Разработка на софтуер

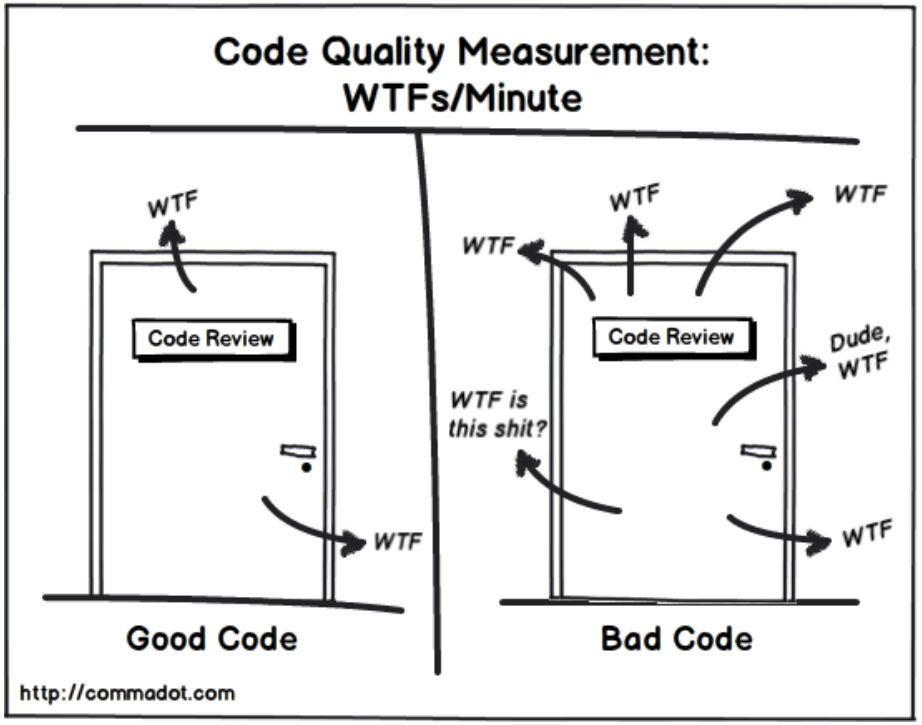
Лекция 12 – Тестване

Милен Спасов

Защо има бъгове в кода?

- Сложността на проектите нараства постоянно
- Честа смяна на изискванията
- Динамиката на Agile процеса
- Крайни срокове и компромиси в качеството/архитектурата/имплементацията
- Липса на автоматизация
- Хората правят грешки

- Лошото качество на кода води до:
  - Загуба на репутация
  - Загуба на пари
  - Трудна поддръжка
  - Повишена сложност и ниска предвидимост при разработка на нови функционалности



Видове тестване според перспективата

- **Black box**
  - Тества функционалността на системата от гледна точка на потребител
  - Тестера няма знание за имплементацията
  - Не е нужно знание за програмиране
  - Тестера знае какво трябва да прави проекта, но не как е реализирано технически

- **White box**
  - Тества вътрешната структура на системата
  - Unit test е най-очевидният пример за такъв вид тестване
  - Тестера знае какво трябва да прави системата и точно как е реализирано технически
  - Необходимо е задълбочено техническо познание
  - Тестера избира подходящи данни и параметри, за да тества различните пътища на логиката на системата

- **Gray box**
  - Комбинация между двата подхода
  - Полезно е, защото позволява функционално тестване, заедно с тестване на архитектурата и имплементацията на системата

Видове тестване според приложението им (1)

- **Unit testing**
  - Тества дали индивидуални компоненти от кода изпълняват изискванията си.
  - Обикновено се прави от прогамиста, който имплементира функционалността.
  - Тества компонента в изолация и симулира външните зависимости.
  - Добър начин за откриване на дефекти възможно най-рано.

- **Integration testing**
  - Тестване на група компоненти или модули след интеграцията им.
  - Изпълняват се след unit тестването и преди функционалното тестване.
  - Потвърждава, че модулите работят и заедно, не само отделно.

- **Functional testing**
  - Тестване на цяла функционалност от системата.
  - Няма нужна от детайлно познаване на имплементацията и технологията.
  - Тестът е насочен към проверка на функционалността от гледната точка на потребител на системата.
  - Това е тест срещу описанието във всяко user story.

- **System testing**
  - Тестване на завършена и интегрирана система.
  - Предвидено за тестване отвъд функционалното описание на системата.
  - Тества поведението на системата при интеграция с външни системи.

Видове тестване според приложението им (2)

- **Static and dynamic code analysis**
  - Помага на програмистите да проверят кода преди ревю и интеграция.
  - Може да наложи правила на програмиране и стилизация.
  - Може да засече излишен код или логически грешки.

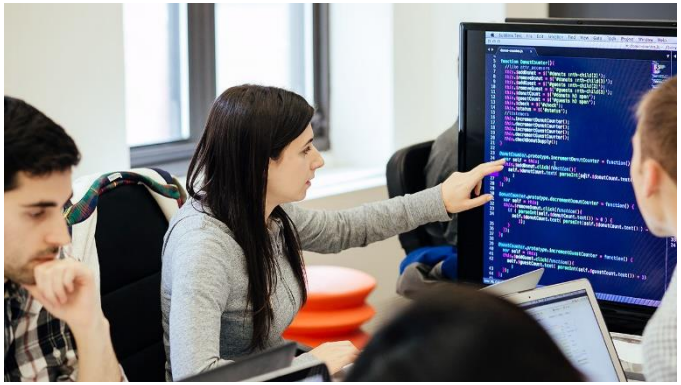
- **Smoke testing/Sanity testing**
  - Бърз тест за проверка на нова версия на проекта.
  - Извършван като бърз тест на всяка нова версия за потвърждение дали основните функционалности работят.
  - Насочен към откриване на прости грешки, които са достатъчно сериозни за отхвърляне на новата версия.

- **Regression testing**
  - Проверява функционалности, които са разработени преди и не са променяни в новата версия.
  - Насочен към откриване на проблеми причинени от нови върху стари и непроменяни функционалности.
  - Може да бъде трудно за изпълнение ако не се автоматизира.

- **Load testing**
  - Подлагане на системата на стрес (брой заявки, големи обеми данни и т.н.) и измерване на поведението й.
  - Помага да се определи капацитета и лимитите на натоварване на системата.

Какво е QA?

?



Какво прави един QA и какво е Test Case

- **За да си свърши работата един QA минава през следните стъпки**
  - Анализира изискванията на клиента и user story-тата, по които се работи и ще бъдат тествани
  - Взима предвид изисквания, които може да не са функционални или експлицитно написани (напр. брой паралелни потребители, време за реакция на системата, използваемост от потребителя и др.)
  - Прави стратегия за тестване на проекта (еднократно ако няма такава)
  - Прави план за тестването
  - Като част от тестовия план може да направи regression test
  - Пише test cases за всеки отделен тест
  - Изпълнява ги и отчита резултатите
  - Логва бъгове за всички несъответствия
  - Проследява и валидира вече оправени бъгове

- **Test case – последователност от стъпки за изпълнението на теста**
  - Test preconditions – въвеждане на системата в начално състояние за теста
  - Test execution – описание на стъпките за тестване
  - Expected result – очакван резултат
  - Actual result – наблюдаваният реален резултат
  - Test postconditions – стъпки след края на теста (напр. изтриване на данни или оставяне на системата в някакво състояние)

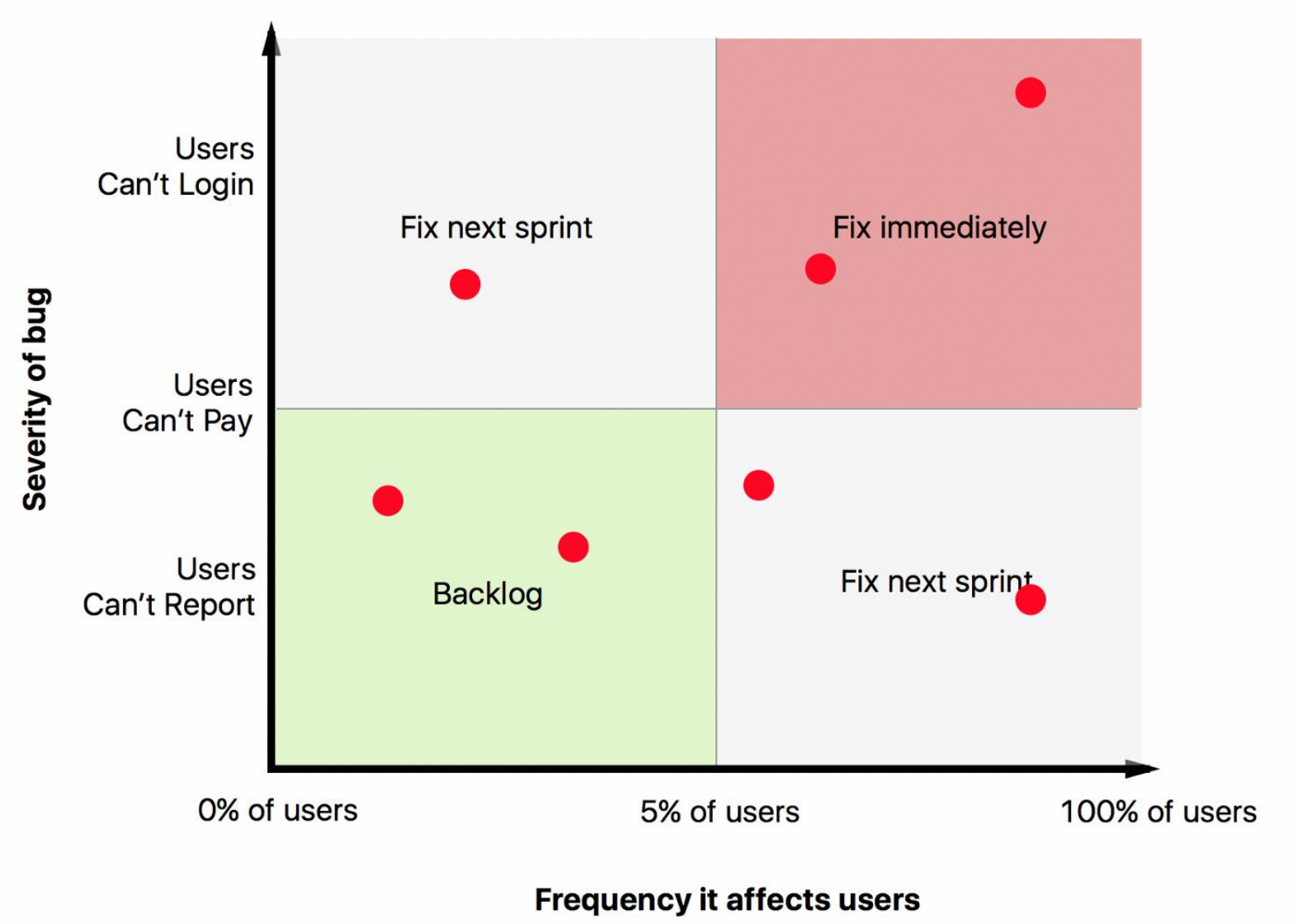
Ръчно vs. автоматизирано тестване

- **Manual**
  - (+) По-вероятно да се открият смислени за потребителя проблеми.
  - (+) Краткосрочно е по-евтиното и лесно решение.
  - (+) По-гъвкаво и с възможности за много различни видове тестове.
  - (+) Може да се извърши от по-неопитен екип.
  - (-) Не могат да се преизползват или да се изпълняват всеки път от началото до края, защото отнемат много време.
  - (-) Някои неща са трудни, бавни и непрактични за тестване на ръка.

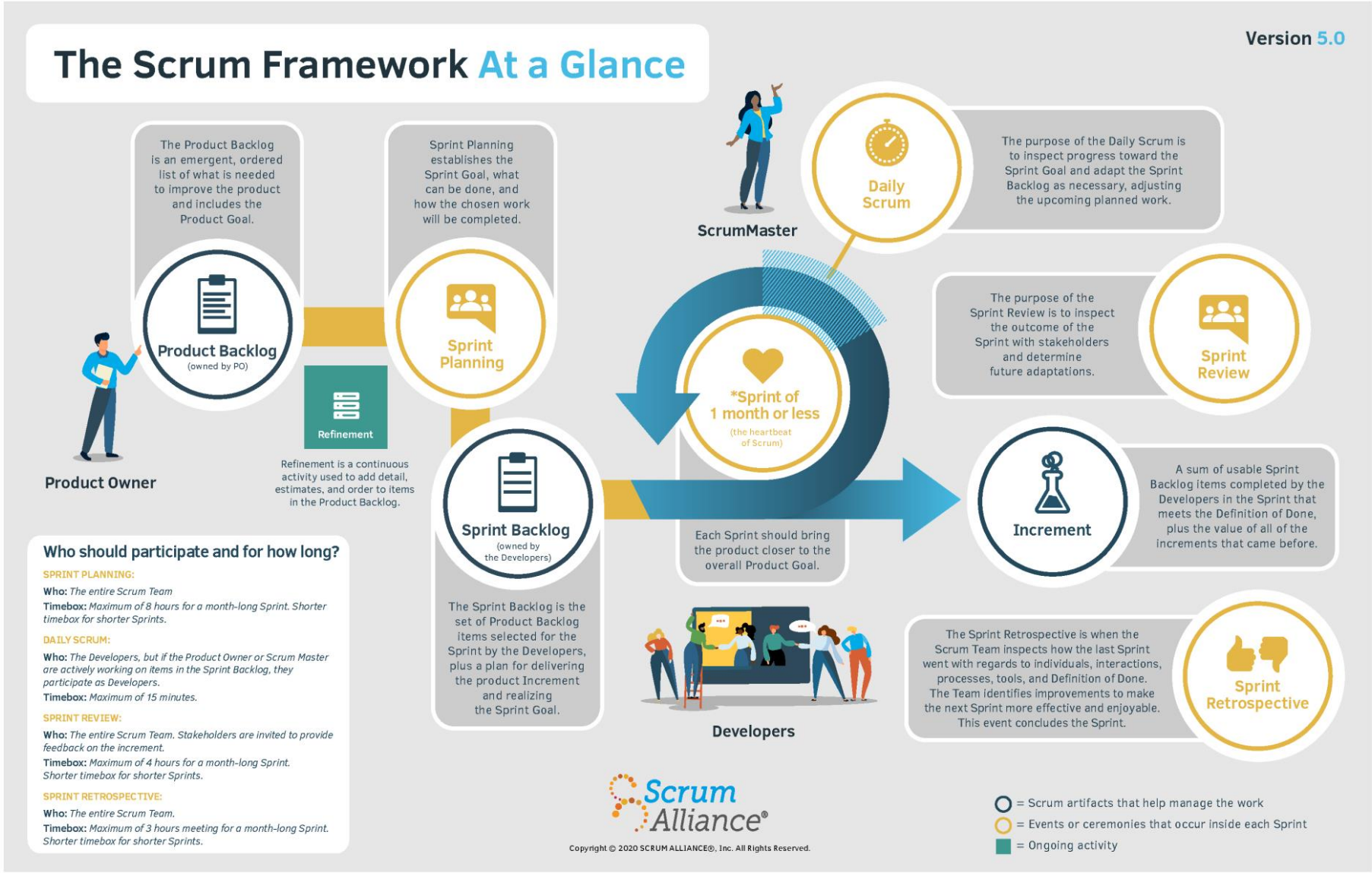
- **Automation**
  - (+) Изпълняват се всички тестове бързо и ефективно.
  - (+) Позволява на QA да изпълнява всички тестове на всяка нова версия, което не би било възможно с ръчни тестове.
  - (+) В дългосрочен план е по-изгодно и осигурява по-добро качество.
  - (-) Някои tool-ове за автоматично тестване имат ограничения и са скъпи.
  - (-) Не са подходящи за всеки проект – например за система в много начален етап на разработка, която постоянно се променя или за проект, за които не се предвижда дълъг roadmap или поддръжка.

Приоритизация на бъгове

Приоритизация на бъгове



Къде в този процес се включва тестването?




Thank
You
Asante
Shukria
Dhanyavadagalu
Vinaka
köszönöm
Kiitos
Maake
Manana
Dankon
Matondo
감사합니다
Kam Sah Hammida
Dankscheen
спасибо
Dank Je
Mauruuru
Biyan
Terima Kasih
Taiku
Blagodaram
Ngiyabonga
Dziekuje
Chokrane
Diolch i Chi
Tack
Grazie
Juspaxar
Arigato
Mochchakkeram
Gracias
Tingki
நன்றி
धन्यवाद
cảm ơn bạn
Gràzas
Dàkujem
Bedankt
Ua Tsaug Rau Koj
Paldies
Gratias Tibi
Khap
Obrigado
Děkuji
Nirringrazzjak
Di Ou Mèsi
Kia Ora
Kop Khun
ありがとう
Suksama
Hvala
Djiere Dieuf
Welalin
Matur Nuwun
Rahmat
谢
谢
хвала
Danke
Merci
Go Raibh Maith Agat
Misaotra
ขอขอบคุณคุณ
Salamat
Najis Tuke
Eskerrik Asko