Poznan University of Technology

Towards semi-automatic size measurement of user interfaces in web applications with IFPUG SNAP

Project Supervisor

Dr. Mirosław Ochodek

Submitted by

Hassan Mansoor

Submission Date

September 30th , 2016

The Department of Informatics
Poznan University of Technology
Institute of Computing Science
ul. Piotrowo 2, 60–965 Pozna´n, Poland

Acknowledgements

The authors of this report would like to express their deepest appreciation and thanks, first off to Allah the Almighty for all the blessings He bestowed upon us and for making us capable enough to do this project.

I express my gratitude to Poznan University of Technology, Poland for giving us this opportunity to conduct and execute a project of this magnitude and for letting us getting a first-hand experience and taste of what it is like to work on a real time project.

I also thank my project supervisor, Dr. Mirslow Ochodek, for his valuable guidance and displaying great patience while working with me. He motivated me tremendously to work on the project and in spite of difficulties kept pushing me to do my best. If it wasn't for his constant help, the project would never have been completed on time.

Lastly we would like to thank our families, our parents and our elders for supporting us morally and spiritually during the course of this project.

Document Information

Category	Information
Customer	Poznan University of Technology
Project Title	Towards semi-automatic size measurement of user interfaces in web applications with IFPUG SNAP
Document	Master's Thesis
Document Version	1.0
Identifier	Thesis Document
Status	Final
Author(s)	Hassan Mansoor
Approver(s)	Dr.Mirslow Ochodek
Issue Date	

Definition of Terms, Acronyms and Abbreviations

Term	Description

Table of Contents

CHAPTER 1

CONTEXT AND PRELIMINARY INVESTIGATION

1. Introduction

Software size measures are probably the most frequently used measures in software development projects. They are, for instance, used as independent variables in many models of effort estimation, benchmarking and pricing. They also constitute a natural normalizing factor for many indirect measures (e.g., defect density).

The International Function Point Users Group (IFPUG) has developed two complementary software measurement methods. The first one is called Function Point Analysis (FPA) [1] and is probably the most recognized functional size measurement (FSM) method [10]. The second one is a non- functional size measurement method called Software Non-functional Assessment Process (SNAP)[8]. Although, the latter is an emerging method, its initial beta tests performed in August and early September 2012 gave promising results [9]. Both of the methods seem complementary and could be used standalone or conjointly [2].

The SNAP method allows to measure different types of non-functional requirements. Its model consists of four main categories and fourteen subcategories. Each of them corresponds to different aspects addressed by non-functional requirements. In the thesis, the focus is on the SNAP's sub-category 2.1: User Interfaces (UI). It regards ``unique, user identifiable, independent graphical user interface elements added or configured on the user interface that do not change the functionality of the system but affect non-functional characteristics (such as usability, ease of learning, attractiveness, accessibility)."[8]. To measure the size of an NFR related to usability one has to identify all unique UI Elements and their properties that have to be configured to meet requirements. Unfortunately, applying the method to measure size of modern user interfaces could be a cumbersome task, especially when taking into account their complexity. Another problem emerges when SNAP is used for pricing software development projects because it might be difficult to achieve transparency in tracking which UI Elements were configured to meet certain non-functional requirements. Therefore, in this context, the following problem seems worth of investigating:

Core Problem: *Given a set of screens defined by an elementary process, provide a list of the properties of UI Elements that were configured to meet non-functional requirements.*

1.1 Aims and Scope

In this thesis Core Problem is considered in the context of web application development. According to Forrester Research [3] web technologies are becoming increasingly critical to organizations' web and mobile application strategies. They reported that 27\% of the surveyed organizations plan to move away from native development in favor of web technology (HTML5, JavaScript, and CSS). Taking the above into consideration, the aim of the study is *to propose a semi-automated approach to measure size of non-functional requirements associated with the SNAP sub-category 2.1. The method shall accept the input consisting of a set of exemplary screens of an application (HTML and CSS) and rules for mapping HTML elements into UI Elements recognized by the users. It shall return a list of UI Elements and properties that were configured.* It is assumed that based on the provided output, the user of the method might decide which of the properties should be included in the SNAP measurement.

Remark: The thesis contains some excerpts from the paper entitled "Towards semi-automatic size measurement of user interfaces in web applications with IFPUG SNAP" [7] co-authored by the author of the thesis and his supervisor. The paper was presented at the IWSM Mensura 2016 conference.

The thesis is organized as follows:
Chapter II describes the detail background about IFPUG SNAP and the how this method works, furthermore it will describe SNAP subcategory 2.1 and SNAP counts in this category with the help of example and a brief overview about HTML and CSS technologies.
Chapter III consists of our proposed approach and its working, this chapter also describes set of input that our tool is required for processing and the different types of rules on the set of the input and expected output.
Chapter IV discusses functional and nonfunctional requirements of the prototype tool and its architecture is described by different technical diagrams; furthermore it discusses the implementation and maintenance of the tool.
Chapter V shows the results of preliminary validation of the proposed approach and the prototype tool. It discusses the study design in detail and the obtained results from the tool.
Chapter VI concludes the obtained results.

The thesis is supplemented with the following appendices: (A) CD-ROM with the digital version of the thesis and source code of the prototype tool, (B) a preprint of the research paper co-authored by the thesis candidate and his supervisor that was presented at the IWSM Mensura 2016 conference [7].

CHAPTER II

BACKGROUND

2.1 IFPUG SNAP in a nutshell

SNAP is the acronym for "Software Non-functional Assessment Process," a measurement of non-functional software size. SNAP point sizing is a complement to a function point sizing, which measures functional software size. In the late-1970's, Allan Albrecht of IBM introduced one of the most well recognized sizing methods, which is called Function Point Analysis (FPA) [2, 4]. FPA measures the functionality which was requested and received by the user, independent of the technical details involved. Since 1986, the International Function Point Users Group (IFPUG) has continued to maintain Albrecht's method, which has become a de facto industrial standard.

In 2008 IFPUG starts to work on a framework called SNAP to measure the non functional aspects of the software system. The main objective was to ensure that non functional framework can be used to establish a link between a non functional size and the effort to provide non functional requirements. The first version of the framework was released in 2010, later on in 2013 IFPUG releases beta version of the framework which gives the promising results.

The SNAP model consists of four categories and fourteen sub-categories to measure the non-functional requirements. These non-functional requirements are then mapped to the relevant sub-categories. Each subcategory is sized, and the size of a requirement is the sum of the sizes of its sub-categories. These categories and subcategories are listed in Table 2.1:

1. Data Operations	***2. Interface Design***
1.1. Data Entry Validations	**2.1. User Interfaces**
1.2. Logical and Mathematical Operations	**2.2. Help Methods**
1.3. Data Formatting	**2.3. Multiple Input Methods**
1.4. Internal Data Movements	**2.4. Multiple Output Formats**
1.5. Delivering Added Value to Users by Data Configuration	
3. Technical Environment	**4. Architecture**
3.1. Multiple Platform	**4.1. Component Based Software**

3.2. Database Technology **3.3. Batch Processes**	**4.2. Multiple Input / Output interfaces**

Table 2.1 IFPUG SNAP's categories and subcategories (adapted from [1]).

The IFPUG SNAP process to assess nonfunctional requirements consists of six steps, as presented in Figure 2.1.

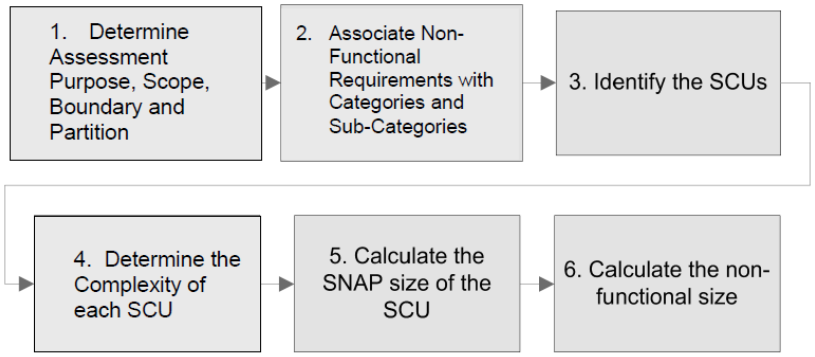

Figure 2.1 SNAP Diagram (adapted from [1])

In the first step, the assessment purpose, scope, boundary and partition of the application are determined. The assessment purpose explains why the measurement is going to be performed (e.g., effort estimation, pricing, quality assessment). It also determines the scope of measurement project (i.e., a set of non-functional requirements to be measured). Finally, the boundary needs to be defined by identifying what is external to application and what is the border between the user and the application. In step 2, all the non-functional requirements in the scope are associated to the categories and subcategories of the SNAP model based on their technical implementations. In step 3, for each of the subcategories of SNAP one has to identify SNAP Counting Units (SCU). The sizing is done separately per each SCU. In step 4, one has to determine the complexity of each SCU associated to a given subcategory of SNAP by answering the assessment questions specific to the subcategory. These questions are related to the parameters that affect the complexity of requirements. The assessment rating is the answer to these questions and the values within each SCU are mapped to the non-functional size. In step 5, after all the complexity parameters have been provided, the size of each SCU, expressed in SNAP Points (SP), is calculated. Then the size of all SCUs are added together to obtain the calculated SP for the subcategory. Finally, in the step 6, we calculate the non-functional size of all requirements in the scope. The size is the total non-functional size obtained by summing the size contributed by all the subcategories.

2.2 Measuring size of UI-related non-functional requirements

The SNAP subcategory 2.1 is related to end-user experience. It covers the design of user interface process and the methods that allows the user to interact with the application ('unique, user identifiable, independent graphical user interface elements added or configured on the user interface that do not change the functionality of the system but affect non-functional characteristics (such as usability, ease of learning, attractiveness, accessibility).' [1]

To measure the size of category 2.1 the user has to provide a set of screens with all the properties of the UI Element that needs to be configured for a given elementary process. Elementary process is defined as "the smallest unit of activity that is meaningful to the user, which must be self-contained and leaves the business of the application being counted in a consistent state" [3]. The unique properties of each unique UI Element are counted and sized based on Table 2.2.

	UI Type Complexity		
	Low	**Average**	**High**
	<10 Properties added or Configured	**10-15 Properties added or Configured**	**16+ Properties added or Configured**
SP =	**2*# unique UI elements**	**3*# unique UI elements**	**4*# unique UI elements**

Table 2.2 SNAP sizing for user interfaces (adapted from [3])

The size measurement can be illustrated by an example use case below [3]:

Example: Assume that there is a text "Partner" on the screen that is hard coded and few times later the company introduced a new policy that the text should be replaced with "Business Partner".

The analysis found that the word "Partners" should be replaced in the following UI elements (Note: since SNAP counts the number of unique UI elements, there is no need to estimate the number of occurrences of each unique UI element)

SCU 1: Get a new text Business Partners:

Header, labels, radio button, drop-down list

SCU 2: Modify the details of the text:

Header, labels

SCU 3: Send message to a new Business Partner:

Header, labels,

SCU 4: End services to a Business Partner:

Header, labels

Changing the text in these UI elements is considered one property. UI type complexity is Low

SP = 2*# unique UI elements per each SCU:

SP = 2*(4 + 2 + 2 + 2) = 20 SP

2.3 Basic technologies for implementing web-application user interfaces

2.3.1 HTML (HyperText Markup Language)

HyperText Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading (CSS), and JavaScript, it forms the keystone technologies for the websites World Wide Web (WWW). In 1991[6] the first publically html document was available on internet that was composed 18 html elements called “HTML Tags”. It is a language for web browsers that can transform these html tags to visual elements for the user interaction and these visuals can be enhanced by the use of style sheets CSS.

HTML elements are the building blocks of the HTML pages these pages composed of HTML tags each tag is defines by “<tag name> “ and a closing tag “</tag name>” where tag name is the html element name. Each html tag has attributes that are its properties defined inside the tags to enhance the elements. With the growth of software development more and more IT organizations are moving towards web applications and are widely used from past few years.

HTML 5 is the newer version of HTML released in October with additional features and supports. With HTML 5 the application and website can be developed that functions like desktop application. The user does not need to download and install apps for multiple devices. Another aspect of HTML 5 is the applications allows the user to use some functions offline or the system is not connected. This feature is introduced to make the application consistent and the user can access the data locally if system is offline or not connected. With HTML5 you can make use of variety of UI elements such as animation, games, movies etc. Graphical effects such as 3D, vector graphics etc are also

supported by HTML5. The major advantage with these applications is that JavaScript engine run faster these applications in real time. Several new tags have been added for example <audio> for audio tag, <video> for video tag etc.

Web pages build using HTML are formed by hieratical HTML DOM structure which is parent nodes, child nodes and their attributes. The hieratical structure is used to form elementary processes, the structure also gives flexibility to get the Xpath (a query language to select nodes) to get the DOM of specified nodes. An example of HTML DOM with Xpath is given below:

<html> ----Root node

 ---Parent node

<input type=”submit” /> ---Child node

</html>

2.3.2 CSS (Cascading Style Sheets)

CSS stands for Cascading Style Sheets. It is designed to describe how the HTML elements should be displayed on the screens to the user or in other media. It is often used to design visual styles on the webpage written in HTML and XHTML. CSS is independent of HTML and it can be used with any XM- markup language to style the elements.

The primary purpose of designing CSS is to provide the separation from markup language document so that the elements can de styles independently such as layout, fonts, color etc. This separation improves the accessibility and provides more flexibility in the specification of presentation characteristics [5]. This separation also made it possible to style different methods for single markup document in different contexts. It also helps to present the elements differently on different screens resolutions for better understanding. Bootstrap is the famous framework used these days to design elements. It used to design visual styles in better way and make the screens responsive screens to display visuals. Bootstrap introduces different styling properties that help to achieve complex design faster and easier. The sample code of CSS is shown below:

```
Body {
    Background-color: white;
   Color: black;
```

}

Combining these two technologies HTML and CSS together results strong flexible and interactive interfaces for web application that conforms to the principles of human computer interaction (HCI) and Usability. Day by day new features are added to these technologies that boost the usage of web applications to provide interactive and robust web interfaces.

CHAPTER III

A PROPOSED METHOD

3.1 Overview of the proposed method

An overview of the proposed method of identifying UI Elements and properties that were configured based on analysis of HTML pages and corresponding CSS files. The overview of the proposed procedure is presented in Figure 3.1.

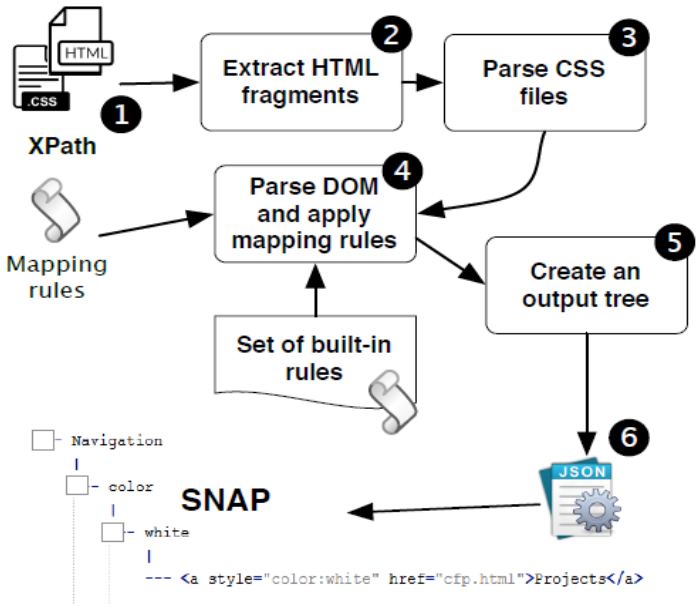

Figure 3.1 The overview of the steps of proposed method

For each elementary process that is affected by NFRs, the method takes a set of HTML files with XPaths defining the screens accompanied by the CSS files (1). Then, the HTML page is parsed to extract all the HTML tags on the provided XPaths (2). In the next step, CSS files are processed to identify all the CSS properties of the extracted HTML tags (3). Then, the DOM tree of the extracted HTML fragments is traversed and the defined mapping rules are applied (4). These mapping rules provides mapping between the HTML tags and UI Elements (as perceived by the users). After applying the rules, all the properties of the uniquely identified elements are collected (5) and used to provide the output in the form of JSON file (6).

3.2 Input

To method required input to process the set of screens and the input consist of few things that are necessary to send the tool for process and calculating the size.

- The first and most essential input is set of HTML screens, these screens can be either generated by the target application or the prototypes or designed screens.
- The second input is expected to be CSS files that are created to meet the non-functional requirements, the CSS files that we are referring here are custom designed style sheets that are used for the target application. The user can also provide the other CSS file for example bootstrap CSS the most popular and widely used style sheets if it needs to be included in the measurement.
- Last but not the least input that is supposed to provide is set of Xpath expressions that defines the fragment of the page as screens. As with growing HTML technology the web page become more complex and on HTML page can contains many Elementary processes, by using Xpath the tool will only extract the all the relevant nodes of the target screens and perform further processing.
- Apart from these three essential inputs there can be one more optional input that is the set of custom mapping rules between the HTML tags and UI Elements and a set of Custom UI properties names.

3.3 UI Elements mapping Rules

In order to apply the rules of 2.1 SNAP subcategory we have to be able to map between HTML tags which are used by the browsers to render the view for the user and UI Elements which are the UI Elements recognized by the users.

As a result of the investigation, three types of such mapping rules between HTML tags and UI Elements were proposed. While traversing through the DOM tree, the rules are applied for each HTML tag one by one until any of them is satisfied. Therefore, the rules should be ordered from the most specific to more general ones.

R1: "if a HTML tag is Y then UI Element is Z"

The rule R1 says that whenever there is a tag <Y> it should be classified as UI Element of type Z. For example we have an HTML tag <a> which is recognized as a hyperlink in HTML it could be also treated as UI Element "Hyperlink" in the user's domain.

R2: "if a HTML tag is Y AND has attributes [attr1=value1 AND . . . AND attrn=valuen] then UI Element is Z"

The rule R2 is more expressive than the rule R1. It says that if there is a HTML tag <Y> and it has certain attributes set to given values than it is treated as UI Element Z. The rule becomes useful if the name of HTML tag is insufficient to map it to a UI Element. For instance, to identify HTML tags that represent UI Element Button we would have to define at least two rules:

- R1-type rule: "if a HTML tag is BUTTON then UI Element is Button"
- R2-type rule: "if a HTML tag is INPUT AND has attributes [type=submit] then UI Element is Button"

We might also extend the set of rules with custom rules. For instance, let assume that we implemented a custom JavaScript code and defined CSS class called "button" that makes a DIV tag appearing and working like a Button. To treat such a DIV as Button, we could formulate the following rule:"if a HTML tag is DIV AND has attributes [class=button] then UI Element is Button."

R3: "if a HTML tag is Y AND is inside a UI Element X OR HTML tag W then UI Element is Z"

The rule R3 says that if there is an HTML tag named Y that is a parent node and inside this node there is a child element that can be a HTML tag or an UI Elements W then this UI Element is named as Z. For instance a UI Element Drop-down implemented as an HTML tag *SELECT* and its child node implemented as *OPTION* tag and assume that by configuring any property of either SELECT tag or OPTION tag results change in the whole UI Element, then according to the rule R3 the whole element will be treated as *Drop-down* that is the UI Element.

3.4 UI Properties mapping rule

In addition to HTML-UI Elements mappings, it is required to distinguish HTML and CSS properties that relate to UI from the properties used for different purposes. Therefore, another input of the proposed method is the list of UI-related HTML and CSS properties. The default list is based on the set of HTML tags attributes and CSS properties provided by W3C consortium (W3schools), (tags). The list can be easily extended to incorporate custom properties.

3.4 Output

The output generated based on the processing of HTML pages has a tree structure with four levels:

→UI Element

→UI Property

→Values

→HTML Fragments

At the first level there are UI Elements. In the second level there are UI Properties that were configured for a given UI Element and their values are placed at the third level. At the fourth level some excerpts from HTML page are provided showing the exact place in the HTML code where the UI Elements were configured. The default output format is JSON so it can be easily processed by external tools.

CHAPTER IV

A PROTOTYPE TOOL

In order to be able to validate the proposed method, a prototype tool was designed and implemented.

4.1 FUNCTIONAL REQUIREMENTS

We assume that there is a single actor called **Measurer**. It is any person that performs SNAP measurement. Such actor has a goal: to measure the non-functional size of requirements related to usability for a given SCU (elementary process). The full documentation of use case that allows obtaining the goal is presented in Table 4.1, while the use case diagram is presented in Figure 4.1.

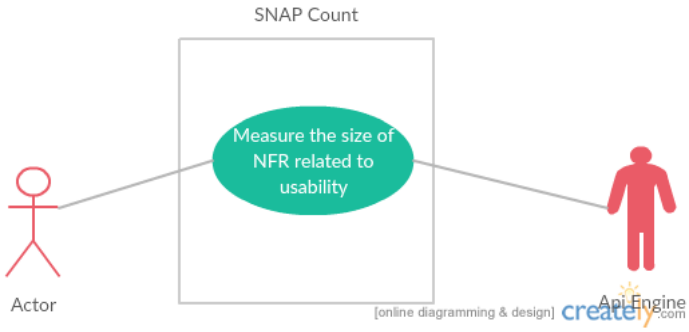

Figure 4.1 Use case diagram for the prototype tool (own work).

ID:	UC1
Name: Measure the size of NFR related to usability	
Main actors: Measurer **Additional actors:** System - the prototype tool **Level:** User **Priority:** High	
Description: The use cases tell how a Measurer can measure the non-functional size of screens defined by a single elementary process.	
Trigger: The Measurer wants to measure the size of NFR and understand which UI Elements and their properties were configured.	
Pre-condition: The Measurer collected HTML and CSS files of the screens defined by a given elementary process of the target web application	
Post-condition: The SNAP size and the tree-like structure is returned to the Measurer.	
Main Scenario: 1. Measurer chooses the option to measure size of UI-related NFRs for an elementary process. 2. System asks the Measurer to provide details about the elementary process and screens. 3. Measurer provides general information about the elementary process.	

4. Measurer provides a set of HTML pages (with XPaths defining the screens) and CSS files for a given elementary process. 5. System provides information about the UI Elements and UI Properties that were configured together with the calculated SNAP size. 6. Use case finishes successfully.
Alternative Scenario and extensions: **4. A** Measurer would like to change the mapping rules. **4. A.1** Measurer provides custom mapping rules. **4. A.2** Go to the step 5.
Exceptions: **5.A** User provide invalid HTML or CSS files. 5. A.l System informs about the wrong format of the files. 5.A.2 Use case end unsuccessfully. **5.A** User gives invalid XPath expression 5.A.1 Use case end unsuccessfully
Additional requirements:

Table 4.1. Use case showing how Measurer can measure the size of NFRs related to usability with the use of the prototype tool.

4.2 NON FUNCTIONAL REQUIREMENTS

The purpose of the software application is to illustrate the proposed approach (be a proof of concept). Therefore, only few non-functional requirements were considered:

- NFR1: The time of processing a HTML and CSS file should not exceed 30 seconds in 90% of cases (we assume that there could be some large web pages for which processing time could be longer).
- NFR2: The generated report should have a form of an interactive web page allowing re-calculating on-line the SNAP size based on the choice of UI Elements and UI Properties.
- NFR3: The non-functional size is calculations are convergent with the SNAP manual (the results is an integer).
- NFR4: The user is able to add custom mappings between HTML tags and UI Elements.

4.3 Architecture

4.3.1 Approach Analysis

The prototype tool has a form of web application. The decision regarding its architecture was preceded with the SWOT analysis presented in Table 4.2.

	Helpful	Harmful
Internal Origin	**Strengths** • Saves times to manually calculate NFR. • Easy to use • High accuracy • Automated process • Custom mapping rules can be added	**Weakness** • Tool is based on HTML and CSS • The modern web pages have JavaScript's the tool currently does not support. • Few rules
External Origin	**Opportunities** • Provide transparency of requirements between customer and user • Estimate the cost of non-functional requirements more clearly	**Threats** • The target elementary process consist of JavaScript • Users do not want to use it.

Table 4.2. SWOT analysis of the approach.

4.3.2 Decision Matrix

Table 4.3 shows the most important architectural decisions made while implementing the prototype tool compared with the skill levels of the author. The chosen set of technologies seemed to match the capabilities of the potential developer.

Skill Level	1	2	3	4	5	6	7	8	9	10
Software /Hardware Installation								*		
Software/Hardware maintenance								*		
OOP concepts and programming										*
.NET									*	
XML								*		
Json									*	
HTML							*			
CSS							*			

4.3.3 Sequence Diagram

The interaction between the objects and the tool is represented in the form of UML sequence diagram in Figure 4.2. The mechanism of passing the messages are between three objects user, web interface and API that is the parsing engine. The user object represents the end user of the application, the web interface is the graphical user interface that is used by the user to send the messages to the tool and API is our parsing engine the process the messages and return the json to the web interface. The messages are constructed in a sequential way the API object is active to the session until it sends the generated JSON to the web interface. All the messages are sent and received through web interface for the ease of user readability and understanding.

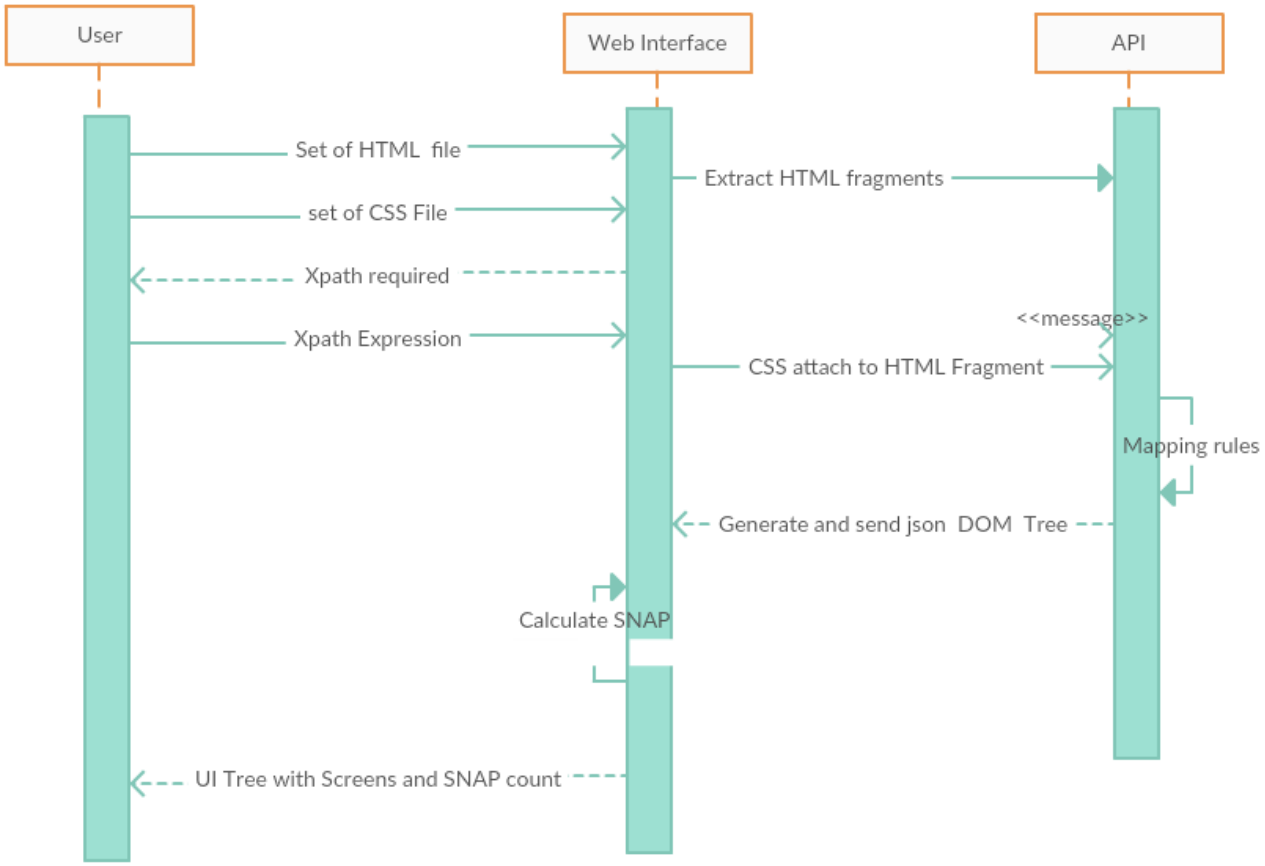

Figure 4.2 Sequence Diagram showing communication between the user and the tool.

4.3.4 Flow Chart

The algorithm of processing HTML and CSS files is presented in Figure 4.3.

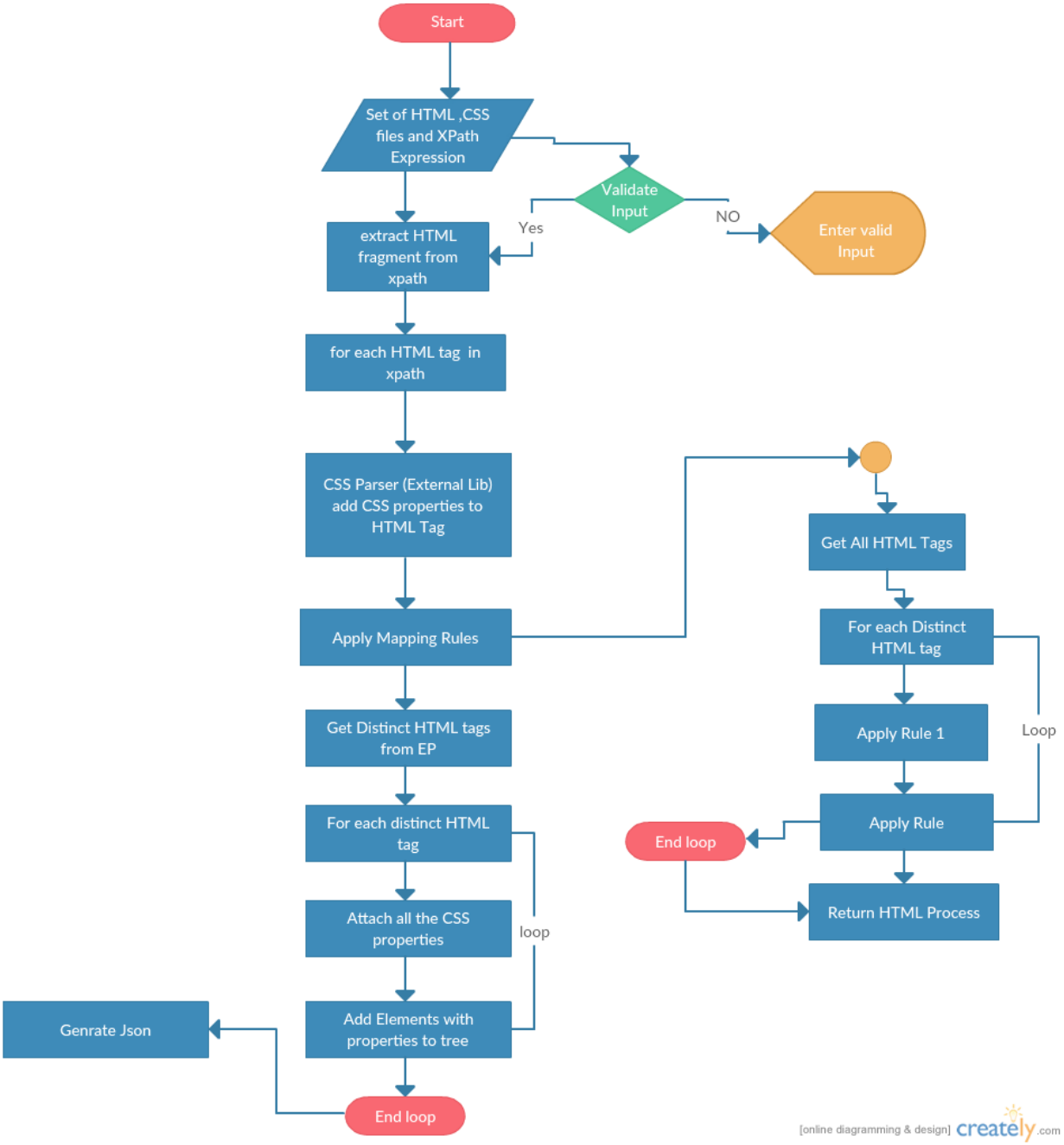

Figure 4.3 The algorithm of processing HTML and CSS files.

4.4 Implementation and maintenance

The most important classes of the tool are presented on the class diagram shown in Figure 4.4. The *Package CSS parser* is an external library this we used in our tool to inline the CSS the CSS parser requires HTML file and CSS file as an input for the inlining of CSS with HTML elements. *Class HTMLDOMParser.cs* is the main parsing engine that extracts the xpath path and applies rules and generate three structure for the output JSON. *Class RuleSet1Aand1B.cs* defines the implementation of the tool mapping rules and apply rules to the HTML tags and return it to the *HTMLDOMParser.cs.* Class *Xml_Reader.cs* reads the defined UIElements from *UIElements.xml* and send to the rules from which the rule can extract the UIElement name.The *Package Tree* contains the tree structure for the json. The package consists of 5 classes each class represent tree level in the json DOM structure. *Elements.cs, EP_screens.cs, UI_elements.cs, Properties.cs, Property_Values.cs* are the classes defined in the tree package.

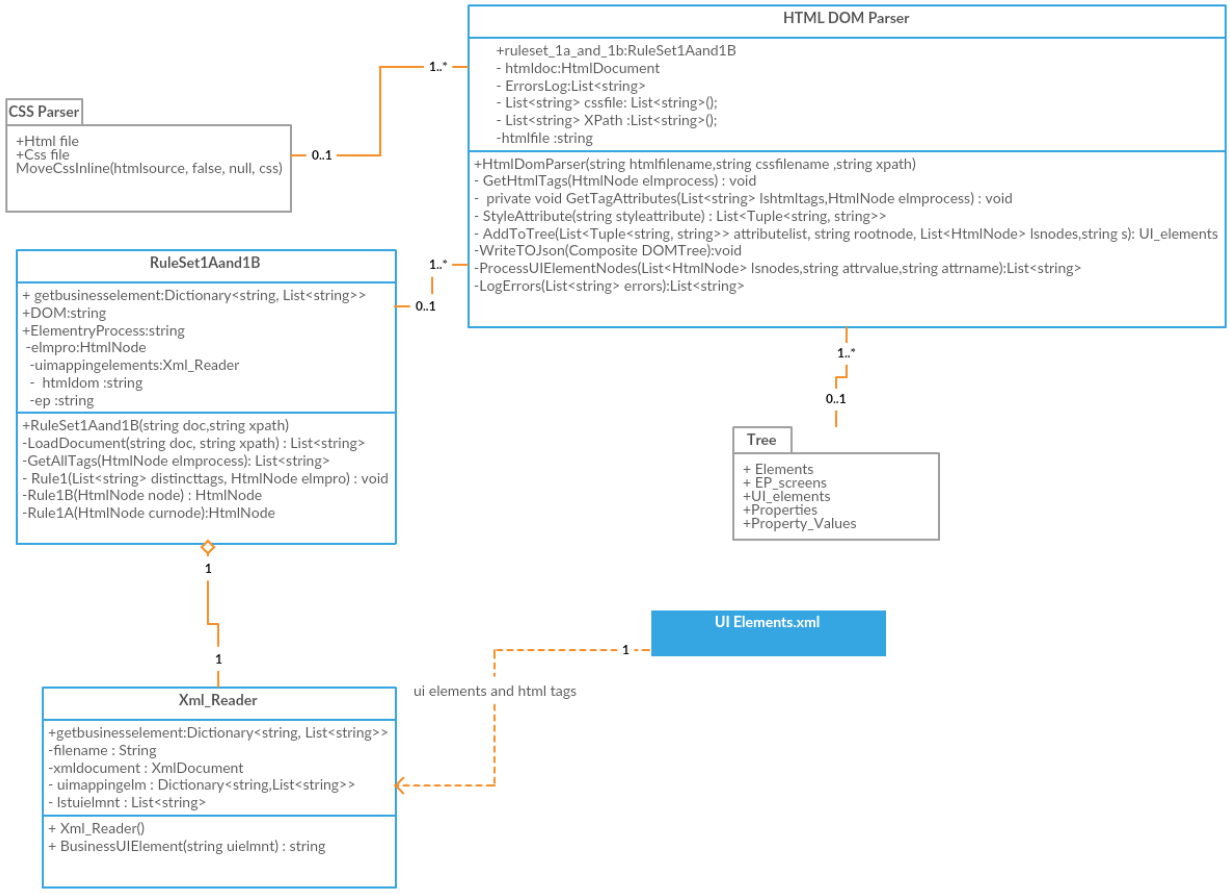

Figure 4.4 Class diagram showing the most important classes of the parsing tool.

The output JSON file format has a tree-like structure. Each node in the tree is named according to its responsibility. The file contains all the required information, e.g., elementary process name, HTML

files, XPath, and errors that encounter during the processing. The format of the JSON file is given below with the explanation of its keys:

```
{ “EP_name": 'Yahoo login",
 “EP_screens”: [
   {
      “HTML_file”: “yahoo.html”,
      “CSS_files”: ["main.css", “other.css”],
      “XPaths”: [“//div[@id = ‘login’]”, “//div[@id = ‘other’]”],
     “UI_elements” : [
      {
         “UI_element_name”: “inputform”,
         “properties”: [
           {
              “property_name”: “method”,
              “property_values”: [
                {
                   “value”: “POST”
                   “HTML_fragments”: [“<form method=\”POST\” action=\”https...”, ...]
                },
                {
                   “value”: “action”
                   “HTML_fragments”: [“<form method=\”POST\” action=\”https...”, ...]
                },
                ...
             ]
           },
           {
              “property_name”: “action”,
             ...
           },
        ]
      }
     ]
   }
 ]
}
```

Explanations of the keys:

- EP_name - the name of the elementary process
- EP_screens - a list of screens used by the elementary process
- HTML_file - the name of the HTML file containing the screen
- CSS_files - the names of CSS files that were included to the analysis for the screen
- XPaths - xpaths that defines the screen
- UI_elements - the list of UI elements found, for each UI element:
- UI_element_name - the name of UI Element
- properties - a list of properties that were configured, and for each property:
- property_name - the name of the HTML property
- property_values - the list of values that were set for the property, for each:
- value - the value itself
- HTML_fragments - a list of HTML fragments were the particular value was set.

This hierarchical structure is written into a JSON file that is used by any language to transform into Graphical user interface for readability. The JSON format is chosen to make it technology independent for graphical user interface and tree structure make it fast and easy to process in any language. The final output that presented to the user as Graphical interface for the interaction with the tool and better understandability. An example of the user interface is presented in Figure 4.5.

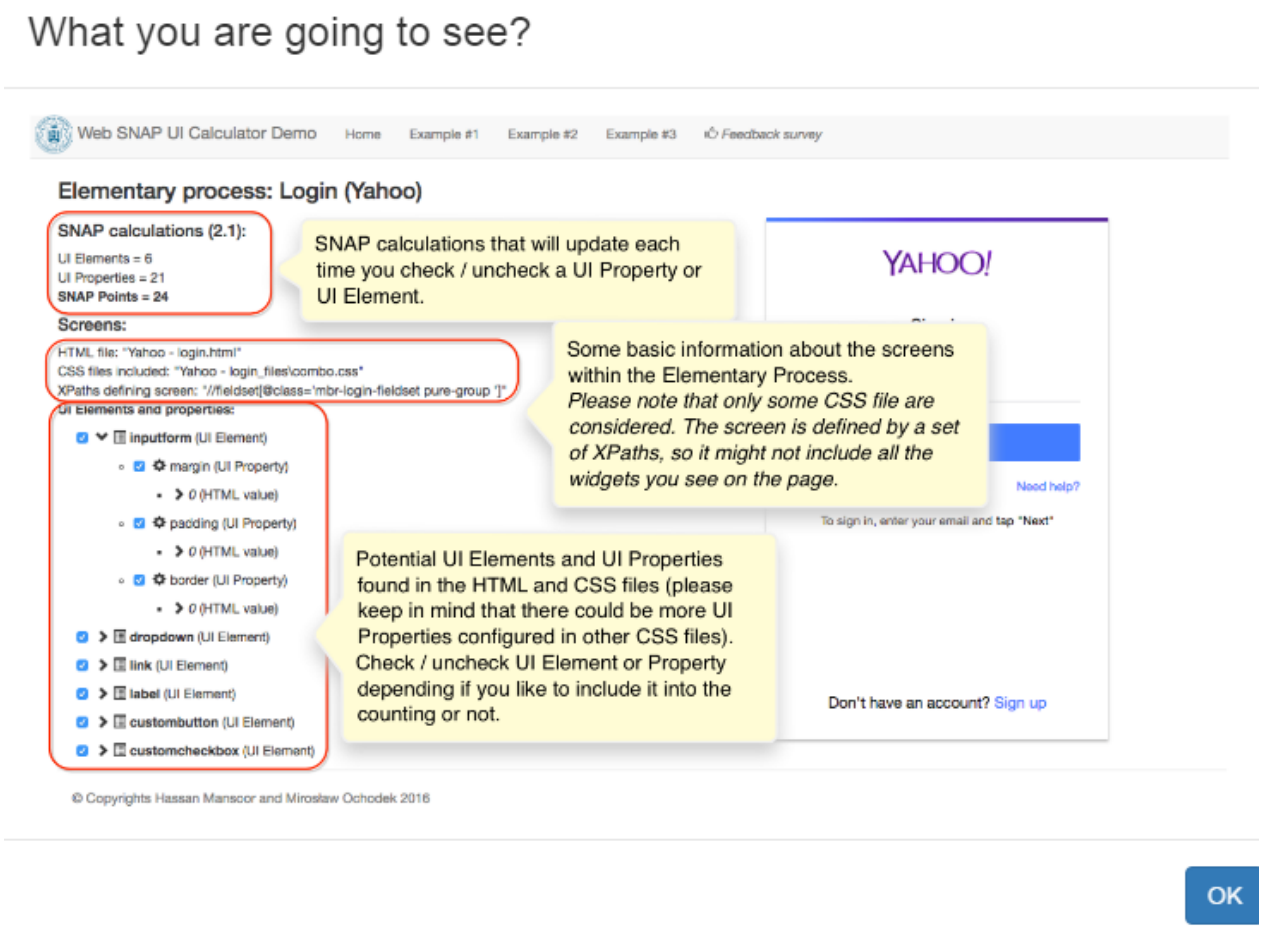

Figure 4.5. An example of the interactive report user interface with the description of its content.

Chapter V

PRELIMINARY VALIDATION

In this chapter we describe the preliminary validation by considering different design of different websites and analyze the different results.

5.1 Study goal

The goal of the study was to investigate *if the proposed method and its implementation in the form of the prototype tool is able to process web pages and provide the expected result.*

The expected outcome differs depending on the configuration of the tool, i.e., rules for mapping between HTML tags and UI Elements, and rules for identifying UI related HTML attributes and CSS properties. Therefore, the correctness of the output provided by the tool has to be investigated in a given context.

5.1 Study design

To establish a context that would be fully understood by the author, three web pages were considered as objects of the study: yahoo.com, github.com, and hitechmobile.sg. For each of these websites, we selected a single elementary process to analyze. Because the correctness of the results has to be verified manually, the selected processes had to be rather simple.

Once the elementary processes were selected, we downloaded web pages (HTML) and corresponding CSS files (we did not take into account external CSS files). In addition, XPaths defining the screens were determined. We used also a predefined set of rules. Finally, such input was provided to the prototype tool and the output report was generated. The output was compared to the original pages to investigate the differences.

5.2 Analyzed Cases

From our study design we will analyze each web page separately in detail.

Yahoo.com

This is the first case that we considered to analyze was yahoo.com. It is a website for checking emails and instant messages, and much more. For our study we decided to choose a login elementary process of this application, which is an independent separate web page. The results of its processing are presented in Figure 5.1.

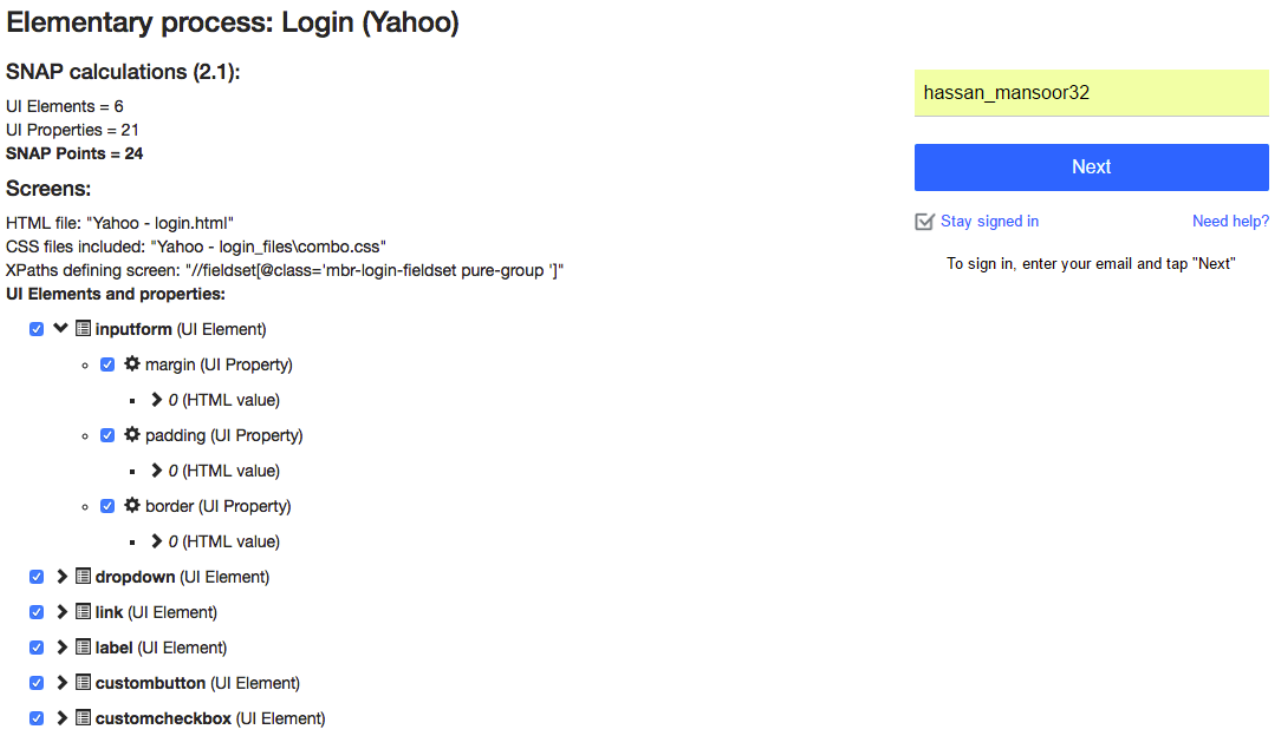

Figure 5.1 SNAP UI calculations for the yahoo.com- login screen.

The presented output shows the elementary process name, SNAP calculations, and a tree-like structure showing UI Elements and Properties. As we can see that the tool identified 6 unique UI elements that are the HTML tags that are root nodes and below them we have properties that is the next level of the tree and then we have values of these properties at the leaf we have structure of HTML element.

Git Hub.com

The second case was the github.com - one of the popular version control systems for maintaining code. We selected the sign-up elementary process as an object of the study. We downloaded the web pages and corresponding CSS files. The web page displaying the sing-up form included other controls related to different elementary processes. To specify our target elementary process we used the following XPath //div[@class='setup-wrapper']. The results provided by the prototype tool are presented in Figure 5.2.

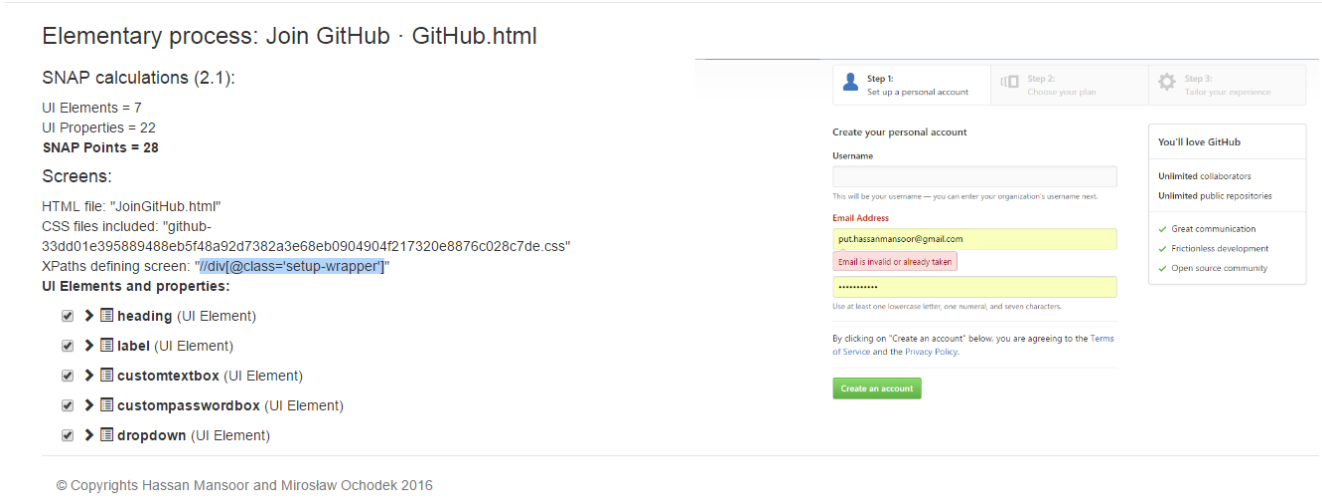

Figure 5.2 SNAP UI calculations for github.com sign-up page.

As shown in the figure, the elementary process includes 7 UI Elements.

Hitech mobile .sg

The last case was the hitechmobile.sg application. This is a website for online shopping for mobile phones in Singapore. The selected elementary process was regarding viewing product details. This elementary process states that the user can view the details of the product which he/she wants to purchase. The results provided by the tool are presented in Figure 5.3:

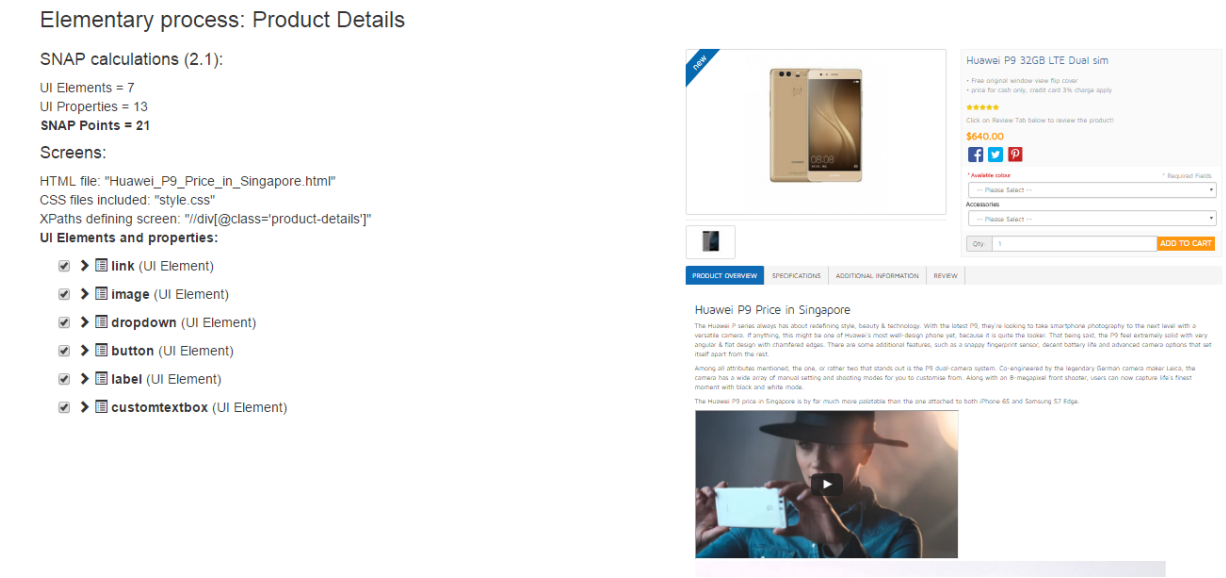

Figure 5.3 SNAP UI calculations for Hitechmobile view product details.

5.3 Results

The summary of the results obtained for the investigated cases are presented in Table 5.1. The table shows the UI Elements and UI Properties (with the number of different values configured put in the brackets) that were found to be configured for the selected elementary processes.

	yahoo.com	hitecmobile.com.sg	github.com
Elementary process	Login to the system	View product information	Signup for a GitHub account
Input HTML	Yahoo – login.html	huawei mate 8 price.html	Join GitHub – GitHub.html
Input CSS	combo.css	style.css	github-33dd01e39588....css
Screen (XPath)	//fieldset[@class='mbr-login-fieldset pure-group ']	//div[@class='product-details']	//div[@class='setup-wrapper']

Configured UI Elements	• **Form**
	  ○ margin (1)
	  ○ padding (1)
	  ○ border (1)
	• **Hyperlink**
	  ○ tabindex (1)
	  ○ text-decoration (1)
	  ○ color (1)
	  ○ font-weight (1)
	  ○ font-size (1)
	• **Drop-down list**
	  ○ margin (1)
	  ○ padding (1)
	  ○ drop-down-type (1)
	• **Label**
	  ○ for (1)
	  ○ display (1)
	  ○ margin (1)
	  ○ cursor (1)
	  ○ font (1)
	  ○ color (1)
	• **Button**
	  ○ tabindex (1)
		• **Hyperlink**
		  ○ color (1)
		• **Image**
		  ○ width (1)
		  ○ max-width (1)
		  ○ height (1)
		• **Drop-down list**
		  ○ height (1)
		  ○ float (1)
		  ○ clear (1)
		  ○ margin-bottom (1)
		  ○ width (1)
		  ○ background (3)
		  ○ border (1)
		  ○ display (1)
		• **Button**
		  ○ title (1)
		• **Label**
		  ○ for (4)
		• **Text box**
		  ○ maxlength (1)
		  ○ cols (1)
		  ○ rows (1)
			• **Header**
			  ○ margin-top (1)
			  ○ margin-bottom (2)
			  ○ font-size (2)
			  ○ font-weight (1)
			  ○ line-height (1)
			  ○ letter-spacing (1)
			  ○ padding (1)
			  ○ overflow (1)
			  ○ border-bottom (1)
			• **Drop-down list**
			  ○ padding (3)
			  ○ margin (3)
			  ○ font-size (1)
			  ○ list-style (1)
			  ○ margin-top (2)
			  ○ border-top (1)
			• **Label**
			  ○ autocapitalize (1)
			  ○ autofocus (1)
			  ○ for (3)
			• **Text box**
			  ○ autocapitalize (1)
			  ○ autofocus (1)
			  ○ size (1)
			  ○ width(1)
			• **Password box**
			  ○ size (1)
			  ○ width (1)

# Configured UI Elements	5	6	5
# UI Properties	18	17	24
SNAP Points (SP)	20	24	20

Table 5.1 Results of the preliminary validation (values in brackets represent the number of distinct values set to a UI Property)

5.4 Discussion

We observed that the prototype tool was capable of processing the selected pages and calculating the number of UI Elements and properties that were configured. During the validation, we made the following observations regarding the current version of the method and prototype tool:

- Sensitivity to HTML and CSS errors — the proposed approach is sensitive to errors in CSS and HTML. The positive side effect is that the tool might help to improve the quality of the HTML code of the application being measured.
- JavaScript modifying HTML — modern rich internet applications often rely on JavaScript to generate and modify HTML. In extreme cases, downloaded sources of web pages might not contain any HTML tags corresponding to UI Elements. To overcome this issue we recommend using web browser plug-in, such as Firebug (http://getfirebug.com) that allow investigating and downloading source of HTML page as it is at runtime. On the other hand,

we found a drop-down UI Element in the source of Yahoo's sing-in HTML page that was not visible to us at the moment we had visited the page (it would be in a different state of the page).

- Processing output HTML — we process HTML pages that are generated by an application. These are snapshots that capture screens in certain states. Consequently, the HTML page we are processing might not reveal all the properties that were configured. For instance, if we consider a form that uses some visual effects to show validation errors, we might not identify these properties if an empty form was downloaded and processed. To overcome this issue, we should provide many instances of the same page in different states to obtain a union of configured properties.
- Importance of custom mapping rules — the set of mapping rules is an essential component of the tool. For instance, our built-in set of mappings rules included HTML tags such as INPUT type="hidden" which did not seem important from the perspective of the processed UIs. Consequently, it caused identification of few UI Elements that were false positively classified as configured. It is also important to define custom rules to be able to distinguish different UI Elements that are represented by the same HTML tag. For example, without a set of custom mapping rules, we would not be able to distinguish between images and icons. However, in some cases, distinguishing such elements might be impossible (image and icon are distinguished based on their content and not based on HTML / CSS properties).

5.4 **Threats to Validity**

There are some threats to validity of study that should be discussed. The first one relates to constructs validity and regards the understanding of the rules of 2.1 subcategory. Although the proposed solution was not officially validated and accepted by the IFPUG Non-Functional Sizing Standards Committee responsible for the development of SNAP, the presentation given at the IWSM Mensura conference was attended by the president of the committee who expressed positive feedback about the proposed approach. Therefore, it seems we may assume that the risk of misunderstanding the SNAP rules is minor. An important threat to internal validity relates to the manual verification of the correctness of the output. Even though we selected simple pages to analyze, their HTML and CSS code seemed complex. There were also fragments in the HTML code that were not rendered in the browser. Finally, the most important threats relate to external validity. The analyzed cases were simple and do not constitute a valid sample of all potential web

applications. There are also questions related to the usability of the tool and its potential acceptance by the industry which remained opened.

Chapter VI

Conclusion

In this thesis an approach measure the non-functional size of user interface of web applications with the help of IFPUG SNAP method was presented. IFPUG SNAP method introduced as a compliment for functional point analysis which used for measuring the functional requirements.
We investigated the problem which emerges when it comes to pricing software development projects using IFPUG SNAP and to achieve transparency to track that which UI elements and their properties were configured to meet the non-functional requirements. For the solution of this problem we developed a method and a prototype tool that process a set of HTML pages and CSS files to identify the UI elements and their properties that were configured.
The proposed approach can be customized by providing mapping rules these are the rules for mapping between HTML tags and UI Elements.
The proposed solution was preliminary validated based on three elementary processes belonging to three different websites. Although the prototype tool seemed sensitive to errors in HTML code, we were able to obtain valid results that show potential usefulness of the proposed approach. The results of the study were also presented at the IWSM Mensura conference 2016 and gain lots of attention, also from the leader of IFPUG Non-Functional Sizing Standards Committee.

Future work As future work, we consider extending the types of mapping rules between HTML tags and UI Elements to cover more specific (and rarely occurring cases). We would also like to provide the possibility of comparing different versions of HTML pages to identify the changes in the configuration of UI Elements resulted from enhancement projects. Another interesting area of research would be to adapt the proposed approach to analyze code in templates used to generate HTML pages available in many technologies (e.g., Java JSP, Django, Velocity, etc.) rather than the output HTML pages. Finally, we are planning to further validate the proposed method by conducting a survey and obtaining feedback from the SNAP users community.

References

[1] IFPUG, Function Point Counting Practices Manual, Release 4.3.1, ISBN 978-0-9753783-4-2, 2010.

[2] M. Ochodek and B. Ozgok, "Functional and Non-functional Size Measurement with IFPUG FPA and SNAP—Case Study," in Software Engineering in Intelligent Systems. Springer, 2015, pp. 19–33.

[3] Forrester Consulting, "The Rise Of Web Technology: Addressing Development Complexity And Productivity Demands," 2015.

[4] SNAP Points, https://en.wikipedia.org/wiki/SNAP_Points.

[5] CSSW3C, http://www.w3schools.com/css/, (last visited 20.10.2016).

[6] HTMLW3C. http://www.w3schools.com/html/(last visited 20.10.2016).

[7] H. Mansoor and M.Ochodek, "Towards semi-automatic size measurement of user interfaces in web applications with IFPUG SNAP", Proceedings of IWSM Mensura 2016, pp 201-211, DOI 10.1109/IWSM-Mensura.2016.8, 2016

[8] IFPUG, "Software Non-functional Assessment Process (SNAP) — Assessment Practices Manual, Release 2.3, ISBN 978-0-9903007-3-1," 2015.

[9] C. Tichenor, "A New Metric to Complement Function Points," CrossTalk—The Journal of Defense Software Engineering, vol. 26, no. 4, pp. 21–26, July/August 2013.

[10] ISO/IEC, "ISO/IEC 14143-1: Information technology—Software measurement—Functional size measurement — Part 1: Definition of concepts", 2007.

APPENDICES

Appendix A:

The CD contacting the code of application and the document is delivered.

Appendix B:

Research paper