

Refactoring for Dynamic Languages

Rafael Reia

Instituto Superior Técnico
Universidade de Lisboa

26th Jun, 2015

- 1 Introduction
 - Motivation
 - Objectives
 - Definitions
- 2 Related Work
- 3 Solution
 - Architecture
 - Evaluation

1 Introduction

- Motivation
- Objectives
- Definitions

2 Related Work

3 Solution

- Architecture
- Evaluation

Motivation

```
(define (fibs n)
  (let ((fibs
        (let loop ((previous 0)
                  (current 1)
                  (index 0))
          (if (= index n)
              (list)
              (cons current
                    (loop current
                        (+ previous current)
                        (+ index 1)))))))
    (for ([fib (in-list fibs)])
      (displayln fib))))
```

Motivation

```
(define (print-list fibs)
  (for ([fib (in-list fibs)])
    (displayln fib)))

(define (fibs n)
  (let ((fibs
        (let loop ((previous 0)
                  (current 1)
                  (index 0))
          (if (= index n)
              (list)
              (cons current
                    (loop current
                        (+ previous current)
                        (+ index 1)))))))
    (print-list fibs)))
```

Motivation

```
(define (print-list fibs)
  (for ([fib (in-list fibs)])
    (displayln fib)))

(define (compute-fibonacci n)
  (let loop ((previous 0)
              (current 1)
              (index 0))
    (if (= index n)
      (list)
      (cons current
              (loop current
                    (+ previous current)
                    (+ index 1))))))

(define (fibs n)
  (let ((fibs
         (compute-fibonacci n)))
    (print-list fibs)))
```

Copy Paste - What can go wrong



easyJet

Porto is a gem in sunny Portugal. Famed for its port no less. #NewRoute. For info on booking click here: <http://bit.ly/1KScjXb> — in Porto, Portugal.

Album: Timeline Photos

Shared with: Public

Copy Paste - What can go wrong



easyJet

Kefalonia is a glorious paradise and the largest island in the Ionian Sea. #NewRoute. For info on booking click here: <http://bit.ly/1KScjXb> — in Kefalonia, Greece.

Album: Timeline Photos

Shared with: Public

Objectives

- Correct
- Useful
- Simple to use

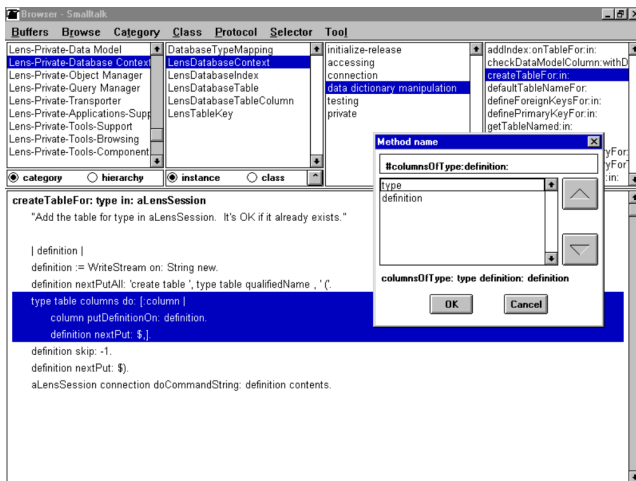
- Refactoring Correctness
- Classification of Refactoring tools

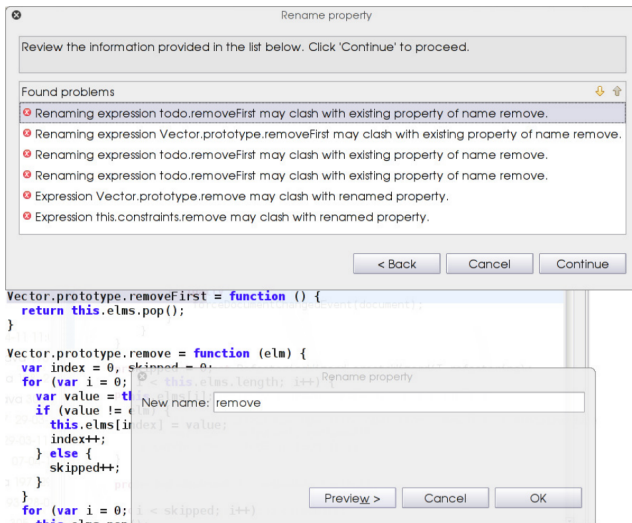
- 1 Introduction
 - Motivation
 - Objectives
 - Definitions
- 2 Related Work
- 3 Solution
 - Architecture
 - Evaluation

How we Refactor

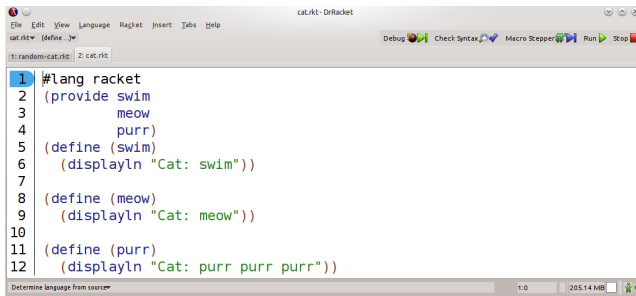
- Refactoring is constant
- Refactoring differs among users
- Refactoring tools are underused

```
1 (define (factorial n)
2   (if (= n 1)
3       1
4       (* n (factorial (- n 1)))))
```





DrRacket's import

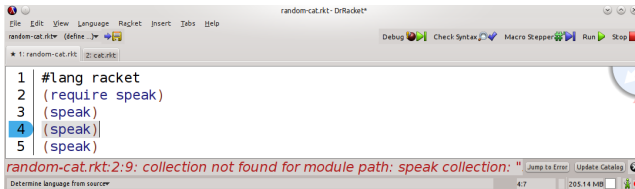
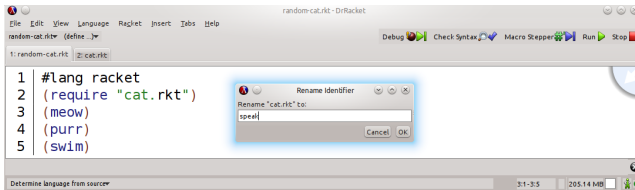


The screenshot shows the DrRacket IDE window titled "cat.rkt - DrRacket". The menu bar includes File, Edit, View, Language, Racket, Insert, Tabs, and Help. Below the menu bar, there are tabs for "cat.rkt" and "(define...)", and a toolbar with buttons for Debug, Check Syntax, Macro Stepper, Run, and Stop. The main text area displays the following Racket code:

```
1 #lang racket
2 (provide swim
3   meow
4   purr)
5 (define (swim)
6   (displayln "Cat: swim"))
7
8 (define (meow)
9   (displayln "Cat: meow"))
10
11 (define (purr)
12   (displayln "Cat: purr purr purr"))
```

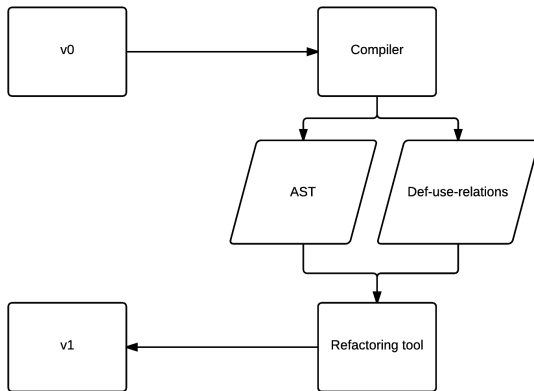
At the bottom of the window, there is a status bar with the text "Determine language from source", a version number "1.0", and a memory usage indicator "205.14 MB".

DrRacket's Rename



- 1 Introduction
 - Motivation
 - Objectives
 - Definitions
- 2 Related Work
- 3 Solution
 - Architecture
 - Evaluation

Architecture



Validation - Extract function

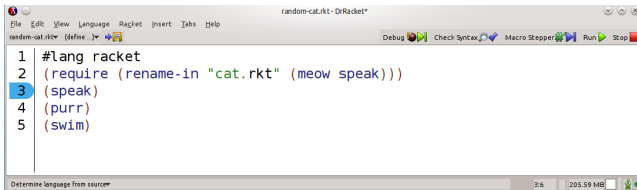
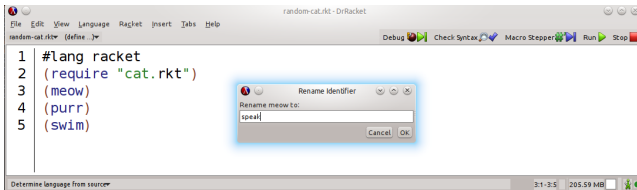
```
1 #lang racket
2 (define (fibs n)
3   (let ((fibs
4         (let loop ((previous 0)
5                   (current 1)
6                   (index 0))
7           (if (= index n)
8               (list)
9               (cons current
10                    (loop current
11                        (+ previous current)
12                        (+ index 1)))))))
13   (for ([fib (in-list fibs)])
14     (displayln fib)))
```

```
1 #lang racket
2 (define (fib-print fibs)
3   (for ([fib (in-list fibs)])
4     (displayln fib)))
5 (define (fibs n)
6   (let ((fibs
7         (let loop ((previous 0)
8                   (current 1)
9                   (index 0))
10          (if (= index n)
11              (list)
12              (cons current
13                   (loop current
14                       (+ previous current)
15                       (+ index 1)))))))
16     (fib-print fibs)))
```

Validation - Extract function

```
1 #lang racket
2 (define (fib-print fibs)
3   (for ([fib (in-list fibs)])
4     (displayln fib)))
5 (define (fib-compute n)
6   (let loop ((previous 0)
7              (current 1)
8              (index 0))
9     (if (= index n)
10         (list)
11         (cons current
12               (loop current
13                     (+ previous current)
14                     (+ index 1))))))
15 (define (fibs n)
16   (let ((fibs
17         (fib-compute n)))
18     (fib-print fibs)))
```

Validation - Rename



Validation - Add prefix

```
1 #lang racket
2 (require plot3d)
3 (define (xyz->pos p)
4   (pos (cx p) (cy p) (cz p)))
5 (define (xyz->dir p)
6   (dir (cx p) (cy p) (cz p)))
7 (define (pos->dir p)
8   (dir (pos-x p) (pos-y p) (pos-z p)))
```

```
1 #lang racket
2 (require (prefix-in pct: plot3d))
3 (define (xyz->pos p)
4   (pct:pos (cx p) (cy p) (cz p)))
5 (define (xyz->dir p)
6   (pct:dir (cx p) (cy p) (cz p)))
7 (define (pos->dir p)
8   (pct:dir (pct:pos-x p) (pct:pos-y p) (pct:pos-z p)))
```

- Refactoring Correctness
- Usability and Simplicity

Thank you
Questions?