**ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОМУ ПО**

[Требования описанию процесса управления ИТ-проектом (для магистров) 2](#_Toc171790882)

[Требования к научной деятельности магистранта 7](#_Toc171790883)

[Требования к разрабатываемому программному обеспечению на базе платформы 1С:Предприятие 8](#_Toc171790884)

[Требования к разрабатываемому ПО на языке высокого уровня 9](#_Toc171790885)

[Требования к разрабатываемому программному обеспечению для мобильных платформ 10](#_Toc171790886)

[Требования к разрабатываемому сетевому ресурсу (веб-сайта, порталу и т.п.) 11](#_Toc171790887)

[Требования к разрабатываемому интеллектуальному сервису (с использованием машинного обучения, нейронных сетей и интеллектуального анализа данных) 13](#_Toc171790888)

[Требования к разрабатываемому интерактивному графическому приложению (компьютерной игре) 16](#_Toc171790889)

[Требования к тестированию программного продукта (для 3 раздела преддипломной практики) 20](#_Toc171790890)

## Требования описанию процесса управления ИТ-проектом (для магистров) – ЭТОТ РАЗДЕЛ ТОЛЬКО ДЛЯ МАГИСТРОВ

* + В магистерской диссертации должен быть подробно отражен **процесс управления ИТ-проектом** и **процесс разработки ПО**.
  + Для **управления ИТ-проектом** выбирается одна из методологий управления проектами: Agile, Scrum, Kanban, Scrumban, Lean, Prototype, RAD, FDD, Кайдзен. Также могут использоваться стандарты управления проектами: PMBoK, IMBOK, SWEBOK, ICB, ISO 10006, ОРМЗ и т.п.
  + Для **разработки ПО** выбирается одна из методологий: Waterfall Model (каскадная модель или "водопад"), V-Model, Incremental Model (инкрементная модель), RAD Model (быстрая разработка приложений), Iterative Model (итеративная или итерационная модель), Spiral Model (спиральная модель), ХР (Extreme Programming), Microsoft Solutions Framework (MSF).
  + Управление проектом должно осуществляться посредством надлежащего применения и интеграции логически сгруппированных процессов управления проектом, объединенных в 5 групп:Процессы инициации, Процессы планирования, Процессы планирования, Процессы исполнения, Процессы мониторинга и контроля, Процессы закрытия.
  1. **Процессы инициации** – это принятие решения о начале выполнения проекта.

Процессы инициации состоят из следующих шагов:

* выбор проекта и определение деловых потребностей;
* сбор информации;
* определение целей проекта;
* определение ограничений и допущений;
* разработка описания продукта;
* определение обязанностей менеджера проекта;
* определение требований к человеческим ресурсам (кадры, квалификация);
* оценочное определение ресурсов;
* окончательная доработка **Устава проекта** и назначение **менеджера проекта**.
  1. **Процессы планирования** – это определение критериев успеха проекта и разработка рабочих схем их достижения. Они направлены на разработку планов по составляющим проекта (расписание, стоимость, бюджет, качество, персонал, риски, взаимодействие, контракты и пр.) и их интеграцию в целостный, согласованный документ - **План проекта**.

К процессам планирования относятся:

* **Планирование содержания** – на основе Устава проекта и других входных документов, составляется документ, описывающий: а) уточненное описание продукта и результатов поставки; б) классификацию возможных изменений и способ их обнаружения; в) порядок оценки и включения изменений в проект.
* **Определение содержания** – разбиение, декомпозиция целей проекта на меньшие и более управляемые части (подцели). Глубина декомпозиции должна обеспечивать возможность назначения законченных групп работ и исполнителей на эти части. Результатом определения содержания является **ИСР (Иерархическая Структура Работ)**, англ. WBS (Work Breakdown Structure).
* **Определение состава операций** – подготовка перечня всех операций, выполняемых по проекту, и **уточнение ИСР**. Операции, не включенные в уточненную ИСР, считаются не включенными в проект и не подлежат выполнению.
* **Планирование ресурсов** – определение потребности (состава, количеств) в людских и материальных ресурсах, необходимых для выполнения операций проекта. На этом этапе должны быть определены стейкхолдеры проекта. К ключевым стэйкхолдерам относятся:
* **менеджер проекта** – лицо, ответственное за конечные результаты проекта и управляющее проектом. Как правило, менеджером проекта является сам выпускник;
* **заказчик** – физическое или юридическое лицо – будущий потребитель продукта проекта;
* **подрядчик** – юридическое лицо, сотрудники которой выполняют работы проекта; исполняющей организацией может выступать как внешняя организация, так и временная структура внутри самой заказывающей организации;
* **спонсор** – лицо или группа лиц (физических или юридических), обеспечивающее проект финансовыми и другими ресурсами;
* **члены команды проекта** – группа, которая выполняет работы проекта. Состав команды проекта как правило формируется по решаемым задачам: frontend-разработчик, backend-разработчик, fullstack-разработчик, DevOps-интегратор, системный аналитик, аналитик данных, ML-специалист, тестировщик.

Также на этом этапе:

* оценивается компетентность, знания и навыки стейкхолдеров;
* проводится анализ проекта на соответствие требованиям стэйкхолдеров;
* проводится работа по привлечению стэйкхолдеров в проект: в качестве экспертов, в качестве членов комиссий по изменениям, итогам и сдаче-приемке проекта, в качестве получателей отчетов по проекту;
* если существуют расхождения между стэйкхолдерами, то с помощью компромиссов проблема должна решаться в пользу заказчика.
* **Определение взаимосвязей операций** – выявление взаимосвязей и взаимозависимостей операций, построение сетевых диаграмм работ проекта. Часть операций связана между собой жесткой логикой, другие операции могут выполняться в произвольной последовательности, т.е. связаны мягкой логикой.
* **Оценка длительности операций** – установление количества единиц времени на операции проекта, вычисление резервов времени и критического пути с минимальной гибкостью по времени.
* **Оценка стоимости** – количественная оценка возможных затрат на вовлекаемые ресурсы, составление сметы и плана управления стоимостью.
* **Планирование управления рисками** – установление подхода и мероприятий (когда, как и что делать) при угрозе или наступлении нежелательных и незапланированных событий и отклонений, с целью их предотвращения или эффективного реагирования.
* **Составление расписания** – анализ данных о последовательности и длительности операций и требуемых ресурсах с целью создания расписания исполнения проекта.
* **Разработка бюджета** – определение сметной стоимости по отдельным пакетам работ и проекту в целом.
* **Разработка плана проекта** – интеграция данных предыдущих процессов и составление согласованного Плана проекта – в виде одного документа или собрания документов.
* **Вспомогательные процессы** планирования устанавливают стандарты качества, распределение ролей и ответственности, информационные потребности участников и способы взаимодействия, выявляют риски и последствия их воздействия на цели проекта и т.д.
  1. **Процессы исполнения** – это координация людей и других ресурсов для выполнения плана. Основной процесс - **исполнение плана проекта**, т.е. непосредственное выполнение составляющих его операций. Вспомогательные процессы обеспечивают гарантии качества, комплектацию/распределение работ и информации, проведение совещаний о ходе работ и идентификацию изменений, развитие навыков и знаний команды, сбор предложений поставщиков, управление отношениями с поставщиками.
  2. **Процессы мониторинга и контроля** – определение соответствия плана и исполнения проекта поставленным целям и критериям успеха и принятие решений о необходимости применения корректирующих воздействий и определение необходимых корректирующих воздействий, их согласование, утверждение и применение. Сюда относятся:
* **Отчетность по исполнению** – есть сбор и распространение информации о ходе проекта и прогнозах.
* **Общее управление изменениями** – есть координация изменений по проекту в целом.
* Вспомогательные процессы обеспечивают удостоверение правильности выполнения работ, фиксацию и принятие изменений, контроль и изменение расписания и бюджета, соответствие стандартам качества и устранение причин его снижения, отслеживание и выявление рисков, оценку мероприятий по снижению рисков.
  1. **Процессы закрытия** – формализация выполнения проекта и подведение его к упорядоченному финалу. К этим процессам относится:
* проверка и тестирование конечного продукта;
* окончательные расчеты со всеми участниками проекта, финансовое закрытие;
* окончательное обновление документов проекта;
* завершение отчета о выполнении проекта;
* сбор, интеграция накопленных знаний и формирование архива проекта;
* официальная приемка проекта заказчиком, передача и запуск в эксплуатацию;
* освобождение задействованных ресурсов
  + На всех этапах должна проводиться **оптимизация управления проектом** – использование методов **исследования операций и методов оптимизации** для определения критических точек проекта, осуществления рационального распределения ресурсов, эффективного планирования и принятия решений, в том числе в условиях риска и неопределенности.
  + В соответствии с выбранной методологией управления проектами в магистерской диссертации необходимо отразить все вышеперечисленные этапы разработки проекта.

Карта связи процессов управления проектами и областей знаний

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Область**  **знаний** | **Группы процессов управления проектом** | | | | |
| **Инициация** | **Планирование** | **Исполнение** | **Мониторинг и контроль** | **Закрытие** |
| **1.**  **Управление**  **интеграцией**  **проекта** | Разработка  устава  проекта. | Разработка плана  управления  проектом. | Руководство  и управление  работами  проекта. | Мониторинг и  контроль работ  проекта.  Интегрированный  контроль  изменений. | Закрытие  проекта или  фазы. |
| **2.**  **Управление**  **содержимы**  **м проекта** |  | Планирование  управления  содержанием.  Сбор требований.  Определение  содержания.  Создание ИСР. |  | Подтверждение  содержания.  Контроль  содержания. |  |
| **3.**  **Управление**  **сроками**  **проекта** |  | Планирование  управления  расписанием.  Определение  операций.  Определение последовательности операций.  Оценка ресурсов  операций. Разработка  расписания. |  | Контроль  расписания. |  |
| **4.**  **Управление**  **стоимостью**  **проекта** |  | Планирование  управления  стоимостью. Оценка  стоимости.  Определение  бюджета. |  | Контроль  стоимости. |  |
| **5.**  **Управление**  **качеством**  **проекта** |  | Планирование  управления  качеством. | Обеспечение  качества. | Контроль  качества. |  |
| **6.**  **Управление**  **человечески**  **ми**  **ресурсами**  **проекта** |  | Планирование  управления  человеческими  ресурсами. | Набор  команды  проекта.  Развитие  команды  проекта.  Управление  командой  проекта. |  |  |
| **7.**  **Управление**  **коммуникац**  **иями**  **проекта** |  | Планирование  управления  коммуникациями. | Управление  коммуникац  иями. | Контроль  коммуникаций. |  |
| **8.**  **Управление**  **рисками**  **проекта** |  | Планирование  управления рисками.  Идентификация  рисков.  Качественный анализ  рисков.  Количественный  анализ рисков.  Планирование  реагирования на  риски. |  | Контроль рисков. |  |
| **9.**  **Управление**  **закупками**  **проекта** |  | Планирование  управления  закупками. | Проведение  закупок. | Контроль закупок. | Закрытие  закупок. |
| **10.**  **Управление**  **заинтересов**  **анными**  **лицами** | Определение  заинтересова  нных сторон. | Планирование  управления  заинтересованными  сторонами. | Управление  вовлечением  заинтересова  нных сторон. | Контроль  вовлечения  заинтересованных  сторон. |  |

## Требования к научной деятельности магистранта– ЭТОТ РАЗДЕЛ ТОЛЬКО ДЛЯ МАГИСТРОВ

1. **Во введении** к диссертации должно быть указано **количество научных публикаций**, опубликованных лично выпускником, со ссылкой на список литературы. Все научные публикации должны быть отражены в списке литературы магистерской диссертации
2. Обязательное количество научных публикаций, которые должны быть опубликованы выпускником – **минимум 3**, индексированных в РИНЦ.

Допускается замена научной публикации на **документ, подтверждающий право интеллектуальной собственности** (патент, свидетельство о регистрации программы для ЭВМ)

1. Во введении к диссертации должно быть указано **количество научных и научно-практических конференций,** на которых выпускник докладывал результаты своей проектной и научно-практической работы – **минимум 2 конференции.** Подтверждения участия в конференциях в виде **сертификата участника размещаются** в Приложении к диссертации.

***Пример формулировки в диссертацию:***

По результатам подготовки диссертации было опубликовано ХХ статей [ХХ, ХХ].

Результаты проектной и научно-практической работы докладывались на ХХХХ конференциях. На алгоритмические и программные решения получен документ на право интеллектуальной собственной: патент/ свидетельство о регистрации программы для ЭВМ №000000000000000 от ХХ.ХХ.20ХХ [ХХ, ХХ].

## Требования к разрабатываемому программному обеспечению на базе платформы 1С:Предприятие

* 1. Обязательна загрузка в ЭУ вместе с ВКР / выгрузка в репозиторий Gitflic или Github **файла с выгрузкой конфигурации 1С \*.dt** (возможно приложить архив всей папки с конфигурацией, с комментариями о версии платформы 1С, для которой выполнялся проект).
  2. Конфигурация должна содержать следующий минимальный набор объектов:
* Справочники – **не менее** **15**, из них **не менее 3 иерархических.**
* Документы – **не менее** **10**.
* Константы – **не менее** **2**.
* Перечисления – **не менее** **3**.
* Общие модули – **не менее** **1**.
* Обработки – **не менее** **1**.
* Отчеты – **не менее** **5**, в том числе сложные – **не менее 3**.   
  Сложным отчетом считается отчет, для построения которого следует обратиться к данным **не менее 3-х объектов конфигурации**. Для решения задач учета следует использовать специализированные объекты (планы и регистры).
* Подсистемы - **не менее 4.**
* Регистры накопления – **не менее 3.**
  1. Отчеты, представленные в конфигурации должны иметь **печатные формы**.
  2. Проанализировать необходимость использования **регистров сведений** и **регистров расчета**. Разработать необходимые регистры сведений и регистры накопления. Анализ отразить в ВКР.
  3. Разработать механизм проведения документов по регистрам накопления, а также по иным регистрам конфигурации, при наличии
  4. Создать **не менее 3-х ролей пользователей** конфигурации.
  5. Объем собственноручно разработанного программного кода должен быть **не менее 500 логических строк кода**. Пустые строки, строки комментариев, строки только с управляющими символами подсчету не подлежат. Программный код, созданный автоматически с использованием платформы 1С:Предприятие не учитывается.
  6. Для основных сценариев использования конфигурации должны быть созданы бизнес-процессы.
  7. Заполнить конфигурацию тестовыми данными, соответствующими предметной области и выбранной организации. **Не менее 10 примеров для каждого объекта конфигурации.**

## Требования к разрабатываемому ПО на языке высокого уровня

* 1. Программное обеспечение разрабатывается на языке **Python** c использованием сред разработки **PyCharm** / **VS Code** / **Google Colab / Yandex DataSphere**.
  2. Обязательна загрузка в ЭУ вместе с ВКР / выгрузка в репозиторий Gitflic или Github **исполняемого \*.exe файла и файлов программного кода \*.py** (при необходимости архивируется вся папка проекта, с комментариями об используемых дополнительных библиотеках и версиях), либо **файла программного кода \*.ipynb**, если работа выполнялась с использованием языка программирования Python и сервисов Google Colab и Yandex DataSphere.
  3. Объем собственноручно разработанного программного кода должен быть **не менее 2 000** **логических строк кода**. Пустые строки, строки комментариев, строки только с управляющими символами подсчету не подлежат.
  4. Обязательно должен быть сформирован **исполняемый \*.exe** файл
  5. В качестве приоритетного решения следует рассматривать программное обеспечение, построенное с использованием клиент-серверной архитектуры. Разработка программного решения, не предполагающего сетевое взаимодействие должна быть обоснована.
  6. Обеспечить хранение данных приложения в СУБД, поддерживающей SQL-запросы. Рекомендуется PostgreSQL.
  7. В приложении должно быть **не менее 10 оконных форм**.
  8. Реализовать **не менее одного диалогового окна** для получения данных от пользователя со специальной формы.
  9. Реализовать функционал доступа к файловой системе.
  10. Реализовать панель управления / настройки в приложении, где добавить **не менее 5 пунктов** **меню**.
  11. Реализовать возможность формирования документов в форматах **.docx, .xlsx** и т.п. из разрабатываемого приложения.
  12. Предусмотреть **не менее 3-х ролей пользователей**. Предоставить учетные данных всех созданных пользователей.
  13. Разработать панель администратора.
  14. Разработать личный кабинет пользователя.
  15. Предусмотреть функционал «Справка по системе».

В случае разработки ПО **для микроконтроллеров и программируемых логических интегральных схем:**

* 1. Разработка кода должна выполняться с использованием аппаратного обеспечения.
  2. Результатом выполнения выпускной квалификационной работы должен быть работающий программно-аппаратный комплекс.

## Требования к разрабатываемому программному обеспечению для мобильных платформ

1. Приложение должно быть разработано под операционную систему Android. Язык написания приложения Java или Kotlin.
2. Обязательна загрузка в ЭУ вместе с ВКР / выгрузка в репозиторий Gitflic или Github исполняемого **\*.apk** **файла-приложения для Android** и исходных файлов с программным кодом.
3. Работа над проектом должна вести в git-репозитории в системе Gitflic или Github. Работа над ВКР должна представлять собой **не менее 50 коммитов от имени студента**, происходящих **в течение периодов прохождения преддипломной практики и подготовки ВКР**. Коммиты должны отражать итеративное развитие проекта, то есть не просто создание целиком готовых файлов, а их изменение по мере развития функционала проекта.
4. **Использование готовых универсальных кроссплатформенных фреймворков запрещается**. Используемые средства разработки не регламентируются, однако рекомендуется использовать Android Studio для уменьшения проблем со сборкой приложения в аудитории.
5. Определить не менее **5 взаимосвязанных сущностей** в рамках предметной области.
6. Для организации сущностей использовать модель MVC или MVVM.
7. Для организации интерфейса в файлах представлений использовать не менее одного макета ConstraintLayout.
8. **Не менее одного макета** представления сформировать как для портретного, так и для альбомного режима.
9. Ресурсы, в том числе строковые, вынести из кода и расположить в отдельных предусмотренных для этого файлах и папках в структуре проекта.
10. Контроллеры представлений выполнять с помощью класса Fragment и библиотеки поддержки androidx. Разработать взаимодействие фрагментов на разных этапах жизненного цикла.
11. У каждой сущности должна быть форма редактирования объекта, а также форма списка объектов, построенная на базе класса RecyclerView.
12. Обеспечить хранение данных приложения в СУБД, поддерживающей SQL-запросы. Допустимо использование библиотек ORM для упрощения доступа к данным. Рекомендуется использование SQLite.
13. Реализовать панель управления в приложении, где добавить **не менее 5 пунктов меню.**
14. Реализовать **не менее одного диалогового окна** для получения данных от пользователя со специальной формы.
15. Добавить в функционал возможность вызова стандартных приложений через неявные намерения (Intent).
16. В основной сущности должен быть реализован функционал доступа к файловой системе и получение изображений с камеры.
17. Реализовать взаимодействие основной сущности с внешним веб-сервисом.

## Требования к разрабатываемому сетевому ресурсу (веб-сайта, порталу и т.п.)

* 1. Сетевой ресурс должен быть разработан с применением языков программирования – PHP, JavaScript или Python c использованием фреймворков Django / Flask.
  2. Обязательна загрузка в ЭУ вместе с ВКР / выгрузка в репозиторий Gitflic или Github **файлов веб-страниц в формате \*.html, \*.css, \*.php, \*.js** (при необходимости архивируется вся папка проекта, с комментариями об используемых версиях) или **файлами программного кода \*.py** (при необходимости архивируется вся папка проекта, с комментариями об используемых дополнительных библиотеках и версиях), если работа выполнялась с использованием языка программирования Python.
  3. Работа над проектом должна вести в git-репозитории в системе Gitflic или Github. Работа над ВКР должна представлять собой **не менее 50 коммитов от имени студента**, происходящих **в течение периодов прохождения преддипломной практики и подготовки ВКР**. Коммиты должны отражать итеративное развитие проекта, то есть не просто создание целиком готовых файлов, а их изменение по мере развития функционала проекта.
  4. Дополнительно сетевой ресурс должен быть размещен на бесплатном хостинге.
  5. При работе над сетевым ресурсом **недопустимо подменять разработку** ресурса **разворачиванием готового программного обеспечения** и установку готовых дополнительных программных модулей для работы с ним. Допускается разработка собственных программных модулей для работы с имеющимися программными системами и/или модификация таковых модулей. **Запрещено использование CMS и готовых плагинов к ней**. Исключением для данного пункта могут быть работы, тема которых явно указывает на данный тип деятельности.
  6. При разработке сетевых ресурсов студентом должно быть предложено программное и/или аппаратное решение, готовое к внедрению в выбранный бизнес-процесс.
  7. Объем программного кода должен быть **не менее 2 000 логических строк кода**, в том числе и строк, написанных с применением языка разметки. Пустые строки, строки комментариев, строки только с управляющими символами подсчету не подлежат.
  8. Разработать структуру Web-страниц. Сетевой ресурс должен содержать **не менее 10 страниц**.
  9. Разработать макет каждой страницы сетевого ресурса.
  10. Предусмотреть форму обратной связи.
  11. Разработать **не менее 10 пунктов бокового и/или верхнего меню, «хлебные крошки».**
  12. Обеспечить хранение данных приложения в СУБД, поддерживающей SQL-запросы.
  13. Предусмотреть **не менее 3-х ролей пользователей**, например, администратор, менеджер, клиент. Предоставить учетные данных всех созданных пользователей.
  14. Разработать панель администратора сетевого ресурса.
  15. Разработать личный кабинет пользователя сетевого ресурса.
  16. Обеспечить корректное отображение сетевого ресурса в основных браузерах (Mozilla Firefox, Chrome, Яндекс.Браузер, Microsoft Edge)
  17. Обеспечить корректное отображение сетевого ресурса на разных устройствах (ПК, ноутбук, смартфон)
  18. Текст статей в сетевом ресурсе должен быть читабельным, грамотно сформулированным, корректным и соответствовать предметной области ВКР. Не должно быть пустых страниц в сетевом ресурсе.

## Требования к разрабатываемому интеллектуальному сервису (с использованием машинного обучения, нейронных сетей и интеллектуального анализа данных)

1. Интеллектуальный сервис разрабатывается на языке Python с использованием сервисов Google Colab или Yandex DataSphere.
2. Обязательна загрузка в ЭУ вместе с ВКР **файла/ов программного кода \*.ipynb** (при необходимости архивируется вся папка проекта, с комментариями об используемых дополнительных библиотеках и версиях, а также с используемыми датасетами). Обязательна выгрузка в репозиторий Gitflic или Github **файла/ов программного кода \*.ipynb** (при необходимости архивируется вся папка проекта, с комментариями об используемых дополнительных библиотеках и версиях, а также с используемыми датасетами)
3. Работа над проектом должна вести в git-репозитории в системе Gitflic или Github. Работа над ВКР должна представлять собой **не менее 50 коммитов от имени студента**, происходящих **в течение периодов прохождения преддипломной практики и подготовки ВКР**. Коммиты должны отражать итеративное развитие проекта, то есть не просто создание целиком готовых файлов, а их изменение по мере развития функционала проекта.
4. Объем собственноручно разработанного программного кода должен быть **не менее 2 000** **логических строк кода**. Считаются только строчки алгоритмического кода, строки кода разметки типа HTML или JSON, пустые строки, строки комментариев, строки только с управляющими символами подсчету не подлежат.
5. Создаваемое приложение для демонстрации работы приложения обязательно должно позволять:

* загружать и сохранять данные для обучения моделей из локальных файлов или из внешних хранилищ данных (типа kaggle или коннекторов для подключения хранилищ на облачных сервисах);
* проводить обучение моделей, сохранять обученную модель в файл по результатам обучения;
* загружать обученную модель из файла.
* применять обученную или загруженную модель по назначению
* оценивать и сохранять результаты работы модели для обучения новых версий модели

1. Дизайн приложения должен быть отражён в отчёте. Загрузка данных в приложение отражается в отчёте в **разделе** **загрузки данных**, обучение моделей в приложении - в **разделе обучения моделей**, процесс применения модели и оценки её результатов – в **разделе тестирования модели**.
2. В git-репозитории должен присутствовать файл **requirements.txt**, описывающий список необходимых для работы приложения и модели библиотек, которые надо будет установить проверяющему в пустое виртуальное окружение. Файл необходимо положить в корневой каталог репозитория.
3. В git-репозитории должен присутствовать в корневом каталоге файл **main.py**, который запускает приложение для работы с моделью.
4. Все демонстрируемые в отчёте модели должны быть сохранены в сдаваемом git-репозитории в файле с названием **model** и расширением, которое удобно для конкретной выбранной архитектуры (например, модели на PyTorch сохраняются с расширением **.pt**, то есть будет **model.pt**). Если в отчёте демонстрируется больше 1 модели, то к названию файла добавляется порядковый номер (напр., model1, model2 и т.д.). При проверке будет ожидаться, что работать эти модели будут так же, как описано в отчёте.
5. Для всех демонстрируемых моделей, в отчёте должны быть сохранены данные для их обучения. При проверке будет ожидаться, что модель, обученная на этих данных, будет совпадать по метрикам своей эффективности с моделью в файле и описанием этих метрик в отчёте.
6. При работе моделей, ожидается достижений уровня эффективности, сопоставимого с уровнем **Accuracy >= 70**. Если к конкретной модели в конкретной предметной области этот уровень не применим, то можно согласовать с руководителем работы другую метрику эффективности, отразив её в отчёте.
7. Реализация GUI интеллектуального сервиса может быть в виде «оконных» или web форм, а также с использованием возможностей чат-ботов по отрисовке интерфейсных компонентов. Конкретный формат GUI согласовывается с руководителем ВКР.
8. В ВКР обязательно должно быть математическое описание выбранной модели.

**Методологические требования к разработке интеллектуального сервиса**

1. Основной формат демонстрации навыков – **создание, обучение, тестирование и улучшение модели машинного обучения** в выбранной предметной области, сопровождаемое **визуальным анализом** получаемых результатов. Формат сохраняется не зависимо от сложности модели.
2. Создание модели предполагает, что студент умеет применять основные способы создания моделей в программном коде, как в плане написания алгоритмов «с нуля», так и в плане использование стандартных готовых библиотек – от базовых библиотек типа NumPy и Pandas, до библиотек глубокого обучения типа PyTorch и Tensorflow.
3. Создание модели предполагает, что студент владеет не только типовыми способами создания моделей, но и способен адаптировать модель под работу в конкретной предметной области, осуществляя математически грамотный подбор гиперпараметров моделей, структуру ансамблевых моделей или топологий нейронных сетей.
4. Уверенное владение моделями моделей предполагает знакомство студентов с алгоритмами оптимизации и функциями подсчёта потерь, способность объяснить математический смысл этих алгоритмов и функций и их отношение к целевой функции, выражаемой моделью.
5. Обучение модели предполагает получение данных, на основе которых проводится обучение, подготовку данных к работе, а так же проведение самого процесса обучения модели с визуализацией промежуточных и итоговых результатов.
6. Получение данных предполагает, что студент знает откуда вообще можно брать данные для обучения моделей. Знаком как с публичными ресурсами типа kaggle, так и с подходами к самостоятельной добыче данных через web-скрейпинг, публичные API или через коннекторы для подключения хранилищ на облачных сервисах.
7. **Генерация случайных данных не допускается**, поскольку случайные данные автоматически лишают результаты работы конкретной практической ценности.
8. Получение данных предполагает, что студент понимает, какие форматы данных необходимы разным типам моделей для успешной работы в выбранной предметной области, способен привести исходные данные в нужный формат, произведя их очистку и необходимые трансформации, такие как нормализация, векторизация, приведение к тензорам, в том числе с использованием готовых конструкций загрузки данных в профессиональных библиотеках.
9. Получение данных предполагает способность студента определить, какие данные полезны, а какие нет, применяя для этого разные способы отбора и фильтрации признаков, в том числе на основе подходов визуального анализа.
10. Получение данных предполагает способность студента выделять из данных тренировочные и тестовые выборки, знать стандартные способы и инструменты такого выделения, уметь применять эти способы с учётом нужд предметной области и задачи анализа.
11. Обучение модели предполагает, что студент знает, как запустить такое обучение в рамках выбранного им способа создания модели. На минимальном уровне – умеет применять стандартные инструменты обучения моделей, представленных в рабочих библиотеках. На максимальном уровне – способен самостоятельно реализовать алгоритм обучения модели методом градиентного спуска.
12. Обучение многослойных моделей предполагает способность студента запускать автоматические фазы обучения (т.н. «эпохи»), отслеживать и визуализировать изменение эффективности модели на каждом этапе обучения.
13. По результатам обучения, в некоторых случаях - и в процессе обучения, необходимо проводить тестирование моделей. Для этого надо знать метрики оценки эффективности моделей, уметь производить тестирование и применять метрики к результатам тестирования, владеть инструментами визуализации метрик.
14. Тестирование моделей предполагает от студента способность не только произвести тестирование, но и сделать содержательные выводы из его результатов. Содержательными считаются выводы о работоспособности модели, полезности результатов обучения, может быть возможных способах улучшения модели.
15. Улучшение модели машинного обучение предполагает внесение изменений в модель с целью получить улучшенные результаты. От студента ожидается способность сформулировать предложение об улучшении модели по результатам первичного тестирования, обосновать его с помощью математического и визуального анализа, выполнить это улучшение (изменив параметры модели или пересоздав модель заново с новой топологией) и заново провести обучение и тестирование, оценив результаты улучшения (то есть фактически повторить весь путь с улучшенной моделью). Это должно быть отражено в ВКР.

Студенты должны демонстрировать все вышеперечисленные пункты по отношению к любому типу модели машинного обучения, представленному в стандартных рабочих библиотеках.

Таким образом, студент предоставляет не только работающую модель, но и приложение как среду, в которой он тестировал работу своей модели и в которой её работу может протестировать руководитель ВКР или другой проверяющий. Отсутствие среды, в которой можно проверить работу модели, является нарушением правил сдачи работы.

## Требования к разрабатываемому интерактивному графическому приложению (компьютерной игре)

1. Интерактивное графическое приложение разрабатывается на языке **Python** c использованием интерпретатора **PyCharm** или **Google Colab**.
2. Обязательна загрузка в ЭУ вместе с ВКР **исполняемого \*.exe файла** ивыгрузка в репозиторий Gitflic или Github **файлов программного кода \*.py** (при необходимости архивируется вся папка проекта, с комментариями об используемых дополнительных библиотеках и версиях) если работа выполнялась с использованием языка программирования Python, либо **файла программного кода \*.ipynb**, если работа выполнялась с использованием языка программирования Python и сервисов Google Colab и Yandex DataSphere.
3. Работа над проектом должна вести в git-репозитории в системе Gitflic или Github. Работа над ВКР должна представлять собой **не менее 50 коммитов от имени студента**, происходящих **в течение периодов прохождения преддипломной практики и подготовки ВКР**. Коммиты должны отражать итеративное развитие проекта, то есть не просто создание целиком готовых файлов, а их изменение по мере развития функционала проекта.
4. Объем собственноручно разработанного программного кода должен быть **не менее 2 000** **логических строк кода**. Считаются только строчки алгоритмического кода, строки кода разметки типа HTML или JSON, пустые строки, строки комментариев, строки только с управляющими символами подсчету не подлежат.
5. При описании внутренней архитектуры приложения, студент должен описать **не менее** **10 сущностей игровой системы** (из которых **минимум 5** должны быть реализованы самостоятельно) и **не менее 5 игровых систем** (из которых **минимум 3** должны быть реализованы самостоятельно).
6. При описании интерфейса приложения, студент должен описать **не менее 4 игровых экранов и не менее 15 элементов интерфейса** на этих экранах (из которых **не менее 8** взяты не в готовым виде, а реализованы самостоятельно). Элементом интерфейса считается любой интерфейсный объект, реагирующий на события в игровой системе (например, шкала здоровья или индикатор возможности действия) или сам инициирующий событийную цепочку по факту ввода от игрока (например, кнопка).
7. Архитектура приложения обязательно должна содержать локально сохраняемые данные, такие как пользовательские настройки, сохранения, пароли и т.д.
8. Для реализации приложения можно использовать любые игровые движки и middleware, при этом имея ввиду требование о реализации достаточного объёма собственного функционала поверх заимствованного.
9. В сдаваемом git-репозитории должен в корневом каталоге присутствовать **файл readme.md**, описывающий способы управления игровым процессом, а также содержащий ссылки на скачивание запускаемых сборок игрового приложения.
10. Хранение сборок в самом git-репозитории не разрешается. Можно выгрузить сборки на любой внешний ресурс для скачивания или, что предпочтительнее, осуществить релиз своего приложения на специализированной площадке типа itch.io.
11. Необходимым условием является наличие сборки, которую можно запустить под Linux, для запуска на компьютерах Университета для проверки. **Сборки только под Windows недостаточно**. Если проект нельзя собрать для Linux или хотя бы для Web-запуска из браузера под Linux, то запускаемую сборку необходимо установить в соответствующий докер-контейнер.

**Методологические требования к разработке интерактивного графического приложения**

1. **Основной формат демонстрации навыков –** выполнение полного комплекса работ по разработке интерактивного графического приложения (далее «игрового приложения»), в частности: определение требований к игровому приложению, планирование отвечающего этим требованиям функционала, как в плане системной архитектуры, так и в плане графического интерфейса, с последующей реализацией этого функционала, его тестированием на предмет соответствия требованиям и улучшение приложения по результатам тестирования.
2. Формат демонстрации навыков в целом сохраняется не зависимо от природы конкретного игрового приложения. Планирование, реализация, тестирование и улучшения являются неотъемлемыми частями процесса создания любого такого приложения.
3. На этапе определения требований к игровому приложению возможна вариативность, в том числе и в плане демонстрируемых студентом навыков. Рассматривается три основных подхода к определению требований к игровому приложению: требования научной модели, требования дизайн-фокуса и требования коммерческого продукта. Эти три способа требуют демонстрирования разных наборов навыков на этапе определения требований к игровому приложению.
4. Определение требований к игровому приложению через требования научной модели используется когда целью работы ставится создание игровой модели какого-то сложного процесса. Примерами приложений, моделирующих подобные сложные процессы являются игровые приложения в стиле Microsoft Flight Simulator, Kerbal Space Program, Civilisation. Требования к игровому приложению в этом случае проистекают из требований к точности реализации и визуализации работы научной модели и описанных в этой модели процессов.
5. При выборе этого способа определения требований к игровому приложению, наличие готовой научной модели, выраженной в соответствующем наборе научных публикаций, является необходимым граничным условием. Студент может взять реализацию игрового приложения на основе научной модели как задание только если предъявит руководителю научные публикации, описывающие соответствующую научную модель.
6. Если студент решает в своей работе определять требования к своему игровому приложению через призму требований научной модели, то от него будут ожидаться
   1. Знание о научных моделях в выбранной им предметной области, описывающих моделируемый процесс. Способность адресовать к научным публикациям, обосновывать ими элементы дизайна игрового приложения.
   2. Умение реализовывать выбранную научную модель в программном виде, как в плане внутренней рабочей логики этой модели, так и в плане визуализации её работы и демонстрации возникающей в этой модели интересных процессов.
7. Определение требований к игровому приложению на основании дизайн-фокуса предполагает наличие внешнего бизнес-процесса или группы бизнес-процессов, в котором существует проблема. Решение этой проблемы и становится дизайн-фокусом игрового приложения. Требования к игровому приложению в этом случае определяются на основании бизнес-анализа исходной группы бизнес-процессов и определения точек применения игрового приложения для разрешения проблем в этих процессах.
8. При выборе этого способа определения требований к игровому приложению, необходимо наличие у студента доступа к специалисту, который поможет ему описать конкретные бизнес-процессы и создаст тем самым возможность для их анализа с целью последующего выделения дизайн-фокуса игрового приложения. Студент может использовать в своей работе бизнес-процессы Университета (и тогда специалистом может выступать руководитель работы) или внешней организации (и тогда специалист должен быть из внешней организации). Отсутствие доступа к специалисту делает невозможным определение требований к игровому приложению через дизайн-фокус.
9. Если студент решает в своей работе определять требования к своему игровому приложению через дизайн-фокус, то от него будут ожидаться:
   1. Навыки анализа бизнес-процессов и выделения в них тех точек, которые могут быть улучшены с помощью игрового приложения.
   2. Знание технологий и подходов геймификации бизнес-процессов и способность применить эти знания для предложения улучшения в бизнес-процессах за счёт игрового приложения.
10. Определение требований к игровому приложению на основании требований коммерческого продукта предполагает, что студент в состоянии определить провести маркетинговое профилирование целевой аудитории продукта, может сформулировать потребности этой целевой аудитории и тем самым обосновать требования к игровому приложению как потребительским характеристик, интересующих целевую аудиторию.
11. Если студент решает в своей работе определять требования к своему игровому приложению как к коммерческому продукту, то от него будут ожидаться:
    1. Навыки поиска других продуктов в выбранной им категории игровых приложений, определения тех их ключевых качеств, на которые реагирует аудитория.
    2. Способность профилирования аудитории выбранной категории игровых приложений с точки зрения интересующего конкретных представителей конкретного профиля набора потребительских качеств.
    3. Способность выделить целевую аудиторию для своего продукта и спланировать потребительские качества собственного продукта, соответствующие маркетинговому профилю целевой аудитории.
12. Сформулировав требования к своему игровому приложению одним из трёх вышеописанных способов, студент переходит к планированию реализации игрового приложения. Необходимо спланировать первую версию приложения, удовлетворяющего заданным требованиям, как в плане программной архитектуры, так и в плане интерфейсов и игровых взаимодействий.
13. На этапе планирования от студента ожидаются:
    1. Знания о типовых способах составления проектной документации для игрового приложения (далее «дизайн-документа»).
    2. Умение правильно структурировать дизайн-документ, отразив в нём как требования к игровому продукту, так и способы их реализации.
    3. Владение способами визуализации схем игровых взаимодействий и необходимых алгоритмов, видов игровых объектов и отношений между ними, способов управления игровым процессом.
14. Спланировав первую версию приложения, студент переходит к его реализации. На этом этапе, от студента ожидаются:
    1. Знания общих архитектурных схем, стандартных алгоритмов и типовых структур данных, позволяющих реализовать запланированное в дизайн-документе.
    2. Умение преобразовывать требования продукта (будь то потребительские качества, требования научной модели или схемы геймификации) в конкретный функционал игрового приложения.
    3. Владение инструментарием создания игровых приложений, будь то промежуточные библиотеки типа pygame или Ren’Py, или интегрированные решения типа Unity.
15. Реализовав первую версию приложения, студент должен протестировать её относительно требований и сделать выводы о необходимых улучшениях. Конкретный характер тестов будет зависеть от способа формирование требований.
    1. В случае научной модели, необходимо реализовать архитектуру автоматического тестирования корректного выполнения научной модели во всех игровых сценариях, показать работу этой архитектуры.
    2. В случае коммерческого продукта, необходимо предложить игровой продукт целевой аудитории, собрать с неё обратную связь. После чего на основании этой обратной связи сделать выводы о том, насколько продукт отвечает потребностям аудитории в рамках сформулированного профиля, предложить и реализовать улучшения продукта.
    3. В случае дизайн-фокуса, необходимо согласовать конкретную схему проверки со специалистом, владеющим улучшаемым с помощью игрового приложения бизнес-процессом. Согласованную схему проверки будет необходимо выполнить, отразить в отчёте работы и после чего, на основании результатов, сделать выводы о работоспособности предложенной геймификации и внести улучшения.

В итоге, от студентов ожидается внятное объяснение требований, которые они сформулировали к реализуемому игровому приложению, способов выполнения этих требований, способов верификации выполнения этих требований, а также улучшений, которые были сделаны в тех местах, где требования с первого раза не выполнялись.

## Требования к тестированию программного продукта (для 3 раздела преддипломной практики)

1. Для разрабатываемого приложения необходимо выбрать **минимум 4 метода тестирования** из нижеприведенных видов тестирования:

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Вид тестирования** | **Метод тестирования** | **Описание метода** |
| 1 | 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 |
| 2 | По уровню функциоанльного тестирования | Смоук-тестирование (дымовое, **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. |
| 3 | В зависимости от исполнителей | Альфа-тестирвоание | 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. |
| 4 | 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. |
| * Тестирование производительности **(performance testing)** | определение стабильности и потребления ресурсов в условиях различных сценариев использования и нагрузок. |
| * 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). |
| * Тестирование масштабируемости **(scalability testing)** | тестирование, которое измеряет производительность сети или системы, когда количество пользовательских запросов увеличивается или уменьшается. |
| * Объемное тестирование **(volume testing)** | это тип тестирования программного обеспечения, которое проводится для тестирования программного приложения с определенным объемом данных. |
| * Стрессовое тестирование **(stress testing)** | тип тестирования направленный для проверки, как система обращается с нарастающей нагрузкой (количеством одновременных пользователей). |
| * Инсталляционное тестирование **(installation testing)** | тестирование, направленное на проверку успешной установки и настройки, обновления или удаления приложения. |
| * Тестирование интерфейса **(GUI/UI testing)** | проверка требований к пользовательскому интерфейсу. |
| * 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. |
| * Тестирование локализации **(localization testing)** | проверка адаптации программного обеспечения для определенной аудитории в соответствии с ее культурными особенностями. |
| * 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. |
| * Тестирование надежности **(reliability testing)** | один из видов нефункционального тестирования ПО, целью которого является проверка работоспособности приложения при длительном тестировании с ожидаемым уровнем нагрузки. |
| * 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. |
| * 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. |
| 5 | По запуску кода на исполнение | Статическое | 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 проводится на работающей системе, не может быть осуществлено без запуска программного кода приложения. |
| 6 | По доступу к коду и архитектуре | Метод белого ящика | метод тестирования ПО, который предполагает полный доступ к коду проекта. |
| Сетод серого ящика | метод тестирования ПО, который предполагает частичный доступ к коду проекта (комбинация White Box и Black Box методов). |
| Метод черного ящика | метод тестирования ПО, который не предполагает доступа (полного или частичного) к системе. Основывается на работе исключительно с внешним интерфейсом тестируемой системы. |
| 7 | 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. |
| Приемочное тестирование | проверяет соответствие системы потребностям, требованиям и бизнес-процессам пользователя. |
| 8 | По степени автоматизации | Ручное тестирование |  |
| Автоматическое тестирование |  |

1. **Сформировать тест план.** Тест план должен отвечать на следующие вопросы: Что необходимо протестировать? Как будет проводиться тестирование? Когда будет проводиться тестирование? Критерии начала тестирования. Критерии окончания тестирования.

**Основные пункты тест плана**:

* Идентификатор тест плана (Test plan identifier);
* Введение (Introduction);
* Объект тестирования (Test items);
* Функции, которые будут протестированы (Features to be tested;)
* Функции, которые не будут протестированы (Features not to be tested);
* Тестовые подходы (Approach);
* Критерии прохождения тестирования (Item pass/fail criteria);
* Критерии приостановления и возобновления тестирования (Suspension criteria and resumption requirements);
* Результаты тестирования (Test deliverables);
* Задачи тестирования (Testing tasks);
* Ресурсы системы (Environmental needs);
* Обязанности (Responsibilities);
* Роли и ответственность (Staffing and training needs);
* Расписание (Schedule);
* Оценка рисков (Risks and contingencies);
* Согласования (Approvals).

1. **Сформировать тест-кейсы** (тестовые сценарии). Обязательные атрибуты тест кейса:

* **Предусловия** (PreConditions) — список действий, которые приводят систему к состоянию пригодному для проведения основной проверки. Либо список условий, выполнение которых говорит о том, что система находится в пригодном для проведения основного теста состояния.
* **Шаги** (Steps) — список действий, переводящих систему из одного состояния в другое, для получения результата, на основании которого можно сделать вывод о удовлетворении реализации, поставленным требованиям.
* **Ожидаемый результат** (Expected result) — что по факту должны получить.

1. На все найденные ошибки **сформировать баг-репорт**. Баг-репорт должен обязательно содержать атрибуты: приоритет и серьезность.
   * **Серьёзность** показывает, насколько баг влияет на возможность работать в программе. Виды:

|  |  |
| --- | --- |
| **Серьёзность бага** | **Как влияет на систему** |
| S1 — Блокирующая (Blocker). | Приложение не запускается ни на одном устройстве или выдаёт ошибку при загрузке.  *Пример: приложение не открывается, поэтому невозможно сделать покупку.* |
| S2 — Критическая (Critical). | Часть функционала не работает, но это не блокирует процесс.  *Пример: Скидка по купону не работает, но товар всё равно можно купить без скидки.* |
| S3 — Значительная (Major). | Часть логики работает некорректно, но пользователь может решить проблему другим способом.  *Пример: поиск в приложении не выдаёт нужный товар, но его можно найти, если искать по брендам.* |
| S4 — Незначительная (Minor). | Не нарушает логику приложения.  *Пример: кнопка «Купить» уехала за экран и видна только наполовину.* |
| S5 — Тривиальная (Trivial). | Не относится к логике приложения и не влияет на общее качество продукта.  *Пример: малозаметная опечатка в меню.* |

**Приоритет** — это критерий, который показывает, насколько быстро нужно исправить дефект. Приоритет может быть:

● высокий — исправить в первую очередь;

● средний — исправить, когда разобрались с первой категорией багов;

● низкий — исправить, когда разобрались с багами других приоритетов.

1. Произвести отладку разработанной системы.