International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(4): 821-837
The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085)

Model-based Web Components Testing: Prioritization Using MIDS and Centrality Measures

Ahmed Al-Herz and Moataz Ahmed
Information and Computer Science Department,
King Fahd University of Petoleum and Minerals,
Dhaharan 31261, Saudi Arabia
{alherz, moataz}@kfupm.edu.sa

ABSTRACT

Web applications testing and verification is becoming a highly challenging task. A number of model-based approaches has been proposed to deal with such a challenge. However, there is no criteria that could be used to aid practitioners in selecting appropriate approaches suitable for their particular effort. In this paper we present a set of attributes to serve as criteria for classifying and comparing these approaches and provide such aid to practitioners. The set of attributes is also meant to guide researchers interested in proposing new model-based Web application testing and verification approaches. The paper discusses a number of representative approaches against the criteria. Analysis of the discussion highlights some open issues for future research. In response to one of the issues, we present an approach for prioritizing components for testing to maximize confidence given a limited number of test cases to be executed. Some initial results are reported in the paper.

KEYWORDS

Web applications, model-based testing, testing prioritization, Web verification.

1 INTRODUCTION

Web applications are becoming more complex. As more and more services and information are made available over the Internet and intranets, Web sites have become extraordinarily complex, while their correctness is often crucial to the success of businesses and organizations. Although traditional software testing is already a notoriously hard, time-consuming and expensive process, Web-site testing presents even greater challenges. Complexity arises due to several factors, such as a larger number of hyperlinks, more complex interaction, frequently changing Web pages, and increased use of distributed servers. Moreover, the environment of Web applications is more complex than that of typical monolithic or client-server applications – Web applications interact with many components, such as CGI scripts, browsers, backend databases, proxy servers, etc., which may increase the risk of interoperability issues. Furthermore, many Web applications have a large number of users with no training on how to use the application – they are likely to exercise it in unpredictable ways. Therefore, Web sites that are critical to business operations of an organization should be tested thoroughly and frequently 9.
Modeling helps to manage the complexity of these systems. Several

papers in the literature have studied the problem of web applications modeling for the sake of managing the overall development complexity. Modeling support is essential to provide an abstract view of the application. It can help designers during the design phases by formally defining the requirements, providing multiple levels of detail as well as providing support for testing prior to implementation. Support from modeling can also be used in later phases to support verification. Different models have been proposed, while others have been adapted from existing modeling techniques for other types of software
[1][2][3][4][5][6][8][22][23][24][25][26][27][28][29][30][31][32][33].

In this paper we focus on Web applications testing and verification and study the different model-based approaches for managing associated complexity. In the domain of model-based testing, it is generally understood that the model is an abstraction or simplification of the behavior of the application to be tested. The model is captured in a machine readable format with the sole purpose of acting as both test sequence (trace) generator and oracle. There are many approaches to proposing a model for the purpose of Web application verification and testing. This paper studies some models that are currently applied in the field of verification and testing of web applications. Our literature survey revealed that some approaches focuses on testing the navigational aspects of web applications. Others concentrate on solving problems arising from user interaction with the browser in a way that affects the underlying process. Others are interested in dealing with static and dynamic behavior. In our bid to carry out a critical survey of the literature on using models for testing and verification of Web applications, we discovered that a common ground for classifying and comparing existing approaches is not available. This motivated our research to come up with a set of attributes serve as criteria for classifying and comparing various modeling approaches to Web application testing and verification. This set of attributes is presented in Section 2.

The analysis of a number of representative approaches against the criteria highlights some open issues for future research as discussed later. An issue of interest in this paper is that a typical Web application consists of a large number of components (i.e., front-end pages and backend processing). A Web page can be static—where content is constant for all users—or dynamic—where content changes with user input. A typical Web application could also be distributed. Accordingly, even regression testing could take weeks to test all of the test cases from a previous version 13. Due to time and resources constraints, it would be desirable to help the tester prioritize the test cases in a way that maximize confidence given a limited number of test cases to be executed. However, the problem of prioritizing Web application components for testing did not catch enough researchers' attention. In this paper we propose an approach for an approach for prioritizing components to be tested. Such prioritization could then be used to prioritize corresponding test cases.

The rest of paper is organized as follows: Section 2 gives the comparison

and categorization criteria. Section 3 discusses different approaches found in the literature in light of the criteria. Section 4 presents an approach for suggesting a prioritization as which component to be tested first. Finally we conclude and highlight some possible future work in Section 5.

2 COMPARISON AND CATEGORIZATION CRITERIA

System modeling is a new emerging technology. System models are created to capture different aspects of the system behavior. Several modeling languages have been developed to model state-based software systems, e.g., State Charts, Extended Finite State Machine (EFSM) 14, and Specification Description Language (SDL) 15. System modeling is very popular for modeling state-based systems, e.g., computer communications systems, industrial control systems, etc. System models are used in the development process, e.g., in partial code generation, or in the testing process to design test cases. Over the years, several model-based test generation 141617 and test suite reduction 18 techniques have been developed.

Modeling can be viewed from three different perspectives: the objective problem (security, testing etc.), the particular problem at hand (a specific case with its own characteristics e.g., ecommerce application), and finally the model type (e.g. FSM, SDL, etc.). There is still much uncertainty as to which model-based approach suits which type of Web application testing and/or verification effort. Assessing a model-based approach, in our own view, should not only be based on the underlying model expressiveness, but also on characteristics of the overall approach. We address this type of uncertainty by proposing a set of attributes to allow for classification and comparison of approaches. These assessment attributes offer more, beyond their usefulness in carrying out comparison of approaches. They can also serve as guidance to researchers attempting to develop model-based Web application testing and verification approaches. We discuss these attributes in the sequel.

Aspects Coverage: This attribute considers the Web application aspects that are being modeled by the models. These aspects are classified into three categories namely, static, dynamic and interaction aspects.

Static aspects: Static aspects of web applications include static HTML pages and the hyper links that connect the static pages with other static HTML pages. When the user clicks on a static link, a request is sent to the server to retrieve the target page.

Dynamic aspects: These aspects of web application include dynamic HTML pages that contain dynamic content and links. Dynamic contents and links are generated by backend processing based on inputs obtained from users or other supporting software.

Interaction aspects: These aspects take into consideration the user interaction with the web application. User interactions may include back page, switching to another page by typing the URL in the browser, opening multiple pages at the same time. Models can capture these types of user interactions and represent the effect on the content, behavior or the navigation.

Underlying Model: Web applications components are represented using different conceptual models, for example, some uses object relation diagram others use finite state machines model.

Perspective of Modeling: Web application models can be analyzed from different perspectives, like navigation, and behavior. These perspectives can be static or dynamic.

Objectives of the Model: Web application models have different objectives, some models objective is testing, other models objectives are implementation or design verification and model verification against a set of properties.

Source Code Requirement: Verification or testing can be a white box or a black box testing or verification. If white box testing is used by a model then the source code is required while, the black box testing requires test cases only.

Tool Support: Some models are supported by tools for automatic model generation, verification or testing, while other models are still not supported.

Expressiveness: Some models represent and convey structural, behavioral and functional aspects of web applications components for both external and internal view of the component more effectively in this case the expressiveness would be high, while other models may represent only the structural aspect or the behavioral aspect. Some models represent the external relations between components only.

Complexity: This attribute determines the complexity of the models, some models needs complex model to represent the components in term of the size and the attribute needed to represent entities and relations.

3 CRITICAL SURVEY

In this section, we present a summary discussion of some representative works based on our set of attributes. The list of considered approaches in our study is not exhaustive, but we gave attention to those works we considered representative with regard to the subject under discussion. We also discuss the shortcomings associated with the different approaches considered. It is worth noting here that we used subjective ratings in evaluating the different approaches, e.g., high expressiveness and low complexity. Future work will investigate applying more quantitative objective ratings.

3.1 Model Checking-based Verification of Web Application

Miao et al. [1] focus on automated verification of Web applications by using model checking method. The approach involves two models, the design model and the implementation model of a Web application. To verify if an implemented Web application performs in accordance with its design, the approach analyzes the design model to generate properties in temporal logic formulas that are model checked on the implementation model. Their work focuses on black-box automated verification of a Web application by using model checking method. The approach involves two formal models: a design model denoted by *WA*D, from which the temporal logic properties for a Web application are derived, and an

implementation model, denoted by *WA*I, which is model checked in order to verify those derived properties. An Object Relation Diagram (ORD) is employed to represent the design structure of a Web application, i.e., design model. Aiming at the verification of the external behavior of a Web application from client's point of view, *WA*D is intended to describe Web pages, software components interacting directly with the Web pages, and their relationships. The Kripke structure used for model checking is employed to model the implementation of a Web application, it is a type of state transition graph consisting of nodes representing the reachable states of the system and edges representing the state transitions of the system. All properties generated from *WAD* are model checked on *WAI* by using model checker *SMV (Symbolic Model Verifier).* SMV will provide a diagnostic sequence in the stack whenever a violation of the property is detected.

With regard to the tool support, this approach offers a prototype which automatically analyzes the design model to build the properties in CTL and delegates the task of property verification to the existing model checker *SMV* where the implementation model is typed in manually.

The model's level of expressiveness is considered to be moderate. While it provides a way to describe the components and the relation between them and the external view of the model very effectively, the model does not describe the low-level details and the internal behavior of each component.

The approach is considered to be of moderate complexity; the directed graph describes the external relation between components.

3.2 Testing Web Applications by Modeling with FSMs

In this approach the authors builds hierarchies of Finite State Machines (FSMs) that model subsystems of the web applications 2. This approach proceeds in two phases. Phase 1 builds a model of the web application. This is done in four steps: (1) the web application is partitioned into clusters, (2) logical web pages are defined, (3) FSMs are built for each cluster, and (4) an Application FSM is built to represent the entire web application. Phase 2 then generates tests from the model defined in Phase 1.

Tool support: They developed a research prototype in Java. It has a graphical editor to input the FSMs and the constraint descriptions. It also generates expected outputs in the form of the next state (LWP) to serve as a simple test oracle. Path generation includes edge coverage and roundtrip. Input selection is based on using an input value database. The resulting sequences of test inputs are made executable by transforming them into an *Evalid* script.

With regard to the level of expressiveness, it is high in the lowest level and low in the highest level of the hierarchy. The low level details of operations and interconnection can be observed and described; at the higher level in the hierarchy, however, the model becomes more abstract, and some of details become invisible.

The approach is considered to be of high complexity in the lowest level and low complexity in the highest level of the

hierarchy. At the low level of the hierarchy, details of operations and interconnection are modeled by FSM which require many and complex interactions but in the higher level in the hierarchy the model becomes more abstract and simpler.

3.3 An Object-Oriented Web Test Model for Testing Web Applications

Kung et al. in 3 propose a model that extends traditional test models, such as control flow graph, data flow graph, and finite state machines to web applications for capturing their test-related artifacts. Based on the proposed test model, test cases for validating web applications can be derived automatically. In this methodology, both static and dynamic test artifacts of a web application are extracted to create a Web Test Model (WTM) instance model. Through the instance model, structural and behavioral test cases can be derived systematically to benefit test processes. Test artifacts are represented in the WTM from three perspectives: the object, the behavior, and the structure.

From the object perspective, entities of a web application are represented using object relation diagram (ORD) in terms of objects and inter-dependent relationships.

In particular, an ORD = (V, L, E) is a directed graph, where V is a set of nodes representing the objects, L is a set of labels representing the relationship types, and ($E \subseteq V \times V \times L$) is a set of edges representing the relations between the objects, There are three types of objects in WTM: client pages, server pages, and components, to accommodate the new features of web applications, new relationship types are introduced in addition to those in the object-oriented programs. The new relationship types, navigation, request, response, and redirect are used to model the navigation, HTTP request/ response, and redirect relations introduced by web applications, respectively. Thus, in the ORD, the set of labels L = I, Ag, As, N, Req, Rs, Rd, where I: inheritance, Ag: Aggregation, As: association.

From the behavior perspective, a page navigation diagram (PND) is used to depict the navigation behavior of a web application. The PND is a finite state machine (FSM). Each state of the FSM represents a client page. The transition between the states represents the hyperlink and is labeled by the URL of the hyperlink. The PND of a web application can be constructed from an ORD. To deal with the dynamic navigation (the construction of client pages can be dynamic at runtime based on the data submitted along with the HTTP requests or the internal states of the application. Hence, the same navigation hyperlink may lead to different client pages). To model this behavior a guard condition enclosed in brackets is imposed on the transition in the PND. The guard condition specifies the conditions of the submitted data or internal system states that must be true in order to fire the transition. To detect the errors related to navigation behavior a navigation test tree is employed. A navigation test tree is a spanning tree constructed from a PND, by analyzing the tree; they can check some properties, such as reachability and deadlock, of the navigation behavior. At the same time, a set of object state diagrams (OSDs) are

used to describe the state behavior of interacting objects. It can represent the state-dependent behavior of an object in a web application. The state-dependent behavior for an aggregate object then can be modeled by a composite OSD (COSD) of the corresponding OSDs.

The structure perspective of the WTM is to extract both control flow and data flow information of a Web application. To capture control flow and data flow information, the Block Branch Diagram (BBD) and Function Cluster Diagrams (FCD) are employed in the WTM. The BBD is similar to a control flow graph. It is constructed for each individual function of a Web application to describe the control and data flow information, including the internal control structure, variables used/defined, parameter list, and functions invoked, of a function. Therefore, the BBD can be used for traditional structural testing of each individual function; the FCD is a set of function clusters within an object. Each function cluster is a graph G = (V,E), where V is a set of nodes representing the individual functions and $E \subseteq V \times V$, is a set of edges representing the calling relations between the nodes.

The approach offers a very high level of expressiveness. Different models are used to describe external, behavioral and internal aspects of components which can express the model effectively.

The approach is considered to be of very high complexity. Many models are used to describe the internal, behavioral and external structure of components so the overall system model is very complex.

3.4 Formal Verification of Web Applications Modeled by Communicating Automata

Haydar et al. in 4 devise an algorithm to convert the observed behavior, which they called a browsing session, into an automata based model. In case of applications with frames and multiple windows that exhibit concurrent behavior, the browsing session is partitioned into local browsing sessions, each corresponding to the frame/window/frameset entities in the application under test. These local sessions are then converted into communicating automata. They did an implementation for a framework which includes the following steps: The user defines some desired attributes through a graphical user interface prior to the analysis process. For example, reachability properties, and the checking for frame errors , frames having same name are not active simultaneously. These attributes are used in formulating the properties to verify on the application. A monitoring tool intercepts HTTP requests and responses during the navigation of the Web Application Under Test (WAUT). The intercepted data are fed to an analysis tool, which continuously analyzes the data in real time (online mode), incrementally builds an internal data structure of the automata model of the browsing session, and translates it into XML-Promela. The XML-Promela file is then imported into aSpin, an extension of the Spin model checker. ASpin then verifies the model against the properties, furthermore the model checking results include counterexamples that facilitate error tracking.

The approach is supported with a framework that is composed of; GUI to collect desirable properties from the user, network monitoring tool to intercept HTTP request and response, analysis tool that builds the communicating automata based on the received data. The model is fed into aSpin for verification.
The approach offers a low level of expressiveness, as the model describes a session or multiple sessions, which may not give a full description of the complete model of the system; it depends on how the user will interact with the application.
The approach is considered to be of high complexity; based on the user input the FSM can get complex.

3.5 Verifying Interactive Web Programs

Licata et al. in 5 describe a model checker designed to identify errors in web software. A technique for automatically generating novel models of web programs from their source code was presented. These models include the additional control flow enabled by user operations. They presented a powerful base property language that permits specification of useful web properties, along with several property idioms that simplify specification of the most common web properties. The authors model a web program P by its web control-flow graph (WebCFG). The WebCFG is an augmented control-flow graph (CFG). User interaction control flows are being added to the model to build a sound verification tool. The authors reduce user operations to primitive user operations proposed by Graunke et al. 8. All traditional browser operations can be expressed in this calculus; they just account for switch and submit. Then they construct the WebCFG completely automatically from the source of a web program using a standard CFG construction technique followed by a simple graph traversal to add the post-web-interaction nodes and the web-interaction edges. The resulting model and properties are checkable by language containment. This work doesn't address the concurrency issues resulting from multiple simultaneous accesses to a server by different clients.
With regard to tool support, the authors implemented their own model checker tool to support their approach.
The approach models are meant to prove properties of interactive web sites by discovering user operation- related bugs, as well as providing a method for verifying all-paths properties of interactive web sites.
The approach offers high level of expressiveness. CFGraph describe details of behaviors of components and how these interact with each other. In addition, adding the user operations to the model makes the model describe the behavioral aspect based on the user operations.
The approach is considered to of very high complexity; CFGraph is very complex, especially when the user operation is involved in the model.

3.6 Web Site Analysis: Structure and Evolution

Ricca et al. in 6 adapts an approach to analyze, test, and restructure web application based on a reverse engineering paradigm. They didn't

propose models and formalisms to support the design of web applications; instead, based on the assumption that a web application already exists, they investigate different well established methods for the analysis, testing and re-structuring of traditional software systems, adapting them to the case of Web applications. In [6] web application is modeled as a graph; nodes and edges are split into different subsets. Nodes subsets are a set of all web pages; a set of frames for one web page; and a set of all frames.

Edges are also split into three subsets according to the kind of target node; a set of hyperlinks between pages or a relation showing the composition of web page into frames; a set of the relations between frames and pages; as they show which page in which frame is loaded; and a set of relations showing the loading of a page into a particular frame. The name of the frame is given as a label next to the link. This model is implemented in ReWeb. The ReWeb 7 tool consists of three modules: a Spider, an Analyzer and a Viewer. The Spider downloads all pages of a target web site, starting from a given URL and providing the input required by dynamic pages, and then it builds a model of the downloaded site. The Analyzer uses the UML model of the web site and the downloaded pages to perform several analyses. Since the structure of a Web application can be modeled with a graph, several known analysis, working on graphs, such as flow analysis and traversal algorithms can be applied. The Viewer provides a Graphical User Interface (GUI) to display the Web application view as well as the textual output (reports) of the analyses.

With regard to supportability, the approach is supported by the ReWeb tool. The ReWeb tool can periodically download the entire set of pages in a site. Results of the analyses are then provided to the user, by exploiting different visualization techniques. Colors are employed in the history view, while structural and system views are enriched with powerful navigation facilities. Pop-up windows associated to nodes are used to show the textual results of the structural analyses.

The level of expressiveness is low; the model described by directed graph only. The approach is considered to be of low complexity; only a directed graph is involved in the model.

3.7 Summary

Table 1 shows the summary of the 6 different methods described.

Table 1. Summary of Findings

Method	Aspect type	Model	Perspective	Objective	Source code	Tool Support	Expressiveness	Complexity
Miao et al.	Static + dynamic	ORD	Navigation + behavior	Implementation verification against design	Yes	Prototype	Moderate	Moderate
Andrwes et al.	Static + dynamic	FSM, AFSM	Navigation + behavior	Testing	No	Prototype	Low	Low
Kung et al.	Static + dynamic	ORD, PND, OSD, BBD, FCD	Behavior + navigation	Testing	Yes	None	Very High	Very High

Haydar et al.	Static + dynamic	Communicating automata	Navigation + behavior	Model verification against defined properties	No	GUI + network monitoring tool + analysis tool	Low	High
Licata et al.	Interaction	WebCFG	Interaction behavior	Model verification against interactive properties	Yes	Implement a model checker	High	Very High
Ricca et al.	Static	Directed graph	Navigation	Original design verification during evolution and Testing	Yes	ReWeb	Low	Low

4 COMPONENTS TESTING PRIORITIZATION

From Table 1 we can see that methods discussed are lacking ways to prioritize Web application components for testing. This untreated aspect is very important especially when we know that development and deployment cycles of Web applications are dramatically becoming short, and testing is often considered a cost-intensive and time-consuming process. Here, we give several suggestions which could be investigated more thoroughly in future works. First solution is to apply an algorithm to find the minimum independent dominating set on the graph based model, then we can consider these set as the highest priority components to test. The rationale here is that these dominating components can be regarded as super components because they are connected to many other components. Also the components in this way are either dominating or dominated by others; so, all components that may lead to other components can be tested. Another suggestion is to rank components based on several graph centrality measures which have been used extensively in social and biological networks to find important nodes [36][38][39]. In this research, we use three basic centrality measures namely the degree centrality measure, the *betweeness* centrality measure and the closeness centrality measure. The degree centrality measure suggests that an important node is involved in a large number of interactions. For directed networks, there are two notions of degree centrality: one based on fan in-degree and the other on fan out-degree. A node with high fan in-degree or fan out-degree is ranked higher than those of less degree; since high degree means that most probably many components will leads to this component or a component may use many services from the others. *Betweeness* centrality measure can be used to rank components. The measure reflects the intuition that an important node will lie on a high proportion of paths between other nodes in the network. Closeness centrality measures can be applied to a graph based model to rank components [34]. The rationale behind such a ranking is that the components that are close to many

components is important and greatly affect the overall system behavior.
In the sequel, we apply these methods on an ORD model design (Figure 1) to test their suitability. To conclude, we give analyses of their corresponding performance.

4.1 Minimum Independent Dominating Set Method (MIDSM)

A dominating set D of a graph G(V, E) is a subset of V in which each vertex v ∈ (V – D) is adjacent to at least one vertex u ∈ D, i.e., (v, u) ∈ E. An independent dominating set is a dominating set (where D is independent, i.e., (u, v) ∉ E, for all u, v ∈ D). Since finding the minimum independent dominating set is NP-Complete problem 21, we will use a greedy algorithm to find a set that is as minimum as possible. First, we will find the minimum independent dominating set by using a greedy algorithm which can be applied on undirected graph and it will choose a node with maximum degree and delete the neighbors. So, the first step is to convert the model to an undirected graph, the result can be seen in Figure 2.

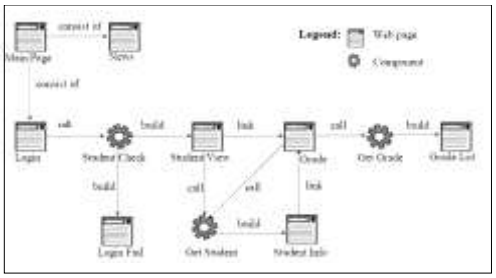

Figure 1. An ORD design model[1].

If we apply the greedy algorithm we will choose node H which corresponds to the grade web page as the first node because it has degree of four which is the maximum and delete all neighbors. Now we can select either node A or node D since they have the highest degree which is two, let us select A assuming there is no any other criteria for selection. Now we can select node D and then select node K. So, the final set is H, A, D and K.

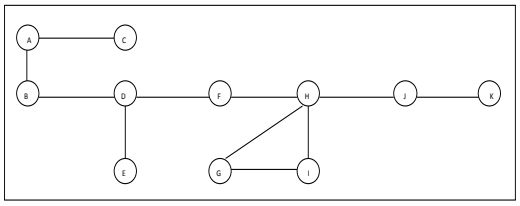

Figure 2. The undirected graph of the ORD model.

Analysis

The grade web page is used and uses more components than the other nodes so it is indeed an important page. The main page was selected as an important page but it is not since it only contains links to two pages so it is static. Student check component is important since it check for the validity of the user. The grade list web page is selected as an important page but it is not since it only contains the final results which depend on the get grade component which is more important. In addition, the method missed by two pages which is more than that of the other components. The weakness of this approach is when there is more than one node with the same degree; in this case, which one to select? We could define more criteria for selection like the type of the node and the type of the edge which can impact the selection. Another weakness is not considering the importance of the direction which may impact the importance of the components. Also, if we delete the neighbors, we might

actually delete an important component or page.

4.2 The Degree Centrality (DC) Measure

The *degree centrality* measure is perhaps the simplest centrality measure, though it is often a highly effective measure of the influence or importance of a node [35]. The idea behind using the degree centrality measure of importance in a network is the following: An important node is involved in a large number of interactions. Formally, for an undirected graph G, the degree centrality of a node $u \in V(G)$ is given by $DC(u) = deg(u)$ 19. For directed networks, there are two notions of degree centrality measure: one is based on fan in-degree and the other one is based on fan out-degree. Considering the ORD model design (Figure 1), let us rank the components based on the fan in-degree and fan out-degree. On the one hand, the Get Student component and the "Grade" page" have fan in-degree of value 2 which is the highest degree. The News, Login, Login Fail, Student View, Student Info, Grade List pages, and the Student Check component with degree of value 1. The Main page has the lowest degree with degree of value 0. On the other hand, the Main, Student View and Grade pages, and the Student Check component have fan out-degree of value 2 which is the highest degree. The results are depicted in Figure 3 and Figure 4.

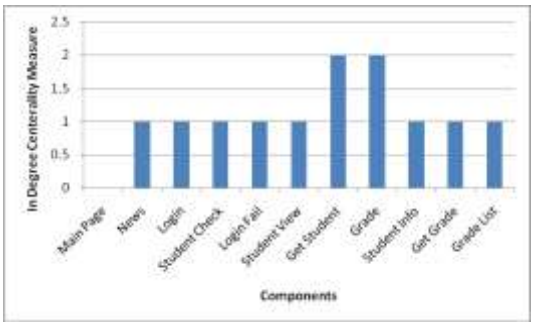

Figure 3. The in degree centrality measures of the ORD model components.

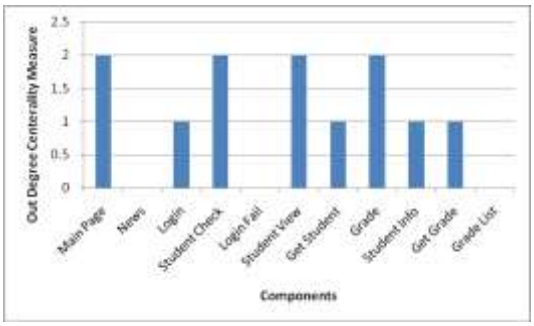

Figure 4. The out degree centrality measures of the ORD model components.

Analysis

The result of the fan-in degree centrality show better ranking of importance because if the components which have high fan in-degree fail then many other components will fail to get the services. Get student and grades page are used by more components than the other components, so any failure in these components will make the other component fail. The issue is that we might have many components with same degree, the question is how we can prioritize these with same degree; we might add more criteria like the component type and the fan in-edges types. Considering the results of the out degree centrality measure, we can see good ranking from another perspective, though it is less effective than the fan-in

degree centrality measure. Components with high fan out degree acts as main hubs that allow a service or a user to use or transfer to many components, if such components fail then all other connected components and services cannot be reached. Returning to our example, main page will direct the user to either news or login components, while news components has fan out degree of value 0, which means it does not use or lead to any services so it is less important, therefore it should have lower testing priority.

4.3 Betweeness Centrality (BC) Measure

Betweenness centrality 203637 is a fundamental measurement concept for the analysis of networks. The book by Hage and Harary (1991) shows some of its many descriptive and predictive uses [40].

The idea behind this measure is the following: An important node will lie on a high proportion of *shortest paths* between other nodes in the network. Formally, for distinct nodes, u, v, w $\in$ V(G), let σuv be the total number of shortest paths between u and v and σuv(w) be the number of shortest paths from u to v that pass through w. Also, for w $\in$ V(G), let V (u) denote the set of all ordered pairs, (u, v) in V(G) $\times$ V(G) such that u, v, w are all distinct. Then, the betweenness measure of w, BC(w), is given by

$$BC(w) = \sum_{(u,v)\in V(w)} \frac{\sigma uv(w)}{\sigma uv} \quad 20. (1)$$

It is in most cases only an approximation to assume that information flows along shortest paths; normally it will not, and variations of betweenness centrality such as "flow betweenness" and "random walk betweenness" have been proposed to allow for this. In many practical cases however, the simple (shortest path) betweenness centrality gives quite informative answers. 35

Now let us use the *betweeness* measure to rank the importance of the components. First, all shortest paths between any pairs of components in the model are found. Then we will go over all components and see on which paths they exist. The Main page and the News page do not come between any other components in a path so their BC is 0. Login exists on 8 paths so its BC is 8. Student Check comes between 14 components on different shortest paths so its BC is 14. Student view's BC is 15. Get student comes between 4 components on different shortest paths so its BC is 4. Student info page's BC is 3. Grade component's BC is 13. Get grade component exists on 7 paths so its BC is 7. The grade list page is not between any other pages so its BC is 0. From the results we can see that student view page has the highest BC then student check component and then the grade page, after that login, get grade, and get grade and student info page. The result is shown in Figure 5.

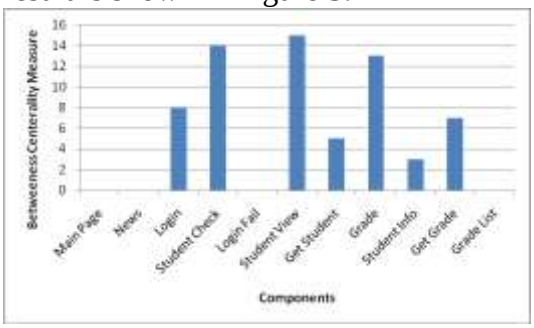

Figure 5. The betweeness centrality measures of the ORD model components.

Analysis

The results show good ranking because if any components which comes between many other components fail, then the other components will fail to reach the other component, which means those components with high BC are bottle nicks so they are important and their priority in testing should be high. The weakness in this approach is that we might have components with the same BCs, the question is which components is more important within these components, so we need to add more attributes like the type of components, and the type of edges in these components.

4.4 Closeness Centrality (CC) Measure

The basic idea behind this type of measures is the following: An important component is typically close to, and can communicate quickly with, the other components in the network. Sabidussi in 34 defined the term closeness centrality of node CC(u) as follows:

$$CC(u) = \frac{n-1}{\sum_{v \in V, u \neq v} d(u,v)} = \frac{1}{\bar{d}}. \quad (2)$$

where d(u, v) is the distance between node u and node v; V is the set of all nodes; $\bar{d}$ is the average distance between u and the other nodes. For directed networks, the centrality is called output closeness centrality when d(u, v) is defined as the path length from u to v. Also, some vertices may not be reachable from vertex u—two vertices can lie in separate "components" of a network, with no connection between the components at all. In this case, *closeness* as described above is not well defined. The usual solution to this problem is simply to define closeness to be the average geodesic distance to all *reachable* vertices, excluding those to which no path exists 35.

Let's rank the components in Figure 1 using the closeness centrality measure. Consider the Main page, the summation of all shortest distances is 34 and the average is 3.4, so the CC (Main page) is 0.294118. On the other hand, the closeness centrality measure of the News component is 0 since it does not lead to another component. In addition, the closeness centrality measure of the Get Grade component is 1 which the highest measure since it leads to one component and the distance is 1. It should be noted that the highest possible value is 1 which is attainable when the distances between a node and all reachable node are one. The measures of all components can be found in Figure 6.

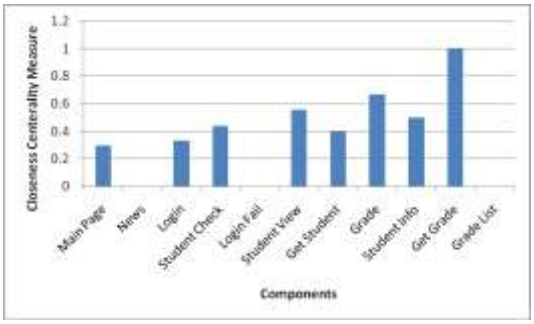

Figure 6. The closeness centrality measures of the ORD model components.

Analysis

The Get Grade component has the highest closeness centrality measure, which contradict the previous measures we found, this is because the closeness centrality measure takes only the reachable components into consideration and ignores those components that cannot be reached. If we discard the Get Grade component we will see a reasonable ranking with the Grade component as the highest ranked

component and then the Student View component which are indeed important. Therefore, the closeness centrality measure will give better result if the component model does not contain single sequential links or we can eliminate those components from the process of ranking. As we mentioned before, adding more attributes to the ranking process may lead to better results, however adding attributes will be less effective since most results are distinct and not like the previous measure where we have similar results.

5 CONCLUSION AND FUTURE WORK

In this paper we proposed set of attributes for classifying and comparing model-based Web applications testing and verification approaches. We discussed six different representative analysis models that are currently applied in the field. We summarized our discussion in Table 1 which reveals that the methods discussed are lacking ways to prioritize web application components for testing. We suggested four methods to allow for prioritizing components: MIDSM in addition to three centrality measures namely degree (DC), betweeness (BC) and closeness centrality (CC) measures. We illustrated the suggested methods on an ORD design model. The results show that the MIDSM has some shortcomings and may miss important components and consider not important components. The fan in-DC measure and BC measure show better results than the out-DC measure and the CC measure, with some issues. The issues can be addressed by incorporating more attributes and criteria for selections like the type of the components, and the edges, these attributes in addition to others can be investigated in future work. Also, we plan to investigate combining the DC measure and BC measure together to rank the components by assigning a percentage for each measure in future work as well. The percentage can be learned from experience, and using machine learning methods to find the best percentage. It is worth noting here though that in this paper we only demonstrated the approach using an illustrative example; in future work, we will conduct more rigorous analysis of the different methods.

Acknowledgements. The authors wish to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for utilizing the various facilities in carrying out this research.

6 REFERENCES

1. Miao, H., Zeng, H.: Model Checking-based Verification of Web Application. Proceedings of 12th IEEE International Conference on Engineering Complex Computer Systems. pp. 47--55 (2007).
2. Andrews, A., Offutt, J., Alexander, R.: Testing Web Applications by Modeling with FSMs. Software Systems and Modeling. vol. 4, no.3. pp. 326--345 (2005).
3. Kung, D. C., Liu, C. H., Hsia, P.: An Object-Oriented Web Test Model for Testing Web Applications. In: Proceedings of the 1st Asia-Pacific Conference on Web Applications, pp. 111--120. IEEE Press, New York (2000).
4. Haydar, M., Petrenko, A., Sahraoui, H.: Formal Verification of Web Applications Modeled by Communicating Automata. In: Proceedings of the 24th IFIP International Conference on Formal Techniques for Networked and Distributed Systems, pp. 115-132. Madrid, Spain (2004).

5. Licata, D. R., Krishnamurthi, S.: Verifying interactive web programs. In: Proceedings of the IEEE International Conference on Automated Software Engineering, pp. 164--173. IEEE Computer Society (2004).
6. Ricca, F., Tonella, P.: Web site analysis: Structure and evolution. In: Proceedings of the International Conference on Software Maintenance, pp. 76--86 (2000).
7. Ricca, F., Tonella, P.: Building a tool for the analysis and testing of Web applications: Problems and solutions. In: Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems Genova. vol. 2031. pp. 373--388. Italy (2001).
8. Graunke, P. T., Findler, R. B., Krishnamurthi, S., Felleisen, M.: Modeling web interactions. In: Pierpaolo Degano, editor, Proceedings of the Programming Languages and Systems, 12th European Symposium on Programming, ESOP 2003, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003. LNCS. vol. 2618. pp. 238--252. Springer (2003).
9. Benedikt, M., Freire, J., Godefroid, P.: VeriWeb: Automatically Testing Dynamic Web Sites. In: Proceedings of 11th International World Wide Web Conference (2002).
10. Sampath, S., Bryce, R., Viswanath, G., Kandimalla, V., Koru, A.G.: Prioritizing User-Session-Based Test Cases for Web Application Testing. In: Proceedings of IEEE Int'l Conf. Software Testing, Verification, and Validation, pp. 141--150 (2008).
11. Bryce, R. C., Sampath, S., Memon, A. M.: Developing a Single Model and Test Prioritization Strategies for Event-Driven Software. IEEE Transactions On Software Engineering. vol. 37, no. X, XXXXXXX (2011).
12. Korel, B., Tahat, L. H., Harman, M.: Test Prioritization Using System Models. In: Proceedings of the 21st IEEE International Conference on Software Maintenance (2005).
13. Rothermel, G., Untch, R. H., Chu, C., Harrold, M. J.: Prioritizing Test Cases for Regression Testing. IEEE Trans. Software Eng.. vol. 27, no. 10. pp. 929--948 (2001).
14. Cheng, K., Krishnakumar, A.: Automatic Functional Test Generation Using The Extended Finite State Machine Model. In: Proceedings of ACM/IEEE Design Automation Conf., pp. 86--91 (1993).
15. Dssouli, R., Saleh, K., Aboulhamid, E., En-Nouaary, A., Bourhfir, C.: Test Development For Communication Protocols: Towards Automation. Computer Networks, 31, pp. 1835--1872 (1999).
16. Dick, J., Faivre, A.: Automating the Generation and Sequencing of Test Case from Model-Based Specification. In: Proceedings of International Symposium on Formal Methods, pp. 268--284 (1992).
17. Vaysburg, B., Tahat, L., Korel, B.: Dependence Analysis in Reduction of Requirement Based Test Suites. In Proceedings of ACM International Symposium on Software Testing and Analysis, pp. 107--111 (2002).
18. Korel, B., Tahat, L., Vaysburg, B.: Model Based Regression Test Reduction Using Dependence Analysis: In: Proceeding of IEEE International Conf. on Software Maintenance, pp. 214--223 (2002).
19. Nieminen, J.: On centrality in a graph. Scandinavian Journal of Psychology 15, pp. 322--336 (1974).
20. Freeman, C.: A set of measures of centrality based on betweenness. Sociometry 40, pp.35--41 (1977).
21. Garey, M. R., Johnson, D. S.: Computers and intractability. A guide to the theory of NP-completeness. W. H. Freeman, San Francisco (1979).
22. Conallen, J.: Modeling web application architectures with UML. Communications of the ACM. vol. 42, no. 10. pp. 63--71 (1999).
23. de Alfaro, L.: Model checking the world wide web. In: Gerard Berry, Hubert Comon, and Alain Finkel, editors, Proceedings of the Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France. LNCS. vol. 2102. pp. 337--349. Springer (2001).
24. Alpuente, M., Ballis, D., Falaschi, M.: A rewriting-based framework for web sites verification. Electr. Notes Theor. Comput. Sci. vol. 124, no. 1. pp. 41--61 (2005).

25. Chen, J., Zhao, X.: Formal models for web navigations with session control and browser cache. In: ICFEM. pp. 46--60 (2004).
26. Bordbar, B., Anastasakis, K.: MDA and analysis of web applications. In: Dirk Draheim and Gerald Weber, editors, Proceedings of the Trends in Enterprise Application Architecture. LNCS. vol. 3888. pp. 44--55. Springer (2005).
27. Winckler, M., Palanque, P.: Statewebcharts: A formal description technique dedicated to navigation modelling of web applications. In: DSV-IS. pp. 61--76 (2003).
28. Han, M., Hofmeister, C.: Modeling and verification of adaptive navigation in web applications. In: ICWE. pp. 329--336 (2006).
29. Di Sciascio, E., Donini, F., Mongiello, M., Piscitelli, G.: Web applications design and maintenance using symbolic model checking. In: Proceedings of the European Conference on Software Maintenance and Reengineering, pp. 63--72. IEEE Computer Society (2003).
30. Castelluccia, D., Mongiello, M., Ruta, M., Totaro, R.: Waver: A model checking-based tool to verify web application design. Electr. Notes Theor. Comput. Sci.. vol. 157 no. 1. pp. 61--76 (2006).
31. Bellettini, C., Marchetto, A., Trentini, A.: Webuml: reverse engineering of web applications. In: SAC, pp. 1662--1669 (2004).
32. Wu, Y., Outt, J.: Modeling and testing web-based applications. Technical report, George Mason University (2002).
33. Syriani, J., Mansour, N.: Modeling web systems using SDL. In: Adnan Yazici and Cevat Sener, editors, Proceedings of the Computer and Information Sciences - ISCIS 2003, 18th International Symposium, Antalya, Turkey, November 3-5, 2003. LNCS. vol. 2869. pp. 1019--1026. Springer (2003).
34. Sabidussi,G.: The centrality index of a graph. Psychometrika, 31, 58–603 (1966).
35. Newman M.E.J.: Mathematics of networks. In The New Palgrave Encyclopedia of Economics, 2nd edition. Palgrave Macmillan, Basingstoke, (2007).
36. Freeman, C.: Centrality in social networks. 1. Conceptual clarification. Social Networks 1: 215-239, (1979).
37. Freeman, C.: The gatekeeper, pair-dependency and structural centrality. Quantity and Quality 14:585-592, (1980).
38. Ma, H. W., Zeng, A. P., The connectivity structure, giant strong component and centrality of metabolic networks. Bioinformatics, 19, 1423–1430, (2003).
39. Jeong, H., Mason, S. P., Barabasi, A. L., Oltvai, Z. N., Lethality and centrality in protein networks. Nature, 411, 41–42, (2001).
40. Hage, P. and Harary,F.: Exchange in Oceanea: A Graph Theoretic Analysis. Oxford: Clarendon Press (1991).