**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет «Высшая школа экономики»**

Факультет компьютерных наук

Департамент программной инженерии

**Рабочая программа факультативной дисциплины**

Методы разработки программного обеспечения

для уровня подготовки – бакалавриат/магистратура/аспирантура

Разработчики программы

Кошелев В.К. - разработчик компании JetBrains, канд. физ.-мат. наук - [vladimir.koshelev@jetbrains.com](mailto:vladimir.koshelev@jetbrains.com)

Мицюк А.А. - ст. Преподаватель, канд. комп. наук - [amitsyuk@hse.ru](mailto:amitsyuk@hse.ru)

Одобрена на заседании департамента программной инженерии «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Руководитель департамента

Авдошин С.М. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утверждена Академическим советом образовательной программы «Программная инженерия» «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г., № протокола\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Академический руководитель образовательной программы

Шилов В.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утверждена Академическим советом образовательной программы «Прикладная математика и информатика» «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г., № протокола\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Академический руководитель образовательной программы

Конушин А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 20\_\_

*Настоящая программа не может быть использована другими подразделениями университета и другими вузами без разрешения подразделения-разработчика программы.*

# Область применения

Настоящая программа учебной дисциплины устанавливает требования к образовательным результатам и результатам обучения студента и определяет содержание и виды учебных занятий и отчетности. Программа предназначена для преподавателей, ведущих дисциплину «Методы разработки программного обеспечения», студентов и слушателей, желающих принять участие в работе факультатива.

# Цели освоения дисциплины

В результате освоения учебной дисциплины студенты и слушатели должны овладеть

следующими образовательными результатами:

* Понимать устройство систем контроля версий и уметь их использовать,
* Владеть технологиями непрерывной интеграции,
* Понимать подходы к тестированию программных продуктов,
* Уметь разрабатывать программное обеспечение в команде,
* Владеть методами планирования с использованием систем трекинга,
* Понимать особенности использования открытого исходного кода.

# Приблизительный тематический план учебной дисциплины

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название раздела** | **Всего часов** | **Аудиторные часы** | | | **Самостоятельная работа** |
| Лекции | Семинары | Практические занятия |
| 1 | Системы контроля версий (СКВ) и работа с ними | 8 | 2 |  | 2 | 4 |
| 2 | Непрерывная интеграция | 8 | 2 |  | 2 | 4 |
| 3 | Тестирование в масштабных программных проектах | 8 | 2 |  | 2 | 4 |
| 4 | Практики командной работы с программным кодом | 8 | 2 |  | 2 | 4 |
| 5 | Планирование и работа с системами трекинга | 8 | 2 |  | 2 | 4 |
| 6 | Особенности разработки и использования открытого программного обеспечения | 8 | 2 |  | 2 | 4 |
| 7 | Исследования и разработка | 8 | 2 |  | 2 | 4 |
| 8 | Социальные аспекты жизни программиста | 8 | 2 |  | 2 | 4 |
|  | **Итого:** | 64 | 16 |  | 16 | 32 |

Данный тематический план содержит приблизительный список тем и распределение часов.

# Формы контроля знаний студентов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тип контроля** | **Форма контроля** | **Модуль** | | **Параметры** |
| **1-й** | **2-й** |
| Текущий | Домашние задания | \* | \* | Индивидуальные и групповые задачи на программирование |
| Итоговый | Экзамен |  | \* | Устный экзамен с билетами; на подготовку даётся 80 минут или больше |

# Содержание дисциплины

Раздел 1. Системы контроля версий (СКВ) и работа с ними

* Какие бывают СКВ?
* История средств и методов контроля версий ПО.
* Какие бывают репозитории?
* Что такое моно-репозитории и в чём их отличие от других?
* Почему современные компании массово переходят на моно-репозитории?

Материалы по теме:

* Scott Chacon, Ben Straub. Pro Git (2nd ed. Edition). Apress, 2014. - Имеется в открытом доступе: <https://git-scm.com/book/en/v2>

Раздел 2. Непрерывная интеграция

* Что такое непрерывная интеграция (CI)?
* Зачем применяется CI и в чём существо используемых методов?
* Какие современные методы развёртывания ПО применяются в промышленном программировании?
* В чём отличие библиотеки и приложения, когда речь идёт о развертывании?
* Что такое контейнеры и как они применяются?

Материалы по теме:

* <https://medium.com/@gwright_60924/software-delivery-101-environments-and-applications-8fff2c4e2524>
* <https://medium.com/@gwright_60924/an-introduction-to-devops-17ec373238f6>
* <https://medium.com/@gwright_60924/continuous-integration-ci-e81032bb8502>
* <https://medium.com/@gwright_60924/continuous-delivery-cd-2200a5ab945b>
* <https://medium.com/@gwright_60924/containers-enhancing-cicd-and-operations-321bb2049148>
* <https://medium.com/@gwright_60924/the-cloud-anything-as-a-service-xaas-and-infrastructure-as-code-iac-9870edc18e90>

Раздел 3. Тестирование в масштабных программных проектах

* Зачем нужны модульные тесты?
* В чём состоят особенности больших проектов, когда речь идёт о тестировании?
* Что такое интеграционное и регрессионное тестирование, в чём отличия?
* Какие практики тестирования применяются?
* Что такое разработка через тестирование (TDD)?

Материалы по теме:

* <https://medium.com/@TechMagic/get-started-with-behavior-driven-development-ecdca40e827b>
* <https://medium.com/javascript-scene/tdd-changed-my-life-5af0ce099f80>
* <https://www.youtube.com/watch?v=qkblc5WRn-U>

Раздел 4. Практики командной работы с программным кодом

* Что такое стиль кода?
* Зачем нужны соглашения о стиле кода? Нужны ли они в действительности?
* Что такое linter и как с ним работать?
* Что такое рефакторинг? Как проводить рефакторинг? Когда вам нужен рефакторинг, а когда не нужен?
* Как правильно распределить изменения по коммитам?
* Что такое рецензирование кода (code review)?
* Как готовить код к рецензированию?
* Как проводить рецензирование чужого кода?
* Как организовать рецензирование кода в команде?

Материалы по теме:

* <http://google.github.io/styleguide/>
* <https://medium.com/acellere/benefits-of-static-code-analysis-a453b5d4a5e9>
* <https://medium.com/palantir/code-review-best-practices-19e02780015f>

Раздел 5. Планирование и работа с системами трекинга

* Что такое баг-трекер (bug/issue tracker)?
* Какие используются эффективные практики документирования ошибок и требований?
* Что такое релизные циклы? Как их планировать?
* Как должна выглядеть хорошая запись об ошибке для себя/другого программиста?
* Что особенно важно не забыть при документировании проблем и ошибок в ПО?

Материалы по теме:

* <https://www.scrum.org/resources/what-is-scrum>
* Scott Chacon, Ben Straub. Pro Git (2nd ed. Edition). Apress, 2014. - Имеется в открытом доступе: <https://git-scm.com/book/en/v2>

Раздел 6. Особенности разработки и использования открытого программного обеспечения

* Что такое открытое программное обеспечение?
* В чём особенности проектов с открытым исходным кодом?
* Всегда ли ПО с открытым кодом разрабатывается сообществом?
* Как эффективно использовать ПО с открытым исходным кодом и не наступать на сопутствующие грабли?
* Какие существуют практики участия в коллективно разрабатываемых проектах?
* Каковы особенности использования ПО с открытым исходным кодом в коммерческих разработках?

Материалы по теме:

* <https://www.fsf.org/ru>
* <https://www.gnu.org/licenses/gpl-faq.en.html>

`

Раздел 7. Исследования и разработка

* Чем исследовательские статьи могут быть полезны прикладному программисту?
* Как правильно инженеру читать статьи? И надо ли вообще их читать?
* Как и где искать новые способы решения задач, которые надо решать?
* Чем отличается характер работы в исследовательских (R&D) компаниях или отделах компаний от работы обычного прикладного/системного программиста?
* Стоит ли описывать в исследовательских статьях свои собственные результаты? Что это даёт инженеру, который не занимается научной работой?

Материалы по теме:

* <https://medium.com/@mattetti/research-and-development-in-a-startup-dc46c076bc8c>
* <https://medium.com/west-stringfellow/optimizing-research-and-development-593a323074ce>

Раздел 8. Социальные аспекты жизни программиста

* Что такое интервью? Когда и какие они бывают у программистов?
* Как правильно готовиться к и проходить интервью?
* В чём состоят особенности работы программиста в распределённых командах из многих участников?
* Почему работа современного программиста не может рассматриваться в отрыве от социальных аспектов?

Материалы по теме:

* <https://medium.com/@TheDailyKobo/how-to-prepare-for-a-job-interview-6bb83b0e0702>
* <https://medium.com/the-mission/feeling-burned-out-so-did-i-here-are-5-ways-to-reinvigorate-yourself-892ed9c5d1dc>

# Критерии оценки знаний, навыков

Текущий контроль предусматривает выполнение студентам набора домашних заданий. Накопленная оценка (О\_накопленная) за курс выставляется по сумме баллов, полученных за выполнение этих домашних заданий. Оценка за домашнее задание выставляется с учетом полноты выполнения задания, а также способности студента объяснить примененные при решении приемы.

На последнем занятии студенты сдают устный экзамен (О\_экзамен) по всему материалу курса. Студент получает не менее 80 минут на подготовку к ответу.

# Образовательные технологии

В рамках лекционных занятий студенты знакомятся с методами, инструментами и приёмами современной разработки программного обеспечения в условиях масштабных проектов. Практические занятия, а также самостоятельная домашняя работа служит приобретению практических навыков по использованию методов и инструментов, которые были рассмотрены на лекциях.

# Оценочные средства для текущего контроля и аттестации студента

Задания курса будут размещяться в вики-системе ФКН <http://wiki.cs.hse.ru/> на странице, соответствующей данному курсу.

В части 5 данной программы приводятся все темы (разделы) курса. Каждая тема содержит в своём описании список вопросов, которые будут рассмотрены в ходе курса, и одновременно являются примерным перечнем вопросов для подготовки и самопроверки студентов к экзамену.

# Порядок формирования оценок по дисциплине

О\_накопленная формируется на основании суммы баллов, полученных в ходе курса, по следующей схеме:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Балл** | 0 | 1-19 | 20-29 | 30-39 | 40-49 | 50-59 | 60-69 | 70-79 | 80-89 | 90-99 | ≥100 |
| **О\_накопленная** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Результирующая оценка вычисляется по формуле:

О\_результирующая = min(0,8\*О\_экзамен + 0,4\*О\_накопленная; 10)

# Учебно-методическое и информационное обеспечение дисциплины

Основная литература отсутствует.

Дополнительная литература перечислена в разделе 5 данной программы для каждого раздела курса отдельно.

В качестве программных средств будут использоваться редакторы исходного кода (Vim и другие) или интегрированные среды разработки (IntelliJ IDEA, Visual Studio и другие).

# Материально-техническое обеспечение дисциплины

Занятия проходят в аудитории, оборудованной проектором и «белой» доской.

Для выполнения заданий используются компьютеры, подключенные к сети Интернет.