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 полное представаление обо всем цикле разработки ПО.

В больших корпорациях за счет накопленных ресурсов 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 продукта (== отстутсвие сбоев),
- Отчуждаемость и осязаемость результатов (== 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,
- контролирует, что его команду и отдельных участников справедливо оценили, инициирует решение проблем,
- 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 почту с соответствующей темой и полным описанием ситуации. !!

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, при необходимости - подключатся для обсуждения к команде
- Оценивание промежуточного (По итогам итерации) 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

Распределение по командам и проектам

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 :(

Свою тему предложить в данном курсе тоже 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. Необходимо использовать репозиторий или созданный преподавателем, или предоставленный заказчиком.

Процедура распределения в команды и по проектам

Часть студентов до распределения (Или даже до начала курса) будут распределены в проекты повышенной сложности.

- В помощь для коммуникации и мирного разделения участников и проектов создается чат с названием «**mse-поиск-команды**». Чат это место для студентов, где они сами между собой договариваются (или не договариваются).
    - Преподватели никак не используют содержимое этого чата для распределения по проектам (не читают, не заглядывают и отказываются воспринимать любым способом) - смысл чата в том, чтобы студенты сами между собой договорились.
- Преподаватели озвучивают срок, когда будет открыта форма выбора и когда ее закроют
    - Студенты (**один** человек из команды) вносит состав команды И указывает в форме список **из всех номеров проектов** в порядке убывания интереса к ним (от самых интересных, до наименее интересных).
        - Повторные заполнения формы одним и тем же человеком И/ИЛИ заполнение формы другими людьми из команды будут удалены из таблицы.
        - Команды, **которые не укажут список из ВСЕХ номеров проектов** - отмечаются как желающие работать над рандомным проектом.
        - Команды, **где меньше пяти человек бакалавров** - отмечаются как желающие на добавление неприкаянных бакалавров (== тех, которые никак в форме не отметились).
        - Команды, **где больше пяти человек бакалавров** - отмечаются как группа неприкаянных баклавров (== ваш выбор проектов и магистра аннулируется, бакалавры распределяются по командам и проектам в случайном порядке).
    - Заполнить форму можно только один раз.
- После закрытия формы, преподаватели распределяют команды по проектам с помощью следующего алгоритма:
    - Определение команд:
        - Из результатов заполнения формы составляется список
            - полных команд (== где указано нужно число участников)
            - неполных команд (== где указано меньшее чем нужно число участников)
            - неприкаянных студентов (== студенты, которые ни в одной команде не упомянуты или попали в эту категорию, потому что в команде их оказалось больше нужного числа)
        - Неполные команды дополняются неприкаянными студентами случайным образом
        - Если после предыдущего шага остаются неприкаяные бакалавры - из них случайным образом формируются команды
    - Распределение проектов по командам
        - Команды делятся на три группы -
            - Заполнившие форму корректно (указавшие список из ВСЕХ номеров проектов),
            - Заполнившие форму некорректно (== есть проблемы со списком приоритетов),
            - Прочие команды (сформированые из прикаянных студентов)
        - Среди **команд, заполнивших форму корректно**, происходит распределение тем на основании времени заполнения формы И приоритетов выбора. Принцип - кто раньше заполнил, тому проект и достался.

- **Оставшиеся проекты распределяются случайным образом среди остальных команд** (некорреткно заполнивших форму и полностью состоящих из неприкаянных)
- Поскольку в данном курсе проектов меньше, чем участников, то остаток бакалавров распределяется в проекты **FastTrack**.

Работа в репозитории

Как работать в репо:

1. Главная ветка - main ИЛИ 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. Любые мержи только и исключительно через PR
5. 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 (если не получается настроить - обратитесь к преподавателям)
4. Каждая issue подразумевает индивидуальную работу. Issue нужно создавать и планировать так, чтобы у каждой был только один исполнитель (assignee).

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 интергрироваться 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

В части проектов ставятся задачи с большим уклоном в DevOPS.

Особенности проекта:

- Необходимость общения с заказчиком и выяснения деталей / требований остается
- Все участники в данном проекте - исполнители
- Если кто-то из участников берет на себя (даже если временной и частично) функции лидера проекта, то получает за это дополнительный бонус
- Задачи 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
  - 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

1. Разработка проекта будет вестись в рамках адаптированной версии гибких методологий разработки.
2. Процесс разработки будет организован как четыре последовательные итерации длительностью примерно месяц.
3. 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 (https://se.moevm.info/doku.php/courses:mse:idea_and_assignments#%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0_%D0%B2_%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B8)
   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**
   1. Презентация (PDF) где указан:
      1. план на текущую итерацию,
      2. результаты текущей итерации,
         1. Укажите конкретно, что было сделано + скрины / ссылки
      3. план на следующую итерацию
         1. В плане должны быть конкретные задачи по продукту - пофиксить это,

добавить такую то фичу ….
2. В плане не должны быть задачи - созвонится с заказчиком, переварить его замечания, подготовить отчет, уточнить требования с учетом принятых решений (потому что это техническая работа, которая заказчику не интересна)
3. В плане не должны быть задачи без понятного результата и способа проверки достижения этого результата: изучить что-то, разобраться с чем-то , подумать о чем-то
2. Скринкаст с демонстрацией фич проекта (не более 2 минут). Скринкаст надо сжать и залить в репо.
3. Разместите все материалы (и презентацию (PDF), и видео, и другие ВНЕШНИЕ документы нужно продублировать в репо) в репозитории в отдельной ветке **reports**
   1. На каждую итерацию создавайте PR для ветки reports (reports → main), мержить его не нужно
   2. В ветке reports создайте файл **reports.MD** в корне репозитория. Укажите в нем ссылки на материалы по каждой итерации (материалы это презентации, скринкасты, документы, вики страницы и тд).
3. Общение с заказчиком - постановка задачи, предъявление результатов

Ответ на вопрос: «**Зачем нужна ветка reports, если у нас в репо есть wiki?**». Смысл данной ветки в том, чтобы собрать в одном месте (одно место это файл reports.MD) все матералы по каждой итерации. Ваши заказчики (как и реальные заказчики) не хотят, не любят и не будут блуждать по репозиторию, разыскивая отчетные материалы. Им нужна одна понятная, общая форма отчетности, которую они могут легко найти и в которой можно быстро и без больших умственных усилий увидеть сухой остаток работы команды на той или иной итерации.

Ответ на вопрос: «**В чем смысл презентаций по итерациям?**». В реальных проектах, как правило, оплата работ происходит по этапно + заказчики (Не смотря на все их особенности) хотят иметь контрольные точки для понимания за что они платят свои деньги исполнителям (вашей команде). Обычно, такие этапы / контрольные точки организуют в формате демо и короткой презентации, где команда объясняет, какие содержательные результаты по проекту она получила за прошедший период. Задачи исполнителей - **показать осязаемый прогресс и «товар лицом»**. Задача заказчика - **понять, а есть ли прогресс, достигнуты ли реальные успехи, заслуживают ли исполнители оплаты**. Часто заказчики (когда это крупные организации) дают свои шаблоны для отчетов / презентаций и даже демо (чтобы вся иерархия начальников у ЗАказчика тоже могла потом это оценить, принять решение, подшить в архив и тд).

Итерация 1

Сроки: 12.02.2025 - 26.02.2025 (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)
4. Провести установочную встречу с заказчиком (подсказка для тех, у кого на одном

заказчике много проектов - будет продуктивнее, если вы объедините силы с другими командами чтобы за одну встречу спросить по всем проектам)
5. Подготовлена вики-страница (или если ее нельзя сделать - MD файл в ветке **reports** (см. выше)) с подробной постановкой задачи, собранными и проанализированными требованиями, сценариями использования и макетами UI. В числе прочего на этой странице явно указано:
    1. Описана основная проблема, которую решает проект
    2. Указаны категории пользователей, что для каждой из них важно в проекте
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
8. Бакалаврам (== каждый бакалавр должен принять участие **и запушить код в репозиторий**, если не понятно как - магистры уточняют у преподавателя)
    1. ИЛИ создать и запушить примеры по используемым технологиям в playground (код и инструкция в md формате о том, как это запустить и что оно делает)
    2. ИЛИ создать и запушить прототип на заглушках / заготовку будущего проекта .

Если на первой итерации команда смогла сделать работоспособный прототип - это дает дополнительный бонус.

Итерация 2

Сроки: 27.02.2025 - 26.03.2025 (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, а также 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 reports.MD (из ветки reports) под заголовком «**Итерация 2**»

Итерация 3

Сроки: 27.03.2025 - 12.05.2025 (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 reports.MD (из ветки reports) под заголовком «**Итерация 3**»

Итерация 4

Сроки: 13.05.2025 - 26.05.2025 (включительно)

Задачи:

1. Подготовить версию 3 (максимально работоспособная версия)
    1. Требования версии 2
    2. Приложение реализует не менее 80% от количества сценариев использования
    3. Финализированные тесты (автоматизация из запуска через Github Actions (или иной способ) - желательна, но не обязательна)
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 reports.MD (из ветки reports) под заголовком «**Итерация 4**»

Данная итерация оценивается приемущественно по обратной связи Заказчика. ЗАказчик оценивает итоговое впечатление о проекте (и данная итерация - ваша возможность по согласованию с Заказчиком впечатление улучшить через какие-то доделки).

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?**
    1. 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).
    2. Если вы делатете консольную утилиту - опишите консольный интерфейс
    3. Если вы делаете какую-то совсем внутреннюю штуку без интерфейсов во вне,то
        1. Опишите юзкейсы подробнее (Как ее работа влияет на пользователя, в каких кейсах и тд)
        2. Нарисуйте схему системы и покажите место вашего результата
        3. Опишите основные программные интерфейсы (какие будут у вас классы, как в коде ваши наработки будут подключатся к остальной системе)
2. **Можно ли описывать юзкейсы / макеты не в том формате, как в презентации?**
    1. Можно, но посогласованию с заказчиком

Как преподаватели проверяют итерации и формируют оценки за них

Каждый проект по итогам итерации будет проверен одним из преподавателей. Смысл проверки:

- Проверить наличие необходимых материалов и проверить, что выполнены требуемые действия (в том виде и в том формате, как это указано на вики).
- Оценить по статистке действий в репозитории, вклад каждого участника команды (по статистике действий в репозитории).
- Оценить, насколько команда своими действиями увеличивает автобусный фактор (это хорошо) или уменьшает его (это плохо). Это включает в себя оценку качества ведения задач, планирования, документирования информации, полученной от заказчиков.
- (больше для итераций 2-4)
    - Оценить качество протипа по категориям осязаемости и отчуждаемости результатов (понять, насколько прототип готов для использования другим человеком, насколько он понятен постороннему).
    - Насколько команда учла и исправила замечания с предыдущих итераций.

From:
https://se.moevm.info/ - **МОЭВМ Вики [se.moevm.info]**

Permanent link:
https://se.moevm.info/doku.php/courses:mse:idea_and_assignments

Last update: