QE role in a cross functional team

DevConf 2019

Marek Haičman
QE and Product Owner of Security Compliance team
January 25, 2019
AGENDA

- Intro
- Problem statement
- Mitigation proposals
- Discussion
Disclaimer

This presentation is based on my observation of our Security Compliance team, and 4 or so closest teams to me, during last two-ish years.

There is no scientific research behind it, nor any literature quoted.
PROBLEM STATEMENT
Team Roles

● Quality Engineers
  ○ Responsible for the quality of the product

● Developers
  ○ Core of the team, most numerous
  ○ Responsible for developing new features, fixing bugs
Team Roles

- Quality Engineers
  - Responsible for the quality of the product

- Developers
  - Core of the team, most numerous
  - Responsible for developing new features, fixing bugs and quality
Team Roles

● Product Lead
  ○ Prioritize work
  ○ Responsible for vision and planning of the work, and quality of the product
  ○ Works collaboratively with the rest of the team

● People Manager
  ○ Performance, responsibilities
Cross functional team

- Team acts as one entity
- Prioritization of work is done by Product Lead, with inputs from others
- Team members share load - most of the tasks are defined to be actionable by anyone
  - Focused talk: How to Transform Dev and Test Silos into a Team
    (https://sched.co/JckP)
Inefficiencies we’ll talk about

- "over the wall" communication, insufficient feedback loop
- QE backlog separate entity, not considered by Product Lead
- capacity bottlenecks (either Devel, or QE)
- developers not quality-driven (bad low-level test coverage)
MITIGATION PROPOSALS
OVER THE WALL

How to break the wall?

Prerequisite: Developers consider quality to be also their responsibility.

Abandon you own silo.

- Take development tasks.
- Try development workflows.
- Contribute code to upstream
OVER THE WALL
Benefits of breaking the wall.

● Take development tasks.
  ○ Better overview of the project
  ○ Motivates cleaner specification of tickets
● Try development workflows.
  ○ Optimizations to the processes
● Contribute code to upstream
  ○ Personal growth of technical skills
QE BACKLOG HANDLED SEPARATELY

Won’t this happen magically when team is cross functional?

- The importance of quality might not be pronounced enough
- Non-product requirements, in their raw form might not be perceived as useful by the team.
  - Example: Quality specific reporting, integration into company-wide test environments.
- QE might feel like being on their own.
QE BACKLOG HANDLED SEPARATELY

Work on perception of QE tasks.

For QE tasks to be prioritized, QE needs to be seen as a good counterpart, and needs to be heard.

- Understand how to help successful delivery
  - Help translate non-product reqs into team reality
- Test coverage should be justifiable, aligned with goals
- Areas of focus fully agreed by the team
- Important bugs are important
- Strive not to be destructive
  - “This state cannot be shipped!”
QE BACKLOG HANDLED SEPARATELY

Work on perception of QE tasks.

- Talk to others, don’t try to save the world all by yourself.
- If you end up having requests which are opposite to each other, connect the requesters.
- Highlight misunderstandings, and push for finding solutions.
QE BACKLOG HANDLED SEPARATELY

Benefits of good communication.

- Promotes importance of the quality
- Team is happy when bugs are found
- Positive reinforcement of the QE role in the team
CAPACITY BOTTLENECKS

What are tasks best done by a QE?

- Writing tests
- Designing tests
- Proposing testing activities
- Running tests / review results
- Designing test plans
- Designing test strategies
CAPACITY BOTTLENECKS

What are tasks best done by a QE?

- Writing tests
- Designing tests
- Proposing testing activities
- Running tests / review results
- Designing test plans
- Designing test strategies
CAPACITY BOTTLENECKS

What are test levels QE should deal with?

- Acceptance testing
- System testing
- Integration testing
- Unit testing
CAPACITY BOTTLENECKS

What are test levels QE should deal with?

- Acceptance testing
- System testing
- Integration testing
- Unit testing

All of them.
CAPACITY BOTTLENECKS

Some questions that might arise.

“Developers are biased, and will try to make tests pass.”

- Review their tests
- Design is done beforehand

“What about maintenance of the tests, I cannot review every commit in the project.”

- Bug in test is like any other bug.
- Let’s not assume deliberate gaming of the system.
DEVELOPERS NOT QUALITY-DRIVEN
(bad low-level test coverage)

When previous ideas are put in place and QE:

- has voice within the team,
- has overview of unit/functional tests and put them into test plans and strategies,
- defines tests so anybody in the team can develop them,

then the team should be quality-driven just fine.
DISCUSSION
THANK YOU

mhaicman@redhat.com