**Особенности тестирования мобильных приложений в целом**

Мобильный пользователь ожидает, что устанавливаемые им приложения просты, интуитивно понятны, работают всегда и везде без сбоев. Если ожидания не оправдываются, то пользователь просто-напросто устанавливает аналогичное приложение от другого автора, которых в сфере мобильных разработок всегда достаточно. Поэтому качество приложения является одним из главных факторов его популярности.

Чтобы понять особенности тестирования мобильных приложений следует принимать во внимание моменты,[принципиально отличающие мобильные приложения от десктопных](http://www.enterra.ru/): специфичность ОС для мобильных платформ, различные компании-изготовители устройств и конфигурации комплектующих, функциональность устройств как коммуникаторов и т.п.

В связи с этими особенностями, подход к [разработке приложений](http://www.enterra.ru/mobile-application-development) и, в частности, тестированию на мобильных устройствах довольно сильно отличается от десктопного. Возникает множество дополнительных важных нюансов и требований, которые необходимо протестировать. Немного осветим основные отличия в некоторых видах тестирования:

**Тестирование обновлений** – частые обновления операционной системы (относительно десктопов) приводят к необходимости обновлять приложение. Обновление должно проходить просто и не требовать от пользователя специфических знаний. Следует так же проверять различные возможные пути установки приложения (Wi-Fi, 3G, установка с ПК, на SD).

**Тестирование интернационализации** позволяет на раннем этапе процесса [разработки мобильного приложения](http://www.enterra.ru/mobile-application-development) убедиться в поддержке культурных особенностей других стран (главным образом, в языковой поддержке). Интернационализация в мобильных приложениях крайне распространена, так как является относительно простым способом серьёзного увеличения целевой аудитории. В процессе могут возникнуть многие специфичные для мобильных платформ проблемы, такие как недостаток свободного пространства на экране.

**Тестирование удобства пользования (usability)**. Этот вид тестирования является одним из важных, так как в условиях высокой конкуренции юзабилити приложения входит в список основных параметров, влияющих на популярность продукта. Позволяет выявить части приложения, которые недостаточно привлекательны или вызывают затруднения в навигации или использовании на сенсорных экранах. Следует так же убедиться, что модель потребления ресурсов приложением соответствует целевой аудитории. Например, приложения-напоминания не должны вызывать чрезмерное потребление энергии. Зачастую это тестирование проводится в виде бета-тестирования.

**Нагрузочное тестирование** подразумевает наблюдение за использованием памяти и системных ресурсов; кроме того, нагрузочное тестирование позволяет выявить “узкие” места в приложении, связанные с производительностью, а также обнаружить опасные утечки памяти.

**Случайное тестирование**(fuzzy testing, “monkey” testing) – приложение должно корректно реагировать на возникновение случайных и непредсказуемых событий. Мобильные устройства чаще других попадают в условия, в которых получают хаотичную бесполезную информацию (например, незаблокированный девайс в кармане), потому приложение должно адекватно реагировать на подобные потоки данных.

**Мультиплатформенное и мультидевайсовое тестирование** – приложение должно правильно работать на всех конфигурациях всех устройств, для которых разрабатывалось. Из-за фрагментации мобильных телефонов задача тестирования всех доступных видов устройств, на которых используются различные сборки ОС, и которые имеют различные разрешения экранов, функционал и аппаратное обеспечение, настолько же важна, насколько трудновыполнима.

**Лабораторное тестирование** – имитация реальных условий качества связи и окружающей среды. Обычно неочевидно, как поведёт себя приложение, например, при нестабильном сигнале Wi-Fi или с нулевым балансом на счету в сети 3G. Данный вид тестирования позволяет проверить подобные случаи.

**Аттестационное тестирование** используется для подтверждения соответствия приложения стандартам, лицензионным соглашениям и условиям использования:

**Android**

* Установочный файл приложения (.apk) должен быть согласован с [Program Policies](http://www.android.com/us/developer-content-policy.html" \t "_blank) .
* В случае публикации обновлённой сборки желательно придерживаться порядка управления версиями (например, принятого порядка нумерации версий).
* Приложение не должно противоречить документу [UIG](http://developer.android.com/guide/practices/ui_guidelines/index.html) .

Так же для отдельных магазинов приложений (Amazon App Store, Samsung Apps, Yandex.Store и подобных) могут существовать свои собственные требования и гайдлайны.

**iPhone**

* Приложение должно иметь уникальное имя.
* Приложение должно принадлежать к определённой категории.
* Нужно предоставить ссылку для осуществления обратной связи с разработчиком.
* Приложение не должно противоречить документу [Human Interface Guidelines](http://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf" \t "_blank).
* Приложение может быть отказано в сертификации по причинам наличия запрещённых материалов, непредвиденных задержек в работе или повторении существующих функций.

**Windows Phone 7-8**

Приложение должно соответствовать всем категориям [app certification requirements](http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184843(v=vs.105).aspx" \t "_blank).

**Перечень моментов, которые должны быть протестированы**

**Размер экрана и touch-интерфейс:**

* Все элементы должны быть такого размера, чтобы пользователь мог однозначно попасть по ним.
* Отсутствие пустых экранов в приложении – пользователь не должен оказываться в ситуации, в которой не очевидно, что сейчас происходит и что делать.
* Следует проверять многократное быстрое нажатие на кнопку – часто при этом может случиться падение приложения. Так же следует проверять мультитач – нажатие на несколько кнопок одновременно.
* Следует проверять наличие или отсутствие «нативных» жестов (pinch-to-zoom, doubletap) – если, например, поддерживается зум части приложения, то должен использоваться жест по умолчанию. А если нет необходимости выделять картинку, то по даблтапу она не должна выделяться.

**Ресурсы устройства:**

* Утечки памяти. Особенно может проявляться на окнах, с большим количеством информации (длинные списки как пример), во время задач с длительным workflow (когда пользователь долго не выходит из приложения), при некорректно работающем кэшировании изображений.
* Обработка ситуаций нехватки памяти для функционирования ОС, когда приложение активно или работает в фоне.
* Недостаток места для установки или работы приложения.
* Отсутствие в некоторых устройствах поддерживаемых приложением функций (3G, SD-карта и т. п.).
* Установка или перенос приложения на карту SD.

**Различные разрешения экрана и версии ОС:**

* Ретина и обычные экраны. На ретина-экранах элементы интерфейса и текст отображаются мельче. Картинки для ретина-экрана могут попасть в неретина версию и тогда будут слишком большими.
* Адаптация приложения к портретной и альбомной ориентациям устройства.
* Версии ОС. Приложение не должно устанавливаться на неподдерживаемые устройства. Обязательна проверка на всех доступных из поддерживаемых девайсов.
* Поддержка необходимых медиа-файлов данной моделью и ОС, потому что отдельные разработчики могут урезать поддержку работы с некоторыми форматами.
* Соответствие, используемых в приложении view, их смысловому назначению и концепциям платформы. Проектные решения, которые имеют смысл для одной платформы, могут выглядеть и быть неуместными в контексте другой платформы.

**Реакция приложения на внешние прерывания**

* Входящие и исходящие SMS, MMS, звонки, оповещения других приложений.
* Выключение устройства, изъятие аккумулятора, разрядка устройства.
* Переход в режим ожидания (в том числе и с защитой паролем). Смена ориентации устройства в режиме ожидания.
* Отключение и подключение провода.
* Отключение и включение сети, Bluetooth, авиарежима, GPS.
* Потеря связи с сервером или прокси (подключение есть, но пакеты не доходят).
* Отключение и подключение SD-карты, дополнительных устройств вроде физической клавиатуры или гарнитуры.
* Зарядка устройства.
* Работа с акселерометром.
* Работа с физической клавиатурой (если в списке поддерживаемых моделей есть такие).

**Платный контент внутри приложения:**

* Соответствие цены и содержимого, заявленного в приложении, тому, что попадает к пользователю.
* Восстановление покупки (обновление приложения).

**Интернационализация (проверять и в портретном, и в ландшафтном режиме!):**

* Проверка корректности перевода.
* Проверка того, что все надписи входят в соответствующие формы, кнопки и т.п.
* Проверка форматов дат, разделителей в числах, специфических особенностей локализации (вроде пробела перед знаком вопроса во французской, верхних индексов “o” и “a” в порядковых числительных в испанской и других нетривиальных моментов).

**Постоянная обратная связь с пользователем:**

* У всех нажимаемых элементов должно быть нажатое состояние (отклик на действие) – благодаря этому пользователь всегда будет видеть, действительно ли нажатие случилось. В Android-приложениях у элементов может быть ещё одно состояние – focused.
* Реакция кнопок на нажатие. Скорость отклика элементов должна быть достаточно высокой. Желательно использовать для проверки этого пункта самые слабые устройства среди поддерживаемых.
* Сообщения при загрузке контента или прогресс-бар.
* Сообщения при ошибке доступа к сети, BT, GPS.
* Наличие понятных сообщений при попытке удалить важную информацию.
* Наличие экрана или сообщения при окончании процесса или игры.
* Наличие и синхронность звуков или вибрации с уведомлениями и другими событиями на экране.

**Обновления:**

* Убедиться, что поддерживаются те же версии ОС, что и предыдущая версия (если новая версия приложения использует новые возможности ОС, то для старых поддерживаемых версий ОС необходимо создание урезанной версии приложения).
* Проверка адекватного обновления (сохраняются все данные пользователя и т. п.).

**Аттестационное тестирование.**  
Различные версии ОС могут использовать отличные соглашения и правила (интерпретация жестов, правила навигации). Следует учитывать это при поддержке приложением нескольких версий ОС.

**Проблемы, с которыми сталкивается тестировщик мобильного приложения**

*Первый и самый важный момент заключается в том, что тестирование мобильных телефонов займёт больше времени, чем вы предполагаете, даже если вы и так допускаете, что оно займёт больше времени.* (Peter-Paul Koch).

Мобильные устройства работают от аккумуляторов, и потому вынуждены автоматически переходить в режим ожидания спустя пару минут бездействия. Это значит, что вам придётся включать телефон перед каждым тестированием, что при одновременном тестировании нескольких телефонов, занимает приличное количество времени. Конечно, на многих устройствах можно отключить автоматическую блокировку (или хотя бы сделать время отключения довольно большим), но желательно всё же работать с самыми распространёнными среди пользователей настройками ОС.

Переход в режим ожидания особенно неприятен при использовании некоторых автоматизированных систем тестирования, требующих время для обработки данных (например, снятие и сравнение скриншотов). Возможно, что телефон перейдёт в режим ожидания прямо посреди теста.  
Затем появляется проблема собственно набора определённого текста (например, адреса тестируемой страницы). Придётся аккуратно вводить длинные тексты в несколько устройств с разными интерфейсами.

Для тестирования в условиях входящих звонков, смс, вам придётся переставлять SIM-карту из одного устройства в другое, для этого зачастую надо вынимать аккумулятор. Особенно остро эта проблема стоит при тестировании особенностей и услуг мобильных операторов на нестандартных и дорогих тарифах. Кроме того, вставив SIM-карту, придётся подождать, пока телефон не включился.

Часто приходится передавать тестируемые виджеты по Bluetooth, что так же довольно утомительно в условиях разнообразности интерфейсов.

Для тестирования GPS придётся вооружиться дополнительным инструментарием от энтузиастов и надеяться, что он работает достаточно похоже на реальные условия.

Для проверки слабого или отсутствующего Wi-Fi и 3G-сигнала обычно приходится либо сооружать лабораторию, либо использовать различные хитрости вроде коробочек из фольги.

Создание скриншотов и видео на мобильных устройствах так же зачастую нетривиальная работа, особенно если тестируется телефон, отключенный от компьютера по условиям теста или по каким-то иным причинам. Например, встроенная возможность снятия скриншота экрана на Android-устройствах появилась сравнительно недавно – с четвёртой версии. А про бесплатный способ снимать видеопоток с экрана Apple-устройства без jailbreak вообще мало кто может сказать что-то вразумительное.

Обычно все тесты производятся на одном устройстве, затем на другом и т. д. Это не так оптимально, как тестирование на всех устройствах одновременно, поскольку не даёт возможности просто сравнить вывод и выполнение тестов на устройствах, однако, ввиду вышеописанных трудностей, это наиболее экономичный по времени способ.

**Захват видео с экрана устройства**

Зачастую возможность записать видео воспроизводимой ошибки очень полезна – она помогает более подробно описать баг и таким образом сэкономить время разработчиков, а то и избежать переоткрытия бага ввиду непонимания.

[Android screen capture](http://sourceforge.net/projects/ashot/) – приложение для передачи видеопотока с экрана Android-устройства на монитор компьютера. Правда программа не умеет пока записывать видео – только автоматически делать серию скриншотов при изменении экрана. Но ничто не мешает использовать десктопные скриншотеры с возможностью записи видео. Требует установленного Android SDK.

[Reflector](http://www.reflectorapp.com/) – платный (от 13$) инструмент для записи видео с iOS-устройств без проводного подключения к десктопу. Работает под Windows и Mac. Есть десятиминутный триал.  
Так же для iPad и iPhone существует приложение Display Recorder, но оно постоянно то исчезает из AppStore, то появляется. На данный момент поиск в AppStore ничего не даёт (обратите внимание, что Display Recorder HD – это другое приложение, не имеющее функции записи экрана). Различные источники называют различные цены за приложения (от двух до десяти долларов).

**Эмуляторы**

Эмулятор – программа, полностью или частично копирующая функционал и поведение устройства или другой программы.

Некоторые из преимуществ использования эмулятора:

* оперативное тестирование приложения, когда целевой мобильный телефон недоступен (или оказывается в дефиците);
* тестирование сложных или опасных сценариев, которые невозможно или не рекомендуется проверять на реальных мобильных телефонах (например, тесты, которые каким-либо образом могут вывести телефон из строя или нарушить условия соглашения с оператором сотовой связи).

Минусы:

* зачастую эмуляторы очень требовательны к ресурсам, так как наиболее качественные из них эмулируют работу приложения с самых нижних уровней;
* то, что приложение работает на эмуляторе, не значит практически ничего, ведь пользователи будут запускать приложения на реальных мобильных телефонах, которые всегда отличаются даже от самых лучших эмуляторов.

Тестирование на целевом мобильном телефоне – это самый верный способ убедиться в правильном функционировании приложения, поскольку вы выполняете приложение на том же аппаратном обеспечении, которое будет у ваших пользователей.

Для всех распространённых мобильных ОС предлагаются бесплатные (для разработчиков) и довольно функциональные эмуляторы. Например, для Android есть официальный Android SDK, который включает в себя эмулятор мобильного устройства, реализующий все аппаратные и программные особенности типичного устройства. Такие же «нативные» решения есть и для iOS, и для Windows Phone.

Так же, конечно, есть и альтернативы:

[MobiOne Developer](http://www.genuitec.com/mobile/features/) – это mobile Web IDE для Windows, помогающее разработчику программировать, тестировать, отлаживать упаковывать и внедрять мобильные веб-приложения на устройства. Имеет много полезных функций вроде просмотра исходников страницы и инспектор кода. Решение платное (на данный момент – 100$), есть 15-дневный триал, поддерживает iOS и Android.

[Electric Mobile Studio 2012](http://www.electricplum.com/studio.aspx) – эмулятор для тестирования веб-приложений iOS под Windows. Поддерживает jQuery Mobile, Mobile Web JavaScript, HTML5. Встраивается в Visual Studio 2012, позволяет эмулировать работу с геолокацией, переключаться между типом устройства «на лету» и прочее. Решение платное (на настоящий момент – около 40$), есть семидневный триал. Так же в сети ещё можно найти более старые бесплатные lite-версии.

[Opera Mobile Emulator](http://www.opera.com/developer/mobile-emulator) и [Opera Mini Simulator](http://www.opera.com/developer/opera-mini-simulator) позволяют протестировать веб-приложение под соответствующим браузером. Оба продукта бесплатны (второй вообще онлайновый – не требует установки на компьютер).

[BlueStacks App Player](http://www.bluestacks.com/app-player.html) позволяет запускать Android-приложения на Windows XP-8 и MacOS. Судя по всему, приложение представляет собой виртуальную машину, которая не эмулирует поведения устройств, но может быть полезна для тестирования функциональности приложения в условиях недоступности других способов.

**Облачные платформы устройств**

Позволяют удалённо протестировать свой продукт на множестве различных устройств, передавая данные о тестирования разработчику. Самые знаменитые – [Perfecto Mobile](http://www.perfectomobile.com/portal/cms/services/) и [Device Everywhere](http://www.keynotedeviceanywhere.com/). Суть этих решений в том, что у них стоит стенд с реальными мобильными устройствами, подключенными по кабелю, и веб-камера, передающая изображения с телефона. Внутрь фотографии телефона вставлено изображение с веб-камеры. Управляется мышкой, либо с помощью автоматического скрипта. Также Perfecto Mobile и Device Anywhere удалённо предоставляют устройства «напрокат». У них стоит множество различных телефонов, и час работы с одним телефоном стоит около $15. Можно оформить подписку. Есть бесплатные триальные версии.

Плюсы:

* почти нет вмешательства в тестируемое приложение;
* один тестовый скрипт используется для тестирования на всех нужных мобильных платформах;
* многообразие устройств (iOS, Android, WP, Blackberry и другие);
* нет ограничений в приложениях с технологической точки зрения – можно тестировать хоть нативные приложения, хоть HTML5.

Минусы:

* большая цена;
* задержки при взаимодействии с телефоном из России.

**Автоматизированное воспроизведение скриптовых тестов**

При выборе подходящего инструмента следует принимать во внимание его принцип работы. Наиболее распространены два варианта:

* Воспроизведение теста происходит по обращению к экрану, без анализа самого экрана и элементов интерфейса. Обычно такое воспроизведение осуществляется через координаты жестов на экране. Главный плюс – обычно нет необходимости модифицировать приложение. Главный минус – зависимость тестов от размера экрана, ориентации устройства, дизайна приложения.
* Воспроизведение теста посредством обращения к интерфейсным элементам приложения. В тесте указаны метки для форм, кнопок, текстбоксов и прочей визуальной «начинки». Главный плюс – даже существенные изменения в интерфейсе приложения вряд ли повлияют на работу теста. Главный минус – придётся просить разработчиков собирать тестировщикам версии приложений с дополнительными библиотеками.

[UIAutomation](http://developer.apple.com/library/ios/#documentation/DeveloperTools/Reference/UIAutomationRef/_index.html) – стандартное решение от Apple, которое позволяет выполнять написанные на JavaScript тестовые сценарии как в эмуляторе, так и на устройстве. Входит в состав Instruments. Компилировать дополнительные библиотеки не требуется. Начиная с Xcode 4.3 появилась возможность записи тестов через рекодер.

[Robotium](https://code.google.com/p/robotium/) – пожалуй, самый известный на текущий момент инструмент для автоматизации тестирования Android-приложений. “It’s like Selenium, but for Android” – утверждают разработчики. Тесты пишутся на Java (есть сторонние решения, позволяющие писать их, например, на [Python](https://github.com/Top-Q/pyrobotium)). Возможности запускать тесты на устройстве нет, только в эмуляторе. Необходимо добавлять библиотеку в сборку приложения.

[MonkeyRunner](http://developer.android.com/tools/help/monkeyrunner_concepts.html) поставляется в составе Android SDK, позволяет выполнять функциональное тестирование приложения под Android, предоставляя API для управления устройством. MonkeyRunner является более низкоуровневым по сравнению с Robotium, и не требует исходного кода приложения. Тесты пишутся на Python, или с помощью рекордера, выполняются как в эмуляторе, так и на реальных устройствах, подключенных к компьютеру. Большой минус этого решения в том, что жесты записываются в координатах, проверка результатов только путём сравнения скриншотов, что очень усложняет использование одного скрипта для тестирования на нескольких устройствах, а так же делает скрипты неподходящими для регрессионного тестирования в случае изменения GUI приложения.

[TestStudio](http://www.telerik.com/automated-testing-tools/ios-testing.aspx) – бесплатное приложение для автоматического тестирования на платформе iOS. Базируется на обращении к компонентам приложения, а не на скриншотах. Имеется как рекодер, так и возможность писать и редактировать тесты вручную. Позволяет тестировать web и native компоненты. Интересная особенность: можно записывать и воспроизводить тесты на устройстве без подключения к компьютеру.

[AppThwack](https://appthwack.com/) – интересный сервис для тестирования на Android-устройствах (поддержка iOS обещается в скором времени). Вы загружаете своё приложение на ресурс, оно устанавливается на настоящие устройства (ассортимент переваливает за сотню) и подвергается «исследованию» – запуск, замеры используемой памяти и загрузки процессоров, выявление ошибок и проблем, нагрузка небольшим манки-тестом. По результатам исследования создаётся отчёт со скриншотами. Имеется триал (неделя), цена – от 29$ в месяц за тестирование на 10 наиболее распространённых устройствах.

[JamoSolution](http://www.jamosolutions.com/) – одна из самых многообещающих платформ, на которой сейчас разрабатывается несколько инструментов (например, M-eux test и SeeTest). Она позволяет тестировать iOS, Android, Windows Phone и другие платформы. Поддерживается запись тестов (record&play). Работает через установку на устройстве приложения-агента, что освобождает разработчика от модифицирования своего приложения. Есть триальная версия.

[EggPlant](http://www.testplant.com/products/eggplant/mobile/) от студии TestPlant позволяет запускать свой тестовый скрипт на множестве устройств одновременно, определяя выходные данные методом распознавания картинки на экране. Поддерживает тестирование на устройствах Android и iOS и их эмуляторах, а так же на эмуляторе Windows Phone. Приложение разработано под Windows, Linux, Mac. Есть триальная версия.

[Squish](http://www.froglogic.com/squish/gui-testing/) – платное (2400$) средство для автоматического тестирования Qt, Web, Java, iOS и других приложений. Поддерживает запись тестов, воспринимает скрипты на Javascript, Python, Perl или Tcl. Есть 30-дневная триал-версия.

[Sikuli](http://www.sikuli.org/) – open sourсe инструмент для автоматизации тестирования GUI Java-приложений (в том числе и Android). Открытая кросс-платформенная визуальная среда создания сценариев-скриптов, которая ориентирована на программирование графического интерфейса при помощи изображений (скриншотов). Особенность – скрипт, задающий последовательность действий, позволяет использовать скриншоты – чтобы дать команду нажать кнопку, достаточно подставить в скрипт скриншот этой кнопки. Поддерживает написание скриптов на Java и Python.

[MonkeyTalk](https://www.gorillalogic.com/monkeytalk) – бесплатный инструмент для тестирования Android и iOS-приложений. Имеет собственный мощный скриптовый язык (можно писать скрипты и на Javascript), позволяет создавать и хранить тестовые проекты (тест-кейсы, тест-сьюты). Так же имеется интеграция с Eclipse, есть рекодер. Требует вставки своей библиотеки в приложение.

[Robot Framework](https://code.google.com/p/robotframework/) – это open-source фреймворк для автоматизации приемочного тестирования и разработки через приемочные тесты (ATDD), имеющий широкий функционал. Поддерживает дополнительные библиотеки (можно использовать собственные, написанные на Python или Java) – именно с помощью уже реализованных библиотек и появляется возможность тестирования приложений на Android и iOS.

**Нагрузочное тестирование**

[HP Virtual User Generator](http://h30499.www3.hp.com/t5/LoadRunner-Support-Forum/HP-Virtual-User-Generator-for-SAP-Business-Objects-Web/td-p/5584401). Служит для нагрузки сервера мультиплицированием входящего траффика. Тестировщик создаёт скрипт, запускает его на устройстве, HP VuGen перехватывает траффик и имитирует запросы к серверу с подобной информацией, но от нескольких тысяч или миллионов устройств одновременно.

[Neoload](http://www.neotys.com/product/mobile-load-testing.html) позволяет эмулировать правдоподобные условия сети (эмуляция 3G, 3G+, H+, 4G LTE). Может настраиваться для эмуляции нагрузки с разношёрстного списка устройств (есть предустановки для устройств iPhone5, Samsung Galaxy Tab II, Nokia Lumia 800, Blackberry Bold 9900 и других), основной скрипт нагрузки можно записать с помощью любого реального устройства по Wi-Fi. Есть бесплатный месячный триал, стоимость самой дешёвой лицензии (на пять пользователей) – 1200 евро. Поддерживает устройства на базе iOS, Android, Windows Phone и другие.

**Манкитестинг**

[Monkey](http://developer.android.com/tools/help/monkey.html) является инструментом стресс-тестирования для Android, содержащимся в Android SDK. Генерирует псевдослучайные действия пользователя. Позволяет настроить «хаотичность», интервал между событиями, их тип и т. п. Модификация кода приложения не требуется. Тестировать можно как на эмуляторе, так и на подключенном устройстве.

[Anteater](https://www.redant.com/anteater) – инструмент для манкитестинга iOS-приложений. Имеет более широкий функционал, чем Monkey для Android. По-видимому, существует только под Mac. Бесплатен.

**Сервисы для бета-тестирования**

[uTest](http://www.utest.com/) – сообщество из 45 тысяч профессиональных тестеров из 180 стран. Реальные пользователи протестируют работу приложения. Платный. (iOS, Android, Windows Phone)

[The Beta Family](http://thebetafamily.com/) – бесплатный сервис для тестирования приложения. Заводите аккаунт, заливаете бета-версию приложения, рассылаете приглашение на тестирование, обрабатываете результаты тестирования. Можно выбрать тип бета-тестеров: private или public. Если public, то приложение смогут тестировать все желающие. Работает с iOS, Android, Windows Phone.

[Zubhium](https://zubhium.com/). Предоставляет SDK для Android, с помощью которого вы в свое приложение встраиваете код для автоматического сбора информации об ошибках. Разработчик выкладывает бета-версию, приглашает тестеров, получает информацию. Стоимость подписки от 10$/мес. Есть 30-дневный триал.

**Сборщики статистики**

Полезно знать, как пользователи работают с вашим приложением: какие функции наиболее востребованы, какие кнопки нажимаются, какие настройки меняются, какие ошибки совершаются, сколько времени пользователь проводит на различных экранах приложения. Неплохо так же иметь статистику по пользователям – какая версия ОС у их девайсов, где они географически расположены и т.д. Самый простой путь сбора такой статистики – воспользоваться готовой системой сбора аналитической информации. Например:

* [Flurry](http://www.flurry.com/) (бесплатная) (iOS, Android, Windows Phone)
* [BugSence](http://www.bugsense.com/) (бесплатная) (iOS, Android, Windows Phone)
* [Apsalar](http://apsalar.com/) (бесплатная) (iOS, Android)
* [Google Analytics](http://www.google.com/analytics/) (бесплатная) (iOS, Android)
* [Mixpanel](https://mixpanel.com/) (платная) (iPhone, Android)
* [Localytics](http://www.localytics.com/) (платная) (iOS, Android, Windows Phone)
* [Bango](http://bango.com/) (платная) (Android, Windows Phone)

У каждой системы есть свои изюминки: обновление статистики в реальном времени (Localytics), суперточность с отслеживанием уникальных ID каждого пользователя (Bango), наличие средств для проведения опроса среди пользователей (Apsalar) и т. д. Естественно, есть и море отличий: в интерфейсе, в средствах анализа, в наличии дополнительных API, в стоимости, в наборе поддерживаемых платформ.

**Другие полезные инструменты**

[Fake GPS](https://play.google.com/store/apps/details?id=com.lexa.fakegps&hl=en)– приложение для Android-устройств, позволяющее установить произвольные данные в модуле геолокации.  
Так же, в ранее упоминавшемся Android SDK есть неплохой спектр мелочей, облегчающих тестирование приложений под Android – консольные возможности установки, удаления и запуска приложений, просмотр в реальном времени и вывод логов работы устройства в файл, перезагрузка приложения и так далее. Описание всех этих возможностей легко находятся в Интернете. Много полезного можно найти в книге от разработчика SDK Diego Torres Milano: “Android Application Testing Guide”.

**Комплексные решения**

[TestQuest Pro](http://www.bsquare.com/products/testquest-automated-testing-platform) – инструмент для полного автоматического тестирования, разработанный для компаний и предприятий. Поддерживает функциональное, нагрузочное, регрессивное тестирование, тестирование качества и взаимодействия.

[Experitest.com](http://experitest.com/) – позволяет проводить автоматическое, ручное (удалённо) и облачное тестирования на большом спектре устройств (2500$ в год за автоматическое тестирование, 4000$ в год за ручное).

Стоит отметить существование компаний, специализирующихся на тестировании, в том числе и на тестировании приложений на мобильных устройствах. Например, [Qulix QA](http://www.qa.qulix.com/) производят всестороннее тестовое покрытие – верификация работы приложения относительно ОС, платформ, языков и др; прохождение сертификации для подписи продуктов и попадания в маркет; тестирование приложений на реальных мобильных устройствах.