**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 и архитектура программного обеспечения

Software Architecture and Design

**Язык(и) обучения**

русский

Трудоемкость в зачетных единицах: 3

Регистрационный номер рабочей программы: 003669

Санкт-Петербург

2019

**Раздел 1. Характеристики учебных занятий**

**1.1. Цели и задачи учебных занятий**

Цель изучения дисциплины: ознакомление студентов 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 ПО.

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

Программа курса предназначена для студентов 3 курса бакалавриата, освоивших программу курсов «Основы программирования», «Практикум на ЭВМ» и «Разработка программного обеспечения» (или аналогичных им). Максимальная эффективность программы будет обеспечена при условии, что студент 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:

* УКБ-1 – способен осуществлять систематизированные поиск, сбор, структурирование, критический анализ и синтез необходимой информации, применять системный подход для решения поставленных задач;
* УКБ-2 – способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений, в т.ч. финансовых, участвовать в разработке и реализации проектов, в т.ч. предпринимательских;
* 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 и практические занятия в аудитории. Практические занятия проводятся в группах по два-три человека и включают элементы групповых проблемных работ, используют технологии «мозговой штурм» и включают в себя анализ реальных ситуаций профессиональной деятельности.

**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 | 15 |  |  | 15 |  |  |  |  | 1 |  |  |  | 67 |  | 10 |  | 10 | 3 |
|  | 2-100 |  |  | 10-25 |  |  |  |  | 10-25 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| ИТОГО | 30 |  |  | 15 |  |  |  |  | 1 |  |  |  | 67 |  | 10 |  |  | 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 6 |  | зачёт |  |

**2.2. Структура и содержание учебных занятий**

Период обучения (модуль): семестр 6.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Введение в проектирование ПО | лекции | 2 |
| по методическим материалам | 4 |
| II. | Объектно-ориентированное проектирование | лекции | 2 |
| практические занятия | 2 |
| по методическим материалам | 4 |
| III. | Моделирование, язык UML | лекции | 1 |
| практические занятия | 3 |
| по методическим материалам | 10 |
| IV. | Архитектурные стили | лекции | 2 |
| по методическим материалам | 4 |
| V. | Предметно-ориентированное проектирование | лекции | 2 |
| практические занятия | 2 |
| по методическим материалам | 8 |
| VI. | Шаблоны проектирования | лекции | 3 |
| практические занятия | 3 |
| по методическим материалам | 25 |
| VII. | Проектирование распределённых приложений | лекции | 2 |
| практические занятия | 4 |
| по методическим материалам | 8 |
| VIII. | Примеры архитектур | лекции | 1 |
| практические занятия | 1 |
| по методическим материалам | 4 |
| IX. | Промежуточная аттестация | зачёт | 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.
2. Практика по проектированию: проектирование интерпретатора командной строки.

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. Краткий обзор диаграмм UML.
2. Практика по UML: по данному техническому заданию нарисовать диаграмму случаев использования разрабатываемого приложения и диаграмму активностей для бизнес-процесса предприятия, для которого разрабатывается приложение.
3. Практика по UML: по данному техническому заданию построить модель данных разрабатываемого приложения в виде диаграммы классов, диаграмму компонентов и диаграмму развёртывания.

Раздел 4: 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 5: Предметно-ориентированное проектирование.

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».
2. Практика по проектированию: провести анализ и построить модель предметной области для компьютерной игры в жанре Roguelike по данному техническому заданию.

Раздел 6: Шаблоны проектирования.

1. Понятие шаблона проектирования. Структурные шаблоны. Рекурсивная композиция, шаблоны «Компоновщик» и «Декоратор». Выделение алгоритма в объект, шаблон «Стратегия».
2. Практическое занятие: уточнить модель компьютерной игры Roguelike, построенной на предыдущем занятии, используя шаблоны «Стратегия» для поддержки различных поведений мобов и «Декоратор» для поддержки временных эффектов, накладываемых на мобов.
3. Порождающие шаблоны: «Абстрактная фабрика», «Фабричный метод». Клонирование объектов, шаблон «Прототип». Поэтапное конструирование объекта, шаблон «Строитель».
4. Практическое занятие: уточнить модель компьютерной игры Roguelike, построенной на предыдущем занятии, используя шаблоны «Строитель» для инициализации карты, «Абстрактная фабрика» для создания мобов и предметов на карте и «Прототип» для поддержки клонирования персонажей и предметов.
5. Поведенческие шаблоны. Шаблон «Цепочка обязанностей». Отложенное выполнение. Шаблон «Команда». Подходы к реализации поведения, основанного на модели конечного автомата, шаблон «Состояние».
6. Практическое занятие: уточнить модель компьютерной игры Roguelike, построенной на предыдущем занятии, используя шаблон «Команда» для поддержки взаимодействия с пользователем. Оформить итоговую архитектуру системы в виде диаграмм компонентов и классов.

Раздел 7: Проектирование распределённых приложений.

1. Понятие распределённой системы. Виды сущностей в распределённой системе, способы организации взаимодействия, варианты размещения, типичные архитектурные стили. Удалённые вызовы: RPC, RMI, примеры: protobuf, gRPC. Понятие веб-сервиса. SOAP. Пример технологии разработки веб-сервисов: Windows Communication Foundation. Очереди сообщений, пример: RabbitMQ. Архитектурный стиль REST. Микросервисы. Архитектурный стиль Peer-to-Peer, примеры архитектур.
2. Практическое занятие: реализовать консольный сетевой чат, используя технологию gRPC для обеспечения сетевого взаимодействия.
3. Практическое занятие: оформить сетевой чат, разработанный на предыдущем занятии, в виде Docker-контейнера.

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 компьютерной игры Battle for Wesnoth: основные архитектурные соображения, компонентная архитектура, Wesnoth Markup Language, внутренняя модель данных. Архитектура подсистемы многопользовательской игры. Выводы.

**Раздел 3. Обеспечение учебных занятий**

**3.1. Методическое обеспечение**

**3.1.1 Методические указания по освоению дисциплины**

Для успешного усвоения дисциплины необходимо посещение занятий и самостоятельная работа с материалами лекций и рекомендованной литературой. Особое внимание следует уделить части курса, посвящённой шаблонам проектирования – на лекциях будут разобраны только те шаблоны, которые потребуются на практиках, остальные шаблоны, включённые в список вопросов к зачёту, оставлены для самостоятельного изучения.

Практические занятия рекомендуется посещать все, поскольку задания на них связаны и зависят друг от друга. Практические занятия проводятся в группах по 2-3 человека, при этом допустимо, чтобы состав групп менялся от занятия к занятию.

**3.1.2 Методическое обеспечение самостоятельной работы**

Основная и дополнительная литература.

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

Для получения зачёта необходимо сдать теоретическую часть курса в виде устного зачёта, а также сдать все задачи с практических занятий. В случае пропуска практического занятия задачу, дававшуюся на нём, требуется досдать в течение семестра или на зачёте, в случае пропуска двух и более практических занятий без уважительной причины на зачёте даётся дополнительная задача, которую необходимо сдать в дополнение к задачам с практик. На решение задач в течение зачёта отводится два академических часа, разрешается пользоваться литературой.

Теоретическая часть зачёта 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 зачёт. Количество и содержание дополнительных вопросов – на усмотрение преподавателя, принимающего экзамен. Каждый ответ оценивается по шкале от 0 (нет ответа) до 5 (очень хороший ответ), далее оценка усредняется. Если усреднённая оценка выше 3, теоретическая часть зачёта считается сданной.

Если сданы и теоретическая, и практическая части зачёта, ставится оценка «зачёт», в противном случае «незачёт».

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

***Пример списка вопросов для устного зачёта:***

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. Принципы выделения абстракций предметной области.
6. Принципы SOLID.
7. Закон Деметры. Принципы хорошего объектно-ориентированного кода.
8. Моделирование, визуальные модели, виды моделей.
9. Язык UML. Проектирование структуры системы, диаграммы классов, объектов, пакетов, компонентов и размещения UML.
10. Диаграмма случаев использования UML.
11. Диаграмма активностей UML.
12. Моделирование поведения, диаграммы конечных автоматов UML.
13. Диаграммы последовательностей UML, диаграммы коммуникации UML.
14. Диаграммы составных структур, коопераций, временные диаграммы, диаграммы обзора взаимодействия, диаграммы потоков данных.
15. Понятие архитектурного стиля, трёхзвенная архитектура, Model-View-Controller, Sense-Compute-Control.
16. Структурный и объектно-ориентированный стили, слоистые архитектурные стили.
17. Пакетная обработка, каналы и фильтры, Blackboard.
18. Publish-Subscribe, Event-based-стили.
19. Peer-to-peer.
20. Domain-Driven Design, основные структурные элементы модели предметной области.
21. Паттерны «Агрегат», «Фабрика», «Репозиторий».
22. Паттерны «Компоновщик», «Декоратор», «Стратегия».
23. Паттерны «Адаптер», «Прокси», «Фасад».
24. Паттерны «Мост», «Приспособленец», «Спецификация».
25. Паттерны «Фабричный метод», «Шаблонный метод», «Абстрактная фабрика».
26. Паттерны «Одиночка», «Прототип», «Строитель».
27. Паттерны «Посредник», «Команда», «Цепочка ответственности».
28. Паттерны «Наблюдатель», «Состояние», «Посетитель», «Хранитель».
29. Архитектура распределённых систем: понятие распределённой системы, виды сущностей и взаимодействий, типичные архитектурные стили.
30. Удалённые вызовы процедур (RPC). Protobuf, gRPC.
31. Удалённые вызовы методов (RMI).
32. Веб-сервисы, SOAP. WCF.
33. Очереди сообщений, RabbitMQ.
34. REST.
35. Микросервисы, peer-to-peer.
36. Развёртывание и балансировка нагрузки, Docker.
37. Архитектура системы контроля версий Git.
38. Архитектура системы контроля версий Mercurial.
39. Архитектура компьютерной игры Battle for Wesnoth.

***Примеры задач для практик:***

**Практика 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 через Process (или его аналоги);
  + если введено что-то, чего интерпретатор не знает;
* пайплайны (оператор «|»).

Задача выполняется в командах по 2-3 человека в течение пары, результатом является диаграмма классов, описывающая структуру системы, и рассказ о том, как оно должно работать. В конце пары одно из решений защищают у доски.

**Практики 2 и 3: автомобильный завод**

Крупному автомобильному заводу требуется информационная система для учёта дефектов сборки, возникающих при сборке автомобиля на конвейере.

Работа завода организована следующим образом:

* Завод получает набор заказов от дилеров на производство определённого количества автомобилей в определённой комплектации и определённого цвета
* Отдел планирования производства составляет производственный план, определяющий с учётом загруженности конвейера и поставок комплектующих, в каком порядке автомобили будут запущены в производство
* Процесс производства начинается со сборки кузова, после сборки кузова попадают в покраску, где находятся относительно долгое время (порядка нескольких дней), после покраски происходит сборка всех остальных комплектующих автомобиля на конвейерной линии
* После сборки автомобиль попадает в отдел контроля качества, где проходит серию стендовых испытаний и ходовые испытания
* В случае успешного прохождения проверок автомобиль отгружается заказчику

На каждом из этапов производства и тестирования могут быть выявлены дефекты сборки (от погнутой детали кузова или царапин до сбоев в работе программы управления инжектором). В случае обнаружения дефекта автомобиль снимается с конвейера и отправляется в ремонтную зону, где дефект устраняется и автомобиль возвращается на конвейер либо в зону тестирования. Каждый участок конвейера имеет несколько (от одной до четырёх) ремонтных зон, в каждой зоне имеется несколько (от одного до шести) ремонтных мест, где может находиться один автомобиль. Каждая ремонтная зона обслуживается ремонтной бригадой, имеющей бригадира и нескольких ремонтников. Бригады работают в ремонтных зонах посменно.

Целью разработки информационной системы учёта дефектов является повышение качества сборки, минимизация числа дефектов и минимизация времени, проводимого автомобилем в ремонте. Для достижения этой цели разрабатываемая система должна обладать следующей функциональностью:

* Возможностью вносить данные о ремонтных зонах, ремонтных местах, ремонтных бригадах и их бригадирах
* Возможностью зарегистрировать дефект после поступления автомобиля в ремонтную зону и выполнения первоначальной диагностики:
* Место дефекта, отмечаемое на схеме автомобиля
* Возможная причина дефекта
* Номер автомобиля, дата поступления
* Ремонтник, выполнявший диагностику и ремонт
* Возможностью зафиксировать время начала и конца ремонта, рабочего, выполнявшего ремонт и ремонтное место, занимаемое автомобилем в процессе ремонта
* Возможностью для бригадира назначить ремонт конкретному рабочему
* Возможностью для рабочего зарегистрироваться в системе как доступного для получения заданий в начале смены
* Диспетчер должен иметь возможность видеть свободные и занятые места в ремонтных зонах своего участка конвейера, а также наличие свободных рабочих в зоне, и возможность направить автомобиль для ремонта в свободную ремонтную зону
* Отдел контроля качества должен получать в конце каждой смены сводный отчёт по выполненным ремонтам по каждому участку конвейера, включая общее количество ремонтов, статистику по местам дефекта, среднее время ремонта, детальную информацию о каждом ремонте
* Начальник цеха должен получать в конце каждой смены отчёт об общем количестве ремонтов и суммарном времени ремонтов
* Бригадир каждой бригады в конце каждой смены должен получать информацию по каждому ремонтнику бригады о количестве выполненных ремонтов и времени, затраченном на каждый ремонт, а также суммарному времени, которое работник занимался ремонтами.
* Бухгалтерия в конце каждого месяца должна получать отчёт по каждому ремонтнику всех ремонтных бригад о количестве отработанных смен.

Вашей задачей будет в командах по два-три человека проанализировать данное предложение и построить на языке UML следующие модели:

1. диаграмму случаев использования разрабатываемого приложения;
2. диаграмму активностей для всего бизнес-процесса предприятия, для которого разрабатывается приложение;
3. модель данных разрабатываемого приложения в виде диаграммы классов;
4. диаграмму компонентов разрабатываемого приложения;
5. диаграмму развёртывания разрабатываемого приложения.

**Практика 4: 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.

**Практика 5: Roguelike, структурные шаблоны**

Модифицировать модель игры, построенную на предыдущей практике, чтобы:

* использовать в Roguelike паттерн "Стратегия" для поддержки различных поведений мобов;
* используя паттерн "Декоратор", реализовать для игрока возможность конфузить мобов. Моб, находящийся под эффектом конфузии, перемещается, случайно выбирая соседнюю клетку. Эффект должен быть временным.

Результатом работы должна быть диаграмма классов, описывающая применение данных шаблонов в архитектуре игры.

**Практика 6: Roguelike, порождающие шаблоны**

Модифицировать модель игры, построенную на предыдущей практике, чтобы:

* использовать шаблон «Строитель» для инициализации карты;
* использовать шаблон «Абстрактная фабрика» для создания мобов и предметов на карте;
* использовать шаблон «Прототип» для поддержки клонирования персонажей и предметов (например, как результат применения персонажем заклинания либо как свойство некоторых мобов – для игр подобного жанра типично «размножение» слабых противников на карте, например, грибов или мышей, при этом клонированный объект получает характеристики своего «родителя»).

Результатом работы должна быть диаграмма классов, описывающая применение данных шаблонов в архитектуре игры.

**Практика 7: Roguelike, поведенческие шаблоны**

Модифицировать модель игры, построенную на предыдущей практике, чтобы использовать в Roguelike шаблон "Команда" для поддержки взаимодействия с пользователем. Результатом работы должна быть диаграмма классов, описывающая применение этого шаблона в архитектуре игры.

**Практика 8: сетевой чат на gRPC**

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

**Практика 9: Docker**

Оформить сетевой чат, разработанный на предыдущем занятии, в виде Docker-контейнера.

**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**

Учебно-вспомогательный и инженерно-технический персонал должен иметь соответствующее образование и обладать навыками организации работы с пользовательскими программными продуктами в локальной сети компьютерного класса и в Интернете.

**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**

**3.4.1 Список обязательной литературы**

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)

**3.4.2 Список дополнительной литературы**

1. 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)

**3.4.3 Перечень иных информационных источников**

Не требуются.

**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)