МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**Национальный исследовательский ядерный университет «МИФИ»**

 **Институт Интеллектуальных Кибернетических Систем**

**КАФЕДРА КИБЕРНЕТИКИ**

**Направление подготовки 09.04.04 Программная инженерия**

**План управления конфигурацией ПО**

“Веб ресурс StudMephi”

|  |  |
| --- | --- |
| Выполнили | Андрей Совков  Азат Султанов  Никита Румянцев  Светлана Тихомирова  Анастасия Богатырева  Анна Виноградова  Лидия Антонова  Стас Сегаль  Никита Олейников  Алексей Ильичёв  Александр Слиньков  Полина Горбатенко  Алексей Локтионов  Максим Ключников  Дмитрий Буянов  Дмитрий Сорокин |
| № группы | Б15-506, Б15-503 |
| Руководители | Козырев В.П. |

**Москва 2018**

**Оглавление**

[**1. Введение**](#_o9kzefdb2rzc) **3**

[1.1. Цель](#_58njkd3r10kh) 3

[1.2. Область применения](#_ngy0gof32s3a) 3

[1.3. Ссылки](#_cr0zakjxz0yf) 3

[1.4. Термины, определения и соглашения](#_sc0y3rr301gh) 3

[**2. Среда УК ПО**](#_zhqkpcu63d7y) **5**

[2.1. Инструменты УК ПО](#_bwmtm4rsx2vw) 5

[2.2. Организация и ответственности](#_5y6filtk3iiw) 5

[2.3. Взаимодействие УК ПО с другими процессами ЖЦ ПО](#_5e1xllbv3ddn) 6

[**3. Мероприятия процесса УК ПО**](#_zc4ls9hoty9x) **6**

[3.1. Идентификация конфигурации](#_1co12ufk8tuh) 6

[3.2. Базовые версии и трассируемость](#_vps8uhovmo4x) 7

[3.2.1. Каталог планов и стандартов](#_pyblxns62t6x) 8

[3.2.2. Каталог среды жизненного цикла ПО](#_jk7xqtmx4xzd) 8

[3.2.3. Каталог комплектации ПО](#_35wblfdol5az) 8

[3.2.4. Трассируемость](#_rpke7pj9plo) 8

[3.3. Регистрация проблем и управление изменениями](#_z9of5tj8fg72) 9

[3.3.1. Регистрация проблем](#_ikgm52nth7hx) 9

[3.3.2. Рассмотрение изменений](#_gaezmw2gazi) 10

[3.3.3. Внесение изменений](#_3htppp65j4ku) 11

[3.3.4. Закрытие сообщения о проблеме](#_avttgj5d4hjr) 12

[3.3.5. Жизненный цикл сообщения о проблеме](#_mrxc3ohbvpsh) 13

[3.3.6. Жизненный цикл запроса на изменение](#_gs4mpuden0fb) 14

[3.3.7. Регистрация проблем и управление изменениями после сертификации](#_o56sbcd40j9w) 15

[3.4. Учет состояния конфигурации](#_cu2uonmumhk1) 15

[3.5. Архивирование, воспроизведение и выпуск](#_xhi03yietwaz) 15

[3.6. Контроль среды ЖЦ ПО](#_lqjoqqzb3uz3) 16

[3.6.1. Контроль планов и стандартов](#_133h3ypy4bb4) 16

[3.6.2. Контроль инструментальных средств](#_cxz5r3pr1wa0) 16

[3.7. Контроль данных (документов) ЖЦ ПО](#_gu9ndb6pfdj) 16

[**4. Критерии перехода**](#_48xlobws014t) **18**

[**5. Документы УК ПО**](#_omy125pwoccq) **18**

[**Приложение А. Данные ЖЦ ПО**](#_tm7q5imog0li) **18**

# 1. Введение

## 1.1. Цель

Данный документ определяет мероприятия, которые будут выполняться в рамках проекта “Веб-ресурс StudMephi” для достижения целей процесса управления конфигурацией ПО, определенных в документе КТ-178B [1].

## 1.2. Область применения

Требования и положения настоящего Плана распространяются на все работы, выполняемые в рамках данного проекта “Веб-ресурс StudMephi”.

Положения, описанные в данном документе, могут быть неприменимы при выполнении реального проекта и должны быть пересмотрены с учетом требований реального проекта в каждом конкретном случае.

## 1.3. Ссылки

[1] КТ-178В. Квалификационные требования. Часть 178В. Требования к ПО бортовой аппаратуры и систем при сертификации авиационной техники.

[2] План разработки ПО.

[3] План верификации ПО.

[4] СКУ. Руководство пользователя программного продукта. Инструкция по применению.

## 1.4. Термины, определения и соглашения

Таблица 1. Термины, определения и соглашения

|  |  |
| --- | --- |
| Термин | Определение, толкование |
| Сообщение о проблеме | Документ процесса УК ПО, содержащий описание несоответствия в данных ЖЦ ПО или в описании процессов ЖЦ ПО. |
| Запрос на изменение | Документ процесса УК ПО, созданный для внесения изменений в данные ЖЦ ПО с целью устранения несоответствия, описанного в СП. |
| Артефакт | Атомарное данное ЖЦ ПО. Артефактом может быть, например, требование или файл исходного кода |
| Агрегация | Составное данное ЖЦ ПО, которое включает в себя однотипные артефакты и/или агрегации. |
| Конфигурация | Составное данное ЖЦ ПО, которое может включать в себя любые артефакты, агрегации и конфигурации |
| Категория запроса на изменение | Атрибут запроса на изменение, идентифицирующий процесс ЖЦ ПО, данные которого будут меняться в рамках этого ЗИ |
| Формальная инспекция | Способ верификации документов, основанный на экспертной оценке их правильности, выполняемой одним или несколькими инспекторами, как правило, с использованием проверочных перечней. |
| Единица конфигурации | Совокупность данных ЖЦ ПО, которая в целях УК ПО рассматривается как единое целое. |
| Тип единицы конфигурации | Тип документов ЖЦ ПО, соответствует разделам 11.1 – 11.20 КТ-178В |
| Базовая версия документа | Утверждённая зарегистрированная конфигурация одной или более единиц конфигурации, которая в дальнейшем служит основой для последующей разработки и изменяется только в соответствии с процедурами управления изменениями. |
| Проблема | Тип сообщений в системе GitHub «Issue» с возможностью присвоения тегов, связки с версиями данных ЖЦ ПО, назначения ответственных разработчиков и комментирования изменений. |

Таблица 2. Сокращения

|  |  |
| --- | --- |
| Термин | Определение, толкование |
| ГУИ | Группа управления изменениями |
| ЕК | Единица конфигурации |
| ЖЦ | Жизненный цикл |
| ЗИ | Запрос на изменение |
| МИ | Менеджер изменений |
| ПО | Программное обеспечение |
| СП | Сообщение о проблеме |
| СПР | Система поддержки разработки |
| УК | Управление конфигурацией |
| ФИ | Формальная инспекция |

# 2. Среда УК ПО

## 2.1. Инструменты УК ПО

В качестве системы конфигурационного управления в проекте используется программное обеспечение управлениями проектами GitHub. Она служит инструментальным средством поддержки процесса УК ПО, обеспечивая регистрацию проблем, управление изменениями, учет состояния конфигурации и контроль за состоянием среды ЖЦ ПО. Хранение данных ЖЦ ПО, идентификация конфигурации и трассировка, осуществляется с использованием системы контроля версий GitHub.

Таблица 3 содержит перечень инструментальных средств, которые будут использоваться для поддержки процесса УК ПО.

Таблица 3. Инструментальные средства процесса УК ПО

|  |  |
| --- | --- |
| Инструментальное средство | Предназначение |
| GitHub | Система контроля версий и управления проектами |

Процедуры работы с GitHub и порядок ее настройки описаны в настоящем документе в п.3.3.

При начале работы с исходным кодом Автор помечает соответствующую проблему (ЗИ) тегом "In Progress". После завершения работы с кодом и системой GitHub Автор закрывает соответствующую запись о проблеме, предварительно пометив её тегом "Released".

## 2.2. Организация и ответственности

Все участники процессов ЖЦ ПО так или иначе вовлечены в процесс УК ПО.

В рамках процесса УК ПО каждый участник проекта может иметь одну или несколько ролей, дающих определенные полномочия в GitHub. Роли в процессе УК ПО назначаются в зависимости от ролей участника проекта в других процессах ЖЦ ПО и описаны ниже:

* *Автор*. Имеет полномочия на создание новых ЕК, но не имеет полномочий на проведение ФИ собственных задач. Роль назначается любым участникам проекта для выполнения конкретной задачи, описанной в ЗИ.
* *Инспектор*. Имеет полномочия создавать и изменять записи о ФИ, включая замечания и ответы на вопросы проверочных перечней. Замечания и ответы представлены в виде комментариев к соответствующей проблеме GitHub.
* *Менеджер изменений*. Имеет полномочия проводить анализ СП и создавать, корректировать ЗИ.
* *Инженер по качеству*. Имеет полномочия проводить аудиты и создавать соответствующие записи в GitHub.
* *Посетитель*. Имеет полномочия просматривать данные ЖЦ ПО, но не редактировать их.

Участник проекта с любой ролью имеет полномочия на создание СП.

Каждый процесс ЖЦ ПО взаимодействует с процессом УК ПО при любом обращении к данным ЖЦ ПО. Это отражено в планах разработки ПО [2], верификации ПО [3] и УК ПО посредством ссылок на выполняемые процедуры и мероприятия УК ПО и явное определение ответственного. Так определяется ответственность каждого из участников процессов ЖЦ ПО за выполнение соответствующих процедур и мероприятий УК ПО.

## 2.3. Взаимодействие УК ПО с другими процессами ЖЦ ПО

Все процессы разработки ПО и интегральные процессы взаимодействуют посредством процесса УК ПО, таким образом, все данные, создающиеся или модифицирующиеся в ходе выполнения процессов разработки ПО и интегральных процессов, являются входными данными для процесса УК ПО, все процессы разработки ПО и интегральные процессы получают входные данные из процесса УК ПО.

Управление конфигурацией требований к системе осуществляется с помощью GitHub.

# 3. Мероприятия процесса УК ПО

## 3.1. Идентификация конфигурации

Цель мероприятия по идентификации конфигурации – однозначно определить каждую единицу конфигурации (ЕК), включая ее последовательные версии для того, чтобы обеспечить основу для управления единицами конфигурации и возможность ссылаться на них.

Различается четыре вида единиц конфигурации: артефакт, агрегация, конфигурация и запись. При создании агрегаций и конфигураций в системе образуются иерархические связи между ЕК, при этом каждая ЕК имеет атрибут «Имя», использующийся для удобочитаемости. Эти имена могут быть и неуникальными, так как идентификация осуществляется только через идентификатор.

В рамках проекта определяются три конфигурации, используемые для организации процессов ЖЦ разрабатываемого ПО: каталог планов и стандартов, каталог среды ЖЦ ПО и каталог комплектации ПО. Элементы каталогов представляют собой ссылки на конкретные версии ЕК, (логически) входящих в них.

Каталог планов и стандартов идентифицирует актуальные планы и стандарты (т.е. содержит ссылки на них).

Каталог среды жизненного цикла (раздел 11.15 КТ-178B [1]) идентифицирует используемые инструменты разработки и верификации, описания их использования и, возможно, другие данные, необходимые для использования инструментов (например, конфигурационные файлы, шаблоны и т.п.).

Каталог комплектации ПО определяет конфигурацию разрабатываемого программного продукта (раздел 11.16 КТ-178B [1]). Каталог комплектации ПО создается в начале работ по проекту и заполняется по мере появления и обновления данных ЖЦ. В ходе разработки ПО каталог комплектации ПО содержит ссылки на актуальные данные ЖЦ. Указание актуальных планов, стандартов и данных среды ЖЦ ПО осуществляется путем включения в каталог комплектации ПО соответствующих версий каталога планов и стандартов и каталога среды ЖЦ ПО.

В формате записей в системе представляются сообщения о проблемах, запросы на изменение и результаты верификации (записи ФИ).

ПРИЛОЖЕНИЕ A содержит перечень данных ЖЦ ПО.

## 3.2. Базовые версии и трассируемость

Цель введения базовых версий – заложить основу для дальнейших мероприятий жизненного цикла ПО, дать возможность ссылаться на эти версии, контролировать их и обеспечить трассируемость между единицами конфигурации. Базовые версии устанавливаются для следующих ЕК: артефактов, агрегаций и конфигураций, имеющих категорию контроля 1 (см. ПРИЛОЖЕНИЕ A).

Любая ЕК в рамках данного проекта может находиться в одном из двух состояний: “In Progress” или “Released”. Установление базовой версии ЕК сопровождается ее переводом в состояние “Released” в ходе процесса формальной инспекции, описанного в Плане верификации [4]. Эволюция от предыдущей к последующей базовой версии ЕК происходит в рамках процесса управления изменениями, описанного в разделе 3.3 данного документа.

Базовая версия агрегации может состоять только из базовых версий агрегаций и артефактов, входящих в нее.

Конфигурация может состоять из данных, отнесенных к обеим категориям контроля. Если в конфигурацию входят данные, отнесенные к категории контроля 1, то в базовую версию конфигурации могут входить только базовые версии таких данных.

### 3.2.1. Каталог планов и стандартов

Установка базовой версии каталога планов и стандартов производится по результатам ФИ, проводимой для проверки согласованности планов и стандартов. Наличие базовых версий отдельных планов или стандартов само по себе не является указанием для разработчиков о том, что именно эти версии являются актуальными и должны использоваться разработчиками ПО (они могут оказаться несогласованными между собой).

### 3.2.2. Каталог среды жизненного цикла ПО

Установка базовой версии каталога среды ЖЦ ПО производится по результатам ФИ, проводимой для проверки исполнения ЗИ, на основании которого выполнено изменение каталога среды ЖЦ ПО. Указание актуальной версии среды ЖЦ ПО производится путем включения в каталог комплектации ПО соответствующей версии каталога среды ЖЦ ПО.

### 3.2.3. Каталог комплектации ПО

Каталог определяет актуальную конфигурацию разрабатываемого ПО и включает ссылки на все данные ЖЦ, используемые при его разработке. Планы, стандарты и инструменты разработки включаются в каталог комплектации ПО путем указания ссылок на соответствующие (актуальные) версии каталога планов и стандартов и каталога среды ЖЦ ПО.

Установка базовой версии каталога комплектации ПО производится по результатам ФИ, проводимой для проверки исполнения ЗИ, на основании которого выполнено изменение каталога комплектации ПО. ФИ, проводимая при установлении базовой версии каталога комплектации ПО, отвечает целям рассмотрения соответствия ПО требованиям (раздел 8.3 КТ-178B [1]).

### 3.2.4. Трассируемость

Трассируемость требований к ПО высокого уровня на системные требования осуществляется с помощью установления связи типа «порожден из» между соответствующими единицами конфигурации.

Трассируемость описания проекта ПО и требований к ПО низкого уровня на требования к ПО высокого уровня, исходного кода ПО на описание проекта ПО и требования к ПО, тестовых примеров и процедур на проект ПО и требования к ПО осуществляется с помощью установления связи типа «порожден из» между соответствующими единицами конфигурации.

Трассируемость между предыдущей и последующей версиями единицы конфигурации устанавливается с помощью запросов на изменения (для ЕК 1-й категории контроля данных) и истории изменений, сохраняющейся для каждой единицы конфигурации.

Трассируемость между базовыми версиями единицы конфигурации устанавливается в рамках процесса управления изменениями, описанного в разделе 3.3 данного документа.

Трассируемость между причинами изменений и изменениями устанавливается с помощью связи сообщений о проблемах и запросов на изменения.

Трассируемость единиц конфигурации и процессов, которые их определяют, осуществляется путем выбора соответствующих типов, правильного размещения единиц конфигурации в соответствии с ПРИЛОЖЕНИЕ A и использованием запросов на изменения соответствующих категорий (см. раздел 3.3.2).

## 3.3. Регистрация проблем и управление изменениями

В данном разделе рассмотрена общая схема управления изменениями, реализованная в системе GitHub. Документации по использованию системы GitHub не полностью охватывает ЖЦ ПО с момента начала разработки и заканчивая сертификацией, поэтому при работе в рамках проекта будут использоваться лишь некоторые положения раздела.

### 3.3.1. Регистрация проблем

Любой участник проекта, обнаруживший несоответствие в данных ЖЦ ПО, несоответствие выполняемых процессов процессам, описанным в планах, недостатки результатов процесса, ненормальную работу ПО и т.п., обязан зафиксировать это несоответствие в виде сообщения о проблеме. При этом он должен внести следующую информацию:

* Заголовок сообщения о проблеме – короткий текст, описывающий суть проблемы;
* Описание проблемы – подробное описание несоответствия;
* Список единиц конфигурации и их версий, в которых несоответствие было обнаружено.

В случае, если предлагается улучшение процесса, в списке единиц конфигурации указываются планы и стандарты, описывающие рассматриваемый процесс.

Любое предполагаемое изменение в данных ЖЦ ПО, в том числе и не связанное с устранением проблем (например, добавление новой функциональности), также обрабатывается через сообщение о проблеме. В этом случае в описание проблемы заносится информация о предлагаемом изменении, а список единиц конфигурации, в которых обнаружено несоответствие, может быть не заполнен.

Если участник проекта не обладает достаточными знаниями для точной и полной идентификации единиц конфигурации и их версий, в которых несоответствие было обнаружено, то список таких ЕК может быть заполнен не полностью.

После того, как вся необходимая информация о проблеме или предлагаемом изменении заполнена, сотрудник помечает сообщение о проблеме тегом "Scheduled".

В документации по использованию системы GitHub взаимодействие с системным процессом и изменение требований к системе не предусмотрено. В связи с этим сообщения о проблемах, обнаруженных в системных требованиях, в рамках данного проекта не рассматриваются.

### 3.3.2. Рассмотрение изменений

После создания любое сообщение о проблеме проходит процедуру рассмотрения группой управления изменениями, возглавляемой менеджером изменений. ГУИ формируется с таким расчетом, чтобы в нее входили сотрудники, хорошо разбирающиеся в предметной области и данных, в которых выявлена проблема, и являющиеся участниками процессов ЖЦ ПО, данные которых могут быть затронуты во время устранения проблемы. В порядке исключения МИ может производить единоличные рассмотрения СП, если указанные в них проблемы достаточно просты и МИ обладает достаточными знаниями для корректной оценки проблемы.

Рассмотрения СП группой управления изменениями происходят с регулярностью, определяемой руководителем проекта. Перед собранием ГУИ менеджер изменений выявляет сообщения о проблемах, помеченные тегом «Scheduled», переводит их в состояние «Analysis». ГУИ проводит рассмотрение проблем, находящихся в состоянии «Analysis».

Для каждого рассматриваемого сообщения о проблеме ГУИ определяет необходимые действия, при этом МИ фиксирует принимаемые решения в системе GitHub:

- Если проблема признана актуальной, МИ может дополнить описание проблемы или внести дополнительные данные с указанием версий в список единиц конфигурации, в которых обнаружена проблема. После этого ГУИ анализирует, какие изменения нужно произвести, чтобы исправить проблему, и МИ создает необходимое количество запросов на изменения соответствующих категорий. В системе GitHub запросы на изменения помечаются следующими дополнительными тегами:

* «Process» – запросы на изменения данной категории используются для создания и внесения изменений в планы и стандарты.
* «Requirements» – запросы на изменения данной категории используются для создания и внесения изменений в требования к ПО.
* «Project» – запросы на изменения данной категории используются для создания и внесения изменений в описание проекта ПО.
* «Code» – запросы на изменения данной категории используются для создания и внесения изменений в исходный код.
* «Integration» – запросы на изменения данной категории используются для исполняемого объектного кода
* «Test» – запросы на изменения данной категории используются для создания и внесения изменений в тестовые примеры и процедуры.
* «Catalog» – запросы на изменения данной категории используются для создания и внесения изменений в каталог комплектации ПО и каталог среды ЖЦ ПО.

Категория запроса на изменение определяет процесс ЖЦ ПО, выходные данные которого будут изменены для устранения проблемы. Таким образом, при создании запросов на изменения МИ определяет процессы ЖЦ, которые нужно выполнить или повторить для устранения проблемы.

В каждом отдельном запросе на изменение МИ указывает данные ЖЦ, подлежащие изменению, и заполняет информацию о предполагаемом изменении, при этом детализация должна быть достаточной для того, чтобы автор мог корректно реализовать изменения без анализа сообщения о проблеме.

- Если необходимо дополнительное исследование или по каким-то причинам анализ проблемы должен быть отложен на более поздний период, то такие сообщения о проблемах будут повторно рассматриваться на следующих собраниях ГУИ, пока не будут помечены тегами «ToDo» или «Archive» .

- перевести сообщение о проблеме в состояние Archive, если проблема признана неактуальной или эта проблема уже решена. МИ заполняет комментарий, в котором описывается причина отклонения сообщения о проблеме, и закрывает проблему функцией “Close issue”

После того, как сообщение о проблеме переведено в состояние «ToDo», соответствующими процессами проводятся мероприятия по внесению изменений с целью устранения проблемы в рамках запросов на изменения, созданных для этого сообщения о проблеме.

### 3.3.3. Внесение изменений

Любое изменение единицы конфигурации приводит к изменению ее идентификации – появлению в системе новой версии данной единицы конфигурации.

Сотрудник, которому поручено внесение изменений, переводит запрос на изменение в состояние «ToDo», тем самым сигнализируя, что работа по изменению единиц конфигурации начата. Сотрудник создает новые версии единиц конфигурации в рамках запроса на изменение, если единицы конфигурации отнесены к категории контроля 1, при этом в качестве исходных версий для изменений могут служить только последние базовые версии.

После того, как сотрудник внес все необходимые изменения в рамках данного запроса на изменение, он создает в СПР GitHub комментарий о формальной инспекции. По результатам ФИ устанавливаются новые базовые версии единиц конфигурации (см. План разработки ПО [2] и План верификации ПО [3]). После того, как все новые базовые версии установлены, при закрытии записи о формальной инспекции запрос на изменение помечается тегом «Released» .

В случае если сотрудник, выполняющий изменения по некоторому ЗИ, обнаруживает невозможность его завершения, он создает СП с указанием причин этого. При появлении новых СП, относящихся к уже изменяемому объекту и/или уже выполняемому ЗИ, МИ может назначить повторное рассмотрение исходного СП (т.е. СП, по которому был создан выполняемый ЗИ). В результате этого рассмотрения выполняемый ЗИ может быть изменен или же могут быть созданы дополнительные ЗИ для данного СП. Такая ситуация возможна, если, например, МИ определил, что проблема содержится в исходном коде, но при изменении исходного кода выяснилось, что нужно также поменять архитектуру ПО. В таком случае МИ создает запрос на описания проекта ПО, работа над исходным кодом будет остановлена, а процессы ЖЦ ПО будут повторены начиная от разработки описания проекта ПО.

В случае если во время работы над изменением принято решение отказаться от внесения изменения, МИ переводит запрос на изменение из состояния «ToDo» в состояние Archive, занося в комментарий причину перевода. При этом все изменения, сделанные в рамках данного запроса на изменение, помечаются в системе системой GitHub как некорректные, МИ откатывает измененные единицы конфигурации к последним базовым версиям, с которых начиналось изменение в рамках данного ЗИ. Таким образом, работа по другому ЗИ начнется с изменения базовых версий. Промежуточные версии, помеченные как некорректные, будут доступны только для просмотра.

Данные ЖЦ категории контроля 2 (см ПРИЛОЖЕНИЕ A) могут быть изменены без создания запроса на изменение и проведения формальной инспекции.

### 3.3.4. Закрытие сообщения о проблеме

Ежедневно МИ проверяет, для каких СП мероприятия по внесению изменений завершены (все запросы на изменения, созданные для данного СП, переведены в состояние «Archive»), МИ переводит такие сообщения о проблемах в состояние «Released».

### 3.3.5. Жизненный цикл сообщения о проблеме

Рисунок 1 показывает жизненный цикл сообщения о проблеме.

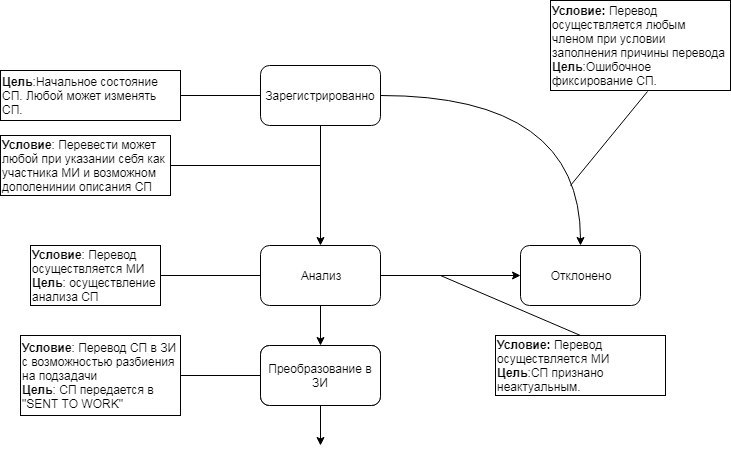


Рисунок 1. Жизненный цикл сообщения о проблеме.

### 3.3.6. Жизненный цикл запроса на изменение

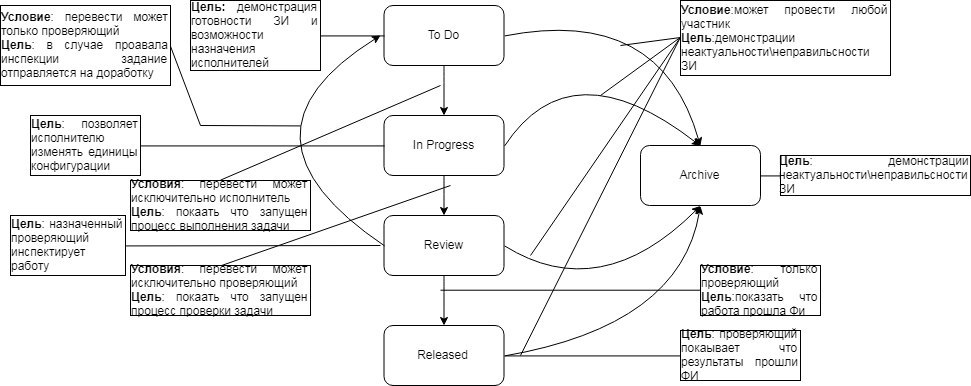
Рисунок 2 показывает жизненный цикл запроса на изменение.****

Рисунок 2. Жизненный цикл запроса на изменение

### 3.3.7. Регистрация проблем и управление изменениями после сертификации

Регистрация проблем и управление изменениями после сертификации в рамках документации по использованию системы GitHub не рассматривается.

## 3.4. Учет состояния конфигурации

Цель мероприятий учета состояния конфигурации заключается в том, чтобы предоставить данные для управления конфигурацией в процессах жизненного цикла ПО, принимая во внимание идентификацию конфигурации, базовые версии, сообщения о проблемах и управление изменениями.

Учет состояния конфигурации обеспечивается возможностями системы GitHub. Все действия над единицами конфигурации в этой системе протоколируются. Участники проекта в любой момент времени могут получить следующую информацию:

* Дата и время создания очередной версии единицы конфигурации
* Идентификатор пользователя, создавшего очередную версию единицы конфигурации
* Текущее состояние единицы конфигурации (см. разделы 3.2, 3.3.5, 3.3.6)

## 3.5. Архивирование, воспроизведение и выпуск

Архивирование (резервное копирование) данных ЖЦ ПО, хранящихся в системе GitHub, предусматривается документацией по использованию системы GitHub.

Данные ЖЦ ПО документации по использованию системы GitHub, хранящиеся под управлением системы GitHub, являются оригиналами и должны воспроизводиться исключительно из системы GitHub.

Мероприятия по выпуску ПО предусматривают создание базовой версии каталога комплектации ПО.

## 3.6. Контроль среды ЖЦ ПО

### 3.6.1. Контроль планов и стандартов

Управление планами и стандартами проекта осуществляется в соответствии с данным Планом. Сведения о действующих версиях планов и стандартов проекта включаются в каталог комплектации ПО путем задания в нем актуальной базовой версии каталога планов и стандартов.

### 3.6.2. Контроль инструментальных средств

Каталог среды ЖЦ ПО содержит перечень инструментальных средств, используемых для разработки, производства, верификации и загрузки ПО. Данные инструментальные средства будут помещены под конфигурационное управление (см. ПРИЛОЖЕНИЕ A) и будут иметь категорию контроля 2. Управление этими средствами осуществляется инженером, назначенным руководителем проекта и имеющим роль Автора. Сведения об используемых версиях инструментальных средств включаются в каталог комплектации ПО путем задания в нем актуальной базовой версии каталога среды ЖЦ.

## 3.7. Контроль данных (документов) ЖЦ ПО

Таблица 4 определяет соответствие процедур УК ПО, реализующих цели процесса УК ПО согласно КТ-178B для данных ЖЦ ПО, относящихся к 1-й и 2-й категориям контроля данных.

Таблица 4. Цели и процедуры процесса УК ПО.

|  |  |  |
| --- | --- | --- |
| Цель процесса | Ссылка на раздел КТ-178B | Ссылка на раздел данного документа, описывающий соответствующие процедуры |
| Идентификация конфигурации | 7.2.1 a-d | 3.1 |
| Базовые версии | 7.2.2 a | 3.2.1 - 3.2.4 |
| 7.2.2 b | 3.2.4 |
| 7.2.2 c | 3.2.1 - 3.2.4. Все данные ЖЦ находятся под контролем системы GitHub, в том числе и библиотеки. |
| 7.2.2 d | 3.2.1 - 3.2.4, 3.3.6 |
| 7.2.2 e | 3.2.5, 3.3.6 |
| Трассируемость | 7.2.2 f, g | 3.2.5, 3.3.6 |
| Сообщения о проблемах | 7.2.3 | 3.3.1-3.3.5 |
| Управление изменениями – целостность и идентификация | 7.2.4 a, b | Обеспечивается применением системы GitHub. |
| Управление изменениями - отслеживание | 7.2.4 c, d, e | 3.3.2-3.3.4, 3.3.6 |
| Рассмотрение изменений | 7.2.5 | 3.3.2 |
| Учет состояния конфигурации | 7.2.6 | 3.4 |
| Воспроизведение | 7.2.7 a | 3.5.3 |
| Защита от несанкционированных изменений | 7.2.7 b(1) | 3.5 |
| Выбор, обновление, воспроизведение носителей | 7.2.7 b(2), (3), (4), c | 3.5.4 |
| Выпуск | 7.2.7 d | 3.5.5 |
| Хранение документов | 7.2.7 e | Процедуры управления документами устанавливаются настоящим планом УК ПО. |

# 4. Критерии перехода

Процесс УК ПО начинается одновременно с процессом планирования создания ПО и заканчивается с выпуском последней версии ПО в рамках проекта.

# 5. Документы УК ПО

Документами процесса УК ПО являются:

* Сообщения о проблемах
* Запросы на изменения
* Каталог комплектации ПО
* Каталог среды ЖЦ ПО
* Каталог планов и стандартов ПО

Подробно все документы УК ПО описаны в ПРИЛОЖЕНИЕ A.

# Приложение А. Данные ЖЦ ПО

Таблица 5 содержит перечень данных ЖЦ ПО. В столбце «Вид» указан вид и идентификатор ЕК соответствующих данных ЖЦ ПО. GitHub позволяет создать иерархическую структуру для хранения данных ЖЦ ПО, в столбце «Место в иерархии» корнем является корень репозитория GitHub. Атомарные ЕК объединяются в агрегацию верхнего уровня, имя которой приведено в столбце «Место в иерархии», при этом структура хранения может диктовать необходимость использования агрегаций промежуточных уровней. В столбце «Вид» используются следующие сокращения: СП – сообщение о проблеме, ЗИ – запрос на изменение, ФИ – формальная инспекция, Агр – агрегация, К – конфигурация.

Таблица 5. Данные ЖЦ ПО уровня критичности B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Данные ЖЦ ПО | Место в иерархии | Процесс - источник данных | Вид | Катего­рия контроля |
| План управления конфигурацией ПО | /План УК ПО | Планирование | Агр | 1 |
| План разработки ПО | /План разработки | Планирование | Агр | 1 |
| План верификации ПО | /План верификации | Планирование | Агр | 1 |
| Стандарт на разработку требований к ПО | /Стандарт на разработку требований к ПО | Планирование | Агр | 1 |
| Стандарт на проектирование ПО | /Стандарт на проектирование ПО | Планирование | Агр | 1 |
| Каталог планов и стандартов | /Каталог планов и стандартов | УК ПО | К | 1 |
| Каталог среды ЖЦ ПО | /Каталог среды ЖЦ ПО | УК ПО | К | 1 |
| Требования к системе | /Требования к системе | Разработка | Агр | 2 |
| Требования к ПО | /Системные требования | Разработка | Агр | 1 |
| Инструментальные средства ЖЦ ПО | /Тестовое окружение | Планирование | Агр | 2 |
| Описание проекта ПО | /Проект ПО | Разработка | Агр | 1 |
| Исходный код | /Code | Разработка | Агр | 1 |
| Ключи компиляции, редактирования связей и загрузки | /Code/build\_params | Интеграция | Агр | 1 |
| Исполняемый код | /Code/build | Интеграция | Агр | 1 |
| Протоколы сборки и загрузки | /Code/build\_logs | Интеграция | Агр | 1 |
| Тестовые примеры | /Тесты | Верификация | Агр | 1 |
| Результаты испытаний | /Результаты тестов | Верификация | Агр | 2 |
| Отчеты об анализе структурного покрытия | /Результаты тестов /Структурное покрытие | Верификация | Агр | 2 |
| Сообщения о проблемах | Issues | УК ПО | СП | 2 |
| Запросы на изменения | Issues | УК ПО | ЗИ | 2 |
| Записи о ФИ | Issues | Верификация | ФИ | 2 |