Ingredients for a successful project

Tim Starling
Nairobi 2025

About me

- Studied physics
- Started editing Wikipedia in October 2002
- First MediaWiki commit May 2003
- WMF employee since 2006
- CommTech since 2023

About this talk

- A look back at some extensions and other projects
- Some nostalgia
- Some tech-speak

What is success?

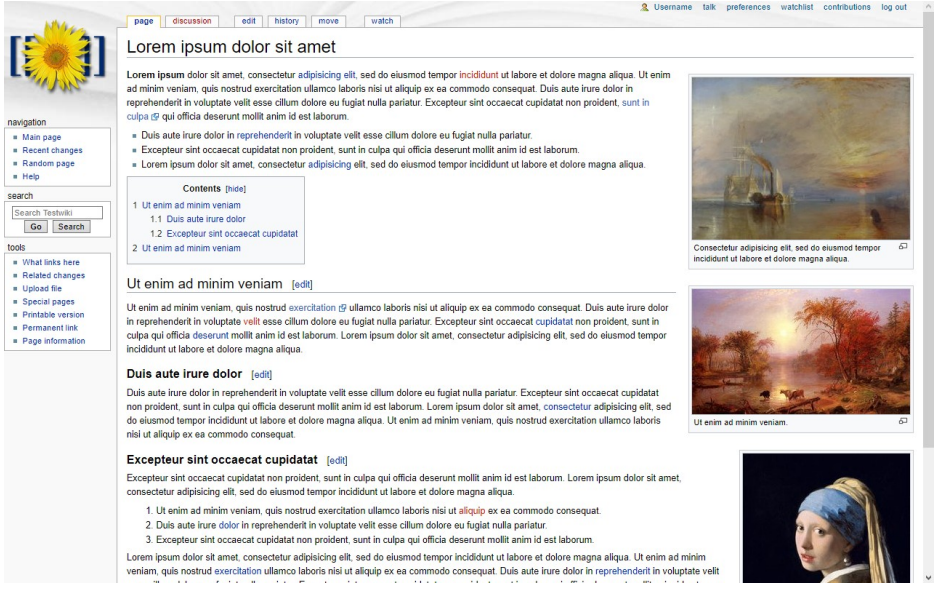
- Provided returns exceeding investment
- Met its own goals
- Advanced the mission

Example: Cite extension

- December 2004
- Ævar Arnfjörð Bjarmason
- 340 lines of code
- A simple footnote feature which transformed Wikipedia

Example: MonoBook skin

- Default skin from 2004 to 2010
- Gabriel Wicke
- Ported from TAL to plain PHP by Brooke Vibber
- Superseded by Vector but was not a failure
- Met and exceeded its own modest goals



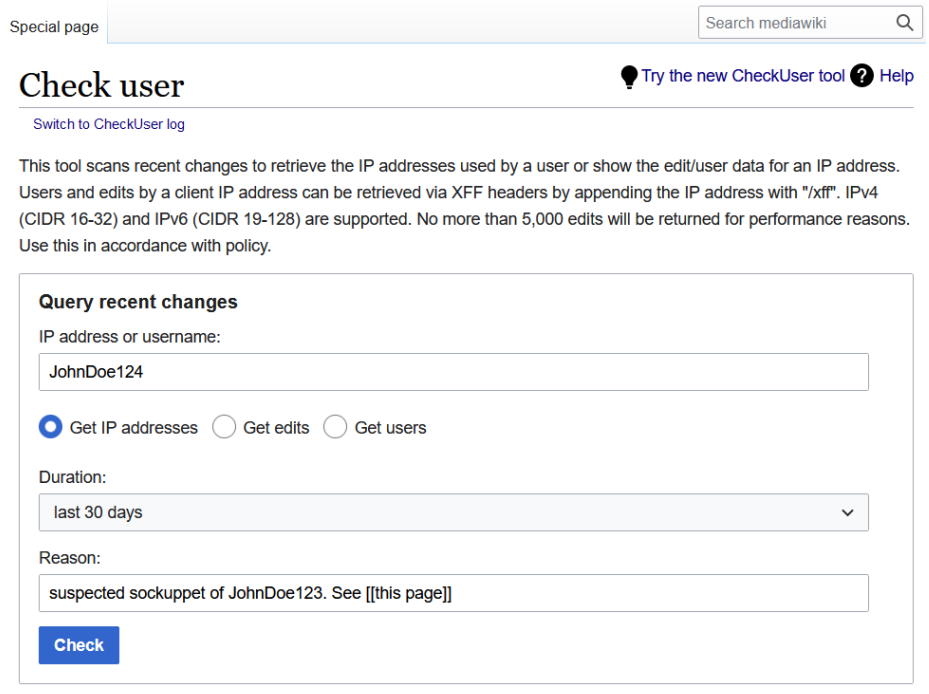
What ingredients make a successful project?

What ingredients make a successful project?

- An idea that satisfies a clear need

Example: CheckUser

- August 2005 initial commit by Tim Starling: **153 lines**
- April 2007: CheckUser 2.0 by Aaron Schulz
- Humble beginnings but the clear need for such a thing drew maintainers to the project
- Now **40,000 lines**



Counter-example: Dynamic Dates

- July 2003
- **98 lines** by Tim
- Show content-area dates according to user preferences
- Oiled squeaky wheel
- Gave very new developer some experience (my second commit)
- Undeployed with costly content migration

```
# Do replacements
# TODO: month capitalisation?
if ( $datePreference == 0 ) {
    # no preference
    $text = preg_replace( $rxDMY, '[[$2 $1|$1 $2]] [[$3]]', $text);
    $text = preg_replace( $rxYDM, '[[$1]] [[$4 $3]]', $text);
    $text = preg_replace( $rxMDY, '[[$1 $2]], [[$3]]', $text);
    $text = preg_replace( $rxYMD, '[[$1]] [[$3 $4]]', $text);
    $text = preg_replace ( $rxDM, '[[$2 $1|$1 $2]]', $text);
} else if ( $datePreference == 1 ) {
    # MDY preferred
    $text = preg_replace( $rxDMY, '[[$2 $1]], [[$3]]', $text);
    $text = preg_replace( $rxYDM, '[[$4 $3]], [[$1]]', $text);
    $text = preg_replace( $rxMDY, '[[$1 $2]], [[$3]]', $text);
    $text = preg_replace( $rxYMD, '[[$3 $4]], [[$1]]', $text);
    $text = preg_replace ( $rxDM, '[[$2 $1]]', $text);
} else if ( $datePreference == 2 ) {
    # DMY preferred
    $text = preg_replace( $rxDMY, '[[$2 $1|$1 $2]] [[$3]]', $text);
    $text = preg_replace( $rxYDM, '[[$4 $3|$3 $4]] [[$1]]', $text);
    $text = preg_replace( $rxMDY, '[[$1 $2|$2 $1]] [[$3]]', $text);
    $text = preg_replace( $rxYMD, '[[$3 $4|$4 $3]] [[$1]]', $text);
    $text = preg_replace ( $rxDM, '[[$2 $1|$1 $2]]', $text);
    $text = preg_replace ( $rxMD, '[[$1 $2|$2 $1]]', $text);
} else if ( $datePreference == 3 ) {
    # YMD preferred
    $text = preg_replace( $rxDMY, '[[$3]] [[$2 $1]]', $text);
    $text = preg_replace( $rxYDM, '[[$1]] [[$4 $3]]', $text);
    $text = preg_replace( $rxMDY, '[[$3]] [[$1 $2]]', $text);
    $text = preg_replace( $rxYMD, '[[$1]] [[$3 $4]]', $text);
    $text = preg_replace ( $rxDM, '[[$2 $1]]', $text);
}
return $text;
```

Dynamic dates rationale

- Following quotes excerpted from a 23,000 word discussion.
- **Wikipedia talk:Manual of Style/Dates and numbers** archives 1-6 are mostly about whether the day or the month should go first in a date.

Dynamic dates rationale

Some UK-English writers have, in the last months, managed to sneak in not through a community process but through backdoor discussions, the privilege to use their date style on "their pages". It is one of Wikipedia's biggest virtues that "pages are not owned by anyone." Yet Mav proposes exactly that. Editors will first have to

1) *Check: article has dates in it? Yes: Use that style - No: Proceed*
2) *Check: article is about a British subject? Yes: Use British style - No: Proceed*
3) *Use "American" style*

Any newbie who doesn't get that complicated process and uses what he is used to will find a long, insulting essay on his personal talk page. Not consistently, of course: On some pages the original...

— Erik Möller (Eloquence) June 2003

Dynamic dates rationale

Users of American English and British English, of American dating and international dating have worked perfectly happily together. Everyone has been able to understand the other's dating system. Nobody has been confused.

From Alaska to New South Wales, Scotland to South Africa and further afield, everyone has been able to follow the arrangement, nobody has been treated like second class contributors, ordered to abandon their own form of English and write a form they never use. Good quality articles (and the occasional not so good quality article) have been written.

***No-one had a problem** until Eloquence set himself up as the **Wikipedia Language Policeman**, there to force his form of dating on the English speaking world, even though the majority of the English speaking world does not use his choice of language.*

*That isn't consensus and it sure as hell isn't democracy. It is simply a regrettable and avoidable form of **linguistic apartheid**.*

— Jtdirl, June 2003

Dynamic dates rationale

- Nobody:
- Young Tim: let's convert dates when the page is viewed, then you can write and see whatever format you want.

Dynamic dates lessons

- Think very carefully before you add a wikitext feature.
- Backwards compatibility constraints are severe
- The archive will be there forever
- Migration is expensive
- (Did we learn that? How about Graph?)

What ingredients make a successful project?

- An idea that satisfies a clear need
- Right scale and ambition

Example: AbuseFilter

- June 2006 by Andrew Garrett
- Initially 1100 lines
- Detects vandalism with written rules
- Relatively complex expression syntax parser
- Ambitious for the time, but achievable
- Now 33,000 lines

Counter-example: Flow

- Work started 2013
- Installed 2014 with ~**20,000** lines of code
- Deprecated 2016 "due to widespread criticism of the design and functionality"

Flow ambitions

- Not just a discussion system: a system for workflow automation on WMF wikis. A replacement for templates.

  “In many cases local wikis use templates to encourage workflow within them. The goal for Flow's workflow models is to be dynamic enough to be managed by local wiki administrators to cover use cases currently handled by workflow suggestions inside templates. In other words, Flow will implement a whole bunch of Lego pieces, and the individual communities will stick them together into the various workflows they need.”

  — Erik Bernhardson

Flow ambitions

- A cross-wiki micro-blogging platform

  “Flow is cross-wiki. Eventually a discussion can take place across wikis and appear on pages on different wikis.”

  — Erik Bernhardson

Flow ambitions

- Blank slate design: visual editing, no wikitext
- Stores Parsoid HTML
  - Despite Parsoid maintainers saying that Parsoid HTML is undocumented and unstable
  - Despite abuse control requiring wikitext version of every comment
  - Temporary issues will be overcome with time... right?

Flow lessons

- Remember where you are and who you work for
  - We can't scale up to build out big ideas like a commercial venture
- Personal enthusiasm is essential but blinds us to negative feedback
  - Do our users actually want this?
- Unmet ambitions can undermine a project

What really killed Flow?

- Why not just add wikitext editing to Flow?
- Let's come back to this after we talk about...

What ingredients make a successful project?

- An idea that satisfies a clear need
- Right scale and ambition
- Minimal complexity

Minimal complexity

- Write the simplest code that can work (no simpler)
- If a concept needs explaining, try to get rid of it
- Coin jargon sparingly

Introducing an abstraction

- How many callers would benefit?
  - One? YAGNI. Just get the job done.
  - Two? DRY, but significant differences may still warrant separate implementation.
  - Three? OK, definitely time to do it.
  - ...
  - Eight? Oops, now we have technical debt.

Pushing back on features

- Product manager: I think we should add *Feature X*

Possible responses:

- That would add complexity which may threaten the long-term viability of the project.
- I don't know how to do that within the current architecture.
- That would be very complicated. Maybe we could do *Y* instead?

Counter-example: FlaggedRevs

Warning: Flagged Revisions is very clunky, complex and not recommended for production use, despite the "stable" tag. This extension has not been deployed (newly installed) to any Wikimedia wiki since 2014. See Code stewardship review: FlaggedRevs.

— various MediaWiki.org editors, derived from a "health warning" by James Forrester

FlaggedRevs complexity

FlaggedRevs supports multiple dimensions each can have several levels, and multiple "tiers" in each dimension. Meaning I can lock a page to show the stable version to users if reviewers say "accuracy" of the revision is at least level 3 (at least level 3, is called "tier3" or "pristine") and its "depth" is at least 2 (and "tone" is at least level 1 which translates to tier2 or quality and tier1 or checked respectively) and have different tiers in another page.

This four dimensional chess is not enabled in production, only one wiki has more than one dimension (Hebrew Wikisource) and only a couple of wikis use more than one tier (Finnish Wikipedia, English Wikibooks, etc.) and even in those wikis, it's not widely used as it's inherently complex.

— Amir Sarabadani

Wikipedia quality background

- October 2005: Siegenthaler incident
- December 2005: BLP policy
- August 2006: Wikimania discussions about Wikipedia 1.0, quality architecture.
- Germans super keen
- March 2007: Contractors hired

Aaron Schulz

- Wikipedia editor since 2005
- Hired 2007 to work on FlaggedRevs
- Rigorous, intelligent
- Your best friend if you're stuck down a rabbit hole
- No fear of complexity
  - (opposite of Amir S.)

FlaggedRevs development

- November 2007: 2.5 kloc
- May 2008: Deployed to dewiki
- December 2012: Deployed to enwiki
  - Community demanded additional features before allowing this
- Now: 14 kloc

FlaggedRevs lessons

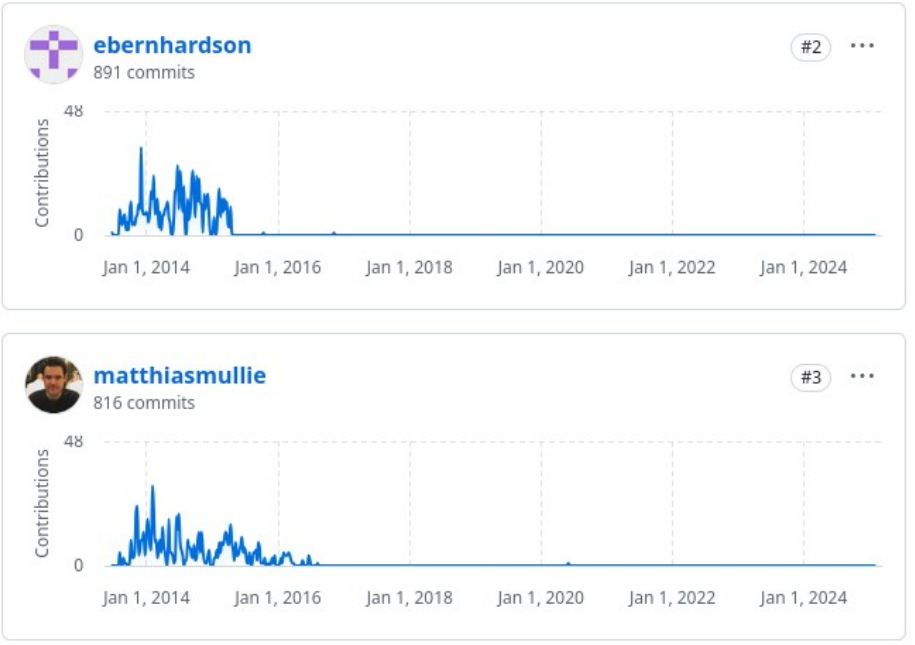
- Code review should include review of architecture, requirements
- Decisions to be made by Aaron vs. Amir cage match

What really killed Flow?

- Why not just add wikitext editing to Flow?
  - Flow code is complex and daunting
  - Twice the size of LiquidThreads
  - Abstractions support non-existent use cases

Flow human factors

- Terry and Brandon left
- Nobody advocating for continuation
- Management reassigned the two main developers



What ingredients make a successful project?

- An idea that satisfies a clear need
- Right scale and ambition
- Minimal complexity
- Easy and conventional operations

Example: CategoryTree

- July 2006, 650 lines by Daniel Kinzler
- PHP, JS
- Write once, just works
- Rarely needs maintenance

---

r15834 | daniel | 2006-07-27 03:12:30 +1000 (Thu, 27 Jul 2006)

Adding Ajax based CategoryTree extension.

Pending: internationalization

(weeee... first commit to the MW repository... thanks Tim!)

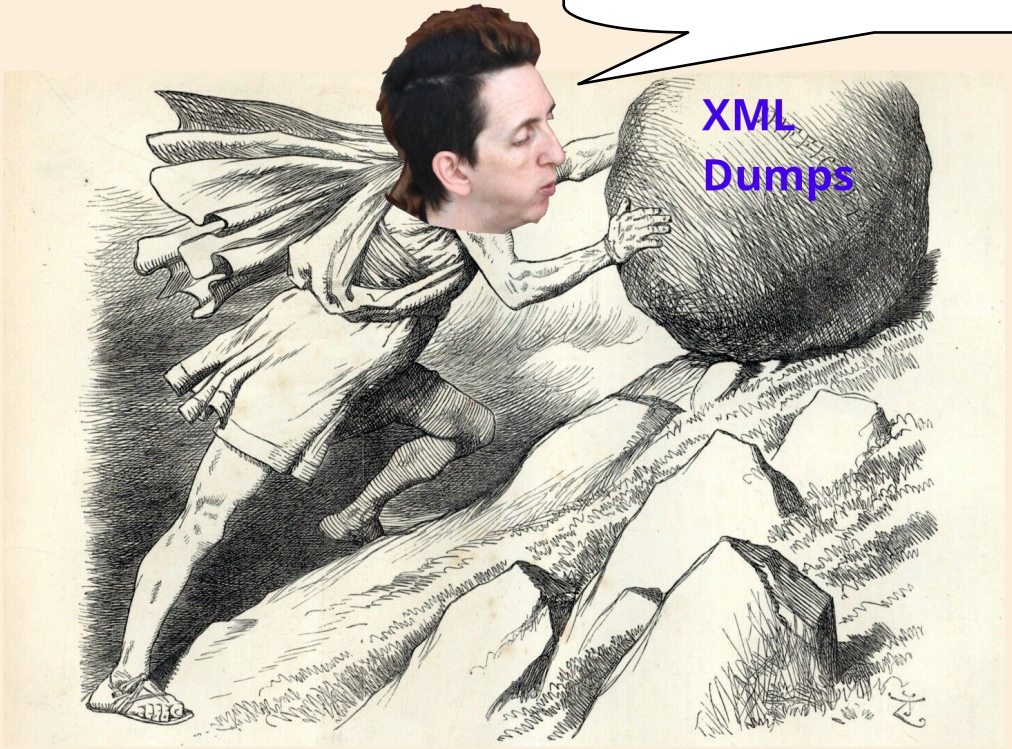
Counter-example: DumpHTML

- March 2005, initially 180 lines by Tim
- Eventually grew to ~2000 lines
- Parse all Wikipedia articles and make HTML archives
- WiderNet eGranary — charity burning copies of the internet for children in Africa — contacted me and got me interested

Counter-example: DumpHTML

- Really cool those 2 or 3 times it worked
- Took months to complete a run
- Like MW job queue but the jobs and workers were fragile
- Required constant monitoring and tweaks
- Probably would have needed 1 FTE forever

I guess this is my life now



DumpHTML end of life

- I stopped working on it in 2008
- Kiwix tried to keep it going until 2013

Beware, cowboy!

DumpHTML required a lot of work and is permanently broken since August 2008. It stopped working shortly after it was split from core in 2008; complicated in 2009; worsened by ResourceLoader in 2010 and then in 2011 and later.

The only alive human known to have managed using dumpHTML with success is Kelson,

DumpHTML lessons

- Is your project so amazing that you want to spend the rest of your life on it?
- If no, make it so that it just keeps working by itself, nobody wants to push that rock for you.
- No, not even ops.

Conclusions

- Ask users what they want
- Try to write code so that it won't need a health warning
- Be boring and conventional

What ingredients make a successful project?

- An idea that satisfies a clear need
- Right scale and ambition
- Minimal complexity
- Easy and conventional operations
- **What ingredients would you add?**

Image credits

- Chapters meetup 2009 Aevar by **Polimerek.** CC BY-SA 4.0
- MonoBook screenshot by **Bartosz Dziewoński**. GPL v2.
- Aaron Schulz March 2013 by **Myleen Hollero**. CC BY-SA 3.0.
- Salisbury Sisyphus by **John Tenniel**. The Punch, April 9, 1887. Public domain.
- SJ and Ariel Glenn at Wikimania 2010 by **Sage Ross**. CC BY 3.0.