**Комплекс инструментальных средств настройки и администрирования программно-технических комплексов  
Системные программные утилиты «СПРУТ!»**

План верификации

**(на магнитном носителе)**

**Листов 43**

2021

**АННОТАЦИЯ**

Данный документ содержит план верификации программного обеспечения системных программных утилит «СПРУТ!» комплекса инструментальных средств настройки и администрирования программно-технических комплексов, реализуемых на базе КРОСС.

**СОДЕРЖАНИЕ**

[1. Введение 5](#_Toc88823439)

[2. Объект верификации 7](#_Toc88823440)

[3. Стратегия и организация верификации 10](#_Toc88823441)

[3.1. Стратегия верификации 10](#_Toc88823442)

[3.2. Организация верификации 12](#_Toc88823443)

[3.3. Участники верификации и распределение ответственности 14](#_Toc88823444)

[4. Методы и средства верификации 15](#_Toc88823445)

[4.1. Методы анализа и просмотра 15](#_Toc88823446)

[4.2. Методы тестирования 16](#_Toc88823447)

[4.2.1. Тестирование методом «белого ящика» 17](#_Toc88823448)

[4.2.2. Тестирование методом «черного ящика» 17](#_Toc88823449)

[4.3. Инструментальные средства разработки и верификации ПО 19](#_Toc88823450)

[5. Порядок проведения этапов верификации 21](#_Toc88823451)

[5.1. Этап 1. Верификация требований к ПО «СПРУТ!» 22](#_Toc88823452)

[5.1.1. Задачи 22](#_Toc88823453)

[5.1.2. Методы и критерии 22](#_Toc88823454)

[5.1.3. Входы и выходы 23](#_Toc88823455)

[5.2. Этап 2. Верификация проекта ПО «СПРУТ!» 24](#_Toc88823456)

[5.2.1. Задачи 24](#_Toc88823457)

[5.2.2. Методы и критерии 24](#_Toc88823458)

[5.2.3. Входы и выходы 25](#_Toc88823459)

[5.3. Этап 3. Верификация исходных кодов ПО «СПРУТ!» 26](#_Toc88823460)

[5.3.1. Задачи 26](#_Toc88823461)

[5.3.2. Методы и критерии 26](#_Toc88823462)

[5.3.3. Входы и выходы 28](#_Toc88823463)

[5.4. Этап 4. Функциональное тестирование и верификация результатов тестирования ПО «СПРУТ!» 29](#_Toc88823464)

[5.4.1. Задачи 29](#_Toc88823465)

[5.4.2. Методы и критерии 30](#_Toc88823466)

[5.4.3. Входы и выходы 31](#_Toc88823467)

[6. Отчетность по верификации 32](#_Toc88823468)

[7. Анализ результатов верификации 35](#_Toc88823469)

[7.1. Анализ результатов верификации требований к ПО 35](#_Toc88823470)

[7.2. Анализ результатов верификации требований к ПО 36](#_Toc88823471)

[7.3. Анализ результатов верификации исходных кодов 37](#_Toc88823472)

[7.4. Анализ результатов верификации на этапе функционального тестирования 38](#_Toc88823473)

[Перечень сокращений 41](#_Toc88823474)

[Перечень ссылочных документов 42](#_Toc88823475)

# **Введение**

Настоящий план предназначен для определения стратегии и организации мероприятий по верификации, необходимых для достижения требуемого качества при разработке программного обеспечения, комплекса инструментальных программных средств для создания и администрирования программно-технических комплексов (ПТК) автоматизированных систем контроля и управления технологических процессов на базе комплекса распределенных средств сетевой обработки (КРОСС) (краткое название «СПРУТ!»). ПО «СПРУТ!» должно войти в состав КРОСС в качестве инструментального средства, предназначенного для автоматизации выполнения вспомогательных функций ПТК, таких как:

* управление конфигурацией ПТК, входящих в состав автоматизированных систем контроля и управления (АСКУ);
* централизованный контроль состояния программных и технических средств узлов ПТК;
* управление узлами ПТК;
* управление учетными записями пользователей и группами пользователей;
* контроль целостности пакетов ПО;
* восстановление ПО узлов ПТК АСКУ;
* выполнение задач периодического обслуживания ПТК.

Определяемые данным планом мероприятия по верификации программного обеспечения «СПРУТ!» должны обеспечить как реализацию всех требований, предъявляемых к программному обеспечению, так и выявление и устранение всех несоответствий на ранних стадиях проектирования.

План разработан в соответствии с требованиями ГОСТ Р МЭК 61513 [1], ГОСТ Р МЭК 62138 [2] и устанавливает объем, методы и средства проверки, критерии оценки соответствия характеристик и функций программного обеспечения «СПРУТ!».

# **Объект верификации**

Объектом верификации по настоящему плану является инструментальный программный комплекс «СПРУТ!», который верифицируется в процессе разработки, а затем в процессе сопровождения.

ПО «СПРУТ!» должно обеспечивать:

* управление конфигурацией ПТК, входящих в состав автоматизированных систем контроля и управления (АСКУ);
* централизованный контроль состояния программных и технических средств узлов ПТК;
* управление узлами ПТК;
* управление учетными записями пользователей и группами пользователей;
* контроль целостности пакетов ПО;
* восстановление ПО узлов ПТК АСКУ;
* выполнение задач периодического обслуживания ПТК.

ПО «СПРУТ!» представляет собой набор программных средств, необходимых пользователю для управления конфигурацией программно-технических комплексов автоматической системы управления техническим процессом (далее – ПТК АСУ ТП), наладки ПТК АСУ ТП, выполнения задач периодического обслуживания комплекса.

ПО «СПРУТ!» состоит из следующих компонентов:

* клиента комплекса «СПРУТ!»;
* сервера комплекса «СПРУТ!», в который входят:

1. модуль управления конфигурацией узлов;
2. модуль управления узлами;
3. модуль диагностики;
4. конфигурационные файлы;
5. база данных;
6. репозиторий.

Функциональным назначением комплекса «СПРУТ!» является выполнение следующих административных функций:

* централизованная настройка конфигурации программно-технического комплекса верхнего уровня (далее ВУ) АСУ ТП;
* поддержка возможности настройки конфигураций ПТК АСУ ТП для нескольких проектов на одном сервере программного обеспечения (далее СПО);
* установка/переустановка узлов ПТК;
* установка по сети системного, программного обеспечения (далее ПО), прикладного ПО, проведение тонкой настройки ПО узлов ПТК;
* обновление ПО;
* проверка и поддержание целостности ПО узлов ПТК;
* обеспечение удобного интерфейса для периодического обслуживания ПТК администратором системы.

# **Стратегия и организация верификации**

## **Стратегия верификации**

Целями верификации ПО «СПРУТ!» являются:

* создание высоконадежного ПО за счет наиболее раннего выявления ошибок проектирования и кодирования ПО, обнаружения потенциально опасных и (или) нестабильных в работе проектных решений и способов кодирования ПО;
* доказательство того, что результат разработки ПО соответствует, предъявленным к нему требованиям.

При выборе стратегии верификации ПО «СПРУТ!» учитывались требования ГОСТ Р МЭК 62138 [2], который устанавливает требования к процессу верификации ПО, выполняющего функции категорий В и С.

Процесс верификации проводится сверху вниз, начиная от общих требований, заданных в техническом задании. Цели верификации ПО достигаются посредством последовательного выполнения следующих мероприятий:

* просмотр и анализ требований к ПО - цель просмотра и анализа обнаружить и зарегистрировать ошибки, которые могли быть внесены в процессе разработки требований к ПО;
* просмотр и анализ проекта ПО - цель просмотра и анализа обнаружить и зарегистрировать ошибки, которые могли быть внесены во время разработки проекта ПО и что проект ПО соответствует требованиям к ПО;
* просмотр и анализ исходного кода - цель этих просмотра и анализа выявление и регистрация ошибок, которые могли быть внесены в процессе кодирования ПО. Просмотр и анализ подтверждают, что выходные результаты кодирования являются точными, полными и могут быть верифицированы. Прежде всего проверяются корректность кода по отношению к требованиям к ПО и архитектуре ПО и соответствие стандартам на кодирование;
* функциональное тестирование ПО «СПРУТ!» - тестирование проводится методом «черного ящика». В ходе данного тестирования проверяются все функциональные требования системы. Функциональное тестирование нацелено на выявление следующих ошибок: неверная или пропущенная функциональность, ошибки интерфейса, ошибки в структурах данных или ошибки доступа к базам данных, проблемы снижения производительности, ошибки загрузки, ошибки многопользовательского доступа, ошибки инициализации и завершения, проблемы сохранения резервных копий и способности к восстановлению работы, проблемы безопасности;
* анализ тестового покрытия – в ходе анализа должно быть подтверждено, что тестовые наборы основаны на требованиях к ПО, что все требования к ПО были протестированы. Должны быть разработаны тестовые наборы для проверки функционирования в области допустимых значений и для проверки на устойчивость к ошибкам входных данных. Так же должны быть разработаны тестовые наборы для проверки устойчивости ПО при возникновении ошибок и возникновении неисправности ТС.

На всех этапах разработки ПО «СПРУТ!» должна осуществляться проверка согласованности результатов, полученных на каждом этапе разработки с требованиями, установленными на предыдущих этапах, и получение данных, обеспечивающих анализ и оценку показателей качества реализации требований.

## **Организация верификации**

Верификация должна проводиться компетентными лицами, не принимавшими участие в разработке ПО «СПРУТ!».

Верификация ПО должна проводиться на каждом этапе жизненного цикла ПО в соответствии с данным Планом.

Все обнаруженные в ходе верификации несоответствия должны быть устранены до начала следующего этапа разработки ПО.

Верификация ПО должна проводиться параллельно с процессом разработки с тем, чтобы можно было своевременно зарегистрировать и исправить обнаруженные ошибки.

Каждый этап верификации должен завершаться оформлением отчета, в котором должны быть указаны:

* объект верификации;
* оцениваемые показатели;
* порядок проведения проверки (методика верификации данного этапа, выполняемые действия, контролируемые параметры и данные);
* используемые программно-технические средства;
* критерии принятия решения, о соответствии проверяемого ПО и (или) документации предъявленным требованиям;
* результаты проверки ПО и (или) документации с указанием соответствия (несоответствия) предъявленным требованиям;
* выводы об успешном выполнении этапа верификации, либо о необходимости доработки ПО и (или) документации и проведении повторной верификации, включая, при необходимости, предыдущие этапы;
* перечень выявленных ошибок и недостатков (при обнаружении таковых), которые должны быть устранены на данном этапе проектирования, оформленный в виде приложения к протоколу верификации;
* перечень выявленных ошибок и недостатков (при обнаружении таковых), которые затрагивают предыдущие этапы и требуют проведения повторной верификации, а также перечень функций, верификация которых невозможна на данном этапе и переносится на последующие этапы, оформленные в виде приложений к протоколу верификации;
* заключение о возможности перехода к верификации следующего этапа.

Если при устранении несоответствия требуется корректировка, то откорректированный вариант документа или программного продукта должен быть повторно предъявлен на верификацию.

Верификация следующего этапа возможна только после устранения всех обнаруженных несоответствий и проведения повторной верификации измененных фрагментов ПО.

## **Участники верификации и распределение ответственности**

Верификацию ПО «СПРУТ!» проводит верификационная группа из числа сотрудников отдела компьютерных и интегрированных систем.

Верификацию требований к ПО «СПРУТ!» проводят сотрудники отдела компьютерных и интегрированных систем не участвующие в разработке ТЗ на ПО «СПРУТ!».

Верификацию проекта ПО «СПРУТ!» проводят сотрудники отдела компьютерных и интегрированных систем не участвующие в разработке проекта ПО «СПРУТ!».

Верификацию исходных кодов ПО «СПРУТ!» проводят сотрудники отдела компьютерных и интегрированных систем не участвующие в кодировании ПО «СПРУТ!».

Верификацию результатов функционального тестирования ПО «СПРУТ!» проводят сотрудники отдела компьютерных и интегрированных систем не участвующие в разработке требований к ПО «СПРУТ!» и не участвующие в кодировании ПО «СПРУТ!».

# **Методы и средства верификации**

## **Методы анализа и просмотра**

Методы анализа и просмотра (обзора) должны применяться для верификации проектной документации, исходных кодов и тестового покрытия. Анализ должен обеспечить доказательство корректности разрабатываемых компонентов ПО и детально исследовать функциональность, эффективность, прослеживаемость и надежность компонентов ПО.

Анализ трассируемости должен подтвердить полноту реализации на текущем этапе всех требований предыдущего этапа разработки.

Просмотр (обзор) должен обеспечить качественную оценку степени корректности компонентов ПО и может включать методы инспекции и проверки. Метод инспекции заключается в рассмотрении документации на предмет отсутствия ошибок и аномалий. Метод проверки – в установлении соответствия документации, разработанной на предыдущем этапе проектирования.

Просмотры и анализы должны проводиться сверху вниз, начиная от общих требований до детальных требований на модули и их взаимодействие.

Просмотры и анализы должны обеспечивать оценку точности, полноты и верифицируемости требований, архитектуры ПО, а также исходных кодов программ.

## **Методы тестирования**

Методы тестирования должны применяться для верификации ПО на этапах автономного и функционального тестирования.

Реализация тестирования делится на три этапа:

1. Создание тестового набора путем ручной разработки или автоматической генерации для конкретной среды тестирования.
2. Прогон программы на тестах с оформлением протокола тестирования.
3. Оценка результатов выполнения программы на наборе тестов с целью принятия решения о продолжении или остановке тестирования.

Тестирование должно обеспечить проверку внутренней непротиворечивости и полноты реализации требований и подтвердить соответствие испытываемых компонентов ПО исходным функциональным требованиям, а также выявить все несоответствия и различия между ожидаемыми и полученными результатами для последующего исследования и устранения.

Помимо тестирования общего функционирования должно быть предусмотрено стрессовое тестирование для критических ситуаций (вне области допустимых требованиями входных данных), важных для обеспечения качества и надежности функционирования ПО «СПРУТ!».

### Тестирование методом «белого ящика»

Тестирование методом «белого ящика» включает в себя стратегию модульного тестирования, при котором тестирование ведется на модульном уровне и работы по тестированию направлены на исследование внутреннего устройства модуля. На этом уровне тестирования проверяется управляющая логика, проявляющаяся на модульном уровне. В ходе модульного тестирования каждый модуль тестируется как на соответствие требованиям, так и на отсутствие проблемных участков программного кода, могущих вызвать отказы и сбои в работе ПО. Тестовые наборы разрабатываются для того, чтобы все ветви в данном модуле были проверены хотя бы один раз, все логические решения рассмотрены во всевозможных условиях, циклы были выполнены с использованием верхних и нижних границ и проконтролированы структуры внутренних данных. Проверяется обработка как верных, так и неверных входных данных, а так же выдача сообщений об ошибках.

### Тестирование методом «черного ящика»

Тестирование «чёрного ящика»  это метод тестирования функционального поведения программы с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве тестируемого объекта. Под стратегией понимаются систематические методы отбора и создания тестов для тестового набора. Стратегия поведенческого теста исходит из требований к ПО. При тестировании функционального поведения используются следующие методы:

1. Эквивалентное разбиение. Тестирование с использованием подмножества входных данных. Применяют три основных типа эквивалентных классов: значения внутри границы диапазона, за границей диапазона и на границе.
2. Функциональное тестирование. Функциональное тестирование проверяет ПО в отношении функциональных требований с целью обнаружения несоответствия требованиям к ПО.
3. Тестирование безопасности. Тестирование безопасности включает в себя проверку работы механизмов доступа к системе и к данным. Для этого разрабатываются тестовые процедуры, которые пытаются преодолеть защиту системы. Во время тестирования проверяется степень безопасности и ограничения доступа, выявляя таким образом соответствие установленным требованиям к безопасности и всем применяемым правилам по безопасности ПО.
4. Тестирование перегрузок. При тестировании перегрузок выполняется проверка ПО без учета ограничений архитектуры с целью выявления технических ограничений ПО. Эти тесты проводятся на пике обработки транзакций и при непрерывной загрузке большого объема данных. Тестирование перегрузок измеряет пропускную способность ПО и его эластичность на всех аппаратных платформах. От системы требуется обработка огромного количества данных или выполнение большого числа функциональных запросов в течение короткого периода времени.
5. Тестирование производительности. Тесты производительности проверяют, удовлетворяет ли программное приложение требованиям по производительности. Применяя тестирование производительности, можно замерить и составить отчеты по таким показателям, как скорость передачи входных и выходных данных, общее число действий по вводу и выводу данных, среднее время, затрачиваемое базой данных на отклик на запрос, и интенсивность использования центрального процессора.

## **Инструментальные средства разработки и верификации ПО**

Инструментальные средства разработки ПО «СПРУТ!» (САПР, транслятор, компилятор) должно использоваться из числа ранее применяемых на предприятии для аналогичного ПО.

Инструментальные средства верификации (тестовое ПО, стенды проверки блоков связи с объектом) должно использоваться из числа проверенных и уже примененных на предприятии.

Инструментальные средства тестирования (тестовое ПО, макеты или имитаторы) должно использоваться из числа проверенных и уже примененных на предприятии для тестирования подобного ПО. При необходимости инструментальные средства тестирования должны быть модифицированы по отдельным требованиям и проверены.

Используемые для подтверждения правильности функционирования аппаратные и программные средства специальной верификации не подвергаются. Документально должно подтверждаться, что программные и аппаратные средства соответствуют своему назначению.

Структура и состав средств, используемых в процессе верификации должны быть уточнены в процессе проведения верификации каждого компонента ПО и приведены в соответствующих программах и методиках.

Список инструментального ПО используемого при разработке ПО «СПРУТ!» приведен в табл. 1.

Т а б л и ц а 1

| Название ПО | Версия | Описание |
| --- | --- | --- |
| Разработка БД «СПРУТ!» | | |
| PostgreSQL | 9.4 | СУБД PostgreSQL |
| Разработка ПО «СПРУТ!» | | |
| Qt | 5.5.1 | Кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++ |
| Ruby | 1.9.2 | Высокоуровневый язык программирования |

# **Порядок проведения этапов верификации**

На основании рекомендаций ГОСТ Р ИСО/МЭК 12207 [3] для разработки программного обеспечения «СПРУТ!» в соответствии с V-моделью жизненного цикла программных средств, жизненный цикл разрабатываемого ПО должен разбиваться на следующие этапы:

* разработка требований к ПО «СПРУТ!»;
* разработка проекта ПО «СПРУТ!»;
* кодирование;
* функциональное тестирование.

Работы по верификации программного обеспечения «СПРУТ!», разделяются на следующие этапы:

* этап 1 – верификация требований к ПО «СПРУТ!»;
* этап 2 – верификация проекта ПО «СПРУТ!»;
* этап 3 – верификация исходных кодов ПО «СПРУТ!»;
* этап 4 – функциональное тестирование и верификация результатов тестирования ПО «СПРУТ!».

## **Этап 1. Верификация требований к ПО «СПРУТ!»**

### Задачи

Установление соответствия требований ТЗ на ПО «СПРУТ!» требованиям нормативных документов.

Подтверждение что функции, выполняемые ПО «СПРУТ!» определены, функциональные, технические требования и требования по безопасности описаны.

Обнаружение ошибок в требованиях, которые могли быть допущены в процессе разработки требований к ПО «СПРУТ!».

Регистрация обнаруженных ошибок.

Проверка корректирующих действий проводимых по исправлению обнаруженных ошибок.

Проверка внесенных исправлений в ТЗ на ПО «СПРУТ!», в случае необходимости.

### Методы и критерии

Методом проведения процедур верификации на данном этапе является экспертная оценка формулируемых целей и задач исходя из нормативных документов, выбранных для проведения процесса верификации.

В качестве критерия для проводимых оценок выступает требование максимального удовлетворения рекомендациям по проведению процедур верификации, изложенным в стандарте ГОСТ Р МЭК 62138 [2].

В ходе верификации требований ТЗ на ПО «СПРУТ!» методом просмотра и анализа проверяется:

* отсутствие противоречий между требованиями ТЗ на ПО «СПРУТ!», предъявляемыми к разра­батываемому ПО, и требованиями действующих НД;
* отсутствие противоречий между отдельными требованиями ТЗ на ПО «СПРУТ!»;
* полнота требований ТЗ на ПО «СПРУТ!» – оценивается наличие требований для всех режимов работы ПО и наличие требований по взаимодействию с внешними программными средствами;
* точность требований ТЗ на ПО «СПРУТ!» – оценивается четкость формулировок требований ТЗ на ПО, не допускающая их неоднозначную трактовку;
* верифицируемость требований ТЗ на ПО – оценивается, что каждое требование может быть верифицировано;
* реализуемость требований ТЗ на ПО - оценивается, что каждое требование может быть реализуемо.

### Входы и выходы

Входными данными процесса верификации на этапе верификации требований к ПО «СПРУТ!» являются:

* план верификации программного обеспечения «СПРУТ!»;
* техническое задание.

Выходные результаты процесса верификации на этапе верификации требований к ПО «СПРУТ!» содержатся в следующих документах:

* отчет по верификации требований к программному обеспечению «СПРУТ!»;
* запросы на изменение, сформированные в ходе верификации требований к ПО «СПРУТ!».

## **Этап 2. Верификация проекта ПО «СПРУТ!»**

### Задачи

Обнаружение ошибок в проекте ПО «СПРУТ!» которые могут быть допущены в процессе проектирования ПО.

Регистрация обнаруженных ошибок.

Проверка корректирующих действий проводимых по исправлению обнаруженных ошибок.

Проверка внесенных исправлений в проект ПО и в ТЗ на ПО «СПРУТ!», в случае необходимости.

### Методы и критерии

В ходе верификации проекта ПО «СПРУТ!» методом просмотра и анализа проверяется:

* отсутствие несоответствий между требованиями ТЗ на ПО «СПРУТ!» и требованиями проекта ПО;
* отсутствие противоречий между отдельными требованиями проекта ПО «СПРУТ!»;
* точность требований проекта ПО «СПРУТ!» – оценивается четкость формулировок требований проекта ПО, не допускающая их неоднозначную трактовку;
* совместимость с объектным компьютером: гарантировать, что не существует никаких противоречий между архитектурой ПО и программно-аппаратными возможностями объектного компьютера;
* верифицируемость требований проекта ПО – оценивается, что каждое требование может быть верифицировано;
* трассируемость требований проекта ПО – оценивается, что требования технического задания были реализованы в требованиях проекта ПО.
* реализуемость требований проекта ПО - оценивается, что каждое требование может быть реализовано.

### Входы и выходы

Входными данными процесса верификации на этапе верификации проекта ПО «СПРУТ!» являются:

* план верификации программного обеспечения «СПРУТ!»;
* техническое задание;
* описание программы.

Выходные результаты процесса верификации на этапе верификации проекта ПО «СПРУТ!» содержатся в следующих документах:

* отчет по верификации проекта программного обеспечения «СПРУТ!»;
* запросы на изменение, сформированные в ходе верификации проекта ПО «СПРУТ!».

## **Этап 3. Верификация исходных кодов ПО «СПРУТ!»**

### Задачи

Обнаружение ошибок в исходном коде ПО «СПРУТ!» которые могут быть допущены в процессе кодирования ПО.

Регистрация обнаруженных ошибок.

Проверка корректирующих действий проводимых по исправлению обнаруженных ошибок.

Проверка внесенных исправлений в проект ПО, в ТЗ на ПО «СПРУТ!» и в исходный код в случае необходимости.

Подтверждение, что результаты процесса кодирования точны, полны и верифицируемы.

### Методы и критерии

Методом просмотра и анализа проведения процедур верификации на данном этапе является независимый сквозной аудит текстов исходных кодов программ из состава ПО «СПРУТ!» и экспертная оценка выпущенной программной документации.

В качестве критерия для проводимой верификации выступают формальные правила написания программ на языках программирования Си++ и Ruby, отсутствие синтаксических и логических ошибок в текстах программ, соответствие разработанного программного кода требованиям проекта.

В ходе верификации исходного кода ПО «СПРУТ!» методом просмотра и анализа проверяется:

* исходный код читаем, понятен и пригоден к проверке;
* исходный код соответствует требованиям проекта и технического задания;
* исходный код не содержат потенциально опасных участков;
* исходный код не содержат вредоносный код;
* исходный код не содержат недокументированных функций;
* исходный код согласуется с потоком данных и потоком управления, которые определены в архитектуре ПО;
* исходный код не содержит операторов и структур, которые не могут быть проверены, и для испытаний в него не требуется вносить изменения;
* правильность и непротиворечивость исходного кода, включая оценку конкуренции в использовании ресурсов, обработки исключительных ситуаций, использования неинициализированных переменных или констант, неиспользуемых переменных или констант, а также нарушения целостности данных.

### Входы и выходы

Входными данными процесса верификации на этапе верификации исходных кодов ПО «СПРУТ!» являются:

* план верификации программного обеспечения «СПРУТ!»;
* техническое задание;
* описание программы;
* текст программы.

Выходные результаты процесса верификации на этапе верификации исходного кода ПО «СПРУТ!» содержатся в следующих документах:

* отчет по верификации исходного кода программного обеспечения «СПРУТ!»;
* запросы на изменение, сформированные в ходе верификации исходного кода ПО «СПРУТ!».

## **Этап 4. Функциональное тестирование и верификация результатов тестирования ПО «СПРУТ!»**

### Задачи

Проверка что существуют тестовые наборы для каждого требования к ПО.

Проверка что тестовые наборы разработаны так, чтобы верифицировать корректность функционирования и сформировать условия, которые выявляют потенциальные ошибки.

Функциональное тестирование ПО «СПРУТ!», в ходе которого должно быть продемонстрировано, что программный комплекс «СПРУТ!» выполняет требования ТЗ.

Обнаружение ошибок в коде ПО «СПРУТ!» которые могут быть допущены в процессе разработки ПО.

Регистрация обнаруженных ошибок.

Проверка корректирующих действий проводимых по исправлению обнаруженных ошибок.

Проверка внесенных исправлений в проект ПО, в ТЗ на ПО «СПРУТ!» и в исходный код в случае необходимости.

Проверка результатов тестирования на предмет, что испытания выполнены в полном объеме в соответствии с тестовыми процедурами и результаты испытаний корректны.

### Методы и критерии

В ходе верификации тестовых наборов для функционального тестирования компонентов ПО «СПРУТ!» методом анализа и обзора проверяется:

* соответствие требований ТЗ и методов испытаний;
* наличие тестовых наборов для каждого требования к ПО;
* наличие тестовых наборов для тестирования на устойчивость к ошибкам входных данных;
* наличие тестовых наборов, разработанных на основе требований к ПО с учетом потенциальных источников ошибок.

Функциональное тестирование осуществляется методом «черного ящика». В качестве критерия для функционального тестирования выступает:

* ПО «СПРУТ!» функционирует в соответствии с требованиями ТЗ;
* компоненты ПО «СПРУТ!» полностью и правильно интегрированы в программный комплекс;
* ПО «СПРУТ!» правильно взаимодействуют с техническими средствами;
* ПО «СПРУТ!» устойчиво функционируют при отказах и сбоях технических средств, ошибках во входных данных;
* все данные, которыми обменивается ПО «СПРУТ!», соответствуют проектным спецификациям структур данных.

### Входы и выходы

Входными данными процесса верификации на этапе функционального тестирования компонентов ПО «СПРУТ!» являются:

* план верификации программного обеспечения «СПРУТ!»;
* техническое задание;
* тестовые процедуры и тестовые наборы для функционального тестирования.

Выходные результаты процесса верификации на этапе функционального тестирования компонентов ПО «СПРУТ!» содержатся в следующих документах:

* протоколы функционального тестирования;
* отчет по верификации на этапе функционального тестирования ПО «СПРУТ!»;
* запросы на изменение, сформированные в ходе функционального тестирования и верификации ПО «СПРУТ!».

# **Отчетность по верификации**

Отчетная документация по верификации программного обеспечения «СПРУТ!» должна включать:

* отчеты по выполнению верификации на каждом из этапов;
* запросы на изменение, сформированные в ходе верификации на каждом из этапов;
* протоколы или отчеты автономного тестирования программного обеспечения;
* протоколы функционального тестирования программного обеспечения.

Отчет по верификации требований должен включать следующие пункты:

* задачи верификации на этапе разработки требований к ПО;
* методы и критерии верификации;
* результаты верификации;
* заключение.

Отчет по верификации проекта ПО должен включать следующие пункты:

* задачи верификации на этапе разработки проекта ПО;
* методы и критерии верификации;
* результаты верификации;
* заключение.

Отчет по верификации исходных кодов должен включать следующие пункты:

* задачи верификации на этапе кодирования ПО;
* методы и критерии верификации;
* результаты верификации;
* заключение.

Отчет по верификации на этапе функционального тестирования должен включать следующие пункты:

* задачи верификации на этапе функционального тестирования;
* методы и критерии верификации;
* результаты верификации;
* заключение.

Запрос на изменение должен включать следующие пункты:

* номер запроса;
* автор запроса;
* описание проблемы (описание причин возникновения изменений, влияние предлагаемых изменений, к чему может привести невыполнение предлагаемых изменений);
* решение о корректирующих действиях;
* отметка о реализации корректирующих действий;
* краткий отчет о проверке внесенных изменений;
* статус (подан/утвержден/отложен/отклонен/закрыт).

Протоколы функционального тестирования программного обеспечения должны включать следующие пункты:

* объект испытаний;
* цель испытаний;
* конфигурацию технических средств, примененную при тестировании;
* дополнительные данные, связанные со временем, последовательностью событий и т.д.;
* отчеты о несоответствиях заданию;
* заключение о соответствии критериям приемки.

# **Анализ результатов верификации**

## **Анализ результатов верификации требований к ПО**

Все мероприятия, проводимые в ходе верификации требований, должны быть зафиксированы в отчете по верификации требований. Должен быть проведен анализ, что все требуемые мероприятия выполнены.

Все обнаруженные в ходе верификации требований несоответствия должны быть зафиксированы в отчете по верификации требований.

На каждое несоответствие необходимо сформировать Запрос на изменение в системе управления проектами Redmine.

Сформированный Запрос на изменение должен быть проанализирован и вынесено решение о корректирующих действиях по устранению данного несоответствия.

После устранения несоответствий выявленных в ходе верификации требований к ПО необходимо провести повторную верификацию скорректированных требований.

Результаты повторной верификации должны быть зафиксированы в отчете по верификации требований.

Результаты повторной верификации должны быть проанализированы, в случае отсутствия несоответствий принимается решение о переходе к следующему этапу разработки ПО.

## **Анализ результатов верификации требований к ПО**

Все мероприятия, проводимые в ходе верификации проекта ПО, должны быть зафиксированы в отчете по верификации проекта ПО. Должен быть проведен анализ, что все требуемые мероприятия выполнены.

Все обнаруженные в ходе верификации проекта ПО несоответствия должны быть зафиксированы в отчете по верификации проекта ПО.

На каждое несоответствие необходимо сформировать Запрос на изменение в системе Redmine.

Сформированный Запрос на изменение должен быть проанализирован и вынесено решение о корректирующих действиях по устранению данного несоответствия.

В случае если корректирующие действия затрагивают только проект ПО, после устранения несоответствий выявленных в ходе верификации проекта ПО необходимо провести повторную верификацию скорректированных требований проекта ПО.

В случае если корректирующие действия затрагивают и требования ТЗ и требования проекта ПО, после устранения несоответствий выявленных в ходе верификации проекта ПО необходимо провести повторную верификацию скорректированных требований ТЗ и проекта ПО.

Результаты повторной верификации должны быть зафиксированы в отчете по верификации проекта ПО.

Результаты повторной верификации должны быть проанализированы, в случае отсутствия несоответствий принимается решение о переходе к следующему этапу разработки ПО.

## **Анализ результатов верификации исходных кодов**

Все мероприятия, проводимые в ходе верификации исходных кодов ПО, должны быть зафиксированы в отчете по верификации исходных кодов. Должен быть проведен анализ, что все требуемые мероприятия выполнены.

Все обнаруженные в ходе верификации исходных кодов ошибки и несоответствия должны быть зафиксированы в отчете по верификации.

На все обнаруженные ошибки и несоответствия необходимо сформировать Запрос на изменение в системе Redmine.

Сформированный Запрос на изменение должен быть проанализирован и вынесено решение о корректирующих действиях по устранению данной ошибки или несоответствия.

В случае если корректирующие действия затрагивают только исходный код, необходимо провести повторную верификацию измененного исходного кода.

В случае если корректирующие действия затрагивают исходный код и требования проекта ПО, необходимо провести повторную верификацию измененного исходного кода и скорректированных требований проекта ПО.

В случае если корректирующие действия затрагивают требования ТЗ, требования проекта ПО и исходный код, необходимо провести повторную верификацию скорректированных требований ТЗ и проекта ПО и измененного исходного кода.

Результаты повторной верификации должны быть зафиксированы в отчете по верификации проекта ПО.

Результаты повторной верификации должны быть проанализированы, в случае отсутствия несоответствий принимается решение о переходе к следующему этапу разработки ПО.

## **Анализ результатов верификации на этапе функционального тестирования**

Все мероприятия, проводимые в ходе верификации на этапе функционального тестирования, должны быть зафиксированы в отчете по верификации. Должен быть проведен анализ, что все требуемые мероприятия выполнены.

Функциональное тестирование должно быть выполнено в соответствии с заранее определенными тестовыми наборами и тестовыми процедурами.

Результаты функционального тестирования должны быть зафиксированы в протоколах тестирования.

Протоколы тестирования должны быть проанализированы на предмет:

* все тестовые процедуры выполнены;
* результаты всех тестов зафиксированы.

Должен быть проведен анализ тестового покрытия, в случае определения необходимости в дополнительном тестировании необходимо сформировать Запрос на изменение в системе Redmine. Данный запрос должен быть проанализирован и вынесено решение о создании дополнительных тестовых процедур и/или тестовых наборов и проведении дополнительного тестирования. По результатам дополнительного тестирования оформляется протокол.

Все обнаруженные в ходе верификации на этапе функционального тестирования ошибки и несоответствия должны быть зафиксированы в отчете по верификации.

На все обнаруженные ошибки и несоответствия необходимо сформировать Запрос на изменение в системе Redmine.

Сформированный Запрос на изменение должен быть проанализирован и вынесено решение о корректирующих действиях по устранению данной ошибки или несоответствия.

В случае если корректирующие действия затрагивают исходный код, требования проекта ПО или требования ТЗ необходимо провести повторную верификацию измененного исходного кода, проекта ПО и требований ТЗ.

Результаты повторной верификации должны быть зафиксированы в отчете по верификации на этапе функционального тестирования.

Результаты повторной верификации должны быть проанализированы, в случае отсутствия несоответствий принимается решение о переходе к следующему этапу разработки ПО.

**Перечень сокращений**

|  |  |  |
| --- | --- | --- |
| АСКУ | – | автоматизированная система контроля и управления |
| АСУ ТП | – | автоматизированная система управления технологическим процессом |
| АЭС | – | атомная электростанция |
| БД | – | база данных |
| ВУ | – | верхний уровень |
| КРОСС | – | комплекс распределенных средств сетевой обработки |
| НД | – | нормативная документация |
| ПО | – | программное обеспечение |
| ПТК | – | программно-технический комплекс |
| САПР | – | система автоматизированного проектирования |
| СПО | – | сервер программного обеспечения |
| СУБД | – | система управления базами данных |
| ТЗ | – | техническое задание |
| ТС | – | технические средства |

**Перечень ссылочных документов**

1. ГОСТ Р МЭК 61513-2020 «Атомные станции. Системы контроля и управления, важные для безопасности атомной станции. Общие требования»

2. ГОСТ Р МЭК 62138-2010 «Атомные электростанции. Системы контроля и управления, важные для безопасности. Программное обеспечение компьютерных систем, выполняющих функции категории В и С»

3. ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств»

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц)  в  документе | Номер  документа | Входящий номер сопрово-дительного документа и дата | Подпись | Дата |
| изме-ненных | заме-  ненных | новых | аннулиро-ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |