МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

*Изображение выглядит как текст, Шрифт, снимок экрана, логотип

Автоматически созданное описание*

**ЛАБОРАТОРНАЯ РАБОТА №3**

**«**Объектно-ориентированное программирование**»**

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБс-324», «АВТФ»  *Петров Максим Игоревич*  «28» октября 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Ассистент кафедры ЗИ*  *Исаев Глеб Андреевич*  «\_\_\_» \_\_\_\_\_\_ 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2024

**Цели и задачи работы:** изучение основных принципов объектно-ориентированного программирования и основ юнит-тестирования.

**Задание к работе:** самостоятельно решить задачи в соответствии с индивидуальным вариантом.

Задания:

Реализовать классы с базовым набором операций (private, public) на основе лабораторной работы 1 на C++. Все структуры.

Реализовать класс с базовым набором операций (private, public) на основе лабораторной работы 1 на Go. Очередь.

Реализовать покрытие тестами (не менее 90%) в проектах 1 и 2. Создать HTML-отчет о покрытии, который предоставляет визуальный анализ того, как много кода было протестировано.

Выполнить сериализацию и десериализацию данных бинарного и текстового форматов для проектов 1 и 2.

UML – диаграммы.

Теоретическая часть: Анализ российских стандартов оценки качества ПО. Основные виды тестирования.

# **UML – Диаграммы**

## Массив:

|  |
| --- |
| Array |
| - capacity: size\_t  - size: size\_t  - string: \*arr |
| +Array()  + ~Array()  +void ShowArray() const  +void addToEnd(string value)  +void addAtIndex(size\_t index, string value)у  +string getIndex(size\_t index) const  +void removeAtIndex(size\_t index)  +void replaceAtIndex(size\_t index, string value)  +bool find(const string& value) const  +size\_t getSize() const;  +void resize() |

## Списки:

|  |
| --- |
| SinglyLinkedList |
| - head: Node\*  - size: elementCount |
| + SinglyLinkedList()  + ~SinglyLinkedList()  + bool isEmpty() const;  + void print()  + void pushFront(string value)  + void pushBack(string value)  + void popFront()  + void popBack()  + void removeAt(string value)  + bool find(string value)  + size\_t getSize() const;  + void clearSList();  + Node\* getHead() const; |
| <<struct>> Node  - data: string  - next: Node\*  + Node(string value) |

|  |
| --- |
| DoublyLinkedList |
| - head: DoubleNode\*  - tail: DoubleNode\*  - size: elementCount |
| + DoublyLinkedList()  + ~DoublyLinkedList()  + bool isEmpty() const;  + void pushFront(string value)  + void pushBack(string value)  + void popFront()  + void popBack()  + void removeAt(string value)  + bool find(string value)  + void print()  + size\_t getSize() const  + void clearDList()  + DoubleNode\* getHead() const  ) |
| <<struct>> DoubleNode  - data: string  - next: DoubleNode\*  - prev: DoubleNode\*  + DoubleNode(string value) |

## Стек:

|  |
| --- |
| Stack |
| - data: string\*  - head: int  - capacity: int |
| + Stack(size\_t size)  + Stack()  + ~Stack()  + void push(string value)  + string pop()  + string peek()  + bool isEmpty()  + size\_t size() |

## Очередь:

|  |
| --- |
| Queue |
| - data: string\*  - front: int  - rear: int  - capacity: int  - size: size\_t |
| + Queue(int cap)  + Queue()  + ~Queue()  + bool isempty()  + void push(string value)  + string pop()  + string peek()  + int Size() |

## Хеш-таблица:

|  |
| --- |
| HashTable |
| - TABLE\_SIZE: const size\_t = 100  - table: KeyValuePair\* [TABLE\_SIZE]  - sizetable: int |
| + int sizetable  + Hash\_table()  + ~Hash\_table()  + void insert(const string& key, const string& value)  + bool get(const string& key, string& value)  + bool remove(const string& key)  + int size() const  + HNode\*\* getTable() { return table; } |
| <<struct>> HNode  - key: string  - value: string  - next:HNode\* |

## Дерево:

|  |
| --- |
| CompleteBinaryTree |
| - root: NodeT\*  - size: size\_t |
| + CompleteBinaryTree()  + ~CompleteBinaryTree()  + void print()  + string toString()  + void insert(int value)  + bool search(int value)  + bool isComplete()  + void clearTree()  + NodeT\* getRoot() const { return root; }  + size\_t getSize() const { return size; }  + void setRoot(NodeT\* newRoot) { root = newRoot; } |
| <<struct>> NodeT  - data: int  - left: NodeT\*  - right: NodeT\*  + NodeT(int value)  - NodeT\* root  - size\_t size  - NodeT\* \_insert(NodeT\* node, int value)  - bool search(NodeT\* node, int value)  - bool isComplete(NodeT\* node, int index, int totalNodes)  - int countNodes(NodeT\* node)  - string \_toString(NodeT\* node)  - void printTree(NodeT\* node, int depth)  - void clear(NodeT\* node) |

Теоретическая часть

Настоящий стандарт является составной частью серии международных стандартов SQuaRE, которая состоит из следующих разделов:

- раздел "Менеджмент качества" (ИСО/IEC 2500n),

- раздел "Модель качества" (ИСО/МЭК 2501n),

- раздел "Измерение качества" (ИСО/МЭК 2502n),

- раздел "Требования к качеству" (ИСО/МЭК 2503n),

- раздел "Оценка качества" (ИСО/МЭК 2504n),

- раздел "Расширение SQuaRE" (ИСО/МЭК 25050 - ИСО/МЭК 25099).

Для выполнения разнообразных функций как в бизнесе, так и для персонального назначения в современных условиях все большее распространение получают программные продукты и преимущественно программные вычислительные системы. Реализация целей и задач для удовлетворения личных потребностей, для успеха в бизнесе и/или для безопасности человека опирается на высококачественные программное обеспечение и системы. Высококачественные программные продукты и преимущественно программные вычислительные системы имеют важное для заинтересованных сторон значение в производстве материальных ценностей и предотвращении возможных негативных последствий.

У программных продуктов и преимущественно программных вычислительных систем много заинтересованных сторон, в число которых входят разработчики, приобретатели, пользователи или клиенты компаний, использующих преимущественно программные вычислительные системы. Подробная спецификация и оценка качества программного обеспечения и преимущественно программных вычислительных систем являются ключевыми факторами в обеспечении полезности для заинтересованных сторон. Оценка может быть выполнена на основе определения необходимых и требуемых характеристик качества, связанных с задачами заинтересованных сторон и целями системы, включая характеристики качества, относящиеся к системе программного обеспечения и данным, а кроме того, и воздействие системы на ее заинтересованные стороны. Важно, чтобы, по возможности, характеристики качества были определены, измерены и оценены с использованием проверенных или широко распространенных показателей и методов измерения. Для идентификации соответствующих характеристик качества, которые могут далее использоваться для определения требований, критериев их удовлетворения и соответствующих показателей, могут быть использованы модели качества из настоящего документа.

Настоящий международный стандарт разработан на основе ИСО/МЭК 9126 "Программная инженерия - Качество продукта", который был разработан для удовлетворения вышеуказанных нужд и в котором были определены шесть характеристик качества и описана модель процесса оценки программного продукта.

ИСО/МЭК 9126 был заменен двумя связанными между собой стандартами: ИСО/МЭК 9126 "Программная инженерия - Качество продукта" и ИСО/МЭК 14598 "Программная инженерия - Оценка продукта".

Данный международный стандарт является результатом пересмотра ИСО/МЭК 9126-1. В него входят те же характеристики качества программного обеспечения с некоторыми поправками:

- область применения моделей качества была расширена, с тем чтобы включить в себя вычислительные системы и качество при использовании с системной точки зрения;

- в качестве характеристики качества при использовании было добавлено "Покрытие контекста" с подхарактеристиками, "Полнота контекста" и "Гибкость";

- как характеристика, а не подхарактеристика функциональности была добавлена "Безопасность", с подхарактеристиками "Конфиденциальность", "Целостность", "Безотказность", "Отслеживаемость" и "Подлинность";

- была добавлена как характеристика "Совместимость" (включая функциональную совместимость и сосуществование);

- были добавлены следующие подхарактеристики: "Функциональная полнота", "Емкость", "Защищенность от ошибки пользователя", "Доступность", "Готовность", "Модульность" и "Возможность многократного использования";

- подхарактеристики соответствия были удалены, поскольку они являются в соответствии с законами и правилами частью общих требований к системе, а не частью характеристики качества;

- модели внутреннего и внешнего качества были объединены в составе модели качества продукта;

- там, где это представилось возможным, специфичные для программного обеспечения определения были заменены на универсальные;

- нескольким характеристикам и подхарактеристикам были даны более точные названия.

Полный перечень изменений приводится в приложении A.

Данный международный стандарт предназначен для применения в сочетании с другими частями международных стандартов серии SQuaRE (ИСО/МЭК 25000 - ИСО/МЭК 25099) и ИСО/МЭК 14598 до тех пор, пока он не заменен серией международных стандартов ИСО/МЭК 2504n.

На рисунке 1 (адаптирован из ИСО/МЭК 25000) показана организация серии международных стандартов SQuaRE, которая представлена семействами стандартов, называемых также разделами.

|  |
| --- |
|  |
| Image |

Рисунок 1 - Организация серии международных стандартов SQuaRE

Серия стандартов SQuaRE состоит из следующих разделов стандартов:

- ИСО/МЭК 2500n - раздел "Менеджмент качества". Международные стандарты, входящие в этот раздел, определяют общие модели, термины и определения, используемые далее во всех других международных стандартах серии SQuaRE. В разделе также представлены требования и методические материалы, касающиеся функций поддержки, которые отвечают за управление требованиями к программному продукту, его спецификацией и оценкой;

- ИСО/МЭК 2501n - раздел "Модель качества". Международные стандарты, которые входят в этот раздел, представляют детализированные модели качества вычислительных систем и программного обеспечения, качества при использовании и качества данных. Кроме того, представлено практическое руководство по использованию модели качества;

- ИСО/МЭК 2502n - раздел "Измерение качества". Международные стандарты, входящие в этот раздел, включают в себя эталонную модель измерения качества программного продукта, математические определения показателей качества и практическое руководство по их использованию. В этом разделе представлены показатели внутреннего качества программного обеспечения, показатели внешнего качества программного обеспечения и показатели качества при использовании. Кроме того, определены и представлены элементы показателей качества (ЭПК), формирующие основу для вышеперечисленных показателей;

- ИСО/МЭК 2503n - раздел "Требования к качеству". Международные стандарты, которые входят в этот раздел, определяют требования к качеству на основе моделей качества и показателей качества. Такие требования к качеству могут использоваться в процессе формирования требований к качеству программного продукта перед разработкой или как входные данные для процесса оценки;

- ИСО/МЭК 2504n - раздел "Оценка качества". Международные стандарты, которые входят в этот раздел, формулируют требования, рекомендации и методические материалы для оценки программного продукта, выполняемой как оценщиками, так и заказчиками или разработчиками. Кроме того, в них представлена поддержка документирования показателя измерения как модуля оценки;

- ИСО/МЭК 25050-25099 - раздел "Расширение SQuaRE". Международные стандарты этого раздела в настоящее время включают в себя требования к качеству готового коммерческого (коробочного) программного обеспечения и общему промышленному формату для отчетов по удобству использования.

Модели качества данного международного стандарта в сочетании с ИСО/МЭК 12207 и ИСО/МЭК 15288 могут использоваться, в частности, для процессов, связанных с определением требований, для верификации и валидации с особым акцентом на спецификации и оценки требований к качеству. В ИСО/МЭК 25030 определено, каким образом модели качества можно использовать для требований к качеству программного обеспечения, а ИСО/МЭК 25040 описывает применение модели качества в процессе оценки качества программного обеспечения.

В сочетании с ИСО/МЭК 15504, который относится к оценке процессов программного обеспечения, настоящий международный стандарт обеспечивает:

- основы определения качества программного продукта в процессах "поставщик-потребитель";

- поддержку анализа, верификации и валидации и основы количественной оценки качества в процессах поддержки;

- поддержку настройки целей качества в процессе управления организацией.

Настоящий стандарт может быть использован в сочетании с ИСО 9001, который посвящен процессам обеспечения качества, для обеспечения:

- поддержки определения цели качества;

- поддержки анализа, верификации и валидации проекта.

     1 Область применения

Настоящий стандарт определяет:

a) модель качества при использовании, в состав которой входят пять характеристик, некоторые из которых, в свою очередь, подразделены на подхарактеристики. Эти характеристики касаются результата взаимодействия при использовании продукта в определенных условиях. Данная модель применима при использовании полных человеко-машинных систем, включая как вычислительные системы, так и программные продукты;

b) модель качества продукта, в состав которой входят восемь характеристик, которые, в свою очередь, подразделены на подхарактеристики. Характеристики относятся к статическим и динамическим свойствам программного обеспечения и вычислительных систем. Модель применима как к компьютерным системам, так и к программным продуктам.

Характеристики, определяемые обеими моделями, применимы к любым программным продуктам и компьютерным системам. Характеристики и подхарактеристики обеспечивают единую терминологию для определения спецификации, измерения и оценки качества систем и программного обеспечения. Модели предоставляют также множество характеристик качества, с которыми для полноты картины можно сравнить заявленные требования к качеству.

Примечание - Несмотря на то, что область применения модели качества продукта относится к программному обеспечению и компьютерным системам, многие характеристики применимы также и к более широкому кругу систем и служб.

ИСО/МЭК 25012 определяет модель качества данных, которая дополняет данную модель.

Область применения моделей не включает в себя чисто функциональные свойства (см. C.6), однако в нее включена функциональная пригодность (см. 4.2.1).

Область применения моделей качества включает в себя спецификацию поддержки и оценку программного обеспечения и преимущественно программных вычислительных систем с разных точек зрения, которые связаны с их приобретением, требованиями, разработкой, использованием, оценкой, поддержкой, обслуживанием, обеспечением качества и управлением им, а также менеджментом и аудитом. Модели могут, к примеру, использоваться разработчиками, приобретателями, персоналом обеспечения качества и управления им, а также независимыми оценщиками, в особенности ответственными за спецификацию и оценку качества программного продукта. Деятельность во время разработки продукции, при которой могут быть использованы модели качества, включает в себя:

- определение требований к программному обеспечению и системе;

- подтверждения полноты определения требований;

- определение целей проектирования программного обеспечения и системы;

- определение целей тестирования программного обеспечения и системы;

- идентификацию критериев контроля качества в рамках обеспечения качества;

- определение критериев приемки программного продукта и/или преимущественно программной вычислительной системы;

- установление необходимых для этого показателей характеристик качества.

     2 Соответствие

Любое требование к качеству, спецификация качества или оценка качества соответствуют настоящему стандарту только в тех случаях, если:

a) используются модели качества, определенные в 4.1 и 4.2; или

b) используется адаптированная модель качества, все изменения которой обоснованы и для которой обеспечивается отображение на стандартную модель.

     3 Основы модели качества

     3.1 Модели качества

Качество системы - это степень удовлетворения системой заявленных и подразумеваемых потребностей различных заинтересованных сторон, которая позволяет, таким образом, оценить достоинства. Эти заявленные и подразумеваемые потребности представлены в международных стандартах серии SQuaRE посредством моделей качества, которые представляют качество продукта в виде разбивки на классы характеристик, которые в отдельных случаях далее разделяются на подхарактеристики. (Некоторые подхарактеристики разделяются далее на под-подхарактеристики.) Подобная иерархическая декомпозиция обеспечивает удобную разбивку качества продукта на классы. Однако множество подхарактеристик, связанных с характеристикой, избранной для представления типичных проблем, необязательно будет исчерпывающим.

Измеримые, связанные с качеством свойства системы называют свойствами качества, связанными с соответствующими показателями качества. Чтобы прийти к показателям характеристики или подхарактеристики качества в случаях, когда характеристика или подхарактеристика не может быть непосредственно измерена, необходимо идентифицировать подмножество свойств, которое в совокупности покрывает характеристику или подхарактеристику, получить показатели качества для каждого свойства и, объединив их в вычислительном отношении, достигнуть полученного показателя качества, соответствующего характеристике или подхарактеристике качества (см. приложение C). На рисунке 2 показаны отношения между характеристиками и подхарактеристиками качества и свойствами качества.

|  |
| --- |
|  |
| Image |

Рисунок 2 - Структура, используемая для моделей качества

К настоящему времени в серии SQuaRE имеются три модели качества: модель качества при использовании и модель качества продукта, определенные в настоящем стандарте, и модель качества данных, определенная в ИСО/МЭК 25012. Совместное использование моделей качества дает основание считать, что учтены все характеристики качества. Данные модели обеспечивают множество характеристик качества, в которых заинтересован широкий круг лиц, таких как: разработчики программного обеспечения, системные интеграторы, приобретатели, владельцы, специалисты по обслуживанию, подрядчики, профессионалы обеспечения и управления качеством и пользователи.

Не все характеристики качества из полного множества, обеспечиваемого этими моделями, значимы для конкретной заинтересованной стороны. Тем не менее каждая категория заинтересованных лиц должна быть учтена при анализе и рассмотрении важности характеристик качества для каждой модели до завершения формирования набора характеристик качества, которые будут использоваться, чтобы установить, например, требования к производительности продукции и системы или критерии оценки.

     3.2 Модель качества при использовании

Модель качества при использовании определяет в 4.1 пять характеристик, связанных с результатами взаимодействия с системой: результативность, производительность, удовлетворенность, свободу от риска и покрытие контекста (см. рисунок 3 и таблицу 3). Каждая характеристика применима для различных видов деятельности заинтересованных лиц, например, для взаимодействия оператора или поддержки разработчика.

|  |
| --- |
|  |
| Image |

Рисунок 3 - Модель качества при использовании

Качество при использовании системы характеризует воздействие продукции (система или программный продукт) на заинтересованную сторону. Оно определяется качествами программного обеспечения, аппаратных средств, операционной среды, а также характеристиками пользователей, задач и социальной среды. Все эти факторы вносят свой вклад в качество системы при использовании.

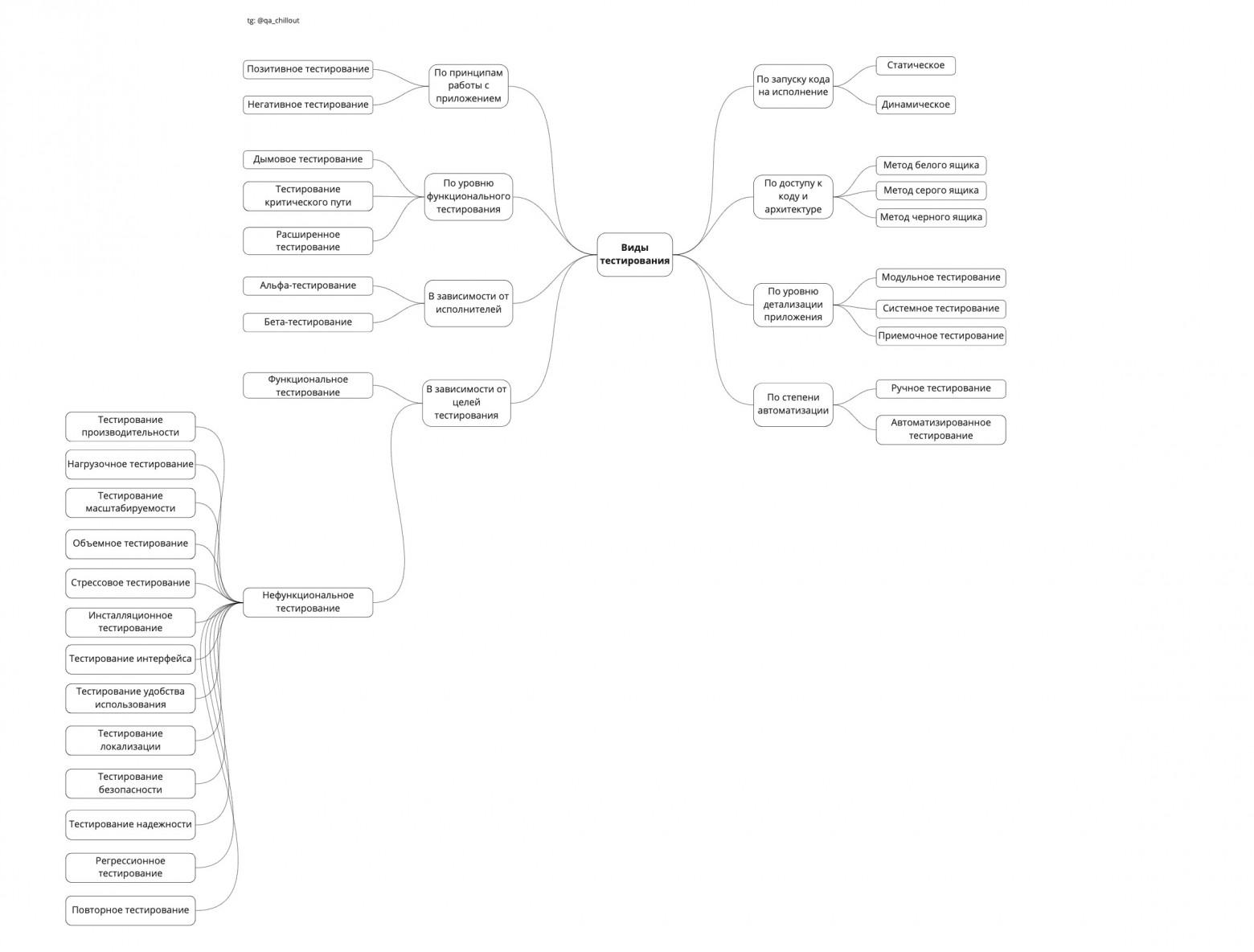
Термины и определения для каждой характеристики качества при использовании приводятся в 4.1.

Примеры показателей качества при использовании приводятся в техническом отчете ИСО/МЭК ТО 9126-4, который должен быть заменен ИСО/МЭК 25024.

     3.3 Модель качества продукта

Модель качества продукта описана в 4.2. Она сводит свойства качества системы/программного продукта к восьми характеристикам, которыми являются: функциональная пригодность, уровень производительности, совместимость, удобство пользования, надежность, защищенность, сопровождаемость и переносимость (мобильность). Каждая характеристика, в свою очередь, состоит из ряда соответствующих подхарактеристик.

Основные виды тестирования ПО

Вид тестирования — это совокупность активностей, направленных на тестирование заданных характеристик системы или её части, основанная на конкретных целях.  
  


1. Классификация по запуску кода на исполнение:  
   * Статическое тестирование — процесс тестирования, который проводится для верификации практически любого артефакта разработки: программного кода компонент, требований, системных спецификаций, функциональных спецификаций, документов проектирования и архитектуры программных систем и их компонентов.
   * Динамическое тестирование — тестирование проводится на работающей системе, не может быть осуществлено без запуска программного кода приложения.
2. Классификация по доступу к коду и архитектуре:  
   * Тестирование белого ящика — метод тестирования ПО, который предполагает полный доступ к коду проекта.
   * Тестирование серого ящика — метод тестирования ПО, который предполагает частичный доступ к коду проекта (комбинация White Box и Black Box методов).
   * Тестирование чёрного ящика — метод тестирования ПО, который не предполагает доступа (полного или частичного) к системе. Основывается на работе исключительно с внешним интерфейсом тестируемой системы.
3. Классификация по уровню детализации приложения:  
   * Модульное тестирование — проводится для тестирования какого-либо одного логически выделенного и изолированного элемента (модуля) системы в коде. Проводится самими разработчиками, так как предполагает полный доступ к коду.
   * Интеграционное тестирование — тестирование, направленное на проверку корректности взаимодействия нескольких модулей, объединенных в единое целое.
   * Системное тестирование — процесс тестирования системы, на котором проводится не только функциональное тестирование, но и оценка характеристик качества системы — ее устойчивости, надежности, безопасности и производительности.
   * Приёмочное тестирование — проверяет соответствие системы потребностям, требованиям и бизнес-процессам пользователя.
4. Классификация по степени автоматизации:  
   * Ручное тестирование.
   * Автоматизированное тестирование.
5. Классификация по принципам работы с приложением  
   * Позитивное тестирование — тестирование, при котором используются только корректные данные.
   * Негативное тестирование — тестирование приложения, при котором используются некорректные данные и выполняются некорректные операции.
6. Классификация по уровню функционального тестирования:  
   * Дымовое тестирование (smoke test) — тестирование, выполняемое на новой сборке, с целью подтверждения того, что программное обеспечение стартует и выполняет основные для бизнеса функции.
   * Тестирование критического пути (critical path) — направлено для проверки функциональности, используемой обычными пользователями во время их повседневной деятельности.
   * Расширенное тестирование (extended) — направлено на исследование всей заявленной в требованиях функциональности.
7. Классификация в зависимости от исполнителей:  
   * Альфа-тестирование — является ранней версией программного продукта. Может выполняться внутри организации-разработчика с возможным частичным привлечением конечных пользователей.
   * Бета-тестирование — программное обеспечение, выпускаемое для ограниченного количества пользователей. Главная цель — получить отзывы клиентов о продукте и внести соответствующие изменения.
8. Классификация в зависимости от целей тестирования:  
   * Функциональное тестирование (functional testing) — направлено на проверку корректности работы функциональности приложения.
   * Нефункциональное тестирование (non-functional testing) — тестирование атрибутов компонента или системы, не относящихся к функциональности.  
       
     1. Тестирование производительности (performance testing) — определение стабильности и потребления ресурсов в условиях различных сценариев использования и нагрузок.
     2. Нагрузочное тестирование (load testing) — определение или сбор показателей производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству).
     3. Тестирование масштабируемости (scalability testing) — тестирование, которое измеряет производительность сети или системы, когда количество пользовательских запросов увеличивается или уменьшается.
     4. Объёмное тестирование (volume testing) — это тип тестирования программного обеспечения, которое проводится для тестирования программного приложения с определенным объемом данных.
     5. Стрессовое тестирование (stress testing) — тип тестирования направленный для проверки, как система обращается с нарастающей нагрузкой (количеством одновременных пользователей).
     6. Инсталляционное тестирование (installation testing) — тестирование, направленное на проверку успешной установки и настройки, обновления или удаления приложения.
     7. Тестирование интерфейса (GUI/UI testing) — проверка требований к пользовательскому интерфейсу.
     8. Тестирование удобства использования (usability testing) — это метод тестирования, направленный на установление степени удобства использования, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.
     9. Тестирование локализации (localization testing) — проверка адаптации программного обеспечения для определенной аудитории в соответствии с ее культурными особенностями.
     10. Тестирование безопасности (security testing) — это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.
     11. Тестирование надёжности (reliability testing) — один из видов нефункционального тестирования ПО, целью которого является проверка работоспособности приложения при длительном тестировании с ожидаемым уровнем нагрузки.
     12. Регрессионное тестирование (regression testing) — тестирование уже проверенной ранее функциональности после внесения изменений в код приложения, для уверенности в том, что эти изменения не внесли ошибки в областях, которые не подверглись изменениям.
     13. Повторное/подтверждающее тестирование (re-testing/confirmation testing) — тестирование, во время которого исполняются тестовые сценарии, выявившие ошибки во время последнего запуска, для подтверждения успешности исправления этих ошибок.

Источники

1. СИСТЕМНАЯ И ПРОГРАММНАЯ ИНЖЕНЕРИЯ / [Электронный ресурс] // Консорциум кодекс : [сайт]. — URL: <https://docs.cntd.ru/document/1200121069>

2. Фундаментальная теория тестирования / [Электронный ресурс] // Habr : [сайт]. — URL: <https://habr.com/ru/articles/549054/>

Вывод

Во время выполнения лабораторной работы №3 я познакомился и подробно изучил Unit-тесты, и научился пользоваться данными текстами, а также пользоваться библиотекой Boost test и самостоятельно реализовал свои тесты. Познакомился с сериализацией и десериализацией и реализовал их.

Ссылка на репозиторий: <https://github.com/Mixassss/2nd-3-laba.git>