СОДЕРЖАНИЕ

ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ПРОДУКТА «ФАКТОР»

Для развития и поддержание жизненного цикла программного продукта "Фактор" в команде должны быть роли:

- Продакт оунер
- Java-разработчик
- Тестировщик
- Инженер сопровождения

Любая ошибка или улучшение оформляется в виде задачи.

Несколько задач формируют релиз.

Ниже описан процесс работы с задачами на "Фактор".

Ставим

Самое главное — написать историю, откуда пришла задача, в чём боль заказчика, зачем всё это вообще нужно.

В задаче не должно быть способа решения. Как лучше сделать — решает исполнитель.

Делаем

Можно приступать, если:

1. Знаем, как заказчик будет использовать.
2. Решили, как показать заказчику, как продемонстрировать.
3. Есть цели, требования, тесты. Если чего-то нет — задачу делать рано, надо обсудить и добавить.

Закрываем

Важно первые результаты тестирования получить как можно быстрее. Начинаем тестирование ключевых резолведов не позже 8 рабочих часов. Если не успеваем, отдаем что-то коллегам-QA.

- При актуализации тестов удалять избыточные тест-кейсы заказчиков.
- Делать внутреннее тестирование, потом отдавать внутренним заказчикам.

Проверяем, что

1. Требования реализовали, функционал работает.
2. У всех заказчиков обновили настройки, связанные с изменением general-функционала.
3. Есть/добавили автотесты и они актуализированы/проходят.
   a. unit-тесты есть и подключены в CTS;
   b. функциональные тесты есть;
   c. есть тесты на производительность;
   d. крупные задачи проверены на большом объёме данных: нет NPE, скорость устраивает;
   e. конфиги актуальны.
4. Дополнили документацию.
5. Примеры из документации есть в автотестах.
6. Сделали ревью кода, если требовалось.
7. Написали закрывающий комментарий, если задача упомянута в релиз ноутс.
8. Показали на http://factor.hflabs.ru/address-compare, если применимо.

Работаем с ветками и коммитим

- Основная разработка идёт в master, но не стоит превращать masterв помойку.
- Когда ключевые задачи релиза сделаны в части разработки и не позже, чем за 6 рабочих дней до конца месяца, создаём ветку. Начинается разработка следующего релиза в мастере.
- Изменения по задачам делаются в самой маленькой версии и протягиваются до master. Распространенные сценарии по работе с коммитами есть в FAQ по работе с GIT

Перед коммитом

- Сообщить коллегам о планах, чтобы было минимум наложений.
- Проверить, что тесты не падают.
- Проверить, что нет других коммитов. Если есть, запустить тесты перед запуском со своими правками.
- Запустить локально или через remout run:

  базовый тест на функциональность

  Critical Test Suite

  тесты двух заказчиков

Планируем и расставляем приоритеты

- Переоткрытые задачи фиксить в 1ую очередь
- Из незапланированных задач делать только срочно нужные заказчикам

Задачи поддержки

- Все QA держат команду в курсе процессов у Заказчика, чтобы в сложный момент кто-то мог быстро перехватить поддержу
- Блокер проблемы у Заказчика решать командой
- Включать известные задачи поддержек в релиз
- Делить задачи по поддержкам между 2мя людьми
- Задачи на поставки проверять и закрывать, не дожидаясь установки Заказчиком

ЗАКРЫВАЕМ

Закрываем

Важно первые результаты тестирования получить как можно быстрее. Начинаем тестирование ключевых резолведов не позже 8 рабочих часов. Если не успеваем, отдаем что-то коллегам-QA.

- При актуализации тестов удалять избыточные тест-кейсы заказчиков.
- Делать внутреннее тестирование, потом отдавать внутренним заказчикам.

Проверяем, что

1. Требования реализовали, функционал работает.
2. У всех заказчиков обновили настройки, связанные с изменением general-функционала.
3. Есть/добавили автотесты и они актуализированы/проходят.
    a. unit-тесты есть и подключены в CTS;
    b. функциональные тесты есть;
    c. есть тесты на производительность;
    d. крупные задачи проверены на большом объёме данных: нет NPE, скорость устраивает;
    e. конфиги актуальны.
4. Дополнили документацию.
5. Примеры из документации есть в автотестах.
6. Сделали ревью кода, если требовалось.
7. Написали закрывающий комментарий, если задача упомянута в релиз ноутс.
8. Показали на http://factor.hflabs.ru/address-compare, если применимо.

ЗАКРЫВАЮЩИЙ КОММЕНТАРИЙ

Закрытие любой задачи для заказчика и фичи в кор должно заканчиваться закрывающим комментарием. Он оформлен по строгому шаблону.

Как понять, что для задачи нужен закрывающий комментарий? Она упомянута в Релиз ноутс релиза.

Как это улучшает жизнь

1. Сразу понятно, что сделано по задаче.
2. Ускоряется регрессия. Видны пункты, которые нужно учесть при проведении регрессии: добавить в РН и в инструкцию по переходу.
3. Быстрее собрвть релиз ноутс.

Как использовать

Можно копировать шаблон себе в блокнот или сразу в комментарий в jira и заполнять по мере работы над задачей — это позволит сразу все структурировать, а не собирать в последний момент.

Пункты не пропускаем. Если в них нечего писать, пояснить причину.

Шаблон

Общий вводный комментарий по задаче: зачем и что сделали.

h4. Разработка
h5. Что и как сделано
h5. На что обратить внимание тестировщику

h4. Документация
h5. Общие статьи Фактора
Что обновлено:
- [Название статьи|https://confluence.hflabs.ru/]

Новые страницы:
- [Название статьи|https://confluence.hflabs.ru/pages/viewpage.action?pageId=629932114]

h5. Описание CR в спейсе заказчика
Что обновлено:
- [Название статьи|https://confluence.hflabs.ru/]

Новые страницы:
- [Название статьи|https://confluence.hflabs.ru/pages/viewpage.action?pageId=629932114]

h4. Тестирование
h5. Что проверили руками
h5. Автотесты
[Ссылка на группу тестов 1|https://tc.hflabs.ru] --- что проверяется и зачем нужны
{code}
- test_01
- test_02
{code}

h4. Для Release Notes
Смысл доработки для бизнеса

h4. Для инструкции по обновлению
О чем необходимо обязательно упомянуть в инструкции

h4. Изменения в программно-аппаратной платформе
Какие требования для ПАП изменились с пруфами. Например, обязательный переход с HDD на SSD из-за <причины>.
Перечень задач для QA по своим заказчикам, для которых нужно проверить соответствование их ПАП новым требованиям и предупредить о новых требованиях.

h4. Куда изменение было закомичено (ветка и ревизия)
Необходимо для истории, чтобы быстро потом понимать ушло ли изменение заказчику
Factor: master(19.12+) #767964a13cf270f8942a0a8e7470dcc2af1b0669
19.7 #4567964a13cf270f8942a0a8e7470dcc2af1b0669

По пунктам «Документация» и «Тестирование» пояснения не требуются.

Из пункта «Для Релиз ноутс» должно быть понятно, чем задача полезна для бизнеса (очень круто, если там ссылка на фичу в РН). Если про доработку не нужно писать в Релиз ноутс, то необходимо указать причину.

В пункте «Для инструкции по обновлению» должен быть готовый текст для инструкции. Если ничего добавлять в инструкцию не нужно, то должна быть указана причина.