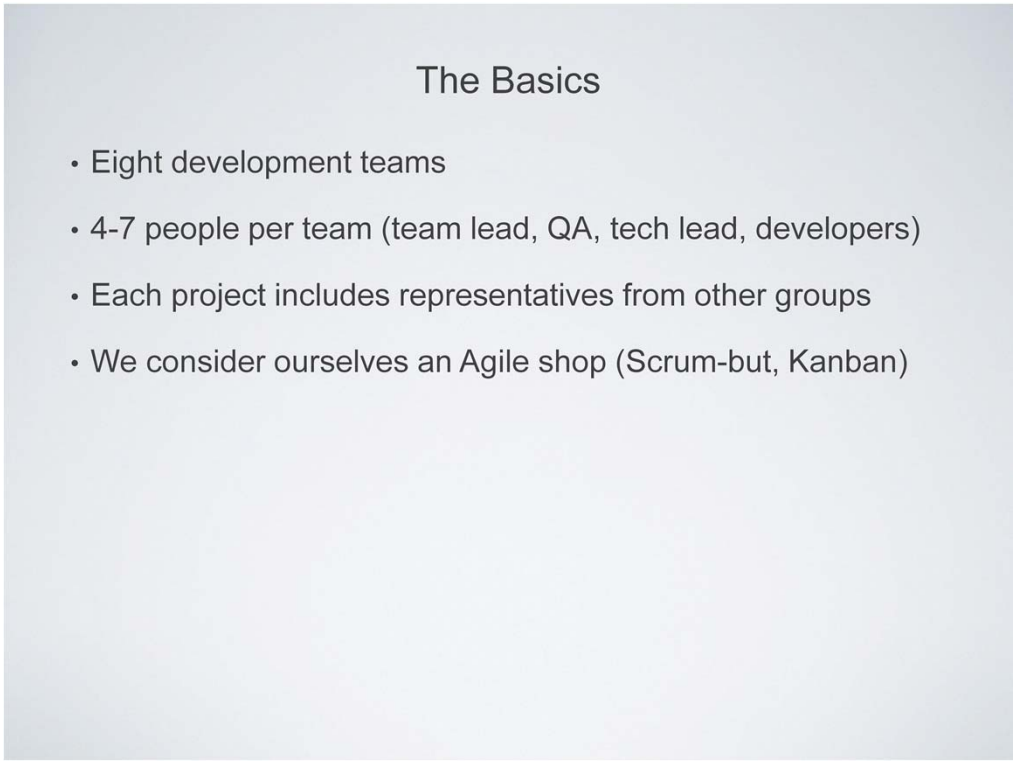
Top 10 Software Development Practices

Joe Diana
Texas Legislative Council
NALIT PDS 2013



Examples of typical team roles...

Each team has 4-7 people grouped (roughly) by business process or client group.

Team leader is responsible for leading a project and making sure the team knows what it needs to be doing on a day to day basis; team leads may not code on larger teams.

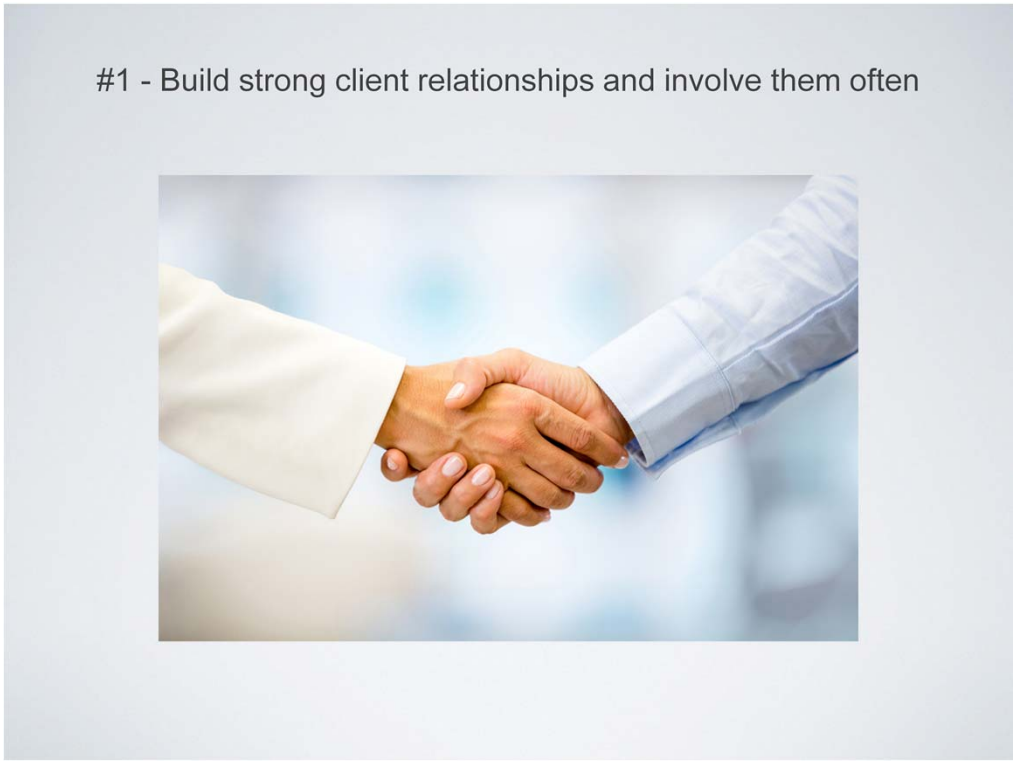
QA works with the team leader to define requirements, design mock-ups, and create testing strategies. They will review work but they are not simply testers!

Technical lead is responsible for architecture and figuring out the hard stuff.

We have 2-4 additional developers other than tech lead.

Largest team is 7 people.

We also have "team leads" from our other sections (office consultants, training/marketing, Infrastructure & Operations) to represent those groups throughout a project.



Having a good relationship with clients and understanding their business process is the starting point for a successful project.

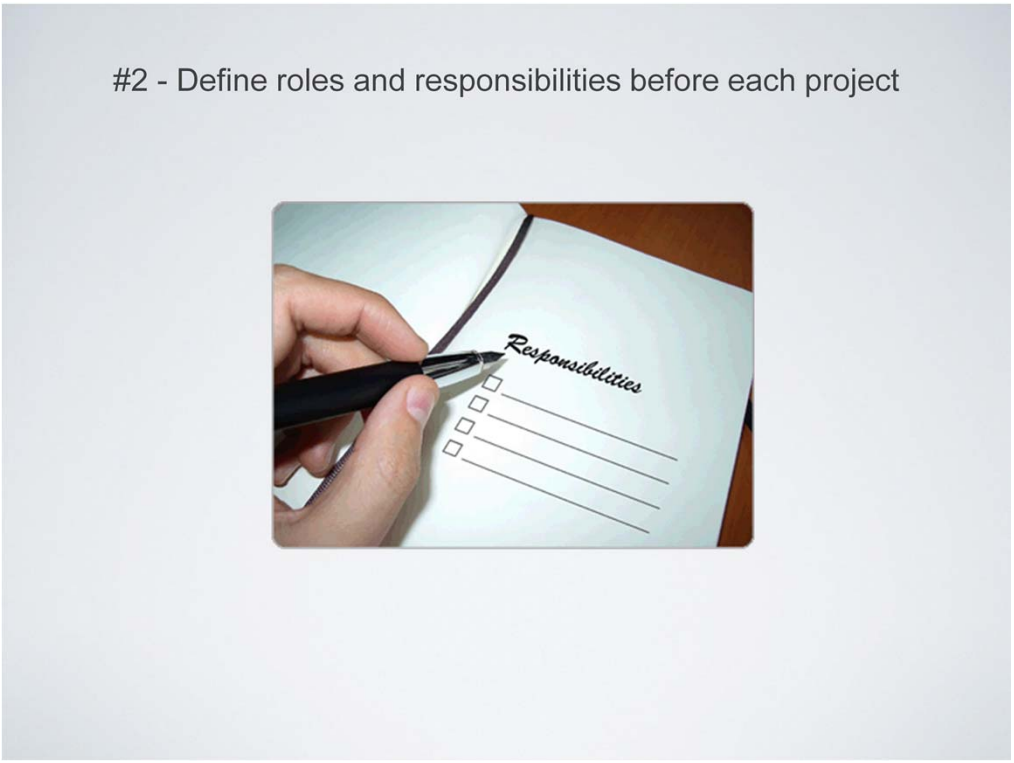
Involve clients as much as possible throughout the development process.

This will vary depending on client availability, but typically:
1) Clients have appreciated being involved in the design, review, and testing of features from the very beginning. They get more opportunity to make tweaks to the app as it's developed. This creates better client buy-in for the finished application.
2) Teams can get more timely feedback when they have a good working relationship.

Example
This may involve a weekly status meeting and separate working meetings to review mock ups and define requirements.

Clients will then see working software every 1-2 weeks and are given opportunity for hands-on testing at various stages of a project.

Disclaimer
Scope creep has to be monitored closely as clients will come up with new requests throughout.

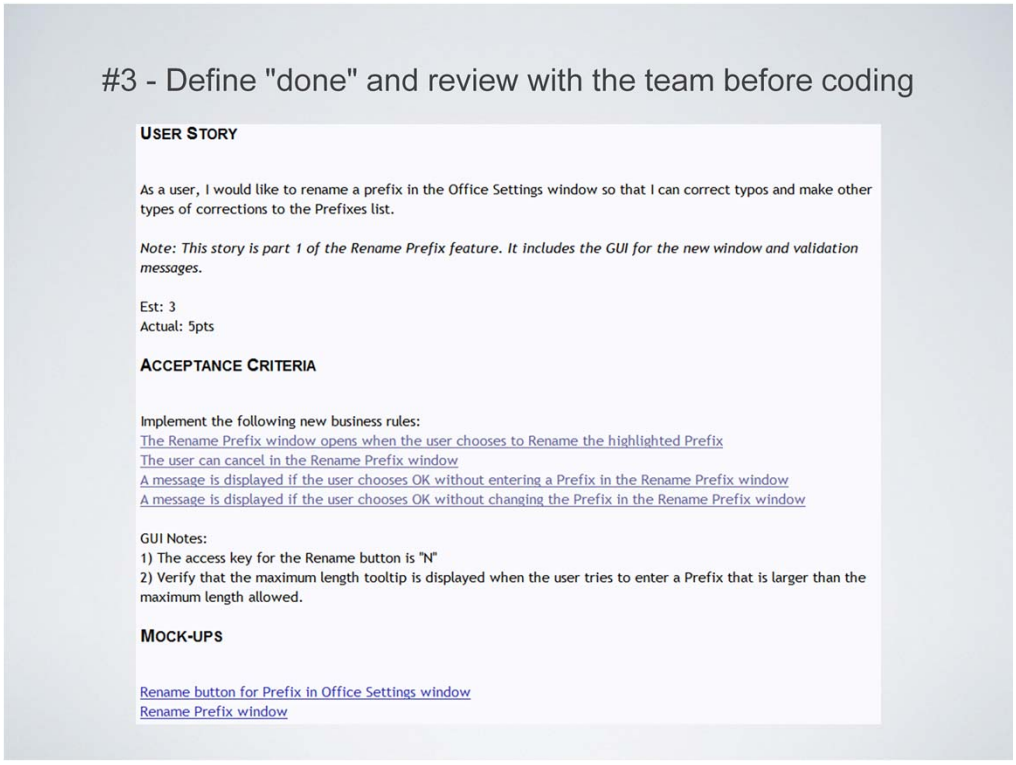


At the beginning of a project, review the roles and responsibilities of everyone involved with the project.

Why?
Unless a team has worked together for a long time and everyone's responsibilities are well understood, there is almost always some frustration or friction amongst the team because:
1) someone isn't doing what others "think they should be doing", or
2) people start getting into other people's business.

Discussing expectations, responsibilities, and time commitments at the beginning of a project can greatly reduce the potential for this type of conflict.

Make sure people know what they should be doing and then hold them accountable for it.



When we moved to Agile several years ago, one of the biggest struggles teams had was agreeing when a development task was done.

Each developer would have their own interpretations of done.

There would be discussions if it was "done" or "done-done".

Teams have found it useful to be more explicit in defining requirements (aka "acceptance criteria") and then reviewing these in a "specs workshop" to make sure everyone is on the same page.

This is done a day to a couple of weeks before a story is to be developed.

Business Rule Example

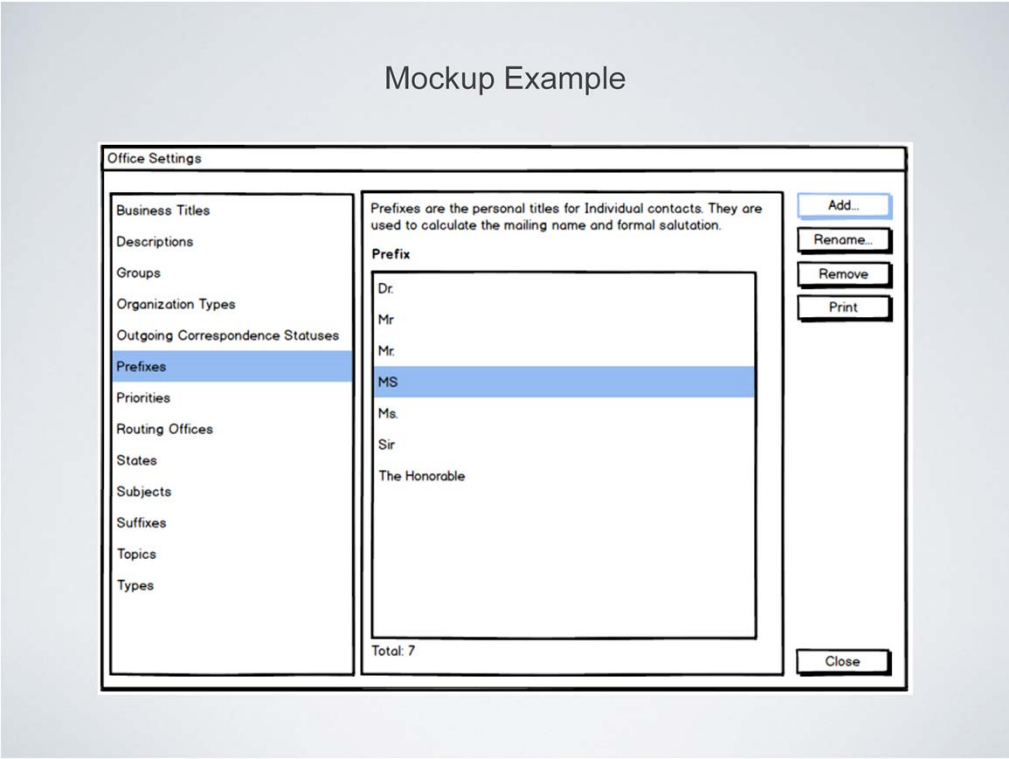
BUSINESS RULE

User can rename a prefix in the Office Settings window without a confirmation message if the prefix is NOT ASSOCIATED with any contacts.

EXAMPLES

User renames prefix that is not associated with any contacts and new prefix is NOT already in list
Given that the "Rename Prefix" window is displayed from the Office Settings window for a prefix that is not associated with any contacts
And the Prefix field is set to a prefix that is not already in the office's Prefixes list
When the user clicks OK
Then the Rename Prefix window closes
And the prefix is updated in the database
And the renamed prefix is displayed in the office's Prefixes list and is highlighted
NOTE: the prefix does not need to be in the list in alphabetical order - legacy CMS keeps it at the same position in the list

User renames prefix that is not associated with any contacts and new prefix is already in list (EX: Change "Mr" to "Mr.")
Given that the "Rename Prefix" window is displayed from the Office Settings window for a prefix that is not associated with any contacts
And the Prefix field is set to a prefix that is already in the office's Prefixes list (even if it is in the list with a different case)
When the user clicks OK
Then the Rename Prefix window closes
And the prefix is deleted from the database
And the prefix that was in the Prefix field of the Rename Prefix window is the highlighted Prefix



PROS
- This helps spread knowledge amongst the team. Less experienced developers or those lacking business knowledge get a better understanding of the system and are less likely to introduce critical bugs.
- Lots of questions can get discovered and answered before writing any code.
- A better design often results from the discussion.

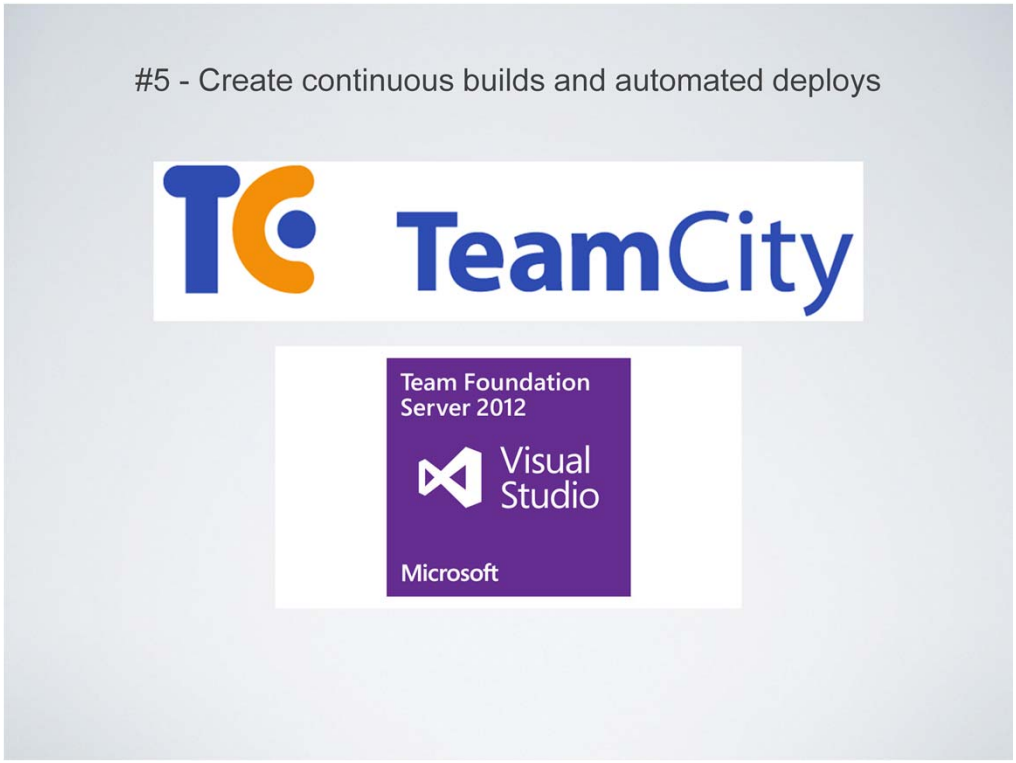
CONS
- Teams can get stuck in analysis paralysis during the specs workshop.
- Finding a concise format for the acceptance criteria is an art and varies from team to team and project to project.

*** Teams defining acceptance criteria and reviewing it before developing has contributed greatly to improved quality, design, and understanding of our systems for support ***

Teams hold a daily stand-up meeting (or scrum) for 5-15 minutes each morning to review work in progress and roadblocks.

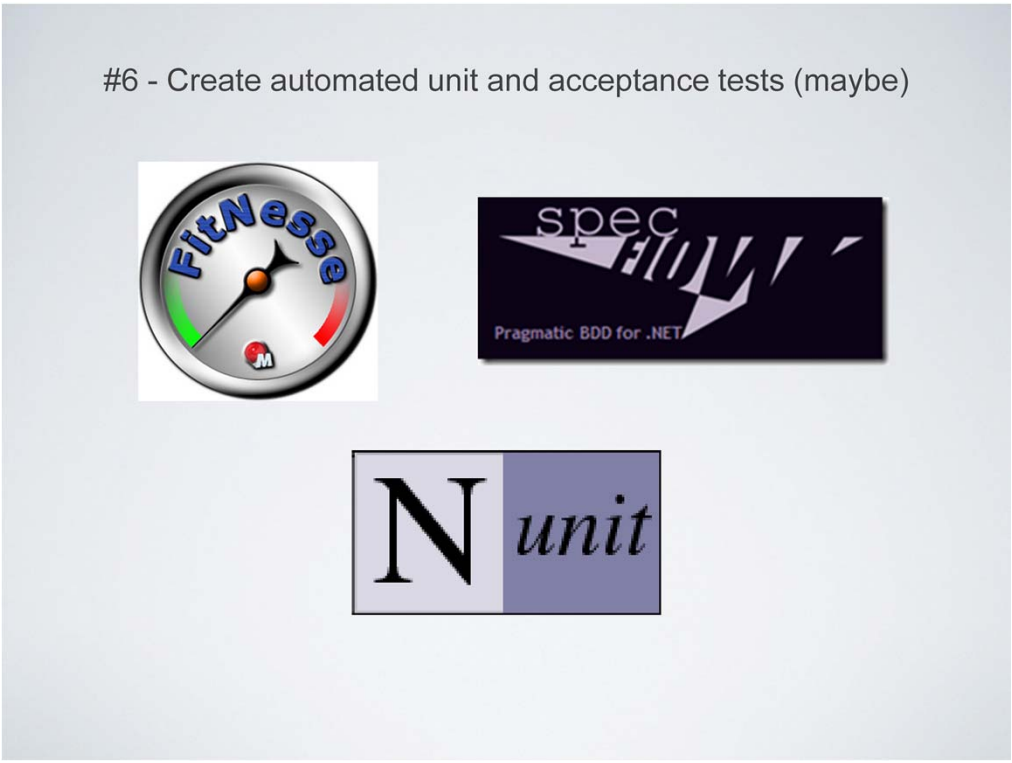
Great way, especially during session, to get an understanding of what needs to get done for the day and to review any major issues.

Teams report progress in two ways:
- Traditional scrum way: Each person says what they did yesterday, what they will do today, and if they are roadblocked on anything.
- By story: The team leader will call out each story (work item) and people will speak to what is going on with it.



We use TeamCity by JetBrains integrated with TFS to build a project every time code is checked-in and then create automated build scripts to deploy code to each of our environments.

We've spent a lot of time building automated deploy infrastructure, but it's paid off because we find issues with our builds much sooner and deploys have been easier and more reliable.



Where possible, we create automated unit and acceptance tests, but only what makes sense for the project and team.

This has been one of the most difficult and controversial things we've tried to do and you have to employ different strategies if it's a legacy application or your building from scratch.

Everybody has their own opinion on how useful it is to create automated tests and whether the focus should be on acceptance-level or unit tests. Every team does it differently.

Two Big Potential Benefits

1) Quality of the application is improved. You get instant feedback during development and have automated tests that you can run at will to help you find bugs when you change the system.
2) The architecture of the application is improved for those that do Test Driven Development and write their tests before coding anything. And with tools like ReSharper, you can generate code from your tests to help speed up your development.

Disclaimers

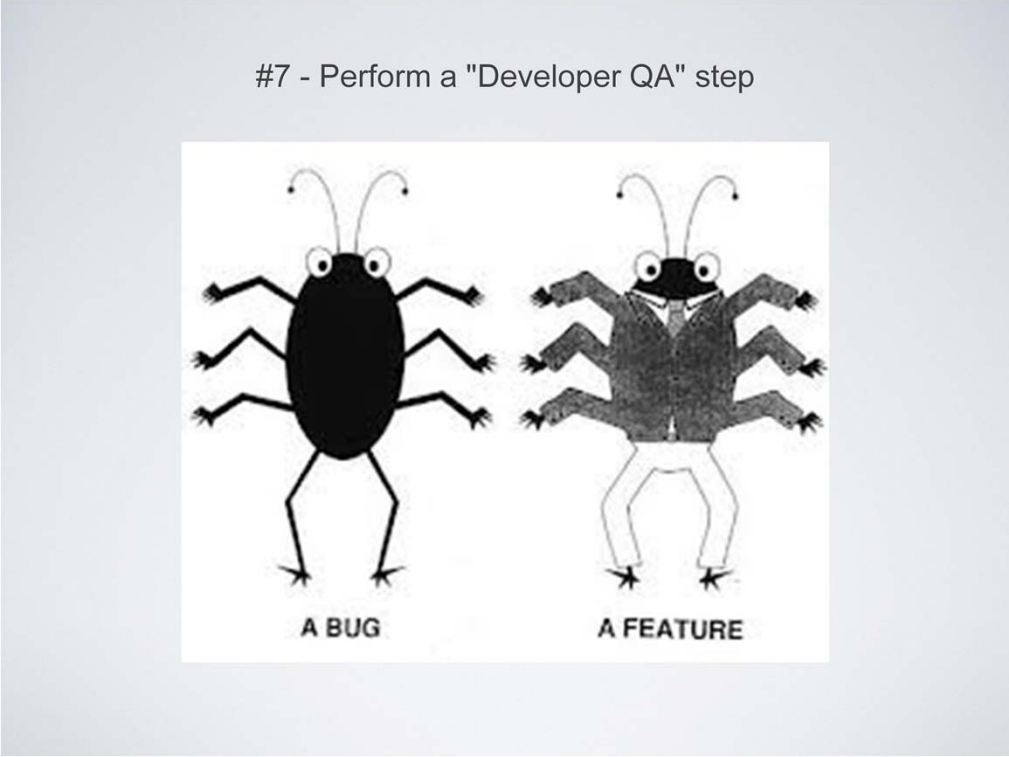
1) It's difficult to get buy-in from everyone.
2) Writing useful tests is an art. If there is not care taken to create a good testing

infrastructure, tests will break simply because of a code change and not because there is a bug. The team will spend a large amount of time maintaining tests and will get frustrated quickly.

3) Finding the right balance between automating higher-level acceptance tests and lower-level unit tests can be difficult. A team could spend a couple of years refining their process.
4) There can be a significant time investment per development task. (Unscientific/rough estimate - 15-30% to create automated tests)
5) There is still a need to do manual testing as automated tests will not cover everything (e.g. usability testing, UI code).

Bottom-Line

Automating tests is worth doing for applications that require a high-level of quality, but a team needs to invest time to learn it properly. Start small and not try to do too much at once.



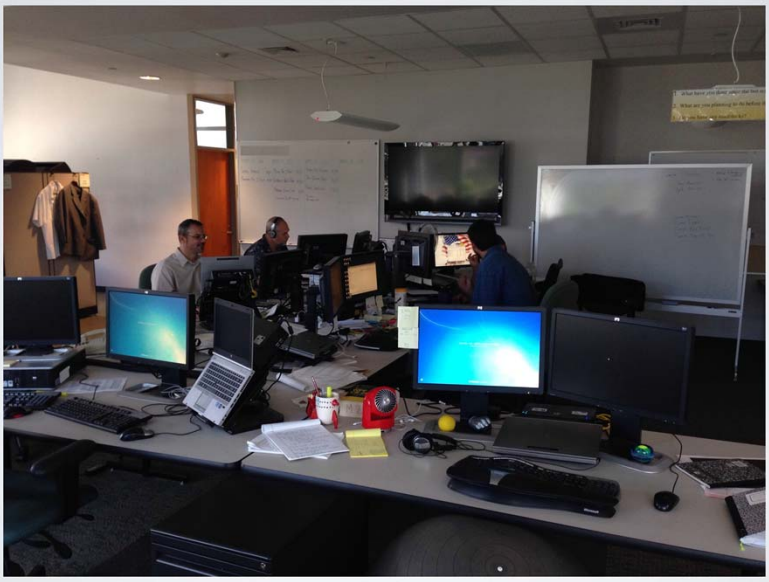
One of the most valuable practices that we've started doing is to incorporate a "Developer QA" step in our process.

After one or more developers finish a story, another developer is assigned to do a "Dev QA" review.

This entails not only testing the feature according to the acceptance criteria, but may also include reviewing any automated tests, performing a code review, doing performance testing, etc.

This process has improved quality as much as anything else as bugs are caught much earlier in the development process.

#8 - Hold informal developer test sessions for major releases



After creating a certain number of features, maybe every 4-6 weeks, a team will hold a "PQ" or Production Quality informal test session.

This involves the development team doing some exploratory and sometimes performance testing to make sure a set of features work properly.

Our goal is to find as many issues up front before the client gets the system.

This can save time because any issue found in production, even minor, can take a lot of coordination to get fixed and deployed.

We also end up providing a better user experience to the user and a system that is less buggy.

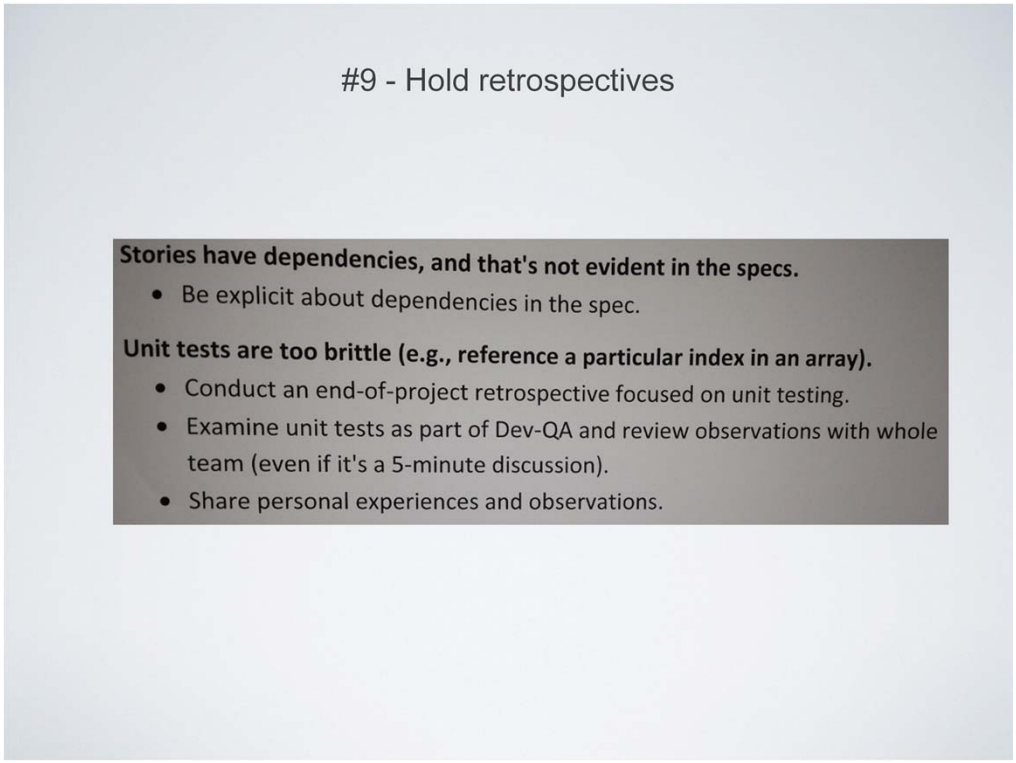
General Process
A team will dedicate 2-3 days for this starting with a test session for half a day that covers specific features and windows.

The team may focus on usability, performance, interaction with other features, general regression, etc.

Issues are added to a whiteboard or in a bug tracking database.

They then will prioritize issues found and try to fix the minor ones over the course of a day or two and then retest.

This process is lightweight, but lots of minor issues can be dealt with quickly.



As part of a desire to continually improve, teams will hold a retrospective every two weeks.

Teams also hold an end of project retrospective.

Things that have worked well
1) Teams pick 2-3 things they will commit to working on and post these on the wall in the team working area.
2) At the start of the next retrospective, the team reviews their progress on what they previously committed to working on.

For end of project reviews, teams will commit to a few things to do better into their next project.

Challenges
1) For a team that has been together for a while, these can sometimes get stale.
2) People tend to get frustrated if things are discussed but there is no action taken.

Reducing concurrent projects and daily work in progress will improve productivity, morale, and quality.

This is a core principle of Kanban and something we have been trying to work on.

As much as possible, we are encouraging teams to finish what they start and work with clients to prioritize what's next.