Extreme Containment Measures

Keeping your bug reports under control
Note: je me présente à la slide suivante.
Hi. For those who don't know me I'm J-F, and I'm a gnome-a-holic.
I've been involved in one way or another since roughly 2005, and obviously my first touchpoints with the gnome community were... In this talk I'm going to present my analysis of the big issues we now face in gnome when it comes to bug management.
I have some rather strong opinions, but of course those are just my suggestions, so if you think that...
“Reducing our core apps’ software inventory”

http://jeff.ecchi.ca/blog/2012/10/08/
reducing-our-core-apps-software-inventory/
Prerequisites and assumptions of this talk
But then if you really want to know
Inbox Zero
action-based email
The Original 43 Folders Series
Some of my ways of thinking and working have been influenced by methodologies like Getting Things Done.
Manufacturing is a fascinating thing.
Now, those books are interesting and all, but if you really think about it, their core essence has always been about something else.

Lean and just-in-time manufacturing is a *fascinating* concept. And while I was doing studies in management I started thinking about those things and seeing how when you think about it, everything is always about flow, throughput, adaptation to change and avoiding waste. Stuff has to flow in >------->>>---> and flow out just as fast. As soon as you start accumulating, even...
Inventory is a LIABILITY
If this is the only key thing you should retain from Eliyah Goldratt's book in our context of bug management, it is this: In the manufacturing world this means stuff that you have produced. Now you might think "wait, you're wrong, the product it worth money, so it's an asset"... it's natural to think that way. In our context, it means cognitive weight (GTD), wasted time when searching for existing bug reports.
Well, we have a problem here. We are accumulating excess inventory. LOTS of it. And it's slowing us.

Of course, pretty much every project has this problem, but due to the fact that GNOME is such a large ecosystem of pro

Maybe you're thinking that I'm exaggerating. Well, let me show you the problem we have, in a very
This is the global amount of open bug reports in GNOME since the beginning. I'm missing data between 2001 and 2005.

That's 45 thousand open bug reports we have now. This is NOT sustainable.

And this stuff is so old and messy that it doesn't even represent the real amount of bugs our users experience.
This is the part where there are lots of graphs
So who's responsible for this? Well, everyone to some extent. It's really easy to let things slip and drift over time. Going to show you some examples, those are not meant to point fingers, they're just meant to illustrate how different.
epiphany. It was kept fairly nicely under control, and then... the port to webkit happened and many things changed including the app's branding/positioning.

We can infer that the green bugs were filed by developers right from the start - gradually declining huge cleanup opportunity
empathy. generic example. Fairly under control. That big dive at the end, that was me.
evolution - notice they actually use the NEW status

AMAZING improvement. If I'm not mistaken, a big part of that is thanks to André Klapper. Thanks dude!

still a huge amount of bug reports left to deal with
gnome-shell: oh boy. I have many things to say about the way this project handles its bug reports but
GTK critically needs help. Lots of old bugs that are certainly not relevant anymore. Needs an army of testers.

A prominent contributor once said to me "I am not subscribed to GTK bug mails. I don't look at the bugs".
pitivi. this is how things should be. At least like this, or even better than this.

NEW bugs are more numerous than unconfirmed ones

Kept under control. Less than 150 at all times

This graph is going to come crashing down in the next pitivi release as I'll close a bunch of those bugs.
rhythmbox: mostly in a similar situation to GTK. Needs help. Tons of potential duplicates and ancient bug reports.
What is quality?
Let's talk about quality
What *is* quality? This is an open question. Come on guys, how should we, GNOME, define how we...
“How could we have 3000 [bug reports] while our product is delightful and customers love it & use it every day?”
We can probably agree that the amount of bug reports we have in gnome bugzilla is no strong indication of the quality.

In his article, Joel actually advocates for something pretty radical: he says: "Do not allow more than two weeks for fixing a bug.

Now that's a pretty radical thing to do, and that does not quite apply to our ethos as open source developers.
How do you actually keep stuff under control?
So how do we do it?
And then you're thinking "okay that's easy if you're only dealing with feature requests... what about..."
WARNING!

MAINTAIN THE QUARANTINE
DEADLY FORCE WILL BE USED TO PROTECT THIS AREA
Well, we need to get serious about bug control.
A process of ongoing improvement
And by serious I mean many things, derived from the "kaizen" philosophy in the manufacturing world.
清潔 Seiketsu
整理 Seiri
清掃 Seiso
整頓 Seiton
躾 Shitsuke
The 5 S system: seiri, seiton, seiso, seiketsu, and shitsuke
sanitizing (get rid of stuff), organizing, cleaning, maintaining the daily cleanliness, and discipline in
Investing a bit of effort in cleaning continuously pays tremendously in the long run.
We have 45 thousand bugs. We need to start cleaning now, and we need to keep cleaning in the future.
Inbox Zero is basically just a mindset when it comes to dealing with your email. When you receive an email, you can either open it and respond to it right away, or you can choose to ignore it and wait until you have time to deal with it. We can apply this methodology to how we handle bug reports in GNOME.

We can apply this methodology to how we handle bug reports in GNOME.
Decisions in FLOSS projects:

- “Yes” → CONFIRMED + target to a milestone
  ...or fix immediately.

- “No for now,
  yes if you provide a patch” → Enhancement
  (+ gnome-love/HELPWANTED)

- “No, out of scope. Sorry”
Decision-making ~boils down to these 3 possibilities. NEEDINFO is not listed here, because it is a state of indecision.

Confirming bugs is actually pretty important. And one thing I would strongly advocate for is that we...
Indecision is the mind killer
Indecision is paralysis. Leaving a 7 years old bug report open in an unconfirmed state because “nobody has made a decision” is not only unprofessional, it’s also a waste of resources.

Whenever possible, you want a clear yes or no. If we don’t decide now, we never will. So if a bug is in your “maybe” category, try to close it with a final decision.

Using target milestones is super important and useful too. It allows you to know which bugs were fixed for what version.
Fix it immediately?
And by fix immediately, I mean fix it, and then *close the bug report*, unless the feature is still broken.
"Refactoring" is not a bug
What do I mean here? You got a fix for a bug that you have committed to your master branch but you're a bit unsatisfied with the code's quality.

*cough* gnome shell *cough*

By the way, do not close bugs until the fix has been merged to the main development branch. If they are only committed to the master branch, they might be reintroduced in the next commit.
GNOME is actually pretty infamous for daring to say "no, out of scope".

But saying no without ending up on the frontpage of slashdot is a delicate art.

you can’t implement everything out there due to insufficient manpower or because feature X would...
The key to consistent politeness
Canned replies. Use them. If there is none available, create one and save it as a text file on your computer for reuse.

Example: "Hi and thanks for reporting this bug. Pitivi's core is currently being replaced by GStreamer.

You should particularly have a set of custom canned replies for NEEDINFO with debugging instructions.
Special saved search queries are your friends
Bug searchability is paramount. Renaming bugs saves time.
Notice the little "Edit" hyperlink in the bug title? Use that to rename the bug report at the same time.

And then use a good bug title/summary. Make a complete sentence with exact words. Choosing the right words improves clarity, makes it easier to find the bug.
"Share my desktop item in the menu if Vino is a registered handler in MC5"

"Only show the "Share my desktop" menu item if Vino is actually present and verified as a registered handler in MC5"

"Show where files come from"?

"Provide context – allow users to know the path / location from where files and folders in search results come from"
Here are some examples of good and bad bug titles
"Incoming call's ringtone doesn't play if the sound theme is muted in GNOME's sound preferences"

"Support images with EXIF rotation"

"H.264 stream in an AVI file doesn't play properly with h264parse"

"GMail IMAPx message count is incorrect on emptied folders"

"Login screen width is resized differently after an incorrect password auth attempt"
Here are some other examples of good, easily searchable titles.

First item: people searching that for words such as "sound" or "ringtone" alongside "call" will be able to find that bug re... For the "GMail IMAPx message count" thing, people searching for gmail, imap, mail or message count will be able to...
Use these keywords extensively:
- gnome-love (need patch - easy stuff)
- HELPWANTED (need patch - hard stuff)
- usability
- ui-review ("controversial, need the design team's input")
- other stuff in bugzilla.gnome.org/describekeywords.cgi
Go nuts. But don't go insane.

- Divide and conquer
- Regularity over quantity

http://cgit.freedesktop.org/gstreamer/
gstreamer/tree/scripts/five-bugs-a-day.pl
Divide and conquer: don't do it alone, do it in teamwork. Maybe bug triage days. Use voice (hangouts).

Tim's script to force him to revisit ten bugs per day
In summary:
- Prioritize clarity and searchability
- Move fast and avoid staying in limbo
- Be firm but extremely polite
- Do regular cleanup sessions
To recap:
- NEEDINFO everything you’re unsure about. In six months, with a special search query, close any remaining NEEDINFO.
- If you are a developer or know the project’s vision, show some courage and learn to say no or “patch or it won’t happen.”
- Do regular cleanup sessions every few months or when undergoing a significant technological or ideological change.
- Search and verify all bugs older than 400 days, 800 days. Chances are they are fixed, obsolete, or out of scope.
That's all, folks!
Let's discuss.

http://jeff.ecchi.ca
That's it. Questions or comments? This topic is quite large and I'm quite sure I must have forgotten...