redhat.

Twice a user, half a developer

Documenting highly technical topics

Mikhail “Misha” Ramendik
June 2018

Introduction: Misha Ramendik

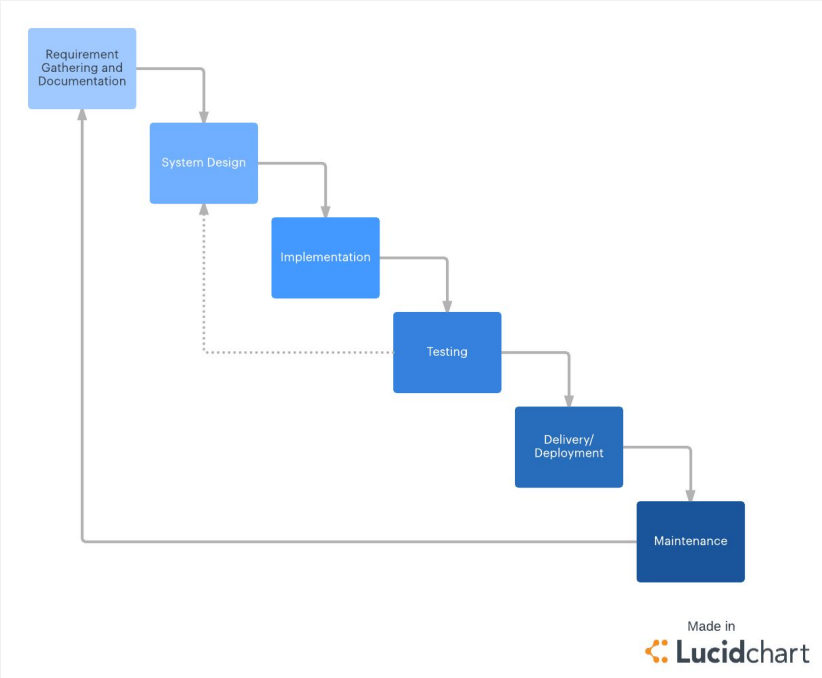
- Senior Technical Writer at Red Hat
- Limerick, Ireland (remote). Originally from Russia.
- Worked for Dell and IBM before; 9 years at IBM
- Technical Writer since 1998
- misha-r@redhat.com
- mr@ramendik.ru

The highly technical topic

- Much of the work technical writers do is for “Joe User” who normally has a UI to use. We have “happy paths”, minimizing the number of screenshots, and other concepts working well there.
- But sometimes, the topic at hand is instead highly technical
- Often this is about installation, administration, other backend work
- The audience is technical too: operations engineers, administrators, even developers.
- The audience can be more technical than the writer, and even for this audience, we must deliver results to a high standard

The older approaches

- Take the full specification, pick relevant parts, edit
  - In Agile there is no specification to take
- Have the SME write notes and/or lecture the writer, edit
  - Context is everything: very easy to miss things



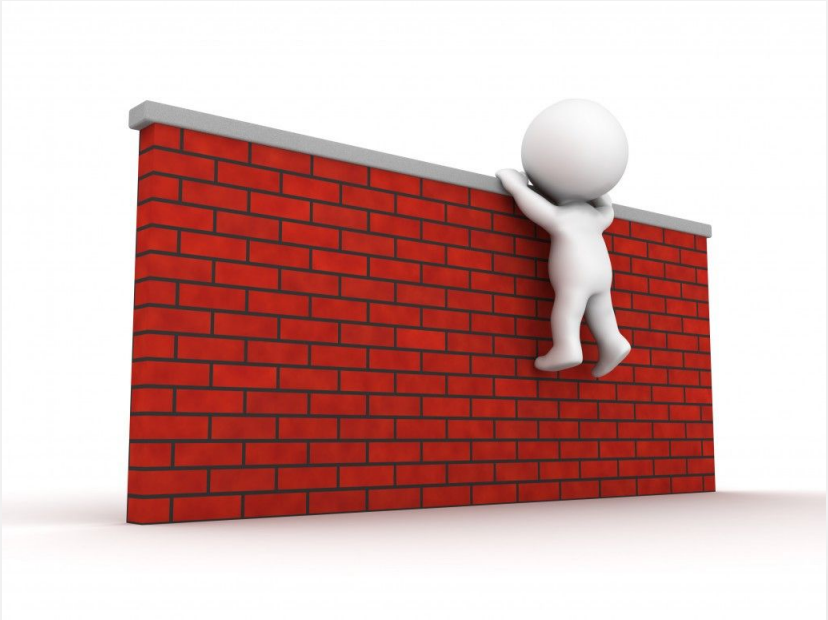
Context is everything: the sides of the wall

- Many modern concepts: DevOps, some parts of Agile, to a degree Open Source - aim to remove the “wall” between the developer and the user
- This should certainly apply to cases when the user is an expert…
- And yet the wall very clearly exists to a degree, and has to exist
- The developer SME and the customer SME operate in different *contexts*, they have different needs and different “default” understanding
    - For example, the developer might say “edit the foo configuration file” without mentioning which of the possible files it is
    - ...or just send an example of the edited file, without mentioning which lines have to be changed and which were there in original
    - Both things happened to me in the last month, with very good and helpful SMEs

So what side are *we* on?

Understanding the customer side: “twice a user”

- Understand what the customer needs to do
- The developer is *not always the best expert* on this
- QA can have more knowledge but we do also need to work with field engineers sometimes
- Hands-on experience can be key, but is not everything
- **We apply our logic to create a model** of what the customer is doing:
  - What they have
  - What they *don’t* have
  - What they want to accomplish
  - What steps can lead there
- Working out that model can take time (and tech research!), but is often worth it
- Need to model different kinds of needs - thus, *twice a user*
- When we finally have the model, we often get sudden strange questions for developers



Working with SMEs in Dev and QA: “half a developer”

- When we understand what a customer needs to do, we see the gaps in the process we initially got from developers
- This can seem like detective work!
- It can be very engaging for us…
- And can also detect critical bugs at lage stage - happened to me about five times over four companies! Even after QA completed their work!
    - QA can work off test scripts from original designers or developers, while we apply our own model to ask our questions
- ...but the questioning can also feel a bit like interrogation as we “probe for inconsistencies”
- To minimize this effect, we need to do our research to avoid asking too much of things that are too banal
    - “A good question contains half the answer”
- The writer needs to be *half a developer*, not enough to actually develop anything - but enough for SMEs to feel “we are on the same side”.

Q & A