ЛАБОРАТОРНАЯ РАБОТА №1

Курс «Технологии разработки программного обеспечения»

Тема: Составление спецификаций требований заказчика программного продукта.

Цель: Научиться формализовать требования заказчика программного продукта и оформлять техническое задание на создание программного продукта.

Вариант 3

«Информационная система кинотеатра»

Оглавление

[ОБЩИЕ СВЕДЕНИЯ 3](#_Toc87634141)

[НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ 3](#_Toc87634142)

[ХАРАКТЕРИСТИКА ОБЪЕКТОВ АВТОМАТИЗАЦИИ 3](#_Toc87634143)

[ТРЕБОВАНИЯ К СИСТЕМЕ 4](#_Toc87634144)

[Требования к структуре и функционированию системы 4](#_Toc87634145)

[Требования к квалификации персонала 5](#_Toc87634146)

[Требования к эргономике и технической эстетике 6](#_Toc87634147)

[Требования к функциям, выполняемым системой 6](#_Toc87634148)

[Требования к техническому обеспечению 6](#_Toc87634149)

[СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ 7](#_Toc87634150)

[ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ СИСТЕМЫ 8](#_Toc87634151)

[ИСТОЧНИКИ РАЗРАБОТКИ 8](#_Toc87634152)

# ОБЩИЕ СВЕДЕНИЯ

Информационная система кинотеатра. Работы создаются разработчиком в соответствии с календарным планом Проекта. По окончании каждого из этапов работ разработчик сдает заказчику соответствующие отчетные документы этапа.

# НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

Информационная система кинотеатра предназначена для повышения оперативности и работоспособности данного предприятия. Основное назначение системы автоматизация учетной и информационной деятельности в работе кинотеатра. В рамках рабочего проекта автоматизируется информационная деятельность:

- Учет финансовой деятельности;

- Учет рабочего состава;

- Анализ хозяйственной деятельности.

Основными целями создания информационной системы кинотеатра является:

- Оптимизация рабочего процесса;

- Уменьшение количества времени необходимого для учета покупки и пробития билетов;

- Перевод документооборота между вокзалами;

- Предоставление полной и гибкой системы отчетности.

# ХАРАКТЕРИСТИКА ОБЪЕКТОВ АВТОМАТИЗАЦИИ

|  |  |  |  |
| --- | --- | --- | --- |
| Структурное подразделение | Наименование процесса | Возможность автоматизации | Решение об автоматизации в ходе проекта |
| Билетные кассы | Продажа билетов | Возможна | Будет автоматизирован |

|  |  |  |  |
| --- | --- | --- | --- |
| Кинопоказ | Оператор использует аппаратуру для показа кино | Возможна | Не будет автоматизирован |
| Залы ожидания | Места для ожидания показа | Невозможна | Не будет автоматизирован |
| Буфет | Место куда можно прийти поесть или купить что-либо | Невозможна | Не будет автоматизирован |
| Места для просмотра | Места для просмотра кино | Невозможна | Не будет автоматизирован |
| Охрана | Охрана | Невозможна | Не будет автоматизирован |

# ТРЕБОВАНИЯ К СИСТЕМЕ

## Требования к структуре и функционированию системы

Информационная система кинотеатра должна быть централизованной, т.е. все данные должны располагаться в центральном хранилище. Система кинотеатра должна иметь трехуровневую архитектуру.

В системе предполагается выделить следующие функциональные подсистемы: Подсистема сбора, обработки и загрузки данных, которая предназначена для реализации процессов сбора данных из систем источников, приведения указания данных к виду, необходимому для наполнения подсистемы хранения данных. Подсистема хранения данных, которая предназначена для хранения данных в структурах, нацеленных на принятие решений. Подсистема формирования и визуализации отчетности, которая предназначена для формирования бизнес-ориентированных витрин и отчетности.

Смежными системами для кинотеатра являются:

− Информационные системы оперативной обработки данных

пользователя

− Информационные системы планирования

− Информационная система управления предприятием

− Информационно-справочная система

Взаимодействие со смежными системами может осуществляться с

помощью:

− Информационной системы управления предприятием с использованием

промежуточной базы данных.

Для обеспечения надежной работы информационной системы и ее компонентов необходимо подвергать систему диагностике. Диагностирование должно осуществляться программными средствами, входящими в комплект поставки ПО.

## Требования к квалификации персонала

От персонала, использующего информационную систему кинотеатра требуется следующие требования.

Администратор – знание предметной области, работы с БД и знание интерфейса. Навыки администрирования приложения.

Требования к сохранению работоспособности системы в различных вероятных условиях

В зависимости от различных вероятных условий система должна выполнять требования, приведенные в таблице.

|  |  |
| --- | --- |
| Вероятное условие | Требование |
| Нарушение в работе системы внешнего электроснабжения аппаратуры | Уведомление администрации о пропаже электричества |

|  |  |
| --- | --- |
| Выход из строя сервера подсистемы хранения данных | Уведомление администратора подсистемы хранения данных и администратора подсистемы сбора, обработки и загрузки данных. |
| Перепады напряжения во внешнем электроснабжении серверного оборудования величиной до 30% | Изменений нет, работа продолжается |

## Требования к эргономике и технической эстетике

Подсистема формирования и визуализации отчетности данных должна обеспечивать удобный для конечного пользователя интерфейс, отвечающий следующим требованиям:

− Реализация локализованного интерфейса

− Интерфейсы каждой подсистемы должны быть типизированы

− Часто используемые функции должны иметь “горячие клавиши”

## Требования к функциям, выполняемым системой

|  |  |
| --- | --- |
| Функция | Задача |
| Управляет процессами сбора, обработки и загрузки данных | Создание, редактирование и удаление процессов сбора, обработки и загрузки данных |
| Формирование последовательности выполнения процессов сбора, обработки и загрузки данных |
| Протоколирование результатов сбора данных | Извещение пользователей о внештатных ситуациях в системе |

## Требования к техническому обеспечению

Система должна быть реализована с использованием специально выделенных серверов. Сервер базы данных должен быть развернут на IBM Power 595 Server 9119-FHA. Сервер сбора, обработки и загрузки информации должен быть развернут на ARTLINE Business R27 v10.

# СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ

|  |  |
| --- | --- |
| Стадии | Этапы работ |
| Исследование и обоснование создания АС | Сбор и анализ данных автоматизированного объекта включая сбор сведений о зарубежных и отечественных аналогах |
| Разработка и оформление требований к системе |
| Техническое задание | Разработка аванпроекта |
| Разработка технического задания на АС |
| Эскизный проект | Разработка прототипов |
| Технический проект | Разработка окончательных решений |
| Разработка решений по организационному обеспечению |
| Разработка решений по техническому обеспечению |
| Разработка алгоритмов автоматизируемой деятельности |
| Разработка решений по информационному обеспечению |
| Разработка решений по лингвистическому обеспечению |
| Разработка решений по программному обеспечению |
| Разработка решений по методологическому обеспечению |
| Разработка проектно-сметной документации |
| Согласование решений по связям видов обеспечения |
| Составление заказной документации |

# ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ СИСТЕМЫ

При приеме заказчику передается инструкция по эксплуатации системы и проверяется функционирование автоматизированной информационной системы на реальных данных, подготовленных и введенных заказчиком по имеющейся инструкции, или, в случае неготовности заказчика, на контрольном примере. Готовый программный продукт подвергается многократному тестированию. Работы по сдаче проводятся на технических средствах разработчика. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие. Полученный программный продукт должен соответствовать разработанному ТЗ и ТП.

# ИСТОЧНИКИ РАЗРАБОТКИ

Настоящее техническое задание разработано на основе следующих документов и информационных материалов: − ГОСТ 24.701-86 “Надежность автоматизированных систем управления” − ГОСТ 34.602-89 “Информационная технология”

Контрольные вопросы:

1. Кратко опишите основные этапы разработки программного продукта.

Этап 1 – Определение проблемы

Перед тем как приступать к кодированию, необходимо четко сформулировать проблему, которую Ваша будущая программа должна решать. Так как, не имея хорошего определения проблемы, Вы можете потратить много усилий и времени на решение не той проблемы, которую требуется решить.

На данном этапе проводится простая формулировка сути проблемы без каких-либо намеков на ее возможные решения, при этом формулировать ее следует на языке, понятном пользователю, т.е. она должна быть описана с пользовательской точки зрения.

1. Что представляет собой артефакт программного продукта?

Артефакт — это любой созданный искусственно элемент программной системы.

К элементам программной системы, а, следовательно, и к артефактам, могут относиться исполняемые файлы, исходные тексты, веб страницы, справочные файлы, сопроводительные документы, файлы с данными, модели и многое другое, являющееся физическим носителем информации. Другими словами, артефактами являются те информационные элементы, которые тем или иным способом используются при работе программной системы и входят в ее состав.

1. Опишите основные стратегии разработки ПО.

Основные модели разработки ПО:

* Каскадная модель, или «водопад»;
* V-образная модель, разработка через тестирование;
* Инкрементная модель;
* Итеративная (или итерационная) модель;
* Спиральная модель.

Каскадная модель

В этой модели разработка осуществляется поэтапно: каждая следующая стадия начинается только после того, как заканчивается предыдущая. Если всё делать правильно, «водопад» будет наиболее быстрой и простой моделью. Применяется уже почти полвека, с 1970-х.

«Водопад» подходит для разработки проектов в медицинской и космической отрасли, где уже сформирована обширная база документов (СНиПов и спецификаций), на основе которых можно написать требования к новому ПО.

При работе с каскадной моделью основная задача — написать подробные требования к разработке. На этапе тестирования не должно выясниться, что в них есть ошибка, которая влияет на весь продукт.

V-образная модель

Это усовершенствованная каскадная модель, в которой заказчик с командой программистов одновременно составляют требования к системе и описывают, как будут тестировать её на каждом этапе. История этой модели начинается в 1980-х.

V-модель подходит для проектов, в которых важна надёжность и цена ошибки очень высока. Например, при разработке подушек безопасности для автомобилей или систем наблюдения за пациентами в клиниках.

Инкрементная модель

Это модель разработки по частям (increment в переводе с англ. — приращение) уходит корнями в 1930-е. Рассмотрим её на примере создания социальной сети.

1. Заказчик решил, что хочет запустить соцсеть, и написал подробное техническое задание. Программисты предложили реализовать основные функции — страницу с личной информацией и чат. А затем протестировать на пользователях, «взлетит или нет».
2. Команда разработки показывает продукт заказчику и выпускает его на рынок. Если и заказчику, и пользователям социальная сеть нравится, работа над ней продолжается, но уже по частям.
3. Программисты параллельно создают функциональность для загрузки фотографий, обмена документами, прослушивания музыки и других действий, согласованных с заказчиком. Инкремент за инкрементом они совершенствуют продукт, приближаясь к описанному в техническом задании.

Инкрементная модель подходит для проектов, в которых точное техзадание прописано уже на старте, а продукт должен быстро выйти на рынок.

Итеративная модель

Это модель, при которой заказчик не обязан понимать, какой продукт хочет получить в итоге, и может не прописывать сразу подробное техзадание.

Итеративная модель подходит для работы над большими проектами с неопределёнными требованиями, либо для задач с инновационным подходом, когда заказчик не уверен в результате.

Спиральная модель

Используя эту модель, заказчик и команда разработчиков серьёзно анализируют риски проекта и выполняют его итерациями. Последующая стадия основывается на предыдущей, а в конце каждого витка — цикла итераций — принимается решение, продолжать ли проект. Эту модель начали использовать в 1988 году.

Спиральная модель похожа на инкрементную, но здесь гораздо больше времени уделяется оценке рисков. С каждым новым витком спирали процесс усложняется. Эта модель часто используется в исследовательских проектах и там, где высоки риски.

1. Кратко охарактеризуйте системы принципов разработки ПО (PSP, TSP, CMM).

PSP/TSP

Одна из последних разработок Института программной инженерии Personal Software Process / Team Software Process. Personal Software Process определяет требования к компетенциям разработчика. Согласно этой модели, каждый программист должен уметь:

- Учитывать время, затраченное на работу над проектом;

- Учитывать найденные дефекты;

- Классифицировать типы дефектов;

- Оценивать размер задачи;

- Осуществлять систематический подход к описанию результатов тестирования;

- Планировать программные задачи;

- Распределять их по времени и составлять график работы.

- Выполнять индивидуальную проверку проекта и архитектуры;

- Осуществлять индивидуальную проверку кода;

- Выполнять регрессионное тестирование.

Team Software Process делает ставку на самоуправляемые команды численностью 3-20 разработчиков. Команды должны:

- Установить собственные цели;

- Составить свой процесс и планы;

- Отслеживать работу;

- Поддерживать мотивацию и максимальную производительность.

Последовательное применение модели PSP/TSP позволяет сделать нормой в организации пятый уровень CMM.

SMM стратегия

Самым именитым стандартом качества следует считать Capability Maturity Model (CMM) – модель оценки уровня зрелости процессов разработки вместе с его производными. Он был создан SEI (Software Engineering Institute), который финансируется за счет Министерства обороны США и является структурной единицей Университета Карнеги-Меллона. Первая официальная версия стандарта вышла в 1993 г., хотя работы над ним начались гораздо раньше – основные его положения были опубликованы еще в 1986 г.

Успех CMM предопределило несколько факторов. Этот стандарт был одним из первых, изначально учитывающих специфику создания ПО. Он оказался достаточно прост и прозрачен как для понимания, так и для использования, и регламентировал, «что», а не «как» делать, а потому подходил для различных моделей жизненного цикла, методологий разработки и не накладывал каких-либо ограничений на стандарты документирования, инструментарий, среду и языки, применяемые в проектах. И, пожалуй, основным фактором, предопределившим популярность CMM, явилось сотрудничество SEI с Министерством обороны США, что де-факто означало использование стандарта при реализации проектов по заказу этого ведомства.

1. Состав персонала разработки программного продукта.

Администратор

1. Укажите основные метрики качества программного продукта.

Метрики программного обеспечения можно разделить на три категории

Метрики продукта — описывает характеристики продукта, такие как размер, сложность, особенности дизайна, производительность и уровень качества.

Метрики процесса — эти характеристики могут использоваться для улучшения деятельности по разработке и сопровождению программного обеспечения.

Метрики проекта — эти метрики описывают характеристики и исполнение проекта. Примеры включают число разработчиков программного обеспечения, штатное расписание в течение жизненного цикла программного обеспечения, стоимость, график и производительность.

Метрики качества программного обеспечения можно разделить на три категории:

- Метрики качества продукции;

- Показатели качества в процессе;

- Метрики качества обслуживания.

1. В чем заключается процесс инспектирования проекта? Укажите принципы инспектирования.

Инспектирование программ – это просмотр и проверка программ с целью обнаружения в них ошибок.

Для начала процесса инспектирования программы необходимы следующие условия: наличие точной спецификации кода (без полной спецификации невозможно обнаружить дефекты в проверяемом программном компоненте); члены инспекционной группы должны хорошо знать стандарты разработки; в распоряжении группы должна быть синтаксически корректная последняя версия программы (нет смысла рассматривать код, который «почти

завершен»).

Сам процесс инспектирования должен быть относительно коротким (не более двух часов) и сосредоточенным только на выявлении дефектов, аномалий и несоответствий стандартам. Инспекционная группа не должна предлагать способы исправления дефектов или рекомендовать какие-либо изменения в других программных компонентах.

После инспектирования автор изменяет программу, исправляя обнаруженные ошибки. На этапе доработки координатор принимает решение о том, необходимо ли повторное инспектирование. Если повторное инспектирование не требуется, все обнаруженные дефекты фиксируются документально.

1. Приведите примеры функциональных и нефункциональных требований к программному продукту.

При разработке программного обеспечения функциональные требования определяют функции, которые должно выполнять все приложение или только один из его компонентов. Функция состоит из трех шагов: ввод данных — поведение системы — вывод данных. Он может вычислять, манипулировать данными, выполнять бизнес-процессы, устанавливать взаимодействие с пользователем или выполнять любые другие задачи.

Другими словами, функциональное требование — это то, ЧТО приложение должно или не должно делать после ввода некоторых данных.

Функциональные требования важны, поскольку они показывают разработчикам программного обеспечения, как должна вести себя система. Если система не соответствует функциональным требованиям, значит, она не работает должным образом.

Нефункциональные требования определяют стандарты производительности и атрибуты качества программного обеспечения, например, удобство использования системы, эффективность, безопасность, масштабируемость и т.д.

В то время как функциональные требования определяют, что делает система, нефункциональные требования описывают, КАК система это делает. Например, веб-приложение должно обрабатывать более 15 миллионов пользователей без какого-либо снижения производительности, или веб-сайт не должен загружаться более 3 секунд.

Если приложение не соответствует нефункциональным требованиям, оно продолжает выполнять свои основные функции, однако не сможет обеспечить удобство для пользователя.

Нефункциональные требования важны, поскольку они помогают разработчикам программного обеспечения определять возможности и ограничения системы, которые необходимы для разработки высококачественного программного обеспечения. Следовательно, нефункциональные требования так же важны, как и функциональные требования для успешного внедрения продукта.

1. Что представляют собой варианты использования программного продукта?

Варианты использования это - описание последовательности действий, которые может осуществлять система в ответ на внешние воздействия пользователей или других программных систем. Варианты использования отражают функциональность системы с точки зрения получения значимого результата для пользователя, поэтому они точнее позволяют ранжировать функции по значимости получаемого результата.

1. Приведите схему процесса анализа С-требований.

Требования заказчика к системе фиксируются разработчиками посредством проведения, специально организованного опроса-интервью. Но перед этим необходимо идентифицировать пользователей системы, т.е. указать категории лиц, которые должны или гипотетически могут в настоящий

момент или в будущем воспользоваться разрабатываемой системой. В зависимости от масштабов системы эта задача может оказаться не тривиальной, поэтому требует особого внимания, поскольку при создании системы необходимо учесть требования всех заинтересованных лиц.

1. Приведите схему процесса анализа D-требований.

Разработчикам программного обеспечения нужна база для проектирования и разработки. Эта база представляет собой детальные требования (D-требования). D-требования состоят из полного списка конкретных свойств и функциональности, которую должна иметь программа. Каждое из этих требований пронумеровано, помечено и отслеживается по ходу разработки. D-требования должны быть согласованы с С-требованиями.