УДК 004.05

JACOCO КАК ИНСТРУМЕНТ ДЛЯ ОТЧЕТА О ПОКРЫТИИ КОДА

Богословский Д.В., студ., Казаков В.Е., к.т.н., доц., Соколова А.С., ст. преп.
Витебский государственный технологический университет
г. Витебск, Республика Беларусь

В настоящее время существует инструментарий, позволяющий проанализировать, в какие строки были вхождения во время проведения тестирования, благодаря чему можно значительно увеличить покрытие, добавив новые тесты для конкретных случаев, а также избавиться от дублирующих тестов. Проведение такого анализа кода и последующая оптимизация покрытия достаточно легко реализуется в рамках тестирования белого ящика при модульном, интеграционном и системном тестировании; при тестировании же черного ящика задача становится довольно дорогостоящей, так как требует много времени и ресурсов на установку, конфигурацию и анализ результатов работы как со стороны тестировщиков, так и разработчиков.

Существует несколько технологий покрытия с открытым исходным кодом для Java. Большинство из них специально подходят для конкретного инструмента и не предлагают документированный API, который позволяет встраивать их в разные контексты. Двумя лучшими и широко используемыми инструментами с открытым исходным кодом являются EMMA и Cobertura. Оба инструмента больше не поддерживаются оригинальными авторами и не поддерживают текущие версии Java. Из-за отсутствия регрессионных тестов обслуживание и добавление функций затруднены.

JaCoCo предоставляет новую стандартную технологию анализа покрытия кода в средах на основе Java VM. Основное внимание уделяется созданию легкой, гибкой и хорошо документированной библиотеки для интеграции с различными инструментами сборки и разработки. Задачи Ant, подключаемый модуль Maven и подключаемый модуль EclEmma Eclipse предоставляются в качестве справочных сценариев использования. Многие поставщики инструментов и проекты с открытым исходным кодом интегрировали JaCoCo в свои инструменты.

JaCoCo использует набор различных счетчиков для расчета показателей покрытия. Все эти счетчики получены из информации, содержащейся в файлах классов Java, которые в основном являются инструкциями байт-кода Java и информацией отладки, необязательно встроенной в файлы классов. Этот подход позволяет эффективно использовать инструменты и анализировать приложения на лету, даже если исходный код недоступен. В большинстве случаев собранная информация может быть преобразована обратно в исходный код и визуализирована до уровня детализации строки. Во всяком случае, у этого подхода есть ограничения. Файлы классов должны быть скомпилированы с отладочной информацией для расчета покрытия на уровне линии и обеспечения выделения источника. Не все конструкции языка Java могут быть напрямую скомпилированы в соответствующий байт-код. В таких случаях компилятор Java создает так называемый синтетический код, который иногда приводит к неожиданным результатам покрытия кода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. JaCoCo – Documentation [Электронный ресурс]. – Режим доступа: https://www.jacoco.org/jacoco/trunk/doc. – Дата доступа: 21.04.2020.

2. GitHub - JaCoCo [Электронный ресурс]. – Режим доступа: https://github.com/jacoco/jacoco. – Дата доступа: 21.04.2020.

3. Intro to JaCoCo [Электронный ресурс]. – Режим доступа: https://www.baeldung.com/jacoco. – Дата доступа: 21.04.2020.

УДК 685.34.05:685.011.56

АЛГОРИТМ ДЕЛЕНИЯ ДУГИ НА УЧАСТКИ

Буевич Т.В.[1], к.т.н., доц., Буевич А.Э.[2], к.т.н., доц.

[1]Витебский государственный технологический университет
г. Витебск, Республика Беларусь
[2]Витебская ордена «Знак Почета» государственная академия ветеринарной медицины
г. Витебск, Республика Беларусь

Разработан алгоритм деления траектории в виде дуги на узлы (точки) на заданном расстоянии n_0 друг от друга. На рисунке 1 изображена расчетная схема алгоритма разделения дуги на участки равной длины. Представленная в векторной форме дуга описывается координатами конечных точек $1(x_1, y_1)$ и $2(x_2, y_2)$ и коэффициентом кривизны дуги r. На рисунке 1 обозначены также: $l_Д$ – длина дуги, l – расстояние между точками 1 и 2, Δl – уточненное расстояние между точками P_i, R – радиус дуги, F – центральный угол.

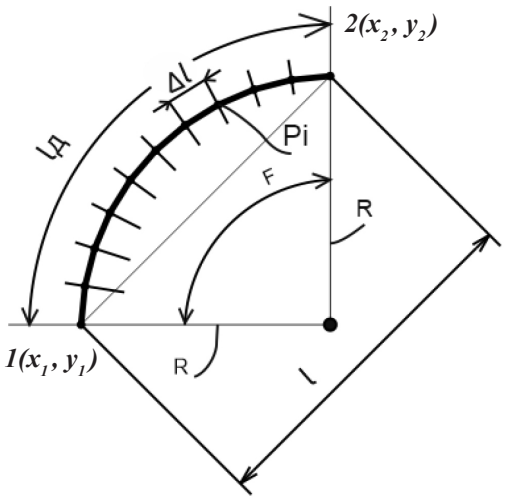

Рисунок 1 – Расчетная схема

Расстояние между точкам 1 и 2 определяется по выражению:

$$l = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}.$$

По коэффициенту кривизны r вычисляется радиус дуги R и затем центральный угол F:

$$R = \frac{r*l+l/r}{4};$$
$$F = \operatorname{acos}\frac{(2*R^2-l^2)}{2*R}.$$

Рассчитывается длина дуги $l_Д$ и находится целое число N отрезков длины n_0, которые помещаются в заданной длине:

$$l_Д = F \cdot R;$$
$$N = [l_Д / n_0].$$

Определяется угловой шаг Δf и приращения по координатным осям $\Delta x, \Delta y$:

$$\Delta f = F / N;$$
$$\Delta x = R \cdot \cos(\Delta f);$$
$$\Delta y = R \cdot \sin(\Delta f).$$