Содержание

Введение....2
1 Организационная структура предприятия....3
2 Основные виды деятельности предприятия....4
3 Порядок ведения технической документации....6
4 Порядок осуществления интеграции программных модулей на предприятии....8
Требования к ПО. Классификация требований....8
Классификация требований к ПО....8
Валидация требований....9
Использование функциональных прототипов....10
Разработка и реализация требований....10
Верификация требований к ПО....11
5 Порядок ревьюирования программных продуктов на предприятии....13
Системы контроля версий....13
Copy-paste....13
Локальная система контроля версий....13
Централизованная система контроля версий....14
Распределенная система контроля версий....15
Распределенная система контроля версий....15
Заключение....16
Список используемой литературы....17

Введение

С 18 мая по 21 июня 2023 года я проходил учебную и производственную практику на АО «ФНПЦ «ПО «Старт» им. М.В. Проценко», г. Заречный.

Целью данной практики являлось:

- закрепление теоретических знаний, полученных в ходе изучения профессиональных модулей ПМ.02 «Осуществление интеграции программных модулей», ПМ.03 «Ревьюирование программных продуктов»;
- формирование у обучающихся профессиональных компетенций в условиях реального производства.

Задачи практики:

- закрепление и совершенствование приобретенного в процессе обучения опыта практической деятельности студентов в сфере изучаемой специальности;
- освоение современных производственных процессов, технологий;
- адаптация обучающихся к конкретным условиям деятельности предприятий различных организационно-правовых форм.

В ходе написания отчета были использованы учредительные, регистрационные документы, предоставленная отчетность, учебные и методические пособия.

Введение

С 18 мая по 21 июня 2023 года я проходил учебную и производственную практику на АО «ФНПЦ «ПО «Старт» им. М.В. Проценко», г. Заречный.

Целью данной практики являлось:

- закрепление теоретических знаний, полученных в ходе изучения профессиональных модулей ПМ.02 «Осуществление интеграции программных модулей», ПМ.03 «Ревьюирование программных продуктов»;
- формирование у обучающихся профессиональных компетенций в условиях реального производства.

Задачи практики:

- закрепление и совершенствование приобретенного в процессе обучения опыта практической деятельности студентов в сфере изучаемой специальности;
- освоение современных производственных процессов, технологий;
- адаптация обучающихся к конкретным условиям деятельности предприятий различных организационно-правовых форм.

В ходе написания отчета были использованы учредительные, регистрационные документы, предоставленная отчетность, учебные и методические пособия.

1 Организационная структура предприятия.

Акционерное Общество «Федеральный научно-производственный центр «Производственное объединение «Старт» имени М.В.Проценко» является современным предприятием, обладающим уникальными технологическими возможностями, разрабатывает и выпускает конкурентоспособную наукоемкую высокотехнологичную продукцию, используемую как в сфере обеспечения национальной безопасности страны, так и для нужд атомной энергетики, предприятий топливно-энергетического комплекса, железнодорожного транспорта и прочих промышленных объектов. Как непосредственный участник в деле создания ядерного щита нашей страны, «Старт» более пятидесяти лет работает в сфере ядерно-оружейного комплекса, входящего в состав предприятий Госкорпорации «Росатом», специализируется на выпуске радиотехнических, электромеханических, электронных приборов и систем.

Дата образования предприятия: 20 июля 1954 года Совет Министров СССР принял Постановление о строительстве приборного завода №1134 (ныне ПО «Старт»).

Генеральный директор – Байдаров Сергей Юрьевич.

2 Основные виды деятельности предприятия.

Основными направлениями деятельности АО «ФНПЦ «ПО «Старт» им. М.В.Проценко» являются:

- изготовление спецтехники в интересах Госкорпорации «Росатом»;
- системы безопасности и технические средства охраны;
- АСУ ТП и различное оборудование для атомной энергетики;
- АСУ ТП и различное оборудование для топливно-энергетического комплекса;
- продукция для железнодорожного транспорта;
- металлообрабатывающее оборудование;
- система обеспечения инструментом;
- твердосплавный режущий инструмент;
- нормализованные изделия.

Ключевыми технологиями производства АО «ФНПЦ «ПО «Старт» им. М.В.Проценко» являются:

- литье с использованием технологии послойного синтеза;
- литье под давлением порошковых материалов;
- вакуумное литье сталей и цветных сплавов;
- литье под давлением термопластичных пластмасс с давлением впрыска 2500 бар;
- горячая объемная штамповка;
- изготовление корпусов по технологии НТСС;
- технологии литейного производства;
- каркасно-шкафное производство;
- механическая обработка материалов;
- термическая и химико-термическая обработка;
- автоматизация измерений;
- сварочное производство;
- изготовление металлокерамических и металлостеклянных изделий;

- производство печатных плат (в т.ч. рельефных);
- производство чувствительных элементов, волноводных трактов;
- гибридная микроэлектроника (СВЧ и НЧ - микроузлы).

Услугами по проведению испытаний в заводских лабораториях АО «ФНПЦ «ПО «Старт» им. М.В.Проценко» являются:

- периодические и специальные испытания;
- испытания на электромагнитную совместимость;
- входной контроль;
- метрологические испытания.

3 Порядок ведения технической документации.

Порядок ведения технической документации на предприятии зависит от многих факторов, таких как размер предприятия, виды документов, специфика деятельности и др. Однако, основные этапы ведения технической документации на любом предприятии могут быть следующими:

Разработка и утверждение правил ведения документации. Это включает стандарты наименования и описания документов, сроки их хранения, доступ к документам и т.д.

Разработка и утверждение шаблонов и форм документов. На предприятиях должны быть установлены образцы документов, которые должны быть заполнены при выполнении определенных процессов.

Регистрация документов. Вся документация должна регистрироваться в специальной журнальной книге или в электронном виде. Включает в себя наименование документа, дату, номер, автора, подпись руководителя, ответственного за регистрацию.

Утверждение документов. Документы должны быть утверждены компетентными лицами на предприятии и иметь отметки об утверждении.

Хранение документов. Документы должны храниться в соответствии с определенными правилами и сроками. Для обеспечения сохранности документов должны быть установлены требования к условиям их хранения и ведения электронной базы данных.

Очистка документации. Принятую практику удалять старые документы, которые уже не могут быть актуальными или могут привести к конфликтам.

Обновление документов. В процессе работы документация может устаревать, поэтому необходимы процедуры по перепроверке документов и обновлению их содержания.

В целом, порядок ведения технической документации может быть осуществлен как в бумажном виде, так и в электронном. Это зависит от возможностей предприятия, требований законодательства и решений руководства.

4 Порядок осуществления интеграции программных модулей на предприятии.

Требования к ПО. Классификация требований.

Классификация требований к ПО.

Когда требования собраны, команда, которая будет заниматься проектом, может приступить к их классификации и разбивке на категории.

Благодаря классификации требований можно точнее оценить сроки и определить компоненты решения. В процессе работы у вас также могут появиться идеи, касающиеся реализации проекта.

Требования к ПО могут быть:

1. Функциональными и нефункциональными.

Функциональные требования описывают функции, которые должно выполнять ПО. Например, предоставлять канал коммуникации для пользователя или переводить данные из одного формата в другой. То есть, речь идет о функционале продукта.

Нефункциональные требования касаются таких вещей, как доступность, надежность, способность к восстановлению, поддерживаемость, масштабируемость, производительность, безопасность и прочие «…ость».

2. Производными и навязанными.

3. Ориентированными на продукт или на процесс его разработки.

«Программа должна проверять права пользователя» — это требование к продукту.

«Программа должна разрабатываться инкрементально. В ходе разработки должна использоваться непрерывная интеграция» — это требование к процессу.

4. Разной приоритетности. Для назначения приоритета может использоваться шкала с фиксированными значениями «обязательно», «крайне желательно», «желательно» и «необязательно».

5. Разного масштаба. Одни требования касаются проектирования

отдельных компонентов, другие — архитектуры всего ПО. Нефункциональные требования чаще всего касаются всей программы в целом.

6. Изменчивыми или стабильными. Например, изначально может быть ясно, что требования будут меняться в ходе жизненного цикла продукта. В таком случае реализация программы должна быть толерантной к внесению изменений.

Валидация требований

Когда требования выяснены и классифицированы, нужно проверить их, обсудив со стейкхолдерами. Вы должны быть уверены, что поняли и записали все точно и что требования в целом действительно удовлетворяют нужды стейкхолдеров. Требования, которые не прошли валидацию, являются всего лишь «хотелками» тех, кто их высказал.

Если вы практикуете итеративную разработку, валидация требований будет осуществляться регулярно, при этом будут обсуждаться требования, касающиеся отдельных частей решения, т. е., они будут разбиты на логичные группы.

Обычно для проверки требований команда, реализующая решение, воспроизводит свое понимание требований стейкхолдерам. При этом может применяться начальный проект (бизнес-проект или технический, или даже оба), на котором показывается, как будут реализованы нужды стейкхолдеров.

Такие обсуждения для выяснения, все ли правильно всё поняли, устраиваются итеративно на этапах планирования. Обычно в них участвуют кросс-функциональные команды (дизайнеры, бизнес-аналитики, технические эксперты).

В некоторых случаях перед реализацией может потребоваться дополнительная подготовительная работа. Чтобы лучше продемонстрировать, как команда понимает требования, создается функциональный прототип.

В ходе валидации может выясниться, что команда не может полностью удовлетворить все требования каждого стейкхолдера. Ваша задача (как технического эксперта) — продемонстрировать, что можно сделать, и обсудить

уступки, на которые можно пойти при существующих ограничениях. Эти компромиссы должны быть приемлемыми для главных стейкхолдеров, а также укладываться в рамки бюджета, технических возможностей и прочих ограничений.

Использование функциональных прототипов

Функциональные прототипы помогают нам:

- проверить, что требования поняты правильно;
- проверить допущения разработчика;
- получить фидбэк, в котором могут содержаться новые требования.

Вы должны учитывать, что чисто внешние проблемы или проблемы, связанные с качеством, могут отвлечь стейкхолдеров. Следует постоянно подчеркивать, что именно основной функционал является самым важным в этом показе.

То, как создается прототип, определяется командой, которая занимается проектом. Кто-то предпочитает прототипы, в которых вообще не задействовано ПО (аналогичные тем, которые создаются при сборе требований). Другие отдают предпочтение специальным инструментам, с помощью которых можно быстро создать «черновик» будущей программы.

Какой бы вариант вы ни выбрали, следует учитывать, сколько времени уйдет на создание прототипа, и насколько эффективно он продемонстрирует основной функционал.

Разработка и реализация требований

Когда требования прошли валидацию у стейкхолдеров, можно приступать к разработке ПО (т. е., к реализации этих требований).

Основной интерес вашей организации — получить прибыль от программного решения. Ваша задача и ответственность — попытаться использовать методы, снижающие стоимость разработки и поддержки.

Сделать это можно, например, путем повторного использования

компонентов (внутри одной программы или из других продуктов), применения выверенных шаблонов и работы с проверенными инструментами (фреймворками) с хорошей документацией.

Специфические требования, в частности, ограничительные, могут очень сильно влиять на стоимость программ. Например, если ваш набор навыков не соответствует проекту или требования противоречат существующей архитектуре (или даже просто недостаточно хорошо в нее вписываются). Команда, занимающаяся проектом, должна найти возможные компромиссы, касающиеся таких требований.

Работая над проектом, вы должны понимать и даже ожидать, что довольно значительная часть требований изменится. Вам нужно научиться распознавать, где именно изменения будут неизбежны, и стараться проектировать программу таким образом, чтобы в нее можно было внести эти изменения.

Верификация требований к ПО

Реализация любых требований должна проверяться. Это должно делаться как на уровне отдельных функций, так и на глобальном уровне (когда дело касается, например, нефункциональных требований). В общем, мы проверяем, насколько созданное ПО соответствует заявленным требованиям.

Спланировать, как будет верифицироваться каждое требование, это одна из важных задач.

Разработчики верифицируют требования при помощи приемочного тестирования. Эти тесты демонстрируют, насколько полно были выполнены требования. Делается это путем показа, что конечный пользователь может использовать созданное ПО в предусмотренных сценариях.

В случаях, когда продемонстрировать верификацию сложнее, например, когда речь идет о нефункциональных требованиях, может понадобиться некое моделирование. Скажем, чтобы протестировать производительность, может создаваться специальное ПО, симулирующее сотни или тысячи запросов к

системе.

По мере развития программного обеспечения реализация каждого нового требования может непреднамеренно влиять на ранее реализованные требования. Это можно обойти, автоматизировав приемочные тесты. Для этой цели существует множество инструментов и библиотек.

Не следует считать приемочное тестирование единственно важным. Не забывайте покрывать свой код и другими тестами (например, модульные и интеграционными).

Приемочные тесты различаются по уровню сложности, который зависит от критериев приемки. Также в разных компаниях могут использовать разную терминологию и разные методы, из-за чего приемочное тестирование можно спутать со сквозным, функциональным или тестированием сценариев.

Помните, что суть каждого приемочного теста — просто формальная проверка того, что реализованное решение удовлетворяет требованиям, которые выдвигались к программному обеспечению. Делается это путем копирования поведения пользователей при запуске приложения.

5 Порядок ревьюирования программных продуктов на предприятии.

Системы контроля версий

Система контроля версий является прежде всего инструментам, а инструмент призван решать некоторый класс задач. Итак, система контроля версий – это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определенной версии. Мы хотим гибко управлять некоторым набором файлом, откатываться до определенных версий в случае необходимости. Можно отменить те или иные изменения файла, откатить его удаление, посмотреть кто что-то поменял. Как правило системы контроля версий применяются для хранения исходного кода, но это необязательно. Они могут применяться для хранения файлов совершенно любого типа.

Как хранить различные версии файлов? Люди пришли к такому инструменту как системы контроля версий не сразу, да и они сами бывают очень разные. Предложенную задачу можно решить с применением старого доброго copy-paste, локальных, централизованных или распределенных систем контроля версий.

Copy-paste

Известный метод при применении к данной задаче может выглядеть следующим образом: будем называть файлы по шаблону filename_{version}, возможно с добавлением времени создания или изменения.

Данный способ является очень простым, но он подвержен различным ошибкам: можно случайно изменить не тот файл, можно скопировать не из той директории (ведь именно так переносятся файлы в этой модели).

Локальная система контроля версий

Следующим шагом в развитии систем контроля версий было создание локальных систем контроля версий. Они представляли из себя простейшую базу данных, которая хранит записи обо всех изменениях в файлах.

Одним из примеров таких систем является система контроля версий RCS, которая была разработана в 1985 году (последний патч был написан в 2015 году) и хранит изменений в файлах (патчи), осуществляя контроль версий. Набор этих изменений позволяет восстановить любое состояние файла. RCS поставляется с Linux'ом.

Локальная система контроля версий хорошо решает поставленную перед ней задачу, однако ее проблемой является основное свойство — локальность. Она совершенно не преднезначена для коллективного использования.

Централизованная система контроля версий

Централизованная система контроля версий предназначена для решения основной проблемы локальной системы контроля версий.

Для организации такой системы контроля версий используется единственный сервер, который содержит все версии файлов. Клиенты, обращаясь к этому серверу, получают из этого централизованного хранилища. Применение централизованных систем контроля версий на протяжении многих лет являлась стандартом. К ним относятся CVS, Subversion, Perforce.

Такими системами легко управлять из-за наличия единственного сервера. Но при этом наличие централизованного сервера приводит к возникновению единой точки отказа в виде этого самого сервера. В случае отключения этого сервера разработчики не смогут выкачивать файлы. Самым худшим сценарием является физическое уничтожение сервера (или вылет жесткого диска), он приводит к потерю кодовой базы.

Несмотря на то, что мода на SVN прошла, иногда наблюдается обратный ход — переход от Git'а к SVN'у. Дело в том, что SVN позволяет осуществлять селективный чекаут, который подразумевает выкачку лишь некоторых файлов с сервера. Такой подход приобретает популярность при использовании

монорепозиториях, о которых можно будет поговорить позже.

Распределенная система контроля версий

Для устранения единой точки отказа используются распределенные системы контроля версий. Они подразумевают, что клиент выкачает себе весь репозиторий целиком заместо выкачки конкретных интересующих клиента файлов. Если умрет любая копия репозитория, то это не приведет к потере кодовой базы, поскольку она может быть восстановлена с компьютера любого разработчика. Каждая копия является полным бэкапом данных.

Все копии являются равноправным и могут синхронизироваться между собой. Подобный подход очень напоминает (да и является) репликацией вида master-master.

К данному виду систем контроля версий относятся Mercurial, Bazaar, Darcs и Git. Последняя система контроля версий и будет рассмотрена нами далее более детально.

Распределенная система контроля версий

Для устранения единой точки отказа используются распределенные системы контроля версий. Они подразумевают, что клиент выкачает себе весь репозиторий целиком заместо выкачки конкретных интересующих клиента файлов. Если умрет любая копия репозитория, то это не приведет к потере кодовой базы, поскольку она может быть восстановлена с компьютера любого разработчика. Каждая копия является полным бэкапом данных.

Заключение

С 18 мая по 21 июня 2023 года я прошел учебную и производственную практику в АО «ФНПЦ «ПО «Старт» им. М.В. Проценко», г. Заречный.

В результате прохождения практики на предприятии я получил практический опыт:

интеграции модулей в программное обеспечение;

отладке программных модулей;

измерении характеристик программного проекта;

использовании основных методологий процессов разработки программного обеспечения;

оптимизации программного кода с использованием специализированных программных средств.

Считаю, что задачи, поставленные передо мной в начале практики, были выполнены. Цель практики достигнута.

Список используемой литературы

1. Гниденко, И. Г. Технология разработки программного обеспечения: учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва: Издательство Юрайт, 2023. — 235 с. — (Профессиональное образование). — ISBN 978-5-534-05047-9. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/472502.

2. Черткова, Е. А. Программная инженерия. Визуальное моделирование программных систем: учебник для среднего профессионального образования / Е. А. Черткова. — 2-е изд., испр. и доп. — Москва: Издательство Юрайт, 2023. — 147 с. — (Профессиональное образование). — ISBN 978-5-534-09823-5. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/473307.

3. Грекул, В. И. Проектирование информационных систем: учебник и практикум для среднего профессионального образования / В. И. Грекул, Н. Л. Коровкина, Г. А. Левочкина. — Москва: Издательство Юрайт, 2023. — 385 с. — (Профессиональное образование). — ISBN 978-5-534-12104-9. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/476534.

4. Проектирование информационных систем: учебник и практикум для среднего профессионального образования / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук; под общей редакцией Д. В. Чистова. — Москва: Издательство Юрайт, 2023. — 258 с. — (Профессиональное образование). — ISBN 978-5-534-03173-7. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/471492.

5. Григорьев, М. В. Проектирование информационных систем: учебное пособие для среднего профессионального образования / М. В. Григорьев, И. И. Григорьева. — Москва: Издательство Юрайт, 2023. — 318 с. — (Профессиональное образование). — ISBN 978-5-534-12105-6. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/476536.

6. Древс, Ю. Г. Имитационное моделирование: учебное пособие для среднего профессионального образования / Ю. Г. Древс, В. В. Золотарёв. — 2-е

изд., испр. и доп. — Москва: Издательство Юрайт, 2023. — 142 с. — (Профессиональное образование). — ISBN 978-5-534-11951-0. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/475680.

7. Боев, В. Д. Компьютерное моделирование систем: учебное пособие для среднего профессионального образования / В. Д. Боев. — Москва: Издательство Юрайт, 2023. — 253 с. — (Профессиональное образование). — ISBN 978-5-534-10710-4. — Текст: электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/473033.