Automated Testing – процесс проверки программного обеспечения на соответствие требованиям, который выполняется специальными программыми средствами; тесты выполняются без мануального, ручного вмешательства.

Artifact – любой результат разработки автотестинга, напр. скрипт, тест кейс, экселевский документ.

Automated Test Case – реализация TC в скриптовом файле (или любом другом) в форме, которая может быть выполнена инструменами AT (SilkTest).

Manual Test Case (MTC) – реализация тест кейса в мануальном виде (SoftTest)

Silktest - инструмент для автотестирвоания функциональности приложений.

Автоматическое тестирование в первую очередь предназначено для программ, для которых работоспособность и безопасность при любых входных данных являются наиважнейшими приоритетами: веб-сервер, клиент/сервер SSH, sandboxing, сетевые протоколы. (<http://habrahabr.ru/blogs/code_review/128503/>)

**Автотестинг и тест-дизайн**

Успех автоматизации тестирования во многом зависит от того, как тесты будут разработаны. Это напрямую влияет на то, насколько быстро тесты будут разрабатываться, [насколько легко](http://automated-testing.info/knowledgebase/article/podvodnye-kamni-avtomatizacii-testirovanija) в них можно будет сориентироваться, насколько хорошо они смогут [локализовать проблему](http://automated-testing.info/knowledgebase/article/rasprostranennye-oshibki-avtomatizacii-testirovanija), насколько хорошо они смогут выявлять проблему вообще. На эти и многие другие параметры влияет то, как [будут структурированы тесты](http://automated-testing.info/knowledgebase/article/anatomija-frejmvorka-avtomaticheskogo-testirovanija). То есть немаловажную роль в этом играет тест-дизайн. Ведь именно тестовый сценарий или тестовое предписание служит основой для [реализации автоматического теста](http://automated-testing.info/knowledgebase/article/avtotesty-kriterii-zavershennosti). Поэтому, если повлиять на процесс на стадии тест-дизайна, то можно заметно улучшить [процесс автоматизации тестирования](http://automated-testing.info/knowledgebase/article/avtomatizacija-testirovanija-s-ispolzovaniem-agile-podhoda-chast-1). Итак, что нужно тестам, чтобы их потом легче было легче автоматизировать:

* **Все тесты (или большинство) начинаются с некоторого фиксированного состояния** - это соответствует пользовательскому сценарию по работе с некоторой системой. Зачастую, есть общий набор действий, от которого уже потом отталкивается тест. С точки зрения автоматической реализации, будет написан некоторый метод или другой вид модуля, который приводит систему в исходное состояние и потом возвращает в него (если надо). Во многих решениях автоматизации подобная реализация предусмотрена специальными блоками (setUp, tearDown в JUnit-e, appstate в СилкТесте и т.п.).
* **Тесты не зависят друг от друга** - это очень болезненный момент. Нехорошо, когда сбой в одном из тестов повлечет за собой ошибки в других тестах. В этом случае непонятна сегментация тестов. Фактически, получаем один длинный workflow-тест, который разбит просто на подпункты и сбой в одном месте влечет за собой "эффект домино". Чтобы избежать подобного эффекта, лучше сразу предусмотреть возможность выполнения теста как в пакете, так и отдельно. Подобное выравнивание достигается фиксацией начального состояния (см. предыдущий пункт), а также определением дополнительных условий, необходимых именно для данного теста. Зачастую, эта проблема соприкасается с проблемой организации внешних данных для тестов.
* **Одна функция - один тест** - идея заключается в том, чтобы тесты делать не огромными, которые бы охватывали множество разных аспектов или какой-то длинный workflow, а более-менее сосредотачивались на некоторой одной функции. То есть, тест строится на том, что мы перешли в некоторое состояние, из которого доступна тестируемая функция, сделали ввод и проверили результаты заботы функции, а затем вышли. Не больше, не меньше. Если уже затрагивается более одного бизнес-функционала в одном тесте, при этом эти функции не являются взаимо-зависимыми, то в результате наши автоматизированные тесты рискуют пропустить ошибку в одном функционале, если будет выявлена критичная ошибка в другом функционале, который проверялся раньше. С точки зрения дизайна разница в предложенных подходах незначительная, но с точки зрения реализации появляются проблемы с выравниванием состояния приложения, чтобы можно было продолжить тест. Безусловно, глупо плодить тысячи мелких и крайне похожих тестов, но и слишком длинные сценарии вредны. Нужно искать некоторый баланс, удобный и для дизайна и для реализации
* **Имеется фиксированный набор выполняемых команд и верификаций** - имеется ввиду, что было бы неплохо сформировать достаточно узкий и в то же время покрывающий по максимуму набор выполняемых команд с поправкой на параметры. Например, если мы тестируем Content Management System, то у нас для разного типа содержимого есть фиксированный набор операций: перейти - создать и заполнить - сохранить и проверить - переоткрыть - модифицировать - сохранить и проверить. Вот по такому шаблону можно пройтись по всем типам содержимого, специфику составляют только используемые ресурсы. В этом случае, тест-дизайнер делает много дублирований, которые потом устраняются во время автоматизации, а ведь настоящий джедай знает, что Copy/Paste sucks. Более того, есть различные подходы в автоматизации тестирования, которые эффективны именно при определенном дизайне. Так вот тот же keyword-driven подход как раз наиболее эффективен, когда можно выделить фиксированный набор действий. Поэтому, формирование единого каркаса для группы тестов, резко уменьшает работу тест-дизайнеру, а затем и автоматизатору

Процесс тестирования не является ни изолированным, ни разрозненным, поэтому есть возможность подумать об упрощении своей работы еще до того, когда она к вам поступила. А грамотный тест-дизайн позволит минимизировать ваши проблемы в реализации в дальнейшем

**Что может предложить автоматизация тестирования? Или выгоды автоматизации тестирования.**

Автоматизация тестирования может дать возможность выполнять определенные тестовые задачи намного быстрее и эффективнее чем это может быть сделано вручную. Это и так понятно, но, все-таки, существуют еще некоторые плюсы, которые люди не всегда осознают:

1. **Выполнение существующих регрессионных тестов.** Это должно быть самая очевидная задача автоматизации, в частности, когда программы исправляются и дополняются очень часто. Усилия, потраченные на выполнение существующих регрессионных тестов должны быть минимальны, для того, чтобы в нужный момент была возможность выбора  автоматизированных тестов и их запуск с минимальными ручными усилиями и в считанные минуты.
2. **Выполнение большого количества тестов более часто.** Известный факт, что часто-повторяемые задачи наиболее хорошо поддаются автоматизации. Четкая выгода в том, чтобы запускать большее количество тестов, в меньшие сроки, давая при этом косвенную возможность запускать тесты более часто.
3. **Выполнение тестов, которые будет сложно или невозможно сделать.**Предположим, что нам необходимо симулировать нагрузку в 500 пользователей на какую-то систему. Я думаю, что это будет невозможно или крайне затратно сделать ручным способом. С другой стороны, автоматические тесты могут выполнить все это в любой момент, и запустить их сможет любой человек, который даже не будет особо разбираться в логике системы. Другой пример невыполнимой работы может быть следующим. Каждый пользователь, смотря на графический интерфейс, интуитивно ожидает появление какой-то визуализации данных при определенный действиях, но пользователь (тестеровщик) может даже и не представлять, что за этим кроются какие-то скрытые механизмы, такие как, например, вызов и обработка событий, которые нельзя проверить визуально. События и их поведение просто невозможно проверить без использования специальных программных средств.
4. **Использовать сэкономленные ресурсы на более важные и интересные задачи.** Автоматизация обычных и скучных задач, например, повторяемых с разными вариациями данных, дает возможность полагаться на большую степень аккуратности при проведения тестирования и при этом улучшается моральное состояние людей, которые работаю с такими задачами. Также высококвалифицированные работники могут свободно вкладывать свои усилия в проектирование более сложных автоматических тестов, которые впоследствии будут выполняться. Ручная работа некуда не денется. Ручное тестирование является неотъемлемой частью автоматизации тестирования и наоборот. Всегда будут задачи, которые лучше выполнять вручную, более того ручное тестирование будет выполняться намного лучше, если автоматические тесты будут давать уверенность в том, что в протестированном участке нет дефектов.
5. **Постоянство и повторяемость тестов.** Автоматизация дает нам возможность постоянно выполнять тесты с четко одними и теми же данными (входные данные всегда будут те же, а вот выходные могут отличаться на протяжении некоторого времени разработки), чего нам (увы) не может гарантировать человек, выполняя эту же работу вручную, потому уверенность в тестировании повышается. Эти же тесты могут быть выполнены на разных операционных системах и использованием разных баз данных, что тоже даст уверенность в том, что кросс-платформенность продукта на высоком уровне.
6. **Повторное использование тестов.** Усилия, вложенные на принятие решений, что необходимо тестировать, проектирования и формирования тестов может дать возможность использовать или повторно использовать разработанные тесты. В свою очередь это сэкономит время разработки новых тестов. Что необходимо учитывать, что бы такие тесты были стабильными, иначе потраченное время на поддержку тестов будет большим.
7. **Быстрый вывод продукта на рынок.**Как только тесты будут автоматизированы, у вас появиться возможность повторного запуска и следственно общее время, потраченное на выполнение тестирования, будет меньшим.
8. **Повышающаяся уверенность.** Конечно же, осознание и понимание, что исчерпывающее количество автоматических тестов дают постоянно положительные результаты увеличивает доверие и уверенность в том, что не будет никаких непредсказуемых сюрпризов, когда программное обеспечение будет выпущено (конечно, подразумевается, что тесты выполняют свою функции и проверяют ПО на наличие ошибок).

Если  сложить все вышесказанное, то больший объем тестирования, может быть достигнут с меньшими усилиями, давая при этом возможность улучшать как качество, так и продуктивность.

**3 сценария когда автоматизация может быть применима (или не может)**

**Повторяющиеся тесты**

Система, на которой должны выполняться периодические тесты - является хорошим кандидатом для автоматизации. Возьмем в качестве примера, проект поддержки, в котором должны выполняться регрессионные тесты для каждого запроса (PR/CR). Ручное тестирование, как правило, будет громоздким и обязательно возможно приведет к ряду ошибок. В таком случае, автоматизация тестирования может оказаться весьма эффективным средством минимизации затрат на выполнение тестирования.  
  
Стоимость автоматизации и первоначальное инвестирование затрат на создание скриптов обычно больше чем при ручном тестировании. К тому же автоматизация должна быть проанализирована на счет того, когда она окупиться. Вот несколько указателей, которые помогут сделать выбор:

* Сколько повторений тестов ожидается и как долго они будут использоваться?
* [Насколько дорогой инструмент автоматизации](http://automated-testing.info/knowledgebase/article/ocenka-i-vybor-instrumenta-avtomatizacii)?
* Сколько времени/ресурсов потребуется на создание тестов? Автоматизация - не всегда запись/повторение, и для того, что бы построить хорошую автоматизацию необходимо прорабатывать дизайн и написать фреймворк.
* Ожидаются ли постоянные изменения в скриптах?
* Требуют ли тестировщики прохождения тренингов?

«Повторяющие тесты -> автоматизация» это хорошее правило, но не единственное. Вы можете заметить, что автоматизация может быть нецелесообразна, если она с одной стороны дорого стоит, и с другой стороны при этом есть легкие в прохождении ручные тесты.

**Невыполнимые ручные тесты**

Перед тем, как выпускать приложение в использование, возможно, нужно определить насколько приложение поведет себя при большой нагрузке или, к примеру когда оно будет обрабатывать миллион записей в базе данных. Такие тесты обычно хорошо поддаются автоматизации. И даже одиночно автоматизированый тест будет приносить пользу в условиях риска, где стоимость пропущенной ошибки может быть губительной.

**Низкая критичность и вероятность человеческого фактора**

Автоматизация может выполнять некоторые задачи намного лучше, чем можно положиться на человека. Например, если дать инструменту автоматизации и человеку сравнить и проверить два файла Excel с 10,000 строк данных. Я думаю, что ответ очевидный, кто может сделать это быстрее и без ошибок. Но конечно же, люди гораздо лучше для другого рода тестирования. Человек может заметить ошибки в документации. Или проверить GUI, который может смотреться слишком загроможденным!   
  
Автоматизация ограничивается тестированием, которое запрограммировано. Не все, что человеческий ум может наблюдать / анализировать, может быть запрограммировано, и потому, автоматизация будет пропускать некоторые ошибки. Главный вопрос: насколько такие ошибки важны для вас? Нестабильная система может быть не готова к автоматизации, так как серьезные "ручные" ошибки могут быть проигнорированы. Автоматизация лучше подходит для систем, в которых ожидается меньше "человеческих" ошибок и они не так критичны.

**Самое главное, что надо понять и запомнить, автоматизация и ручная работа не соперничают, а прекрасно дополняют друг друга. Необходимо просто выработать правильный баланс между автоматизацией и ручным тестированием.**

Долгое время считалось, что динамический анализ программ является слишком тяжеловесным подходом к обнаружению программных дефектов, и полученные результаты не оправдывают затраченных усилий и ресурсов. Однако, две важные тенденции развития современной индустрии производства программного обеспечения позволяют по-новому взглянуть на эту проблему. С одной стороны, при постоянном увеличении объема и сложности ПО любые автоматические средства обнаружения ошибок и контроля качества могут оказаться полезными и востребованными. С другой – непрерывный рост производительности современных вычислительных систем позволяет эффективно решать все более сложные вычислительные задачи. (<http://habrahabr.ru/blogs/code_review/128503/>)

Автотест (автоматизированный тест) – это скрипт, имитирующий взаимодействия пользователя с приложением, цель которого – локализация ошибок в работе программного обеспечения.  
  
Конечная цель автоматизации тестирования представляет собой некий набор автотестов, которые, при нажатии на кнопку «GO!», будут поочередно запускаться. Ну или всегда можно запустить какой-либо автотест отдельно, если для этого есть необходимость. Каждый такой скрипт проверяет правильность работы определенной части приложения и фиксирует ошибки в случае, если что-то работает не так.  
О том, насколько выгоднее использовать автоматизированное тестирование чем ручное, а так же о плюсах автоматизированного тестирования хорошо написано в [этой](http://habrahabr.ru/blogs/personal/67958) статье. Я же хочу описать возможные проблемы, с которыми может столкнуться тестировщик, решивший использовать автоматизированные тесты.