Qwalkeko
a History Mining Tool

Reinout Stevens
resteven@vub.ac.be
@ReinoutStevens
L. Christophe, R. Stevens, and C. De Roover,
“Prevalence and maintenance of automated functional tests for web applications,”

R. Stevens, and C. De Roover,
“Querying the History of Software Projects using QwalKeko,”
Selenium
public class CheesecakeFactory {
    HtmlUnitDriver driver;

    @BeforeTest
    public void startDriver() {
        driver = new HtmlUnitDriver();
    }

    @AfterTest
    public void stopDriver() {
        driver.close();
    }

    @Test
    public void listCheesecakes() {
        driver.get("http://www.thecheesecakefactory.com/");
        driver.findElement(By.linkText("Menu")).click();
        driver.findElement(By.linkText("Cheesecakes")).click();
        List<WebElement> cheesecakes = driver.findElements(By.xpath("id('leftNav_levelTwo')//li"));
        System.out.println(cheesecakes.size() + " cheesecakes:");
        for (int i=0; i<cheesecakes.size(); i++) {
            System.out.println(i+1 + ". " + cheesecakes.get(i).getText());
        }
    }
}
public class CheesecakeFactory {

    HtmlUnitDriver driver;

    @BeforeTest
    public void startDriver() {
        driver = new HtmlUnitDriver();
    }

    @AfterTest
    public void stopDriver() {
        driver.close();
    }

    @Test
    public void listCheesecakes() {
        driver.get("http://www.thecheesecakefactory.com/");
        driver.findElement(By.linkText("Menu")) .click();
        driver.findElement(By.linkText("Cheesecakes")) .click();
        List< WebElement> cheesecakes = driver.findElements (By.xpath("id('leftNav_levelTwo')//li"));

        System.out.println(cheesecakes.size() + " cheesecakes:" );
        for (int i=0; i<cheesecakes.size(); i++) {
            System.out.println(i+1 + ". " + cheesecakes.get(i).getText());
        }
    }
}
## Corpus

<table>
<thead>
<tr>
<th>Language</th>
<th>#Repositories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>4287</td>
</tr>
<tr>
<td>Python</td>
<td>1800</td>
</tr>
<tr>
<td>Ruby</td>
<td>1503</td>
</tr>
<tr>
<td>C#</td>
<td>558</td>
</tr>
<tr>
<td>JavaScript</td>
<td>237</td>
</tr>
</tbody>
</table>
Refinement

I. Were created before 2013
II. Have over 100 commits in the last year
III. Are larger than 500 KBytes.
IV. Number of SELENIUM files > 40
1. Do Selenium-based functional tests co-evolve with the web application? For how long is such a test maintained as the application evolves over time?

2. How are Selenium-based functional tests maintained? Which parts of a functional test are most prone to changes?
... specify code characteristics through Ekeko relations, leave search to core.logic

collection of all substitutions for ?s and ?e

(ekeko* [?s ?e]
  (ast :ReturnStatement ?s)
  (has :expression ?s ?e)
  (ast :NullLiteral ?e))

such that the following Ekeko relations hold:

ast/2 holds for :ReturnStatement, ?s
has/3 holds for :expression, ?s, and ?e
ast/2 holds for :NullLiteral, ?e

([?s1 ?e2] ... [?sn ?en])

([#<ReturnStatement return null; #<NullLiteral null>]
 ...
 [#<ReturnStatement return null; #<NullLiteral null>]]

?e is the value of the property named :expression of ASTNode ?s

actual search performed by core.logic
QwalKeko

(qwalkeko* [?result ?vars]
(qwal graph start end
  [?locals]
  &goals)))

configuration of the engine

local variables available in goals

move through the graph / change current version
nekoko predicates evaluated in the current version

(in-git-info [current] &conditions) conditions hold in current version (only git data)
(in-source-code [current] &conditions) conditions hold in current version (git data + source)
q=>  move current version to a successor
q=>*  skip an arbitrary number of versions
q<=  move current version to a predecessor

https://github.com/ReinoutStevens/changenodes
Identifying Selenium Files

(defn compilationunit\|selenium [?cu]
  (fresh [?imp ?impname ?str]
    (ast :CompilationUnit ?cu)
    (child :imports ?cu ?imp)
    (has :name ?imp ?impname)
    (name\|qualified-string ?impname ?str)
    (succeeds (string-contains ?str ".selenium"))))
Identifying Selenium Files'

```clojure
(defn find-selenium-files [version]
  (qwalkeko* [?info ?cu]
    (qwal graph version version []
      (in-source-code [curr]
        (fileinfo|edit ?info curr)
        (fileinfo|compilationunit ?info ?cu curr)
        (compilationunit|selenium ?cu))))
)```
Identifying Selenium Files”

```
(map
  (fn [version]
    (let [results (find-selenium-files version)]
      (write-results-to-db results)
      (ensure-delete version))))
(:versions graph))
```
Do Selenium-based functional tests co-evolve with the web application? For how long is such a test maintained as the application evolves over time?
How are Selenium-based functional tests maintained? Which parts of a functional test are most prone to changes?
public class CheesecakeFactory {

    HtmlUnitDriver driver;

    @BeforeTest
    public void startDriver() {
        driver = new HtmlUnitDriver();
    }

    @AfterTest
    public void stopDriver() {
        driver.close();
    }

    @Test
    public void listCheesecakes() {
        driver.get("http://www.thecheesecakefactory.com/");
        driver.findElement(By.linkText("Menu")).click();
        driver.findElement(By.linkText("Cheesecake")).click();
        List<WebDriver> cheesecakes = driver.findElements(By.xpath("id('leftNav_levelTwo')//li"));

        System.out.println(cheesecakes.size() + " cheesecakes:");
        for (int i=0; i<cheesecakes.size(); i++) {
            System.out.println(i+1 + ". " + cheesecakes.get(i).getText());
        }
    }
}
Classification Query

1 (qwalkeko* [?change ?info ?end ?type]
2   (qwal graph version ?end [?left-cu ?right-cu]
3     (in-git-info [curr]
4       (fileinfo|selenium|edit ?info curr))
5     (in-source-code [curr]
6       (fileinfo|compilationunit ?info ?right-cu curr))
7     q<=
8     (in-source-code [curr]
9       (compilationunit|corresponding ?right-cu ?left-cu))
0   (change ?change ?left-cu ?right-cu)
1   (classify-change ?change ?type))))
;; By.<something>(value)
(defn methodinvocation\by [?x]
  (fresh [?name]
    (ast :MethodInvocation ?x)
    (child :expression ?x ?name)
    (name:simple-string ?name "By")))

;; @FindBy(something)
(defn annotation\findBy [?x]
  (fresh [?name]
    (ast :NormalAnnotation ?x)
    (has :typeName ?x ?name)
    (name:simple-string ?name "FindBy")))

(defn changel\affects\findBy [change ?find-by]
  (all
    (changel\affects\node change ?find-by)
    (conde
      [((methodinvocation\by ?find-by))
       [(annotation\findBy ?find-by)]))))
Change Hit Ratio

Change Classification

- assertion
- command
- constant
- demarcator
- location
resteven@vub.ac.be
@ReinoutStevens

https://github.com/ReinoutStevens/damp.qwalkeko/
https://github.com/ReinoutStevens/ChangeNodes/
https://github.com/cderoove/damp.ekeko