

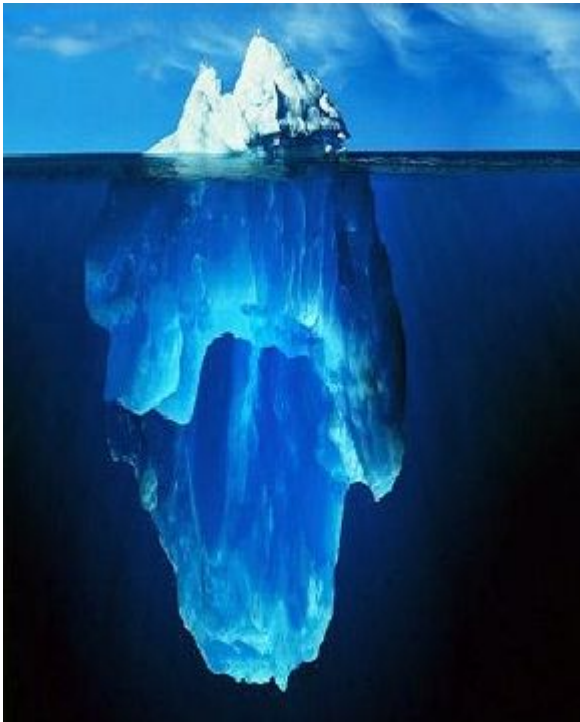
Рефакторинг и анализ Ruby и Rails кода

Андрей Вокин
JetBrains



<http://www.devconf.ru>

Принцип 80-20



- 20% времени - написание нового кода
- 80% времени - поддержание существующего кода

Code that smells

- Runtime errors
- Runtime warnings
- Неиспользуемый код
- Дублированный код
- Большие и сложные методы
- Нарушение code-style соглашений
- Нарушение паттернов фреймворка

Два подхода к оценке качества кода

- Статические инструменты:
Reek, Flay, Flog, Roodi, Saikuro, Metrics_fu
- Инструменты времени выполнения:
Heckle, RSpec, Cucumber, Autotest, RCov, SimpleCov

Статические инструменты

- Проверяют код без его исполнения
- Отсутствуют side-эффекты
- Просты в использовании

При этом:

- Их достаточно сложно реализовать
- Много ложных срабатываний
- Неполное понимание «магии» Rails

Reek

- Имена классов, методов, переменных, модулей
- Использование `instance_of?`, `kind_of?`, `is_a?` вместо полиморфизма
- Дублированный код
- Большие классы, методы
- Большое количество параметров метода
- Вложенные итераторы

Flog

- Присваивания
- Ветвления
- Вызовы

- Балловая система
- На методы с наибольшим количеством баллов стоит взглянуть повнимательнее

Flay

- Ищет дублирование кода
- Анализирует структуру
- Игнорирует разницу в наименовании переменных, констант и пробелах
- Фрагменты кода, указанные Flay - кандидаты на рефакторинг

Roodi

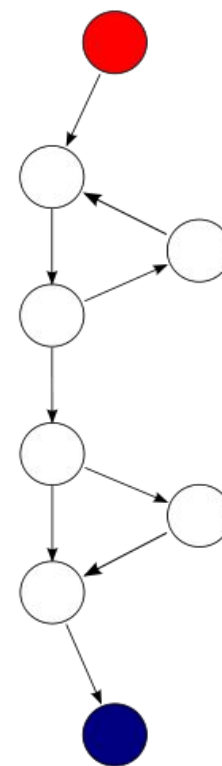
- Присваивание в условиях
- Блоки case без использования else
- Большие модули, классы и методы
- Неправильные имена модулей, классов и методов
- Цикломатическая сложность

Saikuro, Metric_fu

- Saikuro
Цикломатическая сложность
- Metric_fu
Создает отчет по результатам работы Saikuro, Flay, Flog, Reek, Roodi

Что такое цикломатическая сложность?

- $M = E - N + 2P$
- E - количество переходов
- N - количество элементов
- P - количество компонент связности



Runtime инструменты

- Проверяют код, исполнив его
- Учитывают «магию» Rails и все тонкости Ruby

При этом:

- Могут иметь side-эффекты
- Каждый тест работает до первого падения

Runtime инструменты

- Тестирование кода
RSpec, Cucumber, Autotest
- Оценка покрытия кода тестами
RCov, SimpleCov, Heckle

RCov, SimpleCov

- Встраиваются в запуск тестов
- Запоминают строки, исполненные во время работы тестов
- После работы создают отчет о покрытии кода тестами
- Понимают структуру Rails приложения (пропускают config, envoronment...)

Heckle

- Любое логическое изменение кода, полностью покрытого тестами, должно вызывать падение теста

Подход Heckle

- Внести изменение в код
- Запустить тесты
- Проверить, что упал как минимум один тест

Интеграция инструментов оценки качества кода в RubyMine

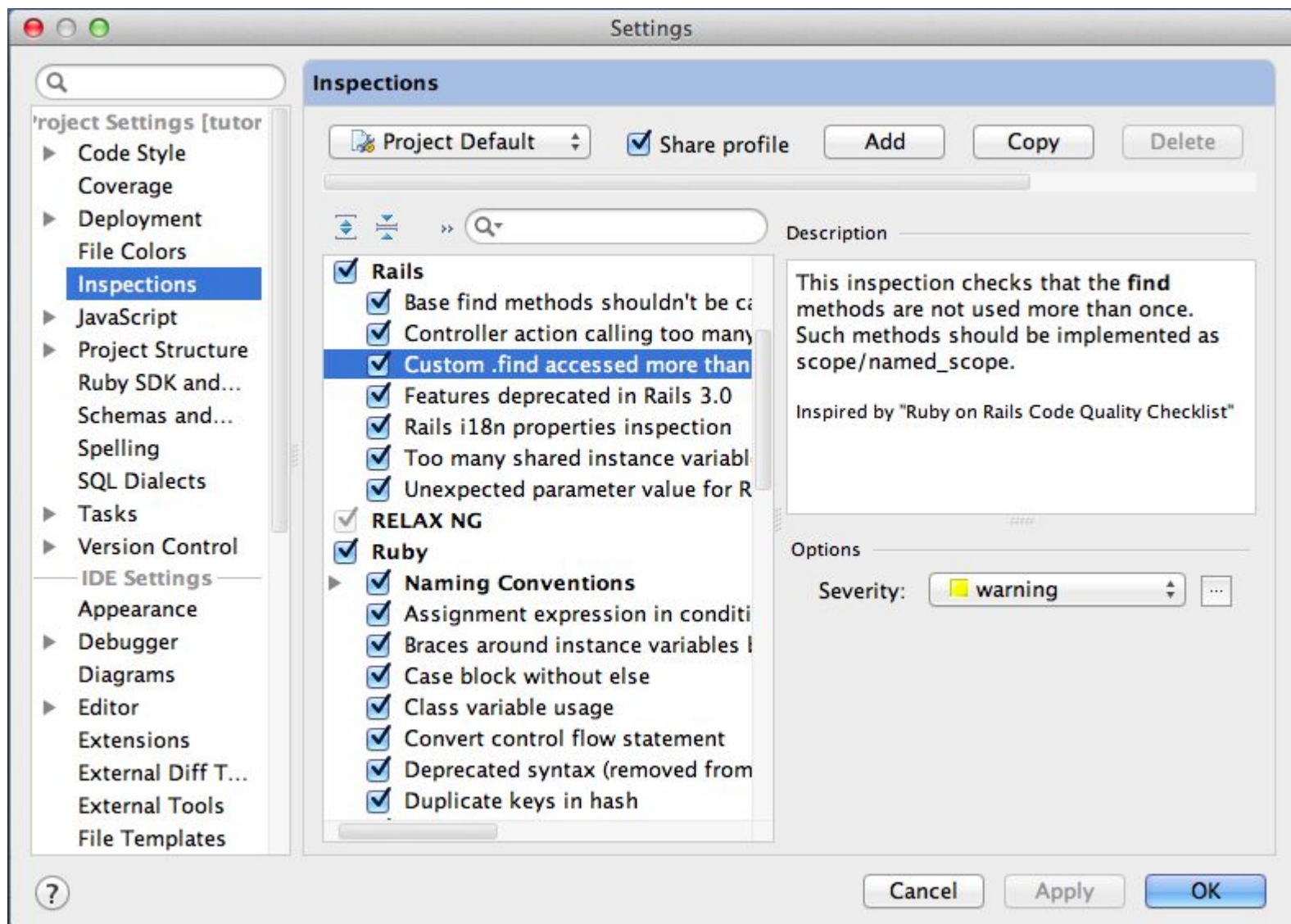
Моментальные инспекции кода

Интеграция тестовых фреймворков (с графическим интерфейсом)

Графическая интеграция SimpleCov

Инспекции кода в RubyMine

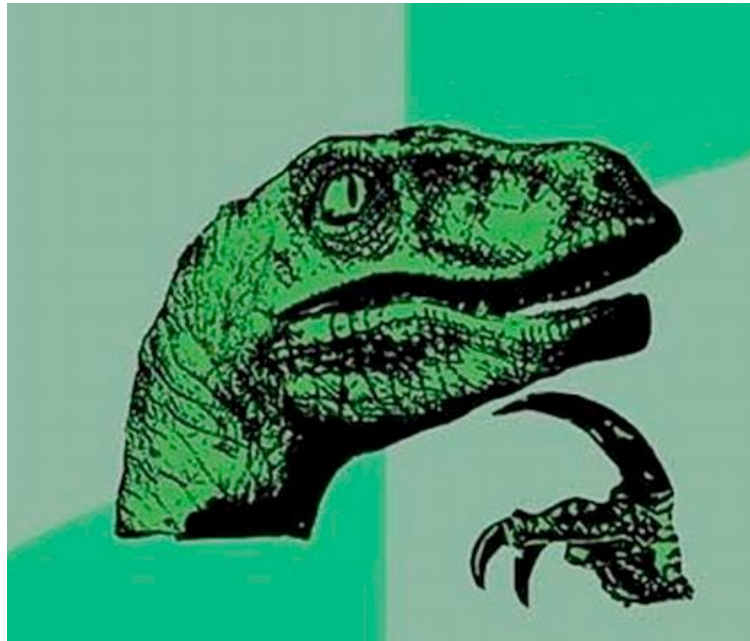
- Учитывают межфайловое взаимодействие
- Понимают DSL Rails
- Не требуют отдельного запуска - работают на лету



The screenshot shows the JetBrains RubyMine IDE interface. The main editor displays a Ruby file named `user_word.rb`. The code includes a `def self.get_for_user` method and a `def create_word_if_necessary` method. A tooltip is visible over the `if !text.nil?` condition in the `create_word_if_necessary` method, with the text "Code block with negative condition more... (⌘F1)". The tooltip is highlighted with a red circle. The IDE's interface includes a Project view on the left, a toolbar at the top, and a status bar at the bottom.

```
46
47 def self.get_for_user(user, text, language_id)
48   result = find_for_user(user, text)
49   if result.nil?
50     word = Word.find_by_text(text)
51     if (word.nil?)
52       word = Word.new(:text => text, :language_id => language_id)
53     end
54     result = UserWord.new(:user => user, :word => word)
55   end
56   result
57 end
58
59 def create_word_if_necessary(text)
60   if !text.nil?
61     word = Word.find_by_text(text)
62     if (word.nil?)
63       word = Word.new(:text => text, :language_id => 1)
64     end
65     self.word = word
66   end
67 end
68
69 def save_with_relations(user, text, new_translations, new_s
70   UserWord.transaction do
71     create_word_if_necessary(text)
72     self.user = user
73   end
74 end
```

Если программно можно искать проблемы в коде...



то можно автоматически и исправлять их

The screenshot shows the JetBrains RubyMine IDE interface. The main editor window displays the file `user_word.rb` with the following Ruby code:

```
46  
47 def self.get_for_user(user, text, language_id)  
48   result = find_for_user(user, text)  
49   if result.nil?  
50     word = Word.find_by_text(text)  
51     if (word.nil?)  
52       word = Word.new(:text => text, :language_id => language_id)  
53     end  
54     result = UserWord.new(:user => user, :word => word)  
55   end  
56   result  
57 end  
58  
59 def create_word_if_necessary(text)  
60   if !text.nil?  
61     # ...  
62   end  
63 end  
64  
65 self.word = word  
66 end  
67 end  
68  
69 def save_with_relations(user, text, new_translations, new_s  
70   UserWord.transaction do  
71     create_word_if_necessary(text)  
72     self.user = user  
73   end  
74 end
```

An IDE tip is displayed over the code, enclosed in a red rounded rectangle. The tip contains the following options:

- Replace 'if !text.nil?' block with 'unless text.nil?' ▶
- Add @param tag ▶

The IDE interface also shows a project structure on the left, a toolbar at the top, and a status bar at the bottom with information like 'Code block with negative condition', '60:11', 'UTF-8', and 'Git: master'.

```
46
47
48 def self.get_for_user(user, text, language_id)
49   result = find_for_user(user, text)
50   if result.nil?
51     word = Word.find_by_text(text)
52     if (word.nil?)
53       word = Word.new(:text => text, :language_id => language_id)
54     end
55     result = UserWord.new(:user => user, :word => word)
56   end
57   result
58 end
59
60 def create_word_if_necessary(text)
61   unless text.nil?
62     word = Word.find_by_text(text)
63     if (word.nil?)
64       word = Word.new(:text => text, :language_id => 1)
65     end
66     self.word = word
67   end
68 end
69
70 def save_with_relations(user, text, new_translations, new_s
71   UserWord.transaction do
72     create_word_if_necessary(text)
73     self.user = user
74   end
75 end
```

Интеграция тестовых фреймворков в RubyMine

- Графический интерфейс
- Симуляция autotest
- Навигация по стэктрейсу
- Отлаживание тестов

The screenshot shows the JetBrains RubyMine IDE interface. The main editor window displays the file `spec_helper.rb` with the following code:

```
6 require File.expand_path("../../config/environment", __FILE__)
7 require 'rspec/rails'
8
9 # Requires supporting ruby files with custom matchers and macros, etc,
10 # in spec/support/ and its subdirectories.
11 Dir[Rails.root.join("spec/support/**/*.rb")].each {|f| require f}
12
13 RSpec.configure do |config|
14   # == Mock Framework
15   #
```

Below the editor, the Run toolbar shows "All specs in: spec". The Console window displays the RSpec log with the following output:

```
Done: 161 of 161 Failed: 4 (21...
Pending: No reason given
161 examples, 4 failures, 155 passed, 2 pending
Finished in 13.52434 seconds
/Users/jetbrains/.rvm/gems/ruby-1.9.2-p290/gems/rake-0.10.4/bin/rake
```

The Test Results window shows a tree view of test results:

- Test Results
 - PagesController
 - SearchController
 - SessionsController
 - TrainingsController
 - POST 'check'
 - unauthorized access
 - authorized access
 - successful attempt

Интеграция SimpleCov в RubyMine

- Отображение покрытия в Project Tree View
- Возможность переключения между разными прогонами

The screenshot shows the RubyMine IDE interface. The top toolbar includes icons for Project, Run, and Settings. The main editor displays the code for `accounts_controller.rb`. The code defines the `AccountsController` class, which inherits from `ApplicationController`. It includes two before_filters: `verify_config` and `verify_users`. The `index` method checks if there are any users; if not, it redirects to `signup`, otherwise to `login`. The `login` method checks for a session and redirects to `admin/dashboard` if one exists, otherwise returns. The `@page_title` is set to include the blog name and the current action.

```
1 class AccountsController < ApplicationController
2
3   before_filter :verify_config
4   before_filter :verify_users, :only => [:login, :recover_pa
5
6   def index
7     if User.count.zero?
8       redirect_to :action => 'signup'
9     else
10      redirect_to :action => 'login'
11    end
12  end
13
14  def login
15    if session[:user_id] && session[:user_id] == self.current
16      redirect_back_or_default :controller => "admin/dashboa
17    return
18  end
19
20  @page_title = "#{this_blog.blog_name} - #{_('login')}"
```

The bottom panel shows the Run console with the command `Run spec`. The output indicates that all tests passed: `Done: 1230 of 1230 (211.953 s)`. The Test Results pane shows a list of controllers, all with green status icons, indicating successful test runs.

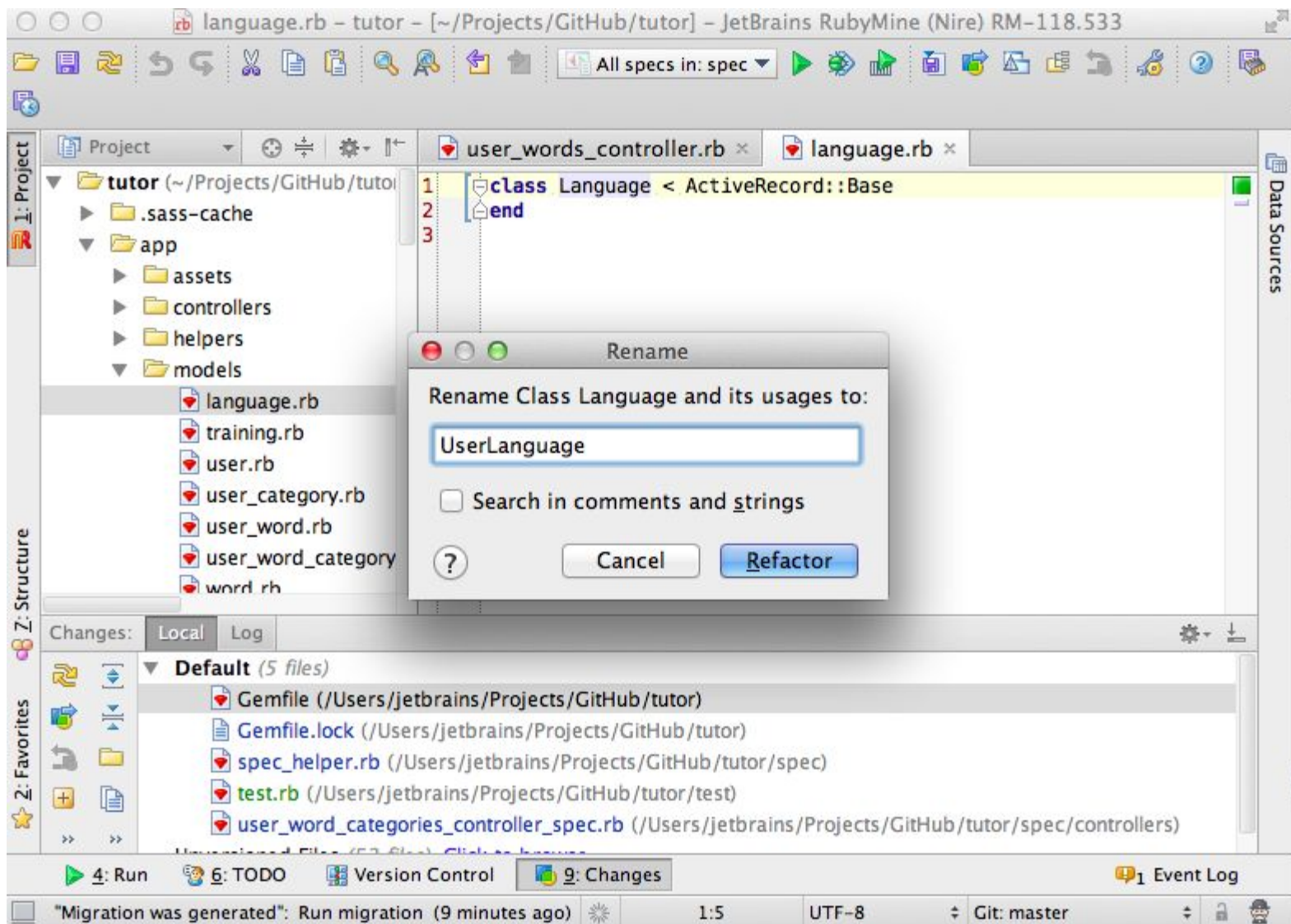
- AccountsController
- Admin::CacheController
- Admin::CategoriesController
- Admin::ContentController
- Admin::DashboardController
- Admin::FeedbackController
- Admin::PagesController

Рефакторинг с RubyMine

- Рефакторинги «понимают» Rails
- Можно откатить результат рефакторинга, минуя контроллер версий

Rename с RubyMine

- Rename локальной или глобальной переменной - это просто!
- Как насчет переименования Rails модели?



The screenshot shows the JetBrains RubyMine IDE interface. The main editor displays a Ruby class definition for `Language` in `user_words_controller.rb`:

```
1 class Language < ActiveRecord::Base
2 end
3
```

The `Find Refactoring Preview` window is open, showing a tree view of usage locations. The selected item is `@languages = Language.all` in `UserWordsController#new`. The preview shows the following code snippet:

```
5 before_filter :set_active_tab
6
7 def new
8   @title = "New word"
9   @languages = Language.all
10  @user_word = UserWord.new
11  @user_word.word = Word.new
12  @user_word.word.text = params[:word]
13  @languages = Language.all
14  @translations = get_translations(:en,
15  @categories = []
```

The IDE interface includes a Project view on the left showing the file structure, a Data Sources panel on the right, and a bottom status bar with various toolbars and a notification area.

The screenshot displays the JetBrains IDE interface. The top toolbar includes icons for file operations and a dropdown menu showing "All specs in: spec". The left sidebar shows the "Project" view with a tree structure of the application, including folders like ".sass-cache", "app", "assets", "controllers", "helpers", and "models", and files like "training.rb", "user.rb", "user_category.rb", "user_language.rb", "user_word.rb", and "user_word_category". The main editor window shows the code for the migration class `20120530095749_rename_language_to_user_language.rb`. The code is as follows:

```
1 class RenameLanguageToUserLanguage < ActiveRecord::Migration
2   def up
3     rename_table :languages, :user_languages
4   end
5
6   def down
7     rename_table :user_languages, :languages
8   end
9 end
10
```

Below the editor, the "Run" tab shows the command executed: `tutor: generate script`. The output of the command is displayed in the console:

```
./Users/jetbrains/.rvm/rubies/ruby-1.9.2-p290/bin/ruby -e $stdout.sync=true;$stderr.sync=true;
... active_record
... create db/migrate/20120530095749_rename_language_to_user_language.rb
Process finished with exit code 0
```

The bottom status bar shows the current state: "Migration was generated": Run migration (moments ago), 1:1, UTF-8, Git: master, and 2 Event Log.

Резюме

- Используйте следующие статические инструменты для проверки вашего кода:
Reek, Flay, Flog, Roodi, Saikuro, Metrics_fu
- Не забывайте про тесты:
Heckle, RSpec, Cucumber, Autotest, RCov, SimpleCov
- Попробуйте RubyMine:
<http://jetbrains.com/ruby>

Вопросы?