**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

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 4 курса бакалавриата, освоивших программу курсов «Программирование» и «Информатика» (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:

* ОПК-1 – способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной деятельности;
* ОПК-3 – способен применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения;
* ОПК-4 – способен участвовать в разработке технической документации программных продуктов и программных комплексов;
* ОПК-5 – способен инсталировать и сопровожать программное обеспеченение для информационных систем и баз данных, в том числе отечественного производства;
* ПКА-1 – способен демонстрировать базовые знания математических и естественных наук, программирования и информационных технологий;
* ПКП-1 – способность проводить под научным руководством исследование на основе существующих методов в конкретной области профессиональной деятельности;
* ПКП-2 – способен решать задачи в области развития науки, техники и технологии с учетом нормативного правового регулирования в сфере интеллектуальной собственности;
* ПКП-4 – способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях;
* ПКП-5 – 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-6 – 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-8 – способен использовать современные методы разработки и реализации конкретных алгоритмов математических моделей на базе языков программирования и пакетов прикладных программ моделирования
* УКБ-3 – 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.

**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 |
| 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.

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».
2. 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.

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.
2. 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 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 лекционных занятий и самостоятельная работа с материалами лекций и рекомендованной литературой.

**3.1.2 Методическое обеспечение самостоятельной работы**

Основная и дополнительная литература.

Также для лучшего усвоения материала возможно выполнение необязательных заданий для самостоятельной работы и представление их для проверки преподавателю. Примеры заданий для курса:

**Roguelike**

Реализовать Roguelike RPG со следующими функциональными требованиями:

* Персонаж игрока, способный перемещаться по карте, управляемый с клавиатуры
  + Непосредственно стрелками (или дополнительной цифровой клавиатурой), не вводом команды
* Инвентарь персонажа, включающий элементы, влияющие на его характеристики, которые можно надеть и снять
* Карта (автоматически сгенерированная или считываемая из файла, на ваше усмотрение)
* Мобы, способные перемещаться по карте
* Боевая система — движущиеся объекты, пытающиеся занять одну клетку карты, атакуют друг друга

Что хочется:

* Модель предметной области в виде диаграммы классов
* Консольная графика, с возможностью далее сделать графический тайловый интерфейс
* Расширяемая и сопровождаемая архитектура
* Юнит-тесты

**Диздок**

Написать архитектурное описание Roguelike

* Общие сведения о системе
* Architectural drivers
* Роли и случаи использования
  + Описание типичного пользователя
* Композиция (диаграмма компонентов)
* Логическая структура (диаграмма классов)
* Взаимодействия и состояния (диаграммы последовательностей и конечных автоматов)

**Instant Messenger**

Разработать сетевой чат с помощью gRPC

* Peer-to-peer, то есть соединение напрямую
* Консольный пользовательский интерфейс
* Отображение имени отправителя, даты и текста сообщения
* Адрес peer-а и порт — параметры
* Указание своего имени — параметром или в конфигурационном файле
* Нужны юнит-тесты

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

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. Каждый ответ оценивается по шкале от 0 (нет ответа) до 5 (очень хороший ответ), результирующая оценка получается следующим образом:

1. Оценки за ответы на два основных вопроса усредняются, результат усреднения делится на два.
2. Оценки за ответы на дополнительные вопросы усредняются, результат усреднения делится на два и складывается с оценкой, полученной в п.1.
3. Если результирующая оценка
   1. в диапазоне от 2.5 до 3.5, за экзамен ставится «удовлетворительно»;
   2. в диапазоне от 3.5 до 4.5, за экзамен ставится «хорошо»;
   3. в диапазоне от 4.5 до 5, за экзамен ставится «отлично».

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

***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. Принципы выделения абстракций предметной области.
6. Принципы SOLID.
7. Закон Деметры. Принципы хорошего объектно-ориентированного кода.
8. Моделирование, визуальные модели, виды моделей, метафора визуализации.
9. Язык UML. Проектирование структуры системы, диаграммы классов, объектов, пакетов, компонентов и размещения UML.
10. Анализ и моделирование требований, случаи использования, диаграмма случаев использования UML.
11. Диаграммы IDEF0, характеристик, Feature tree, моделирование требований в SysML.
12. Диаграмма активностей UML, BPMN.
13. Концептуальное моделирование и моделирование данных: диаграммы «Сущность-связь», ORM.
14. Понятие и примеры CASE-систем.
15. Моделирование поведения, диаграммы конечных автоматов UML.
16. Диаграммы последовательностей UML, диаграммы коммуникации UML.
17. Диаграммы составных структур, коопераций, временные диаграммы, диаграммы обзора взаимодействия, диаграммы потоков данных.
18. Паттерны «Компоновщик», «Декторатор», «Стратегия».
19. Паттерны «Адаптер», «Прокси», «Фасад».
20. Паттерны «Мост», «Приспособленец», «Спецификация».
21. Паттерны «Фабричный метод», «Шаблонный метод», «Абстрактная фабрика».
22. Паттерны «Одиночка», «Прототип», «Строитель».
23. Паттерны «Посредник», «Команда», «Цепочка ответственности».
24. Паттерны «Наблюдатель», «Состояние», «Посетитель», «Хранитель».
25. Антипаттерны «Круговая зависимость», «Последовательная связность», «Вызов предка», «Проблема Йо-Йо», «Активное ожидание», «Сокрытие ошибки», «Магические числа», «Магические строки».
26. Антипаттерны «Божественный объект», «Поток лавы», «Функциональная декомпозиция», «Полтергейст», «Золотой молоток».
27. Design smells.
28. Антипаттерны «Остров автоматизации», «Stovepipe system», «Привязка к поставщику», «Подразумеваемая архитектура», «Проектирование комитетом»
29. Понятие архитектурного стиля, трёхзвенная архитектура, Model-View-Controller, Sense-Compute-Control.
30. Структурный и объектно-ориентированный стили, слоистые архитектурные стили.
31. Пакетная обработка, каналы и фильтры, Blackboard.
32. Publish-Subscribe, Event-based-стили.
33. Peer-to-peer, C2, CORBA.
34. Domain-Driven Design, основные структурные элементы модели предметной области.
35. Паттерны «Агрегат», «Фабрика», «Репозиторий».
36. Говорящие интерфейсы, функции без побочных эффектов, assertions, замкнутые операции.
37. Ограниченный контекст, непрерывная интеграция, карта контекстов.
38. Подходы к интеграции контекстов.
39. Смысловое ядро, приёмы дистилляции, абстрактное ядро.
40. Крупномасштабная структура, метафора системы, разбиение по уровням, типичные уровни в производственных и финансовых системах, уровень знаний, подключаемые компоненты.
41. Архитектура распределённых систем: понятие распределённой системы, виды сущностей и взаимодействий, типичные архитектурные стили.
42. Межпроцессное сетевое взаимодействие, модель OSI, стек протоколов TCP/IP, сокеты, протоколы «запрос-ответ», протокол HTTP.
43. Удалённые вызовы процедур (RPC). Protobuf, gRPC.
44. Удалённые вызовы методов (RMI).
45. Веб-сервисы, SOAP. WCF.
46. Очереди сообщений, RabbitMQ. Enterprise Service Bus.
47. REST.
48. Микросервисы, peer-to-peer.
49. Развёртывание и балансировка нагрузки, Docker.
50. Архитектура системы контроля версий Git.
51. Архитектура системы контроля версий Mercurial.
52. Архитектура компьютерной игры Battle for Wesnoth.

**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**

**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)