Федеральное государственное автономное образовательное
учреждение высшего образования
«Санкт-Петербургский государственный университет
аэрокосмического приборостроения»

Методология программной инженерии

Методические указания к выполнению лабораторных работ

Составители:

к.т.н., стар. преп. П.А. Охтилев, асс. А.Э. Зянчурин

Рецензент:

д.т.н., проф. М.Ю. Охтилев

Санкт-Петербург – 2022

Содержание

Лабораторная работа №2. Моделирование предметной области и разработка спецификации требований к программному обеспечению.

Цель работы

Целью работы является получение практических навыков, необходимых при обследовании объекта автоматизации и разработке спецификации требований к изделию (автоматизированной системе или программному обеспечению).

Задание на лабораторную работу

Разработка спецификации требований в соответствии с вариантом задания. Формализация ограничений, функциональных и нефункциональных требований к изделию, а также требований предметной области. Выполнение лабораторной работы предполагает ознакомление с отечественными и зарубежными стандартами.

Общие рекомендации по выполнению лабораторной работы

Введение

Современные тенденции российского рынка информационных технологий характеризуются возрастанием потребности со стороны крупнейших отраслей и предприятий страны именно в отечественных разработках в следствие чего формируется общее ожидание о будущем росте российского ИТ-рынка на фоне оттока западных ИТ-компаний. В связи с этим все большую актуальность приобретает вопрос импортозамещения западных программных продуктов альтернативными отечественными аналогами и возрастания роли государства в этом процессе. По данным единого реестра Минкомсвязи российских программ для электронных вычислительных машин и баз данных от 12 февраля 2022 года зарегистрировано порядка 7 тыс. программных продуктов, среди которых:

- отечественные операционные системы;
- офисные приложения;
- почта, коммуникации, связь;
- виртуализация и контейнеры;

- системы управления базами данных;
- средства резервного копирования;
- системы автоматизированного проектирования;
- векторные, графические и видео- редакторы;
- средства антивирусной защиты;
- средства централизованного администрирования рабочих станций;
- средства удаленного доступа;
- и мн. др.

Важно подчеркнуть, что объединяющей чертой большинства отечественных программных продуктов является заимствование Open Source решений как технологической базы, которые в массе своей являются западными, в частности, американскими разработками.

Текущее положение дел в российской ИТ-отрасли берет свое начало в 80-е годы XX века, когда государственная политика в области вычислительной техники и программирования изменилась в сторону массового копирования и заимствования программных продуктов с запада. Это привело к деградации и отставанию отечественной школы программной инженерии на десятки лет. Возросла проблема поиска и выбора квалифицированных специалистов, обладающих компетенциями создания сложных программных средств требуемого качества в разумные сроки с учетом ограничений финансовых, временных, кадровых и иных ресурсов.

Существующие на сегодняшний день положения и стандарты программной инженерии базируются на западных принципах и подходах работы, что не учитывает специфику проведения опытно-конструкторских работ по государственным заказам и не может быть применено в «чистом» виде. При этом следует иметь в виду, что государственные заказы будут занимать значительный сегмент рынка отечественного программного обеспечения еще долгое время.

Одной из важных тем современной программной инженерии является разработка и расширенный анализ требований к программному обеспечению. Данная тема весьма трудно поддается формализации, что ведет к усилению влияния человеческого фактора на данный процесс и является скорее искусством нежели ремеслом, что несовместимо с промышленным производством программного обеспечения. Данной теме будут посвящены настоящие методические указания.

Спецификация требований. Общие положения

Результатом выполнения лабораторной работы №2 является документ «Спецификация требований» в соответствии с вариантом задания, включаемый в состав отчета по лабораторной работе и отражающий свойства изделия (автоматизированной системы (АС) или программного обеспечения (ПО)), область его применения и функциональные особенности.

<table>
<tr><th colspan="2">Документ «Спецификация требований»</th></tr>
<tr><td colspan="2">Назначение</td></tr>
<tr><td colspan="2">Данный документ предназначен для фиксации, анализа и детализации требований к изделию (АС или ПО)</td></tr>
<tr><td>Задачи</td><td>Характеристика документа</td></tr>
<tr><td>• описание предметной области;
• цель и постановка задачи разработки автоматизированной/информационной системы;
• словарь данных предметной области;
• задача автоматизации в предметной области;
• диаграмма вариантов использования;
• дерево функций автоматизированной/информационной системы;
• определение бизнес-требований;
• определение функциональных требований;
• определение нефункциональных требований;
• заключение и выводы.</td><td>• однозначно интерпретируемое описание проблемы решаемой в проекте;
• причины возникновения проблемы;
• однозначно интерпретируемое описание практического эффекта от внедрения системы;
• однозначно интерпретируемое описание вариантов (сценариев) использования системы.
• обеспечение согласованности со спецификацией требований;
• обеспечение корректности разрабатываемых моделей как с точки зрения языка моделирования, так и с точки зрения ;
• обеспечение полноты при моделировании предметной области.</td></tr>
</table>

Цитаты классиков программной инженерии:
«Цель разработки программного обеспечения состоит в том, чтобы уложившись в отведенное время и бюджет, разработать качественное программное обеспечение, удовлетворяющее реальные потребности пользователей.» *«Ошибки в требованиях – наиболее часто встречающийся тип ошибок при разработке систем, а их устранение является наиболее дорогостоящим.»* *«Команда должна применить методы и процессы для того, чтобы понять решаемую проблему заказчика до начала разработки изделия.»*

Общее описание изделия

Назначение, соглашения и границы проекта

В рамках данного пункта предлагается формализовать назначение изделия (АС или ПО), требования для которого указаны в этом документе, в том числе редакцию или номер выпуска. Если эта спецификация требований относится только к части изделия, идентифицируйте эту часть или подсистему. Опишите все стандарты или типографические соглашения, включая значение стилей текста, особенности выделения или нотацию. Если вы нумеруете требования вручную, можно определить принятый формат на случай, если кому-нибудь позже понадобиться добавить требование. Кратко опишите ПО и его назначение. Покажите, как связан продукт с пользователями или корпоративными целями, а также с бизнес-целями и стратегиями. Если имеется отдельный документ о концепции и границах проекта, не повторяйте его содержимое, а просто сошлитесь на него. Если спецификацию требований к ПО предполагается разрабатывать постепенно, она должна содержать собственное положение о концепции и границах продукта в качестве подраздела долгосрочной стратегической концепции. Можно предоставить высокоуровневую сводку главной функциональности выпуска или функций, которые он должен выполнять.

Определение вариантов использования изделия

При выполнении анализа информации о предметной области необходимо сопоставлять различные ее представления на предмет обнаружения пробелов, ошибок и противоречий.

Один из точных способов поиска недостающих элементов системы для понимания – матрица *CRUD* (*Create, Read, Update, Delete* – создание, чтение, обновление, удаление). Она позволяет соотнести действия предполагаемой системы с элементами данных, что дает более конкретное представление о том, где как каждый элемент создается, считывается, обновляется и удаляется. Иногда добавляют к названию матрицы букву L указывая, что элемент данных является списком (*list*), или *M* или *C* – соответственно для обозначения перемещения или копирования данных из одного места в другое.

В таблице №1 показан пример матрицы *CRUD* Каждая ячейка указывает, как вариант использования, определенный в крайнем левом столбце, использует элементы данных, показанные в остальных столбцах. Вариант использования может создать (*C*), прочитать (*R*), обновить (*U*) или удалить (*D*) сущность. После создания матрицы посмотрите, нет ли ячейки

столбца, в которой нет одной из этих букв. Например, если сущность обновлена, но до этого ее не создавали, то откуда она взялась?

Таблица №1

Сущность / Варианты использования	Заказ	Химикат	Сотрудник, разместивший заказ на химикат	Каталог поставщика
Разместить заказ	*C*	*R*	*R*	*R*
Изменить заказ	*U, D*		*R*	*R*
Управлять складом химикатов		*C,U,D*		
Создать отчет по заказу	*R*	*R*	*R*	
Редактировать сотрудников, имеющих право размещать заказы			*C,U*	

В таблице №1 показано, что ни одна ячейка в столбце «Сотрудник, разместивший заказ на химикат» не содержит *D*. То есть ни в одном из случаев использования нельзя удалить сотрудника из списка людей, заказывавших химикаты. Интерпретировать это можно тремя способами [3]:

- удаление сотрудника, разместившего заказ на химикат, не является ожидаемой функцией программной системы;
- не был учтен вариант использования, который удаляет сотрудника, разместившего заказ на химикат;
- вариант использования «Редактировать сотрудников, имеющих право размещать заказы» некорректный. Предполагается, что пользователь может удалить из списка сотрудника, размещающего заказ на химикат, но в настоящее время эта функциональность не указана в вариантах использования.

В данном случае невозможно установить, какая интерпретация правильна, но *CRUD* – это надежный способ для обнаружения недостающих требований.

Дерево функций изделия

Дерево функций программной системы автоматизированной/информационной системы следует начать с определения основных ее основных функций в формате списка ниже [3, **Ошибка! сточник ссылки не найден.**].

1. Функция №1.
2. Функция №2.
3. Функция №3.
4. …

Затем, построить дерево функций, которое имеет следующий вид.

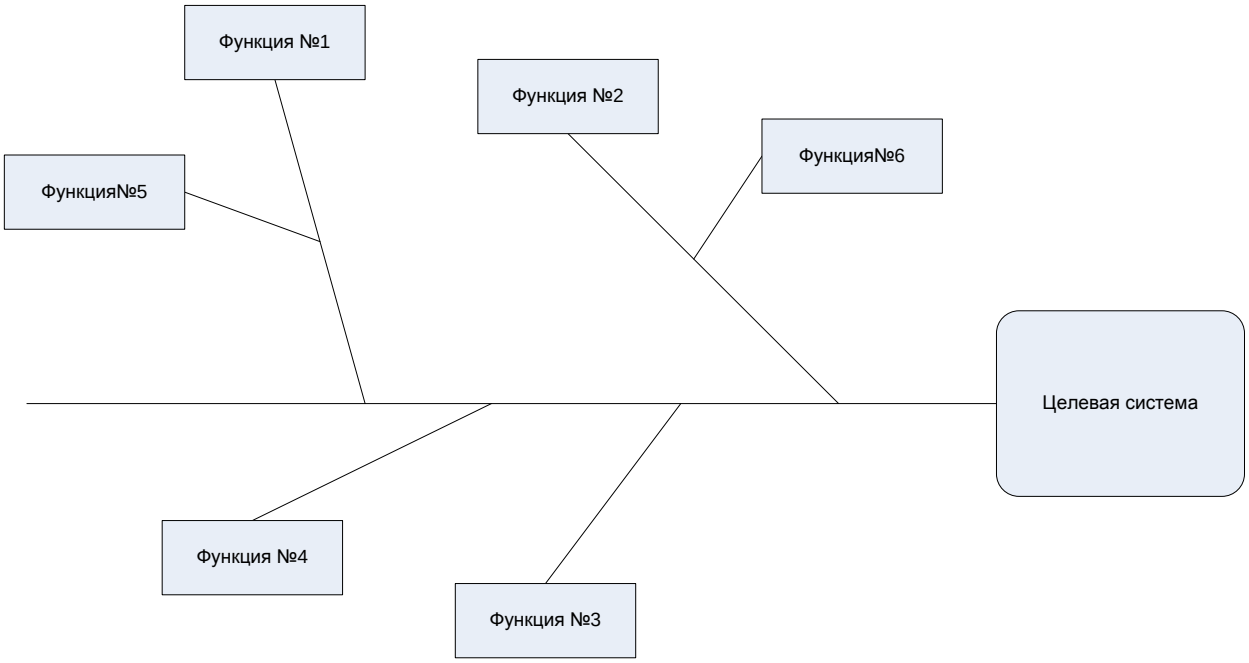

Рис. 1. Пример дерева функций программной системы

Построение дерева функций следующий состоит из следующих шагов.

1. Построить основную линию, ведущую к блоку «Целевая система» (программная система).
2. Построить блоки ассоциированные с основными функциями программной системы и построить «ветки» к основной линии (на рис. 1 «Функция №1», «Функция №2», «Функция №3», «Функция №4»).
3. Построить блоки ассоциированные с второстепенными функциями, но связанные с основными (на рис. 1. «ФункцияN5» второстепенная, но связанная с «ФункцияN1»).

Функции должны отражать вариант использования программной системы и/или должен быть направлен на устранение нежелательных эффектов, возникающих в предметной области. Функций должно быть определено не менее 20.

Операционная среда

В данном пункте необходимо описать среду исполнения, которая включает в себя аппаратные средства, операционные системы и их версии, а также серверы и баз данных вместе с организациями, в которых располагаются соответствующие базы данных, серверы и веб-сайты. Перечислите все остальные компоненты ПО или приложения, с которыми система должна быть совместима. Если в связи с разработкой новой системы нужно произвести значительную работу с технической инфраструктурой, стоит подумать о создании отдельных требований к инфраструктуре, в которой детально изложить подробности этой работы.

Функциональные требования

Функция изделия X. Функциональное требование X

При составлении функциональных требований следует учитывать следующее:

- ограничения предметной области;
- внешние интерфейсы автоматизированной/информационной системы.

Функциональные требования – это операции или действия, которые должны выполняться с объектами предметной области и которые, как правило, ассоциированы с интерфейсными элементами управления. Функциональные требования можно считать действиями программной системы. Кроме того, функциональные требования определяют места или контейнеры, с помощью которых объекты или данные отображаются пользователю. Функциональные требования определяют функциональность (поведение) программной системы, которая должна быть создана разработчиками для предоставления возможности выполнения пользователями своих обязанностей в рамках бизнес-требований и в контексте пользовательских требований.

Для формализации функциональных требований предлагается разработать карточки с описанием их свойств в виде таблицы ниже.

Формулировка требования № X	
Краткое описание:	
Приоритет:	(низкий, средний, высокий, особой важности)
Схема проверки:	(предполагаемые тесты проверки требования)
Связь с атрибутом качества:	(каким атрибутам качества соответствует)
Связь с элементом интерфейса:	(как представлен в пользовательском интерфейсе)

Для данного раздела необходимо указать не менее 20 требований. Формат карточки требований возможно переработать и предложить собственный вариант на защиту с обоснованием.

Требования к данным

Словарь данных предметной области

Формирование словаря данных предполагает идентификацию объектов в исследуемой предметной области, их определение и сопоставление со структурами данных, которые будут использоваться в предполагаемой программной системе.

Простейшие элементы данных

Простейшим элементом данных называется тот, дальнейшая декомпозиция или упрощение которого невозможно или ненужно. К определенным в таблице №1 простейшим элементам данных относятся «Количество контейнеров», «Емкость», «Единица измерения», «Идентификатор заказа» и «Сотрудник, разместивший заказ на химикат». Другие столбцы словаря данных можно использовать для описания относящихся к простейшему элементу данных типу, длине, диапазону числовых значений, списку разрешенных значений (как для «Единицы количества») и других уместных атрибутов [3].

Структура данных

Структура данных (или запись) содержит несколько элементов данных. В таблице №1 показаны следующие структуры данных: «Запрос химиката», «Пункт назначения поставки», «Заказанный химикат» и «Сотрудник, разместивший заказ на химикат». Столбец «Структура или тип данных» в словаре данных – место, где перечисляются элементы, из которых состоит структура, а элементы отделяются знаком «плюс» (+). Структуры могут содержать другие структуры: структура «Сотрудник, разместивший

заказ на химикат» включает структуру «Пункт назначения поставки». Каждый элемент данных в структуре должен быть определен в словаре данных [3].

Если элемент в структуре данных необязателен (значение, которое не должно предоставляться пользователем или системой), заключите его в скобки. В структуре «Заказанный химикат» элемент данных «Поставщик» является необязательным, поскольку сотруднику, разместившему запрос, может быть безразлично или он может просто не знать, кто именно поставляет нужный химикат.

Повторяющаяся группа

Если в структуре данных содержится несколько экземпляров элемента данных, заключите этот элемент в фигурные скобки. Покажите допустимое количество возможных повторов в формате «минимум:максимум» перед открывающей скобкой. Например, «Заказанный химикат» в структуре «Заказ химиката» является повторяющейся группой, которая оформляется так: 1:10{Заказанный химикат}. Это означает, что заказ химиката должен содержать как минимум один, но не более десяти химикатов. Если максимальное количество экземпляров в повторяющемся поле неограниченно, для указания этого факта используйте «n». Например, 3:n{повтор} означает, что определенная структура данных должна содержать не менее трех повторяющихся элементов повтор, а верхней границы на число повторений нет.

Таблица №1

Элемент данных	Описание	Структура или тип данных	Количество	Значение
Заказ химиката	Заказ нового химиката со склада или у поставщика	Идентификатор заказа + Сотрудника, разместивший заказ на химикат + Дата заказа + Счет затрат + 1:10 {Заказанный химикат}		
Пункт назначения	Место куда нужно доставить заказанный химикат			
Количество контейнеров	Количество заказываемых контейнеров определенного размера с химикатом	Положительное целое	3	
Емкость	Объем химиката в	Целое	6	

	заказываемом контейнере			
Единица измерения	Единицы измерения, в которых указывается количество заказываемого химиката	Буквенные символы	10	Граммы
Идентификатор заказа	Уникальный идентификатор	Целое	8	Генерируемый системой порядковый номер, начиная с 1
Заказанный химикат	Описание заказываемого химиката	Идентификатор заказа + Количество контейнеров + Емкость		
Сотрудник, разместивший заказ на химикат	Информация о сотруднике, заказывающем химикат			

Логическая модель данных

Широко используется такая модель данных, как диаграмма «сущность–связь» (entity-relationship diagrams, ERD) (Robertson и Robertson, 1994). Если диаграмма «сущность–связь» представляет логические группы информации предметной области и их взаимосвязи, нужно использовать диаграмму «сущность–связь» в качестве инструмента анализа требований. Анализ диаграммы «сущность–связь» помогает понять и связать компоненты данных компании или системы, даже без предположения, что продукт будет включать базу данных. Создавая эту диаграмму в ходе разработки системы, вы определяете логическую или физическую (реализацию) структуру базы данных системы. Такое представление реализации расширяет или дополняет понимание системы, которое образовалось в процессе анализа, и оптимизирует ее реализацию, скажем, в среде реляционной базы данных.

Сущностями (entities) называются физические элементы (включая людей) или агрегации элементов данных, важных для анализируемого бизнеса или для системы, которую вы планируете создать. Сущности именуются посредством существительных в единственном числе, они показаны в прямоугольниках на диаграмме «сущность–связь». На рис. 13-1 показана часть диаграммы «сущность–связь» для Chemical Tracking System c

использованием нотации Питера Чена, одной из нескольких применяемых для диаграмм такого типа систем обозначений. Другие объекты представляют действующие лица, взаимодействующие с системой (Сотрудник, разместивший заказ на химикат), физические элементы, являющиеся частью бизнес-операций (Контейнер с химикатом), и блоки данных, которые не показаны на уровне 0 диаграммы потоков данных, но показаны на ее более низком уровне (История контейнера, Химикат). В процессе проектирования физической реляционной базы данных сущности обычно становятся таблицами.

Каждая сущность описывается несколькими атрибутами; у разных экземпляров сущности значения атрибутов могут отличаться. Например, в атрибуты каждого химиката включены уникальный химический идентификатор, строгое название химиката и графическое представление его химической структуры. В словаре данных приводятся детальные определения этих атрибутов — это гарантирует, что объекты на диаграмме «сущность–связь» и соответствующие им хранилища данных на диаграмме потоков данных определены одинаково.

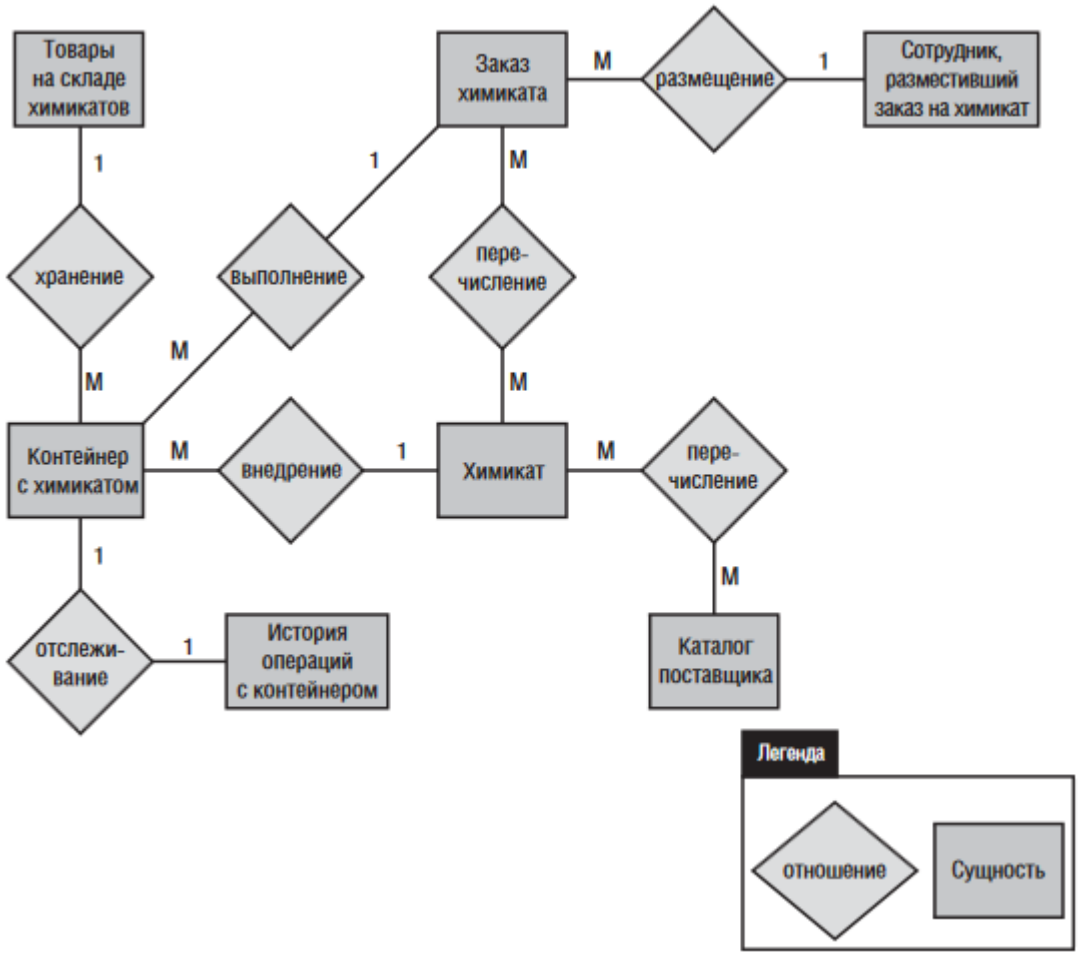

Рис. 2 Фрагмент диаграммы «сущность-связь»

Журналирование событий

Журналирование работы приложения – полезный инструмент разработчика. Даже в примитивном варианте он позволяет экономить временя на поиск ошибок и дефектов ПО. Средства журналирования, как правило, пишутся для уже имеющегося приложения, либо для разрабатываемого приложения. То есть носят прикладной характер. На сегодняшний день не существует правил или концепций их разработки – архитектурные особенности и методика ведения журналов событий зависят от конкретно взятой задачи.

Журналирование (логирование) (англ. logging, tracing) — хронологическая запись с различной (настраиваемой) степенью детализации сведений о происходящих в системе событиях (ошибки, предупреждения, сообщения) куда-либо (файл, приложение мониторинга и т. д.) для последующего анализа. Тема журналирования работы программы довольно широко распространена, многие разработчики прибегают к ведению протоколов, журналов, логов работы программы. Впрочем, эти аспекты приложений часто бывают, доступны и полезны не только разработчикам, но и пользователям программных продуктов. В данном пункте необходимо разработать не менее 5 форматов журналирования для ПО и его подсистем.

Сбор, целостность, хранение и утилизация данных

В данном пункте необходимо описать как получают и обслуживают данные. Например, при открытии канала номенклатуры данных может требоваться первым делом выполнить начальный дамп всей номенклатуры данных в принимающую систему, а после этого использовать каналы для передачи только изменений. Укажите все требования, относящиеся к защите целостности данных системы. Укажите все процедуры, которые могут потребоваться, например резервное копирование, создание контрольных точек, зеркальное отображение или проверка корректности данных. Укажите политики, которые должна применять система для хранения или утилизации данных, в том числе временных данных, метаданных, остаточных данных (таких как удаленные записи), данных в кеше, локальных копий, архивов и промежуточных архивов.

Требования к интерфейсам

Требования к пользовательским интерфейсам

В данном пункте предлагается описать логические характеристики каждого пользовательского интерфейса, который необходим системе. Некоторые из таких характеристик:

- ссылки на стандарты графического интерфейса пользователей или стилевые рекомендации для семейства продуктов, которые необходимо соблюдать;
- стандарты шрифтов, значков, названий кнопок, изображений, цветовых схем, последовательностей полей вкладок, часто используемых элементов управления, графики фирменного стиля, уведомления о зарегистрированных товарных знаках и о конфиденциальности и т.п.;
- размер и конфигурация экрана или ограничения разрешения;
- стандартные кнопки, функции или ссылки перемещения, одинаковые для всех экранов, например кнопка справки;
- сочетания клавиш;
- стандарты отображения и текста сообщений;
- стандарты проверки данных (такие как ограничения на вводимые значения и когда нужно проверять содержимое полей);
- стандарты конфигурации интерфейса для упрощения локализации ПО;
- специальные возможности для пользователей с проблемами со зрением, различением цвета и другими ограничениями.

Требования к пользовательским интерфейсам разрабатывают на основе контекстного сценария. Эти требования могут включать в себя объекты, действия и контексты. Таким образом, из контекстного сценария можно выявить потребность в следующем формате:

*Звонок (**действие**) человеку (**объект**) непосредственно из записи о встрече (**контекст**).*

Информационные требования должны отражать потребность пользователей в данных – это объекты и информация, которые должна представлять система. Если говорить в терминах приведенной выше семантики, бывает полезно считать информационные требования объектами и прилагательными, связанными с этими объектами. Например, учетные

записи, люди, документы, сообщения, песни, изображения, а также их свойства, такие как состояние, дата, размер, автор, тема.

В рамках данного раздела необходимо разработать контекстный сценарий и список требований к пользовательским интерфейсам ПО.

Требования к информационному взаимодействию

Опишите связи продукта и других компонентов ПО (идентифицированные по имени и версии), в том числе другие приложения, базы данных, операционные системы, средства, библиотеки, веб-сайты и интегрированные серийные компоненты. Укажите назначение, форматы и содержимое сообщений, данных и контрольных значений, обмен которыми происходит между компонентами ПО. Опишите соответствия между входными и выходными данными между системами и все преобразования, которые должны происходить с данными при перемещении между системами. Опишите службы, необходимые внешним компонентам ПО, и природу взаимодействия между компонентами. Определите данные, которыми будут обмениваться и к которым будут иметь общий доступ компоненты ПО. Определите нефункциональные требования, влияющие на интерфейс, такие как уровни обслуживания для времени и частоты отклика или меры и ограничения безопасности.

Требования к аппаратным интерфейсам

Опишите характеристики каждого интерфейса между компонентами ПО и оборудования системы. В описание могут входить типы поддерживаемых устройств, взаимодействия данных и элементов управлений между ПО и оборудованием, а также протоколы взаимодействия, которые будут использоваться. Перечислите входные и выходные данные, их формат, разрешенные значения или их диапазоны, а также все временные характеристики, о которых должны знать разработчики. Если такой информации очень много, лучше создать отдельный документ спецификации аппаратного интерфейса.

Нефункциональные требования и атрибуты качества

Нефункциональные требования — это условия, при которых продукт должен работать, и качества, которыми он должен обладать.

Нефункциональные требования описывают эксплуатационные качества к ПО. К нефункциональным требованиям также относят ограничения изделия.

Атрибуты качества описывают дополнительные характеристики продукта в различных “измерениях”, важных для пользователей и/или разработчиков. Атрибуты касаются вопросов портируемости, прозрачности взаимодействия с другими системами, целостности, устойчивости и т.п. Приведем краткий перечень некоторых из них.

Производительность

Краткое описание:	Как быстро и предсказуемо ПО реагирует на ввод информации пользователем и на другие события

Доступность

Краткое описание:	Насколько сервисы ПО доступны, когда и где они нужны

Целостность

Краткое описание:	Насколько хорошо в ПО реализованы механизмы защиты от неточности и потери данных

Совместимость

Краткое описание:	Как быстро и предсказуемо ПО реагирует на ввод информации пользователем и на другие события

Надежность

Краткое описание:	Как долго ПО функционирует до первого сбоя

Безопасность

Краткое описание:	Насколько хорошо в По реализована защита от неправомерного доступа к данным

Устойчивость

Краткое описание:	Как хорошо ПО функционирует в неожиданных условиях работы

Масштабируемость

Краткое описание:	Насколько хорошо ПО справляется с увеличением числа пользователей, транзакций, серверов и других расширений

В данном пункте необходимо описать реализацию каждого из представленных атрибутов, а также дополнить перечень атрибутов качества не менее 10-ю позициями.

Нормативные требования

Требования стандартов

В данном пункте необходимо определить требования и ограничения налагаемые стандартами той предметной области, для которой разрабатывается изделие. Примерами могут служить отраслевые стандарты космической, оборонной, медицинской, транспортной, строительной промышленности и мн. др.

Требования защиты информации

Данный пункт необходимо разработать на основе состава мер защиты информации и их базовые наборы для соответствующего класса защищенности информационной системы (в ред. Приказа ФСТЭК России от 15.02.2017 N 27).

Ссылка на информационный ресурс: https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty/110-prikazy/702-prikaz-fstek-rossii-ot-11-fevralya-2013-g-n-17

Требования к локализации

Требования по интернационализации и локализации обеспечивают возможность использовать продукт в других странах, региональных стандартах и географических районах, отличающихся от тех, в которых он был создан. Такие требования могут быть направлены на разрешение различий в валютах, форматировании дат, чисел, адресов и телефонных номеров, языках, в том числе различных вариантах одного языка (например, американского и британского вариантов английского), используемых символах и наборах символов, личных именах и фамилиях, часовых поясах, международных нормативных актах и законах, культурных и политических традициях, размере используемой бумаги, единицах веса и меры, электрическом напряжении и конфигурации электрических разъемов и во многом другом. Требования по интернационализации и локализации вполне могут повторно использоваться во многих проектах.

Подготовка к защите лабораторной работы

Каждый студент должен по результатам проведенной работы предъявить отчет (в электронном или бумажном виде) и подготовить устный доклад на 3-5 минут, в рамках которого он должен пояснить, что было сделано, какие возникли сложности, проблемы в реализации этапов и какие были сделаны выводы в процессе их реализации.

Список литературы

1. IEEE Swebok V 3.0 Guide to the Software Engineering. Body of Knowledge. A Project of the IEEE Computer Society. Inetrnet resource. URL: www.computer.org (дата обращения: 10.03.2022).
2. *Брукс Ф.* Мифический человеко-месяц или как создаются программы. – Пер. с англ. – СПб.: Символ-Плюс, 2007. –304 с.: ил. ISBN 5-93286-005-7.
3. *Вигерс К., Битти Д.* Разработка требований к программному обеспечению/Пер. с англ. – М.: Издательство "БХВ-Петербург", 2014. – 736 с.: ил.
4. ГОСТ 19.001-77 Единая система программной документации. Общие положения.

5. ГОСТ 19.201-78 Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.
6. ГОСТ 34.003-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения.
7. ГОСТ 34.602-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
8. ГОСТ Р 50922-2006 Защита информации. Основные термины и определения.
9. ГОСТ Р ИСО 18629-1-2010 Системы промышленной автоматизации и интеграция. Язык спецификаций процесса.
10. ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология (ИТ). Системная и прикладная инженерия. Процессы жизненного цикла программных средств.
11. ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология (ИТ). Системная и прикладная инженерия. Процессы жизненного цикла программных средств.
12. *Купер А., Рейман Р., Кронин Д.* Алан Купер об интерфейсе. Основы проектирования и взаимодействия. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 688 с., ил.
13. *Леффингуэл, Дин, Уидриг, Дон.* Принципы работы с требованиями к программному обеспечению. Унифицированный подход.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 448 с.: ил. – Парал. Тит. Англ. ISBN 5-8459-0275-4 (рус.).
14. *Орлик С.* Программная инженерия. Программные требования. Перевод Swebok 2004. 2010 г. Интернет-ресурс. URL: http://swebok.sorlik.ru. Дата обращения: 10.01.2022.
15. *Соммервилл, Иан*. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 624 с.: ил. – Парал. тит. англ. ISBN 5-8459-0330-0 (рус.).

Варианты заданий для выполнения лабораторных работ

Тему для выполнения лабораторных работ следует выбирать в соответствии с темой выпускной квалификационной работы по согласованию с преподавателем. В случае если тема выпускной квалификационной работы не определена научным руководителем, то лабораторные работы следует выполнять в соответствии с вариантами заданий приведенными в таблице ниже, также по согласованию с преподавателем.

Номер варианта задания	Задание на лабораторную работу
1	Разработка программного обеспечения для автоматизированной/информационной системы железной дороги
2	Разработка программного обеспечения для автоматизированной/информационной системы авиакомпании
3	Разработка программного обеспечения для автоматизированной/информационной системы аэропорта
4	Разработка программного обеспечения для автоматизированной/информационной системы морского порта
5	Разработка программного обеспечения для автоматизированной/информационной системы автобусного вокзала
6	Разработка программного обеспечения для автоматизированной/информационной системы школы
7	Разработка программного обеспечения для автоматизированной/информационной системы библиотеки
8	Разработка программного обеспечения для автоматизированной/информационной системы университета
9	Разработка программного обеспечения для автоматизированной/информационной системы службы занятости
10	Разработка программного обеспечения для автоматизированной/информационной системы службы социальной защиты
11	Разработка программного обеспечения для автоматизированной/информационной системы поликлиники
12	Разработка программного обеспечения для автоматизированной/информационной системы обязательного медицинского страхования
13	Разработка программного обеспечения для автоматизированной/информационной системы пенсионного фонда
14	Разработка программного обеспечения для автоматизированной/информационной системы выставочного комплекса
15	Разработка программного обеспечения для

	автоматизированной/информационной системы для организации НИОКР
16	Разработка программного обеспечения для автоматизированной/информационной системы издательства
17	Разработка программного обеспечения для автоматизированной/информационной системы редакции газеты
18	Разработка программного обеспечения для автоматизированной/информационной системы типографии
19	Разработка программного обеспечения для автоматизированной/информационной системы гостиницы
20	Разработка программного обеспечения для автоматизированной/информационной системы киноцентра
21	Разработка программного обеспечения для автоматизированной/информационной системы фирмы по прокату автомобилей
22	Разработка программного обеспечения для автоматизированной/информационной системы букмекерской фирмы
23	Разработка программного обеспечения для автоматизированной/информационной системы фондовой биржи
24	Разработка программного обеспечения для автоматизированной/информационной системы банка
25	Разработка программного обеспечения для автоматизированной/информационной системы лизинговой компании
26	Разработка программного обеспечения для автоматизированной/информационной системы туристического агентства
27	Разработка программного обеспечения для автоматизированной/информационной системы фильмотеки
28	Разработка программного обеспечения для автоматизированной/информационной системы агентства недвижимости
29	Разработка программного обеспечения для автоматизированной/информационной системы страховой организации
30	Разработка программного обеспечения для автоматизированной/информационной системы автошколы
31	Разработка программного обеспечения для автоматизированной/информационной системы оператора связи
32	Разработка программного обеспечения для автоматизированной/информационной системы автосервиса
33	Разработка программного обеспечения для автоматизированной/информационной системы для оказания госуслуг
34	Разработка программного обеспечения для автоматизированной/информационной системы фирмы по сборке и продаже

	компьютеров и комплектующих
35	Разработка программного обеспечения для автоматизированной/информационной системы транспортной фирмы
36	Разработка программного обеспечения для автоматизированной/информационной системы супермаркета
37	Разработка программного обеспечения для автоматизированной/информационной системы книжного магазина
38	Разработка программного обеспечения для автоматизированной/информационной системы ломбарда
39	Разработка программного обеспечения для автоматизированной/информационной системы ГИБДД
40	Разработка программного обеспечения для автоматизированной/информационной системы спортивного клуба
41	Разработка программного обеспечения для автоматизированной/информационной системы интернет-провайдера
42	Разработка программного обеспечения для автоматизированной/информационной системы интернет-магазина
43	Разработка программного обеспечения для автоматизированной/информационной системы интернет-аукциона
44	Разработка программного обеспечения для автоматизированной/информационной системы почтовой службы
45	Разработка программного обеспечения для автоматизированной/информационной системы предприятия ЖКХ
46	Разработка программного обеспечения для автоматизированной/информационной системы рекламного агентства
47	Разработка программного обеспечения для автоматизированной/информационной системы курьерской фирмы
48	Разработка программного обеспечения для автоматизированной/информационной системы ресторанного комплекса
49	Разработка программного обеспечения для автоматизированной/информационной системы службы такси
50	Разработка программного обеспечения для автоматизированной/информационной системы службы технической поддержки