Лекция 1

Вводная часть, структура курса:

- системы контроля версий;
- тестирование ПО;
- командная работа.

Курсовая работа оценивается с точки зрения направлений курса ТРПО: командная работа (стиль кода, распределение задач в команде), история проекта в системе контроля версий (описание коммитов, использование веток для разработки нового функционала), тестирования (полнота покрытия тестами, использование систем автоматизации сборки и технологии непрерывной интеграции).

Системы контроля версий: краткая история развития: RCS, CVS, SVN, git…

Система контроля версий git (https://git-scm.com/book/ru/v2):

Git на локальной машине: создание репозитория (git init), базовые команды (help, add, commit), состояния файлов (status)

Вопросы к экзамену / для самоподготовки:

1. Системы контроля версий: история развития (RCS, CVS, SVN и git), локальные, централизованные и распределенные системы. В чем их отличие.
2. Git: создание репозитория + фиксация изменений, стадии процесса

Лекция 2

Git на локальной машине: подробное рассмотрение команд git status, содержание .gitignore.

Команды git diff, git commit, команды удаления и перемещения файлов git rm, git mv

git log – команда для отображения истории коммитов.

Отмена изменений: git reset, git commit –amend, git checkout

Вопросы к экзамену / для самоподготовки:

3. .gitignore – назначение, примеры команд.
4. Git log – просмотр нужной части истории проекта (выбор время коммита, ключевых слов)

Лекция 3

Git refresh – краткий повтор пройденного.

Основы систем автоматизации сборки – make + Makefile

Git на локальной машине: ветвление

Обзорная часть: зачем нужны параллельные ветки при разработке ПО, немного философии (https://nvie.com/posts/a-successful-git-branching-model/)

Ветки в git, git branch, работа с ветками (checkout, commit, log).

Вопросы к экзамену / для самоподготовки:

5. Git: модели ветвления
6. Git: создание ветки (branch), переключение между ветками

Лекция 4

Git на локальной машине: слияние веток - git merge (fast-forward, recursive merge)

Конфликты при слиянии: природа возникновения, способы решения.

Еще немного философии про управление ветками в процессе разработки

Вопросы к экзамену / для самоподготовки:

7. Git: объединение веток (merge)
8. Git: конфликты и способы их разрешения

Лекция 5

Git refresh – краткий обзор пройденного.

Git на локальной машине: git rebase

Распределенный git: работа с удаленными репозиториями

git clone, git remote, короткие имена для удаленных репозиториев

краткий обзор git fetch, git pull, git push

Удаленные ветки, отслеживание веток

Вопросы к экзамену / для самоподготовки:

9. Git: работа с удаленным репозиторием (git fetch & git pull)
10. Git: работа с удаленным репозиторием (git rebase)
11. Git: работа с удаленным репозиторием (git push)

Лекция 6

Распределенный git: git pull –rebase – создание (псевдо)линейной истории проекта.

Завершение первой части курса про системы контроля версий.

Вторая часть курса: тестирование. Unit тесты.

Многофайловые приложения – жизненная необходимость (тестирование, совместная работа над проектом).

Интерфейсы функций (https://ru.wikipedia.org/wiki/Прототип_функции).

Include guards (https://ru.wikipedia.org/wiki/Include_guard).

Системы автоматизации сборки (https://ru.wikipedia.org/wiki/Автоматизация_сборки), make (https://ru.wikipedia.org/wiki/Make), структура Makefile.

Официальная документация: GNU Make.

Несколько слов про ТЗ к курсовой работе

Вопросы к экзамену / для самоподготовки:

12. Системы автоматизации сборки: цели, задачи, примеры
13. Make – примеры

Лекция 7

Тестирование приложений. Основы. ***Lee Copeland, «A Practitioner's Guide to Software Test Design»***

Модели зрелости тестирования ПО (https://software-testing.ru/library/5-testing/53-2008-10-13-19-38-23)

Типы тестирования (black box and white box):
https://ru.wikipedia.org/wiki/Тестирование_по_стратегии_чёрного_ящика

https://ru.wikipedia.org/wiki/Тестирование_белого_ящика

Невозможно протестировать все => тестирование классов эквивалентности

https://training.qatestlab.com/blog/technical-articles/equivalence-classes-and-boundary-values/

Библиотека CTest (https://github.com/bvdberg/ctest) – краткий обзор

Покрытие кода тестами (https://software.intel.com/sites/default/files/article/401105/code-coverage.pdf)

Вопросы к экзамену / для самоподготовки:

14. Тестирование ПО: разработка теста для функции по ее спецификации
15. Тестирование ПО: статическое, динамическое тестирование
16. Тестирование ПО: методы белого и черного ящика
17. Тестирование ПО: проверка качества покрытия тестами

Лекция 8

Непрерывная интеграция (continuous integration, CI) -
https://ru.wikipedia.org/wiki/Непрерывная_интеграция

TravisCI (https://travis-ci.org/ - https://habr.com/ru/post/140344/

Завершение второй части курса – тестирование ПО.

Базовые стратегии разработки ПО: (http://www.interface.ru/iarticle/files/35798_52840953.pdf)

Каскадная, инкрементная, эволюционная

Вопросы к экзамену / для самоподготовки:

18. Непрерывная интеграция: цели, основные принципы, инструментарий
19. Каскадная стратегия разработки: достоинства и недостатки
20. Инкрементальная стратегия разработки: достоинства и недостатки
21. Эволюционная стратегия разработки: достоинства и недостатки

22. Базовые стратегии разработки ПО, выбор модели

Лекция 9

Гибкие методологии разработки ПО - Agile
(https://ru.wikipedia.org/wiki/Гибкая_методология_разработки ,
https://ru.wikipedia.org/wiki/Agile_Manifesto,

https://agilemanifesto.org/iso/ru/manifesto.html)

Экстремальное программирование (Extreme programming – XP)

(https://ru.wikipedia.org/wiki/Экстремальное_программирование,
https://habr.com/ru/post/197760/)

Вопросы к экзамену / для самоподготовки:

23. Гибкие технологии разработки ПО: Agile манифест
24. Экстремальное программирование (XP): тестирование и рефакторинг
25. Экстремальное программирование (XP): code style и метафора системы
26. Экстремальное программирование (XP): парное программирование и continuous integration
27. Экстремальное программирование (XP): соглашение о кодировании и коллективное владение кодом

Лекция 10-14

Scrum: одна из Agile технологий.

Роли в scrum: команда, scrum-мастер, владелец продукта (product owner)

Артефакты scrum: Product Backlog, Sprint Backlog, Burn-Down Chart, Increment, Sprint Goal, User story and Story points

Ритуалы scrum: каждый ритуал — это встреча лицом к лицу в реальном времени, что позволяет отвлечься от непосредственной работы и дает возможность целевого общения с коллегами о смысле этой работы. Скрам ставит коммуникацию выше документации, именно поэтому он обеспечивает регулярность встреч с четко определенными возможностями различных типов полезного общения лицом к лицу.

Grooming, planning, Daily scrum, sprint review (demo session), retrospective

https://ru.wikipedia.org/wiki/SCRUM

https://www.rulit.me/books/scrum-revolyucionnyj-metod-upravleniya-proektami-read-412856-1.html

Вопросы к экзамену / для самоподготовки:

28. Scrum: основные характеристики
29. Scrum: структура + роли
30. Scrum: структура + ритуалы
31. Scrum: структура + артефакты
32. Scrum: масштабируемость (scaling)