Существует способы улучшения алгоритма для повышения его робастности. Например, можно использовать квадратные или прямоугольные блоки вместо отдельных пикселов изображения. Алгоритм Patchwork является достаточно стойким к операциям усечения, сжатия, изменения гистограммы изображения. Основной недостаток алгоритма – это неустойчивость к геометрическим преобразованиям: сдвигу, повороту, масштабированию. Другим недостатком является малая пропускная способность, что не дает возможности встраивания значительных по размеру цифровых водяных знаков. К примеру, в базовой версии алгоритма для передачи 1 бита скрытой информации необходимо порядка 20 000 пикселов. Таким образом в 20-мегапиксельном изоображении можно передать только 1000 бит, то есть около 120 байт скрытого сообщения.

Список использованных источников:

1. Грибунин, В. Г. Цифровая стеганография / В. Г. Грибунин, И. Н. Оков, И. В. Туринцев. – М.:СОЛОН-Пресс, 2002.–261 с.
2. Семёнов К. П. Алгоритмы встраивания цифровых водяных знаков в растровые изображения / К. П. Семёнов, П. В. Зайцев // Информационная безопастность регионов : научно-практический журнал. – 2011. – №1. – С. 46–50.
3. Bender W. Techniques for Data Hiding / W. Bender, D. Gruhl, N. Morimoto, A. Lu // IBM Systems Journal. – 1996. – Vol. 35.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ПРОСМОТРА РАСПИСАНИЯ ОБЩЕСТВЕННОГО ТРАНСПОРТА НА ПЛАТФОРМЕ XAMARIN С ИСПОЛЬЗОВАНИЕМ УСТРОЙСТВА ТИПА «УМНЫЕ ЧАСЫ»

Жизневский В.С.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Трус В.В. – ст. преподаватель каф. ПОИТ

Все большую популярность набирают устройства типа «умные часы». Некоторые современные модели являются многофункциональными устройствами под управлением полноценных операционных систем. Например, линейка Gear от компании Samsung под управлением операционной системы Tizen. Однако полностью свой потенциал «умные часы» раскрывают при работе в связке со смартфоном. В связи с этим большой перспективой обладают комплексные решения, в состав которых входят как приложение для смартфона, так и приложение для часов.

Не редки ситуации, когда необходимо на ходу составлять маршруты внутри черты города, или же иметь быстрый доступ к актуальной информации движения общественного транспорта, например знать ближайшее время прибытия на остановку автобуса по определенному маршруту. И на данный момент есть приложения для смартфонов, которые предоставляют функции для составления маршрутов, просмотра расписаний движения общественного транспорта. Но использование в некоторых случаях часов дает преимущества, т. к. часы являются более мобильным устройством по сравнению с телефоном. Одним из эффективных сценариев использования часов является отображение наиболее актуальной для пользователя информации. Однако редактирование и ввод информации лучше производить на смартфоне, т. к. он обладает более удобными средствами для ввода информации по сравнению со смарт-часами.

В рамках проекта было принято решение о создании мобильного приложения для просмотра расписания общественного транспорта с использованием устройства типа «умные часы». Помимо основного функционала, в которых входит отображение остановок общественного транспорта на карте, просмотр расписания общественного транспорта, добавление расписаний движения общественного транспорта через остановку в избранное, данное приложение будет подразумевать сценарий использования смарт-часов, а именно отображение расписаний движения общественного транспорта через остановку из списка избранного.

В качестве платформы для части приложения, которая будет работать на стороне смартфона, была выбрана платформа Xamarin, т. к. данная платформа позволяет создавать кросс-платформенные приложения для Android и iOS [1]. В качестве целевых устройств типа «умные часы» были выбраны устройства линейки Gear от компании Samsung под управлением операционной системы Tizen. Согласно отчёту компании Strategy Analytics за 4-й квартал 2018 устройства компании Samsung входят в тройку по количеству проданных, что говорит о популярности моделей данных часов [2]. Для разработки части приложения, работающего на часах будут использованы инструменты Tizen SDK [3]. Тип приложения будет Tizen Web, т. к. приложения такого типа обладают наибольшей совместимостью с устройствами линейки Gear [4].

В качестве компонента отображения ближайшего времени отправления на смарт-часах будет использован виджет.

Предполагаемый сценарий создания и инициализации виджета:

Пользователь синхронизирует список избранного на телефоне и смарт-часах при помощи интерфейса Bluetooth, далее пользователь добавляет виджет в список активных. После, при нажатии на виджет, происходит инициализация, которая подразумевает выбор пользователем расписания движения общественного транспорта через остановку из списка избранного. В результате данных действий на экране виджета будет отображаться ближайшее время отправление транспорта с остановки по маршруту.

Таким образом, данный сценарий использования позволит ещё быстрее получать доступ к наиболее необходимой информации касательно расписаний движения общественного транспорта, что выгодно отличает данное приложение от аналогов.

Список использованных источников:

1. Документация по Xamarin [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.microsoft.com/ru-ru/xamarin.

2. Strategy Analytics Wearable Device Ecosystems (WDE) service report [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.strategyanalytics.com/strategy-analytics/blogs/devices/wearables/wearables/2019/02/28/apple-watch-captures-half-of-18-million-global-smartwatch-shipments-in-q4-2018.

ОБЗОР РЕШЕНИЙ СТАТИСТИЧЕСКОГО АНАЛИЗА КОДА

Жих А.В.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Парамонов А.И. – к.т.н.,доцент

В данной статье рассмотрены проблемы, решаемые с помощью статического анализа кода, а также проанализированы характерные особенности статических анализаторов кода на примере компилятора Roslyn. Проблема статического анализа кода является актуальной, так как языки программирования, как и относящиеся к ним технологии, продолжают развиваться, масштабы проектов увеличиваются, вместе с ними и объем самого кода, а, следовательно, и кодревью усложняется. Статический анализ кода является неотъемлемой частью процесса разработки.

Статический анализ кода — анализ программного кода, который производится без его выполнения. Статический анализ необходимо рассматривать в качестве автоматизированного процесса обзора кода [1]. Кодревью является одним из самых полезных методов для выявления дефектов программного кода. В процессе обзора кода выявляются ошибки, которые могут привести к некорректному поведению. Основным недостатком кодревью является то, что данный процесс трудоёмкий и времязатратным, так как полностью производится человеком, а, следовательно, и дорогостоящий. Говоря о статический анализе, следует подчеркнуть, что он позволяет найти большинство ошибок по мере их появления в коде, а, следовательно, и своевременно их исправить. Данная особенность приводит к улучшению качества самого кода. Следует учитывать и то, что чем раньше ошибка была выявлена, тем меньше будет стоить ее исправление. Статический анализ кода может быть использован, как в своём стандартном виде, с целью поиска дефектов кода, так и с целью обучения либо же более усложненный вариант - проверка кода с точки зрения архитектурного решения. Преимущества статического анализа кода в данных направлениях являются очевидными. Возможность поиска дефектов для них полностью сохраняется только будут усложнены правила распознания того, что является дефектом.

На данный момент находится в использовании широкий спектр анализаторов кода на базе статического анализа, основная специфика которых заключается в том, что они заточены под конкретный язык программирования. Рассмотрим основные особенности реализации статических анализаторов для c# на примере компилятора Roslyn.

Roslyn — это не просто компилятор, а целая платформа, которая разрабатывается корпорацией Microsoft. [2] Она предоставляет возможности для разбора кода и его последовательного анализа. Roslyn также является основой для других анализаторов кода. С помощью данного компилятора можно производить детальный разбор кода для дальнейшего анализа всех поддерживаемых языковых синтаксических конструкций. Коданализ проходит в несколько этапов. Вначале идет сбор необходимых данных, что включает в себя получение проектного решения, затем компиляции и разбор полученных файлов, на основе чего строится неизменяемое дерево синтаксиса и семантическая модель. Roslyn строит дерево синтаксиса на основе программного кода, полученного из проектного файла, состоящее из синтаксических узлов, лексем и дополнительной информации. [3] Полученное дерево используется для анализа конструкций языка. По дереву синтаксиса и происходит перемещение во время анализа, которое осуществляется с помощью рекурсии. Синтаксические узлы содержат основные синтаксические языковые конструкции. На синтаксических узлах основываются