365 Shades of Grey
Release Planning for Samba

Ira Cooper
SambaXP – 2015
Who am I?

- Software Engineer for 15+ years.
- Systems Programmer.
- Systems Engineer.
- Studied software process and release cycles at:
  - TASC Inc.
  - The MathWorks Inc.
  - Red Hat Inc.
  - **None of these companies endorse this talk.**
Why Develop Software?

- Money?
- Enjoyment?
- Prestige?
- Gun to head?
Why Release Software?

- Commercial
  - Don't! (Works for Google Mail and Search!)
  - For the money! (Works for MathWorks!)

- Open Source
  - ????
  - For the “users”?
  - For the developers?
Open Source – Reasons.

- Help the planet.
- Tell the planet about our wonderful software.
- Hope someone will buy our little start-up for money.
- Buzzword compliant.
- Fairness.
- Freedom.

- Many, many, real reasons…..
When to Release Software?

- When it benefits the people “developing” it.
  - Call these people the stakeholders.

- What about the users?
  - Without the stakeholders, there IS no software.
  - There is nothing to release!

- Obvious, but critical insight.
The Lifecycle of a Release

- Development.
- The typical “Glideslope”.
  - Feature Freeze.
  - Code Freeze.
  - Release.
- Maintenance?! - Yes, it matters!
- End of maintenance.
Development.

- Developers do what they do.
  - Write code.
  - Review code.
  - Make things unstable\(^\text{W} \) better!
  - Run amuck!

- This is where feature development SHOULD be done.
The “Glideslope”.

- Taken from flying.

- The goal is a smooth landing.
Feature Freeze.

- Control hand-off from development to management.
- No more features after this date.
  - Exceptions?
    - I've never been somewhere there ISN'T!
    - Or worked on a project that way.
- Bug fixes go in without questions in this phase.
- Minor improvements MAY be taken.
  - Always be suspicious… There be dragons!
See... I warned you.
Code Freeze.

- No changes without written exceptions.
- Exceptions == Day for Day slip on the issue.
  - Just a bug is NOT enough to get in.
  - “It is only a small change.”
  - “It's almost done!”

- Only critical fixes should be made at this point.
- Changes at this point are VERY dangerous.
  - There be more dragons here!
I told you...
How to Train Your Dragons.

• Don't have any?!  
  – Not very realistic.

• Put strong conditions on waivers.  
  – Any waiver should be tracked!  
  – Is it REALLY a day for day slip?  
  – Discipline is needed here!

• May need to be done by committee?
Glideslope Exit?
Policies You Need.

- What about late security issues?
- Late breaking data corruption issues?
- Late severe regressions?

IMHO:
  - Decide based on what it is.
Release!

- Ask Karolin.

- She'll explain it better than I ever could!
Maintenance?

- Each branch has a set of policies that govern it.

- Nobody says they have to be the same!

- Should they be?

  - Depends… who is maintaining them, and why.
    - Those who drive the maintenance, decide policy!
When to Release?

- Think about why we release…

- To benefit the “stakeholders”.

- So shouldn't release timing benefit those people?

- Some policies I've seen…. 
End of Maintenance

- This is a touchy issue.
- This may be a “shade of grey”.
  - No more “feature” backports.
  - No more “security” backports.
- We need to know who cares!
  - Plus, who pays.
What We Do Today.

- ~10-11 month full cycle.
- 9 months of Development.
- 1-2 months of Feature Freeze.
- 1 week of Code Freeze.

- Maintain 3 releases concurrently.
  - 1 with some back ports.
  - 2 with security back ports.
We've Got Dragons!

By Antonella Nigro (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons
Today's Dragons.

- “I need to complete this feature.”
  - Release slips 2-3 months.
  - Some needed that release sooner!

- No real control of the dragons.
  - They run wild!
Problems?

- Clearly this results in about a release a year.

- No real time in Feature Freeze.

- No time at ALL in Code Freeze for the release.

- These things really hurt quality.
Benefits.

• We know what we do today.

• We actually have done this!

• It has worked for three years.
  – Do not knock this fact.
4-6 Month Cycle.

- Development is between the rest of the cycle.
- ~1-2 months for Feature Freeze. (Beta)
- ~1-2 months for Code Freeze. (RC)
- Hard stop on the time lines.
  - No dragons!
There Be Dragons.

- 4-6 months is a LONG time to not get a feature to the field.
- Some developers may be stuck forking on major features.
  - Look at how aggressively Red Hat backports kernel patches…
  - Red Hat does make our Samba versions available via git on git.samba.org from asn.
  - Probably not what we want for our community.
How to Train Your Dragons 2.

- We need to acknowledge the needs of our stakeholders.

- Exceptions are part of life in this type of cycle.

- Control them.
  - Train your dragons.
  - Yes, we may need a dragon tamer…
    - Or do we do it by committee?
Benefits.

- It is better than today.

- It meets the needs of at least one stakeholder better.

- Can we do better?
Linux Kernel

- Releases are ~6-8 weeks.
- Branches beyond 2-3 back are NOT maintained.
  - Anything older is basically kept up by a distro.
- ~2-4 weeks of Development.
- ~2 weeks of Feature Freeze.
- The rest is Code Freeze.
Problems?

- Fast release pace may confuse people.
- Do releases have a real meaning?
- What is a “stable” release?
- We don't have a “Linus.”
  - Can we do it by committee?
Benefits.

- Fast release pace. Code gets to the field fast!
  - RC's can be as fast as 2 weeks!

- Those willing to maintain decide what to maintain.
  - Not the “main line” developer's problem.
  - Well, it is... but we'll be paid for it.
9-12 Month Release Cycle.

- I won't pretend to like this.
  - Need more field feedback.
  - If we had stronger QA, we might get away with it.
  - Some developers need more frequent feature drops.
    - They'll PAY for it.

- It just doesn't meet our needs, and we know it!
Feature Based Release.

- There's a key set of features that must be done.
- Decide on the features.
- Don't ship until they are done.
Benefits.

- Features are what drive releases.
  - There's always a feature for a release!

- Features always make the release.
Disadvantages.

- What if a feature slips?
- What if a feature never ships?

- What if a feature's developer won't admit it slips?
  - Yes, this happens.

- Feature based release, is something to be wary of.
  - But it can work, at times.
Other Plans?

- I welcome you to come up with plans!
- But understand the constraints.
  - Who is paying for it?
  - Who will work with it?
  - Who are the “stakeholders”?
  - Why does it meet our needs?
- I welcome discussions today, and tomorrow!
  - On samba-technical once we are closer.
  - Please not until we ARE closer.
Remember.

- Most software ships late.
- Many projects never ship at all.
- Figuring out what is going when, is a true art.
  - That's why there be dragons!
- Our goal is to meet our stakeholder's needs!
Questions?

By Antonella Nigro (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons
Thanks for Attending!

By Antonella Nigro (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons