Cutting And Packing Algorithms Research Framework

http://caparf.googlecode.com

28 декабря 2009, УГАТУ

Денис Назаров

Проблемы при разработке алгоритмов раскроя и упаковки

- Детали алгоритмов зачастую не описываются в статьях
- Сложно получить/сгенерировать тестовые данные других авторов
- Различные конфигурации компьютеров при тестировании
- Сравнительно небольшое количество обзорных статей

Фреймворк для исследования алгоритмов раскроя и упаковки

CAPARF состоит из:

1. Определений для каждого типа задач
2. Алгоритмов для решения конкретного типа или целого класса задач (нижние/верхние границы, точные методы)
3. Тестовых данных и их генераторов

CAPARF предоставляет:

1. Удобную модель «сценария тестирования»
2. Различные генераторы результатов (например, для статьи в TeX или презентации)
3. Возможность разрабатывать свои алгоритмы, используя существующие наработки, и сравнивать их с алгоритмами других авторов

Преимущества CAPARF

1. Содержит много алгоритмов и тестовых данных для задач раскроя и упаковки
2. Общедоступность: OpenSource проект, размещенный на http://code.google.com под лицензией GPLv3.
3. Кроссплатформенность: написан на Java
4. Унифицированность: алгоритмы для одного и того же типа задач имеют одинаковый интерфейс

Определение задачи в CAPARF

Определение задачи состоит из следующих «типов»:

- `Input` - входные данные для задачи
- `Output` - выходные данные для задачи
- `OutputVerdict` - результат проверки выходных данных

Для тройки `<Input,Output,OutputVerdict>` определяется класс `OutputVerifier`, который и будет осуществлять проверку выходных данных

Пример 1: простейший алгоритм

```
package com.googlecode.caparf.examples;

import java.util.ArrayList;
import java.util.List;

import com.googlecode.caparf.framework.base.Algorithm;
import com.googlecode.caparf.framework.spp2d.Input;
import com.googlecode.caparf.framework.spp2d.Output;

public class OneLineSample extends Algorithm<Input, Output> {
  @Override
  public Output solve(Input input) {
    List<Output.Point2D> placement =
        new ArrayList<Output.Point2D>(input.getRectangles().size());
    int currentHeight = 0;
    for (Input.Rectangle rect : input.getRectangles()) {
      // Размещаем каждый последующий проямугольник на предыдущем
      Output.Point2D lowerLeftPoint = new Output.Point2D();
      lowerLeftPoint.x = 0;
      lowerLeftPoint.y = currentHeight;
      placement.add(lowerLeftPoint);
      // Увеличиваем текущую высоту на высоту прямоугольника
      currentHeight += rect.height;
    }
    return new Output(placement);
  }
}
```

Пример 2: сценарий тестирования

```
// Создаем сценарий тестирования
Scenario<Input, Output, OutputVerdict> scenario =
    new Scenario<Input, Output, OutputVerdict>();

// Добавляем алгоритмы в сценарий
scenario.addAlgorithms(
    new SimpleFit(ItemOrder.NEXT_ITEM, PlacementStrategy.DEFAULT),
    new SimpleFit(ItemOrder.NEXT_ITEM, PlacementStrategy.SHIFT_RIGHTMOST_ITEM),
    new SimpleFit(ItemOrder.FIRST_FIT, PlacementStrategy.DEFAULT),
    new SimpleFit(ItemOrder.FIRST_FIT, PlacementStrategy.SHIFT_RIGHTMOST_ITEM));

// Создаем тестовые данные Berkey и Wang и добавляем их в сценарий
InputSuite<Input> berkeyWangSuite = new InputSuite<Input>()
    .addAll(BerkeyWangGenerator.getReferenceInstances(Type.CLASS_I))
    .addAll(BerkeyWangGenerator.getReferenceInstances(Type.CLASS_II))
    .addAll(BerkeyWangGenerator.getReferenceInstances(Type.CLASS_III))
    .addAll(BerkeyWangGenerator.getReferenceInstances(Type.CLASS_IV))
    .addAll(BerkeyWangGenerator.getReferenceInstances(Type.CLASS_V))
    .addAll(BerkeyWangGenerator.getReferenceInstances(Type.CLASS_VI));
scenario.addInputSuite(berkeyWangSuite);

// Устанавливаем проверочный «скрипт» для сценария
scenario.setVerifier(new OutputVerifier());

// Выполняем сценарий тестирования
CaparfCore<Input, Output, OutputVerdict> invoker =
    new CaparfCore<Input, Output, OutputVerdict>();
invoker.run(scenario);
```

Вклад в разработку CAPARF

Разработке CAPARF можно помочь:

1. Добавив реализацию уже опубликованного «хорошего» алгоритма
2. Добавив определения для нового типа задачи
3. Добавив или указав улучшения/исправления в самом фреймворке

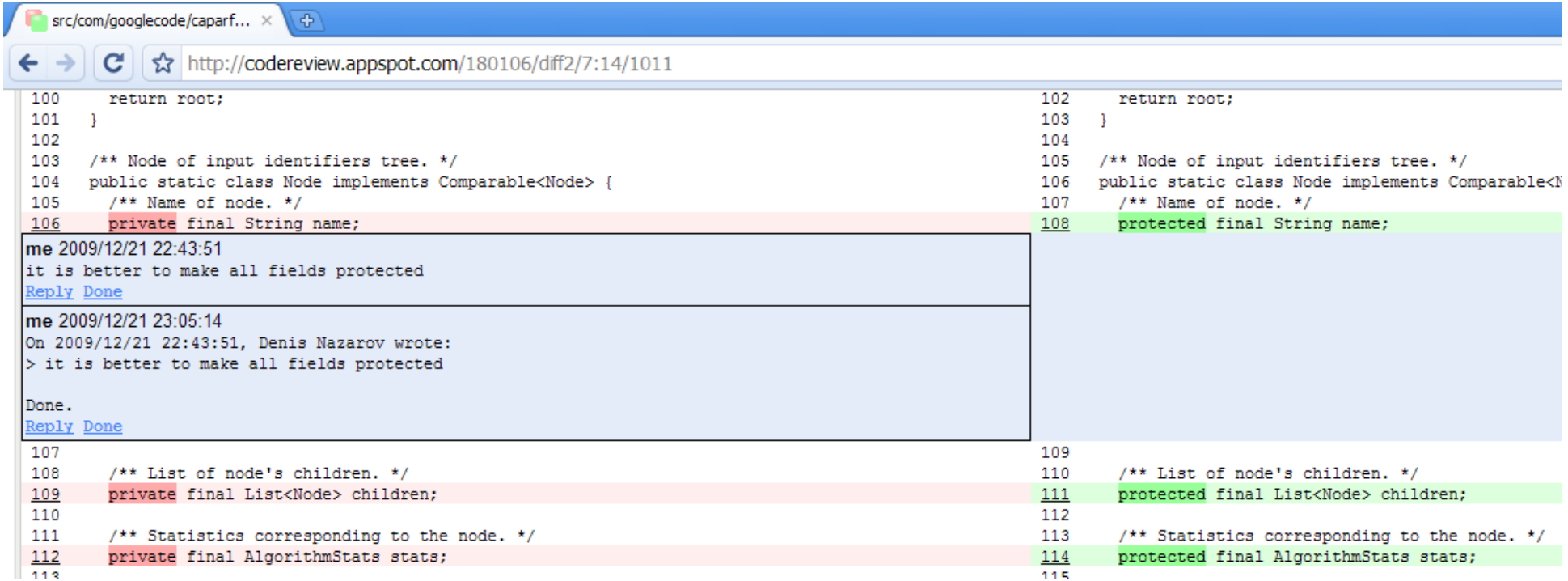
Роли в разработке CAPARF:

contributor — создает различные изменения и отправляет их на code-review (рецензирование),

developer — кроме функций contributor'а также осуществляет code-review предлагаемых изменений, заносит изменения в репозиторий

Code-review в CAPARF

Все изменения в CAPARF проходят code-review при помощи Rietveld Code Review Tool (например, http://codereview.appspot.com/180106/show).



Вопросы ?

Спасибо за внимание!