HPath: Resilient Locators for GUI-based Tests

Renaud Rwemalika, *University of Luxembourg*, renaud.rwemalika@uni.lu

To remain competitive, industrial actors need to deliver products in a timely manner without compromising on their quality. To achieve this goal, companies adopted automated testing at every level of the quality assurance cycle. In this context, Graphical User Interface (GUI) testing is a class of testing that targets the user interface to interact with the system under test (SUT) in a black-box fashion. As such, GUI-based tests are executed at a high level of abstraction, allowing to exercise scenarios relying on larger portions of the SUT.

Typically, GUI-based tests are written as scenarios composed of test steps. Each test step consists of a triple: an action to be performed, the GUI element on which to perform the action and an optional value passed to the targeted element. Consequently, for each step, the test has to be able to uniquely identify the GUI element to interact with. One limitation of such tests is their propensity to break. A test breakage can be defined as a failure of a test not resulting from a regression but due to the obsolescence of the test under SUT evolution. The propensity for a test to exhibit a breakage is called its fragility. This question is especially important in the realm of test script generation, used for instance in capture and replay tools, which needs robust locators to reduce maintenance effort. Indeed, many tools relying on such technologies generate brittle scripts that have a low creation cost, but induce a high maintenance costs due to test breakages at each SUT update. To reduce the fragility induced by locators, *i.e.* locator fragility, the literature offers different strategies such as shortening locator expressions or relying on different properties of the targeted elements.

Property- or DOM-based locators, rely on properties of the eXtensible Markup Language (XML) structure of the Document Object Model (DOM) to locate GUI elements. In the context of web testing, where each page is associated to a DOM, each GUI element can be identified by means of XPath queries. Exploiting the DOM structure instead of directly targeting the rendered page leads to robust locators. Robust locators are defined as locators that are not affected by rendering artifacts (*e.g.*, screen resolution, element colors, text font, etc). However, XPath has one pitfall: it relies heavily on internal properties of the elements they visit (*e.g.*, non-rendered attributes, position in the DOM tree, or hierarchy of tree nodes). Despite being a reasonable strategy for most of the tasks that they are used for (*e.g.*, information retrieval and content formatting), this approach can be problematic in the case of automated GUI testing, and more specifically in user acceptance testing where the reliance on these attributes leaks structural details of the page that should not be present in the tests. Previous research has shown that this leakage of structural details makes tests more sensitive to iso-functional changes (*i.e.*, changes that are not related to the behavior of the application), resulting in test breakage.

With this work, we propose a novel locator approach called HTML Path Language (HPath). HPath attempts at capitalizing on the strengths of both DOM-based locators and their rendering properties. The intuition behind HPath lies in the fact that, if only visible features are permitted in the query, then the resulting locators should be more resilient to iso-functional structural changes. Moreover, relying on rendered properties (such as the label of an input) makes our locators (and, by extension, test scripts) easier to understand as the reliance on internal DOM properties has been minimized. Indeed, HPath relies on the structural information of the HTML document to create robust locators. However, unlike XPath, HPath is aware of the rendering process of the HTML document, allowing to capitalize on visual elements of the documents but also to ignore properties and structural elements that do not affect the rendering of the page.

To validate HPath, we conduct an empirical study based on three large open-source projects, MISO LIMS, Keycloak, and OpenOLAT. We mine more than 300,000 web elements targeted by GUI test suites and their evolution in the form of over 30,000 locator pairs. We assess the effectiveness of HPath at creating expressive locators that are resilient to iso-functional changes, where the resilience of a locator is the inverse of its fragility and account for their robustness to SUT changes. We compare the performance of HPath against two state-of-the-art XPath queries generation algorithms: Robula+ and the algorithm proposed by Nguyen *et al*, which exploits neighboring text elements.

Our results suggest that more than the length of the locator, relying on stable properties can increase locator resilience to change. Indeed, while Robula+ and Nguyen *et al* manage to generate more concise expressions, they are more sensitive to iso-functional structural changes because of their heavy reliance on structural attributes. HPath, on the other hand, manages to generate locators that remain resilient to iso-functional changes, thus, reducing the locator breakage from 67.61% (Robula+) and 74.05% (Nguyen *et al*) to 6.48%. However, in its current form, HPath is not always able to extract rendered properties to create good location paths. Consequently, the algorithm tends to leak the hierarchical structure of the DOM which is the root cause of the breakages when using HPath. Despite this limitation, HPath manages to outperform the two other approaches in term of resilience to change, showing that more important than the length of the locator, the choice of the predicates is what holds the most importance when generating locators resilient to the evolution of the target elements.