# Министерство образования Республики Беларусь

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«Могилевский государственный политехнический колледж»

# Домашняя контрольная работа №1

По дисциплине: «**Тестирование программного обеспечения**»

Группа ПО-455

## Выполнил О. И. Любаль

Шифр 14

2022

**15 Требования к идеальному критерию.**

Требования к идеальному критерию:

1. Критерий должен быть достаточным, т.е. показывать, когда некоторое конечное множество тестов достаточно для тестирования данной программы.

2. Критерий должен быть полным, т.е. в случае ошибки должен существовать тест из множества тестов, удовлетворяющих критерию, который раскрывает ошибку.

3. Критерий должен быть надежным, т.е. любые два множества тестов, удовлетворяющих ему, одновременно должны раскрывать или не раскрывать ошибки программы

4. Критерий должен быть легко проверяемым, например вычисляемым на тестах.

Для нетривиальных классов программ в общем случае не существует полного и надежного критерия, зависящего от программ или спецификаций.

Поэтому мы стремимся к идеальному общему критерию через реальные частные.

**Классы критериев**

1. Структурные критерии используют информацию о структуре программы (критерии так называемого "белого ящика")

2. Функциональные критерии формулируются в описании требований к программному изделию (критерии так называемого "черного ящика")

3. Критерии стохастического тестирования формулируются в терминах проверки наличия заданных свойств у тестируемого приложения, средствами проверки некоторой статистической гипотезы.

4. Мутационные критерии ориентированы на проверку свойств программного изделия на основе подхода Монте-Карло.

**Структурные критерии (класс I).**

Структурные критерии используют модель программы в виде "белого ящика", что предполагает знание исходного текста программы или спецификации программы в виде потокового графа управления. Структурная информация понятна и доступна разработчикам подсистем и модулей приложения, поэтому данный класс критериев часто используется на этапах модульного и интеграционного тестирования (Unit testing, Integration testing).

Структурные критерии базируются на основных элементах УГП, операторах, ветвях и путях.

• Условие критерия тестирования команд (критерий С0) - набор тестов в совокупности должен обеспечить прохождение каждой команды не менее одного раза. Это слабый критерий, он, как правило, используется в больших программных системах, где другие критерии применить невозможно.

• Условие критерия тестирования ветвей (критерий С1) - набор тестов в совокупности должен обеспечить прохождение каждой ветви не менее одного раза. Это достаточно сильный и при этом экономичный критерий, поскольку множество ветвей в тестируемом приложении конечно и не так уж велико. Данный критерий часто используется в системах автоматизации тестирования.

• Условие критерия тестирования путей (критерий С2) - набор тестов в совокупности должен обеспечить прохождение каждого пути не менее 1 раз. Если программа содержит цикл (в особенности с неявно заданным числом итераций), то число итераций ограничивается константой (часто - 2, или числом классов выходных путей).

**Функциональные критерии (класс II)**

Функциональный критерий – важнейший для программной инду¬стрии критерий тестирования. Он обеспечивает, прежде всего, контроль степени выполнения требований заказчика в программном продукте. По¬скольку требования формулируются к продукту в целом, они отражают взаимодействие тестируемого приложения с окружением. При функцио-нальном тестировании преимущественно используется модель «черного ящика». Проблема функционального тестирования – это, прежде всего, грудоемкость; дело в том, что документы, фиксирующие требования к программному изделию (Software requirement specification, Functional specification и т.п.), как правило, достаточно объемны, тем не менее, соот-ветствующая проверка должна быть всеобъемлющей.

Ниже приведены частные виды функциональных критериев.

• Тестирование пунктов спецификации – набор тестов в совокупно¬сти должен обеспечить проверку каждого тестируемого пункта не менее одного раза.

Спецификация требований может содержать сотни и тысячи пунк¬тов требований к программному продукту, и каждое из этих тре¬бований при тестировании должно быть проверено в соответст¬вии с критерием не менее чем одним тестом.

• Тестирование классов входных данных – набор тестов в совокупно¬сти должен обеспечить проверку представителя каждого класса входных данных не менее одного раза. При создании тестов классы входных данных сопоставляются с режимами использования тестируемого компонента или подсис¬темы приложения, что заметно сокращает варианты перебора, учитываемые при разработке тестовых наборов. Следует заме¬тить, что, перебирая в соответствии с критерием величины вход¬ных переменных (например, различные файлы – источники входных данных), мы вынуждены применять мощные тестовые наборы. Действительно, наряду с ограничениями на величины входных данных, существуют ограничения на величины входных данных во всевозможных комбинациях, в том числе проверка ре¬акций системы на появление ошибок в значениях или структурах входных данных. Учет этого многообразия – процесс трудоемкий, что создает сложности для применения критерия

• Тестирование правил – набор тестов в совокупности должен обеспечить проверку каждого правила, если входные и выходные зна¬чения описываются набором правил некоторой грамматики. Следует заметить, что грамматика должна быть достаточно про¬стой, чтобы трудоемкость разработки соответствующего набора тестов была реальной (вписывалась в сроки и штат специалистов, выделенных для реализации фазы тестирования).

• Тестирование классов выходных данных – набор тестов в совокупности должен обеспечить проверку представителя каждого выходного класса, при условии, что выходные результаты заранее расклассифицированы, причем отдельные классы результатов учитывают, в том числе, ограничения на ресурсы или на время (time out).

При создании тестов классы выходных данных сопоставляются с режимами использования тестируемого компонента или подсис¬темы, что заметно сокращает варианты перебора, учитываемые при разработке тестовых наборов.

• Тестирование функций – набор тестов в совокупности должен обеспечить проверку каждого действия, реализуемого тестируе¬мым модулем, не менее одного раза.

Очень популярный на практике критерий, который, однако, не обеспечивает покрытия части функциональности тестируемо¬го компонента, связанной со структурными и поведенческими свойствами, описание которых не сосредоточено в отдельных функциях (т.е. описание рассредоточено по компоненту). Критерий тестирования функций отчасти объединяет особенно-сти структурных и функциональных критериев. Он базируется на модели «полупрозрачного ящика», где явно указаны не только входы и выходы тестируемого компонента, но также состав и структура используемых методов (функций, процедур) и классов.

• Комбинированные критерии для программ и спецификаций – набор тестов в совокупности должен обеспечить проверку всех комби¬наций непротиворечивых условий программ и спецификаций не менее одного раза.

При этом все комбинации непротиворечивых условий надо подтвер¬дить, а условия противоречий следует обнаружить и ликвидировать.

**Стохастические критерии (класс III)**

Стохастическое тестирование применяется при тестировании слож­ных программных комплексов – когда набор детерминированных тестов (X,Y) имеет громадную мощность. В случаях, когда подобный набор невозможно разработать и исполнить на фазе тестирования, можно при­менить следующую методику:

Разработать программы – имитаторы случайных последователь­ностей входных сигналов {х}.

Вычислить независимым способом значения {у} для соответству­ющих входных сигналов {х} и получить тестовый набор (X,Y).

Протестировать приложение на тестовом наборе (X,Y), используя два способа контроля результатов:

Детерминированный контроль – проверка соответствия вычис­ленного значения увк{у} значению у, полученному в результате прогона теста на наборе {х} – случайной последовательности входных сигналов, сгенерированной имитатором.

Стохастический контроль – проверка соответствия множества значений {ув}, полученного в результате прогона тестов на на­боре входных значений {х}, заранее известному распределению результатов F(Y).

В этом случае множество Y неизвестно (его вычисление невоз­можно), но известен закон распределения данного множества.

Критерии стохастического тестирования:

Статистические методы окончания тестирования – стохастиче­ские методы принятия решений о совпадении гипотез о распре­делении случайных величин. К ним принадлежат широко извест­ные: метод Стьюдента (St), метод Хи-квадрат (х2) и т.п.

Метод оценки скорости выявления ошибок – основан на модели скорости выявления ошибок [12], согласно которой тестирование прекращается, если оцененный интервал времени между текущей ошибкой и следующей слишком велик для фазы тестирования

**Мутационный критерий (класс IV)**

Постулируется, что профессиональные программисты пишут сразу почти правильные программы, отличающиеся от правильных мелкими ошибками или описками типа – перестановка местами максимальных значений индексов в описании массивов, ошибки в знаках арифметиче­ских операций, занижение или завышение границы цикла на 1 и т.п. Предлагается подход, позволяющий на основе мелких ошибок оценить общее число ошибок, оставшихся в программе.

Подход базируется на следующих понятиях:

* Мутации – мелкие ошибки в программе.
* Мутанты – программы, отличающиеся друг от друга мутациями.

Метод мутационного тестирования – в разрабатываемую программу Р вносят мутации, т.е. искусственно создают программы-мутанты Р1,Р2... Затем программа Р и ее мутанты тестируются на одном и том же наборе тестов (X,Y).

Если на наборе (X,Y) подтверждается правильность программы Р и, кроме того, выявляются все внесенные в программы-мутанты ошибки, то набор тестов (X, Y) соответствует мутационному критерию, а тестируе­мая программа объявляется правильной.

Если некоторые мутанты не выявили всех мутаций, то надо расширять набор тестов (X,Y) и продолжать тестирование.

**115 Разработка тестов. Проведение тестирования. Анализ результатов.**

Порядок разработки тестов

По внешней спецификации разрабатываются тесты:

* • для каждого класса входных данных;
* • для граничных и особых значений входных данных. Контролируется, все ли классы выходных данных при этом проверяются, и добавляются при необходимости нужные тесты.

По тексту программы проверяется, все ли условные переходы выполнены в каждом направлении (С1). При необходимости добавляются новые тесты.

Аналогично проверяется, проходятся ли пути для каждого цикла: без выполнения тела, с однократным и максимальным числом повторений.

Готовятся тесты, проверяющие исключительные ситуации, недопустимые входные данные, аварийные ситуации.

Функциональное тестирование дополняется здесь структурным. Классы входных/выходных данных должны быть определены в плане тестирования уже во внешней спецификации. Согласно статистике проверка по С1 обычно выявляет 90 % всех ошибок, найденных при тестировании.

Систематическое тестирование предполагает также ведение журнала отладки (Bug Book), в котором фиксируется ошибка (описание, дата обнаружения, автор модуля) и в дальнейшем — исправление (дата, автор).

Приведем так называемые аксиомы тестирования.

* 1. Тест должен быть направлен на обнаружение ошибки, а не на подтверждение правильности программы.
* 2. Автор теста — не автор программы.
* 3. Тесты разрабатываются одновременно или до разработки программы.
* 4. Необходимо предсказывать ожидаемые результаты теста до его выполнения и анализировать причины расхождения результатов.
* 5. Предыдущее тестирование необходимо повторять после каждого внесения исправлений в программу.
* 6. Следует повторять полное тестирование после внесения изменений в программу или после переноса ее в другую среду.
* 7. В те программы, в которых обнаружено много ошибок, необходимо дополнить первоначальный набор тестов.

Тестирование является неотъемлемой частью жизненного цикла программного обеспечения. Само по себе тестирование – длительный процесс проверок на соответствие ожидаемого результата. Нельзя выделить какой-то один этап как важный, каждый из них имеет одинаковый вес. При создании продукта тестировщик не просто играет важную роль, а участвует на каждом этапе разработки от концепции до выхода продукта в свет.

Как уже было отмечено, тестирование – неотъемлемая часть жизненного цикла продукта, однако необходимо понимать, с чего стоит начинать тестирование, и когда оно заканчивается. Например, неопытные специалисты ошибочно думают, что после релиза они могут переключаться на другой проект.

Всего принято выделять 7 этапов тестирования:

Работа с требованиями. Знакомство с требованиями заказчика, что должен из себя представлять итоговый продукт, обсуждение.

Разработка стратегии тестирования. Оценка сроков тестирования, выявление среды тестирования, объединение всей информации, полученной при работе с требованиями.

Создание тестовой документации. Написание сценариев, которые позволят проверить функционал.

Тестирование прототипа. Тестирование основного функционала продукта, корректировка целей, добавление фичей.

Основное тестирование. Выполнение общей проверки продукта.

Стабилизация. На данном этапе происходит работа над устранением багов.

Эксплуатация. Проводится регресс-тестирование, устранение ошибок, которые нашел конечный пользователь.

Прежде, чем переходить к описанию каждого цикла тестирования, необходимо запомнить важное правило: переход к следующему этапу допустим лишь после полного завершения работы на предыдущем. Теперь поговорим о каждом этапе тестирования отдельно.

**Этап 1**. Работа с требованиями

Команда тестирования знакомится с требованиями заказчика, функционалом продукта. На данном этапе важно непрерывное общение команд, необходимо задавать вопросы разработчикам.

Тщательное изучение требований должно:

* выявить противоречия в требованиях;
* помочь определить потенциальные дефекты в функционале.

Общение является ключевым аспектом в разработке продукта, а продуманный roadmap (дорожная карта, стратегический план по реализации функционала продукта по определенным датам) позволит устранить нелогичность будущих внесенных изменений в продукт.

**Этап 2**. Разработка стратегии тестирования и планирование процедур контроля качества

Данный этап важен для лидов или менеджеров, поскольку от понимания полученной на предыдущем этапе информации зависит качество тестирования.

Тест — лид должен:

* резюмировать полученную информацию,
* оценить сроки тестирования,
* разработать стратегию тестирования: определить виды тестирования, которые можно применить к проекту, проанализировать имеющиеся среды и ресурсы, что имеется для проведения тестирования, описать приоритеты для непредвиденных ситуаций, как и где будет вестись тестовая документация;
* определение среды тестирования: какое оборудование необходимо для тестирования,
* составить план, который содержит описание, с чего начинается и чем заканчивается тестирование, и что будет тестироваться.

**Этап 3**. Создание тестовой документации

Цель данного этапа – создать документацию, объем которой будет охватывать детализацию, ход работ, а также вносить ясность для заказчика.

Общение с другими командами, понимание желаний заказчика напрямую влияют на качество тестовой документации. После проведенного тестирования можно проанализировать его успешность.

Тестовая документация может состоять из:

* тестовых сценариев: что и как будет проверяться при регресс-, дымовом и приемочном тестированиях;
* отчетности: результаты тестирования, списка багов и их серьезность;
* методологий тестирования.

Детализация тестовой документации зависит от проекта, поэтому она может отличаться и по охвату, и по формату, и по объему. Для тестировщика важно поддерживать документацию в актуальном виде, вносить любые изменения, связанные с изменением итогового продукта.

**Этап 4**. Тестирование прототипа

При создании и тестировании прототипа продукта необходимо выявить основные отклонения от ожидаемого результата и соответствие с бизнес-стратегией. Здесь же выявляются ошибки в работе логики основного функционала, устраняются найденные уязвимости и дефекты, допущенные на этапе разработки. Заказчик может сам участвовать в процессе тестирования прототипа, чтобы он мог оценить, на каком этапе находится разработка продукта. После тестирования выдвигаются пожелания со стороны заказчика. Новые пожелания необходимо задокументировать, оценить сроки, внедрить в проект и передать на осмотр заказчику.

Самый подходящий метод тестирования прототипа – проведение закрытого бета-тестирования, когда продукт тестирует продукт малое количество людей, которые в итоге будут использовать его после релиза. Это помогает учесть пожелания конечных пользователей.

Очень важно лиду или менеджеру проекта передавать информацию тестировщикам и разработчикам о пожеланиях заказчика, на какие частые сбои в продукте натыкались пользователи для того, чтобы сделать его более понятным.

**Этап 5**. Основное тестирование

Тестирование программного обеспечения является самым длительным и объемным процессом. Здесь формируются репорты о найденных дефектах, выполняется набор тестовых сценариев, создается тестовая среда, выполняется тестирование, виды которого были задокументированы на этапе создания тестовой документации. Смоук- и регресс-тестирования являются одними из основных видов тестирования, которые проводятся на данном этапе.

Важно понимать: невозможно найти все ошибки в продукте. Но и не найти ошибки при тестировании можно считать провалом. Главная цель — не сделать идеальный продукт без ошибок, а найти максимальное количество дефектов, которые потенциально могут сломать систему.

**Этап 6**. Стабилизация

Наверное, самый непонятный с точки зрения формулировки этап. На нем заканчивается работа с пожеланиями заказчика и фиксируются найденные баги. Он является связующим звеном между командой тестирования и командой разработчиков. На протяжении создания онлайн-ресурса команда разработчиков занималась своими делами, реализуя «хотелки» заказчика, а тестировщики репортили о новых дефектах. А на этапе стабилизации разработчики начинают слушать тестировщиков, устраняя то, что уже работает, но некорректно.

Если продукт существует в какой-то большой системе, то на данном этапе также проверяется коммуникация системы и продукта, то есть проводится интеграционное тестирование.

При анализе результатов тестирования выполняется классификация найденных ошибок, общая оценка тестируемого объекта. По результатам анализа может быть принято решение о доработке объекта тестирования и запуска следующей итерации или завершении этапа тестирования.

**95 Тестирование безопасности.**

SECURITY TESTING — это тип тестирования программного обеспечения, который выявляет уязвимости, угрозы, риски в программном приложении и предотвращает злоумышленные атаки от злоумышленников. Целью тестов безопасности является выявление всех возможных лазеек и слабых мест в программной системе, которые могут привести к потере информации, доходов, репутации со стороны сотрудников или посторонних лиц Организации.

Целью тестирования безопасности является выявление угроз в системе и оценка ее потенциальных уязвимостей, чтобы система не перестала функционировать или использовалась. Это также помогает в обнаружении всех возможных угроз безопасности в системе и помогает разработчикам в устранении этих проблем посредством кодирования.

Типы тестирования безопасности:

Существует семь основных типов тестирования безопасности согласно руководству по методологии Open Source Security Testing. Они объясняются следующим образом:

**Сканирование уязвимостей**. Это выполняется с помощью автоматизированного программного обеспечения для сканирования системы на наличие известных сигнатур уязвимостей.

**Сканирование безопасности**: оно включает в себя выявление слабых сторон сети и системы, а затем предоставляет решения для снижения этих рисков. Это сканирование может быть выполнено как для ручного, так и для автоматического сканирования.

**Тестирование на проникновение**. Этот тип тестирования имитирует атаку злоумышленника. Это тестирование включает анализ конкретной системы для проверки потенциальных уязвимостей при попытке внешнего взлома.

**Оценка рисков**: это тестирование включает анализ рисков безопасности, наблюдаемых в организации. Риски классифицируются как Низкие, Средние и Высокие. Это тестирование рекомендует контроль и меры по снижению риска.

**Аудит безопасности**: это внутренняя проверка приложений и операционных систем на наличие уязвимостей. Аудит также может быть выполнен путем построчной проверки кода

**Этический взлом**: это взлом системы программного обеспечения организации. В отличие от злонамеренных хакеров, которые воруют ради собственной выгоды, цель состоит в том, чтобы выявить недостатки в системе.

**Оценка состояния**: это объединяет сканирование безопасности, этический взлом и оценки рисков, чтобы показать общее состояние безопасности организации.

Всегда соглашаются, что стоимость будет больше, если мы отложим тестирование безопасности после фазы внедрения программного обеспечения или после развертывания. Таким образом, необходимо задействовать тестирование безопасности в жизненном цикле SDLC на более ранних этапах.

Давайте посмотрим на соответствующие процессы безопасности, которые будут приняты для каждой фазы в SDLC.

Фазы SDLC представлены в таблице 1.

Таблица 1.

|  |  |
| --- | --- |
| **Фазы SDLC** | **Процессы безопасности** |
| **Требования** | Анализ безопасности для требований и проверка случаев злоупотребления / неправильного использования |
| **дизайн** | Анализ рисков безопасности для проектирования. Разработка плана тестирования, включая тесты безопасности |
| **Кодирование и модульное тестирование** | Статическое и динамическое тестирование и тестирование белого ящика безопасности |
| **Интеграционное тестирование** | Тестирование черного ящика |
| **Тестирование системы** | Тестирование черного ящика и сканирование уязвимостей |
| **Реализация** | Тестирование на проникновение, сканирование уязвимостей |
| **Служба поддержки** | Анализ воздействия патчей |

План испытаний должен включать

* Тестовые сценарии или сценарии, связанные с безопасностью
* Тестовые данные, связанные с тестированием безопасности
* Инструменты тестирования, необходимые для тестирования безопасности
* Анализ результатов различных тестов из разных инструментов безопасности

Тестирование безопасности является наиболее важным тестированием для приложения и проверяет, остаются ли конфиденциальные данные конфиденциальными. В этом типе тестирования тестер играет роль злоумышленника и пытается обойти систему, чтобы найти ошибки, связанные с безопасностью. Тестирование безопасности очень важно в программной инженерии для защиты данных всеми средствами.

**74 Системное тестирование, приемо-сдаточные и сертификационные**

**испытания при разработке сертифицируемого программного обеспечения.**

При разработке массового («коробочного», COTS) программного обеспечения после проведения системного тестирования система проходит этапы альфа- и бета- тестирования, во время которого работу системы проверяют потенциальные пользователи (либо специально выделенные фокус-группы пользователей, либо все желающие). На этом этапе в программную систему вносятся последние незначительные изменения, не влияющие на суть системы. После завершения этой стадии система поступает в продажу конечным пользователям.

При разработке заказного программного обеспечения фазу альфа- и бета- тестирования заменяют приемо-сдаточные испытания. Во время этих испытаний заказчик удостоверяется, что система работает в соответствии с его потребностями (как зафиксированными в техническом задании на систему, так и не зафиксированными).

Заказчик может проводить такие испытания самостоятельно, выполняя заранее подготовленные тесты системы, либо проводить их совместно с представителями коллектива разработчиков. В этом случае тестовые примеры также готовятся разработчиками, например, на основе тестовых примеров, использовавшихся на этапе системного тестирования.

Завершаются приемо-сдаточные испытания либо подписанием акта приемки, либо выдачей заказчиком дополнительных требований к системе, которые должны быть исправлены до приемки системы. После устранения всех недостатков системы приемо-сдаточные испытания повторяются (возможно, по сокращенной программе). После успешного подписания акта система поступает в эксплуатацию заказчику.

Существует специальный вид программных систем, к свойствам которых предъявляются особые требованиями. Примером таких систем могут служить бортовые авиационные программные системы, для которых особое внимание уделяется вопросам безопасности, надежности и отказоустойчивости. Несмотря на то, что большая часть таких систем может быть отнесена к категории заказного программного обеспечения, для получения разрешения на установку системы на борт требуется получение сертификата на летную пригодность.

Таким образом, после проведения системного тестирования и приемо-сдаточных испытаний проводятся сертификационные испытания. Сертификация программного обеспечения – процесс установления и официального признания того, что разработка ПО проводилась в соответствии с определенными требованиями. В процессе сертификации происходит взаимодействие заявителя, сертифицирующего органа и наблюдательного органа.

Заявитель – это организация, подающая заявку в соответствующий сертифицирующий орган на получения сертификата (соответствия, качества, годности и т.п.) изделия.

Сертифицирующий орган – организация, рассматривающая заявку заявителя о проведении сертификации ПО и либо самостоятельно, либо путем формирования специальной комиссии производящая набор процедур направленных на проведение процесса сертификации ПО заявителя. Наблюдательный орган – комиссия специалистов, наблюдающих за процессами разработки заявителем сертифицируемой информационной системы и дающих заключение, о соответствии данного процесса определенным требованиям, которое передается на рассмотрение в сертифицирующий орган. [34]

Основной объект проверки в ходе сертификационных испытаний – соответствие процесса разработки программной системы регламенту и рекомендациям стандарта, на соответствие которому проводится сертификация. Такое соответствие определяется при помощи анализа жизненного цикла сертифицируемой системы и документов, создаваемых на ключевых его этапах. Весь процесс анализа и те свойства системы, которые подвергаются сертификации, описывается в плане сертификационных испытаний, который утверждается совместно заявителем и сертифицирующим органом.

В случае сертификации бортовой системы по стандарту DO-178B (или его аналогам КТ-178, JB-12 и т.п.) план дополнительно определяет уровень влияния отказа программной системы на безопасность полета (уровень отказобезопасности) по которому будет проводиться сертификация. Любые вопросы, которые возникают у сертифицирующего органа относительно содержания плана сертификационных испытаний, должны быть разрешены до начала самих испытаний.

Согласно требованиями DO-178B план сертификационных испытаний (план программных аспектов сертификации) должен включать:

• обзор системы: Этот раздел описывает систему, включая описание ее функций и их размещение в программном и аппаратном обеспечении, ее архитектуру, используемый процессор (процессоры), аппаратно-программный интерфейс, и особенности отказобезопасности;

• обзор программного обеспечения: Этот раздел коротко описывает функции программного обеспечения с акцентом на концепцию обеспечения отказобезопасности и разделения на обособленные части, например, распределение ресурсов, резервирование, несимметрично резервированное программное обеспечение, устойчивость к отказам, стратегии таймирования и диспетчеризирования;

• сертификационные соображения: Этот раздел содержит сводку сертификационного базиса, включая средства подтверждения соответствия, как это определяется программными аспектами сертификации. В этом разделе также заявляется предложенный уровень (уровни) программного обеспечения и приводятся подтверждения правильности этого уровня, полученные в процессе оценки отказобезопасности системы, включая потенциальный вклад программного обеспечения в отказные ситуации;

• жизненный цикл программного обеспечения: Этот раздел определяет жизненный цикл программного обеспечения, который будет использоваться, а также включает сводку его процессов, детальная информация о которых определяется в соответствующих планах программного обеспечения. В сводке разъясняется, как будут удовлетворяться цели каждого процесса жизненного цикла, указываются вовлекаемые организации, организационная ответственность, а также ответственность за процессы жизненного цикла системы и за процесс поддержания контактов в ходе сертификации;

• данные жизненного цикла программного обеспечения: Этот раздел определяет данные жизненного цикла, которые будут выпущены и будут контролироваться в процессах жизненного цикла программного обеспечения. Этот раздел также описывает взаимосвязь данных между собой или с другими данными, определяющими систему, данные жизненного цикла программного обеспечения, представляемые сертифицирующим властям, форму данных, и средства, с помощью которых данные жизненного цикла программного обеспечения могут быть сделаны доступными для сертифицирующих властей;

• план-график: Этот раздел описывает средства, которые заявитель будет использовать для того, чтобы обеспечить для сертифицирующих властей обозримость деятельности в процессах жизненного цикла программного обеспечения и, следовательно, возможность планирования проверок;

• дополнительные соображения: Этот раздел описывает особенности, которые могут повлиять на процесс сертификации, например, альтернативные методы подтверждения соответствия, квалификацию инструментальных средств, ранее разработанное программное обеспечение, вариантное программное обеспечение, которое может быть выбрано по желанию, программное обеспечение, доступное для модификации пользователем, готовое программное обеспечение COTS, используемое без модификаций, программное обеспечение, загружаемое в полевых условиях, несимметрично резервированное программное обеспечение или использование истории эксплуатации продукта.

В процессе самих сертификационных испытаний заявитель предоставляет свидетельства того, что процессы жизненного цикла программного обеспечения удовлетворяют планам программного обеспечения. Заявитель организует доступ сертифицирующего органа к данным жизненного цикла программного обеспечения. При этом минимальный перечень этих данных включает в себя:

• план сертификационных испытаний (план программных аспектов сертификации);

• индекс конфигурации программного обеспечения – документ, который должен однозначно идентифицировать каждый компонент проекта (включая требования, исходные коды, объектный и исполняемый код).

**Список использованных источников**

1 Котляров, В.П. Основы тестирования программного обеспечения /

В.П.Котляров. – М.: Интернет-Университет Информационных Технологий;

БИНОМ. Лаборатория знаний, 2006.

2 Липаев, В.В. Тестирование компонентов и комплексов программ /

В.В.Липаев. – Москва: СИНТЕГ, 2010.

3 http://textarchive.ru/c-1144105-p16.html

4 http://netnado.ru/dokumenti-vhfg1/addfile-12/index.html

5 https://www.simbirsoft.com/blog/tekhniki-test-dizayna-i-ikh-prednaznachenie/

6 https://artstroy.net/selenium-webdriver-vvedenie/