An Entropy-Based Complexity Measure for Web Applications Using Structural Information*

WOO-SUNG JUNG1, EUN-JOO LEE2+, KAP-SU KIM3 AND CHI-SU WU1

1School of Computer Science and Engineering
Seoul National University
Seoul, 151-742 Korea

2School of Computer Science and Engineering
Kyungpook National University
Daegu, 702-701 Korea
E-mail: ejlee@knu.ac.kr

3Department of Computer Education
Seoul National University of Education
Seoul, 132-742 Korea

Web applications tend to get more change requests, evolve faster, and have shorter life-cycles than general applications. However, there are few systematic approaches dealing with their development and management, which results in a degrading structural quality and high maintenance costs. Complexity is a measure that is closely related to maintainability, understandability, and testing efforts. Most of the existing complexity measures for the web are count-based, and they provide a biased view of WAs. Therefore, it is necessary to define another complementary complexity notion. In this paper, entropy-based complexity measures, WCOXIN and WCOXOUT, are proposed for web applications, with a model which is defined using pages, relations, and parameter information. Entropy, which is connected to a system’s average information quantity, can be used as a metric of software aging and structural degradation. Several experiments are conducted to show the effectiveness of the proposed complexity measure, using WANA, a tool especially developed for the experiments. The experimental results reveal that the proposed WCOXIN and WCOXOUT measures effectively reflect the structural changes of web applications, which are not considered by count-based complexity measures.

Keywords: web applications, complexity, entropy, information, measure

1. INTRODUCTION

As the Web has grown rapidly and Web applications (WAs) have filled major roles in the software industry, the concept ‘web crisis’ has emerged from the prior ‘software crisis’ [1]. Web applications tend to get a relatively greater number of change requests [2] and their structures have changed from static and simple to dynamic and complex ones. To make matters worse, in developing and maintaining web applications, there is rarely any systematic process, and there is a lack of necessary artifacts such as analysis and design documents. Therefore, change requests are commonly handled in an ad-hoc manner and applications are developed without any analysis and design phase [3], which eventually results in degradation of the structure and quality of the target Web applications [2-4].

Received April 14, 2009; revised September 21, 2009 & April 30, 2010; accepted May 3, 2010.
Communicated by Chih-Ping Chu.

* This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (MEST) under Grant No. NRF-2007-331-D00407.

† Corresponding author.
As the structures of the Web applications become increasingly complicated, they become hard for developers and maintainers to understand and change, and more maintenance work is required. And at the same time, end users experience difficulties in getting the necessary information from the applications, or using them to perform their jobs.

Web applications are similar to other software in that they have business logic in application domains, however, there are several characteristics that differ from traditional softwares [5]: WAs have hypertext structure, dynamically generate codes, and rapid evolution is required [6-8]. For these reasons, it is hard to apply existing metrics to WAs, and new metrics for WAs should be defined. In existing maintenance approaches, structural systems or object-oriented systems are the main focus and Web applications are not often considered [7]. A complexity measure is a good metric of the degradation of a software [9], and it is closely related to maintainability, understandability, and testing efforts [10, 11]. Empirical results show that as the structural complexity of web applications increases, the maintenance costs increase [12].

Several studies have been conducted for a complexity measure, however, most studies have focused on the complexity of traditional software rather than the complexity of WAs. Zhang et al. proposed a navigational complexity measure for the web using a navigational structure and the number of links, from a user’s point-of-view [13]. Mendes et al. introduced a count-based complexity measure of web applications [14]. However, there are some cases where those count-based measure cannot handle well. Fig. 1 shows one of those cases.

![Graphs of the case of before and after CCA-refactoring.](image)

Figs. 1 (a) and (b) are a graph-based model for the same WA. A controller-centric refactoring (CCA refactoring) is applied to Fig. 1 (a), which results in Fig. 1 (b). The CCA refactoring is a kind of refactoring technique which is proposed by Ping and Kontogiannis [15]. Detailed explanation about CCA refactoring is provided in section 5. More information (for example the structure or architecture of WA) can be gleaned from Fig. 1 (b) than Fig. 1 (a). Based on the more intuitive structure of Fig. 1 (b), where most links are concentrated on the two controller pages, we can infer that the uncertainty is decreased via the information. This indicates that the Fig. 1 (b) is more understandable than Fig. 1 (a) and it also has a lower quantity of information, for average. However, the size and the McCabe’s Cyclomatic Complexity (CC) [16] are increased after CCA refactoring, though the architectural patterns are more ordered. In particular, Fig. 1 (a) has 20 entities and Fig. 1 (b) has 22 entities, where the two controllers are added to Fig. 1 (a), and they
have 40 static-relations in common. In Fig. 1 (a), the CC is 22 and in Fig. 1 (b) it is 24, that is, the CC increases after CCA-refactoring. (The size is also increased after CCA-refactoring.) This example shows that CC or other size-based measures scarcely detect a difference and another measure is required to manage this case.

Besides these count-based metrics, there have been other studies on entropy theory [17-19]. Entropy is used in various areas; in software engineering fields, entropy is applied to measure the cohesion and coupling of a modular system, to design a mathematical model for evaluating software quality, to define complexity measures, etc. [18, 20-22]. Software can be described as an information source containing information, thus, the entropy of a source is regarded as the average information quantity of the source [18]. Entropy-based metrics enable monitoring of a system’s aging and they are also applied to evaluate software degradation [9, 10]. Aging and degradation of a software are principal concepts in software maintenance, however, most studies using entropy have mainly focused on object-oriented systems [17, 19] or general modular software [18].

In our previous work, a complexity measure of web applications, WCOX was defined [23]. In [23], a web application is modeled as a graph composed of nodes and weighted edges; nodes and edges correspond to pages and relations between nodes, respectively. A reference probability for each page is defined with the weights of edges belonging to the page. Jung et al. assumed that the information quantity of a frequently referenced page is larger than that of an infrequently referenced page, when a maintainer reviews web pages statically. It is also assumed that the information quantity of every page is equivalent. That concept of information quantity is used to define WCOX in [23]. However, in [23], it has been failed to distinguish between the understandability and structural complexity. That is, when the target of WCOX is to consider the understandability, the total information quantity is required rather than the average information quantity. At the same time, when the target is to assess structural complexity, not only in-link but also out-link is needed, however, only in-link is considered in [23]. Another problem of WCOX is its handling of parameters. In the web model in WCOX [23], parameters are regarded as a factor that influences the relationship of weights; an edge that has many parameters tends to have a large weight. As the result, the WCOX value is biased and it is hard to distinguish between count-based metrics and entropy-based WCOX.

In this paper, the complexity of WA is defined using an entropy concept, and two kinds of complexity measures are introduced: WCOXIN and WCOXOUT which indicate the in-link complexity and out-link complexity, respectively. A web application model is semi-formally defined by complementing and adjusting the previous model. In addition, two kinds of experiments are newly designed and executed: a controller-centric architecture and an index-based structure. The former is concerned with WCOXIN, the latter with WCOXOUT. Both experiments are intended to show the applicability of the proposed measure, compared with the existing count-based complexity measure. In Fig. 1 (a), the WCOXIN is 3.94, and in Fig. 1 (b) the WCOXIN is 1.19, which shows that WCOXIN is sensitive to this situation, whereas CC or other size-based measures scarcely detect the difference.

The remaining parts of this paper are organized as follows. Section 2 describes works related to the concept of entropy theory and complexity metrics. Section 3 presents the model of web pages on which this research is based. Section 4 explains the proposed complexity metrics and includes verification of the proposed metrics using complexity
properties. In section 5, the results of several different experimental methodologies are presented, to show the applicability of our approach. Finally, section 6 outlines the summary, contributions, and limitations of this study.

2. RELATED WORKS

2.1 Entropy Theory

In thermodynamics, entropy represents disorder; it is regarded as a kind of information quantity in information theory by Shannon [24]. Information quantity implies ‘the degree of unknown’, that is, entropy is maximized when nothing is known about a given issue and entropy is minimized when everything is known about it. High entropy, equivalent to high disorder, means that a given issue has many unknowns.

Entropy is applicable to many different fields. In software engineering, entropy is used to assess the coupling and cohesion of modular systems, to design a mathematical model for quality evaluation, and to define complexity measures [18, 20-22]. Shannon’s entropy [24], which is the most frequently used measure, represents the uncertainty. The interpretation of entropy is two-fold [20]; the uncertainty and the information quantity of an information source. Shannon argued that when a signal conveys information, the greater the uncertainty in the signal, the more information it conveys. Assume that a message $M$ is composed of a symbol set \{$s_1$, $s_2$, ..., $s_n$\}. Harrison explained information as “the amount of surprise conveyed by each symbol in the message” [18]. That is, when $M$ conveys more unexpected symbols, $M$’s information quantity is larger. If the symbols are frequently transmitted and they become familiar to a receiver of the message, the information quantity of $M$ is small. The frequency of a symbol is the occurrence probability.

In Shannon’s entropy theory, a message $s_i$’s information quantity $I(s_i)$ is defined as follows:

\[ I(s_i) = -\log_2 p_i \]

where

- message $m_i = (s_1, s_2, ..., s_n)$, $s_i$ is a symbol.
- $p_i$ is $s_i$’s occurrence probability.

As information is additive [18], the total information content of $m_i$, $H(m_i)$, can be derived from the information quantity of each symbol, by using the following equation:

\[ H(m_i) = -\sum_{i=1}^{n} p_i \cdot I(s_i) = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i. \]

Software design can be considered to have information content, because software designers make several design decisions based on the information content in the design. This indicates that entropy-based metrics are more meaningful than count-based metrics [25]. For example, the information quantity of modules can be used to determine the
manpower demand for each module, in maintenance [20]. Bianchi et al. define four entropy-based metrics, to show that software degradation is related to entropy [10]. They demonstrate that as the number of software links increases, the entropy values also increase. They also insist that a software engineer can monitor software aging using their metrics.

2.2 Complexity Measures

In this subsection, entropy-based complexity and web complexity measures are introduced. Several entropy-based complexity measures have been proposed [17-19, 25]. [17, 19] define a complexity measure for object-oriented software. The main difference between [17, 19] is in calculation of the occurrence probability. In [17], the reference probability, which has a similar meaning to the occurrence probability, is based on the relationship between nodes. That is, if there are many links to/from a node, the reference probability of the node is larger than a node that has few links. In [19], the name occurrence is used, that is, the frequency of the name in the code. Allen has defined a few metrics using entropy and hyper-graphs for general software [25]. Harrison defined a reference probability as the ratio of the frequency of each operator to the total frequency of all operators [18]. Using this reference probability, AICC, an entropy-based complexity measure was defined [18]. Harrison argued that AICC can only be used as an ordinal measure, because the results of addition and subtraction of AICC values are meaningless. That is, the difference between the two AICC measures has no meaning [18].

Although there have been several studies on software complexity measures, there have only been a few studies on web application complexity. However, quality evaluation methodologies have been proposed using various metrics including complexity [14, 26-29]. Olsina and Rossi presented a quality evaluation process for WAs, WebQEM, by utilizing product features, such as navigation, interface, reliability, and so on [29]. In WebQEM, the product features are based on users, not developers, and moreover, the concrete metrics are not mentioned in [29]. Mendes et al. proposed several metrics suitable for static web applications; connectivity, connectivity density, total page complexity, cyclomatic complexity [14] and structure [26]. Connectivity is defined as the total sum of internal links, and connectivity density can be computed by dividing the total number of pages into the connectivity. Total page complexity is a page’s average number of media types. Cyclomatic complexity is measured on the WA graph using McCabe’s CC. Finally, structure complexity tells the main structure of a WA, such as sequence, hierarchy, and network, but the way to measure the structure is not provided in [26]. All of those metrics can be applied to only static pages, though. Germonprez and Zigurs classified the complexity of web sites into three dimensions: cognition, content, and form. They suggested the complexity evaluation framework with these three dimensions [27]. The structural complexity, which is the target of our work, belongs to the form dimension. In [27], the web site structure is closely related with similar formatting between pages and consistent interfaces that manages similar tasks [30]. The target of the framework in [27] is mainly static pages, therefore, it is hard to incorporate it on current WAs. Marchetto suggested a metric suite to evaluate the quality of WAs [28]. In Marchetto’s metric suite, the complexity and the size are used together to measure testability, fault tolerance, and error proneness. In [28], the size/complexity metrics include system size, line of code, number
of attributes, weighted operations per component, reuse ratio, and specialization ratio, however, most of these are size-based. In particular, the weighted operations per component, which indicates a module’s complexity, is defined as the sum of individual operations’ complexity in the module. An operation’s complexity is based on the number of parameters in the operation.

Zhang et al. have defined the structural complexity of a web site [13]; WSC1, WSC2, WSC3, WSC4, WSC5. For convenience, the following terms refer to Zhang’s complexity. The measures are represented by these equations:

\[
WSC_1 = \sum_{i=1}^{n} \text{outlink}(i)
\]

\[
WSC_2 = \frac{WSC_1}{n}
\]

\[
WSC_3 = e - n + d + 1
\]

\[
WSC_4 = \frac{WSC_3}{n}
\]

\[
WSC_5 = \frac{\sum_{i=1}^{n} \text{outlink}(i)^2}{n}
\]

where

- \(n\): the number of nodes (pages)
- \(e\): the number of edges (hyperlinks)
- \(\text{outlink}(i)\): the number of outlinks of node \(i\)
- \(d\): the number of dead-end nodes (unreachable nodes)

These measures differ from the approach proposed in this paper, in that they are defined to assess the navigability of a web site rather than a web application, and only hyperlinks are considered in the relationship between nodes.

Ivan et al. defined the structural complexity indicator for WAs using McCabe’s Cyclomatic Complexity as the ratio of current structure to complete structure [31]. For example, the WA graph is complete binary tree, and then, the indicator value is one. However, the meaning of the complexity indicator is ambiguous in [31]. For example, the indicator value does not imply the number of pages or the structural types. Mao and Lu proposed the web application’s structural complexity for intra/inter level [32]. The intra-level complexity is defined with the control flow and data flow, in a page or a component. The inter-level complexity is measured using the navigation and data interaction among web entities, such as pages, components, and database. The purpose of inter-page complexity is similar to that of WCOX in this paper, which is to measure a WA’s structural complexity. For a WA, the WNG (Weighted Navigation Graph) is generated. The weight in WNG is computed using the static/dynamic page type and the amount of data in dynamic pages, that is, the number of parameters. The inter-page complexity is average weight, which is the result of dividing the total summation of weights by the number of all nodes in a WA. It is similar to our work in that a WA is modeled into a weighted graph using the relation type and the number of parameters, but the target in [32] includes not only pages, but also database and components. Furthermore, the complexity is also obtained in count-based way.

As mentioned, most of the existing studies on web application complexity are count-based, and little is known about the entropy-based web complexity.
3. WEB APPLICATION MODEL

The WA model is established via static analysis for web applications. In this section, some considerations about the model are explained and the model is described with sets and tuples.

3.1 Considerations

The Web application model consists of nodes and their associated weighted edges. Nodes are entities such as static and dynamic pages, resources such as images and moving pictures, CSS components, JavaScripts, other objects used to execute a given business logic, etc. In this paper, we focus on static pages, because the purpose of the complexity measure proposed is to reflect the maintainability for physically existing pages, in the structural view. That is, the passed parameters for the edges and the relationships between pages are regarded as the main factors which influence the structural complexity. For these reasons, the physically existing pages that have `.html`, `.htm`, `.jsp`, `.asp`, etc. as their filename extensions are considered, and the resulting pages that are dynamically generated are not considered in this paper. The links between pages correspond to edges. Weights of edges are determined by the type of links and the number of parameters and they are used to get the reference probability of each page. The types of relationships between pages are classified by some studies as follows [2, 8]: link, submit, redirect, include, build, and load. Two pages have a link relationship when they are related to a hyperlink. A Submit relationship is related to a form. A Redirect relationship occurs when the current page is automatically moved to another page. When the from-page contains the to-page, the pages are considered to have an include relationship. A Load relationship is related to the frame structure. A Build relationship, which occurs when a server page physically generates a resulting page, is excluded, as mentioned. This is because the targets of maintainers, which are the focus of this paper, are mainly physically existing pages.

Lucca et al. set the weight for each relation based on their empirical data and got the couplings between web pages using the weights [4]. According to them, the weight values in descending order were: submit, redirect and link. Based on [4], Lee et al. used a similar approach to [4] as follows: Submit is the highest-weighted relation, redirect is the next highest, and link, include, and load have the smallest weights. In this paper, a relation weight is used to define a reference probability of each page. A reference probability of a page indicates how often the page is referenced. Usually, a page that has more relations than others has a high probability that it will be referenced. Therefore, the weighting scheme should be distinct from existing studies [2, 4] which focus on the structural aspects of a web application. In this paper, two types of linking form are considered: an automatically linked type and a selectively linked type. The former includes redirect, include, and load relationships, which do not require any user intervention. The latter includes link and submit relationships which are activated by a user selection. In conclusion, the weight of the automatically linked case is set to be higher than that of the selectively linked case, because automatically linked pages are always referenced. The number of parameters for a link between two nodes is also an important factor in obtaining the weight of the link. In a previous work [23], the greater the number of link parameters,
the higher the weight of the link. That is, in previous work [23], the weights were defined to be proportional to the number of weights but this is not strictly correct from the viewpoint of the information quantity.

In this web model, the web pages correspond to a message’s symbols in the entropy model. A page that has many relations with others has a higher chance to be accessed by others than a page that has few relations, therefore, more people are familiar with the former page. In other words, the “degree of surprise” [18] of the multi-related page is not high. The “degree of surprise” implies a kind of information quantity. In brief, the number of relations that a page has is inversely proportional to the page’s information quantity. In case of parameters, a page linked with many parameters is more difficult to understand or maintain that page than a page with few parameters [33], which indicates that the page with more parameters is more informative. As a result, we infer that the number of parameters is proportional to the information quantity. In the proposed model, the weight-sum for each page results in a decrease in the information quantity of the page, because a high weight-sum of a page indicates a high reference probability of the page, which results in a low-information quantity of that page. Therefore, the number of parameters should be inversely proportional to the weight, rather than proportional as in [23]. In this paper, the number of parameters for a link is applied to decrease the weight of the link.

It can be argued that only pages and their relations are considered to get WCOX in this paper. Generally, WAs may contain back-end components such as DLLs, EJBs or web services. However, these web components are not considered for WCOX, because they are treated as black-boxes which have been compiled already and maintained independently. WCOX is a complexity whose purpose is to measure the structural tendency of WAs, which is based on the relations among pages, and to give helpful information to developers, for example, information for refactoring the WA. In addition, web elements such as CSS styles, script codes, etc., are also ignored in our work for the same reason.

3.2 Model

The web model proposed in this work is a graph composed of nodes and edges that are weighted and directed.

Definition 1  UID: The universe set that contains unique identifiers.

Definition 2  The web application model: WA.

\[ WA = \langle N, E \rangle; \text{ WA is defined with node set } N \text{ and edge set } E. \]

Definition 3  N: The node set.

\[ N \text{ is the set of nodes such as static pages and dynamic pages which constitute the WA. } n \ (n \in N) \text{ has a unique identifier nID (nID} \in \text{UID)}. \]

Definition 4  E: The edge set.

\[ E \text{ is the set of edges. } e \ (e \in E) \text{ is defined with tuples as follows,} \]

\[ e = \langle e\text{ID, from, to, REL}_\text{SET}e \rangle \]

\[ e\text{ID is a unique identifier for } e. \]
− from, to ∈ N, from is e’s start node and to is e’s end node.
− REL_SETe is a set that contains several data elements related to e.
− REL_SETe = {<rel_id, type, npar>}
  − rel_id is a unique ID for one of the relations belonging to e. There can exist
    one or more relations from ‘from page’ to ‘to page’.
  − rel_id ∈ UID
  − type is rel_id’s type.
  − type ∈ REL_TYPE
    − REL_TYPE = {link, submit, load, redirect, include, (others)}
  − npar is the number of parameters that rel_id has. (npar ≥ 0)
  − getType(rel_id) ∈ REL_TYPE is a function that returns the type of rel_id.
  − getNparam(rel_id) is a function that returns the number of parameters, npar.
  − isRelated(a, b) (a, b ∈ N) is a function that returns true when a and b are related.
    If a and b have no links between them, isRelated(a, b) returns false.
  − getID(a, b) (a, b ∈ N) ⊂ (UID ∪ φ) is a function that returns the id-set of links
    from ‘a’ to ‘b’ when isRelated(a, b) is true. If isRelated(a, b) is false, getID(a, b)
    returns φ. As stated, there can be one or more links from a to b.

4. WEB APPLICATION COMPLEXITY

According to the link direction, two complexity metrics, WCOX_IN and WCOX_OUT,
are defined for in-link and out-link, respectively.

4.1 WCOX Definition

General structural metrics are quantified measures of interactions between modules
[34], and a web application’s structural metrics are used to assess the interactions be-
tween pages in a WA. ‘Interaction’ means linking and parameter passing between pages,
and the structural complexity is a metric that shows the complexity of these interactions.
In prior work [23], only in-links are considered for the definition of complexity, because
it is assumed that maintainers review web pages based on in-links. However, this is not
the case in structural complexity. The structural complexity of web applications incor-
porates the linking structure of the web application model represented as a graph. In the
prior approach, the notion of the average information quantity was applied, but the total
information quantity is applicable to the prior WCOX measure, not the average informa-
tion quantity. This is because the target of the prior WCOX measure is not strictly to
measure the structural complexity. The purpose of the prior WCOX measure [23] was to
assess the understandability via the information quantity per page, therefore, the summation
of each page is more suitable for this than averaging.

In this work, not only in-links but also out-links are considered. In existing studies
that proposed an entropy-based complexity measure of object-oriented systems [17, 19],
the directions of nodes are not considered. For web complexity, Mendes et al. suggested
a complexity metric based only on the number of links [14], and Zhang et al. mainly con-
siders out-links to assess navigability [13]. In-links implies accessibility to a page, and a
page that has many in-links has a high probability that it will be accessed or reused by
others, so in-links are an unsuitable metric of navigability [13]. The purpose of this work is to measure a structural complexity measure for developers and maintainers, which improves understandability and maintainability, rather than navigability for end-users. For that reason, both in-links and out-links should be considered. Moreover, the structural disorder cannot be effectively captured without consideration of directions. For example, Figs. 2 (a)-(c) are regarded as equivalent graphs when the directions are ignored. However, it is clear that Figs. 2 (a) and (c) are ordered and Fig. 2 (b) is not. The structural characteristics of Figs. 2 (a) and (c) are revealed by their directions.

The necessary definitions for the complexity of a web application are as follows,

**Definition 5**  
\[ W(a, b): \text{Weight value between node } a \text{ and } b. \]

\[ W(a, b) = \begin{cases} 0 & \text{if getID}(a, b) = \phi \\ \sum_{r \in \text{getID}(a, b)} \text{weight}(rID) \cdot \left\{ \log_k \left( \text{getParam}(rID) + k \right) \right\}, & \text{otherwise} \end{cases} \]

where

\[ \text{weight}(id) = \begin{cases} w_1, & \text{if getType}(id) \in \{ \text{link, submit} \} \\ w_2, & \text{else if getType}(id) \in \{ \text{load, redirect, include} \} \\ 0, & \text{else} \end{cases} \]

\[ w_1 + w_2 = 1.0, \quad w_1 < w_2 \]

\( k \) is a natural number other than one, which determines the degree of influence of parameters. A high \( k \) value results in a low influence ratio of parameters. The weight of the relationship is classified into two types, as stated in section 3: automatically linking case and selectively linking case; the former has a higher value than the latter.

The web complexity for in-links and out-links is defined as follows,

**Definition 6**  
\[ \text{WCOX}_{IN}(WA) \]

\[ \text{WCOX}_{IN}(WA) = \sum_{i \in N} P(i) \cdot I(i), \]

where

\[ I(i) = -\log_2 P_{IN}(i), \]
\[ P_{\text{IN}}(n) = \frac{\sum_{i \in N} W(i, n)}{\sum_{i \in N, j \in N} W(i, j)} \]

**Definition 7** \( \text{WCOX}_{\text{OUT}}(WA) \)

\[ \text{WCOX}_{\text{OUT}}(WA) = \sum_{i \in N} P(i) J(i), \]

where

\[ J(i) = -\log_2 P_{\text{OUT}}(i), \]

\[ P_{\text{OUT}}(n) = \frac{\sum_{i \in N} W(n, i)}{\sum_{i \in N, j \in N} W(i, j)}. \]

For each direction, the reference probability \( P_{\text{IN}}(n) \) and \( P_{\text{OUT}}(n) \) for the node \( n \) are defined. They are calculated by dividing the total weight-sum in WA into the weight-sum of the in-links or out-links of \( n \). Each reference probability of a node is used to evaluate the information quantity of the node, and to finally obtain the web complexity for the in and out directions. \( J(i) \) is the information quantity of node \( i \) which is based on the concept of the self-information quantity explained in subsection 2.1. Finally, \( \text{WCOX}_{\text{IN}}(WA) \) and \( \text{WCOX}_{\text{OUT}}(WA) \) are obtained using the information quantity per page and the reference probability of the page, as given by Definitions 6 and 7, which means the average information quantity of WA, corresponding to the Shannon’s entropy [24].

In Harrison’s approach [18], the complexity of a modular system, AICC, has been suggested as the average information quantity of operators in a program, where the reference probability of an operator is defined by the occurrence frequency per operator. Harrison argued that the complexity value was not important, focusing on a relative ranking according to the value. In case of the WCOX measure in this paper, a similar approach to AICC is applied. That is, it is not implied that \( P \) is more complex by 0.5 than \( Q \) when \( \text{WCOX}(P) = 3.5 \) and \( \text{WCOX}(Q) = 4.0 \). This proposed WCOX measure can be applied to extract the upper \( k \% \) of clusters with a high complexity, and used for sorting web applications ordered according to the complexity. In particular, WCOX can clearly reveal differences between web applications with a similar size-based complexity, as shown in the experimental results.

### 4.2 Validation

The proposed WCOX measure should be validated according to given complexity criteria. Several properties that could form the basis of a complexity measure have been suggested [35, 36]. The WCOX measure in this paper is defined with a cognitive information quantity, thus Briand’s framework [36] it is not applicable. Briand et al. regards the complexity of a module as an intrinsic attribute, rather than the subjective complexity
perceived by an observer. Weyuker’s complexity axioms have been widely used to validate complexity measures, though they have attracted some criticisms [13, 18, 37]. In this paper, Weyuker’s axioms are also used to validate the defined WCOX measure. Before validation, some concepts in Weyuker’s axioms must be adapted for web applications [13]. These concepts involve the functional equivalence of two modules, composition of modules, permutation of statements order, and renaming modules, which are partly adjusted using Zhang et al.’s adaptation [13], in this work. Zhang et al.’s adaptations cannot be used without modification, because web applications are more focused on business logic in servers than web sites [5].

These adaptations are shown in Definitions 8-11. Assume that \( P \) and \( Q \) are web applications: \( P = \langle N_1, E_1 \rangle \) and \( Q = \langle N_2, E_2 \rangle \).

**Definition 8**  Functional equivalence of WA.

In [35], two modules are said to be functionally equivalent when the outputs of the two modules are identical, even though the modules may have different implementations. For web sites, Zhang et al. defined the functional equivalence of two web sites as nodes, and the start nodes in the two web sites are equivalent [13]. In the case of web applications, not only the node and the start node, but also the application logic of the target WAs must be equivalent. This is because the web site and web application differ in that a web application includes business logic [5].

**Definition 9**  Composition of WA: \( P; Q \).

In terms of the composition of the two modules, Weyuker did not add any relations to the two modules [35]. In this paper, the composition of the two WAs is defined as the union of the nodes and the links for the target WAs, though Zhang et al. did add one or more links between \( P \) and \( Q \). ‘\( P; Q \)’ is denoted as \( \langle N_1 \cup N_2, E_1 \cup E_2 \rangle \).

**Definition 10**  Permutation of the orders in WA.

The permutation of the orders in a WA is defined as a direction change of some edges in WA, without adding any edges.

**Definition 11**  Renaming of WA.

Renaming of a web application \( P \) means that every ID in \( P \) is renamed as a new unique ID.

Using these additional definitions, the proposed WCOX\(_{IN}\) is shown to satisfy the nine axioms of a complexity measure [35]. The proof of WCOX\(_{OUT}\) is omitted in this paper, because the approach used for WCOX\(_{IN}\) can be applied to WCOX\(_{OUT}\) in a similar manner.

**Property 1**  \((\exists P)(\exists Q)(|P| \neq |Q|)\)

There exist two web applications \( P \) and \( Q \) whose WCOX\(_{IN}\) values differ. Though the numbers of edges and nodes that consist of \( P \) and \( Q \) are equal, the WCOX\(_{IN}(P)\) and WCOX\(_{IN}(Q)\) can differ according to their reference probabilities.

**Property 2**  \( c = WCOX_{IN}(P_1) = WCOX_{IN}(P_2) = \ldots = WCOX_{IN}(P_n) \), where \( n \) is a finite number.

There exist finite numbers of WAs with the same WCOX\(_{IN}\). This is a property asso-
associated with the granularity. When \( n \) is one, the granularity of the complexity is excessively fine. On the other hand, when \( n \) is infinite, the granularity is over-coarse. Assume that the WA \( P \) has \( c \) as its WCOX and \( P \) consists of nodes and edges as \( \langle N, E \rangle \). When \( E_n \), which is equivalent to the edge set \( E_n \), is added to the existing edge set \( E_n \), WCOX must be \( c \), because the reference probability per node is fixed. As there can be an infinite number of \( E_n \), WCOX does not satisfy Property 2, which is a distinguishing feature between an entropy-based measure and a count-based measure. In practice, there are infinite cases such that two nodes have the same edge weight.

\[ (\exists P)(\exists Q) \quad (P \neq Q \land |P| = |Q|) \]

Property 3

There exists a case that two WAs \( P \) and \( Q \) provide different functionalities but they have an equal WCOX value. For example, \( P \) and \( Q \) provide search functionality and \( P \)'s target is product \( x \) and \( Q \)'s target is product \( y \), while the structures of \( P \) and \( Q \) are identical. In this example, WCOX(P) and WCOX(Q) are equal, but their functionalities are distinct.

\[ (\exists P)(\exists Q) \quad (P = Q \land |P| \neq |Q|) \]

Property 4

WCOX(P) and WCOX(Q) are distinct, though \( P \) and \( Q \) provide the same functionalities. For example, \( P \) and \( Q \) are WAs that search and return product \( z \), but the searching forms, parameters, etc. in \( P \) and \( Q \) may differ according to their implementation. WCOX(P) and WCOX(Q) then differ accordingly.

\[ (\forall P)(\forall Q) \quad (|P| \leq |P; Q| \land |Q| \leq |P; Q|) \]

Property 5

This is the property of monotonicity. It means that WCOX(P; Q) is larger than a given WCOX(P) or WCOX(Q), where the merged WA composed of \( P \) and \( Q \) is denoted as \( P; Q \). The number of nodes and edges increases when \( P \) and \( Q \) are merged, however, WCOX(P; Q) does not always increase. For instance, in Fig. 3, WCOX(P) = 2 and WCOX(Q) = 0, but WCOX(P; Q) = 1.870, where WCOX(P) is larger than WCOX(P; Q). Therefore, WCOX does not satisfy Property 5.

\[ (\exists P)(\exists Q)(\exists R) \quad (|P| = |Q| \land |R| \neq |P; Q; R|) \]

Property 6.a

When a WA \( R \) is added to \( P \) and \( Q \) with the same WCOX, WCOX(P; R) and WCOX(Q; R) may differ. This is why the addition of \( R \) may result in appending new relations to the existing \( P \) or \( Q \). The newly added relations to \( P \) and \( Q \) may not be equivalent, and, the newly added parameters may differ, so the merged WA \( P; R \) and \( Q; R \) can have a different WCOX.

\[ (\exists P)(\exists Q)(\exists R) \quad (|P| = |Q| \land |P; R| \neq |Q; R|) \]

Property 6.b

Properties 6.a and 6.b differ only in the appended position. That is, Property 6.a is the case that \( R \) is appended to the end, and Property 6.b is the case that \( R \) is appended to the front. In WA, where \( R \) is appended is irrelevant, so Property 6.b allows an explanation of Property 6.a.

\[ (\forall P) \quad |P| \neq |Q| \]

Property 7

If \( Q \) is a permutation of the orders in \( P \), then \( |P| \neq |Q| \).

When the directions of edges in \( P \) are changed, the reference probabilities per node are changed, which changes WCOX of the permuted WA \( Q \).
Property 8 If $P$ is the renaming of $Q$, then $|P| = |Q|$. This is a property that no significance given to names used in a program. WCOX$_{IN}$ is not concerned with the names of nodes, links, and parameters; therefore, WCOX$_{IN}$ satisfies Property 8.

Property 9 $(\exists P)(\exists Q) (|P|; |Q| < |P; Q|)$ It is possible that there exist $P$ and $Q$ where WCOX$_{IN}$ of the merged WA $P; Q$ may exceed the sum of WCOX$_{IN}(P)$ and WCOX$_{IN}(Q)$. This means that when two interacting WAs are merged, the merged WA might have additional complexity, due to their interactions. Assume that $P$ and $Q$ are composed of three nodes, and the reference probability for each node is 0.1, 0.1, and 0.8, respectively. WCOX$_{IN}(P)$ is equal to WCOX$_{IN}(Q)$ as 0.922, and the sum of WCOX$_{IN}(P)$ and WCOX$_{IN}(Q)$ is 1.844. After every reference probability of nodes in the merged $P; Q$ has been made equal by the $P$ and $Q$ relations, WCOX$_{IN}(P; Q)$ is 2.586. This is the case that the complexity of the merged WAs is larger than the cumulative complexity of WAs.

In conclusion, the proposed WCOX$_{IN}$ satisfies Weyuker’s complexity property, except for Properties 2 and 5. Cyclomatic complexity does not satisfy Properties 2, 6, 7, and 9, and the statement count does not satisfy Properties 6, 7, and 8 [35]. The entropy-based, object-oriented complexity by Kim et al. [17] does not satisfy Properties 5 and 7. The complexity metrics proposed in [38] and [39] do not satisfy Properties 7 and 9. The entropy-based complexity measure by Harrison [18] does not satisfy Properties 2, 5, 7, and 9. Zhang et al.’s five types of complexity for web sites [13] do not satisfy the average of Weyuker’s three properties. That is, it is not a necessary condition for complexity metrics to satisfy Weyuker’s nine complexity axioms. The applicability of each of Weyuker’s axioms also depends on the characteristics of the considered metrics. In addition, there is a difference between Weyuker’s target system and WA in this paper. Finally, WCOX$_{IN}$ and WCOX$_{OUT}$ can be considered sufficient to measure a web application’s complexity.

5. EXPERIMENTAL RESULTS

To show the effectiveness in terms of the two different viewpoints of WCOX$_{IN}$ and WCOX$_{OUT}$, experiments were conducted based on the WANA which is an upgrade version of the tool in [23]. Fig. 4 represents the activities and the necessary elements in the Overall processes of WANA used to calculate the complexity metrics. The first step of calculating the metrics is pre-formatting the web pages – eliminating unnecessary code fragments and special characters and so on – in order to detect the fundamental elements easily, which is performed by comparing code patterns. And then, web elements such as
entities, relations and parameters are extracted based on a given WA profile, jsp in this case. Next, the tokenized elements are inserted into a database which follows the schema of WANA’s WA model. Once a WA model for a target web application is constructed, various kinds of analysis can be performed by querying the model. Finally, the results are exported to xls or dot files which can be processed by the external tools.

The main functionalities of WANA consist of three parts: First, it not only calculates the complexity metrics by analyzing the given web applications, but also shows fundamental information such as the number of pages, relations and parameters as shown in Fig. 5. Other analysis results or metrics could easily be added to WANA if they are derived from the same abstract web application model. Second, it provides maintenance simulation for web applications based on a random approach. WANA can generate virtual web
application models based on the various configuration parameters such as the number of pages, the number of relations, the probability of each relation types, a range of form parameter ranges, etc. And then, WANA simulates the maintenance process by adding, deleting and modifying web pages based on the given probability configurations for the given number of phases. Finally, it supports two types of model refactoring: CCA (Controller-Centric Architecture) refactoring, which is described in detail in subsection 5.1 and IBS (Index-Based Structure) refactoring, which is described in detail in subsection 5.2. WANA also have other fundamental functionalities such as visualizing reverse-engineered web graphs with GraphViz [40], finding clone pages based on the structural similarities, extending web model queries, and so on.

Fig. 5 presents the screenshot of WANA. The area to the top-left shows the structure of a target WA, and each page in the area can be selected. The top-right area represents the analysis results of the selected page. The bottom area, which is divided into three sections, shows information of the WA and the selected page, such as the summary of the target WA, the summary of the selected page, and the code of the selected page, ordered from left-to-right.

5.1 WCOX\textsubscript{IN}: Controller-Centric Architecture

Ping and Kontogiannis suggested a methodology for transforming web sites into a controller-centric architecture, denoted as CCA for convenience, based on MVC (Model-View-Controller) [15]. This is a kind of refactoring technique, where a controller page is additionally implemented for the top-layer of the existing web application, to get and manage all requests from clients. After CCA refactoring, several controllers are added to the given web application. The controllers manage processing of a client’s requests and forwarding web pages, on centralized ways. In conclusion, the management strategy is simplified by constructing entry points, that is, controllers, which intercept HTTP requests. In that paper, the relationships between pages are analyzed, navigation flows are modeled, and then the target architecture is established. They argued that their methodology enables simpler and easier management of web pages.

![Fig. 6. WCOX\textsubscript{IN} of before and after CCA-refactoring.](image)

The horizontal axis in Fig. 6 is the number of entities, and the vertical axis is the value of WCOX\textsubscript{IN}. The number of relations is twice that of the number of entities for the simulation. Fig. 6 shows that as the number of entities or relations increases, the value of WCOX\textsubscript{IN} also increases. However, WCOX\textsubscript{IN} of the case of after-CCA-refactoring is relatively small, irrespective of the size.

To compare between the numbers of pages, links and WCOX\textsubscript{IN}, 100 WAs were ran-
domly created, which had 100 entities and 200 relations, denoted as the base state for convenience. To simulate real-world management situations, the base state is evolved via given operations based on a probability, and finally it is restructured according to CCA-refactoring. Operations for evolution are classified into three types: page evolution, relation evolution, and no evolution, which are selected according to a given probability. Page evolution and relation evolution have *insert* and *delete* operations, which are applied to the target web application model. Table 1 shows the experimental parameters.

Table 1. Experimental parameters.

<table>
<thead>
<tr>
<th>Generation</th>
<th>Evolution</th>
<th>Refactoring</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Pages</td>
<td>100</td>
<td># of Phase</td>
</tr>
<tr>
<td># of Rel.</td>
<td>200</td>
<td># of Step</td>
</tr>
<tr>
<td>Submit Rel.</td>
<td>0.2</td>
<td>Delete islo. Nodes</td>
</tr>
<tr>
<td># of Param.</td>
<td>1~5</td>
<td>Page Evolution</td>
</tr>
<tr>
<td>Include Rel.</td>
<td>0.05</td>
<td>- Insert</td>
</tr>
<tr>
<td>Link Rel.</td>
<td>0.55</td>
<td>- Delete</td>
</tr>
<tr>
<td>Redirection Rel.</td>
<td>0.15</td>
<td>Rel. Evolution</td>
</tr>
<tr>
<td>Load Rel.</td>
<td>0.05</td>
<td>- Insert</td>
</tr>
<tr>
<td>No Rel.</td>
<td>0</td>
<td>- Delete</td>
</tr>
<tr>
<td>No Evolution</td>
<td>0.1</td>
<td>CCA</td>
</tr>
<tr>
<td></td>
<td></td>
<td># of Controllers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refactoring Ratio</td>
</tr>
</tbody>
</table>

Fig. 7. Simulation results of real-world management situations (WCOXIN).

Fig. 7 illustrates the number of entities, the number of relations, the value of WCOXIN, and the $\log_2(x)$ which indicates the maximum value of WCOXIN where $x$ is the number of entities for the given WA’s size.

During the evolutionary process, that is, steps 2 to 11, the WCOXIN values are gradually increased, however, the WCOXIN values of the case of after-CCA-refactoring sharply decrease from 7.50 at step 11 to 2.27 at step 12. However, there is no significant change in the number of entities or the number of relations during refactoring, as shown in Fig. 8.

It is hard for count-based or size-based measures to incorporate these situations: the number of entities and relations only increases in Fig. 7. WCOXIN is very sensitive to structural changes such as CCA-refactoring. This implies that in practice WCOXIN may be used as a bad smell detector for specific design patterns, such as MVC.
5.2 WCOX\textsubscript{OUT}: Index-Based Structure

As WCOX\textsubscript{IN} is based on the in-links, it effectively shows the resulting changes of architecture or structural patterns such as CCA-refactoring from the developer’s view. Incidentally, there can be a structural improvement from a user’s point of view. For example, an index page or a sitemap page can be added to classify and arrange disordered web pages. In this paper, a static case is considered; a sitemap page is a typical example that shows the effectiveness of WCOX\textsubscript{OUT}. A sitemap page has multiple out-links to other pages, and it generally has a higher out-weight value than other pages. When a sitemap page is added, a large quantity of out-links are added from the sitemap to others. As a result, the entropy (WCOX\textsubscript{OUT}) is reduced, due to the asymmetric structure.

Fig. 8 shows the resulting WCOX\textsubscript{OUT} of the cases of before and after IBS-refactoring. The experiments mentioned in this subsection also use the same values as those shown in Table 1 of subsection 5.1, except that the refactoring type is IBS and the number of sub-index pages is three. IBS-refactoring denotes the refactoring case that a system is refactored into an index-based structure. The x-axis represents the number of entities and the y-axis denotes the average WCOX\textsubscript{OUT} for 10 simulated WAs with twice as many entities as links. In general, WCOX\textsubscript{OUT} tends to decrease after IBS-refactoring. However, when IBS-refactoring is applied to a small-scale WA, WCOX\textsubscript{OUT} is slightly increased. This is because the addition of a sitemap page may result in some confusion in case of a small WA. In other words, WCOX\textsubscript{OUT} is applicable when additional index pages do not contribute significantly to the overall scale of a WA.

Fig. 9 shows the changes of various measures during evolution. In this experiment, the conditions are largely similar to those shown in Fig. 7 in subsection 5.1, except that
WCOX\textsubscript{OUT} is used rather than WCOX\textsubscript{IN} in Fig. 7. Evolution is conducted from steps 2 to 11 such as in Fig. 7, and step 12 is the result of IBS-refactoring. Unlike the case of WCOX\textsubscript{IN}, the number of relations increases by 258 (about 48%), because a large number of links are added from an index page to others. After IBS-refactoring, WCOX\textsubscript{OUT} is decreased by 18.5%, from 7.50 to 6.33, but the count-based measure considerably increases. The cyclomatic complexity is also increased after IBS-refactoring. If we assume that every page is connected in one component, the cyclomatic complexity before and after refactoring is 256 and 511, respectively. This means that adding index pages without changing the main structure of the web application almost doubles the complexity, which is nonsensical. In short, size-based measures and CC increase during this process, but WCOX\textsubscript{OUT} decreases, which means that WCOX\textsubscript{OUT} is sensitive to the structural improvement from a user’s view. This implies that WCOX\textsubscript{OUT} may be applied as a bad smell detector for specific architectural patterns, such as IBS-refactoring.

Using the WANA tool, a new index site is added, which can have \( n \) sub-index pages, and a tree structure is created. For example, when three index pages are set to be added in WANA, one index page and three sub-index pages, a total of four pages are created and added. Assuming that the number of existing entities is \( N \), the number of added sub-index pages is \( n \), and the probability that a page is connected the index page is \( p \). The number of relations increase by \( n + NP \), where \( n \) is the number of relations from the index page to the sub-index pages, and \( NP \) is the number of relations from the sub-index pages to existing entities.

5.3 Discussion

In this subsection, the features of WCOX\textsubscript{IN} and WCOX\textsubscript{OUT} are shown based on various experiments, and they are compared with other measures, such as the cyclomatic complexity and count-based measures. It is hard for count-based complexity measures to represent the effects of structural changes due to the fact that the complexity grows according to the increased size. WCOX\textsubscript{IN} and WCOX\textsubscript{OUT} effectively incorporate structural changes associated with design improvements, irrespective of the increased size. The two measures commonly deal with structural patterns, but the targets differ in that WCOX\textsubscript{IN} corresponds to the developer’s view and WCOX\textsubscript{OUT} corresponds to the user’s view.

Fig. 10 shows normalized values of CC, the number of relations and entities, WCOX\textsubscript{IN}, WCOX\textsubscript{OUT}, and Zhang’s complexities\cite{13} for five different JSP open sources B2B \cite{41}, GIMS \cite{42}, JSPWiki \cite{43}, PIM \cite{44}, and Unicorn3 \cite{45}. The green lines in the figure represent relative increments of each metrics after refactoring in proportion to their initial values which are the values before refactoring and indicated by zero line. As shown in Fig. 10, only WCOX\textsubscript{IN} and WCOX\textsubscript{OUT} decrease consistently after CCA and IBS-refactoring respectively. Thus, it is difficult or impossible to detect these changes of architectural patterns with other metrics because they have tendency to increase as the size of web applications increase. However, WCOX\textsubscript{IN} and WCOX\textsubscript{OUT} are hardly affected from the size of web applications but from the structural changes caused by intended refactoring. Therefore, WCOX\textsubscript{IN/OUT} can be applied to make a refactoring decision on WAs, for better maintenance.

WCOX\textsubscript{IN} and WCOX\textsubscript{OUT} are entropy-based measures and they are sensitively influenced by architectural changes rather than size changes. Actual costs and efforts may
have a greater correlation with size-based measures. However, complexity measures cover various viewpoints, not only size, but also entropy or structural similarity. These different measures enable us to regard the complexity as a kind of pattern, and facilitate elaborate decisions for a target WA, via pattern analysis. It is expected that the various complexity measures can be combined with several AI-based techniques such as neural networks. The combined measure provides a broad view of target systems. WCOX\textsubscript{IN} and WCOX\textsubscript{OUT} can be utilized to differentiate WAs with a distinct structure and a similar size, in the combined measure.

WCOX\textsubscript{IN}/OUT could be also applied to derive estimated maintenance costs of web applications. Most of software metrics for costs are based on the size such as LOC (Line of Code) or FP (Function Point). However, efforts to maintain a web application are not only affected from its size but from its maintainability caused by architectural pattern. In other words, the maintenance costs could be different even the sizes of web applications are equal. Thus, WCOX\textsubscript{IN}/OUT which could represent the architectural tendency and complexity is one of useful candidates to make up for size-based cost estimation.

Finally, experiments have been conducted to show how the combined refactoring of IBS and CCA affects the WCOX\textsubscript{IN} and WCOX\textsubscript{OUT}. As presented in subsections 5.1 and 5.2, each refactoring improves the structural complexity of WAs, from the point of views of users or developers. In these experiments, IBS refactoring was conducted first to change the structure into index-based one, and then, CCA refactoring has been applied in order to manage the relations as a controller centric way, including the additionally generated ones from the IBS refactoring. The target WAs are 10 simulated WAs which have been randomly generated, and the WCOX\textsubscript{IN/OUT} in Fig. 11 have been averaged on the WCOX\textsubscript{IN/OUT} results of the 10 WAs. Fig. 11 presents the averaged WCOX\textsubscript{IN/OUT} values before (randomized) and after combined refactoring (IBS + CCA refactoring) for diverse scopes of entities and relations.

The results show that both of WCOX\textsubscript{IN} and WCOX\textsubscript{OUT} have been decreased after the IBS + CCA refactoring, indicating that the combined refactoring improves the original structure of a WA.

6. CONCLUSIONS

In this paper, web application complexity metrics, WCOX\textsubscript{IN} and WCOX\textsubscript{OUT}, have been proposed, which are based on entropy. Initially, several relations, such as, link, in-
Before refactoring  

After refactoring  

Fig. 11. WCOX IN/OUT of randomized and IBS + CCA refactored WAs.

clude, redirect, load, and submit, including parameters, are extracted via static analysis of a WA. And then, a web application model is created, which is composed of nodes and directed edges with weights. In the model, a node is a page, an edge is a relationship between two pages, and a weight is obtained from the relation type and parameters. Finally, for in-directions and out-directions, complexity measures WCOX IN and WCOX OUT are defined with a reference probability for each directed edge. It is demonstrated that WCOX IN satisfies most of Weyuker’s complexity axioms [35]. Two experimental models, CCA and IBA, are also presented and experiments are conducted to show the measures’ applicability compared with other count-based metrics and cyclomatic complexity.

The structural aspects of WAs are represented by WCOX IN, WCOX OUT. These metrics indicate various aspects of target WAs better than count-based metrics, because WCOX IN, WCOX OUT are based on the average information quantity, entropy. In addition, they can be used as measures of system aging and degradation, which is the characteristic of an entropy-based measure. The entropy value increases as the granularity of the system becomes finer, and the apparent disorder decreases as the granularity becomes coarser. Therefore, WCOX IN and WCOX OUT can be used to restructure systems by clustering. When the various aspects of measures, which incorporate not only the existing count-based measures, but also the proposed entropy-based measures, are applied to analyze a group of WAs, the analysis yields more precise results. Additionally, WCOX, which is a metric of the structural disorder, can be used as a metric of the kind of ‘bed smells’ that require refactoring, for a single web application whose evolution involves multiple versions.

In future works, dynamic analysis will be added to the modeling process, to supplement the static analysis. WAs with a long maintenance term and multiple versions can facilitate validation of the proposed approach. Since many distinct technologies are used for web applications and a strict web format is not adhered to, it is also necessary to analyze and classify the various types of WAs.
REFERENCES

16. T. J. McCabe, “A complexity measure,” IEEE Transactions on Software Engineer-


Woo-Sung Jung (_DISPLAY_NAME) received the B.S. degree in Computer Science and Engineering from Seoul National University, Korea, in 2003, was a researcher in SK UBCare from 1998 to 2002. He is currently under a combined master’s-doctoral program studying Software Engineering in Seoul National University, Korea. He is a member of the Software Engineering Lab. His research interests include the area of software evolution, software architecture and adaptive software system.

Eun-Joo Lee (DISPLAY_NAME) received her B.S., M.S., and Ph.D. degrees in Computer Science from Seoul National University, Korea in 1997, 1999, and 2005, respectively. She was a research staff member at Samsung Advanced Institute of Technology from Nov. 2005 to Feb. 2006. Currently, she is an Assistant Professor of School of Computer Science and Engineering at Kyungpook National University. Her current interests include software reengineering, software metrics, web engineering, and web services.
Kap-Su Kim (金甲壽) received his Ph.D., M.S. and B.S. degrees in Computer Science and Information Engineering from Seoul National University, Seoul, Korea, in 1996, 1987, and 1985, respectively. He was working for Software Development in Samsung Electronics, Korea, from February 1987 to April 1992. He has been a Professor of Computer Science Education Department, Seoul National University of Education, Seoul, Korea, since 1998. His research interests include software engineering, web engineering and computer science education.

Chi-Su Wu (禹治水) received his B.E. degree in Applied Mathematics from Seoul National University, Korea in 1972 and his M.S. and Ph.D. degrees in Computer Science from Seoul National University in 1977 and 1982, respectively. He served as a researcher at the Loughborough University, UK in 1978. From 1975 to 1982, he was an Associate Professor of Computer Science at Ulsan University, Korea. Currently, he is a Professor of Computer Science and Engineering at Seoul National University, Korea. His current research interests include software engineering and programming languages.