Measuring usability of a Web Runtime widget

This article shows a practical technique to measure the usability of a Web Runtime widget.

Description

Usability is a property depending on multiple factors, that gives an overall measure of how an application can be used to achieve specific goals. These factors include effectiveness (degree of accuracy of the reached goal, compared to the planned one), efficiency (time and steps required to reach the goal) and user satisfaction (an overall measure of the user feelings about the application's usage).

Several methods are available to measure and improve the usability of a mobile applications, and some of them are listen on Nokia Developer Library: http://library.developer.nokia.com/index.jsp?topic=/Design_and_User_Experience_Library/GUID-27839792-942C-4A77-9D8F-9BDE446140C7.html.

End users testing

One of the best and most useful methods to have an actual measure of the usability of an application, is to let final users directly test the application. This method allows to highlight issues and problems that final users could encounter with an application, while trying to reach their goals.

Measuring efficiency

This approach allows to have an explicit measure of the efficiency of a Web Runtime widget, related to a specific goal, by taking into account these factors:

- the number of clicks performed/required
- and the 'time spent to reach a goal

This technique allows to measure efficiency while users are actually using the widget, without interfering with the normal widget's behavior, by silently monitoring clicks and time spent. For this reason, this technique can be also integrated in widgets released to the public, in order to have a continuous measure of usability in large sets of users (of course, in this case, privacy issues must be taken into account, and explicit permission to the user must be asked).

The HTML code

Let's build a sample widget with 3 screens containing some clickable elements, and with a "goal" element placed on the third screen.

```
<html>
 [...]

 </head>
 <body onLoad="javascript:init();">
  <div id="screen_0" class="screen">

   <div class="wrong_input" onclick="alert('go nowhere')">
    Do not click me!
   </div>

   <div class="correct_input" onclick="gotoScreen('screen_1')">
    Go to next screen
   </div>

  </div>

  <div id="screen_1" class="screen" style="display: none;">
   <div class="wrong_input" onclick="alert('go nowhere')">
    Do not click me!
   </div>

   <div class="wrong_input" onclick="alert('go nowhere')">
    This brings you nowhere
   </div>

   <div class="correct_input" onclick="gotoScreen('screen_2')">
```


(C) Copyright Nokia 2013. All rights reserved.

```
    Go to next screen
   </div>

  </div>

  <div id="screen_2" class="screen" style="display: none;">
   <div class="correct_input" onclick="userGoalReached()">
    This is the widget's goal!
   </div>

  </div>
 </body>
</html>
```

Things to note about the above code:

- the `gotoScreen()` function allows the user to go from the current screen to the screen that has the specified ID
- in the last screen (screen_2) the goal is represented by the DIV element containing the "This is the widget's goal!" text. This element calls the `userGoalReached()` function, defined later, that notifies about the goal reaching.

The JavaScript code

Number of clicks and time variables

First, two variables are defined in order to hold the number of clicks performed by the user, and the initial time of the usability analysis:

```
/* number of clicks performed */
var userClicks = 0;

/* analysis initial time */
var startTime = null;
```

Monitoring the user actions

In order to count the performed clicks, the widget must listen to all click events, and increase the overall count (stored in the userClicks variable) for each of them. The following startUsabilityAnalysis() function:

- initializes the number of clicks to zero
- stores the analysis' initial time
- starts listening to the click events performed within the widget

```
function startUsabilityAnalysis()
{
 userClicks = 0;

 startTime = new Date().valueOf();

 document.addEventListener('click', userClick, false);
}
```

For each click event, the `userClick()` function is called, and it has to increment the overall clicks' count by one:

```
function userClick()
{
 userClicks++;
}
```

Reaching the planned goal

When the user reaches the planned goal, the widget has to calculate the time spent to reach it, and to use the measured quantities (clicks and time) in a way that can be useful for a further analysis, as:

- if the testing users are remote, these values can be sent via Ajax calls. Multiple measures can also be performed, in different usage sessions, and sent in an aggregate Ajax call to minimize the network traffic.
- if the test is local, the values can also be shown on screen, so that the final users and/or the developers and designers can immediately easily read them

In this example, the measured values are shown is an alert message, but any other approaches, as the one described above, are possible:

```
function userGoalReached()
{
 document.removeEventListener('click', userClick, false);

 var endTime = new Date().valueOf();

 var totalTime = endTime - startTime

 alert(
  "Performed clicks: " + userClicks + "\n" +
  "Time spent: " + (totalTime / 1000) + " seconds."
 );
}
```

Further improvements

Improvements and adjustments can be done to the code presented in this article, in order to adapt it to the different needs of different widgets. Examples are:

- counting wrong clicks: this examples only counts the global clicks, without checking where the click was performed. In a finer analysis, also the information about the click targets could be useful, in order to see where the user clicked, and use this information to modify or adjust the widget's layout
- defining multiple goals: a widget could have more than a single goal for a specific action. In this case, all the goals could have the same importance, or a goal could be more important than the others. In this case, the goal importance should be quantified and stored with the other values (time and clicks) to allow further analysis
- measuring effectiveness: the proposed approach measures the efficiency of a widget, related to a specific goal. In order to measure effectiveness, a the

(C) Copyright Nokia 2013. All rights reserved.

completeness of the goal must also be measured (e.g.: measuring completeness of entered information).

Downloads

The sample download presented in this article is available for download here: Media:MeasuringUsabilityWidget.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.