**Правительство Российской Федерации**

**Санкт-Петербургский государственный университет**

**Р А Б О Ч А Я П Р О Г Р А М М А**

**УЧЕБНОЙ ДИСЦИПЛИНЫ**

Проектирование программного обеспечения

Software Design

**Язык(и) обучения**

русский

Трудоемкость в зачетных единицах: 3

Регистрационный номер рабочей программы: 053904

Санкт-Петербург

2020

**Раздел 1. Характеристики учебных занятий**

**1.1. Цели и задачи учебных занятий**

Цель изучения дисциплины: ознакомление обучающихся с основными принципами проектирования крупных программных продуктов и комплексов.  
Задачи обучения:

1. Приобретение знаний об архитектуре программного обеспечения в целом, роли архитектуры в жизненном цикле программного обеспечения, целях и содержании деятельности архитектора.
2. Приобретение навыков объектно-ориентированного проектирования, проектирования информационных систем и распределённых приложений.
3. Изучение основных архитектурных стилей, типовых приёмов проектирования, шаблонов проектирования и архитектурных шаблонов.
4. Получение представления о языке UML и других формальных языках как основных средствах описания архитектуры программного обеспечения.

**1.2. Требования подготовленности обучающегося к освоению содержания учебных занятий (пререквизиты)**

Программа курса предназначена для обучающихся 4 курса бакалавриата, освоивших программу курсов «Программирование» и «Информатика» (или аналогичных им). Максимальная эффективность программы будет обеспечена при условии, что обучающийся имел опыт разработки промышленных или исследовательских программных продуктов.

**1.3. Перечень результатов обучения (learning outcomes)**

Результатами обучения являются знание содержания программы курса и представление о возможностях применения ее разделов в практической деятельности, а именно:

1. Умение самостоятельно выбрать подход к проектированию программной системы, создать или описать имеющуюся архитектуру программного продукта.
2. Знание достоинств и недостатков основных существующих подходов к проектированию ПО.
3. Владение типовыми шаблонами проектирования объектно-ориентированных программ.
4. Владение языком UML на уровне, достаточном для создания архитектурных диаграмм и понимания диаграмм, созданных коллегами.
5. Умение оформить и изложить результат проектирования программного обеспечения в виде архитектурного документа или набора документов.
6. Владение принципами проектирования распределённых приложений.

Курс способствует формированию следующих компетенций:

* ПКА-1 – способен применять фундаментальные знания, полученные в области математики, программирования и информационных технологий, и использовать их в профессиональной деятельности;
* ПКА-3 – способен применять современные информационные технологии, в том числе и отечественные, при создании программных продуктов и программных комплексов различного назначения;
* ПКА-4 – способен участвовать в разработке технической документации программных продуктов и программных комплексов с использованием стандартов, норм и правил;
* ПКА-5 – способен инсталлировать и сопровождать программное обеспечение информационных систем и баз данных, в том числе и отечественного происхождения;
* ПКП-1 – способен демонстрировать базовые знания математических и естественных наук, программирования и информационных технологий;
* ПКП-2 – способность проводить под научным руководством исследование на основе существующих методов в конкретной области профессиональной деятельности;
* ПКП-3 – способен решать задачи в области развития науки, техники и технологии с учетом нормативного правового регулирования в сфере интеллектуальной собственности;
* ПКП-5 – способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях;
* ПКП-6 – способен использовать основные методы и средства автоматизации проектирования, реализации, испытаний и оценки качества при создании конкурентоспособного программного продукта и программных комплексов, а также способен использовать методы и средства автоматизации, связанные с сопровождением, администрированием и модернизацией программных продуктов и программных комплексов;
* ПКП-7 – способен использовать знания направлений развития компьютеров с традиционной (нетрадиционной) архитектурой; современных системных программных средств: операционных систем, операционных и сетевых оболочек, сервисных программ; тенденции развития функций и архитектур проблемно-ориентированных программных систем и комплексов в профессиональной деятельности;
* ПКП-9 – способен использовать современные методы разработки и реализации конкретных алгоритмов математических моделей на базе языков программирования и пакетов прикладных программ моделирования;
* УКБ-10 – способен понимать сущность и значение информации в развитии общества, использовать основные методы получения и работы с информацией с учетом современных технологий цифровой экономики и информационной безопасности.

**1.4. Перечень и объём активных и интерактивных форм учебных занятий**

Основной формой обучения проектированию программного обеспечения являются лекционные занятия в аудитории, причём некоторые лекционные занятия проводятся в интерактивном формате, предполагающим активную дискуссию обучающихся и преподавателя.

**Раздел 2. Организация, структура и содержание учебных занятий**

**2.1. Организация учебных занятий**

**2.1.1 Основной курс**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Трудоёмкость, объёмы учебной работы и наполняемость групп обучающихся | | | | | | | | | | | | | | | | | | |
| Код модуля в составе дисциплины,  практики и т.п. | Контактная работа обучающихся с преподавателем | | | | | | | | | | | | Самостоятельная работа | | | | Объём активных и интерактивных  форм учебных занятий | Трудоёмкость |
| лекции | семинары | консультации | практические  занятия | лабораторные работы | контрольные работы | коллоквиумы | текущий контроль | промежуточная  аттестация | итоговая аттестация | под руководством преподавателя | в присутствии  преподавателя | сам. раб. с использованием  методических материалов | текущий контроль (сам.раб.) | промежуточная аттестация (сам.раб.) | итоговая аттестация  (сам.раб.) |
| ОСНОВНАЯ ТРАЕКТОРИЯ | | | | | | | | | | | | | | | | | | |
| Форма обучения: очная | | | | | | | | | | | | | | | | | | |
| Семестр 7 | 30 |  | 2 |  |  |  |  | 2 | 2 |  |  |  | 43 |  | 29 |  | 4 | 3 |
|  | 2-100 |  | 2-100 |  |  |  |  | 2-100 | 2-100 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| ИТОГО | 30 |  | 2 |  |  |  |  | 2 | 2 |  |  |  | 43 |  | 29 |  |  | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Виды, формы и сроки текущего контроля успеваемости и промежуточной аттестации | | | |
| Период обучения (модуль) | Формы текущего контроля успеваемости | Виды промежуточной аттестации | Виды итоговой аттестации  (только для программ итоговой аттестации и дополнительных образовательных программ) |
| ОСНОВНАЯ ТРАЕКТОРИЯ | | | |
| Форма обучения: очная | | | |
| Семестр 7 |  | экзамен |  |

**2.2. Структура и содержание учебных занятий**

Период обучения (модуль): семестр 7.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Введение в проектирование ПО | лекции | 2 |
| по методическим материалам | 3 |
| II. | Объектно-ориентированное проектирование | лекции | 2 |
| по методическим материалам | 3 |
| III. | Моделирование, язык UML | лекции | 6 |
| по методическим материалам | 8 |
| IV. | Шаблоны проектирования | лекции | 8 |
| по методическим материалам | 10 |
| V. | Архитектурные стили | лекции | 2 |
| по методическим материалам | 4 |
| VI. | Предметно-ориентированное проектирование, проектирование информационных систем | лекции | 4 |
| текущий контроль | 2 |
| по методическим материалам | 6 |
| VII. | Проектирование распределённых приложений | лекции | 4 |
| по методическим материалам | 6 |
| VIII. | Примеры архитектур | лекции | 2 |
| по методическим материалам | 3 |
| IX. | Промежуточная аттестация | консультация | 2 |
| экзамен | 8 |

Раздел 1: Введение в проектирование ПО.

1. Программа и программное обеспечение. Понятие архитектуры. Роль проектирования и архитектора в производстве ПО. Понятие архитектурного стиля, архитектурного вида, деградации архитектуры. Design Document, стандарт IEEE 1016-2009. Архитектура и жизненный цикл ПО.

Раздел 2: Объектно-ориентированное проектирование.

1. Сложность, существенная и случайная. Восходящее и нисходящее проектирование. Модульность, сопряжение и связность. Понятие объекта, абстракция, инкапсуляция, наследование, композиция, мутабельность. Методы выделения объектов. Принципы SOLID. Закон Деметры. Абстрактные типы данных, основные принципы абстракции, некоторые принципы написания качественного объектно-ориентированного кода.

*Домашнее задание 1: проектирование CLI*.

Раздел 3: Моделирование, язык UML.

1. Модели в различных областях человеческой деятельности. Роль моделирования в процессе разработки. Виды моделей. Язык UML и его назначение, предпосылки появления. Виды диаграмм. Диаграммы классов, пакетов, объектов и компонентов.
2. Моделирование требований, случаи использования, диаграмма случаев использования UML. Другие виды диаграмм: контекстная диаграмма IDEF0, диаграмма характеристик, дерево характеристик, диаграмма требований SysML. Моделирование бизнес-процессов: диаграмма активностей UML, язык BPMN. Диаграмма развёртывания UML. Моделирование данных: диаграмма «сущность-связь», ORM-диаграммы. Понятие CASE-системы.
3. Диаграммы состояний UML и их применение для описания поведения реактивных систем. Прочие диаграммы UML: Диаграммы последовательности, коммуникационные диаграммы, диаграммы составных структур, диаграммы коопераций, временные диаграммы, диаграммы обзора взаимодействия. Диаграммы потоков данных. Диаграммы IDEF0.

Раздел 4: Шаблоны проектирования.

1. Понятие шаблона проектирования. Структурные шаблоны. Рекурсивная композиция, шаблоны «Компоновщик» и «Декоратор». Выделение алгоритма в объект, шаблон «Стратегия». Сокрытие реализации, шаблон «Адаптер». Проектирование удаленного и отложенного выполнения, шаблон «Заместитель».
2. Инкапсуляция подсистемы, шаблон «Фасад». Разделение ответственности по иерархиям наследования, шаблон «Мост». Оптимизация количества объектов в системе, шаблон «Приспособленец». Порождающие шаблоны: «Абстрактная фабрика», «Фабричный метод», «Одиночка», «Ленивая инициализация», «Пул объектов». Клонирование объектов, шаблон «Прототип». Поэтапное конструирование объекта, шаблон «Строитель».
3. Поведенческие шаблоны. Шаблон «Цепочка обязанностей». Отложенное выполнение. Шаблон «Команда». Подходы к реализации поведения, основанного на модели конечного автомата, шаблон «Состояние». Подходы к обходу сложных структур данных, шаблон «Посетитель», шаблон «Итератор».
4. Антипаттерны. Понятие антипаттерна. Антипаттерны реализации: «Круговая зависимость», «Последовательное сопряжение», «Вызов предка», «Проблема йо-йо», «Активное ожидание», «Сокрытие ошибки», «Магические числа», «Магические строки». Антипаттерны проектирования: «Божественный объект», «Поток лавы», «Функциональная декомпозиция», «Полтергейст», «Золотой молоток». Понятие Design smell. Архитектурные антипаттерны: «Остров автоматизации», «Система печной трубы», «Привязка к поставщику», «Подразумеваемая архитектура», «Архитектура комитетом».

Раздел 5: Архитектурные стили

1. Понятие архитектурного шаблона и архитектурного стиля. Примеры архитектурных шаблонов. Основные архитектурные стили: «Главная программа и подпрограммы», объектно-ориентированный стиль, слоистый стиль, «Клиент-сервер», «Пакетная обработка», «Каналы и фильтры», «Blackboard», стили с неявным вызовом: «Издатель-подписчик», событийно-ориентированные стили, «Peer-to-peer».

Раздел 6: Предметно-ориентированное проектирование, проектирование информационных систем.

1. Понятие предметно-ориентированного проектирования. Модель предметной области, единый язык, элементы модели: сущность, объект-значение, служба, модуль. Жизненный цикл объекта. Шаблоны поддержки жизненного цикла: «Агрегат», «Фабрика», «Репозиторий». Принципы разбиения объектов по модулям. Реализация ограничений предметной области, паттерн «Спецификация».

*Домашнее задание 2: Roguelike*.

1. Стратегические аспекты предметно-ориентированного проектирования. Понятие ограниченного контекста. Непрерывная интеграция. Подходы к интеграции контекстов: «Общее ядро», «Заказчик-поставщик», «Конформист», «Предохранительный уровень», «Отдельное существование», «Служба с открытым протоколом», «Общедоступный язык». Смысловое ядро, дистилляция, подходы к дистилляции: «Неспециализированные подобласти», «Выделенное ядро», «Связный механизм», «Абстрактное ядро». Понятие крупномасштабной структуры, метафора системы, примеры крупномасштабных структур: уровневая архитектура, уровень знаний, подключаемые компоненты. Примеры типичных уровней в информационных системах.
2. Текущий контроль: проектирование в командах по 2-3 человека модели предметной области приложения «Магазин книг» в соответствии с принципами предметно-ориентированного проектирования.

Раздел 7: Проектирование распределённых приложений.

1. Понятие распределённой системы. Виды сущностей в распределённой системе, способы организации взаимодействия, варианты размещения, типичные архитектурные стили. Сетевое взаимодействие: модель OSI, стек протоколов TCP/IP, понятие сокета, протоколы TCP и UDP, протокол HTTP. Удалённые вызовы: RPC, RMI, примеры: protobuf, gRPC.

*Домашнее задание 3: Сетевой чат*.

1. Понятие веб-сервиса. SOAP. Пример технологии разработки веб-сервисов: Windows Communication Foundation. Очереди сообщений, пример: RabbitMQ. Понятие Enterprise Service Bus. Архитектурный стиль REST. Микросервисы. Архитектурный стиль Peer-to-Peer, примеры архитектур. Развёртывание и балансировка нагрузки, пример: Docker.

Раздел 8: Примеры архитектур.

* 1. Архитектура системы контроля версий Git: основные архитектурные соображения, внутреннее представление данных, деревья, коммиты, ссылки, ветки, тэги, packfiles, reflog. Архитектура системы контроля версий Mercurial: основные архитектурные соображения, revlog, changelog, manifest, ревизии, ветки. Статическая структура Mercurial, расширяемость. Выводы. Архитектура системы контроля версий Subversion. Требования, статическая структура, представление ревизий, структура репозитория. Проблемы и ограничения, выводы.

**Раздел 3. Обеспечение учебных занятий**

**3.1. Методическое обеспечение**

**3.1.1 Методические указания по освоению дисциплины**

Для успешного усвоения дисциплины необходимо посещение лекционных занятий и самостоятельная работа с материалами лекций и рекомендованной литературой.

**3.1.2 Методическое обеспечение самостоятельной работы**

Для самостоятельной работы рекомендуется использовать рекомендованную литературу и материалы курса (презентации, конспекты), размещаемые в системе поддержки обучения.

**CLI**

В командах по два-три человека спроектировать простой интерпретатор командной строки, поддерживающий команды:

* cat [FILE] — вывести на экран содержимое файла
* echo — вывести на экран свой аргумент (или аргументы)
* wc [FILE] — вывести количество строк, слов и байт в файле
* pwd — распечатать текущую директорию
* exit — выйти из интерпретатора

При этом:

* должны поддерживаться одинарные и двойные кавычки (full and weak quoting);
* окружение (команды вида “имя=значение”), оператор $;
* вызов внешней программы через Process (или его аналоги);
  + если введено что-то, чего интерпретатор не знает;
* пайплайны (оператор «|»).

Задача выполняется в командах по 2-3 человека.

**Roguelike**

Roguelike – это довольно популярный жанр компьютерных игр, названный в честь игры Rogue, 1980 года выхода. Характеризуется:

* простой тайловой или консольной графикой;
* активным использованием случайной генерации;
* перманентной смертью персонажа и невозможностью загрузить предыдущее сохранение после смерти персонажа;
* чрезвычайно развитым набором игровых правил (чем они и интересны с точки зрения архитектуры);
* высокой свободой действий персонажа (так называемые «игры-песочницы»).

Примеры:

* https://en.wikipedia.org/wiki/NetHack
* https://en.wikipedia.org/wiki/Angband\_(video\_game)
* https://en.wikipedia.org/wiki/Ancient\_Domains\_of\_Mystery

Вашей задачей будет в командах по два-три человека провести анализ и построить модель предметной области для такой компьютерной игры согласно принципам предметно-ориентированного проектирования.

При этом должны быть выполнены следующие функциональные требования:

* персонаж игрока, способный перемещаться по карте, управляемый с клавиатуры;
* карта обычно генерируется, но для некоторых уровней грузится из файла;
* характеристики — здоровье, сила атаки и т.д.;
* экспа и уровни персонажа, с ростом уровня повышаются характеристики;
* инвентарь персонажа, включающий элементы, влияющие на его характеристики, которые можно надеть и снять;
* несколько разных видов мобов, способных перемещаться по карте;
* боевая система — движущиеся объекты, пытающиеся занять одну клетку карты, атакуют друг друга;
* консольная графика, традиционная для этого жанра игр.

**Instant Messenger**

В командах по два-три человека спроектировать и реализовать сетевой чат (наподобие Telegram) с помощью средства удалённых вызовов gRPC:

* peer-to-peer, то есть соединение напрямую;
* графический пользовательский интерфейс;
* отображение имени отправителя, даты и текста сообщения;
* при запуске указываются:
  + адрес peer-а и порт, если хотим подключиться;
  + должно быть можно не указывать, тогда работаем в режиме сервера;
  + наше имя пользователя.
* Реализация допустима на любом языке программирования из поддержанных gRPC

**3.1.3 Методика проведения текущего контроля успеваемости и промежуточной аттестации и критерии оценивания**

***3.1.3.1. Методика проведения текущего контроля успеваемости и промежуточной аттестации***

В формировании итоговой оценки по курсу участвуют домашние задания, задание текущего контроля, устный экзамен.

Домашние задания выполняются в командах по 2-3 человека (разбиение на команды выполняют сами обучающиеся, преподаватель при необходимости может переформировать команды по своему усмотрению). Задания выдаются на срок в две недели и сдаются через систему поддержки обучения. В случае, если задание не сдано до дедлайна, оно считается невыполненным и не может быть досдано. В случае, если в сданном задании есть существенные недочёты, обучающимся сообщаются замечания и задание отправляется на доработку (с дедлайном представления исправленной версии в 2 недели с момента публикации результата проверки), после чего оценивается заново. Итоговая оценка получается как максимальная из двух попыток. Критерии оценивания намеренно организованы так, чтобы домашние задания были не обязательными — сохранена возможность получить высшую оценку и без выполнения домашних заданий.

Текущий контроль успеваемости выполняется в форме проверочной работы, где прямо на занятии обучающимся предлагается разделиться на команды по 2-3 человека, выполнить задание и сдать его на проверку с помощью системы поддержки обучения. Преподаватель может переформировать команды по своему усмотрению. Общение внутри команды и использование любых источников разрешено. В случае пропуска занятия с текущим контролем задание предлагается выполнить индивидуально на экзамене, на что выделяется дополнительно два академических часа в процессе экзамена.

Экзамен проводится в устной форме. Билет состоит из двух вопросов, на подготовку ответа на которые даётся не менее одного академического часа (при подготовке можно пользоваться литературой). Преподаватель вправе задавать дополнительные вопросы по билету, чтобы проверить уровень понимания материала. После ответа на вопросы билета преподаватель вправе задать дополнительные вопросы по любой теме из списка вопросов, вынесенных на экзамен. Количество и содержание дополнительных вопросов — на усмотрение преподавателя, принимающего экзамен.

По желанию преподавателя на экзамен допустимо приглашать других преподавателей с квалификацией не ниже изложенной в п. 3.2.1 как для независимого оценивания ответов обучающихся, так и для коллегиального. В последнем случае оценка за экзамен ставится на основании усреднения оценок каждого принимающего. В спорных ситуациях преподаватель, ведущий дисциплину, имеет право принятия окончательного решения.

***3.1.3.2. Критерии оценивания итогового процента освоения дисциплины***

Итоговый процент освоения дисциплины складывается из трёх факторов: оценки за домашние задания, оценки за текущий контроль и оценки за теоретический экзамен. Домашние задания оцениваются из шкал оценивания, указанных для каждой задачи отдельно. Критерии оценивания и максимальные баллы за каждую задачу доводятся до обучающихся вместе с условием задачи. Итоговый балл за домашние задания вычисляется как сумма баллов за все задания. Для командных заданий всем членам команды ставится одинаковая оценка.

Оценка за текущий контроль ставится по шкале от 0 до 10 по критериям, приведённым в разделе 3.1.4. Всем членам команды ставится одинаковая оценка.

Оценка за экзамен ставится по следующим правилам: ответ на каждый вопрос билета и на дополнительные вопросы оценивается по шкале от 0 (нет ответа) до 10 (очень хороший ответ), далее оценка усредняется. Результат переводится в диапазон от 0 до 100.

Итоговый процент выполнения дисциплины вычисляется как взвешенная сумма оценок за домашние задания, текущий контроль и экзамен по следующей формуле: итоговая оценка = min(100, 0.3 \* (оценка за домашние задания) + 0.1 \* (оценка за текущий контроль) + 0.8 \* (оценка за экзамен)). Перед вычислением итогового процента освоения оценка за домашние задания, оценка за текущий контроль и оценка за экзамен приводятся к диапазону 0–100 линейным преобразованием. Далее применяется следующее правило выставления оценки:

|  |  |  |
| --- | --- | --- |
| Итоговый процент  выполнения, % | Оценка СПбГУ при  проведении зачёта | Оценка  ECTS |
| 90-100 | зачтено | A |
| 80-89 | зачтено | B |
| 70-79 | зачтено | C |
| 61-69 | зачтено | D |
| 50-60 | зачтено | E |
| менее 50 | не зачтено | F |

**3.1.4 Методические материалы для проведения текущего контроля успеваемости и промежуточной аттестации (контрольно-измерительные материалы, оценочные средства)**

***3.1.4.1. Формируемые дисциплиной компетенции***

**ПКА-1** – способен применять фундаментальные знания, полученные в области математики, программирования и информационных технологий, и использовать их в профессиональной деятельности.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКА-3** – способен применять современные информационные технологии, в том числе и отечественные, при создании программных продуктов и программных комплексов различного назначения.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКА-4** – способен участвовать в разработке технической документации программных продуктов и программных комплексов с использованием стандартов, норм и правил.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКА-5** – способен инсталлировать и сопровождать программное обеспечение информационных систем и баз данных, в том числе и отечественного происхождения.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-1** – способен демонстрировать базовые знания математических и естественных наук, программирования и информационных технологий.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-2** – способность проводить под научным руководством исследование на основе существующих методов в конкретной области профессиональной деятельности.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-3** – способен решать задачи в области развития науки, техники и технологии с учетом нормативного правового регулирования в сфере интеллектуальной собственности.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-5** – способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-6** – способен использовать основные методы и средства автоматизации проектирования, реализации, испытаний и оценки качества при создании конкурентоспособного программного продукта и программных комплексов, а также способен использовать методы и средства автоматизации, связанные с сопровождением, администрированием и модернизацией программных продуктов и программных комплексов.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-7** – способен использовать знания направлений развития компьютеров с традиционной (нетрадиционной) архитектурой; современных системных программных средств: операционных систем, операционных и сетевых оболочек, сервисных программ; тенденции развития функций и архитектур проблемно-ориентированных программных систем и комплексов в профессиональной деятельности.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-9** – способен использовать современные методы разработки и реализации конкретных алгоритмов математических моделей на базе языков программирования и пакетов прикладных программ моделирования.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**УКБ-10** – способен понимать сущность и значение информации в развитии общества, использовать основные методы получения и работы с информацией с учетом современных технологий цифровой экономики и информационной безопасности.

□ Формируется дисциплиной.

✓ Развивается дисциплиной.

□ Полностью сформирована по результатам освоения дисциплины.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

***3.1.4.2. Контрольно-измерительные материалы (примеры)***

***Пример списка вопросов для устного экзамена:***

1. Понятие архитектуры, профессия «Архитектор».
2. Архитектурные виды.
3. Роль архитектуры в жизненном цикле ПО.
4. Понятие декомпозиции. Модульность, связность, сопряжение, сложность.
5. Понятия класса и объекта, абстракция, инкапсуляция, наследование.
6. Принципы выделения абстракций предметной области.
7. Принципы SOLID.
8. Закон Деметры. Принципы хорошего объектно-ориентированного кода.
9. Моделирование, визуальные модели, виды моделей, метафора визуализации.
10. Язык UML. Проектирование структуры системы, диаграммы классов.
11. Диаграммы объектов, диаграммы пакетов UML.
12. Диаграммы компонентов, диаграммы развёртывания UML.
13. Диаграмма случаев использования UML.
14. Диаграмма активностей UML.
15. Диаграммы конечных автоматов UML.
16. Диаграммы последовательностей UML.
17. Диаграммы коммуникации UML.
18. Диаграммы составных структур, коопераций, временные диаграммы.
19. Диаграммы обзора взаимодействия, диаграммы потоков данных.
20. Диаграммы IDEF0, характеристик. Feature tree
21. Моделирование требований в SysML.
22. Язык BPMN.
23. Моделирование данных: диаграммы «Сущность-связь»
24. Концептуальное моделирование, диаграммы ORM.
25. Понятие и примеры CASE-систем.
26. Паттерн «Компоновщик».
27. Паттерн «Декоратор».
28. Паттерн «Стратегия».
29. Паттерн «Адаптер».
30. Паттерн «Заместитель».
31. Паттерн «Фасад».
32. Паттерн «Приспособленец».
33. Паттерн «Мост».
34. Паттерн «Фабричный метод».
35. Паттерн «Абстрактная фабрика».
36. Паттерн «Одиночка».
37. Паттерны «Ленивая инициализация» и «Пул объектов».
38. Паттерн «Прототип».
39. Паттерн «Строитель».
40. Паттерн «Посредник».
41. Паттерн «Команда».
42. Паттерн «Цепочка ответственности».
43. Паттерн «Наблюдатель».
44. Паттерн «Состояние».
45. Паттерн «Посетитель».
46. Паттерн «Хранитель».
47. Паттерн «Интерпретатор».
48. Паттерн «Спецификация».
49. Паттерн «Итератор».
50. Антипаттерны «Круговая зависимость», «Последовательная связность», «Вызов предка», «Проблема Йо-Йо».
51. Антипаттерны «Активное ожидание», «Сокрытие ошибки», «Магические числа», «Магические строки».
52. Антипаттерны «Божественный объект», «Поток лавы».
53. Антипаттерны «Функциональная декомпозиция», «Полтергейст», «Золотой молоток».
54. Антипаттерны «Остров автоматизации», «Stovepipe system».
55. Антипаттерны «Привязка к поставщику», «Подразумеваемая архитектура», «Проектирование комитетом».
56. Понятие архитектурного стиля, трёхзвенная архитектура.
57. Model-View-Controller, Sense-Compute-Control.
58. Структурный и объектно-ориентированный стили, слоистые архитектурные стили.
59. Пакетная обработка, каналы и фильтры, Blackboard.
60. Событийно-ориентированные стили, Publish-Subscribe.
61. Понятие Domain-Driven Design, единый язык, изоляция предметной области.
62. DDD, основные структурные элементы модели предметной области.
63. DDD, паттерн «Агрегат».
64. DDD, паттерны «Фабрика», «Репозиторий».
65. Говорящие интерфейсы, функции без побочных эффектов, assertions, замкнутые операции.
66. Ограниченный контекст, непрерывная интеграция, карта контекстов.
67. Подходы к интеграции контекстов.
68. Смысловое ядро, приёмы дистилляции, абстрактное ядро.
69. Крупномасштабная структура, метафора системы, разбиение по уровням.
70. Типичные уровни в производственных системах
71. Типичные уровни в финансовых системах
72. Стили «Уровень знаний», «Подключаемые компоненты».
73. Архитектура распределённых систем: понятие распределённой системы, типичные архитектурные стили.
74. Межпроцессное сетевое взаимодействие, модель OSI, стек протоколов TCP/IP, сокеты, протоколы «запрос-ответ», протокол HTTP.
75. Удалённые вызовы процедур (RPC). Protobuf, gRPC.
76. Удалённые вызовы методов (RMI).
77. Веб-сервисы, SOAP. WCF.
78. Очереди сообщений, RabbitMQ.
79. Архитектурный стиль REST.
80. Микросервисы, peer-to-peer.
81. Развёртывание и балансировка нагрузки, Docker.
82. Архитектура системы контроля версий Git.
83. Архитектура системы контроля версий Mercurial.
84. Архитектура системы контроля версий Subversion.

***Проверяемые компетенции*:** ПКА-4, ПКП-1, УКБ-10

***Критерии оценивания*:** обучающемуся даётся два билета и задаётся несколько дополнительных вопросов по курсу. Ответ на каждый вопрос билета и на дополнительные вопросы оценивается по шкале от 0 (нет ответа) до 10 (очень хороший ответ), далее оценка усредняется. Результат переводится в диапазон от 0 до 100.

***Пример задачи для текущего контроля:***

В команде из 2-3 человек выполнить анализ предметной области и построить модель в виде диаграммы классов для интернет-магазина книг по следующему ТЗ:

1. Интернет-магазин должен иметь веб-интерфейс, но он должен иметь возможность подключения через другие интерфейсы (веб-сервисы и т.п.)
2. Интернет-магазин предназначен для продажи книг, с оплатой заказов через интернет.
3. Пользователь должен иметь возможность добавить книги в онлайн корзину, после чего произвести оплату.
   1. Пользователь может убрать предметы из корзины.
4. Пользователь должен иметь возможность вести список желаемых покупок, т.е. книг, которые он хочет купить позже.
5. Пользователь должен иметь возможность отменить заказ до того, как он отправлен по почте.
6. Пользователь должен иметь возможность оплатить заказ кредитной картой или по счету на оплату.
7. Должна быть возможность вернуть книги.
8. Интернет-магазин должен встраиваться на сайты партнеров в виде мини-каталога, который составляется по основному каталогу, хранящемуся в центральной базе данных.
   1. Мини-каталог должен быть построен на основе XML для того, чтобы была возможность передавать его между системами.
9. Пользователь должен иметь возможность создать учетную запись клиента, чтобы система запоминала данные пользователя (имя, адрес, данные банковской карты и т.д.) и восстанавливала их при входе.
   1. Система должна вести основной список учетных записей в центральной базе данных.
   2. При входе пользователя его пароль должен сверяться с паролем в основном списке паролей, сохраненным в базе данных.
10. Пользователь должен иметь возможность искать книги различными способами поиска – по заголовку, по автору, ключевому слову или категории и после поиска просматривать детальное описание книги.
11. Пользователь должен иметь возможность оставлять отзывы на понравившиеся книги. Оставленные отзывы должны появляться в детальном описании книги. Отзыв должен включать выставленный клиентом рейтинг (1-5), который должен показываться вместе с заголовком книги в списке книг.
    1. Отзывы на книгу должны модерироваться, т.е. им должен присваиваться статус Ok кем-то из администраторов прежде, чем они появятся на сайте.
    2. Длинные отзывы должны обрезаться при выводе детального описания книги. Клиент может щелкнуть по отзыву, чтобы просмотреть полный отзыв на отдельной странице.
12. Должна быть возможность размещения администраторами редакторских отзывов. Они также должны появляться на странице с детальным описанием книги. Редакторские отзывы не имеют рейтинга и не модерируются.
13. Интернет-магазин должен позволять сторонним продавцам (например, магазинам подержанных книг) добавлять свои каталоги книг в основной каталог книг, так чтобы книги этих продавцов присутствовали в результатах поиска.
14. Интернет-магазин должен быть масштабируем со следующими требованиями:
    1. Должна быть возможность управлять до 100 тыс. учетных записей пользователей за первые 6 месяцев работы и затем до 1 млн. пользователей.
    2. Должна быть возможность обслуживать одновременно 1000 посетителей (до 10000 тысяч после 6 месяцев)
    3. Система должна обслуживать 100 поисковых запросов в минуту (1 тыс./мин. после 6 месяцев)
    4. Система должна обслуживать 100 покупок в час (1 тыс./час после 6 мес.)

Обратите внимание, что это должна быть модель предметной области, детали реализации наподобие способа хранения информации в базе данных не важны. Будет оцениваться точность следования ТЗ, соответствие модели сущностям предметной области (в том числе, неявным) и пунктуальность в следовании синтаксису UML. Задание сдаётся в виде файла с диаграммой классов UML в формате .pdf, либо в виде ссылки на проект с диаграммой в каком-либо из существующих онлайн-сервисов рисования диаграмм (например, <https://app.diagrams.net/>).

***Проверяемые компетенции*:** ПКА-1, ПКА-3, ПКА-4, ПКА-5, ПКП-1, ПКП-2, ПКП-3, ПКП-5, ПКП-6, ПКП-7, ПКП-9.

***Критерии оценивания*:** решение оценивается по шкале от 0 (нет решения) до 10 (продемонстрировано знание формальных нотаций, умение выполнить грамотный анализ требований и декомпозицию предметной области, разработана и грамотно описана качественная архитектура).

***Примеры домашних заданий:***

**Задание 1: проектирование CLI**

В командах по два-три человека спроектировать простой интерпретатор командной строки, поддерживающий команды:

* cat [FILE] — вывести на экран содержимое файла;
* echo — вывести на экран свой аргумент (или аргументы);
* wc [FILE] — вывести количество строк, слов и байт в файле;
* pwd — распечатать текущую директорию;
* exit — выйти из интерпретатора.

При этом должны поддерживаться:

* одинарные и двойные кавычки (full and weak quoting);
* окружение (команды вида “имя=значение”), оператор $;
* вызов внешней программы, если введено что-то, чего интерпретатор не знает;
* пайплайны (оператор «|»).

Примеры:

>echo "Hello, world!"

Hello, world!

> FILE=example.txt

> cat $FILE

Some example text

> cat example.txt | wc

1 3 18

> echo 123 | wc

1 1 3

> x=ex

> y=it

> $x$y

Решение должно удовлетворять следующим нефункциональным требованиям:

* легко добавлять новые команды;
* чёткое разграничение ответственности между элементами архитектуры;
* наличие архитектурного описания.

Задача выполняется в командах по 2-3 человека в течение двух недель с момента публикации условия. Результатом должна являться структурная диаграмма (например, диаграмма классов UML), описывающая систему, и текстовое описание того, как спроектированное приложение должно работать. Решение в виде .pdf-файла или ссылки на документ в каком-либо из облачных сервисов хранения сдаётся через систему поддержки обучения.

***Проверяемые компетенции*:** ПКА-1, ПКА-3, ПКА-4, ПКА-5, ПКП-1, ПКП-2, ПКП-3, ПКП-5, ПКП-6, ПКП-7, ПКП-9.

***Критерии оценивания*:** решение оценивается по шкале от 0 (нет решения) до 10 (продемонстрировано умение выполнить грамотную декомпозицию предметной области, разработана и грамотно описана качественная архитектура, выполняющая все требования условия).

**Задание 2: Roguelike**

Roguelike — это довольно популярный жанр компьютерных игр, названный в честь игры Rogue, 1980 года выхода. Характеризуется:

* простой тайловой или консольной графикой;
* активным использованием случайной генерации;
* перманентной смертью персонажа и невозможностью загрузить предыдущее сохранение после смерти персонажа;
* чрезвычайно развитым набором игровых правил (чем они и интересны с точки зрения архитектуры);
* высокой свободой действий персонажа (так называемые «игры-песочницы»).

Примеры:

* https://en.wikipedia.org/wiki/NetHack
* https://en.wikipedia.org/wiki/Angband\_(video\_game)
* https://en.wikipedia.org/wiki/Ancient\_Domains\_of\_Mystery

Вашей задачей будет в командах по два-три человека провести анализ и построить модель предметной области согласно принципам предметно-ориентированного проектирования для такой компьютерной игры.

При этом должны быть выполнены следующие функциональные требования:

* персонаж игрока, способный перемещаться по карте, управляемый с клавиатуры;
* карта обычно генерируется, но для некоторых уровней грузится из файла;
* характеристики — здоровье, сила атаки и т.д.;
* экспа и уровни персонажа, с ростом уровня повышаются характеристики;
* инвентарь персонажа, включающий элементы, влияющие на его характеристики, которые можно надеть и снять;
* несколько разных видов мобов, способных перемещаться по карте;
* боевая система — движущиеся объекты, пытающиеся занять одну клетку карты, атакуют друг друга;
* консольная графика, традиционная для этого жанра игр.

Также требуется:

* использовать шаблон «Стратегия» для поддержки различных поведений мобов (агрессивного, трусливого, пассивного);
* использовать шаблон «Состояние» для динамического переключения поведения мобов
  + мобы с низким здоровьем должны переключаться в трусливый режим;
  + по мере восстановления здоровья переходить в исходный.
* использовать шаблон «Абстрактная фабрика» для создания мобов и предметов на карте;
* использовать шаблон «Команда» для поддержки взаимодействия с пользователем.

На что обратить внимание:

* на разделение системы на компоненты — решения вида «большой клубок классов» будут оценены очень низко;
* на прослеживаемость потока управления — должно быть понятно, с какого места запускается программа, кто кому передаёт управление;
* что все имеют необходимые для своей работы данные — например, стратегии поведения мобов знают про карту;
* на баланс детальности и читаемости диаграммы — она должна быть достаточно детальна, чтобы при реализации не требовалось принимать серьёзных архитектурных решений, но при этом обозрима. Например, старательно выписывать все методы классов и все поля не надо, но ключевые методы и поля всё-таки нужны.

Задача выполняется в командах по 2-3 человека в течение двух недель с момента публикации условия. Результатом должна являться структурная диаграмма (например, диаграмма классов UML), описывающая систему, и текстовое описание того, как спроектированное приложение должно работать. Решение в виде .pdf-файла или ссылки на документ в каком-либо из облачных сервисов хранения сдаётся через систему поддержки обучения.

***Проверяемые компетенции*:** ПКА-1, ПКА-3, ПКА-4, ПКА-5, ПКП-1, ПКП-2, ПКП-3, ПКП-5, ПКП-6, ПКП-7, ПКП-9.

***Критерии оценивания*:** решение оценивается по шкале от 0 (нет решения) до 10 (продемонстрировано владение нотацией, умение выполнить грамотную декомпозицию предметной области, разработана и грамотно описана качественная архитектура, выполняющая все требования условия).

**Задание 3: сетевой чат**

В командах по два-три человека спроектировать и реализовать сетевой чат (наподобие Telegram) с помощью gRPC:

* peer-to-peer, то есть соединение напрямую;
  + и клиент, и сервер должны быть одним и тем же приложением, работающим в разных режимах;
* графический пользовательский интерфейс;
* отображение имени отправителя, даты и времени отправки и текста сообщения;
* при запуске указываются:
  + адрес peer-а и порт, если хотим подключиться;
  + должно быть можно не указывать, тогда работаем в режиме сервера;
  + наше имя пользователя.

Реализация допустима на любом языке программирования из поддержанных gRPC.

Задача выполняется в командах по 2-3 человека в течение двух недель с момента публикации условия. Результат сдаётся через систему контроля версий git и хостинг GitHub либо его аналоги (GitLab, BitBucket) в виде пуллреквеста из отдельной ветки в собственный (изначально пустой) репозиторий. Ссылка на пуллреквест прикладывается как решение в систему поддержки образования. При этом:

* код должен удовлетворять требованиям стиля кодирования выбранного языка и технологии реализации;
* должны быть юнит-тесты и настроенная система непрерывной интеграции, подключённая к репозиторий и запускающая юнит-тесты;
* должны быть комментарии к коду (как минимум, к каждому классу или функции и структуре данных, если выбранный язык программирования не имеет классов);
* репозиторий должен содержать файл с лицензией и файл README.md с описанием процесса сборки и запуска.

***Проверяемые компетенции*:** ПКА-1, ПКА-3, ПКА-4, ПКА-5, ПКП-1, ПКП-2, ПКП-3, ПКП-5, ПКП-6, ПКП-7, ПКП-9.

***Критерии оценивания*:** решение оценивается по шкале от 0 (нет решения) до 10 (продемонстрированы навыки программирования, умение пользоваться системой контроля версий, инструментами CI и модульного тестирования, код грамотно оформлен и не содержит очевидных ошибок и антипаттернов реализации, программа выполняет все функциональные требования, пользовательский интерфейс не имеет очевидных ошибок UX).

**3.1.5 Методические материалы для оценки обучающимися содержания и качества учебного процесса**

Для оценки обучающимися содержания и качества учебного процесса применяется анкетирование в соответствии с методикой и графиком, утвержденными в установленном порядке.

**3.2. Кадровое обеспечение**

**3.2.1 Образование и (или) квалификация штатных преподавателей и иных лиц, допущенных к проведению учебных занятий**

К проведению лекционных занятий должны привлекаться преподаватели, имеющие диплом о высшем образовании по соответствующему направлению.

**3.2.2 Обеспечение учебно-вспомогательным и (или) иным персоналом**

Не требуется.

**3.3. Материально-техническое обеспечение**

**3.3.1 Характеристики аудиторий (помещений, мест) для проведения занятий**

Требуется стандартно оборудованная аудитория с проектором.

**3.3.2 Характеристики аудиторного оборудования, в том числе неспециализированного компьютерного оборудования и программного обеспечения общего пользования**

Доска для письма маркером, мультимедийный проектор.

**3.3.3 Характеристики специализированного оборудования**

Не требуется.

**3.3.4 Характеристики специализированного программного обеспечения**

Не требуется.

**3.3.5 Перечень и объёмы требуемых расходных материалов**

Маркеры для доски, губка

**3.4. Информационное обеспечение**

1. Pooley, R. J. Wilcox, P., Applying UML : Advanced Applications, Oxford : Butterworth-Heinemann. 2004, 202 pp., <https://find.library.spbu.ru/vufind/Record/ocn173660459> (дата обращения: 24.06.2019г)
2. Bjøner, Dines. Software Engineering 3: Domains, Requirements, and Software Design, Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, 768pp., URL: https://find.library.spbu.ru/vufind/Record/978-3- 540-33653- 2 (дата обращения: 15.01.2017г)
3. Bernus, Peter. Mertins, Kai. Schmidt, Günter. Handbook on Architectures of Information Systems, Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, 896pp., URL: https://find.library.spbu.ru/vufind/Record/978-3- 540-26661- 7 (дата обращения: 15.01.2017г)

**Раздел 4. Разработчики программы**

Литвинов Юрий Викторович, кандидат технических наук, доцент кафедры системного программирования, [y.litvinov@spbu.ru](mailto:y.litvinov@spbu.ru)

Брыксин Тимофей Александрович, кандидат технических наук, доцент кафедры системного программирования, [t.bryksin@spbu.ru](mailto:t.bryksin@spbu.ru)