 CCWA를 측정하고, 그 결과에 대한 통계 분석이 필요하다.

참 고 문 헌

[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] 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.

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

[4] W. Harrison, K. Magel, R. Kluczny and A. Decock, "Applying Software Complexity Metrics to Program Maintenance," IEEE Computer, Vol.15, No.9, pp.65-79, Sep., 1982.

[5] S. D. Conte, H. E. Dunsmore, V. Y. Shen, "Software Engineering Metrics and Model," The Benjamin/Cumming Publishing Company, Inc., 1995.

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

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

[8] 류성열, 이성은, 안재홍, "유지보수를 위한 프로그램의 복잡도 측정요소", 숭실대학교논문집 공학편 제26집 제2호, pp.83-90, 1996.

[9] 김태공, 우치수, "프로그램경로에 기반을 둔 복잡도의 척도", 한국정보과학회논문지, Vol.20, No1, pp.34-42, 1993.

[10] 최은만, 남윤석, "재사용 소프트웨어 품질평가도구 개발", http : //se.dongguk.ac.kr/menu_data/reusequality.htm, 2001.

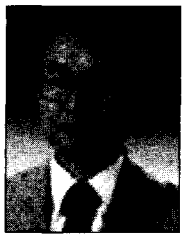
[11] 김유경, 박재년, "객체지향 설계의 특성을 고려한 품질 평가 메트릭스", 한국정보처리학회논문지 제7권 제2호, pp.373-384, 2000.

[12] 최영완, "웹 사이트 공학", http : //www.hci.or.kr/colum/19990802.html, 1999.

[13] 한규정, 이경환, "혼합적 방법에 의한 소프트웨어 복잡도 측정", 한국정보과학회논문지 Vol.16, No.2, pp.148-156, 1989.

[14] L. J. Artur, "Measuring Programmer Productivity and Software Quality," Jon Wiley ^ Sons, Inc., 1985.

[15] 강규욱, "ASP 어플리케이션의 유지보수를 지원하는 모듈리티 측정도구의 설계와 구현", 단국대학교 석사학위 청구논문, 2000.



안 종 근

e-mail : jkahn00@sch.ac.kr

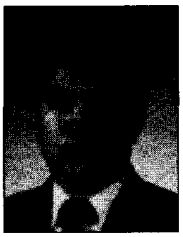
1971년 고려대학교 물리학과 졸업(학사)

1985년 숭실대학교 전산학과 졸업(석사)

1999년 단국대학교 전산통계학과 박사수료

1992년~현재 순천향대학교 정보기술공학부 부교수

관심분야 : 웹 어플리케이션, 소프트웨어 공학, 데이터베이스 등



유 해 영

e-mail : yoohy@dankook.ac.kr

1979년 단국대학교 수학과 졸업(학사)

1981년 단국대학교 수학과 졸업(석사)

2000년 아주대학교 컴퓨터공학과 박사

1983년~현재 단국대학교 정보컴퓨터공학부 교수

관심분야 : 소프트웨어 공학, 시스템 프로그램 등