Роли участников и итерации (что нужно делать)

Идея курсов

Цель обоих курсов - погрузить участников в максимально приближенный (где-то даже через чур) к реальности процесс проектной разработки ПО с реальными коллегами / задачами / проектами / заказчиками и таким образом приобрести целостное и полное представаление о всем цикле разработки ПО.

В больших корпорациях за счет накопленных ресурсов многие сложности и острые углы проектов / особенности жизненного цикла ПО могут годами оставатся для вас тайной. Поэтому вам важно оказатся в среде, где все мрачно, чтобы испытать себя и наглядно ощутить как (по вам) проходит процесс разработки ПО.

Задачи

- Познакомиться с разными ролями в рамках разработки ПО
- Ощутить на себе этапы жизненного цикла разработки ПО
- Столкнутся с коммуникативными и организационными сложностями
- Освоить базовые приемы / понятия проектной работы

Роли участников

Прежде чем излагать разделение студентов на роли, важно отметить, что главная цель всех студентов данного курса - **успешная разработка выбранного проекта**:

- Заказчик доволен (== он получил то, что хотел согласно заданию),
- Стабильная работа (== отстутсвие сбоев),
- Отчуждаемость и осязаемость (== посторонний человек сможет самостоятельно по материалам работы развернуть и использовать ваш продукт).

Поэтому

- Помните, что мяч во взаимодействии с заказчиком перманентно на вашей стороне. Вы бегаете / тормошите / пишете заказчику, наоборот не сработает.
- Стоит накладывать описания ролей ниже на данную цель и рассматривать ваши обязанности через эту оптику,

Бакалавры (3 курс)

Выполняют роль разработчиков / тестировщиков / инженеров проекта. Их основные задачи -

- подготовка кода, тестов, документации и других содержательных материалов,
- презентация материалов,

- предоставление обратной связи по коллегам преподавателям,
- эскалирование проблем магистру или преподавателям.

Магистранты (5 курс)

Выполняют роль тимлида + проджект-менеджера. Поскольку разные люди вкладывают в эти понятия разные вещи, то определим свои требования к данной роли. Магистрант:

- **лично отвечает за свою команду и успех своего проекта,**
- организует общение с заказчиком (установочный созвон + согласование плана на итерацию + периодическое предъявление результатов заказчику, получение обратной связи),
- формализует обратную связь / пожелания / требования заказчика в задачи для команды,
- организует общение с командой (регулярные созвоны для обсуждения прогресса),
- планирует работу (и по сути, и по времени), создает задачи и следит за порядком в репо,
- проверяет результаты бакалавров и контролирует достижение результата,
- выполнять часть обязанностей разработчиков (см. Бакалавры выше),
- презентация результатов,
- предоставление обратной связи о коллегах для преподавателей,
- эскалирует проблемы преподавателям / заказчикам.

Что означает личная ответственость за команду и проект? Это означает что общий успех проекта и действия бакалавров напрямую влияют на баллы магистра за дисциплину (подробнее в документе “Формирование оценки”):

- **Команда**: Магистр должен не просто отправлять задания / доносить информацию до бакалавров, но также и следить что все всё поняли / сделали надлежащим образом. Если по итогам итерации, кто то из бакалавров не выполнил обязательные условия / простаивал и магистр не принял мер, то это его управленческое упущение.
- **Проект**: успешность руководителя = успешность проекта. Поэтому, если проект идет с негативной динамикой (при отсутствии попыток магистра как-то выправить курс), то это отражается на баллах магистра.

Эскалация (эскалирование) проблем

В проектной работе возможны и неизбежны ситуации, когда участник сталкивается с затруднением / проблемой, которую он не может решить в силу органичений собственной должности / недостатка знаний или навыков или других причин. Конструктивное решение в данном случае это эскалация проблемы - передача информации о проблеме на уровень выше (вашему непосредственному начальнику / начальнику выше и тд).

В случае нашего курса цепочки эскалации такие

- Магистры - Заказчики - Преподаватели (если есть проблемы с пониманием постановки задачи, сугубо техническими сложностями)
- Магистры - Преподаватели (Если проблемы с курсом, с магистром, с бакалаврами, с заказчиками)

!! Для эскалации преподвателям используйте раздел **эскалация** на дискорд сервере. !!

Предостережение - эскалация проблемы это **крайняя** мера. В проектной работе ценятся люди, которые обращаются к вышестоящим в редких случаях (сами не справились и есть острая необходимость в помощи). Поэтому, перед эскалацией проблемы проверьте себя:

- Я сформулировать проблему в письменной форме, сформулировал как ее воспроизвести (так что это поймет и другой человек) и обозначил, в чем основная загвоздка
- Я попытался самостоятельно решить проблему несколькими способами
- Я учел рекомендации к тому, как строить коммуникацию (см “Советы по коммуникации”)

Эскалируя проблему, обозначайте в чем ее важность (Чем она мешает и как негативно влияет на процесс) и срочность (как быстро ее нужно решать).

Заказчики

Задачи заказчика

- Сформулировать интересующий проект
- Подготовить набор ссылок по технологиям проекта
- Отвечать на вопросы студентов по проекту
- Оценивание промежуточного и финального результата

Преподаватели курса

Преподаватели представляют собой финальную инстанцию в иерархии ролей. К ним можно обратится по любому вопросу, но не каждый из вопросов целесообразно задавать сразу им (см. Эскалация выше).

Проекты и команды

В курсе будет дан набор проектов для выполнения. Студентам будет необходимо в ограниченный срок выбрать проект (путем заполнения формы). Части студентов будет предложено пройти курс в рамках проекта Fast track.

Будьте внимательны - поменять выбор нельзя :(

По итогам выбора будут организованы проектные команды: 4-5 бакалавров + 1-2 магистра. Команды получают доступ в проектный репо и канал проекта на дискорд-сервере. Репо или созданный преподавателем, или предоставленный заказчиком.

Как работать в репо:

1. Главная ветка - main / master (если заказчик явно не попросил обратного)
2. Нельзя делать прямые коммиты в главную ветку
3. Одна задача == одна ветка (название должно отражать номер и название задачи) == один PR
4. PR мержат и ревьювят магистры. На смерженных PR должны быть апрувы магистров (могут быть и апрувы бакалавров - это дополнительный плюс, но магистры обязательны)

Организация работы с задачами и фичами:

1. Фиксируйте все задачи и фичи как issue в репо
2. Создавайте метки для категорий и milestone для обозначения итераций
3. Организуйте работу с задачами в виде проекта github (тип Board) - он должен отображатся на странице projects вашего репо

Fast track проект

Данный проект призван учить проектной работе в немного иной парадигме - погружаясь в проработанну. область автоматизации образования вы одновременно обучаетесь ролям QA и быстрее примеряете на себя перспективу пользователя. Вам предстоит разобраться со сложившейся структурой и адаптироватся к специфичным требованиям пользователей (студентов и преподавателей). Поэтому роли и итерации здесь трактуются слегка иначе.

Цель проекта - подготовка, интеграция и проверка качества средств автоматизации.

Проект подразумевает разработку инструментов автоматизации, которые должны следовать существующей практике и итнергрироваться в существующую учебную систему. Сами инструменты разрабатваются на языке Python и включают такие компоненты (Применительно к курсовым и лабораторным):

- Программы проверки студенческих решений
- Генераторы условий заданий

Предметы и языки программирования, для которых вы будете вести разработку инструментов автоматизации:

- Программирование (решения студентов на C)
- Информатика и Алгоритмы и структуры данных (решения студентов на Python)
- Организация ЭВМ и систем (решения студентов на ассемблере для RISC-V)

Особенности проекта:

- Необходимость общения с заказчиком и выяснения деталей / требований остается
- Все исполнители в данном проекте - магистры.
- Если кто-то из магистров берет на себя (даже если временной и частично) функции лидера проекта, то получает за это дополнительный бонус
- Задачи магистров в данном проекте:
    - Выполнять свои задания по разработке
    - Интегрировать получаемый результат в moodle
    - Готовить эталонные решения / тесты для своих материалов
- План на итерацию для каждого магистра
    - Подготовить четыре задачи (== разработать четыре инструмента, например - сделать четыре программы для генерации и проверки четырех разных вариантов курсовой) - правила подготовки материалов будут указаны в репо
    - Критерии оценивания результата (могут отличатся, это отправная точка)
        - полнота и ясность обратной связи для студентов
        - надежность решения
        - учет крайних случаев
        - расширяемость и переносимость результата

- По итогам итерации не требуется готовить презентационные материалы / демо, НО результаты должны иметь интсртукцию / необходимые материалы чтобы их можно было проверить и развернуть в отрыве от автора

Итерации

Разработка проекта будет вестись в рамках адаптированной версии гибких методологий разработки.

Процесс разработки будет организован как четыре последовательные итерации длительностью примерно месяц.

Каждая итерация представляет собой концентрированную разработку очередной версии проекта.

Обязательная часть в любой итерации (Кроме первой итерации - там уточнение):

1. Работа с issues в репозитории
    1. Требования к работе с репо выше
    2. Задачи созданы, назначены, поставлены отметки итераций (на текущую (для первой не супер критично) и на следующую итерации)
    3. Задачи в актуальных статусах
    4. Задачи имеют описания
    5. Сообразно итерации настроена автоматизация тестирования / сборки / защиты веток
2. Подготовить презентационные материалы (разместите и презентацию (PDF), и видео в репозитории)
    1. Презентация где указан: план на текущую итерацию, результаты, план на следующую
    2. Скринкаст с демонстрацией фич проекта (не более 2 минут)
3. Общение с заказчиком - постановка задачи, предъявление результатов

Итерация 1

Сроки: 13.02.2024 - 28.02.2024 (включительно)

Задачи:

1. Выбор проектов
2. Получение доступа к репозиториям и чатам
3. Правильно подписать себя (указать в профиле github имя фамилию (== сторонний человек должен глядя на ваш профиль по нику / указанным фамилии имени иметь возможность верно догадатся, кто вы) + в дискорд сервере **Имя.Фамилия.группа**)
4. Провести установочную встречу с заказчиком
5. Подготовлена вики-страница с подробной постановкой задачи, собранными и проанализированными требованиями, сценариями использования и макетами UI
6. Работа с issues в репозитории (см. выше) - создать задачи и фичи; указать теги, версии и описания
7. Подготовить презентационные материалы (см. выше) - только презентация

Если на первой итерации команда смогла сделать хотя бы минимальный прототип - это дает дополнительный бонус.

Итерация 2

Сроки: 29.02.2024 - 27.03.2024 (включительно)

Задачи:

1. Подготовить версию 1 (частично работоспособная версия)
    1. Приложение корректно запускается (без ошибок и сбоев)
    2. Приложение реализует минимум один сценарий использования
    3. Если можно не делать авторизацию - пока не делайте или отключите по умолчанию
    4. Есть инструкция по настройке / развертыванию или скрипты для этого или dockerfile | docker-compose
2. Работа с issues в репозитории (см. выше)
3. Подготовить презентационные материалы (см. выше)
4. Ссылки на материалы (инструкция, презентационные материалы) для проверки итерации собраны в README под заголовком “**Итерация 2**”

Итерация 3

Сроки: 28.03.2024 - 25.04.2024 (включительно)

Задачи:

1. Подготовить версию 2 (почти работоспособная версия)
    1. Требования версии 1
    2. Приложение реализует половину от согласованных сценариев использования
    3. Реализованы базовые тесты (интеграционные, функциональные), желательно через GitHub Actions. Юнит-тесты можно, но как дополнение.
2. Работа с issues в репозитории (см. выше)
3. Подготовить презентационные материалы (см. выше)
4. Ссылки на материалы (инструкция, презентационные материалы) для проверки итерации собраны в README под заголовком “**Итерация 3**”

Итерация 4

Сроки: 26.04.2024 - 29.05.2024 (включительно)

Задачи:

1. Подготовить версию 3 (максимально работоспособная версия)
    1. Требования версии 2
    2. Приложение реализует не менее 90% от количества сценариев использования
    3. Финализированные тесты + автоматизация из запуска через Github Actions (или иной способ)
2. Работа с issues в репозитории (см. выше)

3. Подготовить презентационные материалы (см. выше)
4. Ссылки на материалы (инструкция, презентационные материалы) для проверки итерации собраны в README под заголовком “**Итерация 4**”

Пояснение про сценарии использования и макеты UI

Теория - Презентация про то, как составлять макет и писать сценарии использования (+типичные ошибки)

Вопросы:

1. **А если мое приложение не подразумевает интерфейс пользователя в явном виде?** Обсудите с заказчиком, можно ли сделать хотя бы какой-то отладочный интерфейс для **пользователя** (CLI например) и макетируйте его. Если такие интерфейсы придумать не удается, то возникает вопрос - а как вообще проверить ваши результаты (обсудите с заказчиком).