韓國컴퓨터情報學會 論文誌
第15卷 第8號, 2010. 8.

2010-15-8-4-1

유사도 기반의 웹 어플리케이션 구조 복잡도

정 우 성*, 이 은 주**

A Structural Complexity Metric for Web Application based on Similarity

Woosung Jung*, Eunjoo Lee**

요 약

소프트웨어 복잡도는 대상 시스템의 유지보수성을 평가하는 주요한 메트릭인데 기존의 웹 어플리케이션 기반 복잡도는 대부분 단순히 개수 기반으로 정의되어 실제 개발자나 유지보수자의 관점에서 느끼는 이해도를 반영하기 어렵다. 이를 보완하기 위하여 정보이론의 엔트로피를 이용하여 복잡도를 정의할 수 있으나, 개별 페이지의 정보량을 동일하게 취급하고 있다. 본 논문에서는 웹 어플리케이션의 구조 복잡도를 유사도 및 정보이론에 기반하여 제안하였다. 즉, 엔트로피에 기반하되, 기존의 유사도를 이용하여 타 페이지들과 유사성이 높은 페이지의 내부 정보량은 그렇지 않은 페이지보다 낮도록 정의하여 이러한 단점을 보완하였다. 또한 관점에 따라 각기 다른 유사도를 적용할 수 있도록 함으로써 복잡도를 여러 관점에서 측정할 수 있도록 하였다. 이후 복잡도 속성을 이용하여 이론적으로 검증하였고, 사례 연구를 통하여 본 기법의 유용성을 보였다.

Abstract

Software complexity is used to evaluate a target system's maintainability. The existing complexity metrics on web applications are count-based, so it is hard to incorporate the understandability of developers or maintainers. To make up for this shortcomings, entropy-theory can be applied to define complexity, however, it is assumed that information quantity of each paper is identical. In this paper, structural complexity of a web application is defined based on information theory and similarity. In detail, the proposed complexity is defined using entropy as the previous approach, but the information quantity of individual pages is defined using similarity. That is, a page which are similar with many pages has smaller information quantity than a page which are dissimilar to others. Furthermore, various similarity measures can be used for various views, which results in many-sided complexity measures. Finally, several complexity properties are applied to verify the proposed metric and case studies shows the applicability of the metric.

▸ Keyword : 웹 어플리케이션(web application), 복잡도(complexity), 유사도(similarity)

• 제1저자 : 정우성 교신저자 : 이은주
• 투고일 : 2010. 05. 02, 심사일 : 2010. 05. 18, 게재확정일 : 2010. 05. 30.
* 서울대학교 공과대학 컴퓨터공학부 박사과정 ** 경북대학교 IT대학 컴퓨터학부 조교수
※ 이 논문은 2007년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업 연구임 (NRF-2007-331-D00407)

www.kci.go.kr

I. 서 론

현재 웹의 비중이 늘어나면서, 웹 어플리케이션의 중요성이 증대되고 있다. 웹 어플리케이션에 사용되는 기술은 다른 소프트웨어에 비하여 다양하고, 빠르게 변화하므로, 체계적인 방법론이 부족한 실정이다[1]. 특히, 생명주기가 짧으므로 웹 어플리케이션의 유지보수는 중요한 사항으로 대두되었다. 복잡도(complexity)는 유지보수성이나 이해도에 직접적으로 관련이 있는 메트릭으로, 기존에 존재하는 복잡도들은 대부분 절차적 프로그램이나 객체지향 프로그램을 대상으로 하며, 웹 어플리케이션에는 다양한 기술 및 정의들이 혼재되어 있으므로 웹 관련 복잡도에 대한 연구는 깊이 진행되지 않았다[1][2]. 또한 기존에 존재하고 있는 웹 관련 복잡도는, 유지보수자 관점에서의 이해도와 연관이 있다기보다는, 사용자 관점에서 항해 복잡도와 관련이 있거나[3], 링크의 개수 등 개수 기반으로 정의되었다[4]. [1]에서 개발자나 유지보수자 관점에서의 웹 어플리케이션 복잡도를 엔트로피이론을 이용하여 정의하고 있다. 엔트로피는 시스템의 정보량과 관련이 깊으며, 자주 참조되는 페이지의 정보량이 낮다는 것을 전제하고 있다. 그러나 [1]에서 정의된 복잡도는, 모든 페이지의 정보량이 동일하다는 것을 가정하고 있는데, 실제로는 그렇지 않다. 예를 들어 두 웹 어플리케이션 A, B에서 페이지 수, 연결의 개수 및 유형이 모두 같은데, A의 페이지들은 모두 다르며, B의 페이지들은 모두 같다고 하자. 그렇다면 분명히 사용자가 인지하기에 A, B의 복잡도는 달라야 한다. 따라서 인지되는 복잡도를 정의하기 위해서는 페이지들의 유사한 정도를 이용하는 것이 필요하다.

본 논문에서는 개발자 및 유지보수자의 관점에서 웹 어플리케이션의 구조적 복잡도에 대해 정의한다. 이를 위하여, 기존의 엔트로피 기반에서 정의되었던 복잡도 [1]를 확장하여, 각 페이지 내부의 각기 다른 복잡도를 반영하며, 유사도를 활용하여 체감 복잡도 정의에 활용한다.

본 논문의 구성은 다음과 같다. 2장에서 엔트로피 이론 및 웹 어플리케이션의 복잡도 연구들을 살펴본다. 3장에서는 본 논문에서 정의한 뷰 기반 복잡도의 정의를 설명하고 4장에서는 사례를 통해 본 연구의 유용성을 보인다. 그리고 5장에서 결론 및 향후 연구에 대해 언급한다.

II. 관련 연구

본 장에서는 우선 기존의 엔트로피에 기반한 복잡도에 대해 살펴본다. 이를 위하여 엔트로피의 개념을 소개하고, 엔트로피가 소프트웨어의 품질, 복잡도 등을 반영하고 있는 연구들을 언급한다. 그리고 웹 사이트 및 웹 어플리케이션의 복잡도 메트릭들에 대한 기존 연구들을 언급한다. 엔트로피를 이용하여 웹 어플리케이션의 복잡도를 정의한 연구들이 많지 않으므로, 개수 기반 복잡도, 항해 복잡도 등을 모두 포함해서 소개한다.

2.1 엔트로피 기반 복잡도

엔트로피는 모듈러 시스템의 응집도와 결합도 산정이나, 소프트웨어 품질 평가를 위한 수학적 모델링, 그리고 다양한 복잡도 메트릭을 정의하는 데 이용되며, 대부분 Shannon의 엔트로피[5] (〈식 1〉) 가 쓰인다[6].

$$H(P) = \sum_{i=1}^{n} p_i \log_2 p_i \quad \cdots\cdots \langle 식\ 1 \rangle$$

여기서

$p_i \geq 0 \; (i = 1, 2, ..., n)$

$\sum_{i=1}^{n} p_i = 1 \, (n \geq 1).$

〈식 1〉에서 P=(p_1,p_2, …, p_n)로서 한 소스(source) S=(s_1, s_2, …, s_n)에서 어떤 심볼 s_i가 나타나는 확률로 볼 수 있으며, H(P)은 엔트로피 함수이다. 또한 엔트로피를 소프트웨어가 담고 있는 정보양과 품질 저하 및 노화와 연계지을 수 있다 [6][7][8]. Abd-El-Hafiz는 소프트웨어 시스템을 정보 소스(information source)로 간주하고 정보양에 따라 소프트웨어 모듈을 분류하는 관점에 대해 기술한다 [6]. 만일 프로그램을 어떤 노드와 링크로 구성된 그래프로 표현할 수 있다면, 각 노드를 메시지로 보고 참조 확률 p_i를 구할 수 있으며 전체 그래프의 정보량을 엔트로피를 이용하여 측정 가능하다. Allen은 사이즈, 길이(length), 복잡도, 응집도, 결합도 등 기존의 대부분의 메트릭들이 개수 기반으로 정의되고 있으나, 소프트웨어 설계 자체가 소프트웨어의 정보를 담고 있는 추상 표현이므로 정보 이론(information theory) 을 적용할 이유가 충분하다고 주장한다[7]. Bianch등은 엔트로피를 이용하여 정의한 메트릭으로 소프트웨어의 품질저하(degradation)를 보여준다[8]. [8]에서는 소프트웨어 모델

www.kci.go.kr

에 대하여 네 가지의 엔트로피를 정의하고, 실제 소프트웨어의 유지보수 노력정도를 측정하여 모델 내 링크가 증가하면 엔트로피값도 증가하고, 시스템을 이루는 모듈의 규모(granularity)가 클수록 엔트로피 값은 감소함을 보인다. 즉 엔트로피 메트릭을 이용하여, 소프트웨어 엔지니어가 소프트웨어 시스템 노화(aging)을 모니터링 할 수 있음을 보이고 있다.

엔트로피에 기반하여 소프트웨어의 복잡도를 정의하고 있는 다수의 연구가 존재한다[9][10][11]. Harrison은 엔트로피 개념을 이용하여 소프트웨어의 복잡도를 정의하고, 에러 발생 비율을 반영하는 에러 스팬(error span)을 이용하여 이를 검증하고 있다 [9]. 에러 스팬(error span)은 프록시(proxy) 메트릭으로, 전체 토큰 수를 에러 개수로 나눈 값으로 그 값이 높을수록 에러 발생 빈도가 낮다는 것을 보여준다. Bansiya나 Kim등은 객체 지향 시스템에서 엔트로피를 이용하여 복잡도를 정의하고 있고 엔트로피 값이 높을수록 복잡도가 높음을 보이고 있다[10][11].

엔트로피 기반 복잡도는 수치 자체보다 그 순서에 더 의미가 있다[9]. 즉 어떤 어플리케이션 P의 복잡도가 3.5이고 Q의 복잡도가 4.0이라고 해서 Q가 P보다 0.5만큼 복잡하다는 의미를 부여하지 않는다. 엔트로피 기반 복잡도는 여러 WA들 사이에서 복잡도의 순위를 내거나 WA의 클러스터들에 대하여 복잡도가 높은 상위 20%의 클러스터들을 추출한다거나 하는데 이용될 수 있다.

본 연구에서의 복잡도는 웹의 링크 구조에 기반한 전체 구조적인 복잡도를 가리킨다. 웹 어플리케이션의 구조가 기본적으로 페이지들간의 연결에 기반하고, 웹 어플리케이션이 가지는 다양한 관계들을 반영하여 노드와 링크로 구성된 그래프로 모델링이 가능하므로 엔트로피를 이용하여 복잡도를 정의하는 접근법을 적용할 수 있다. 기존의 웹 어플리케이션의 복잡도는 대부분 항해 구조에 기반하고, in/out 링크의 개수를 이용하여 정의하고 있으나 엔트로피 기반으로 정의된 복잡도는 구조를 담을 뿐 아니라 정보량의 의미까지 추가되므로 실제 유지보수 노력과 연계시킬 수 있다.

2.2 웹 복잡도

웹 어플리케이션과 관련하여 엔트로피를 직접적으로 적용한 연구들은 거의 존재하지 않고, 링크 개수나 항해 구조 등에 기반하여 웹 복잡도를 정의한 연구들이 있다[3][4][12]. [12]에서는 기존의 웹 관련 연구들이 UI쪽에 초점을 맞춘 것과 달리 WA를 구성하는 모든 요소들을 고려하여 클래스 및 메소드의 개수 등을 이용하여 웹 어플리케이션의 복잡도를 구현 전에 예측할 수 있는 방법을 제안하였다. 여기서의 복잡도 예측은 개수 기반이며, 목적 및 범위 역시 본 연구와 차이가 있다. Mendes 등은 정적 웹 어플리케이션에 적합한 다음 메트릭을 소개하고 있다; 내부 링크(internal link)의 총 개수인 connectivity, connectivity를 페이지 수로 나눈 connectivity density, 페이지 내의 media 유형을 고려한 total page complexity, 그리고 cyclomatic complexy [2]. 항해 구조(Navigability structure)는 웹의 복잡도를 측정하는 데 주요한 요소가 된다. Zhang 등은 웹사이트의 구조적 복잡도 메트릭들을 제안하여 navigability를 측정하고자 하였다[3]. 여기서의 관계는 사용자가 클릭으로 이동할 수 있는 하이퍼링크만을 대상으로 한다. Zhang등이 정의한 메트릭은 다음과 같다: Out-link의 총 합인 WSC1, WSC1의 평균값인 WSC2,웹의 그래프 모델에서 독립적인 path의 개수인 WSC3, WSC3의 평균값인 WSC4, 그리고 링크의 분포를 나타내기 위한 WSC5이다. 이들은 사용자 입장에서 항해의 용이함을 측정하기 위한 것으로 본 논문에서의 웹 어플리케이션 복잡도와는 목적에서 차이가 있다. Ivan 등은 McCabe의 순환복잡도(cyclomatic complexity)를 이용하여 완전복잡도 대비 현재 복잡도의 비율로서 웹 어플리케이션의 구조 복잡도를 정의하였다[13]. 예를 들어, 현재 웹 어플리케이션이 완전이진트리 형태로 나타난다면 복잡도는 1이 된다. 그러나 복잡도 지수 자체가 구조 유형이나 개체의 개수 등을 전혀 반영하지 못한다. Mao와 Lu는 웹 어플리케이션의 구조적 복잡도를 내부(intra) 및 상호(inter) 수준에서 정의하였다 [14]. 내부 수준의 복잡도는 페이지나 컴포넌트 내의 제어 및 데이터 흐름을 이용하여 정의하였고, 상호 수준의 복잡도는 페이지나 컴포넌트 등 웹 개체들 사이의 항해 및 데이터 상호작용을 이용하여 정의하였다. 여기서 상호 수준 복잡도가 본 논문에서 정의한 구조적 복잡도와 비견할 수 있지만, 이 역시 개수 기반으로 정의되었다. Chandra와 Manjunath는 웹에 대한 상호작용의 관점에서 항해 복잡도를 정의하였다[15]. 여기서의 복잡도는 사용자가 웹 사이트를 처음 방문하여 목적지에 도달하기까지의 인지적인 부담(cognitive load)을 나타내며 웹을 그래프의 형태로 모델링한 후 Shannon의 엔트로피 이론을 적용하여 정의되었다는 점에서 본 논문과 유사하지만, 그 목적이 웹 사이트의 항해 복잡도 측정이므로 웹 어플리케이션의 유지보수와 관련된 본 논문의 목적과는 차이가 있다. 최근 프레임워크 기반 웹 어플리케이션의 비중이 점차 증대되고 있는데[16], 웹 어플리케이션 프레임워크를 측정하는 연구는 거의 없다[17]. Laakso와 Niemi는 AJAX가능 자바 기반 웹 어플리케이션 프레임워크를 평가하는 기준을 제시하였다[17]. 여기서 GQM(Goal Question Metric) 접근법을 쓰

www.kci.go.kr

는데, 본 논문에서의 목적과 관련된 유지보수 노력도 평가에서는 웹 어플리케이션의 사이즈 메트릭, 즉 HTML이나 SHTML의 파일 개수, 애플릿이나 자바 스크립트, CGI 스크립트 개수, 그리고 코드 라인 수를 활용하고 있다. [1]에서는 엔트로피 기반으로 정의되었으나, 다음 장에서 설명하듯이 실제 인지되는 복잡도를 반영하기 어렵다. 아래 <표 1>은 전기한 웹 복잡도 관련 연구들을 요약한 것이다. 대상이 웹 사이트의 경우는 대부분 항해 용이성(navigability) 측정을 위한 것이며, 웹 어플리케이션의 경우에는 유지보수성과 관련이 있다. 본 논문에서의 대상은 웹 어플리케이션이며, 목적은 유지보수이고 엔트로피 기반 정보량을 이용하였다는 점에서 [1]과 유사하지만, 실제 사용자의 인지적인 익숙함을 반영하기 위하여 유사도를 적용하였다는 점에서 차이가 있다.

표 1. 웹 복잡도 관련 연구 요약
Table 1. Summary of Existing Web Complexity Studies

	웹사이트	웹어플리케이션	웹 어플리케이션 프레임워크
개수	[3][4]	[12][14]	[17]
순환 복잡도		[13]	
엔트로피	[15]	[1]	

III. 유사도 기반 복잡도

3.1 배경

웹 페이지 복잡도는 내용 복잡도를 정의하기 위하여 필요하다. 이전 연구에서는 각 노드(페이지)의 정보량이 동일하다고 가정하였으나, 실제 웹 페이지들을 리뷰할 때, 각 페이지가 담고 있는 정보량이 각기 다르기 때문에 개별 페이지 복잡도를 우선 정의해주어야 한다. 개발자나 유지보수자는 로직이나 데이터 관련 연산을 담당하는 컴포넌트, 페이지 사이의 연결, 파라미터 전달, 클라이언트 및 서버 side script 등에 초점을 둔다. 본 논문에서의 복잡도는 유지보수자들을 타겟으로 삼은 것이므로 유지보수자의 뷰에 해당하는 요소들만을 웹 페이지 복잡도로 삼는다. 극단적으로, 페이지1에서 100개의 이미지 파일을 embed하고 있으며, 외부로 연결이 전혀 없다고 하고, 페이지2에서 100개의 하이퍼링크 및 서브밋 관계가 존재하며 매 관계마다 두 개씩의 파라미터가 있다고 하자. 유지보수자 입장에서 페이지1보다 페이지2를 읽고 이해하여 검증하는 데 들이는 노력이 훨씬 더 클 것이다. 따라서 본 논문에서는 개발자 및 유지보수자의 뷰에 속하는 요소들을 이용하여 웹 페이지의 복잡도를 정의하되, 페이지의 유사도 비율등을 고려하여 실제 느끼는 복잡도를 정의하도록 한다.

페이지를 이해하는데 걸리는 시간이나 비용은 단순히 페이지 개수나, 링크수 기반의 수적인 양에 비례하진 않고, 실질적으로 겹치는 부분을 제외한 복잡도에 더 상관관계가 높을 것이다. 즉 똑같은 100 페이지들로 구성된 웹 어플리케이션을 이해함에 있어서, 유사한 페이지들이 많을 수록 그렇지 않은 경우보다. 이는 실제로 유지보수 비용이 단순히 LOC(lines of codes)등과 같은 개수 기반으로는 산정되기 어렵다는 것을 보여주는 셈이다.

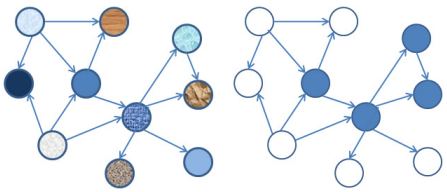

그림 1. 두 웹 어플리케이션 비교 (I)
Fig. 1. Comparison between two WAs (I)

<그림 1>의 (가)와 (나)는 노드 및 연결은 같으나 특정 기준에서의 유사도가 차이가 나는 두 웹 어플리케이션을 모델링한 것이다. 여기서 같은 무늬의 노드는 유사한 페이지들임을 말해주며, <그림 1>의 (가)는 모든 노드들이 각기 서로 다른 경우이다. 기존에 정의된 엔트로피 기반 복잡도([1])나 개수 기반 복잡도([3])에서는 두 WA의 복잡도가 동일할 것이다. 그러나 실제 소스 코드를 읽고 이해하는 입장에서는 유사한 부분을 자주 접할수록 이해도도 높아지고, 덜 복잡하게 느껴질 것이다.

또한 관심이 있는 영역에 차이가 있을 수 있다. 예를 들어 어떤 개발자는 웹 페이지들 사이의 하이퍼링크 연결 구조에 초점을 맞출 수 있고, 또 다른 개발자는 페이지들 사이의 HTML 태그 구조에 초점을 맞출 수도 있다. 개발자들은 자신이 초점을 맞춘 부분을 위주로 보게 되므로, 같은 웹 어플리케이션에 대해서도 아래 <그림 2>와 같이 '유사함'에 대해서도 다른 관점으로 볼 수 있다.

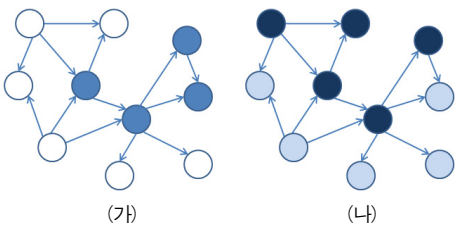

그림 2. 두 웹 어플리케이션 비교 (II)
Fig. 2. Comparison between two WAs (II)

www.kci.go.kr

따라서 개발자나 유지보수자가 초점을 맞추고 있는 부분에 대하여 적합한 유사도 메트릭을 써야할 것이다. 예를 들어 페이지의 HTML 태그 정보를 위주로 보고자 한다면 [18]에서 정의된 유사도 측정 기법을 활용하면 될 것이다. 또한 만일 외부로 향하는 연결 구조에 대해 초점을 맞춘다면 [19]에서 정의된 유사도 WSIM을 이용하면 될 것이다.

3.2. 복잡도 sb-WCOX 정의

본 절에서는 유사도 기반 웹 복잡도인 sb-WCOX(similarity-based Web Complexity)를 정의한다. 아래 〈그림 3〉은 sb-WCOX의 개요를 보여준다. 기존의 웹 어플리케이션 복잡도 메트릭들은 대부분 개수 기반으로서 개발자의 이해도를 고려하기 어려웠고, 이를 보완하기 위해 나온 엔트로피기반 복잡도 WCOX[1]는 페이지 내부의 정보량이 모두 동일하다는 가정을 함으로써 실제 적용되기에는 어려운 점이 있었다. 본 논문에서는 페이지 내부의 정보량을 유사도를 이용하여 각기 다르게 정의하고 있다.

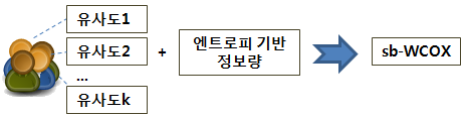

그림 3. sb-WCOX 개요
Fig. 3. Overview of sb-WCOX

웹 어플리케이션은 페이지를 나타내는 노드와 연결을 나타내는 에지로 구성되어 있다고 가정한다. 두 페이지 사이에 여러 관계가 있을 수 있으나 에지는 단 하나 존재하며, 그 모든 연결에 대한 정보를 포함한 가중치를 가지고 있다. 그 가중치는 연결의 유형, 즉 hyperlink, include, submit, load, redirect와, 파라미터의 개수를 고려하여 정의한다[1][20]. 에지의 가중치는 [1]을 참조하여 아래 〈정의 1〉과 같이 정의하였다. 여기서 연결의 방향은 실제 참조되는 확률을 고려하기 위하여 인(in) 방향의 연결만을 고려한다.

[정의 1] 에지 가중치 w(a, b)

$$w(a,b) = \sum_{R \in Rel(a,b)} w(R) \cdot (par(R)+1)$$

여기서

Rel(a, b): 노드 a, b 사이에 존재하는 모든 관계들의 집합 (hyperlink, include, submit, load, redirect)

w(R): 관계 R의 가중치로 0-1사이의 값을 가짐

par(R): 관계 R에서 존재하는 파라미터의 개수

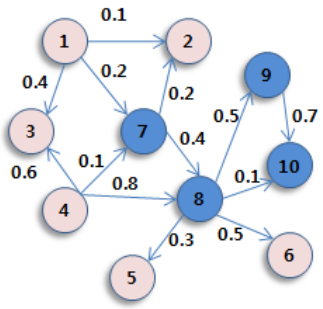

그림 4. 설명을 위한 예시
Fig. 4. An Illustrative Example

〈그림 4〉는 10개의 노드와 13개의 에지로 이루어진 웹 어플리케이션의 예제이며, 해당 가중치는 임의로 주어진 것이다. 여기서 노드 {1, 2, 3, 4, 5, 6}과 노드 {7, 8, 9, 10}이 유사하다고 가정한다.

아래 〈정의 2〉는 에지 가중치를 이용한 참조확률에 대한 정의이다.

[정의 2] 참조확률 P(n)

$$P(n) = \frac{w_{IN}(n)}{\sum_{m \in Pages(WA)} w_{IN}(m)}$$

여기서

n: 노드(페이지)

Pages(WA): WA에 속하는 페이지들의 집합

wIN(n): 노드 n의 in-edge의 가중치

예를 들어 〈그림 4〉에서 노드 7(n7)의 참조확률 P(n7)은 n7의 인에지의 가중치 합인 0.3을 전체 인에지의 가중치 총 합인 4.9로 나눈 0.061이 된다.

이전의 복잡도는, 각 노드 n의 정보량이 동일하다고 가정하여 2장 1절의 〈식 1〉과 같이 정의되었으나 실제로는 그렇지 않다. 즉 해당 페이지 내부가 한 라인으로 이루어진 간단한 경우와, 수천 라인으로 이루어진 복잡한 경우에, 비록 이들이 참조확률이 같더라도 실제 정보량에서는 차이가 날 수 밖에 없다. 따라서 본 논문에서는 실제 정보량을 구하기 위하여 페이지 내부의 정보를 반영한다. 이때 이용하는 것이 페이지의 유사도이다. 예를 들어 WA에 속한 페이지들 사이에 공통된 코드조각이 많다면, 그렇지 않은 경우보다 이해하는 데 노력이 덜 들 것이다. 왜냐하면 중복된 부분들을 반복해서 보게 되는 것과 같기 때문이다. 그리고 유사도를 측정하는 다양한 연구들이 존재하는데, 전기한 바와 같이 관점에 따라 적합

www.kci.go.kr

한 유사도를 선택하여 활용가능하다. 관점에 따른 적절한 유사도 메트릭을 선택하여 해당 페이지와의 유사도를 측정하게 되면 그 결과 유사도 벡터가 나오게 된다. 이를테면 WA가 m개의 웹 페이지로 이루어져있을 경우 어떤 노드 ni의 유사도 벡터는 다음과 같다.

〔정의 3〕 노드 ni의 유사도 벡터 V(ni)
V(ni)=〈sim(ni, n1), sim(ni, n2), …, sim(ni, nm)〉
여기서
m: 전체 페이지의 개수
sim(a, b): 페이지 a, b가 유사하면 1, 그렇지 않으면 0

sim(a, b)에서의 유사성 판단은 페이지를 보는 관점에 따라 적절한 유사도를 이용하여, 두 페이지의 유사도가 정한 임계치를 넘으면 유사하다고 판단하고, 그렇지 않으면 유사하지 않다고 판단한다. 〈그림 4〉의 예시에서 노드 7(n7)의 유사도 벡터 V(n7)은 〈0, 0, 0, 0, 0, 0, 1, 1, 1, 1〉일 것이다. 유사한 비율이 높을수록 이후 정보량은 낮아지는 방향으로 적용된다.

〈정의 2〉〈정의 3〉 및 〈식 1〉을 이용하여 유사도를 이용한 새로운 정보량을 아래 〈정의 4〉과 같이 정의한다.

〔정의 4〕 실제 정보량 Inew(n)

$$I_{new}(n) = (1 - SimVal(n)) \cdot I_{old}(n)$$

여기서

$$SimVal(n) = \frac{|\#of\ 1 \in V(n) - 1|}{|\#of\ nodes - 1|}$$

$$I_{old}(n) = -\log_2 P(n)$$

SimVal(n)의 분자에서 1을 빼는 이유는, 기본적으로 유사도 벡터에서 자기 자신과는 늘 유사하다고 나오기 때문이다. 이때 유사도는 기존에 존재하는 유사도 메트릭을 사용 가능하며, 이해하려는 대상에 따라 각기 다른 종류의 유사도를 사용할 수 있다. 예를 들어 내부 컨텐츠는 제외하고 연결 구조의 유사성만을 고려한다면, 구조 유사도를 이용하여 복잡도를 평가할 수 있을 것이다. 그 경우, 연결 구조의 유사 비율이 높다면 구조 관점에서의 복잡도는 낮게 측정될 것이고, 컨텐츠는 유사하더라도 구조의 유사성이 떨어진다면, 복잡도는 높게 측정될 것이다. 즉, 유지보수자의 목적에 맞는 적절한 유사도를 선택하여 활용할 수 있다. 〈그림 4〉의 예시에서는, 노드 7의 SimVal은 0.333이며, 노드 7의 정보량은 노드 7의 구 정보량인 0.403에 (1-SimVal)인 0.667을 곱한 0.269가 된다. 만일 유사한 페이지가 하나도 없는 경우라면 새로운 정보량은 구 정보량과 같게 된다.

아래 〈정의 5〉는 〈정의 4〉의 실제 정보량 개념을 활용하여 웹 어플리케이션WA의 최종 복잡도를 정의한 것이다.

〔정의 5〕 웹 어플리케이션 최종 복잡도 sb-WCOX(WA)

$$sb-WCOX(WA) = \sum_{n \in Pages(WA)} I_{new}(n) \cdot P(n)$$

sb-WCOX(WA)는 2장에서 소개한 Shannon의 엔트로피 〈식 1〉에 유사도를 반영하여 정의된 새로운 정보량 및 해당 노드의 참조확률을 적용하여 정의되었다.

본 절에서는 유사한 정도가 높은 특수한 경우에 대하여 본 연구에서 제안한 유사도를 이용한 복잡도와 유사도를 고려하지 않은 복잡도의 결과를 비교하기 위하여, 〈그림 1〉의 예시를 이용하여 sb-WCOX의 특징을 설명한다. 그리고 유사도의 관점에 따라 다른 결과를 보여주는 경우에 대하여 같은 WA에 대해 다른 유사도 관점에 따라 다른 복잡도 결과가 나오는 것을 〈그림 2〉의 예를 통해 보인다.

〈그림 1(가)〉는 같은 구조 및 파라미터로 인하여 같은 에지 가중치를 가지되 특정 유사도를 적용하였을 때 유사도가 낮은 페이지들의 집합인 WA이며, 〈그림 1(나)〉는 유사도가 높은 페이지들의 그룹들로 구성된 WA이다. 이들의 가중치가 〈그림 4〉와 같다고 할 때, 정의된 복잡도는 아래 〈표 2〉와 같다.

표 2. 〈그림 1〉의 sb-WCOX
Table 2. sb-WCOX of Fig. 1

	WCOX	sb-WCOX
〈그림 1(가)〉	2.804	2.804
〈그림 1(나)〉		2.522

〈그림 1(가)〉의 경우, 모든 페이지들이 서로 유사하지 않으므로 이 경우 sb-WCOX는 기존 WCOX와 차이가 없다. 그러나 유사한 페이지들로 구성된 〈그림 1(나)〉의 경우 복잡도 값이 줄어들었다. 즉, 유사도가 높은 페이지들로 구성된 WA의 경우 그 인지되는 복잡도가 그렇지 않은 쪽과 차이가 있어야 하며, 기존의 WCOX는 이를 반영하지 못하지만, 본 논문에서 제안한 sb-WCOX가 이를 반영함을 보여준다.

〈그림 2(가)〉와 〈그림 2(나)〉는 동일한 WA에 대하여 각기 다른 관점으로 유사성을 판단한 결과이다. 역시 가중치가 〈그림 4〉와 동일하다고 가정하였을 때, 복잡도를 적용한 결과는 아래 〈표 3〉과 같다.

표 3. 〈그림 2〉의 sb-WCOX
Table 3. sb-WCOX of Fig. 2

	WCOX	sb-WCOX
〈그림 2(가)〉	2.804	2.522
〈그림 2(나)〉		2.086

www.kci.go.kr

두 개발자가 각기 다른 관심사를 가지고 페이지를 검토할 때, 이들의 관심사에 대한 유사도를 적용하면 각기 다른 복잡도 결과가 나와야 하며, 〈표 3〉를 통하여 제안된 sb-WCOX가 이를 반영해 줌을 알 수 있다.

IV. 검증

4.1 이론적 검증

복잡도에 대한 다양한 이론적 검증 방법이 존재한다 [21][22]. [22]은 최적화에 기반하여 복잡도 메트릭들을 선택하는 프레임웍을 제안한다. [21]은 복잡도가 가져야 할 속성을 제안하고 있으며 다수의 복잡도 연구에서 복잡도 검증에 활용하는 기준이 되고 있다. 본 연구에서도 우선 [21]을 이용하여 복잡도 속성의 대부분을 만족함을 확인한다.

다음은 Weyuker의 9가지 복잡도 속성에 대한 검증이다.

속성 1. $(\exists P)(\exists Q)\ (|P| \neq |Q|)$

복잡도가 다른 두 웹 어플리케이션이 존재한다. 노드나 에지의 개수, 혹은 유사한 노드의 비율 등에 따라 다른 복잡도를 가지는 웹 어플리케이션이 무수히 많이 존재한다.

속성 2. 같은 복잡도를 가지는 유한 개의 웹 어플리케이션이 존재한다.

이는 정의된 복잡도의 규모(granularity)에 관한 속성으로 같은 복잡도를 가지는 WA의 개수가 작을수록 미세함을 뜻한다. 참조확률은 에지의 개수로 정의되는데 같은 참조확률에 대하여 생길 수 있는 에지의 경우는 무한하다. 따라서 속성2를 만족하지 않는다.

속성 3. $(\exists P)(\exists Q)\ (P \neq Q\ \&\ |P| = |Q|)$

P, Q가 다른 기능을 제공하는데 복잡도가 같은 경우가 존재한다. 어떤 검색 WA P, Q에서 P는 x라는 제품을, Q는 y라는 제품을 찾아 준다고 하자. 만일 P와 Q의 연결 구조 및 유사 정도가 동일하고 파라미터 개수가 같다면 P와 Q의 복잡도는 같을 것이다. 그러나 기능은 다르다.

속성 4. $(\exists P)(\exists Q)\ (P = Q\ \&\ |P| \neq |Q|)$

P, Q가 같은 기능을 제공해 주더라도, 복잡도는 다른 경우가 존재한다. P, Q가 모두 z 관련 제품을 검색해 주는 시스템이라고 하자. 사용자 정보 입력, 제품 데이터 양식 등에 따라 검색 폼, 파라미터 등이 다를 수 있으므로 P, Q의 복잡도는 다를 수 있다.

속성 5. $(\forall P)(\forall Q)\ (|P| \leq |P;Q|\ \&\ |Q| \leq |\ P;Q\ |)$

이것은 단조성에 관한 속성이다. 어떤 P에 Q가 추가된 WA의 복잡도는 P와 Q의 복잡도보다 크거나 같다. P와 Q가 병합된 웹 어플리케이션의 노드 및 에지의 개수는 증가한다. 그렇다고 하더라도 P+Q의 복잡도가 항상 증가하는 것은 아니므로 속성 5를 만족하지 않는다.

속성 6.a. $(\exists P)(\exists Q)(\exists R)\ (|P| = |Q|\ \&\ |P;R| \neq |Q;R|)$

속성 6.b. $(\exists P)(\exists Q)(\exists R)\ (|P| = |Q|\ \&\ |R;P| \neq |R;Q|)$

property 6.a와 6.b는 프로그램을 앞 혹은 뒤에 추가한 경우에 대한 것이다. 웹 어플리케이션의 경우, 앞에 추가하거나 뒤에 추가하는 것을 구분짓는 것이 무의미하므로 6.a에 관한 것만 설명하기로 한다. 같은 복잡도를 가지는 P, Q에 대해 R을 추가할 경우, 추가된 두 클러스터의 복잡도가 다른 경우가 존재한다. R을 추가할 경우, R과 기존의 P, Q에 대해 새로운 관계가 설정이 될 수 있다. R내부의 페이지와 P 내부의 페이지 사이에 발생하는 관계와, R과 Q 내부 페이지 사이에 발생하는 관계의 유형 및 파라미터 개수가 다를 수 있으므로, 병합된 두 클러스터의 복잡도가 다를 수 있다.

속성 7. P내의 statements의 순서를 변경한 Q가 있을 때 $|P| \neq |Q|$ 이다.

이 속성은 WA에서는 에지 방향이 변경됨을 뜻하는데, 에지 방향이 변경이 되면 참조확률이 달라지므로 복잡도도 달라진다.

속성 8. If P is renaming of Q, then $|P| = |Q|$

속성 8은 프로그램 내의 변수 이름 등에 복잡도의 의미를 부여하지 않는 것이다. 본 논문에서 정의한 복잡도는 노드나 링크의 이름이나 파라미터의 이름과는 상관이 없으므로 속성 8을 만족한다.

속성 9. $(\exists P)(\exists Q)\ (|P| + |Q| < |P;Q|)$

두 웹 어플리케이션이 병합되고, 새로운 관계가 추가될 수 있고, 유사도 벡터도 변화가 생기므로 각각의 복잡도보다 병합 이후의 복잡도가 더 커질 가능성이 있다.

본 논문에서 정의한 복잡도는 Weyuker의 복잡도 속성 2와 5을 제외하고 만족한다. Kim et al.이 정의한 엔트로피 기반 객체지향 복잡도 역시 5와 7을 만족하지 않는다 [11]. 그리고 Zhang등의 다섯 Web site의 복잡도 메트릭에서는 평균 3개의 속성을 만족하지 않는다. 또한 Weyuker의 복잡도 특성이 적용되는 시스템과, 본 논문의 시스템의 특성이 차이가 존재하므로, WCOX는 복잡도를 나타내는 척도로 충분히 이용될 수 있다.

4.2 사례 연구

본 절에서는 jsp로 구현된 오픈 소스인 GIMS[23]을 대상으로 하여 제안한 복잡도 메트릭을 적용한 결과를 보인다.

www.kci.go.kr

GIMS는 개발자 관점에서의 유사한 페이지들이 첨부한 문서에 명기되어 있어, 타 유사도 메트릭과의 비교에 직접적으로 이용가능하다. 실험을 위하여 본 논문에서는 WANA[1]을 기본으로 활용하였다. 아래 <그림 5>는 graphViz[24]를 이용하여 작성한, 노드와 에지로 나타낸 GIMS의 구조이다.

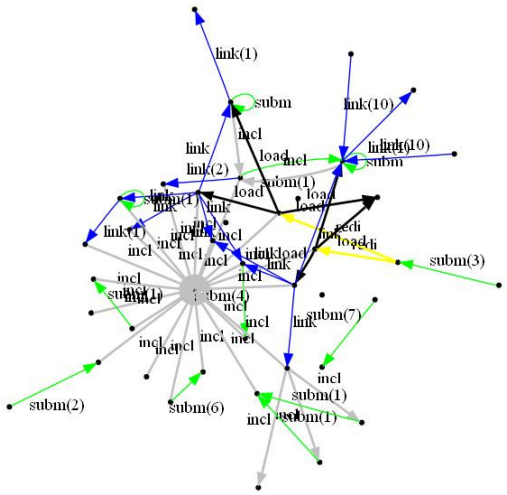

그림 5. GIMS의 구조
Fig. 5. The Structure of GIMS

본 논문에서는 sb-WCOX를 적용하기 위하여 두 가지 기준으로 유사도를 판단하였다. 하나는 웹 어플리케이션의 구조 유사도를 반영하고 있는 WSIM[19]으로, 이는 컨텐츠가 아니라 웹의 연결 구조를 기준으로 유사도를 판단하는 메트릭이다. 또 하나는 개발자들이 파일의 성격에 따라 Board, Login, Info, Main, Workreport의 다섯 그룹으로 나누어 분류한 것을 이용하였다. GIMS는 40개의 jsp 파일을 포함하고 있는데, Board에 9개, Login에 6개, Info에 6개, Main에 5개, Workereport에 10개의 파일이 속해있으며, 4개의 파일은 어느 그룹에도 속해있지 않다. 아래 <표 3>에서는 WSIM의 레벨2를 유사도1, 개발자의 분류를 이용한 것을 유사도2로 칭한다. 유사도 1의 경우 추출된 497개의 유사도 쌍들에 대해 약 상위 20%에 해당되는 99개의 추출된 쌍을 이용하였다. 예를 들어, 유사도1(WSIM의 레벨2)을 적용하였을 때, GIMS의 InfoMy.jsp와 유사한 파일들의 그룹은 {Join.inc, WorkreportAdmin, Join, WorkreportModify.inc, WorkreportModify, WorkreportMy}이다. 그리고 유사도 2, 즉 개발자들이 제시한 유사한 파일들의 그룹에서 InfoMy.jsp과 유사한 파일들은 {InfoAllAdmin, InfoDeleteAdmin, InfoModityAdmin, InfoModifyAdmin.inc, InfoMy.inc}이다. 편의상 확장자 .jsp는 생략하였다.

아래 <표 4>는 GIMS에 대하여 기존의 웹 엔트로피 기반 복잡도 WCOX [1], 그리고 제안된 sb-WCOX를 적용한 결과를 비교해서 보여준다.

표 4. WCOX와의 비교 (GIMS)
Table 4. Comparison with WCOX (GIMS)

	WCOX	sb-WCOX
유사도1	3.450	3.101
유사도2		2.891

그리고 두번째 실험으로 GIMS의 파일을 복제하여 같은 파일들로 구성된 두 폴더를 GIMS'로 두고 같은 실험을 하였다. 즉 파일 사이즈가 두 배로 커졌고, 유사한 파일들의 대상이 두 배로 확장된 경우 각 복잡도 값의 변화를 확인해본다. 해당 실험에서 유사도 1에 해당하는 WSIM적용결과 상위 2026개의 유사도 쌍에 대하여 약 13%에 해당하는 258개를 쌍을 대상으로 결과를 구하였다. 258개인 이유는 유사도 상위 177위에 해당하는 쌍의 개수가 81개이기 때문이다. 아래 <표 5>는 그 결과를 보여준다. 사이즈가 두 배가 됨에 따라 전체적인 복잡도는 모두 증가하였으나, 유사도를 반영한 경우 해당 유사도의 종류에 관계없이 기존의 WCOX보다 낮음을 알 수 있다.

표 5. WCOX와의 비교 (GIMS')
Table 5. Comparison with WCOX (GIMS')

	WCOX	sb-WCOX
유사도1	4.450	4.253
유사도2		3.690

결과적으로 sb-WCOX는, 유사도를 인지되는 복잡도에 반영하여, 적용한 유사도에 따라 각기 다른 복잡도를 제시해 준다.

어플리케이션의 다양한 분야가 존재하지만, 제안된 sb-WCOX는 특정 분야에 관계없이 재사용되는 비율이 높을수록 유용하게 활용될 수 있다. 기존의 개수 기반 복잡도나 순환 복잡도뿐 아니라 엔트로피 기반 복잡도는 유사성에 대한 반영을 하지 않아, 실제 유지보수시에 개발자들의 직접적인 이해도, 즉 인지적인 복잡도를 보여주기 어렵다. 물론 재사용되는 영역이 작을 경우에는 단순 엔트로피기반 복잡도와 차이가 적어지지만, 웹 어플리케이션의 경우 중복된 코드 조각의 비율이 다른 어플리케이션보다 높으므로 [25], 적용성이 있다고 할 수 있다.

www.kci.go.kr

V. 결론

본 논문에서는 실제 유지보수 시의 노력도나 이해도를 추정하기 위한 하나의 방법으로서 유사도와 엔트로피 이론을 이용하여 웹 어플리케이션의 복잡도를 정의하였다. 기존의 웹 복잡도는 대부분 개수 기반으로서 실제 이해도나 유지보수 노력도를 반영하기 어려웠다. 또한 기 정의된 엔트로피 기반 복잡도는 각 개별 페이지 자체의 정보량을 모두 동일하게 취급하여 실제 정보량을 반영하기 어려웠다. 본 연구에서는 이를 보완하기 위하여 유사도를 활용하여 실제 인지되는 복잡도를 정의하였다. 그리고 예시를 통하여 타 메트릭과의 차이점을 보였다.

본 논문에서 페이지라는 용어를 사용하고 있지만, 일반적으로는 웹의 다양한 컴포넌트들까지 포함하는 일반적인 의미로 생각할 수 있다. 특히, 오늘날의 웹 애플리케이션은 페이지 단위 뿐 아니라 프레임워크에서 사용가능한 다른 요소들을 사용하기 때문에 추상화 수준에서 확장이 가능해야 한다. sb-WCOX 복잡도는 구체적인 기술에 종속적이지 않고, 엔티티, 관계 및 파라미터에 기반한 추상적인 모델을 사용하고 있으며, 추상화된 웹 애플리케이션의 경우 본 연구가 가정한 확장된 그래프 구조를 벗어나지 않는다. 그러므로, 본 연구에서 사용하는 페이지는 프레임워크 기반의 웹 어플리케이션 요소들까지도 포함하는 일반적인 의미로 해석될 필요가 있다.

향후 연구로서, 유사도 적용을 단순 벡터에서 보다 정교한 방식으로 확장, 보완하도록 한다. 또한 각종 다양한 유사도 메트릭을 이용할 수 있도록 현재 구현되어 있는 도구 WANA [1]를 확장하여 다양한 시스템에 대해 다양한 유사도 메트릭으로 실험하여 본 연구의 유용성을 평가하도록 할 것이다.

참고문헌

[1] W. Jung, E. Lee, K. Kim, and C. Wu, "A Complexity Metric for Web Applications Based on the Entropy Theory," in Proc. of Asia-Pacific Software Engineering Conference, pp. 511-518, 2008.

[2] 김지현, 오성균, "웹 소프트웨어의 위험분석 모델에 대한 연구," 한국컴퓨터정보학회논문지, 제 11권, 제 3호, 281-289쪽, 2006년 7월.

[3] Y. Zhang, H. Zhu, and S. Greenwood, "Website Complexity Metrics for Measuring Navigability," in Proc. of International Conference on Quality Software, pp. 172-179, 2004.

[4] E. Mendes, N, Mosley and S. Counsell, "Comparison of Web Size Measures for Predicting Web Design and Authoring Effort," IEE Proceedings-Software, Vol. 149, No. 3, pp. 86-92, June 2002.

[5] C. E. Shannon, "A Mathematical Theory of Communications," Bell System Technical Journal, Vol. 27, pp. 379-423, July 1948.

[6] S. K. Abd-El-Hafiz, "Entropies as Measures of Software Information," in Proc. of IEEE Int'l Conference on Software Maintenance, pp. 110-117, 2001.

[7] E. B. Allen, "Measuring Graph Abstractions of Software: An Information-Theory Approach," in Proc. of IEEE Symposium on Software Metrics, pp.182-193, 2002.

[8] A. Bianchi, D. Caivano, F. Lanubile, and C.A.Visagio, "Evaluating Software Degradation through Entropy," in Proc. of Int'l Software Metrics Symposium, pp. 210-219, 2001.

[9] W. Harrison, "An Entropy-Based Measure of Software Complexity," IEEE Trans. on Software Engineering, Vol. 18, No. 11, pp. 1025-1029, 1992.

[10] J. Bansiya, C. Davis, and L. Etzkorn, "An Entropy-Based Complexity Measure for Object-Oriented Designs," Theory and Practice of Object Systems, Vol. 5, No. 2, pp. 111-118, Apr. 1999.

[11] K. Kim, Y. Shin, and C. Wu, "Complexity Measures for Object-Oriented Program Based on the Entropy," in Proc. of Asia-Pacific Software Engineering Conference, pp. 127-136, Nov. 1995.

[12] 오성균, 김미진, "웹 어플리케이션의 복잡도 예측에 관한 연구," 한국컴퓨터정보학회논문지, 제 9권, 제 3호, 27-34쪽, 2004년 9월.

[13] I. Ivan, A Felician and M. Popa, "The Impact of the Operations upon Complexity of Web Applications," in Proceedings of Romanian Symposium on Computer Science, pp. 55-64, 2006.

[14] C. Mao and Y. Lu, "A Method for Measuring the Structure Complexity of Web Application," Wuhan University Journal of Natural Science,

www.kci.go.kr

Vol. 11, No. 1, pp.143-150, 2006.
〔15〕 P. Chandra and G. Manjunath, "Navigational Complexity in Web Interactions," in Proceedings of World Wide Web, pp. 1075-1076, 2010.
〔16〕 이은영, 최병주, 송화정, 황상철, "프레임워크 웹 어플리케이션을 위한 BizUnit 테스트 코드 생성," 정보과학회 논문지:컴퓨팅의 실제 및 레터 제 15권 제 12호, 899-912쪽, 2009년 12월.
〔17〕 T. Laakso and J. Niemi, "An Evaluation of AJAX-enabled JAVA-based Web Application Frameworks," in Proceedings of International Conference on Advances in Mobile Computing and Multimedia, pp. 431-437, 2008.
〔18〕 G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, U. De Carlini, "Comprehending Web Applications by a Clustering Based Approach," in Proc. of the International Workshop on Program Comprehension, pp.261-270, 2002.
〔19〕 W. Jung, E. Lee, and C. Wu, "WSIM: Detecting Clone Pages based on 3-Levels of Similarity Clues," in Proc. of International Conference on Computer and Information Science, 2010 (accepted).
〔20〕 B. J. Lee, E. J. Lee, and C. S. Wu, "Genetic Algorithm Based Restructuring of Web Applications Using Web Page Relationships and Metrics," Lecture Notes in Computer Science, Springer-Verlag, Vol. 4113, pp.697-702, 2006.
〔21〕 E. J. Weyuker, "Evaluating Software Complexity Measures," IEEE Transations on Software Engineering, Vol. 14, No. 9, pp. 1357-1365, Sep. 1988.
〔22〕 J. Tian, "Complexity Measure Evaluation and Selection," IEEE Trans. on Software Engineering, Vol. 21, No. 8, pp. 641-650, Aug. 1995.
〔23〕 GIMS, *http://sourceforge.net/projects/gims2007/*
〔24〕 GraphViz, *http://www.graphviz.org/*
〔25〕 F. Calefato, F. Lanubile, and T. Mallardo, "Function Clone Detection in Web Applications: A Semiautomated Approach," Journal of Web Engineering, Vol. 3, No. 1, pp. 003-021, 2004.

저 자 소 개

정 우 성

1998-2002 : SK UBCare 연구원
2003 : 서울대학교 컴퓨터공학부(학사)
2003-2004 : 서울대학교 전기컴퓨터 공학부(석사과정)
2004-현재 : 서울대학교 전기컴퓨터 공학부(석사/박사통합과정)
관심분야 : 소프트웨어 진화, 소프트웨어 마이닝, 소프트웨어 아키텍처, 웹공학, 적응형 소프트웨어

이 은 주

2005 : 서울대학교 전기컴퓨터공학부 (공학박사)
2006-현재 : 경북대학교 IT대학 컴퓨터학부 조교수
관심분야 : 웹공학, 재공학, 메트릭, 소프트웨어 유지보수

www.kci.go.kr