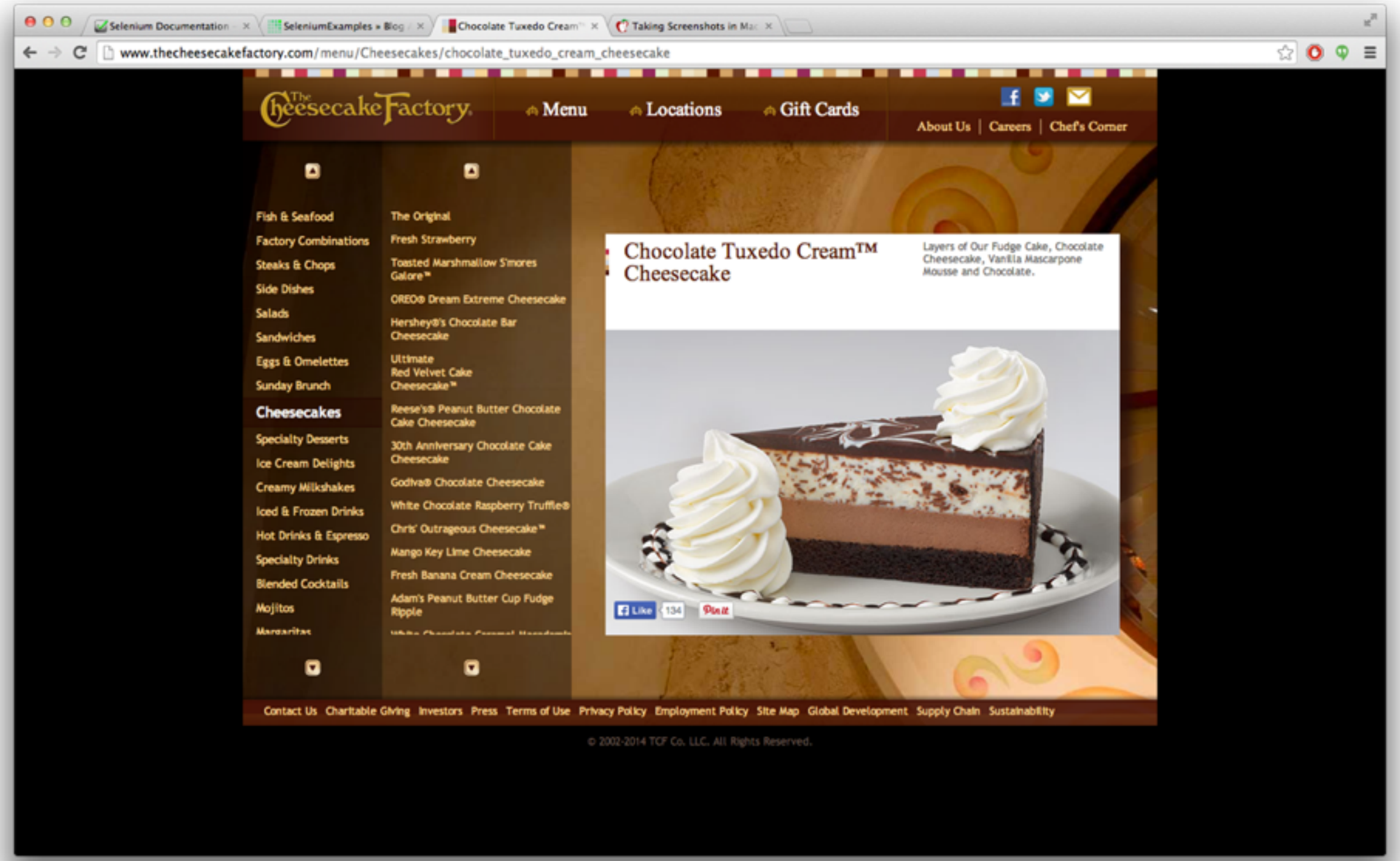
Qwalkeko
a History Mining Tool

Reinout Stevens
resteven@vub.ac.be
@ReinoutStevens

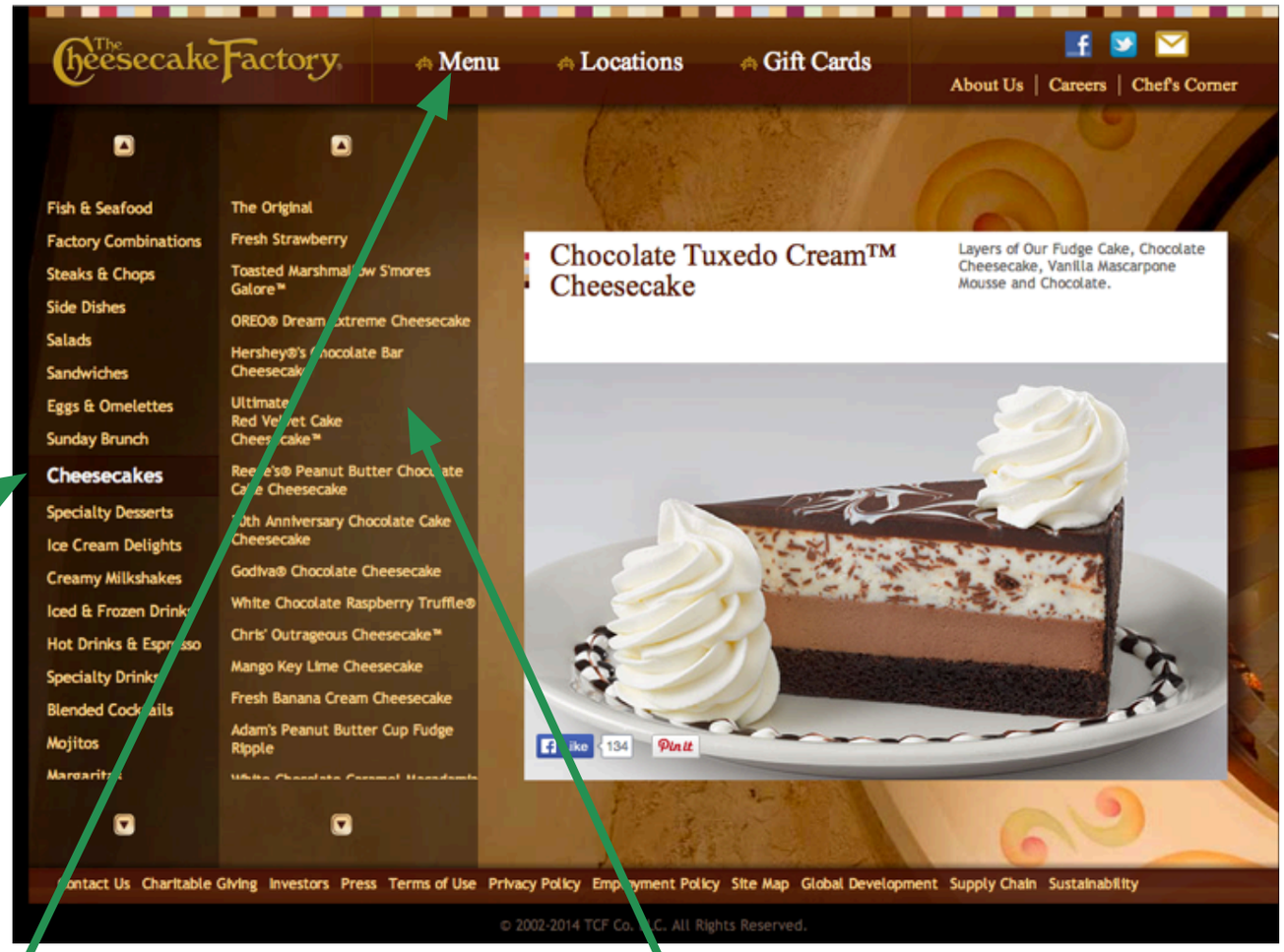
L. Christophe, R. Stevens, and C. De Roover,
"Prevalence and maintenance of automated functional tests for web applications,"
in Proceedings of the 30th International Conference on Software Maintenance and Evolution, 2014, to be published.

R. Stevens, and C. De Roover,
"Querying the History of Software Projects using QwalKeko,"
Hopefully in the Proceedings of the 30th International Conference on Software Maintenance and Evolution, 2014.

Selenium

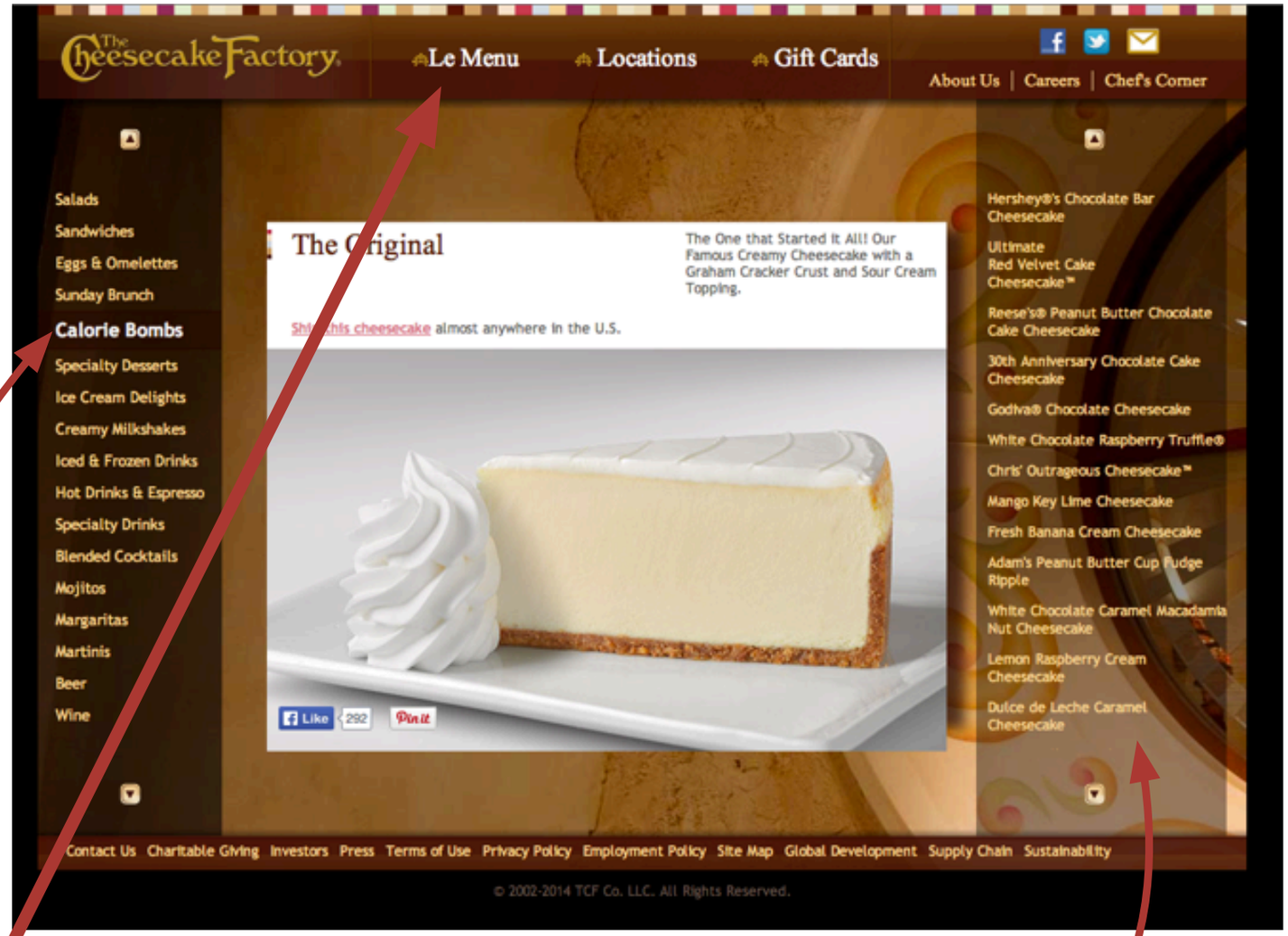

Selenium Documentation
SeleniumExamples » Blog
Chocolate Tuxedo Cream
Taking Screenshots in Ma
www.thecheesecakefactory.com/menu/Cheesecakes/chocolate_tuxedo_cream_cheesecake
The Cheesecake Factory
Menu
Locations
Gift Cards
About Us | Careers | Chef's Corner
Fish & Seafood
Factory Combinations
Steaks & Chops
Side Dishes
Salads
Sandwiches
Eggs & Omelettes
Sunday Brunch
Cheesecakes
Specialty Desserts
Ice Cream Delights
Creamy Milkshakes
Iced & Frozen Drinks
Hot Drinks & Espresso
Specialty Drinks
Blended Cocktails
Mojitos
Margaritas
The Original
Fresh Strawberry
Toasted Marshmallow S'mores Galore™
OREO® Dream Extreme Cheesecake
Hershey®'s Chocolate Bar Cheesecake
Ultimate Red Velvet Cake Cheesecake™
Reese's® Peanut Butter Chocolate Cake Cheesecake
30th Anniversary Chocolate Cake Cheesecake
Godiva® Chocolate Cheesecake
White Chocolate Raspberry Truffle®
Chris' Outrageous Cheesecake™
Mango Key Lime Cheesecake
Fresh Banana Cream Cheesecake
Adam's Peanut Butter Cup Fudge Ripple
Chocolate Tuxedo Cream™ Cheesecake
Layers of Our Fudge Cake, Chocolate Cheesecake, Vanilla Mascarpone Mousse and Chocolate.
Like 134
Pin it
Contact Us Charitable Giving Investors Press Terms of Use Privacy Policy Employment Policy Site Map Global Development Supply Chain Sustainability
© 2002-2014 TCF Co. LLC. All Rights Reserved.

```java
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());
        }
    }
}
```



http://seleniumexamples.com/blog/examples/cheesecake/

```java
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

Language	#Repositories
Java	4287
Python	1800
Ruby	1503
C#	558
JavaScript	237

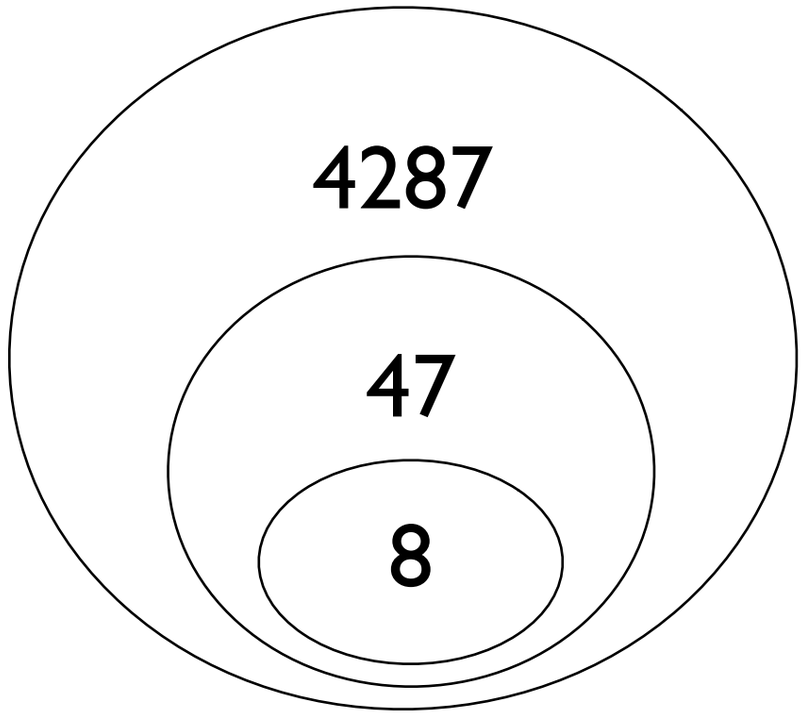
Refinement

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?

Ekeko

... 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

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

([?s_1 ?e_2] ... [?s_n ?e_n])

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

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
ekeko predicates evaluated in the current version

`(in-git-info [current] &conditions)` conditions hold in current version (only git data)

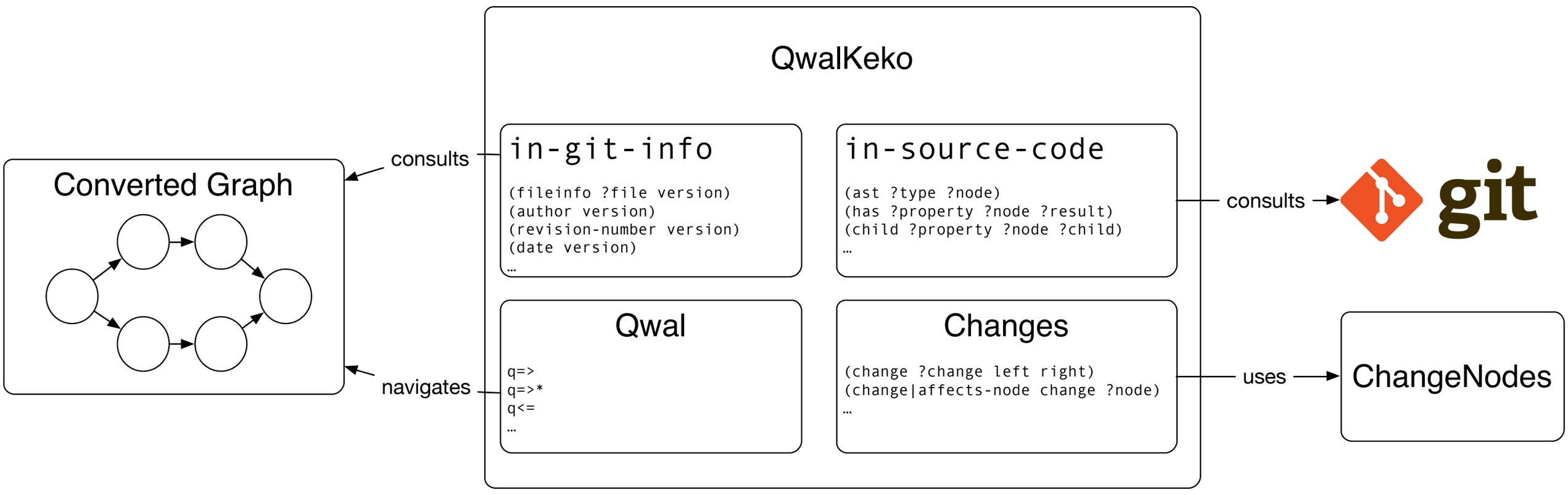
`(in-source-code [current] &conditions)` conditions hold in current version (git data + source)

`q=>` move current version to a successor

`q=>*` skip an arbitrary number of versions

`q<=` move current version to a predecessor

QwalKeko



https://github.com/ReinoutStevens/changenodes

Beat Fluri and Harald C. Gall. Classifying Change Types for Qualifying Change Couplings. In Proceedings of the 14th International Conference on Program Comprehension, 2006.

Identifying Selenium Files

```
1 (defn compilationunit|selenium [?cu]
2   (fresh [?imp ?impname ?str]
3     (ast :CompilationUnit ?cu)
4     (child :imports ?cu ?imp)
5     (has :name ?imp ?impname)
6     (name|qualified-string ?impname ?str)
7     (succeeds (string-contains ?str ".selenium"))))
```

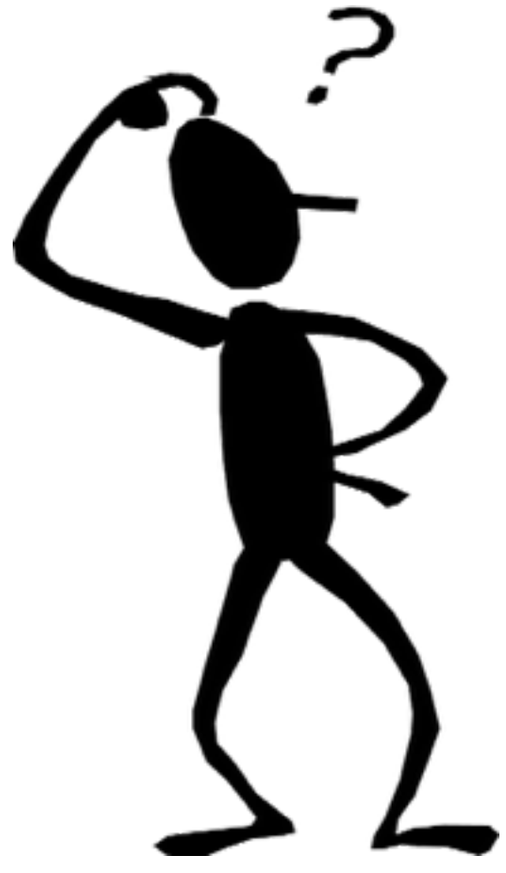
Identifying Selenium Files'

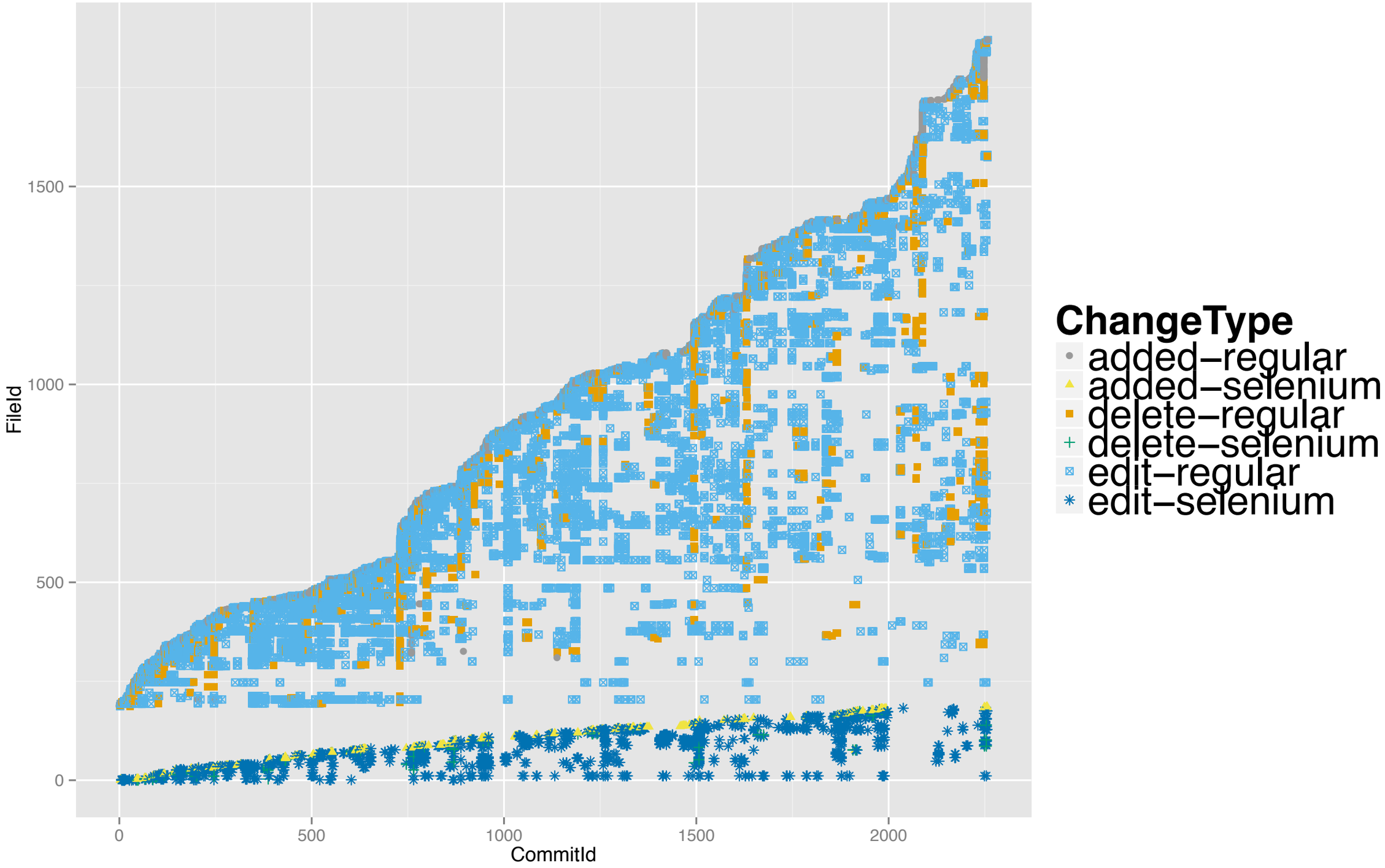
```
1  (defn find-selenium-files [version]
2    (qwalkeko* [?info ?cu]
3      (qwal graph version version []
4        (in-source-code [curr]
5          (fileinfo|edit ?info curr)
6          (fileinfo|compilationunit ?info ?cu curr)
7          (compilationunit|selenium ?cu))))
```

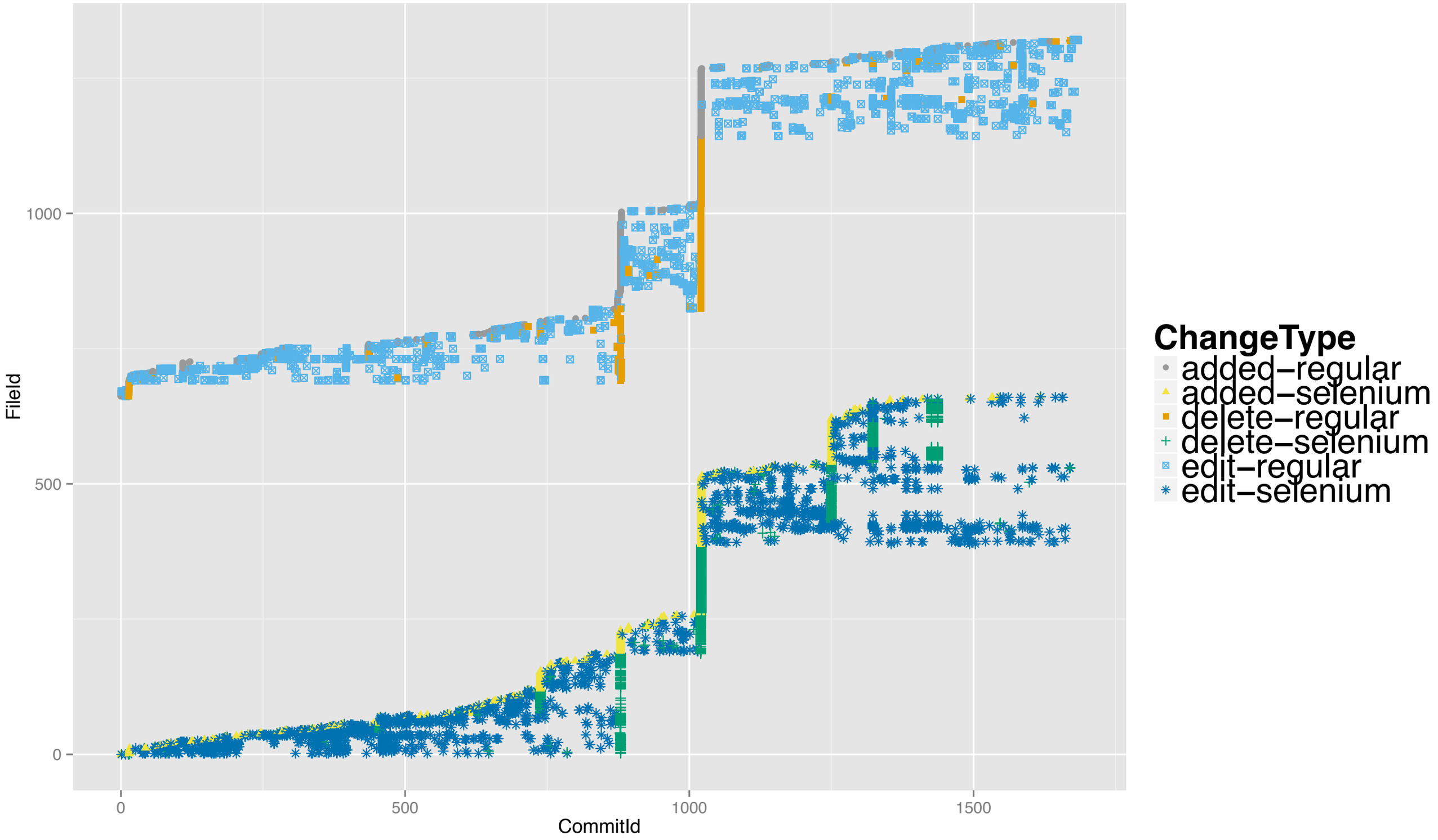
Identifying Selenium Files”

```
1 (map
2   (fn [version]
3     (let [results (find-selenium-files version)]
4       (write-results-to-db results)
5       (ensure-delete version)))
6   (: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?

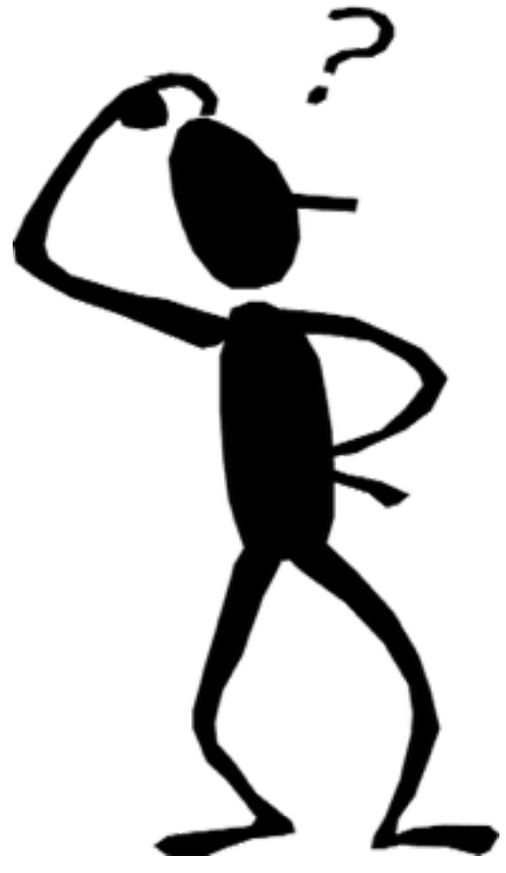



FileId
1500
1000
500
0
0
500
1000
1500
2000
CommitId
ChangeType
added–regular
added–selenium
delete–regular
delete–selenium
edit–regular
edit–selenium


FileId
1000
500
0
0
500
1000
1500
CommitId
ChangeType
added–regular
added–selenium
delete–regular
delete–selenium
edit–regular
edit–selenium


2000
1500
1000
500
0
FileId
0
250
500
750
CommitId
ChangeType
added−regular
added−selenium
delete−regular
delete−selenium
edit−regular
edit−selenium

How are Selenium-based functional tests maintained?
Which parts of a functional test are most prone to changes?



Change Classification

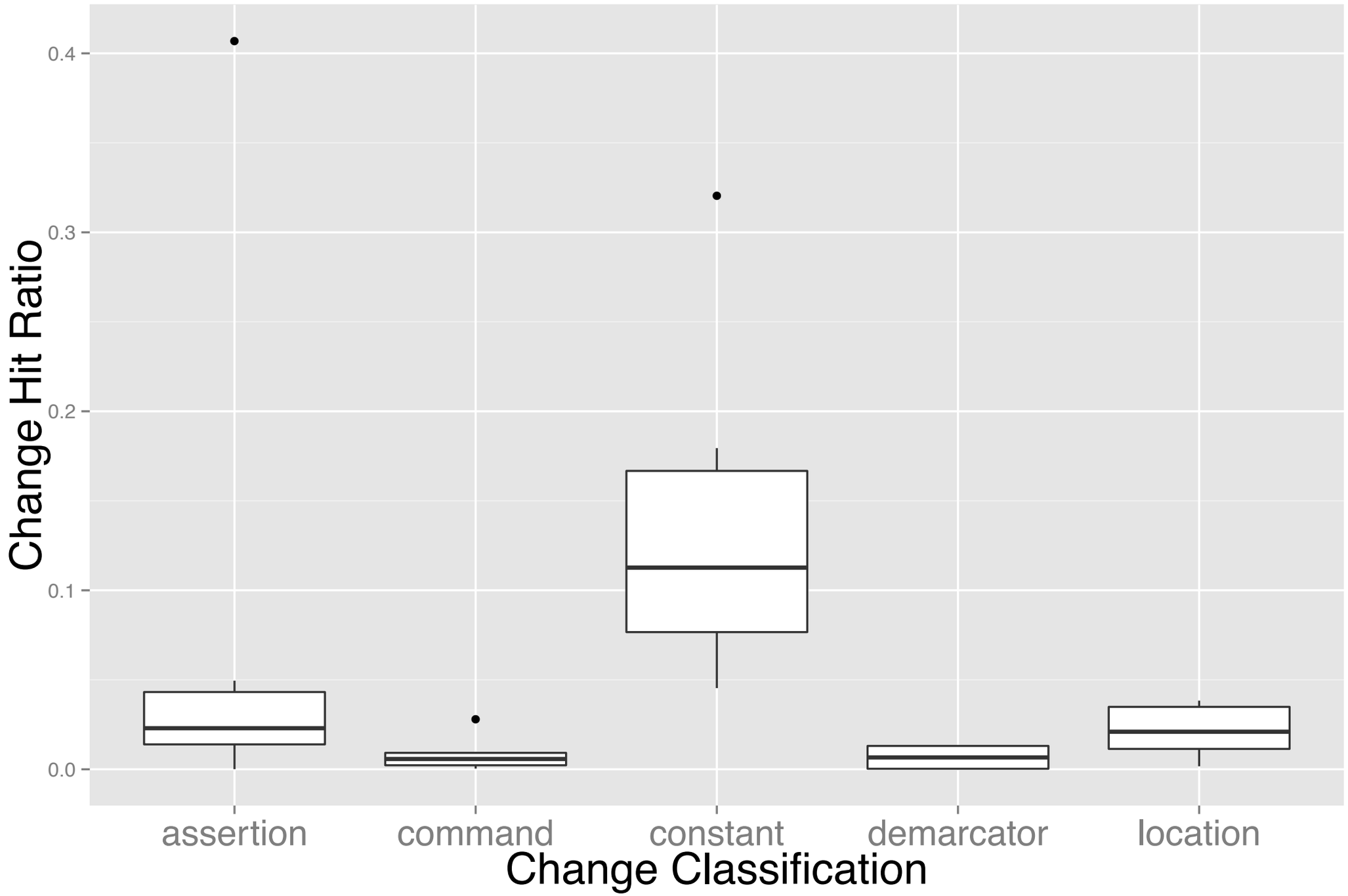
```
public class CheesecakeFactory {

    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<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());
        }
    }
```

- Assertion
- Command
- Constant
- Location
- Demarcator

Classification Query

```
(qwalkeko* [?change ?info ?end ?type]
  (qwal graph version ?end [?left-cu ?right-cu]
    (in-git-info [curr]
      (fileinfo|selenium|edit ?info curr))
    (in-source-code [curr]
      (fileinfo|compilationunit ?info ?right-cu curr))
    q<=
    (in-source-code [curr]
      (compilationunit|corresponding ?right-cu ?left-cu))
    (change ?change ?left-cu ?right-cu)
    (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 change|affects-findBy [change ?find-by]
  (all
    (change|affects-node change ?find-by)
    (conde
      [(methodinvocation|by ?find-by)]
      [(annotation|findBy ?find-by)])))
```


Change Hit Ratio
0.4
0.3
0.2
0.1
0.0
assertion
command
constant
demarcator
location
Change Classification

resteven@vub.ac.be
@ReinoutStevens

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