Министерство науки и высшего образования РФ

КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

МИППС о

Контрольная работа №1 вариант 6 о

По методологии проектирования, разработке и внедрению

информационных систем о

Студент 2 курса, шифр 19-ЗММин-176

Направление 09.04.04 программная инженерия о

Фамилия Суворов о

Имя Даниил Отчество Александрович о

Дата поступления работы

Оценка

Рецензент Попова Ольга Борисовна

«\_\_\_\_\_\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись



**Содержание**

[Введение 3](#_Toc61950164)

[1 Методология функционального моделирования 4](#_Toc61950165)

[2 Методология объектно-ориентированного моделирования 7](#_Toc61950166)

[3 Методология управление проектами 13](#_Toc61950167)

[4 Транзакции запросы и операции 20](#_Toc61950168)

[5 Использование систем контроля версий исходного кода программ 23](#_Toc61950169)

[6 Планирование работ по выполнению проекта «Разработка и внедрение программного 26](#_Toc61950170)

[7 Оценка трудозатрат на выполнение работ по разработке и внедрению программного обеспечения 29](#_Toc61950171)

[8 Автоматизация управления проектом по разработке и внедрению автоматизированной информационной системы 31](#_Toc61950172)

# Введение

В данной контрольной работе представлена информационная система «**Игровой сайт**». Игровой сайт – это сайт, который регулярно пополняется новостями об игровом мире, его новинках, обзорах и прохождениями игр. Другими словами это: записи, новости, мультимедиа, изображения. Новости располагаются в обратном хронологическом порядке: сначала самые новые, дальше – более поздние. Быть посетителем данного ресурса интересно для того, кто интересуется играми. Там можно расслабиться и найти все, что нужно по любой игре. Конкретными целями информационной системы «Игровой сайт» является:

* Оповещение пользователей новинками игровой индустрии.
* Обучение пользованием игровыми ресурсами.
* Для бизнеса. Еще один способ связаться и проинформировать целевую аудиторию.
* Развлечения. На сайте имеются разделы, посвященные юмору.
* Образование. Видео-уроки, посвященные установке, прохождению игр.
* Реклама. Сотрудничество с компаниями создателей игр и другими компаниями, связанными с игровой индустрией: аудиотехника, компьютерная техника.

# 1 Методология функционального моделирования

**Цель работы**

Изучить методологии функционального моделирования IDEF0 и IDEF3. Лабораторная работа направлена на ознакомление с методологиями функционального моделирования IDEF0 и IDEF3, получение навыков по применению данных методологий для построения функциональных моделей на основании требований к информационной системе.

**Введение**

IDEF0 — методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. IDEF3 — это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе.

**Основная часть**

В качестве функциональной модели проекта «Игровой сайт» представлены диаграммы нотации IDEF0 (Рисунки 1-2) и диаграмма IDEF3 (Рисунок 3).

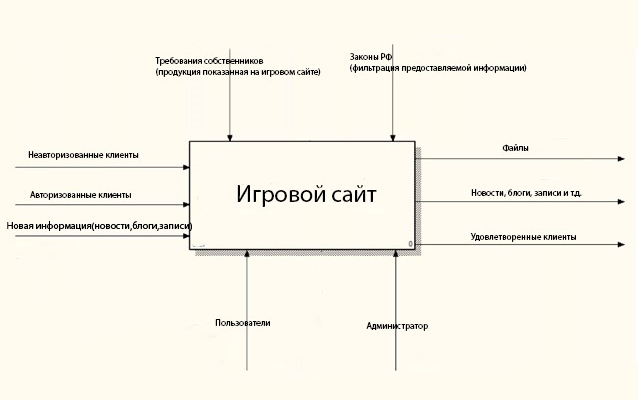


Рисунок 1 - Контекстная диаграмма

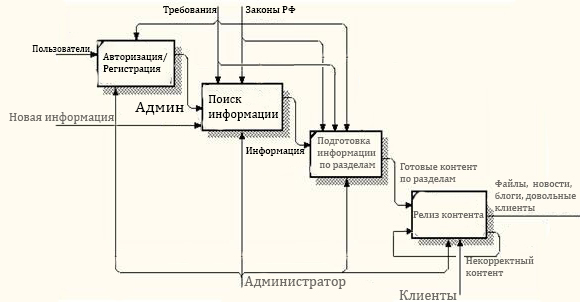


Рисунок 2 - Диаграмма IDEF0 первого уровня декомпозиции

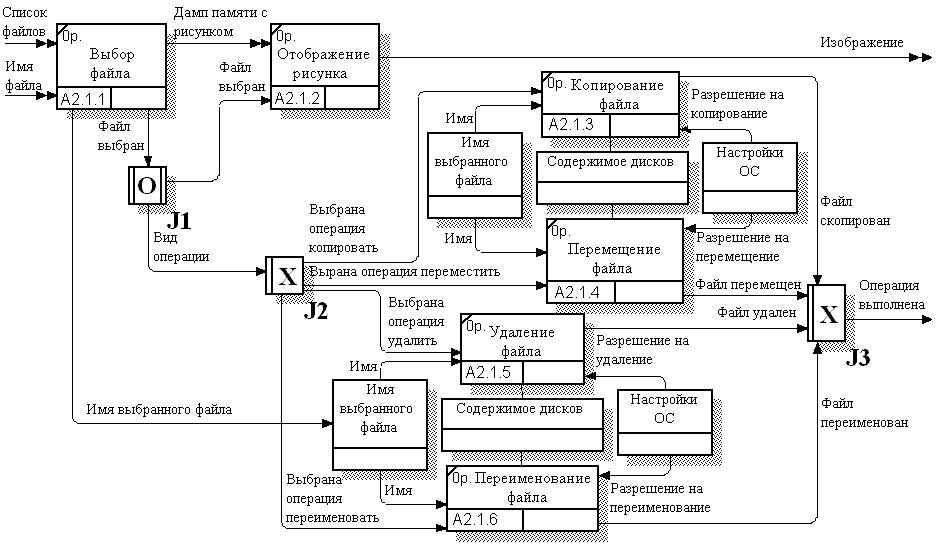


Рисунок 3 - Модель IDEF3 при работе с файлом на игровом сайта

# 2 Методология объектно-ориентированного моделирования

**Цель работы**

Ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML. Лабораторная работа направлена на ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML, получение навыков по применению данных элементов для построения объектно-ориентированных моделей ИС на основании требований.

**Введение**

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

**Диаграмма прецедентов (вариантов использования).**

Основные элементы диаграммы - участник (actor) и прецедент (вариант). Участник — это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Участником может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности. Графически участник изображается “человечком”. Прецедент (use case) - описание множества последовательных событий (включая варианты), выполняемых системой, которые приводят к наблюдаемому участником результату. Прецедент представляет поведение сущности, описывая взаимодействие между участниками и системой. Прецедент не показывает, “как” достигается некоторый результат, а только “что” именно выполняется. Прецеденты обозначаются очень простым образом - в виде эллипса, внутри которого указано его название.

**Диаграммы последовательностей**

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов). Сообщения (вызовы методов) - линиями со стрелками. Возвращаемые результаты - пунктирными линиями со стрелками. Прямоугольники на вертикальных линиях под каждым из объектов показывают “время жизни” (фокус) объектов. Впрочем, довольно часто их не изображают на диаграмме, все это зависит от индивидуального стиля проектирования.

**Диаграмма коопераций**

Диаграмма кооперации предназначена для описания поведения системы на уровне отдельных объектов, которые обмениваются между собой сообщениями, чтобы достичь нужной цели или реализовать некоторый вариант использования. Такое представление структуры модели как совокупности взаимодействующих объектов и обеспечивает диаграмма кооперации. На диаграмме кооперации размещаются объекты, представляющие собой экземпляры классов, связи между ними, которые в свою очередь являются экземплярами ассоциаций и сообщения. Связи дополняются стрелками сообщений, при этом показываются только те объекты, которые участвуют в реализации моделируемой кооперации. Далее, как и на диаграмме классов, показываются структурные отношения между объектами в виде различных соединительных линий. Связи могут дополняться именами ролей, которые играют объекты в данной взаимосвязи. И, наконец, изображаются динамические взаимосвязи — потоки сообщений в форме стрелок с указанием направления рядом с соединительными линиями между объектами, при этом задаются имена сообщений и их порядковые номера в общей последовательности сообщений.

**Диаграмма классов**

Диаграмма классов описывает типы объектов системы и различного рода статические отношения, которые существуют между ними. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами.

**Диаграмма состояний**

Она показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

**Основная часть**

**Диаграмма прецедентов (вариантов использования)**

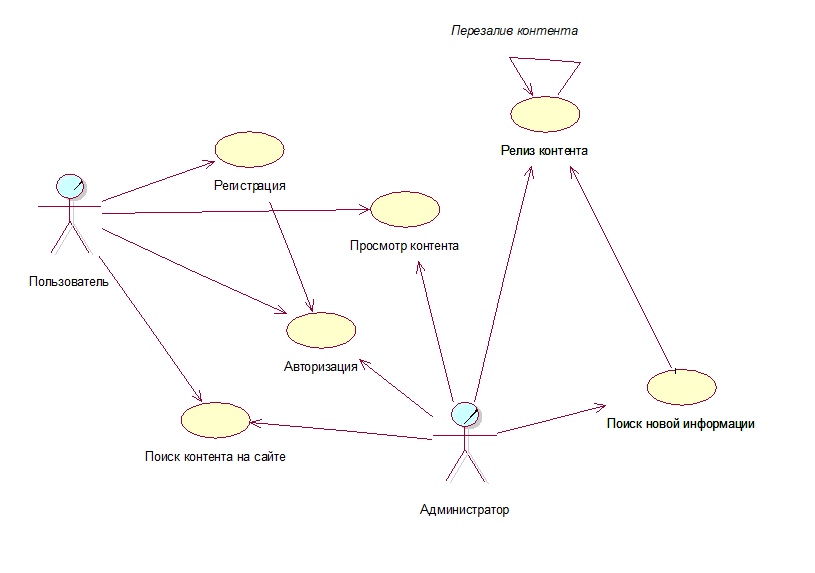


Рисунок 1 - Диаграмма прецедентов

**Диаграммы последовательностей**

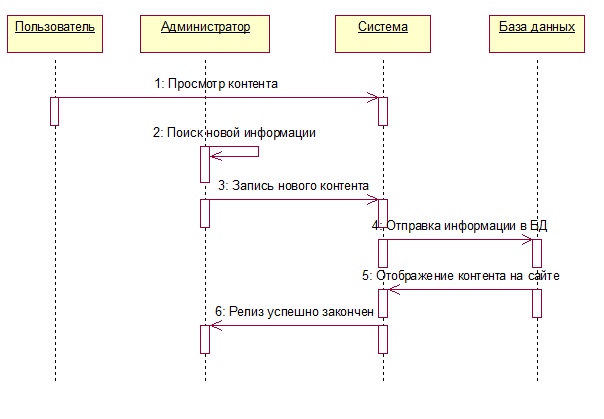


Рисунок 2 - Диаграмма последовательности работы сайта

**Диаграмма коопераций**

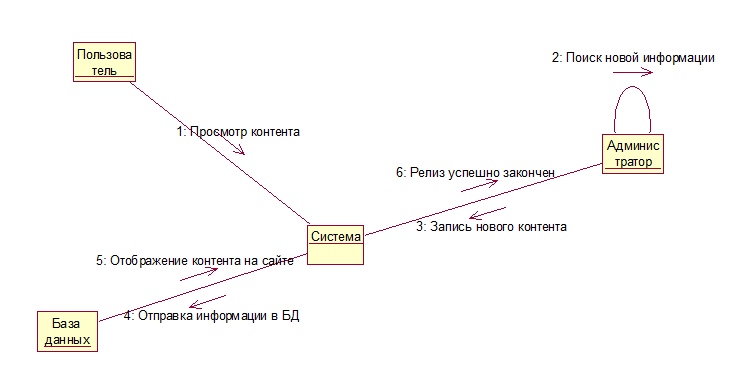


Рисунок 3 - Диаграмма коопераций

**Диаграммы классов**

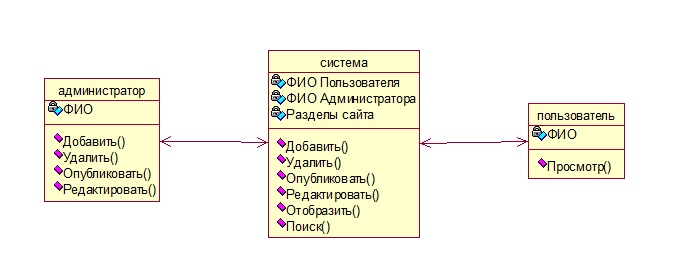


Рисунок 4 - Диаграмма классов

**Диаграмма состояний**

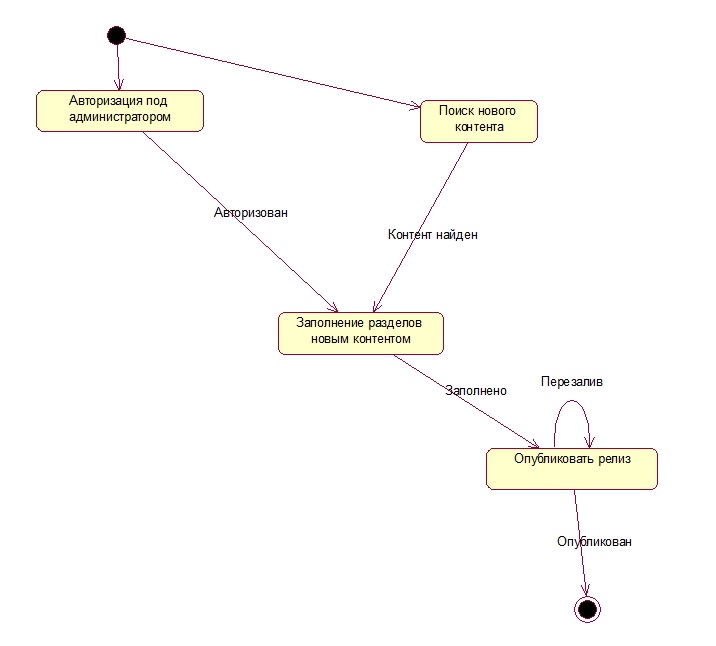


Рисунок 5 - Диаграмма состояний

В результате лабораторного практикума для проекта «Игровой сайт» были построены следующие диаграммы:

* диаграмма прецедентов;
* диаграмма последовательности;
* диаграмма коопераций;
* диаграмма состояний;
* диаграмма классов.

# 3 Методология управление проектами

**Цель работы**

Изучение методологии управления проектами. Получение навыков по применению данных методологий для планирования проекта. Лабораторная работа направлена на ознакомление с основными понятиями методологии управления проектами, получение навыков по применению данных понятий при построении плана проекта, построения графика работ, распределения исполнителей, управления рисками.

**Введение**

Если для создания сетевой диаграммы используются программные средства поддержки управления проектом, каждый этап должен заканчиваться контрольной отметкой. Очередной этап может начаться только тогда, когда будет получена контрольная отметка (которая может зависеть от нескольких предшествующих этапов).

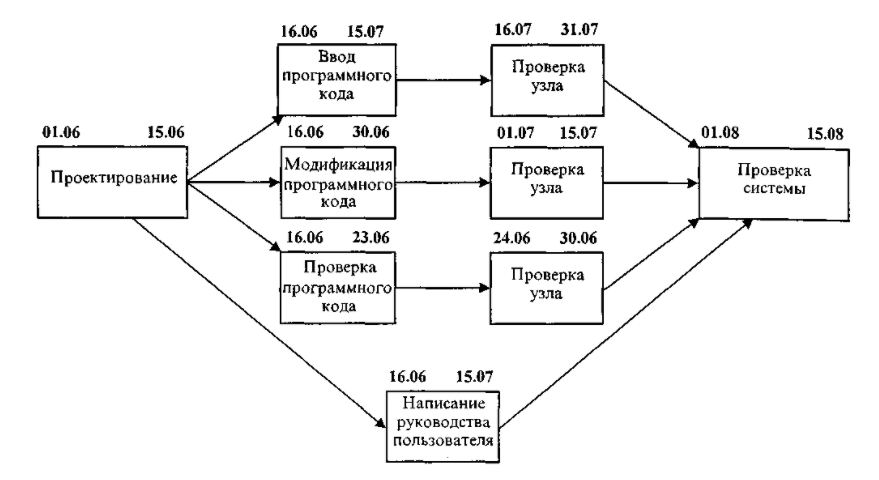


Рисунок 1 - Сетевая диаграмма

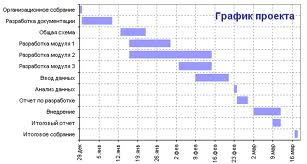


Рисунок 2 - Временная диаграмма длительности этапов

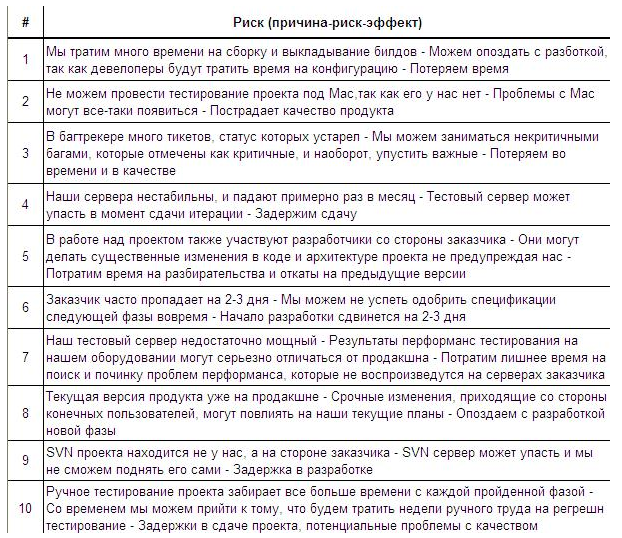
**Список возможных рисков с указанием названия риска, его описание и типа**

Цель этого этапа — выявить некоторое количество неизвестных рисков проекта. Полагаем, что потенциальных проблем вокруг нас бесконечно много, поэтому задачу будем ставить количественно. В начале проекта, неплохо идентифицировать 50–100 рисков, в дальнейшем — по 20–30 штук.

На входе: план проекта, текущий список рисков (если есть).

Процесс: ПМ собирает митинг со всей командой, сообщает о его цель, длительность и agenda; ПМ сообщает о статусе проекта, об основных текущих рисках и проблемах, отвечает на вопросы; Участники митинга озвучивают потенциальные риски. Принимаются все идеи без исключения, без обсуждений и комментариев; ПМ записывает результаты в формате «причина-риск-эффект». Как только цель достигнута, либо время истекло, митинг завершается;

На выходе: обновленный список рисков в формате «причина-риск-эффект».



**Анализ рисков**

Очевидно, что бороться со всеми рисками сразу дорого и малоэффективно. Цель этого этапа — выявить наиболее важные из них. Для каждого риска оценим его Вероятность и Последствия по десятибалльной шкале. Перемножив их, получим Важность. Обозначим также некоторую границу Важности (например, 50), чтобы понять какие риски критичны и далее работать только с ними.

На входе: список рисков.

Процесс: ПМ собирает митинг с тим лидами, сообщает о его цель, длительность и agenda; ПМ оглашает риск, участники митинга оценивают его вероятность и последствия; ПМ записывает оценки, как только цель митинга достигнута, либо время истекло, митинг завершается; ПМ считает важность рисков как Вероятность\*Последствия, сортирует список по убыванию Важности; ПМ обозначает риски, превысившие границу Важности в списке;

На выходе: список критичных рисков;



**Планирование рисков**

Фактически, на этом этапе и происходит управление проектом. Для каждого риска, из списка критичных, необходимо придумать стратегию, которая наш проект от него обезопасит. Всего стратегий используется три:

Transfer. Переносим ответственность за последствия риска на третью сторону (заказчика, компанию партнера, страховую компанию и так далее). Применять эту стратегию есть смысл, если сами мы не можем повлиять на риск и есть на кого эту ответственность переложить.

Accept. Принимаем ответственность за последствия риска на себя, но ничего не делаем, оставляем все как есть. Применять этот подход есть смысл только когда с риском мы поделать ничего не можем, а делать трансфер на третью сторону неоправданно дорого.

Mitigate. Боремся с риском, принимая ответственность на него на себя. Для борьбы с риском хорошо иметь несколько планов. Основной для того, чтобы риск подавить, и отходной, на случай если риск все-таки случился и влияет на проект: Основной план необходимо внедрять сразу, до того, как риск случился. Он должен понижать либо Вероятность, либо Последствия риска. Тут нам поможет запись рисков в формате «причина-риск-эффект». Чтобы понизить Вероятность риска, нужно бороться с его причиной. Чтобы побороть Последствия, нужно защищать предмет его воздействия. Отходной план внедряется в случае, если меры по борье с риском не принесли результатов, риск случился и стал проблемой.

На входе: список критичных рисков.

Процесс: ПМ собирает митинг с руководителями других проектов, сообщает о его цель, длительность и agenda; ПМ оглашает риск, участники митинга определяют стратегию работы с ним, основной план и запасной план (для Mitigate); ПМ записывает планы в риск лист, как только цель митинга достигнута, либо время истекло, митинг завершается; ПМ обновляет план проекта, добавляя основные планы по рискам;

На выходе: список критичных рисков со стратегией и планом на каждый риск, обновленный план проекта.



**Мониторинг и контроль**

Это скорее процесс, чем этап. Его цель — поддерживать список рисков и план проекта в актуальном состоянии.

На входе: спланированный список рисков, план проекта, ежедневные отчеты команды;

Процесс: ПМ выполняет ревизию списка рисков, обновляет оценки, обновляет устаревшие планы; ПМ выявляет случившиеся риски, принимает решение о внедрении отходных планов, обновляет план проекта;

На выходе: обновленный список рисков, обновленный план проекта;



# 4 Транзакции запросы и операции

**Цель работы**

Разработать тесты для приведенных классов, провести регрессионное тестирование, выполнить профилирование программы.

**Введение**

Тестирование по принципу «черного ящика» При тестировании отдельных модулей как «черных ящиков» известны функции программы, и при этом исследуется работа каждой функции. Структура программы или модуля (в зависимости от того, что является объектом тестирования) неизвестна. Тесты представляют собой набор входящих параметров с имеющимися значениями и набор выходящих параметров с ожидаемыми значениями. Если фактически полученные параметры соответствуют ожидаемым, тест пройден успешно. Тестирование по принципу «черного ящика» обеспечивает поиск следующих категорий ошибок:

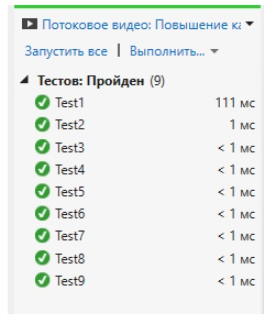
некорректные или отсутствующие функции; ошибки интерфейса; ошибки во внешних структурах данных или в доступе к внешней базе данных; ошибки характеристик (требования к аппаратному обеспечению и т. п.); ошибки инициализации или завершения.

Тестирование по принципу «белого ящика» При тестировании отдельных модулей как «белых ящиков» известны внутренние состав и структура модулей, а исследуются взаимосвязи между элементами программы. Проверяется корректность построения всех элементов программы и правильность их взаимодействия друг с другом. Обычно анализируются управляющие связи элементов, реже — информационные связи. Тестирование по принципу «белого ящика» характеризуется степенью соответствия выполняемых тестов формируемой логике программы. Исчерпывающее тестирование, как и в случае тестирования по принципу «черного ящика», затруднительно. Тестирование по принципу «белого ящика», как правило, используют при разработке программ с повышенными требованиями к надежности (т. е. когда сбой в программе может привести к смерти человека или какой-либо катастрофе). Кроме того, тестирование «белого ящика» используют при тестировании наиболее критических элементов системы, от работоспособности которых зависит правильность работы всех остальных элементов, и для локализации ошибок, выявленных при тестировании по принципу «черного ящика». В остальных случаях исчерпывающее тестирование «белого ящика» проводят редко.

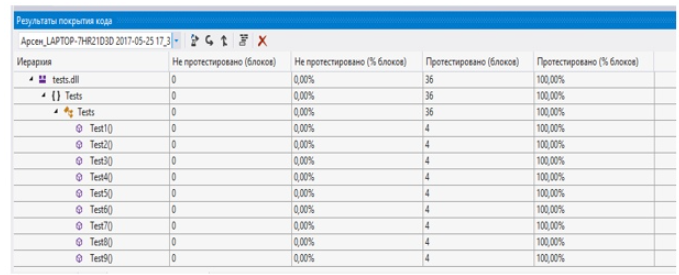
**Основная часть**

Тестовыми вариантами подходят такие тесты, которые проверяют корректность открытия каждого раздела игрового сайта. Все тесты выполнены успешно.

**Результаты тестирования программы**



**Оценка покрытия кода тестами**



**Код тестирования**

Часть кода будет описано ниже:

string ActualResult; //текущая страница

string ExpectedResult = "Главная страница";

string ExpectedResult1 = "Новости";

driver.Navigate().GoToUrl("http://igromir.ru/main");

ActualResult = driver.PageSource;

if (ActualResult.Contains(ExpectedResult))

{

driver.FindElement(By.LinkText("Новости")).Click();

ActualResult = driver.PageSource;

if (ActualResult.Contains(ExpectedResult1))

{

}

else

{ Assert.IsTrue(false, "Главная страница не открывается "); }

}

else

{

Assert.IsTrue(false, "Новости не открываются");

}

driver.Quit();

}

# 5 Использование систем контроля версий исходного кода программ

**Цель работы**

Создать в системе контроля версий (GIT) репозиторий для нового проекта и выполнить все основные действия с исходным кодом программы, связанные с контролем версий.

**Введение**

Архитектура трех деревьев Система контроля версий git имеет архитектуру трех деревьев. Перед тем как перейти к ее описанию, для начала, рассмотрим архитектуру двух деревьев. Для начала введем используемую в системах контроля версий терминологию. Набор файлов, с которым мы работаем в данный момент, называется рабочая копия (working copy). После того, как решено, что все нужные изменения на данный момент внесены, и об этом можно сообщить системе контроля версий, разработчик производит отправку изменений в репозиторий (repository). Репозиторий – это хранилище для нашего проекта, которое обслуживает система контроля версий. Сама операция отправки изменений называется commit. Если нам необходимо взять данные из репозитория, то мы осуществляем операцию checkout. Для названий операций commit и checkout не используют прямой перевод, предпочитают транскрипцию. Для архитектуры двух деревьев регламент работы с репозиторием может выглядеть следующим образом:

Перед началом работы разработчик делает checkout, для того чтобы быть уверенным, что он будет работать с актуальной рабочей копией. Разработчик вносит необходимые изменения в исходный код. Разработчик отправляет изменения в репозиторий (коммитит их). Повторить необходимое количество раз пункты 2 и 3. Система контроля версий git использует архитектуру трех деревьев. Суть ее заключается в том, что дополнительно добавляется ещё одно место, которое можно назвать кэшем или stage в английской терминологии. Рабочая копия и репозиторий идейно не отличается от их аналогов в архитектуре двух деревьев. Наличие дополнительного элемента меняет регламент работы, которой в этом случае выглядит так:

Перед началом работы разработчик делает checkout, для того чтобы быть уверенным, что он будет работать с актуальной рабочей копией. Разработчик вносит необходимые изменения в исходный код. Разработчик отправляет необходимый набор файлов, изменения в которые внесены, в stage для того, чтобы потом построить из них коммит. До того, как изменения будут отправлены в репозиторий, разработчик может добавлять и удалять файлы из stage. Набор файлов в stage, как правило, идеологически связан между собой. Разработчик отправляет изменения в репозиторий (коммитит их). Повторить необходимое количество раз пункты 2 – 4. Наличие stage добавляет гибкости в процесс разработки, вы можете внести изменения в довольно большое количество файлов, но отправить их в репозиторий в разных коммитах со своими специфическими комментариями.

**Основная часть**

Для работы с GIT были использованы следующие команды:

Создание пустого репозитория.

> git init

Добавление файлов в stage.

> git add filename

Создание коммита.

> git commit -m “message”

Просмотр статуса каталога.

> git status

Просмотр коммитов в репозитории с сокращенным выводом информации.

> git log --oneline

git remote add — команда, которой устанавливается подключение к удаленному серверу и git репозиторию на нем (Добавляем ссылку (кот была скопирована ранее) на папку на гитхабе)

> git remote add origin https://github.com/DarkMonsterDan/lab-/wiki

Установление связи между той веткой, в которой вы находитесь и веткой master на удалённом сервере.

> git push -u origin master

Загрузка изменений на сервер Гитхаб:

> git push

Скачивание всех изменений с облака, которые были произведены в репозитории (для работы с актуальной/последней версией проекта):

> git pull

В ходе лабораторного практикума был создан репозиторий и загружены изменения на сервер Github.

# 6 Планирование работ по выполнению проекта «Разработка и внедрение программного

**Цель работы**

На основе технического задания на разработку и внедрение автоматизированной информационной системы сформировать календарный план выполнения работ по проекту.

**Введение**

В качестве исходных данных выступает техническое задание на разработку и внедрение автоматизированной информационной системы, перечень доступных для использования трудовых ресурсов. Для проведения успешного проекта нужно оценить объем предстоящих работ, возможный риск, требуемые ресурсы, предстоящие задачи, определить контрольные точки, стоимость и план работ, которому желательно следовать. Процесс руководство программным проектом включает решение вышеперечисленных задач. Этот процесс начинается перед технической работой, продолжается по мере развития ПО от идеи к реальности и достигает наибольшей интенсивности к концу работ. Основной задачей при планировании является определение структуры распределения работ WBS (Work Breakdown Structure) с помощью средств планирования работ (например, MS Project).

**Основная часть**

План выполнения работ составляется на основе декомпозиции проекта вплоть до постановки элементарных задач, которые могут быть выяснены по результатам предварительного анализа. При этом возможно применение содержательных моделей системного анализа. Например, использование модели декомпозиции типа «жизненный цикл» позволит разбивать отдельные задачи на подзадачи путем определения последовательности действий. Процесс декомпозиции будет определяться принятой моделью жизненного цикла разработки программного обеспечения.

Для каждой элементарной задачи должны быть определены:

* ресурсы, необходимые для решения задачи (в том числе трудовые);
* объем работ, выраженный в принятой системе метрик;
* стоимость работ (может быть вычислена на основе объема работ и стоимости привлекаемых
* ресурсов);
* длительность работ (может быть вычислена на основе объема работ, количества привлекаемых рудовых ресурсов и принятых нормативов производительности). Между отдельными элементарными задачами могут быть определенные зависимости, заключающиеся в том, что одни задачи могут выполняться параллельно, другие — в строгой последовательности (для выполнения одних задач могут требоваться результаты выполнения других).

**Календарный план**

Ниже приведен календарный план проекта. От получения данных до ее сопровождения.

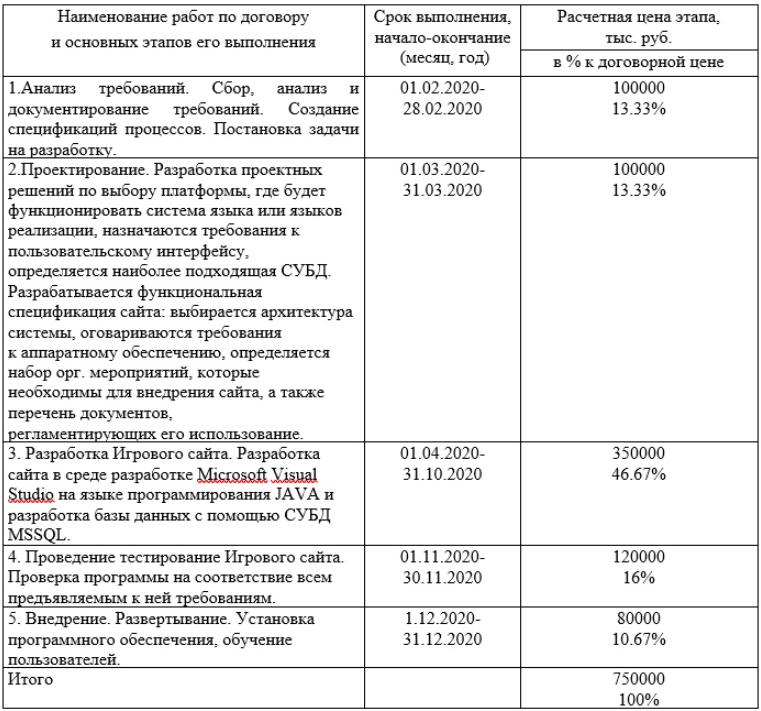


Рисунок 1 – Календарный план проекта “Игровой сайт”

# 7 Оценка трудозатрат на выполнение работ по разработке и внедрению программного обеспечения

**Цель работы**

Расчет трудоемкости и стоимости выполнения работ по проекту, для которого был составлен календарный план в процессе выполнения лабораторной работы № 1. Расчет трудоемкости следует проводить методов функциональных точек.

**Введение**

Функционально-ориентированные метрики косвенно измеряют программный продукт и процесс его разработки. Вместо подсчета LOC-оценки при этом рассматривается не размер, а функциональность или полезность продукта. В качестве количественной характеристики применяется понятие количества функциональных точек FP (function points). Средства реализации: ограничений не накладывается, рекомендуется использование All Fusion Process Modeler, All Fusion Data Modeler, Microsoft Excel, Microsoft Project.

**Основная часть**

Проведена экспертиза проекта в средстве Microsoft Project. Расчет произведен по себестоимости сотрудников (аналитиков, разработчиков, тестировщиков) соответственно, также определено количество Рабочих дней, отведенных на каждый этап. Результат представлен на рисунке 1.

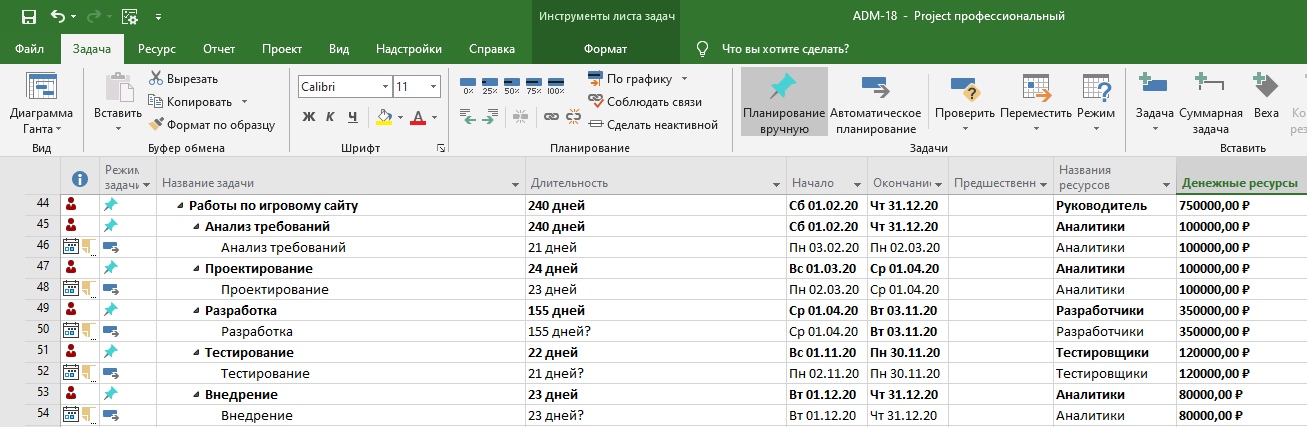


Рисунок 1 – Экспертиза сайта “Игровой сайт” в Microsoft Project

# 8 Автоматизация управления проектом по разработке и внедрению автоматизированной информационной системы

**Цель работы**

Сформировать в Issue Tracking System план работ, имитируя выполнение отдельных задач, выполнить мониторинг работ, сформировать отчет о ходе выполнения проекта.

**Введение**

Система управления проектами (СУП) - набор инструментов, методов, методологий, ресурсов и процедур, используемых для управления проектом. Она может быть как формальной, так и неформальной и помогает менеджеру проекта эффективно завершить проект. Система управления проектами — это ряд процессов и связанных с ними функций контроля, объединенных в единую целенаправленную структуру. Система управления проектами строится на основе плана управления проектом, который описывает то, как будет использоваться система. Содержание системы управления проектом изменяется в зависимости от области приложения, особенностей организации, сложности проекта и доступности необходимых ресурсов. Система строится так, чтобы максимально соответствовать стратегическим целям и производственным ресурсам клиентской организации. При разработке программного обеспечения как большие, так и маленькие софтверные компании используют системы учета задач, ошибок, управления проектами (bug tracker, issue tracking system, project management application). На данный момент такого рода продуктов существует немало. Есть простые системы, функционал которых ограничивается учетом ошибок и отслеживанием их статуса. Есть более сложные, которые позволяют, например, строить различные графики по проектным рискам, интегрироваться с системами версионного контроля, осуществлять сложный поиск по проектной документации и так далее**.**

**Основная часть**

Автоматизированное управление проектов произведено в Azure DevOps Server. Работа распределена по итерациям 2 недели. На каждую задачу распределены сотрудники в соответствии с доступностью на итерации (в среднем это 80 рабочих часов на итерации). Время каждой задачи оценено, за какой промежуток будет выполнена задача. Автоматизированное средство позволяет в режиме реального времени наблюдать на каком этапе находится задача Proposed, Active, Resolve, Closed.

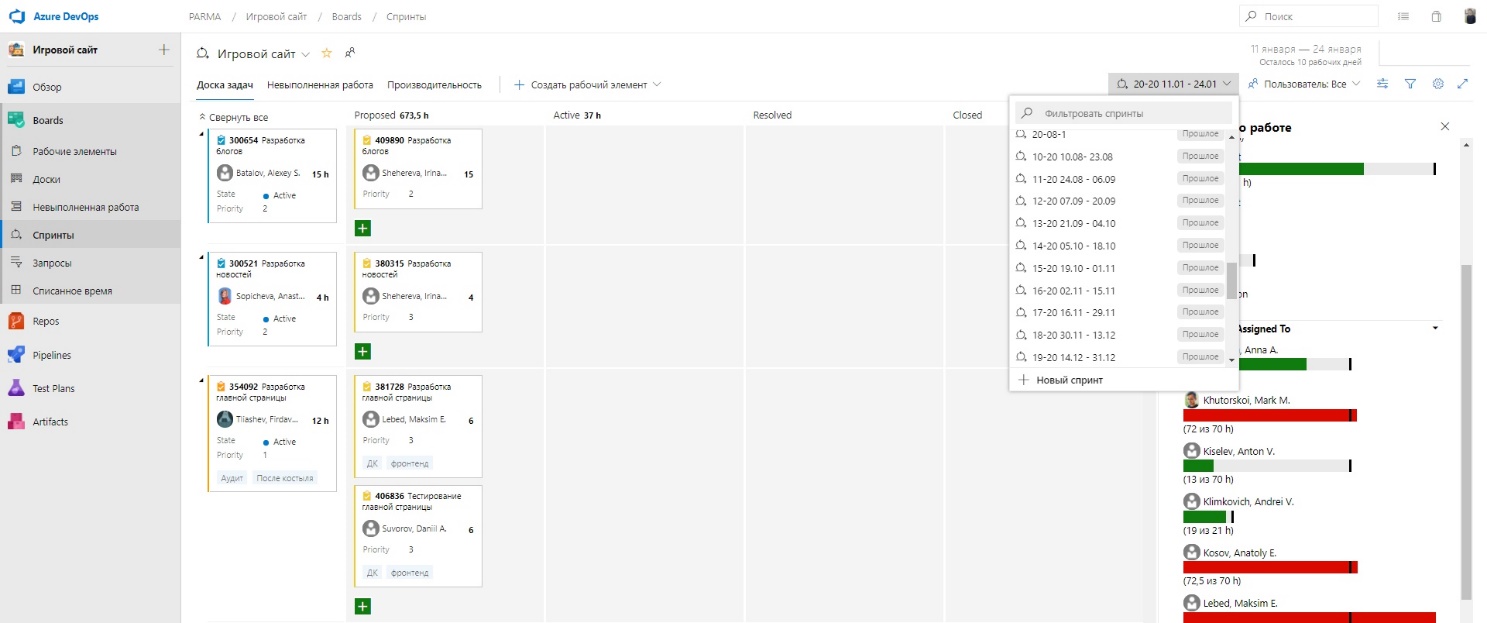


Рисунок 1 – Проект “Игровой сайт” в Azure DevOps Server