Введение

Цель

Данный документ определяет мероприятия, которые будут выполняться в рамках проекта "Стилист" для достижения целей процесса управления конфигурацией ПО, определенных в документе КТ-178С [1].

Область применения

Требования и положения настоящего плана распространяется на проект "Стилист" и обязателен для всех участников команды.

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. Стандарт на кодирование ПО.
5. Github Documentation. — Текст : электронный // Github : [сайт]. — URL: https://docs.github.com/ru
6. Git Manual. — Текст : электронный // Git : [сайт]. — URL: https://git-scm.com/docs/user-manual

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.

Термин	Определение, толкование
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.
СПР	Система поддержки разработки программного обеспечения.

Таблица 2. Сокращения

Термин	Сокращение
ГК	Гарантия качества
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

Среда УК ПО

Инструменты УК ПО

В качестве системы конфигурационного управления используется среда поддержки разработки программного обеспечения Github. Она служит инструментальным средством поддержки процесса УК ПО, обеспечивая хранение данных ЖЦ ПО, идентификацию конфигурации, трассировку, регистрацию проблем, управление изменениями, учет состояния конфигурации и контроль за состоянием среды ЖЦ ПО.

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

Таблица 3. Инструментальные средства процесса УК ПО

Инструментальное средство	Предназначение
Git	Система управления версиями
Github	Система конфигурационного управления

Процедуры работы с СПР Github и порядок ее настройки описаны в Документации Github[5]. Пользовательское руководство для работы с Git приведено в ссылке [6].

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 СПР 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 и изменение существующих ЕК. Назначается инженерам, создающим и изменяющим данные ЖЦ ПО в рамках процессов планирования, разработки и верификации (за исключением СП и записей о ФИ).
- Инспектор. Имеет полномочия создавать и изменять записи о ФИ, включая замечания и ответы на вопросы проверочных перечней. Назначается инженерам, имеющим роли Ведущего и Инспектора в процесе верификации ПО (см. [3]).

- Менеджер изменений. Имеет полномочия проводить анализ СП и создавать ЗИ. Назначается инженерам, ответственным за проведение процедуры рассмотрения изменений в рамках процесса УК ПО (см. **Рассмотрение изменений**).
- Инженер по качеству. Имеет полномочия проводить аудиты и создавать соответствующие записи согласно Плану ГК ПО. В рамках данной работы мероприятия ГК ПО не предусмотрены.
- Посетитель. Имеет полномочия просматривать данные ЖЦ ПО, но не редактировать их.

Участник проекта с любой ролью имеет полномочия на создание СП.
Также в системе СПР предусмотрена вспомогательная роль Администратор, назначаемая инженеру, ответственному за выдачу ролей, настройку пользовательских типов объектов, внесение утвержденных проверочных перечней и аналогичные задачи по настройке системы СПР.
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.

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 СПР Github.

Мероприятия процесса УК ПО

Идентификация конфигурации

Цель мероприятия по идентификации конфигурации – однозначно определить каждую единицу конфигурации (и её последовательные версии) так, чтобы обеспечить основу для управления единицами конфигурации и ссылки на них.

В данном разделе описывается подход к процессу идентификации следующих элементов конфигурации:

- документов;
- сообщений о проблемах;
- запросов на изменения;
- программного кода.

Идентификация документа

Версии документов идентифицируются по их названию. Название имеет составную структуру: имя, отражающее суть документа, нижнее подчеркивание, дата изменения в формате год, месяц, день, час, минута. Итоговый формат названия можно описать в виде следующего шаблона: <Name>_<YYYYMMdd_hhmm>. Пример: План_20220930_1830.

Уникальность идентификации обеспечивается его связью с временем изменения.

Идентификация СП

СП регистрируются в issue соответствующего репозитория проекта на Github [5].

СП идентифицируются уникальным номером. Первому СП проставляется номер 1, каждому последующему на единицу больше. Данная функциональность обеспечивается сервисом Github[5].

Идентификация запроса на изменения

На базе СП могут быть созданы ЗИ, которые регистрируются в соответствующем репозитории проекта в качестве Pull Request на Github[5].

Идентификация запросов обеспечивается присваиванием уникального номера, аналогично идентификации СП (см. описание инструментария).

Идентификация программного кода

Идентификация программного кода осуществляется при помощи системы управления версиями Git[6]. Каждой новой версии программного кода присваивается уникальное имя (уникальность гарантируется hash-функцией, реализуемой системой Git).

Базовая версия и трасируемость

Определения

Базовая версия (Baseline) – 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.

Трассируемость (Traceability) – Доказательство связи между определенными единицами, например, между результатами процессов, между результатом и порождающим его процессом, а также между требованием и его реализацией.

Базовые версии

Для каждого элемента конфигурации устанавливается базовая версия, которые должны быть защищены от изменений средствами, описаными в "управлении изменениями".

Трассируемость

Должна быть обеспечена трассируемость базовой версии/единицы конфигурации к той базовой версии/единице конфигурации, из которой она была получена.

Данное свойство документов обеспечивается стандратом идентификации, а именно временем изменения, зафиксированного в названии.

Трассируемость бизнес-требований, требований к ПО высокого и низкого уровней обеспечивается при помощи способа нумерации требований, описанных в стандарте на разработку требований.

Трассируемость между СП и требованиями к ПО обеспечивается присваиванием СП того же номера, что и у требования ПО.

Трассируемость между ЗИ и СП реализуется средствами Github[5]: возможность просмотра всех ЗИ, относящихся к данному СП.

Трассируемость между ЗИ и изменением в программном коде реализуется средствами Github[5]: возможностью просмотра всех изменений, относящихся к данному ЗИ.

Трассируемость между предыдущей и последующей версиями единицы конфигурации устанавливается с помощью запросов на изменения и истории изменений, сохраняющейся в системе СПР Github для каждой единицы конфигурации.

Регистрация проблем и управление изменениями

В данном разделе рассмотрена общая схема управления изменениями, реализованная в СПР Github[5].

Регистрация проблем

Любой участник проекта, обнаруживший несоответствие требований стандарту или описанию проекта, несоответствие программного кода стандартам или требованиям и 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 с обязательным указанием ссылок на соответсвующие стандарты или требования;
- Список единиц конфигурации и их версий, в которых несоответствие было обнаружено.

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.

Рассмотрение изменений

СП должно быть рассмотрено группой управления изменениями, возглавляемой менеджером изменений. В процессе рассмотрения СП принимается решение, нужно ли данную проблему решать. Если проблема признана актульаной, то описание проблемы может быть дополнено МИ.

Далее производится анализ, какие изменение необходимо провести, чтобы решить проблему, и МИ создает нужно количество ЗИ с привязкой к СП. В каждом запросе на изменении МИ указывает данные ЖЦ, подлежающие изменению, и заполняет детально информацию о предполагаемом изменении, так чтобы автор смог понять необходимые изменения без анализа описания сообщения о проблеме. ЗИ передается в работу автору.

СП может потребовать дальнейшего анализа. Такое сообщение ГУИ откладывается на дальнейшее рассмотрение.

Также СП может быть отклонено, если проблема неактуальна или уже решена с объяснением причины отклонения сообщения о проблеме.

Внесение изменений

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

Автор рассматривает порученный ему запрос на изменение и приступает к изменению единиц конфигурации. В качестве исходных версий для изменений служат только последние базовые версии.

После внесения изменений автор должен создать запись о формальной инспекция. По результатам ФИ устанавливаются новые базовые версии ЕК (см План разработки[2] и План верификации[3]). Ввиду того, что аудит ГК ПО не предусмотрен, ЗИ считается завершенным.

Могут быть случаи, когда невозможно завершить запрос на изменение. Тогда автор создает новое СП с объяснением причины.

Закрытие сообщений о проблеме

СПР Github позволяет автоматически закрывать СП, когда все ЗИ были завершены. Таким образом вмешательство ГУИ не нужно.

Регистрация проблем и управление изменениями после сертификации

Регистрация проблем и управление изменениями после сертификации в рамках данной работы не рассматривается.

Контроль среды ЖЦ ПО

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.

Критерий перехода

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