Towards Automatic Consistency Checking between Web Application and its Mobile Application

Xiangping Chen¹,²,³
¹Institute of Advanced Technology, Sun Yat-sen University, National Engineering Research Center of Digital Life, Guangzhou, 510006, China
²Shenzhen Engineering Laboratory for Mobile Internet Application Middleware Technology, Shenzhen, 518060, China
³Research Institute of Sun Yat-sen University in Shenzhen, Shenzhen, 518057, China
chenxp8@mail.sysu.edu.cn

Zhensheng Xu¹,²
¹School of Information Science and Technology, Sun Yat-sen University, National Engineering Research Center of Digital Life, Guangzhou, 510006, China
²Research Institute of Sun Yat-sen University in Shenzhen, Shenzhen, 518057, China
xuzhensh@mail2.sysu.edu.cn

Abstract—With the increasing usage of mobile devices in daily life, a lot of websites delivered another mobile application version of application with the same services. If the mobile application is inconsistent with the web application, it may imply possible error or misleading information in the mobile application. This paper proposes an approach for automatic consistency checking between mobile application and its web application. Our approach starts from the generation of page and link information by crawling content from web application and simulating user actions in Android application. Page matching and consistency checking algorithms are proposed and used to find out inconsistency in page, element and link. The validation tool is used in 3 mobile application containing more than 300 pages. The result shows that our approach can find out inconsistency with acceptable precision and recall.

Keywords—Consistency checking, Black-Box, Mobile Application

I. INTRODUCTION

With the increasing usage of mobile devices in daily life, the number of mobile applications increased very quickly. By July 2013, the number of applications in Google play has rose through 1 million¹.

With this trend, a lot of websites delivered another mobile application version with the same services. Almost all the top 500 global sites² provide its corresponding android application³.

Because the web application version and mobile application version are provided for the same target users, the functional requirements of the mobile application are already fixed before implementation. The requirement is usually the same as the web application, or subset of the requirements of the web application. In addition, these functionalities are required to be provided in the same way. For example, if a user can access the list of international news by clicking the element “News” in the homepage, it is improper to provide the list of international news through the “Information” element in the interface of mobile application. In addition, the list of international news should be the same.

For a web application containing hundreds and thousands pages, the implementation of its mobile application is time-consuming with a lot of repetitive work. There may rise inconsistency problem that the mobile application include: (1) error page that shows information which is not included in the web application; (2) error element that displays misleading information; (3) error link that is linked to the wrong page. Although, not all the inconsistencies exist between a mobile application and its web application are faults. For example, a mobile application may provide a special sale for mobile users only. However, inconsistency information is still useful to make sure that extra functionalities are added.

Current works[1-5] focus on consistency checking of different software models in an application. The problem in this paper is how to find out the consistencies between two implementations with the same design model/requirements. Some works proposed testing methods and tools[7-10] for Android applications. However, they do not notice the consistency problem.

In this paper, we propose an approach for automatic consistency checking between mobile application and its web application. We abstract both web application and mobile application as a set of pages and link relationships between pages. Our approach starts from the generation of page and link information by crawling content from web application and simulating user actions in Android application. We define the consistency relationship between two applications and propose page matching and consistency checking algorithm to find out inconsistencies in page, element and link. The tool is used in 3 mobile applications containing more than 300 pages. The result shows that our approach can find out inconsistencies with acceptable precision and recall. Our approach can help software...

²http://www.alexa.com/topsites
³https://play.google.com/store
engineers and maintainers in finding possible errors during development and maintenance.

The remainder of the paper is organized as follows: Section 2 gives an illustrative example to show inconsistency problem; Section 3 gives the definition of consistency and a general overview of our solution, the definition of consistency and a general overview of our solution, which is further detailed in the Section 4. Section 5 presents the evaluation of our approach; Section 6 discusses the limitations of our work, Finally, Section 7 discusses some related works before Section 8 concludes our work.

II. An Illustrative Example

For the OSChina website (http://www.oschina.net), its android application is delivered to provide the same functionality as its website. When we click the button with text “推荐阅读” in the main interface of mobile application, we can see the interface shown in Fig 1(a).

In the home page of the web application, we can find a list of articles, as shown in Fig 1(b). These articles are listed without detail information. In the mobile application, the author, the type of paper, publishing time and number of reviews of an article is listed in the interface.

Figure 1. corresponding pages in mobile application and web application

If we click “最新推荐博客文章” in the home page of the web application, we can see the same list as we see in the home page of OSChina. The related information for these articles is not provided in this page. The related information about an article is provided in different ways for mobile application users and web application users. The inconsistency problem may confuse or mislead users.

Figure 2. Corresponding page in the linked page

III. Approach Overview

A. The Consistency Requirement between Web Application and Mobile Application

A web application or mobile application can be abstracted as $AP = (P, L)$ where $P$ is the set of web pages, $L$ is the set of link between web pages. A page in $P$ includes a list of elements. The elements are divided into two types: elements with link and element without link. $P = (E_l, E_i)$. link$=\langle p, e, p_t \rangle$, $L_{link} \in L$, $e \in E$. A link is a link from a web page $p_t$ to another web page $p$, and related to an element $e$ in the web page $p_t$.

We define the consistency between a web application and its mobile application based on the following assumptions:

(1) There exist a web application and a mobile application. The mobile application is developed to provide the same or subset of the services provided by the web application.

(2) A page in the mobile application contains less information than a page in the web application. As a result, when implementing the mobile application based on the web application, the developer divides the contents in a web page into one or more pages in the mobile application.

(3) The mobile applications are implemented considering the user habit of web application. However, because a page in the web application may be implemented as more than one page in the mobile application, there are links between pages which are developed for the same page of web application. If these links are neglected, the traces of user action to acquire the same information in both mobile application and web application should be the same.

For a web application $AP_{web}= (P_{web}, L_{web})$ and a mobile application $AP_{mobile}= (P_{mobile}, L_{mobile})$:

(1) $\forall p_t \in P_{mobile}, \forall p \in P_{web}$

$p, e = p_t, e \rightarrow \text{description}(p, e) = \text{description}(p_t, e)$

Elements in the page of a web application are different from those in the page of a mobile application because they are developed using different programming language. Two elements of different kinds can be viewed as equal when
they are used to provide the same information or functionality. description(e) for the element e cannot be complete. For example, for an element which is a picture, it is hard to generate its description. The extracted description from the application may affect the accuracy for finding a possible consistent page.

(2) \( \forall p_i \in P_{\text{mobile}}, \forall p_j \in P_{\text{web}} \)

\[ \text{consistent}(p_i, p_j) \rightarrow p_i, E_i \in p_j, E_j \land p_e, E_e \in p_j, E_e \]

We assume that a page in a mobile application contains less information and functionality than its corresponding page in the web application version. A page \( p_i \) in a mobile application is consistent with a page \( p_j \) in its web application when all the elements included in \( p_i \) are also included in \( p_j \).

(3) \( \forall l_i \in L_{\text{mobile}}, \forall l_j \in L_{\text{web}} \)

\[ \text{consistent}(l_i, l_j) \rightarrow \\
\text{consistent}(l_i, p_i, l_j, p_j) \land l_i, e \land \text{consistent}(l_i, p_i, l_j, p_j) \]

A page in a web application may consist of multiple pages in the mobile application version. We define that a link \( l_i \) in a mobile application is consistent with a link \( l_j \) in a web application when (1) their source pages are consistent; (2) the target pages are consistent; (3) the related elements are equal.

The consistency relationship between \( AP_{\text{web}} \) and \( AP_{\text{mobile}} \) is defined as:

(1) \( \forall p_i \in P_{\text{mobile}}, \exists p_j \in P_{\text{web}}, \text{consistent}(p_i, p_j) \)

(2) \( \forall l_i \in L_{\text{mobile}}, (\exists l_j \in L_{\text{web}}, \text{consistent}(l_i, l_j)) \lor \\
(\exists p_j \in P_{\text{web}}, \text{consistent}(l_i, p_i, l_j, p_j) \land \text{consistent}(l_i, p_i, l_j, p_j)) \)

For each page \( p_i \), there exists a corresponding consistent page \( p_j \) in \( P_{\text{web}} \). For each \( l_i \) in \( L_{\text{mobile}} \), if \( l_i \)’s source and target pages are consistent with the same page in \( AP_{\text{web}} \), there exists a corresponding consistent link \( l_j \) in \( L_{\text{web}} \).

It should be noted that inconsistency between web application and mobile application do not imply an error in the implementation. A lot of mobile applications provide specific functionalities for mobile user only. However, the inconsistencies are worth of notice during development and evolitional process for software engineers.

B. Approach Overview

Because the implementations of an application in different platforms are different, the first step of our approach is to generate the description of the mobile application and web application using the same model. The description includes page and link information.

Considering consistency requirement in implementing a mobile application for an existing web application, inconsistency may occur at the page level and the element level. Our approach first searches all possible pages in web applications to find out a possible consistent page for a page in mobile application. Because the link is related to an element in the page, the consistency in the element level considers both elements with and without a link. Our approach compares the elements and the links in sequence.

IV. CONSISTENCY CHECKING

A. Generating Page and Link Information

In our approach, we view the application as a black-box and simulate user actions to generate the page and link information. The generation starts from the homepage or main interface of the application.

For each page, we generate descriptions for its elements. Because the descriptions are used in the following comparison, they should be described in the same way. Text and image are two of the most important information for user. In real application, the pictures used in web applications and mobile applications are usually different because they are used for different size of displayers. For the image information, understanding and generating the description of a picture is too hard. As a result, we only use text information as the description of page elements. For an element in page \( p \), it is recorded using the page id, the text information, and its related link. Text and link can be left blank if the element does not contain text information or it is an element without a link. Element with no text and link information will be deleted because the comparison is meaningless.

The required input for next step, searching possible corresponding page, is a page and all its related pages. As a result, width-first search algorithm is better when exploring the pages. When exploring the pages, our approach records the related elements and the ids of source pages and target pages for links. The search omitted a link when the link’s target page is not included in the application or it is already explored before. The search stops when all the pages are explored or the search depth reaches a pre-defined value.

Because the programming language for web applications and mobile applications are different, they have different types of components in the pages. Our approach employs existing parser/tool to extract information from the application.
For the web application, our algorithm traverses pages from the home page and uses a Java HTML parser Jsoup[11] to extract information from each page. Jsoup provides APIs for fetching web page and extracting the text and link information from a page. Because a page in the web application is standalone, we can traverse the pages by using a width-first search. During information extraction, we do not deal with dynamic content which is generated during execution of Javascript. These complex situations will be considered in our future work.

For an android application, our approach simulates user actions to traverse all possible pages. Because a page (interface) in the android application is not a standalone page, fetching a page require the simulation of user actions to access this page from main interface. If the traverse is carried out using a width-first search, all the user actions simulated for a page will be repeated carried out when traversing its related pages. As a result, our algorithm traverses page in Android application using the depth-first search.

We employs an Android test automation framework Robotium [12] to carry out the simulated user actions. Robotium is an Android test automation framework that support for native and hybrid applications. In real application, traversing pages in Android application is time-consuming. For example, we extract 90 pages from an Android application in about 1 hour. The max depth for the depth first search can be changed according to how complex the application is.

B. Page Consistency Checking

Based on the extracted page and link information, our approach traverses all the pages in the mobile application and finds out their corresponding pages. For a page $p_i \in P_{mobile}$, its corresponding page is a page $p_j \in P_{web}$, when the possibility for consistency($p_i, p_j$) is higher than the possibility of consistency($p_i, p_j'$), $p_j' \in P_{net}$ and $p_j' \neq p_i$.

We suppose that the corresponding page of the main interface of a mobile application is the main page of the web application. Page matching starts from finding corresponding pages linked from the main interface. Because we omitted link whose target page is explored before. For a page $p_i$, there is only one link whose target page is $p_i$.

For pages $p_i, p_i' \in P_{mobile}$, $p_j \in P_{web}$, if $\exists l \in L_{mobile}, l.p_i = p_i \land l.p_i' = p_i'$, corresponding($p_i$) = $p_p$, corresponding($p_i'$) = $\{p_j \} \cup \{p\}$, $p$ = $e$.link.$p_i$, then $e \in p_j$, $E_i \land p \in P_{web}$.

The matching is carried out in the set of all possible matched pages $P_{possible} = \{p \mid p = e$.link.$p_i \land e \in p_j, E_i \land p \in P_{web}\}$. For $p_i' \in P_{possible}$, $e \in p_i$:

1. $\text{matched}(e, p_i') = \begin{cases} 1, & \exists e' \in p_i', e.text = e'.text \\ 0, & \forall e' \in p_i', e.text \neq e'.text \end{cases}$

2. $\text{matched}(p_i', p_j') = \sum_{e \in p_i'} \text{matched}(e, p_j'), e \in p_i'$

Based on the number of matched elements, we can find the corresponding page by comparing text information.

corresponding($p_i'$) = \{ $p \mid \text{matched}(p_i, p) \geq \text{matched}(p_i, p_i')$ \}

$p \in P_{possible} \land p_i' \in P_{possible} \land p \neq p_i' \land \text{matched}(p_i, p) \neq 0$\}

If the number of matched elements between pages $p_i'$ and $p_j'$ are all zero, there is no corresponding page for $p_i'$.

C. Element and Link Consistency Checking

We divided the inconsistencies into three types. For a page $p$ in mobile application, there is :

1. Page inconsistency: no corresponding page corresponding($p$) = $\emptyset$

2. Element inconsistency
   $\exists e, (e \in p.E_e \land e \notin \text{corresponding}(p).E_e)$

3. Link inconsistency
   $\exists l, l.p_i = p \land e \notin \text{corresponding}(p).E_i \land \text{corresponding}(l.p_i) \neq \text{corresponding}(p)$

The inconsistent pages can be found in the page consistency checking process. An inconsistent page includes information and functionalities which are not included in the web application. Developers can use this result to see if mobile app specific functionalities are implemented, and analyze the degree of coupling between these new functionalities and existing functionalities.

Element inconsistencies are recorded in the page consistency checking stage when the algorithm tried to compute the number of matched elements. An element inconsistency indicates possible differences in the way to provide information in different applications. The example introduced in section 2 is a classic example of element inconsistency. In some cases, the developer may use image and text to show the same information in different applications. This may bring some undetectable consistency relationship and reduce the accuracy of our method.

Link inconsistency is validated based on the corresponding relationship between pages. Because there may exist more than one page in the mobile application with the same corresponding page in the web application, the links between these pages are not matched to a corresponding link in the web application. The checking filters this kind of links. Other inconsistent links are caused by link inconsistent.

V. EXPERIMENTS

We use our consistency tool to validate the following android applications with their web applications. We choose three application in our experiment. Because we view both web application and Android application as black-boxes, the first step of our experiment is to find out the website address and download the Android application package file (APK).

**TABLE I.** EXPERIMENT PROJECTS

<table>
<thead>
<tr>
<th>Name</th>
<th>Website address</th>
<th>Android app Size (M)</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>oschina</td>
<td><a href="http://www.oschina.net">http://www.oschina.net</a></td>
<td>2.2</td>
<td>1.7,6.5</td>
</tr>
<tr>
<td>zol</td>
<td><a href="http://www.zol.com.cn">http://www.zol.com.cn</a></td>
<td>7.2</td>
<td>3.5,2</td>
</tr>
<tr>
<td>chinadaily</td>
<td><a href="http://www.chinadaily.com.cn">http://www.chinadaily.com.cn</a></td>
<td>3.3</td>
<td>3.2,0</td>
</tr>
</tbody>
</table>
Because the generation of page and link information from the android application is time-consuming, we pre-defined the depth during explosion. Table II shows the page size and execution time in extracting page and link information. The pre-defined depth for explosion of Android application is 3.

**TABLE II. THE SIZE AND EXECUTION TIME OF GENERATION**

<table>
<thead>
<tr>
<th>Project name</th>
<th>Number of pages (web app)</th>
<th>Execution Time</th>
<th>Number of pages (Android app)</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>oschina</td>
<td>4070</td>
<td>23min</td>
<td>145</td>
<td>50min</td>
</tr>
<tr>
<td>Zol</td>
<td>12430</td>
<td>107min</td>
<td>116</td>
<td>31min</td>
</tr>
<tr>
<td>chinadaily</td>
<td>1727</td>
<td>23min</td>
<td>104</td>
<td>36min</td>
</tr>
</tbody>
</table>

The extracted page and link information is saved in a tree structure. Fig 4(a) shows part of the generated information of the main interface of Android application, and Fig 4(b) shows part of the generated information of the home page of web application in the project of oschina.

(a) Android app  (b) web app

![Example of generated information](image)

The precision in checking inconsistencies between pages is relatively low. Because when our algorithm finds a page without corresponding page, it stops matching all its related pages. A page inconsistency usually involves more than one page. The recall of page inconsistency is caused by the matching of some meaningless elements in the page, for example, logo in the page.

**VI. DISCUSSION AND LIMITATION**

During the experiment, we found some important problems that have not yet resolved in this paper.

First, for the application running in the internet, the content in both web application and mobile application is changing all the time. The changes of content during the process of information extraction bring inconsistency problem. Our approach will generate confusing result if the application is updating during consistency checking.

Second, our approach employs a testing tool Robotium[12] to extract page and link information from Android application, as introduced in section 4. In the implementation, our tool supports extraction of information for pre-defined and simple user-defined view types. User-defined types are now widely used in the implementation of Android application, our approach cannot work well in complicated user-defined type.

**VII. RELATED WORK**

Consistency checking is widely used in the field of model-driven software engineering and development. Inconsistencies between multiple views in the same phase or different phases of software lifecycle, may bring follow-on errors and unnecessary rework.

Most works [1][2][3] focus on detecting consistency for UML models because UML is widely used and the consistency of models affects the quality of generated code. These works focus on both intra and inter level consistencies. Some works are proposed considering different aspects in consistency validation. Blanc[4] proposed a meta-model independent consistency validation method. In his approach, model is represented as sequences of elementary construction operations, structural and methodological consistency rules can then be expressed uniformly as logical constraints on such sequences. Egyed’s work [5] focus on an automated approach for detecting and tracking inconsistencies in real time. The performance of his method is not noticeably affected by the model size and common consistency rules but only by the number of consistency rules. Vierhauser et al.[6] proposed an approach for incremental consistency checking for product lines. It can be used to check the consistency of variability models and the consistency of the models with the underlying code base.

To the best of our knowledge, there is not an existing method for consistency checking between a mobile
application and its corresponding web application. Most works focus on the testing of mobile applications, which provide basis for our approach.

Hu et al.[7] present an approach for automating the testing process for Android applications, with a focus on GUI bugs. The approach detects GUI bugs by automatic generation of test cases, feeding the application random events, instrumenting the VM, producing log/trace files and analyzing them post-run. Azim et al. [8] propose two strategies: targeted exploration and depth-first exploration, for the automatic exploration of the Android applications so as to improve the coverage at two granularity levels: activity (high-level) and method (low-level). Amalfitano et al. [9] proposed an automated technique that tests Android apps via their Graphical User Interface (GUI). Its tool Android Ripper[10] is based on a user-interface driven ripper that automatically explores the application’s GUI with the aim of exercising the application in a structured manner. In addition, there are useful testing tools for Android application. Troyd[11] allows developers to write Ruby scripts that can drive the Android application. Robotium[12] is an Android test automation framework that support for native and hybrid applications. Our implementation of consistency validation tool is built based on Robotium.

VIII. CONCLUSION

In this paper, we propose our approach based on the inconsistency problem in developing and using mobile application in daily life. This paper proposes an approach for automatic checking consistency between web application and its mobile application. Our approach starts from the generation of page and link information by crawling content from web application and simulating user actions in Android application. Page matching and consistency checking algorithms are proposed and used to find out inconsistency in page, element and link. Because the number of inconsistency found in our experiment is too large for comprehension. In the future work, we will try to define inconsistency pattern based on mining the inconsistency data. Repetitive inconsistency simplified as a pattern will improve the usability of our approach for real software development.

ACKNOWLEDGMENT

This effort is supported by National Natural Science Foundation of China(No. 61100002), the NSFC Guangdong Joint Fund (No. U1201252), Shenzhen Technology R&D Program for Basic Research(No.JC201105201042A), the Special Foundation for Development of Internet Industry of Shenzhen(No. 20120731105450), the Fundamental Research Funds for the Central Universities, Science and Technology Planning Project of Guangdong Province (No.2012B010900089).

REFERENCES