# Цикл разработки через тестирование

1. В тикетной системе заводятся четыре фиктивных пользователя: plan, test, dev и ref.

Эти пользователи используются в качестве очередей заданий

1. В очередь plan ставятся крупные задания, требующие создания класса, модуля или плагина
2. Программист выбирает из очереди plan любое понравившееся ему задание: переписывает тикет на себя
   1. Создаёт пустой класс, модуль или плагин. Пустой – в смысле, без функций и методов
   2. Запускает уже существующие unit-тесты, чтобы убедится, что нет ошибок синтаксиса, конфликта имён и т.д.
   3. Комитит изменения в главную ветку (master). Комментарий к комиту – «plan номер тикета»
   4. Создаёт описание класса/модуля/плагина в wiki. В результате появляются названия необходимых методов или функций и их описание – для чего они предназначены
   5. На каждый метод или функцию программист создаёт тикет в очереди test
3. Программист выбирает из test любое понравившееся ему задание: переписывает тикет на себя
   1. Создаёт ветку (branch) в системе контроля версий (svn, git) названием которой служит номер выбранного тикета
   2. Пишет для указанной в тикете функции unit-тест
   3. Запускает этот unit-тест – проверяет, чтобы он выдавал ошибку (E). Если тест выбрасывает исключение (F) или не выдаёт ошибки, то его нужно исправить
   4. Cоздаёт коммит с этим тестом и отправляет ветку в удалённый репозиторий. Комментарий к комиту: «test номер тикета»
   5. Тикет перемещает в очередь dev
4. Выбирает из очереди dev понравившийся тикет (исключая тот, unit-тест для которого он сам написал): переписывает тикет на себя
   1. Забирает из удалённого репозитория ветку с № тикета
   2. Читает задание в тикете и unit-тест, и пишет функцию, указанную в задании, пока unit-тест не проходит
   3. Делает комит и отсылает его в удалённый репозиторий. Комментарий к комиту: «dev номер тикета»
   4. Тикет ставит в очередь ref
5. Выбирает из очереди ref понравившийся тикет (но не тот, который только что туда отправил)
   1. Забирает из удалённого репозитория ветку с номером тикета
   2. Делает рефакторинг unit-теста и функции, если это нужно. Делает комит, если был произведён рефакторинг. Комментарий к комиту: «ref номер тикета»
   3. Сливает (merge) ветку c основной веткой (master). Если возникли конфликты, то устраняет их
   4. Проверяет: все ли unit-тесты работают
   5. Если устранялись конфликты, то делается комит
   6. Отправляет ветку master в удалённый репозиторий
6. Переходит на пункт 3, 4, 5 или 6



О – какое-то заданьице скинули



Пришло задание на класс

Рефакторинг метода

Написан сам метод

И опять всё по новому…

Тестировать, тестировать и ещё раз тестировать. Но я хитрее дедушки Ленина

Написан unit-тест для метода

Из одного задания получилось несколько, так как у класса много методов

Plan

Ref

Dev

Test