Технологии разработки качественного программного обеспечения

Осень

Практические занятия осеннего семестра заключаются в командной разработке программного продукта на самостоятельно выбранную тему. Работы выполняются в следующем порядке:

1. Разделиться на команды по 3–4 человека, выбрать тему и сделать краткое описание разрабатываемого продукта (2–3 предложения). Тема может быть любой на ваш выбор. Если не уверены, что тема подходит по сложности (слишком простая или, наоборот, слишком сложная) – дополним или уменьшим функциональность на следующем этапе.
   - Пример_1: бот для телеграма, односложно отвечающий на команды - слишком просто; бот, по запросу агрегирующий информацию с различных сайтов и обрабатывающий её - в самый раз.
   - Пример_2: соцсеть с кучей настроек, лентой новостей, друзьями и т.д. - слишком сложно, даже для 4 человек; сайт с модерируемыми статьями пользователей, возможность подписаться на интересующего автора, комментировать - вполне достаточно.
   - Пример_3: игры под андроид/iOs. Тетрис, змейка, крестики-нолики - очень слабо даже для 1 человека; аркадные стрелялки/головоломки /etc – ok

   Длительность этапа: 1 неделя с первого занятия. Староста должен прислать список с распределением в тимс/на почту. Все студенты, не определившиеся с командой, будут распределены случайным образом.

2. Первый вариант функциональной спецификации (FS). Требуется сформулировать функциональные и нефункциональные требования на разрабатываемый программный продукт. Описание выполняется в виде таблицы, например:

Идентификатор	Требование
F_Login_1	Кнопка «Войти» неактивна, пока хотя бы одно из полей «Логин» или «Пароль» пустое
F_Login_2	В случае возникновения ошибки при попытке авторизации, на экране отображается уведомление с текстом ошибки

F_Pref_1	По нажатию на кнопку «Настройки» открывается экран «Настройки»
…	…

Длительность этапа: 1 неделя. Команда/часть команды приносят спецификацию на занятие, где она обсуждается и вносятся необходимые правки.

3. Финальный вариант FS. С этого момента крупные правки в FS вносятся только в случае форс-мажоров, мелкие должны быть минимизированы и обоснованы.

   Длительность этапа: 2 недели.

4. Первый вариант документа High-Level Design (HLD). В документе должны быть описаны макет дизайна интерфейса (при наличии), общая архитектура приложения, стэк технологий, спроектированы диаграммы классов/модулей/etc, схемы баз данных, API. Все схемы должны сопровождаться описанием.

   Длительность этапа: 2 недели.

5. Финальный вариант HLD. Изменения в HLD вносить можно, при наличии соответствующего обоснования.

   Длительность этапа: 2 недели

6. Разработка программного продукта.
   - Обязательно наличие репозитория проекта на github/gitlab/bitbucket/etc. Репозиторий ведётся согласно github flow/gitlab flow/git flow по выбору команды.
   - Система контроля за разработкой. Любая по выбору команды, самые популярные за предыдущие годы: Github Projects, Trello, Notion. Во всех из них, предоставляется возможность создать kanban-доску, на которой находятся описания задач, необходимых для реализации проекта, с распределением по членам команды. Доски, как и содержание карточек на них, будут проверяться. По желанию, при использовании trello, можно связать его напрямую с репозиторием (есть расширения для всех основных хранилищ) и в карточках ссылаться напрямую на issue, в которых и приводить подробные описания для задач.

     Пример: карточка с содержанием "сделать главную страницу сайта" - неправильно; карточка "сделать главную страницу сайта согласно

пунктов 2.2-2.15 FS и раздела 3 HLD" - лучше, но задача слишком крупная для одной карточки, правильнее будет разбить её на карточки меньшего объёма работы "сделать верхнее меню согласно п. 2.2-2.3" + "сделать ленту публикаций согласно п. 2.4 - 2.7" + и т.д.

Длительность этапа: до зачётной недели включительно. На финальной сдаче проекта (зачётная неделя) должны присутствовать все участники команды, в заочном режиме зачёты проставляться не будут.

Поскольку предмет рассчитан на 2 семестра и во втором полугодии вместо практик будет курсовик с оценкой, то на оценку «удовлетворительно»:

- В качестве основы для написания документации берётся любой проект за прошлые 3 учебных года (курсовая/большая лабораторная/проект с ОПД), вносятся исправления в соответствии согласованными FS и HLD и получаете в этом семестре «зачёт». Дедлайны выполнения сохраняются.
- В весеннем семестре пишете тесты по минимальным границам, обозначенным в задании, и получаете оценку «удовлетворительно». Дедлайны на выполнение также сохраняются.

Все команды, которые решат идти по этому пути, должны быть указаны в списках, присылаемых старостами. Выбрать данный вариант по ходу семестра (после истечения сроков формирования команд) будет нельзя.

Весна

Общая информация

В рамках курсовой работы нужно реализовать три вида тестов – модульные, интеграционные и системные. Время на выполнение каждого вида тестирования – 2 занятия (месяц) считая с первого, т. к. задание выдаётся осенью. В случае, если разработанный в осеннем семестре программный продукт не подпадает под минимальные требования по количеству тестов или данный вид тестов неприменим/нецелесообразен, то предлагается работать над любым программным продуктом с открытым исходным кодом. Примеры: kafka, zookeeper, grafana, elasticsearch, consul, jenkins, atom, Eclipse Che, notepad++ или любой другой популярный продукт на выбранном вами языке. При выборе open-source продуктов для тестирования, метрика покрытия не применяется, т. к. многие из них слишком велики, критерием принятия тестов служит приём пул-реквеста в официальный репозиторий.

Модульное тестирование

Необходимо выполнить модульное тестирование разработанного программного продукта. Кодовая база всего продукта должна быть покрыта тестами на 80% и более, но не менее 25 тестов на каждого члена команды.

Обязательные требования:

1. Тестирование должно производиться автоматически при сборке проекта тем сборщиком, который используется для формирования исполняемого файла (cmake, gradle, maven, ant, etc.) Фреймворки для написания юнит-тестов: junit, testNG, XUnit.Net, Nunit – любые, подходящие для языка вашего проекта и формирующие необходимую отчётность.
2. Применение нескольких техник тест-дизайна: классы эквивалентности, граничные условия, попарное тестирование, etc.

Отчёт по модульному тестированию должен содержать:

1. Описание выполненной работы, использованных инструментах, применённых техниках тест-дизайна.
2. Отчёт о прохождении тестов с результатами и оценкой покрытия кода тестами.
3. Описание процедуры расширения тестового набора на примере добавления нового блока кода, алгоритма, метода.

Интеграционное тестирование

Интеграционное тестирование предполагает наличие нескольких модулей или, если приложение построено в соответствии с микросервисной архитектурой, микросервисов разработанного приложения. В качестве модулей могут быть функциональные части (например: модуль авторизации\аутентификации, модуль взаимодействия с пользователем, модуль интеграции со сторонним сервисом).

Необходимо выполнить интеграционное тестирование нескольких модулей в соответствии с предварительно описанными сценариями тестирования. Каждый сценарий тестирования должен представлять собой некоторый законченный вариант использования системы (кейс) пользователя.

Интеграционное тестирование предполагает командную работу над программным продуктом, поэтому необходимо проводить его с использованием одного из инструментов CI, доступных на рынке: Gitlab, Jenkins, Bamboo, TeamCity, etc.

Минимальное количество сценариев - 10.

Обязательные требования:

1. Предварительное формирование документа, описывающего тестовые сценарии (утверждение сценариев у преподавателя).
2. Применение заглушек (mock-сервисов или mock-объектов) для изоляции от окружения и внешних сервисов или для ускорения прохождения тестов.
3. Рассмотрение негативных сценариев тестирования.
4. Поднятие сервера непрерывной интеграции и запуск задачи по интеграционному тестированию по временному триггеру или по событию изменения кода приложения\интеграционных тестов.

Отчёт по интеграционному тестированию должен содержать:

1. Отчёт о выполненной работе, использованных инструментах, применённых техниках тест-дизайна.
2. Тест-план со словесным описанием тестовых сценариев, которые планируется реализовать в интеграционном тестировании (необходимо предварительно утвердить у преподавателя)
3. Отчёт о прохождении тестов с результатами на сервере непрерывной интеграции.
4. Описание процедуры расширения тестового набора на примере добавления новой функциональной части (или модуля)

Системное/End-to-End тестирование

На данном уровне необходимо протестировать готовый продукт по бизнес-требованиям, сформированным перед началом непосредственного проектирования программного продукта. В качестве тестовых сценариев выбираются основные сценарии использования ПО в полностью рабочем окружении. Целесообразно использование инструментов веб-тестирования или UI-тестирования, таких как selenium, puppeteer и т. д.

Обязательные требования:

1. Предварительное формирование документа, описывающего тестовые сценарии (утверждение сценариев у преподавателя)
2. Поднятие сервера непрерывной интеграции и запуск задачи системного тестирования по временному триггеру или в ручном режиме (кроме тестирования производительности)

Отчёт по системному тестированию должен содержать:

1. Отчёт о выполненной работе, использованных инструментах.
2. Тест-план со словесным описанием тестовых сценариев
3. Отчёт о прохождении тестов с результатами на сервере непрерывной интеграции.

Общие критерии оценки

65–71 баллов – удовлетворительно

72–84 баллов – хорошо

85+ баллов – отлично

За сорванные сроки итоговый балл уменьшается на 15 баллов/месяц, за каждый вид тестов. За не сделанный вид тестов (учитываются только обязательные) количество баллов уменьшается на 30.

Модульное тестирование

Минимальное количество тестов – 20 на каждого члена команды.

	% покрытия кода тестами	количество баллов
1	60 - 70	10
2	71 - 80	15
3	81+	25

Интеграционное тестирование

Минимальное количество сценариев - 10

Каждый сценарий оценивается в 2 балла, максимум баллов – 40

Системное/End-to-End тестирование

Минимальное количество сценариев – 10.

Каждый сценарий оценивается в 2 балла, максимум баллов - 40

В рамках системного тестирования, в случае применимости, возможно произвести тестирование производительности:

1. Нагрузочное тестирование [load testing] (+5 баллов)
2. Тестирование стабильности [stability testing] (+5 баллов)
3. Тестирование восстановления [recovery testing] (+5 баллов)
4. Объёмное тестирование [volume testing] (+5 баллов)