§1 ПОКАЗАТЕЛИ КАЧЕСТВА И ПОВЫШЕНИЕ НАДЕЖНОСТИ ПРОГРАММНЫХ СИСТЕМ

Малыхин А. Ю., Слюсарь В. В.

К ВОПРОСУ О ВОЗМОЖНОСТЯХ ОТЛАДКИ И ТЕСТИРОВАНИЯ ПРИЛОЖЕНИЯ ДЛЯ ОС ANDROID НА ПРИМЕРЕ ПРИЛОЖЕНИЯ ДЛЯ ИЗМЕРЕНИЯ ХАРАКТЕРИСТИК ЭЛЕКТРОТРАНСПОРТА

Аннотация: *На примере программы для измерения характеристик электротранспорта рассмотрены возможности отладки и осуществления логгирования в интегрированной среде программирования Eclipse IDE, в процессе разработки программного приложения для мобильных устройств, использующих операционную систему Android. Продемонстрирован практический пример использования классического метода тестирования «черный ящик» для разрабатываемого Android-приложения. Приведены результаты краткого исследования имеющейся информации по специфичной для ОС Android возможности отправления обнаруженных ошибок в программе непосредственно разработчику и ее практического применения. Экспериментальное получение данных, исследование через разработку ПО (сначала неверный результат, затем тестирование полученного результата, правка исходных данных и программного кода, и, как следствие, работоспособный программный продукт). Приведены специфические методы тестирования программного обеспечения, которые применимы только к программам для ОС Android, а также к программам, получающим и обрабатывающим данные от датчиков электротранспорта. В итоге, рассмотрены отладка, логгирование, тестирование методами белого и черного ящика, отправка отчетов об ошибках. Также показаны примеры данных, которыми можно проверить корректность выполнения различных частей программы для отображения характеристик электротранспорта.*

Ключевые слова: *Тестирование, ОС Android, Eclipse IDE, Характеристики электротранспорта, Приложение, Черный ящик, Белый ящик, Мобильные устройства, Отладка, Отчеты об ошибках*

Использование отладчика в Eclipse IDE для разработки Android приложений

В Eclipse IDE имеется мощный отладчик, который активно используется для разработки

DOI: 10.7256/2305-6061.2014.3.13340

приложений для ОС Android.

Его основные функции:

- Установка точек останова в местах выполнения программы, которые вызывают вопросы, например, возникшая ошибка в своем логе указала, что ошибка возникла в этой строчке. Рационально затем в этой или предыдущей строчке программы установить точку останова и просмотреть значения переменных, и, при необходимости, выполнить программу по шагам;
- Точки останова могут быть условными (выполняться по условию), или быть привязанными к конкретному типу исключения. Например, точка останова может быть привязана к исключению ArrayIndexOutOfBoundsException, и, когда в любом месте выполняющейся в режиме отладки программы произойдет попытка чтения элемента массива с несуществующим номером, то Eclipse автоматически переключится в режим отладки и покажет строчку, где это исключение возникло. Вместе с этим можно просмотреть и значения других элементов;
- Можно отслеживать значение выражения, заданного извне (человеком, выполняющим отладку). Например, можно проверять на равенство две переменных. При пошаговом выполнении результат будет выводиться в специальном поле.

Использование логгирования

Дополнительно при разработке приложений применяется логгирование – запись в специальный журнал сообщений, называемый логом, сообщения о работоспособности или неработоспособности программы. Инструмент, который работает с этими журналами, называется LogCat [1, с. 74]. Он изображен на рис. 2.6.

В Android API есть специальный класс android.util.Log для выполнения записи в журнал сообщений [2]. Класс Log позволяет разбивать сообщения по категориям в зависимости от важности. Для разбивки по категориям используются специальные методы, которые легко запомнить по первым буквам, указывающие на категорию.

Для записи в журнал сообщений в Android имеются несколько уровней «опасности».

- Log.e() - ошибки (error);
- Log.w() - предупреждения (warning);
- Log.i() - информация (info);
- Log.d() - отладка (debug);
- Log.v() - подробности (verbose);
- Log.wtf() - очень серьезная ошибка (What a Terrible Failure) [3].

В первом параметре метода используется строка, называемая тегом. Обычно при-

DOI: 10.7256/2305-6061.2014.3.13340

нято объявлять глобальную статическую строковую переменную TAG в начале кода:

private static final String TAG = “PowerWatcherLog”;

Далее в любом месте программы программист вызывает нужный метод журналирования с этим тегом:

Log.i(TAG, «Сообщение для записи в журнале»);

Тестирование программного приложения

Тестирование несет перед собой задачу обеспечить безошибочную и бессбойную работу приложения методом обнаружения различных ошибок, чтобы в дальнейшем их исправить.

Существует два основных подхода к тестированию.

Тестирование программы как «черный ящик»

Программа рассматривается как черный ящик. Тестовые данные используются только в соответствии со спецификацией программы, то есть без учета знаний о ее внутренней структуре [4, с. 12-13]. При таком подходе обнаружение всех ошибок в программе является критерием исчерпывающего входного тестирования. Последнее может быть достигнуто, если в качестве тестовых наборов использовать все возможные наборы входных данных. В этом случае для исчерпывающего тестирования требуются бесконечные наборы тестов. Однако существуют определенные методологии тестирования, позволяющие из всех возможных тестовых наборов выделить некоторое подмножество тестов, имеющих наивысшую вероятность обнаружения большинства ошибок.

Тестирование программы как «белый ящик»

Стратегия белого ящика, или стратегия тестирования, управляемого логикой программы, позволяет использовать внутреннюю структуру программы. В этом случае программист получает тестовые данные путем анализа логики программы [4, с. 14].

Для тестирования программного приложения, предназначенного для мобильных устройств, более целесообразно использование подхода «черного ящика». При использование этого метода тестирование производится путем непосредственной работы приложения на мобильном устройстве. Считается, что устройство PowerWatcher, а также его библиотека поддержки, несущая код драйвера для него не несет в себе ошибок. Следовательно, при работе на устройстве, в режиме отладки, на дисплее должны отображаться те данные, которые передаются драйвером (то есть данные прошли путь от драйвера до вывода на дисплей не измененными в худшую сторо-

DOI: 10.7256/2305-6061.2014.3.13340

ну). Обозначим сущности, попавшие в «черный ящик» при таком подходе на схеме данных серым прямоугольником (рис. 1).

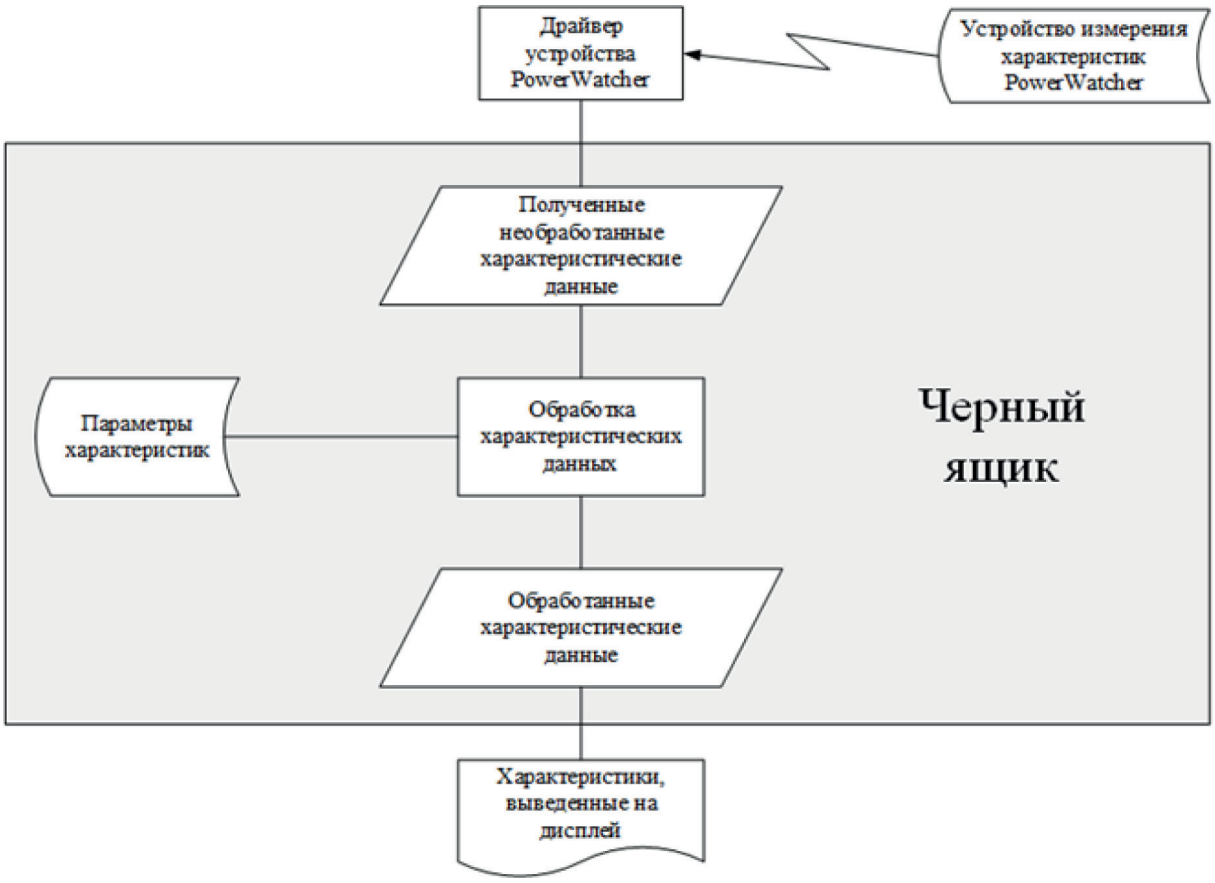

Рис. 1. Иллюстрация тестирования методом «черного ящика»

Для проведения тестирования необходимо обеспечить возможность сравнения данных на выходе драйвера и на дисплее устройства. Данные на выходе драйвера будут выводиться в журнал сообщений LogCat с помощью логирования и визуально сравниваться с данными, отображающимися на дисплее.

При сравнении следует учесть, что данные за каждую секунду собираются, считается их среднее арифметическое, а затем округляются. Это утверждение справедливо для 17 из 21 характеристики.

Реализация тестирования следующая: в классе главной активности создана константа IS_TEST булевского типа, отвечающая за режим тестирования. Если она выставлена в true, то данные тестирования выводятся в журнал, если в false, то не выводятся.

```
public static final boolean IS_TEST = true;
```

В класс PowerWatcherCacher добавляются два метода, которые формируют строки из значений параметров, находящихся в классе в данный момент. Для удобства параметры в строке расположены в порядке размещения их на дисплее с использованием перево-

DOI: 10.7256/2305-6061.2014.3.13340

да строки. Первый метод выводит данные, предназначающиеся для главной приборной панели, второй метод – для второй приборной панели. Эти данные – данные «поданные на вход программе», т.к они переданы напрямую в этот класс от драйвера PowerWatcher. Выходными данными будут считаться данные, отображенные на дисплее.

Проведем этот тест. Были получены следующие записи в журнале сообщений (показана только часть вывода журнала сообщений):

Test First Fragment: 4.48 -11.769 2224.643
3.803 0.0 69338.55
-165.205 0.007 0.0

Сравним теперь эти данные с выведенными показателями на дисплее (рис. 2).

Рис. 2. Снимок экрана главной панели характеристик в процессе выполнения теста

В процессе сравнения становится понятно, что с учетом округления и усреднения нескольких последовательно полученных значений характеристик, данные из журнала сообщений совпадают с данными, выведенными на дисплей. Следовательно, *тестирование методом «черного ящика» для общего случая пройдено.*

Проверка раскраски фона характеристики в предупреждающий цвет

Для такой проверки была взята одна из характеристик - напряжение батареи, и были принудительно последовательно выставлены значения заряда батареи:

0.15 (рис. 3)
30.0 (рис. 4)
50.0 (рис. 5)

DOI: 10.7256/2305-6061.2014.3.13340

70.0 (рис. 6)
100.0 (рис. 7)

Ниже приведены части снимков экрана, показывающие полученные результаты. Очевидно, что цвет фона окрашивается в необходимый цвет (начиная с красного, заканчивая зеленым, с переходом через желтый). Следовательно, ПП АСД реагирует на изменение входных данных корректно, этот *тест пройден.*

Рис. 3. Уровень заряда - 0.15%

Рис. 4. Уровень заряда - 30.0%

Рис. 5. Уровень заряда - 50.0%

Рис. 6. Уровень заряда - 70.0%

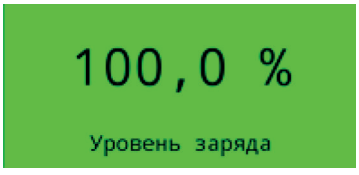

Рис. 7. Уровень заряда - 100.0%

Проверка корректировки некорректных входных данных

Для демонстрации проверки корректировки неверных входных данных ниже рассмотрено тестирование отображения величины «процент рекуперации». Так как это

DOI: 10.7256/2305-6061.2014.3.13340

- проценты, то эта величина должна лежать в отрезке от 0 до 100. Для тестирования отображения некорректных данных были принудительно выставлены следующие значения этой величины:

-130.0 (рис. 8)
-16.145 (рис. 9)
-0.005 (рис. 10)
0.009 (рис. 11)
16.224 (рис. 12)
140.006 (рис. 13)

Результаты тестирования приведены в виде частей снимков экрана, на которых показано отображение величины «процент рекуперации» после корректировки.

Рис. 8. Процент рекуперации: -130.0%

Рис. 9. Процент рекуперации: -16.145%

Рис. 10. Процент рекуперации: -0.005%

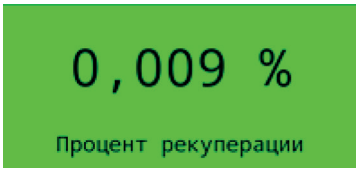

Рис. 11. Процент рекуперации: 0.009%

Рис. 12. Процент рекуперации: 16.224%

DOI: 10.7256/2305-6061.2014.3.13340

Рис. 13. Процент рекуперации: 140.006%

Как видно из снимков экрана, некорректные значения характеристики обрабатываются, и выведенное на дисплей значение не выходит за пределы 0-100 %. Таким образом, *тест на обработку некорректных данных пройден.*

Использование отчетов о сбоях приложения

Рис. 14. Диалоговое окно, возникающее при возникновении необработанной ошибки в приложении

В приложениях Android, которые установлены с магазина приложений Google Play, пользователям доступна отправка отчетов о сбоях приложения [5].

Отправка отчета происходит следующим образом. При возникновении какой-либо ошибки в приложении возникает исключение (Exception), и, если оно должным образом не обработано, приложение завершает свою работу, показывая пользователю специальное системное диалоговое окно (рис. 14).

Если пользователь хочет сообщить об этой ошибке разработчику, он выбирает пункт «отчет» и переходит в форму отправки отчета, где он может выбрать, какую информацию он хочет отправить, и, при желании, указать описание возникшей ошибки.

После нажатия «отправить» отчет будет отправлен разработчику. Программист сможет этот отчет просмотреть и исправить эту ошибку или ошибки.

В отчете находится информация, вполне достаточная для определения, или хотя бы локализации ошибки в программном приложении:

- Версия приложения;
- Версия операционной системы;
- Название устройства;
- Трассировка стека (методы в программном стеке, которые были вызваны до возникновения ошибки);
- Сообщение пользователя (если оно есть);

DOI: 10.7256/2305-6061.2014.3.13340

- Количество повторов за разные временные отрезки;
- Время возникновения ошибки.

Таким образом, описанная функциональность приложений Android позволяет проводить дополнительное тестирование типа «черный ящик» при непосредственном участии пользователей и различных типов устройств. Разработчики получают возможность узнавать практически обо всех ошибках в их программном обеспечении, что повышает качество, стабильность и надежность ПО.

Библиография :

1. Брайан Харди, Билл Филлипс. Программирование под Android. Для профессионалов. СПб.: Питер, 2014. 592 с.
2. Климов А. Отладка // Освой программирование играючи. 2014. URL: http://developer.alexanderklimov.ru/android/theory/debug.php (дата обращения: 17.06.14)
3. Google Inc. Android API Reference // Android Developers. 2014. URL: http://developer.android.com/reference/android/util/Log.html (дата обращения: 17.06.14)
4. Степанченко И. В. Методы тестирования программного обеспечения: Учеб. пособие. Волгоград: ВолгГТУ, 2006. 74 с.
5. Google Inc. Сбои и ошибки ANR (“Приложение не отвечает”) // Google Support. 2014. URL: https://support.google.com/googleplay/android-developer/answer/6083203#export_crashes_and_anrs (дата обращения: 04.07.14)

References:

1. Braian Khardi, Bill Fillips. Programmirovanie pod Android. Dlya professionalov. SPb.: Piter, 2014. 592 s.
2. Klimov A. Otladka // Osvoi programmirovanie igrayuchi. 2014. URL: http://developer.alexanderklimov.ru/android/theory/debug.php (data obrashcheniya: 17.06.14)
3. Google Inc. Android API Reference // Android Developers. 2014. URL: http://developer.android.com/reference/android/util/Log.html (data obrashcheniya: 17.06.14)
4. Stepanchenko I. V. Metody testirovaniya programmnogo obespecheniya: Ucheb. posobie. Volgograd: VolgGTU, 2006. 74 s.
5. Google Inc. Sboi i oshibki ANR (“Prilozhenie ne otvechaet”) // Google Support. 2014. URL: https://support.google.com/googleplay/android-developer/answer/6083203#export_crashes_and_anrs (data obrashcheniya: 04.07.14)

DOI: 10.7256/2305-6061.2014.3.13340