

Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Клочков Юрий Сергеевич
Должность: Ректор
Дата подписания: 05.10.2026 09:26:07
Уникальный программный ключ:
3beb265d5d589e7ff4c954946f3ad99a1e70ac12

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«ТЮМЕНСКИЙ ИНДУСТРИАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт геологии и нефтегазодобычи

Кафедра кибернетических систем

ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические рекомендации
по выполнению курсовых работ (проектов)
для обучающихся направления подготовки
09.04.01 «Информатика и вычислительная техника»
всех форм обучения

Составитель ***В. В. Уваров,***
кандидат технических наук

Тюмень
ТИУ
2021

Технология разработки программного обеспечения: методические рекомендации по выполнению курсовых работ (проектов) для обучающихся направления подготовки 09.04.01 «Информатика и вычислительная техника» всех форм обучения / сост. Уваров В. В.; Тюменский индустриальный университет. – Тюмень: Издательский центр БИК ТИУ, 2021. – 16 с. – Текст: непосредственный.

Методические рекомендации рассмотрены и рекомендованы к изданию на заседании кафедры кибернетических систем «1» сентября 2020 года, протокол № 1.

Аннотация

Настоящие методические указания содержат рекомендации и разъяснения, позволяющие обучающимся по направлению 09.04.01 «Информатика и вычислительная техника» всех форм обучения оптимальным образом организовать процесс выполнения и оформления курсовой работы по дисциплине «Технология разработки программного обеспечения».

1. ВВЕДЕНИЕ

Настоящее пособие содержит общие рекомендации по оформлению курсовой работы по дисциплине «Технология разработки программного обеспечения» в соответствии с требованиями Федерального государственного образовательного стандарта по направлению подготовки 09.03.01 Информатика и вычислительная техника (уровень подготовки кадров высшей квалификации), утвержденного приказом Министерства образования и науки Российской Федерации от "12" января 2016 г. № 5, (зарегистрирован Министерством юстиции Российской Федерации 9 февраля 2016 г., рег. № 41030).

2. ЦЕЛЬ И ЗАДАЧИ ДИСЦИПЛИНЫ

Целью изучения дисциплины является формирование у студентов компетенций по технологиям разработки сложного программного обеспечения.

Задачами дисциплины являются:

- иметь представление о требованиях, предъявляемых к программному обеспечению;
- иметь представление о подходах к разработке программного обеспечения;
- иметь представление о механизмах и технологиях проектирования программного обеспечения;
- иметь представление о механизмах и технологиях реализации программного обеспечения;
- иметь представление о механизмах и технологиях тестирования программного обеспечения.

3. ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

В результате освоения дисциплины у обучающегося должны быть сформированы способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и технологий разработки программного обеспечения.

4. ПРАКТИЧЕСКАЯ РАБОТА

Цель работы: изучение основных подходов к разработке и интеграции программного обеспечения и его реализация, ознакомление и использование свободно распространяемого программного обеспечения.

4.1. Основные сведения

Компонентный подход при создании программного обеспечения, используемый с середины 90-х годов XX века, предполагает построение программного обеспечения из отдельных компонентов физически отдельно существующих частей программного обеспечения, которые взаимодействуют между собой через стандартизованные двоичные интерфейсы.

Архитектура, которая укладывается в компонентный подход, наиболее жизнеспособная и практически победившая при построении комплексных информационных систем и интегрирующих информационных систем – это сервис-ориентированная архитектура.

В основе сервис-ориентированная архитектура лежит использование web-сервисов внешних, по отношению к основному программному обеспечению, модулей или программ. Web-сервисы рассматриваются в контексте интерфейсов распределённых, слабосвязанных заменяемых программных компонентов. Под web-сервисом понимается программная система, идентифицируемая строкой URI, чьи публичные интерфейсы и привязки определены и описаны посредством XML. Межсервисное взаимодействие осуществляется, в основном, с использованием протокола SOAP (Simple Object Access Protocol – простой протокол доступа к объектам) или архитектурного стиля REST (Representational State Transfer – передача состояния представления).

С середины 2010-х годов начал применяться вариант сервис-ориентированная архитектура – микросервисная архитектура, которая ориентирована на взаимодействие небольших, слабо связанных и легко изменяемых модулей (микросервисов).

4.2. Рекомендации для формирования требований к разрабатываемому программному обеспечению

Требования к разрабатываемому программному обеспечению рекомендуется оформлять в форме технического задания.

На техническое задание существует стандарт ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению». В соответствии с этим стандартом техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели (не обязательно для курсовой работы);

- стадии и этапы разработки;
- порядок контроля и приемки.

Требования к разрабатываемому программному обеспечению должны быть сформированы самостоятельно исходя из постановки задачи (выбранный вариант), реализации в сервис-ориентированной архитектуре, знаний языка программирования, на котором планируется разработка программного обеспечения.

4.3. Рекомендации для проектирования программного обеспечения в методологии объектно-ориентированного анализа и проектирования

Необходимость анализа предметной области до начала написания программы была осознана при разработке масштабных проектов. Детальный подход к проектированию разрабатываемого программного обеспечения, обеспечивает корректную реализацию выявленных бизнес-процессов.

Объектно-ориентированный анализ и проектирование (ООАП, Object-Oriented Analysis/Design) - технология разработки программных систем, в основу которых положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов.

Предметная область (domain) - часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения конкретной задачи.

Анализ предметной области, при методологии ООАП, проводится с целью выявления, классификации и формализации информации обо всех аспектах предметной области, влияющих на свойства конечного результата:

- необходимо разработать комплекс моделей предметной области, подлежащей автоматизации. Этот комплекс моделей представляет собой совокупность диаграмм, выполненных в какой-либо нотации; структурированных спецификаций, описывающих элементы модели, а также перечень нормативных и операционных документов предметной области, являющихся первоисточником информации, представленной в диаграммах и спецификациях.
- Степень подробности результатов должна быть достаточной для выполнения следующего этапа работ в соответствии с выбранным подходом и должна быть достаточной для планирования всего проекта.

Классическая схема анализа приведена на рисунке ниже. Построение функциональной модели «как есть» позволяет собрать и представить в формализованном виде информацию о существующем состоянии предметной области, после чего должно последовать переосмысление состава и

технологии, что приводит к построению модели «как надо». И наконец, когда функциональная модель «как надо» построена и все структуры данных собраны, производится построение концептуальной модели данных.



Для решения поставленной задачи необходимо формализовать:

- критерии глубины обследования;
- процессы верификации и интеграции частных результатов;
- выявление интегрирующей основы проекта в условиях большой размерности исходной информации.

Критерий глубины обследования заключается в том, что необходимо представить всю совокупность используемой информации в виде, пригодном для верификации. При этом речь не идет о полной формализации атрибутивного состава каждого документа.

При процессе анализа предметной области рекомендуется ввести два глоссария: глоссарий документов и глоссарий сущностей.

Глоссарий документов представляет собой полный перечень документов предметной области, используемых в модели предметной области до их формализации и обобщения в абстрактные категории, применяемые при разработке информационного и программного обеспечения.

Глоссарий сущностей является полным перечнем информационных сущностей предметной области, которые возникают в результате формализации информационного содержания документов. Формирование глоссария сущностей начинается только после того, как глоссарий документов считается сформированным. Таким образом, глоссарий сущностей формируется после построения функциональной модели "как надо" и до построения концептуальной модели данных. На основе требований, предъявляемых к системе, идентифицируют все сущности. Сущность определяется как некоторый объект, представляющий интерес. Для выявления сущностей рекомендуется применять морфологический анализ требований - потенциально сущностями являются все существительные, входящие в постановку задачи и относящиеся к предметной области.

Анализ проводится следующим образом:

- поступает исходная формулировка задач,
- изучаются и анализируются понятия, на основе которых строится деятельность в данной области, составляется тезаурус. Изучаются тенденции развития и проблемы, стоящие на пути развития,

• исследуются действующие процессы. Основная задача этого этапа формулируется в выявлении переменных, которыми владеют и в определении терминов, которыми оперируют,

- изучение информационных потоков и используемых методов,
- разрабатываются документы.

В результате проведенного анализа появляются:

- словарь данных системы – глоссарий,
- определение функциональных требований к системе,
- определение нефункциональных требований к системе,
- проектные ограничения.

В рамках ООАП исторически рассматривались три графических нотации:

- диаграммы "сущность-связь" (Entity-Relationship Diagrams, ERD),
- диаграммы функционального моделирования (Structured Analysis and Design Technique, SADT),
- диаграммы потоков данных (Data Flow Diagrams, DFD).

Недостаток исторических нотаций связан с отсутствием явных средств для объектно-ориентированного представления моделей сложных систем, а также сложных алгоритмов обработки данных. Поскольку исторические типы диаграмм не указываются характеристики времени выполнения отдельных процессов и передачи данных между процессами, то модели систем, реализующих синхронную обработку данных, не могут быть адекватно представлены в этих нотациях.

Все эти особенности методов структурного системного анализа ограничили возможности широкого применения соответствующих нотаций и послужили основой для разработки унифицированного языка моделирования UML.

В нотации UML полная модель сложной системы представляет собой определенное число взаимосвязанных представлений, каждое из которых адекватно отражает аспект поведения или структуры системы. При этом наиболее общими представлениями сложной системы принято считать статическое и динамическое.

Принцип иерархического построения моделей сложных систем предписывает рассматривать процесс построения моделей на разных уровнях абстрагирования или детализации в рамках фиксированных представлений.

Уровень представления (layer) – способ организации и рассмотрения модели на одном уровне абстракции, который представляет горизонтальный срез архитектуры модели, в то время как разбиение представляет ее вертикальный срез.

При этом первоначальная модель сложной системы имеет наиболее общее представление и относится к концептуальному уровню. Такая мо-

дель, получившая название концептуальной, строится на начальном этапе проектирования и может не содержать многих деталей и аспектов моделируемой системы. Последующие модели конкретизируют концептуальную модель, дополняя ее представлениями логического и физического уровня.

В нотации UML процесс ООАП рассматривается как последовательный переход от разработки наиболее общих моделей и представлений концептуального уровня к более частным и детальным представлениям логического и физического уровня. При этом на каждом этапе ООАП данные модели последовательно дополняются все большим количеством деталей, что позволяет им более адекватно отражать различные аспекты конкретной реализации сложной системы. Общая схема взаимосвязей моделей ООАП представлена на рисунке ниже.



В нотации UML все представления о модели сложной системы фиксируются в виде специальных графических конструкций, получивших название диаграмм. Нотация канонических диаграмм - основное средство разработки моделей на языке UML.

Диаграмма (diagram) – графическое представление совокупности элементов модели в форме связанного графа, вершинам и ребрам (дугам) которого приписывается определенная семантика.

В нотации языка UML определены следующие виды канонических диаграмм:

- вариантов использования (use case diagram) представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм.
- классов (class diagram) - логическая модель, отражающая статические аспекты структурного построения сложной системы.

- кооперации (collaboration diagram) разновидности логической модели, которые отражают динамические аспекты функционирования сложной системы.

- последовательности (sequence diagram) для того же.
- состояний (statechart diagram) предназначены для моделирования поведения системы.

- деятельности (activity diagram) для того же.
- компонентов (component diagram) служат для представления физических компонентов сложной системы и поэтому относятся к ее физической модели.

- развертывания (deployment diagram) для того же.

Совокупность построенных канонических диаграмм является самодостаточной в том смысле, что в них содержится вся информация, которая необходима для реализации проекта сложной системы.

Из документации по вариантам использования аналитики и потребители смогут понять, что будет делать готовая система. Изучив ту же документацию, технические писатели смогут приступить к написанию руководства для пользователей и к подготовке планов по их обучению.

Из диаграмм последовательности и кооперативных диаграмм аналитики и разработчики уяснят, насколько логично работает система, поймут ее объекты и сообщения между ними.

С помощью документации по вариантам использования, а также диаграмм последовательности и кооперативных диаграмм специалисты по контролю качества смогут получить информацию, необходимую им для написания тестовых сценариев.

С помощью диаграмм классов и состояний разработчики получают представление о фрагментах системы и их взаимодействии друг с другом.

Из диаграмм компонентов и размещения эксплуатационный персонал сможет узнать, какие компоненты будут созданы, а также где в сети они должны быть размещены.

Кроме графических элементов, которые определены для каждой канонической диаграммы, на них может быть изображена текстовая информация, которая расширяет семантику базовых элементов. В языке UML предусмотрены три специальных механизма расширения, которые включают в себя следующие конструкции:

- стереотип (stereotype) – новый тип элемента модели, который расширяет семантику метамодели. Стереотипы должны основываться на уже существующих и описанных в метамодели языка UML типах или классах,
- помеченное значение (tagged value) – явное определение свойства как пары "имя – значение". В помеченном значении само имя называют тегом (tag),
- ограничение (constraint) – некоторое логическое условие, ограничивающее семантику выбранного элемента модели.

При графическом изображении диаграмм следует придерживаться следующих основных рекомендаций:

- Каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области. Необходимо учесть все сущности, важные с точки зрения контекста данной модели и диаграммы. Отсутствие тех или иных элементов на диаграмме служит признаком неполноты модели и может потребовать ее последующей доработки.

- Все сущности на диаграмме модели должны быть одного уровня представления. Должна быть обеспечена согласованность не только имен одинаковых элементов, но и возможности вложения отдельных диаграмм друг в друга для достижения полноты представлений. В случае достаточно сложных моделей систем желательно придерживаться стратегии последовательного уточнения или детализации отдельных диаграмм.

- Вся информация о сущностях должна быть явно представлена на диаграммах. В языке UML при отсутствии некоторых символов на диаграмме могут быть использованы их значения по умолчанию, тем не менее, необходимо стремиться к явному указанию свойств всех элементов диаграмм.

- Диаграммы не должны содержать противоречивой информации. Противоречивость модели может служить причиной серьезных проблем при ее реализации и последующем использовании на практике.

- Каждая диаграмма должна быть самодостаточной для правильной интерпретации всех ее элементов и понимания семантики всех используемых графических символов. Общие фрагменты диаграмм могут уточняться или детализироваться на других диаграммах этого же типа, образуя вложенные или подчиненные диаграммы.

- Количество типов диаграмм для конкретной модели приложения строго не фиксировано. Для простых приложений нет необходимости строить все без исключения типы диаграмм. Некоторые из них могут просто отсутствовать в проекте системы, и это не будет считаться ошибкой разработчика. Перечень диаграмм зависит от специфики конкретного проекта системы.

- Любая модель системы должна содержать только те элементы, которые определены в нотации языка UML. Как показывает практика, этих конструкций вполне достаточно для представления большинства типовых проектов программных систем. Не допускается переопределение семантики тех элементов, которые отнесены к базовой нотации метамодели языка UML.

4.4. Рекомендации для реализации программного обеспечения в сервис-ориентированной архитектуре

При реализации, в рамках выполнения курсовой работы, программного обеспечения помимо традиционного подхода к его разработке, необходимо учитывать требования, оформленные в техническом задании, и сервис-ориентированную архитектуру, в которой должен быть реализован проект.

Все информационное взаимодействие разработанного программного обеспечения с внешними информационными системами должно осуществляться посредством web-сервисов. Рекомендуется использовать архитектурный стиль REST (RESTful API).

Реализация хранения табличных данных. Для организации хранения табличных данных рекомендуется использовать свободно распространяемое программное обеспечение PostgreSQL (<https://www.postgresql.org/>). PostgreSQL – это объектно-реляционная база данных с открытым исходным кодом, которая использует и расширяет язык SQL в сочетании со многими функциями, которые обеспечивают безопасное хранение данных и эффективное масштабирование при увеличении рабочих нагрузок.

Рекомендации для реализации хранения пространственных данных. Для организации хранения пространственных данных (данных, имеющие пространственную привязку) рекомендуется использовать свободно распространяемое программное обеспечение PostGIS (<https://postgis.net/>). PostGIS – это расширение с открытым исходным кодом для объектно-реляционной базы данных PostgreSQL, которое добавляет дополнительные типы данных (геометрия, география, растр и другие) и функции, операторы и индексы, которые применяются непосредственно к этим пространственным типам данных. Эти дополнительные функции, операторы, привязки и типы индексов расширяют возможности базовой СУБД PostgreSQL, делая ее быстрой, многофункциональной и надежной системой управления пространственными базами данных.

Рекомендации для реализации ГИС-интерфейса пользователя. Для реализации ГИС-интерфейса пользователя рекомендуется использовать API Яндекс.Карт (<https://tech.yandex.ru/maps%20/>) или OpenStreetMap API (https://wiki.openstreetmap.org/wiki/API_v0.6/). API Яндекс и OpenStreetMap – это набор сервисов, которые позволяют использовать картографические данные и технологии Яндекса или OpenStreetMap.

Рекомендации для реализации геокодирования данных. Для реализации механизма геокодирования данных рекомендуется использовать

свободно распространяемое программное обеспечение Elasticsearch (<https://www.elastic.co/products/elasticsearch>). Elasticsearch – это поисковая система для организации сложного поиска по базе документов, координатам и т.д. с учетом морфологии языка.

Рекомендации для реализации пространственных операций над данными. Для реализации пространственных операций над данными рекомендуется использовать свободно распространяемое программное обеспечение PostGIS (<https://postgis.net/>). Поддерживаемые расширением типы пространственных данных и пространственные индексы позволяют выполнить такие операции на уровне СУБД.

Рекомендации для реализации контекстного поиска. Для реализации контекстного поиска данных рекомендуется использовать свободно распространяемое программное обеспечение Apache Lucene (<http://lucene.apache.org/>). Apache Lucene – это технология индексирования и поиска данных на основе Java. Она также обеспечивает проверку орфографии, выделение совпадений и расширенные возможности анализа данных.

Рекомендации для реализации автоматического распознавания текста. Для реализации автоматического распознавания текста рекомендуется использовать свободно распространяемое программное обеспечение – Tesseract (<https://github.com/tesseract-ocr/>). Tesseract – это программное обеспечение распознавания текстов, поддержка языков осуществляется с помощью дополнительных модулей.

Рекомендации для реализации пакетной загрузки файлов, содержащих пространственные данные. Для чтения файлов для реализации пакетной загрузки данных рекомендуется использовать библиотеку GDAL (<https://www.gdal.org/>). Библиотека GDAL – это библиотека для 155 растровых и 95 векторных геопространственных форматов данных, она представляет единую растровую модель абстрактных данных и единую векторную модель абстрактных данных вызывающему приложению для всех поддерживаемых форматов. Библиотека поставляется с множеством полезных утилит для перевода и обработки данных.

Рекомендации для реализации организации хранения исходных файлов. Для организации хранения исходных кодов использовать свободно распространяемое программное обеспечение GitLab (<https://gitlab.com/>). GitLab – это система управления репозиториями исходного кода и отслеживания ошибок, система, обеспечивающая поддержку полного жизненного цикла программного обеспечения.

Рекомендации для комплексного тестирования. Комплексное тестирование разработанного в сервис-ориентированной архитектуре программного обеспечения рекомендуется проводить с использованием свободно распространяемого программного обеспечения Apache JMeter (<https://jmeter.apache.org/>). Apache JMeter – это программное обеспечение, предназначенное для загрузки функционального поведения теста и измерения производительности разработанного приложения. Он может быть использован для тестирования производительности как на статических, так и на динамических ресурсах, веб-динамических приложениях. Его можно использовать для имитации большой нагрузки на сервер, группу серверов, сеть или объект, чтобы проверить его надежность или проанализировать общую производительность при различных типах нагрузки.

Рекомендации для реализации мониторинга системы. Реализацию мониторинга разрабатываемой системы рекомендуется вести с использованием свободно распространяемого программного обеспечения Zabbix (<https://www.zabbix.com/>). Zabbix – это программное обеспечение, предназначенное для мониторинга в реальном времени миллионов метрик, собранных с десятков тысяч серверов, виртуальных машин и сетевых устройств. Мониторинг серверов, виртуальных машин и сетевых устройств осуществляется за счет установки на них Zabbix агента.

Рекомендации для разработки рабочей документации. Рабочую документацию рекомендовано оформлять с использованием стандарта ГОСТ 34. Несмотря на то, что стандарт не обязателен к использованию, тем не менее, полнота представленных в нем документов обеспечивает целостное представление о разрабатываемой автоматизированной системе и всех видах ее компонентов, в том числе программное обеспечение и базы данных.

Для разработки рабочей документации рекомендуется использовать разделы 6 и 7 стандарта:

- Рабочая документация:
 - разработка рабочей документации на систему и ее части;
 - разработка или адаптация программ (не обязательно для курсовой работы).
- Ввод в действие:
 - подготовка объекта автоматизации к вводу АС в действие (не обязательно для курсовой работы);
 - подготовка персонала (не обязательно для курсовой работы);
 - комплектация АС поставляемыми изделиями (не обязательно для курсовой работы);
 - строительно-монтажные работы (не обязательно для курсовой работы);

- пуско-наладочные работы;
- предварительные испытания (не обязательно для курсовой работы);
- опытная эксплуатация (не обязательно для курсовой работы);
- приемочные испытания.

Порядок выполнения работы

1. Ознакомьтесь с теоретическими основами разработки программного в настоящих указаниях, а также в конспектах лекций.
2. Получите вариант задания у преподавателя.
3. Сформулируйте основные требования к разрабатываемому программному обеспечению.
4. Разработайте программное обеспечение согласно варианту задания.
5. Отладьте программное обеспечение.
6. Проведите тестирование программного обеспечения
7. Покажите результаты работы программного обеспечения преподавателю.
8. Оформите курсовую работу.

Содержание курсовой работы

Курсовая работа должна содержать следующие сведения:

- название и цель работы;
- постановка задачи преподавателем, вариант задания;
- сформулированные требования на разработку программного обеспечения;
- результаты проектирования разрабатываемого программного обеспечения;
- руководство пользователя разрабатываемого программного обеспечения;
- результаты тестирования разрабатываемого программного обеспечения;
- листинг разрабатываемого программного обеспечения с комментариями.

Варианты заданий

1. Разработка комплексной информационной системы по управлению документами в части организации хранения пространственных данных;
2. Разработка комплексной информационной системы по управлению документами в части создания ГИС-интерфейса пользователя;
3. Разработка комплексной информационной системы по управлению документами в части мониторинга системы;

4. Разработка комплексной информационной системы по управлению документами в части автоматического распознавания текста;

5. Разработка комплексной информационной системы по управлению документами в части реализации пакетной загрузки файлов, содержащих пространственные данные;

6. Разработка комплексной информационной системы по управлению документами в части организации хранения исходных файлов;

7. Разработка комплексной информационной системы по управлению документами в части реализации пространственных операций над данными;

8. Разработка комплексной информационной системы по управлению документами в части создания интерфейса пользователя (без ГИС компоненты);

9. Разработка комплексной информационной системы по управлению документами в части организации контекстного поиска;

10. Разработка комплексной информационной системы по управлению документами в части комплексного тестирования и рабочей документации;

11. Разработка комплексной информационной системы по управлению документами в части организации хранения табличных данных;

12. Разработка комплексной информационной системы по управлению документами в части геокодирования данных.

5. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Котляров, Всеволод Павлович. Основы тестирования программного обеспечения / учебное пособие
2. Непейвода, Николай Николаевич. Стили и методы программирования / учебное пособие рекомендовано для студентов вузов, обучающихся по специальностям в области информационных технологий
3. Леоненков, Александр Васильевич. Объектно-ориентированный анализ и проектирование с использованием UML и IBM RationalRose учебное пособие
4. Гагарина Л. Г., Кокорева Е. В., Виснадуд Б. Д. Технология разработки программного обеспечения: учебное пособие / под ред. Л. Г. Гагариной. – М: ИД «ФОРУМ»: ИНФРА-М,– (Высшее образование)

Учебное издание

**ТЕХНОЛОГИЯ РАЗРАБОТКИ
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Методические рекомендации
по выполнению курсовых работ (проектов)

Составитель
УВАРОВ Владислав Вадимович

В авторской редакции

Подписано в печать 03.02.2021. Формат 60х90 1/16. Усл. печ. л. 1,0.
Тираж 100 экз. Заказ № 20-1496.

Библиотечно-издательский комплекс
федерального государственного бюджетного образовательного
учреждения высшего образования
«Тюменский индустриальный университет».
625000, Тюмень, ул. Володарского, 38.

Типография библиотечно-издательского комплекса.
625039, Тюмень, ул. Киевская, 52.