Releasing quicker

<matt.chadburn@bbc.co.uk>, Dec 2011

- Why and how?

Context

- One News
- A *single* code base
- ~30 developers
- Several parallel workstreams
- Hundreds of changes a week

- Not literally a single PAL application, but a single ‘website’ will lots of packages

Continuous Integration

- Lots of automated tests (unit, integration...)
- Automated build processes
- Everyone committing to trunk
- Visible system state
- Production environment clones

- Yay. This is all excellent compared to where we were 3 years ago.

BDD

- BDD = *features* to *code* to ***business value***

Friday, 2 December 2011

- Business value is what it's all about. And BDD helps focus the product team on this fact.
- The quicker we can translate ideas in to features, the better.
- Value is only realised when you can measure it's positive impact with your audience
- And we are slow at releasing things. A daily release would be a significant overhead on resources.
- Because we delay releases we build a lot of stuff up front and then measure it's success
    - ... maybe we should be more agile and build in smaller pieces.

Continuous Delivery

- Immediate deployment of editorial priorities
- Allows immediate fix of minor/major bugs
- Removes release *overheads* and *stress*
- Removes developer *punishments*
- Lowers risk of change

Friday, 2 December 2011

- Don't think of CD is about just being able to push to live, it's about removing the manual-ness from the production chain.
- Punishment = minor mistake gets pushed to stage means 48hr release delay. Crazy.
- "Lowers risk of change" because changes just flow through the system to live as business as usual.

Shepherding code to live

- Write some code
- Run tests on SANDBOX
- Commit to INT, repeats tests on INT
- Code automatically deployed to TEST if tests succeed, repeat tests on TEST
- Tester/Developer notified & has a time window to rubber stamp build
- Build gets deployed to STAGE, automatically load tested
- If load testing is successful code deployed to a single LIVE server
- If no errors on that server, code deployed to server farm

- How it might work.
- Perhaps this happens < 24 hours, perhaps within 60 minutes?
- Responsibility on technical leads to make sure this is done with quality in mind.

Blockers

- Manual release process
- Load testing - manual ‘gatekeeping’
- Inability to smoke test releases on subset of users
- Rollbacks / Feature flags
- Production data in non-production environments
- Dependencies

Friday, 2 December 2011

- Release process delay. 48 hours minimum. What value does it add?
- Automate load testing? It can be done, especially just hammering end points with requests.
- Switching things off is simpler than rolling back? http://martinfowler.com/bliki/FeatureToggle.html
- We have a great test environment & usually poor test data
- Project that have lots of upstream dependencies can only move as quickly as their dependencies move.

Risks

- Not enough tests = Broken stuff getting released
- Lazy developers using it as a means to monkey patch live
- Still in a *manual QA* mentality
- We have no *as live* production clones

- We need more rigorous tests/testers, not stricter release processes and checklists.
- Manual QA should be a communal activity done during development, the less of it the better
- Relationship between developers and test is changing
    ... testers (DiT & manual) are there to help the developers test their features, not to test it for them.
- As Live means we have no environment that is a version for version mirror of live.