November 2019

USER EXPERIENCE IMPROVEMENT BY ANALYZING CUSTOMER FACING PATHS

HP INC

Title
User experience improvement by analyzing customer facing paths

Abstract
User-facing applications are computer programs that typically provide a Graphical User Interface (GUI) to render the program operational, therefore they may be thought of as a collection of pages that are individually connected to one or more user interactions. Bound sequences of interactions ultimately represent use cases – a list of actions or required steps to achieve a goal. This disclosure defines a method capable of browsing an application and offering a visual representation of the actual use cases as implemented, from a user's perspective.

Problems Solved
1. Improve user experience by being able to visually analyze customer facing paths to perform a valid action in a computing system.
2. Improve use case visualization: as the tree provides a visual representation of possible user interactions, it's possible to tell whether desired use cases are unreachable, and whether undesired use cases are reachable.

Description
1. Finding interaction elements: in this phase, all interaction elements of an application's page are discovered by using image recognition techniques.

2. Generating states for previously found elements: For instance, an input text field may hold different strings, but it may also be blank. Each one of these possibilities is considered to be a unique state.

3. Grouping element state combinations into interactions: Different state combinations – for different elements – are grouped into interactions. These represent a series of steps a human user would perform on an application to achieve a goal. Interactions are grouped until the entire state possibility matrix for all elements is covered. i.e.: all possible combinations of actions a user may perform on an application's page are mapped.

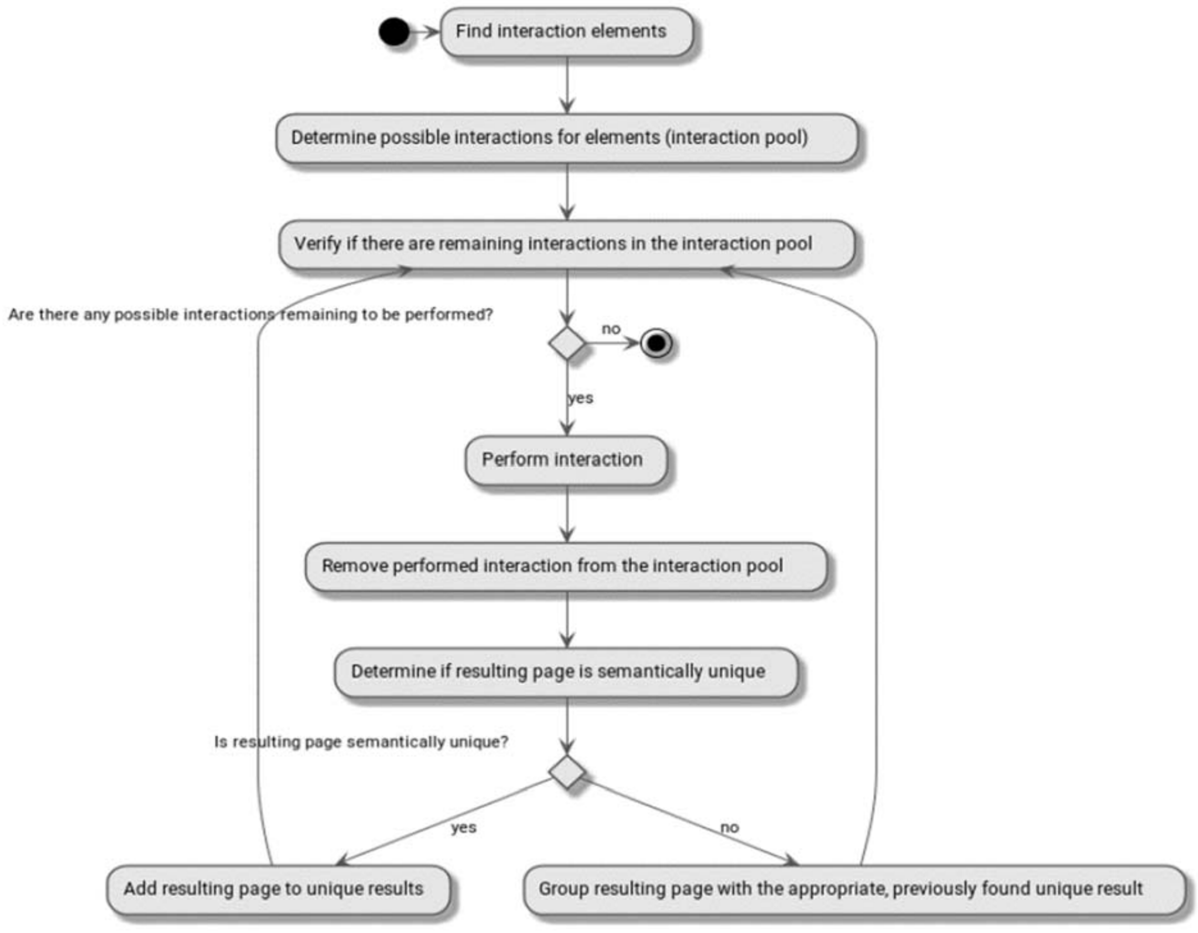
Find interaction elements
Determine possible interactions for elements (interaction pool)
Verify if there are remaining interactions in the interaction pool
Are there any possible interactions remaining to be performed?
no
yes
Perform interaction
Remove performed interaction from the interaction pool
Determine if resulting page is semantically unique
Is resulting page semantically unique?
yes
no
Add resulting page to unique results
Group resulting page with the appropriate, previously found unique result

Figure 1: Method for customer-facing path discovery

4. Perform interactions and determine whether the results are unique: Once the elements are all discovered and, possibly, some restrictions are imposed, the algorithm starts running by the initial pointed page and then traverses the elements through interactions. While it runs, it fills the data structure nodes as it records the relations being disclosed as the operations are taking place. To determine whether or not results are semantically unique, the node's edges are compared. If multiple source nodes point towards the same destination node, that result is semantically equivalent.

The data structure that represents each node in the tree is represented by Figure 2, which presents 3 main sets to hold the information:

a) Element set: holds the element’s position, the element state in the interaction, and its type; The position is represented by two coordinate pairs: one pair for the top-left starting position, and one pair for the bottom-right ending position;

b) Page set: holds the page's ID, an auto-generated incremental number that unambiguously identifies a page in the application; this set also holds a list of all elements belonging in the page, as well as their initial state;

c) Interaction set: holds the source and destination page information and an array of all elements that constitute such particular interaction.

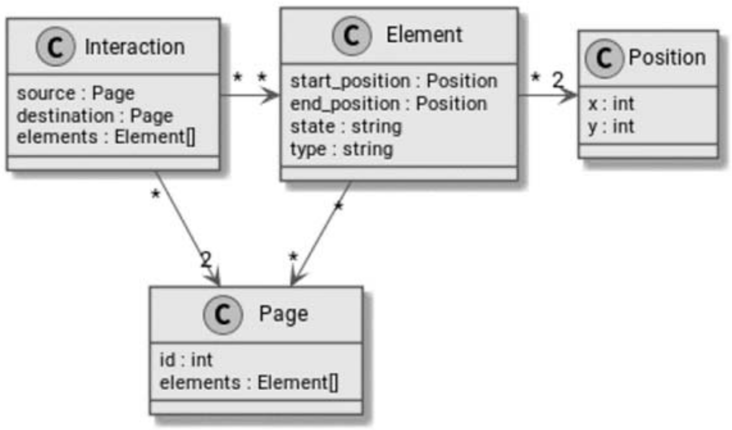

Figure 2: Node data structure

Advantages

The solution herein described poses several advantages over only using manual analysis for user experience improvement:

• It automatically discovers and reveals possible UI path optimizations. If two or more UI paths lead to the same destination, then the shortest route is likely the easiest for the user to perform.

• By the same token, it also reveals redundant paths, uncovering the possibility for redundant UI elements to be removed altogether.

• Bugs may be more easily found: as the procedure traces possible interactions a user may perform, it will also trace these interactions' consequences, which could reveal unexpected behavior. These bugs might be totally unpredictable and could be effectively impossible to find were it not for exploratory test procedures such as the proposed one.

• It is environment independent: this method could be used to test any application that displays a graphical user interface.

Disclosed by Lucia Maciel, Martin Jungblut Schreiner, Ricardo Miotto Redin, Alan Aguirre and Cristiane Machado, HP Inc.