How to Run an Agile Project in Government

by Robert L. Read, PhD, Chris Cairns, and Jesse Taggert

http://18f.github.io/consulting/

In a Nutshell...

This presentation is an attempt to distill **Agile in Government** down to a single 40-minute presentation:

1. What Agile is
2. Why it matters
3. What is unique in the Federal context
4. The mechanics of Agile Program Management via Burnup charts and Agile project management
5. The spirit of Agile Program Management in 16 dictums (if time allows)

(18F Consulting provides deeper coaching on all of this.)

What is Agile? In a sentence:

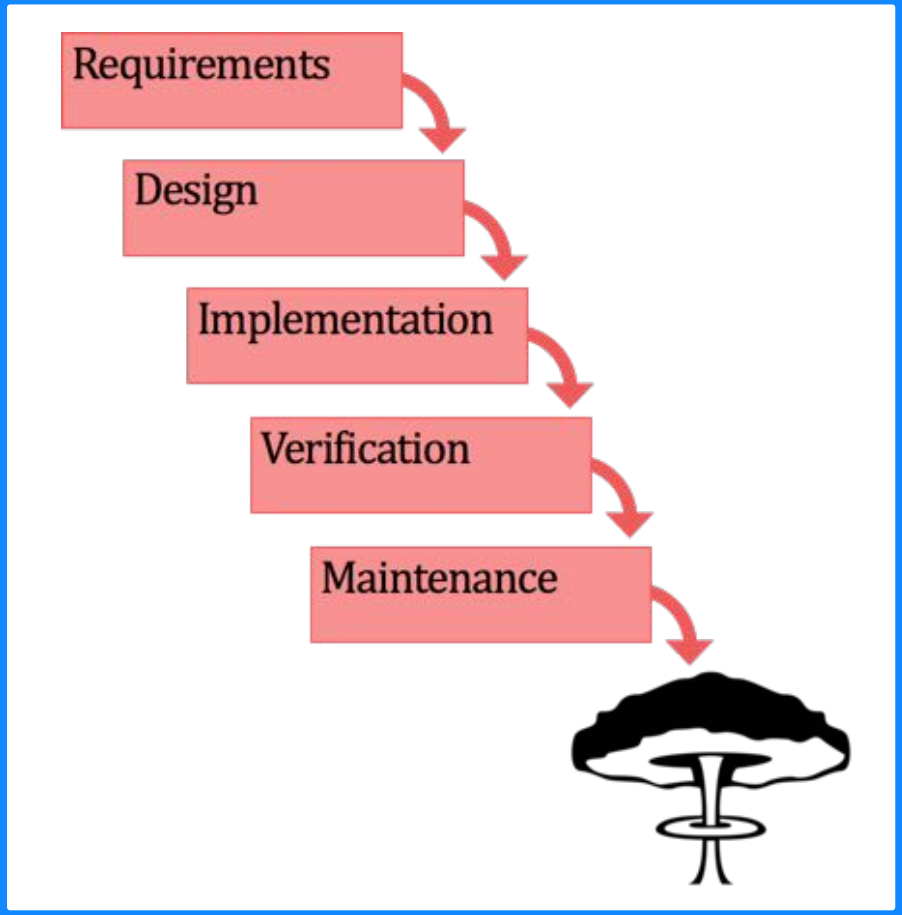
"...value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more."
-- The Agile Manifesto, http://agilemanifesto.org

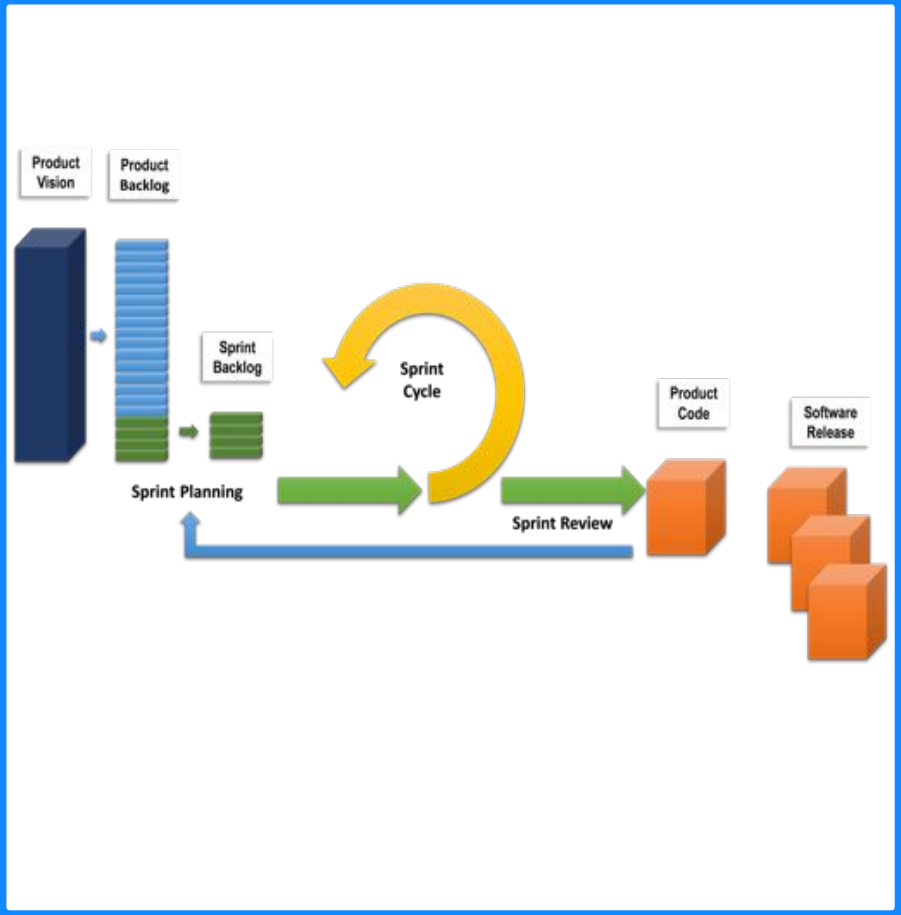
Q: Why Agile Software Development?
A: Risk Mitigation

Waterfall Development



vs

Agile Development



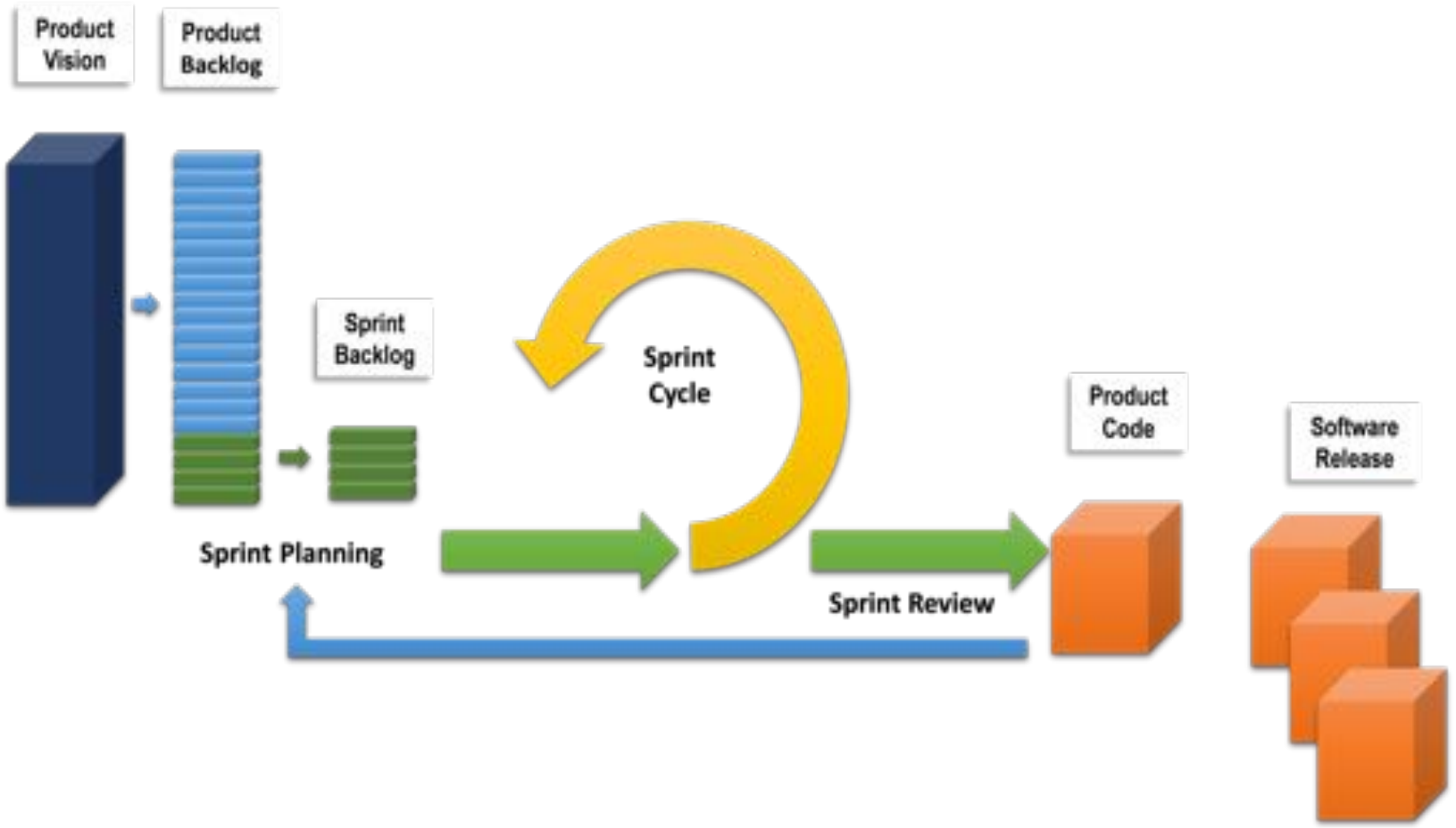
Mega-launches lead to mushroom clouds, and failing often and early is better than failing catastrophically and wasting taxpayer money

Image sources: Mushroom Cloud (Gilad Fried); Agile Development (VersionOne)

Risks of "traditional" methods (e.g., waterfall).

1. "Mega launches" lead to mushroom clouds of failure.
2. Hard to respond to changing needs and requirements.
3. Unclear project status and unknown true progress.
4. Technical and integration problems are discovered late in the project
5. Surprising needs or problems not known until the end.
6. Customer is only involved at the end of the process during "launch."

The Agile way.



Benefits of Agile development.

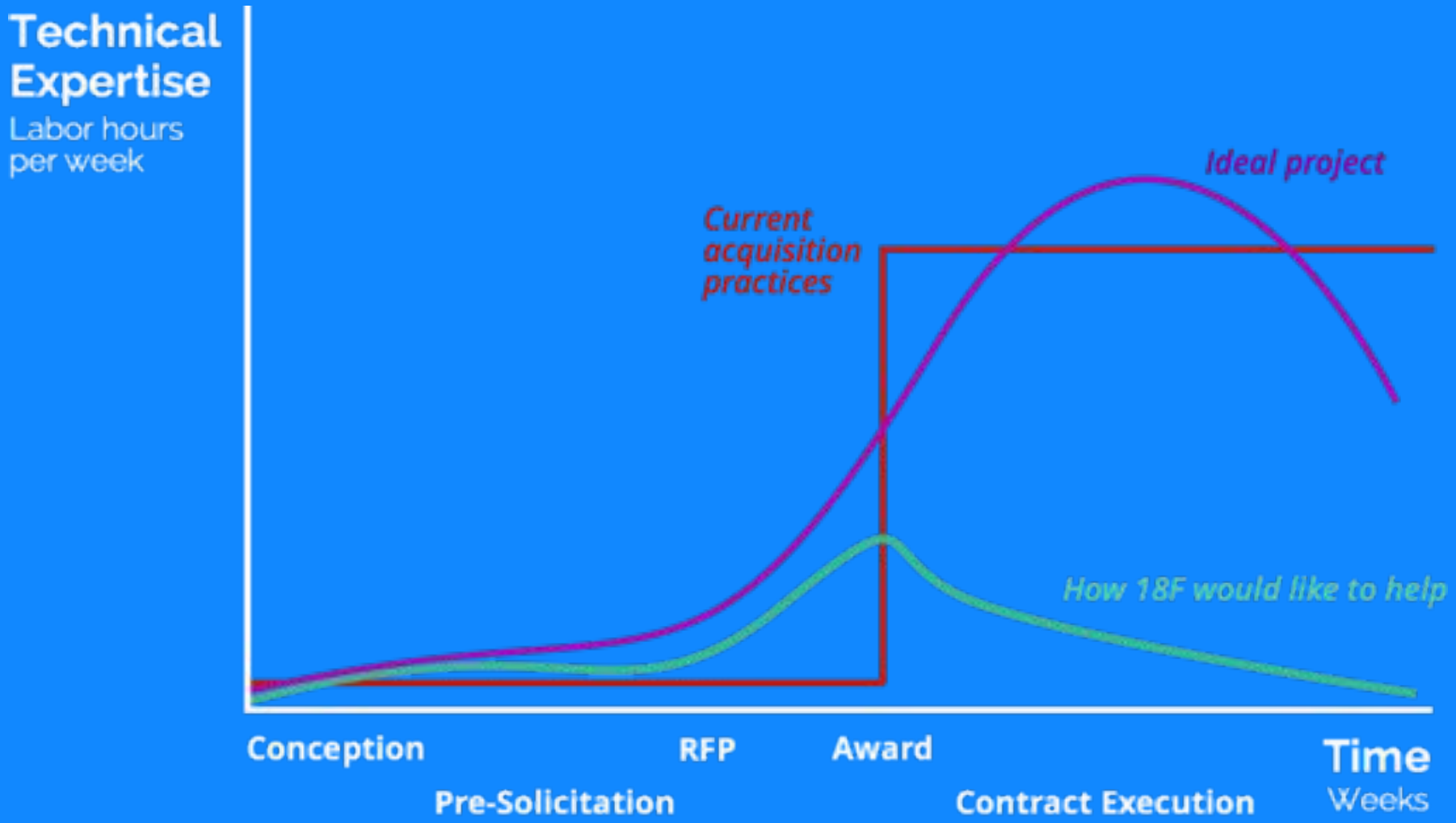
1. With continuous iteration, small failures are quickly course corrected. This leads to faster learning and more **successful outcomes.**
2. User research and usability testing happen throughout the project so the product is **tightly aligned with user needs.**
3. Agile demands continuous integration of all systems, **removing risk of technical integration failure.**
4. You have continuous, accurate **insight into actual project progress** and contractor performance.

Why is Agile harder in Government than in private industry?

1. The outsourcing of technical talent,
2. Habit and tradition,
3. Fear, Uncertainty and Doubt, and, to be fair,
4. the Government's legal insistence on Fairness.

I'll tell you how to address each in turn.

The Acquisition Technology Outsourced Expertise Cliff



How does the Technical Cliff hurt?

- Nobody to help you write the Solicitation…
- No technical Fed to help you evaluate the Solicitation…
- When the award is made, you are overwhelmed and outnumbered.
- Contractors have two bosses.
- And let's face it, those in privileged positions (technologists) sometimes bully those with less knowledge (think Voldemort vs. the muggles.) [The Inmates are Running the Asylum.]

How does Habit and Tradition make this harder?

- Habit of “Big Design Up-Front” (BDUF)
- Over-interpretation of the Paperwork Reduction Act.
- Firm Fixed Price is common contract type and poor choice for Agile. [It’s like: “A weekend in Paris or nothing”.]
- Culture of accepting years and months as acceptable iteration times.
- No culture of prototyping or experimentation.

Fear, Uncertainty, and Doubt

It takes a lot of courage to:

- Take responsibility for success in the next 3 months instead of 2 years from now.
- Face bad news now.
- Be responsible for guiding a team to success when you are inexperienced.
- Give up the security of a complete plan.
- Be prepared to join a team and lose the ability to point fingers.

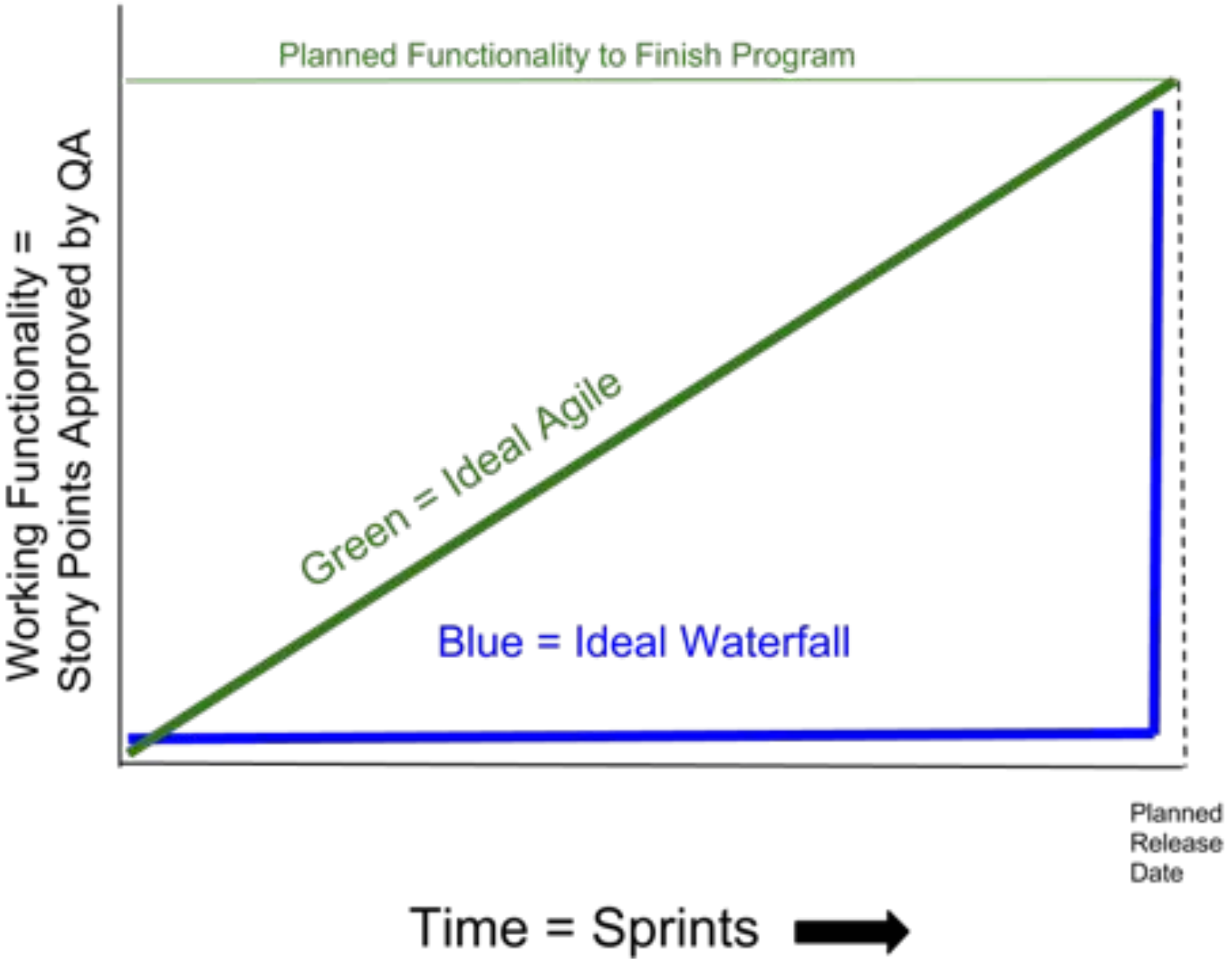
Mechanics: Stories, Estimation, and Burn-Up Charts

Let’s start with the Burn-up chart.

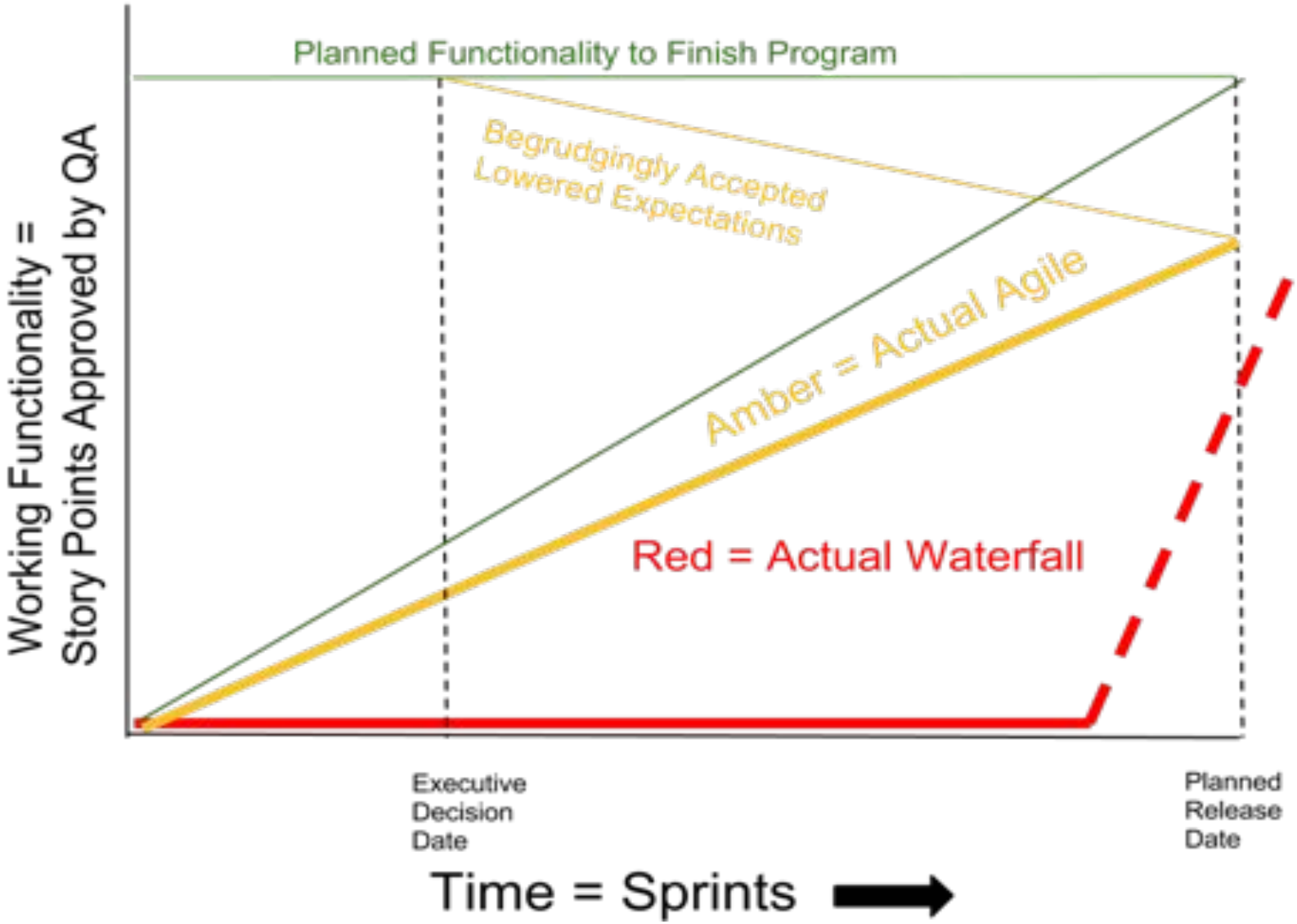
Then we’ll talk about stories and estimation and claiming the stories.

All of this is the same for a non-Federal environment, but even more important in big Federal projects due to contractor relationships.

Comparison of Idea Methods



Comparison of Actual Methods



How do you really manage this way?

The Estimated User Story makes this quantitative graph possible.

Every Story is a small, independently deliverable piece of user functionality.

Stories are estimated in “Story Points” or any other abstract, but consistent measure of complexity, not time (Such as Frosted Birkners.)

A Story is “a promise to have a conversation.”

Stories are the unit of work and means of communicating between developers, designers and the product owners.

They are not requirement documents, and you shouldn’t try to make them so.

Run an Agile project is more demanding than “fire and forget.”

Example Story

"As a Taxpayer visiting the site, I want to be able to understand how much of my tax bill goes to servicing the National Debt."

Stories are short, testable, and estimatable. If a story is too big to be accomplished in one two-week sprint, it should be broken down into smaller stories. A project should have a backlog of 10 to 100 prioritized stories. If you have more than that, you have multiple projects.

Stephen Warren's Principle...

(Paraphrasing):

"No Project Longer than 6 months."

--- Stephen Warren, CIO, Veteran Affairs

https://www.youtube.com/watch?v=6fwZw6UBXQs

Mark Schwartz's Good Technical Practices (CIO USCIS) lead to "Always Be Releasable" - if stories are Independent!

- Modularizing the contract into discrete independent and independently testable components.
- Insisting on repeatable deployment scripting to minimize operational lock-in.
- Insisting on zero-defect iterative delivery and continuous integration, particularly of security scanning, to mitigate the risk of unpleasant surprises late in the schedule.
- Encouraging Test-Driven Development (TDD) in order to mitigate vendor lock-in by delivering executable tests which validate and clarify both internal and external behavior in a durable way throughout the entire contract lifetime.
- Emphasizing Application Programming Interfaces as a mechanism to provide risk-mitigating modularity to the entire contract.

The Team claims Story Points ONLY when the story is Shippable.

This prevents being 90% done for half the time.

Backlog of Independent Stories Necessary for Prioritization

Always Be Releasable, however thinly, so that…

Unpleasant surprises are discovered as early as possible.

Always work in order of your true priorities so that…

Descoping the project is as painless as possible.

Each sprint you are delivering the best value that can be delivered at that point in time.

Writing stories is a shared responsibility...

Because they must be agreed upon and understood by both the Product Owner and the Developers.

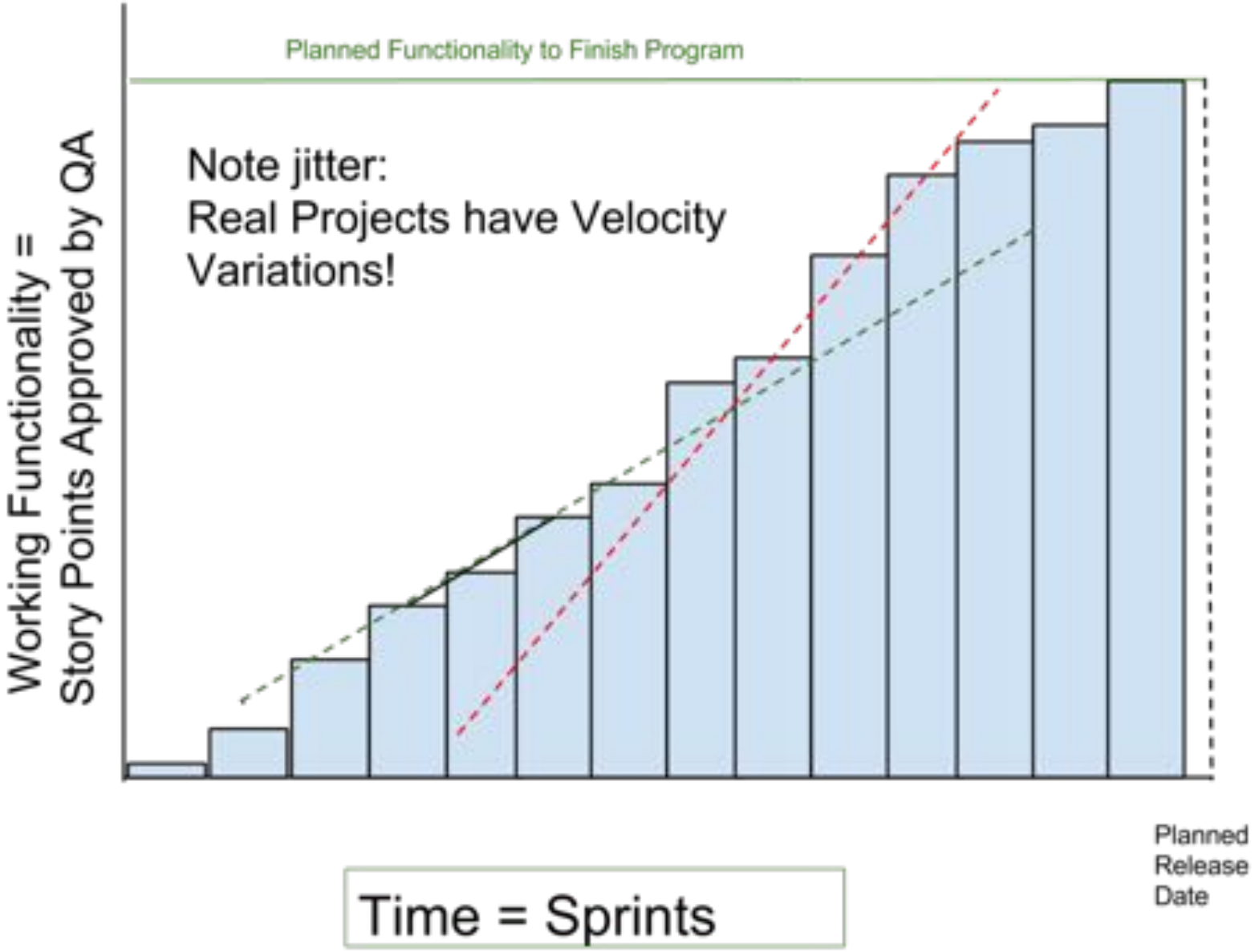
In government, these roles are often further split between Program Manager and Vendor.

It is Developer's job to figure out how to make stories independent.

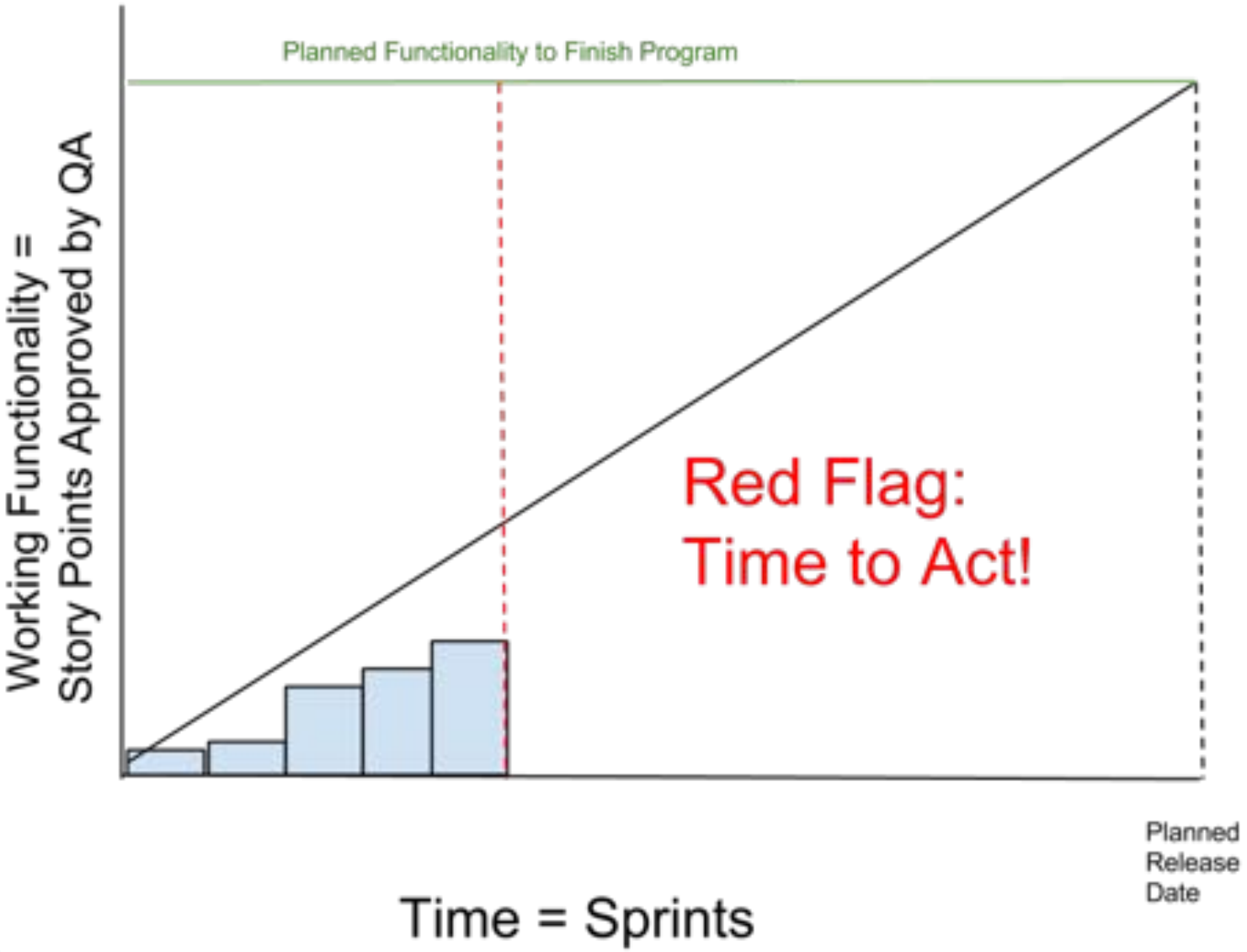
Responsibilities

- Stories are a shared responsibility.
- Product Owner has absolute control of priorities.
- Engineers have absolute control of estimates.
- The interplay is fruitful. This conversation often leads to cheaper approaches or even better solutions than originally imagined.
- “Well, if Blue costs 13 story points and Red only 3, I guess I will prioritize Red!”

An Ideal Agile Project



A Failing Agile Project: Apparent Early



Things go wrong…
Beginners have problems…

- Executive buy-in sometimes lacking.
- Estimation takes courage and practice.
- Always being releasable takes engineering effort.
- Making stories independent takes engineering effort.
- Automated testing is easy to overdo and easy to do wrong.
- Multiple teams can be hard to coordinate.

Guidelines, rather than rules...

- Kanban vs. Scrum vs. XP --- inessential
- Breaking Stories into Tasks --- doesn't matter
- Burndown or Burnup --- whatever
- Retrospectives, Sprint Planning, Daily Standups --- negotiable
- The rules and discipline are there to help you, not to get in the way!

Plan on it taking a while...and start now.

Any organization usually takes about a year to 18 months to learn how to be Agile.

Success brings buyin.

Be Agile: start now, and start small. Run one small project with Agile and learn from it.

What does Agile give you?

Not magic, or a guarantee, or everything you want.

But…

Insight into the unvarnished reality of where you stand and something to do about it.

How to do Program Management better in a mere 16 Agile-inspired steps:

1. First, understand that Waterfall gives a false sense of security that leads to catastrophe.
2. Get buy-in for progress this month rather than a plan for next year.
3. Beg, borrow or steal technical advice as early as you can get it in a project. Use 18F Consulting, non-Fed consultants, or that geek down the street as early as you can in the project.

[You can't run an IT project without IT knowledge.]

If you are already technical, forgive me condescending...

4. Devote 20% of every week to learning.
    a. It will be the hardest and best thing you ever do for your career.
    b. Learn to use GitHub.
    c. Learn to code in Javascript.
    d. Learn to debug javascript in a browser.
    e. Attend a hackathon, even if only to observe.
    f. Understand that a little goes a long way, don't be discouraged.
5. Demand vendors explain it to you so that you understand. If you can't understand, find someone who can.

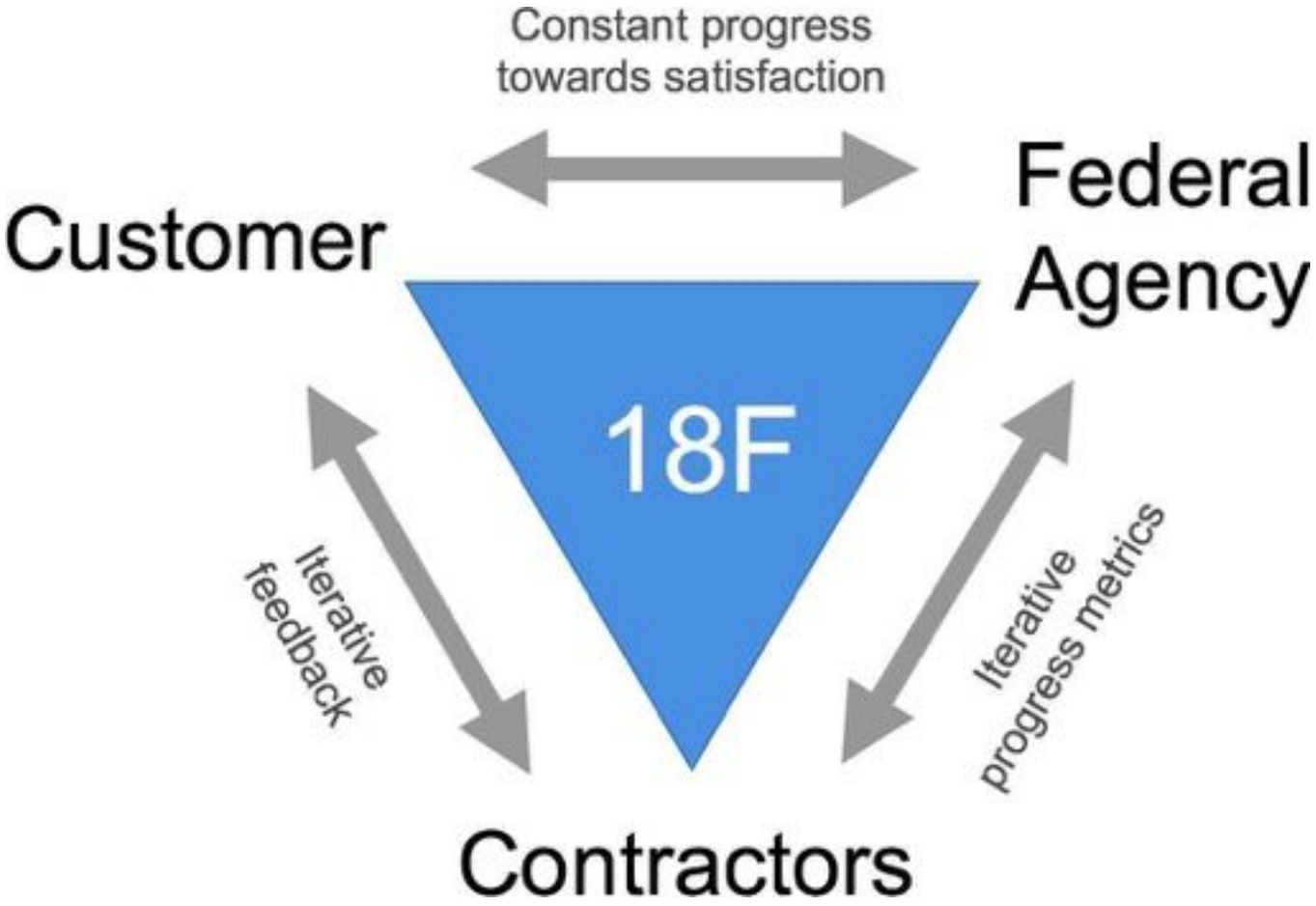
Break Habits …

6. Be brave enough to begin without a complete plan.
7. Demand working prototypes and proof-of-concept experiments as an essential part of every project.
8. Involve the customer from the VERY START and don’t let the PRA discourage you.
9. Anything promised more than a month away doesn’t count. Demand progress at least every month. Make no decisions based on vapor.

Shatter traditions...

10. There is only one team, and you are on it, and the customer is on it. The team fails or succeeds together. The vendor cannot let you down, and vendor has no right to blame you.
11. To lead, you don't have be the smartest person in the room—but you may have to be the best communicator.

18F Communication Triangle
(But imagine a picture of you in the center!)



Reuse the work of others...

12. An army of Open Source programmers and Civic Hackers, and

(see “How to Use More Open Source” https://18f.gsa.gov/2014/11/26/how-to-use-more-open-source/)

Lean on your friends...

13. OMB on contracting: http://www.whitehouse.gov/sites/default/files/omb/procurement/guidance/modular-approaches-for-information-technology.pdf
14. White House on TechFAR: https://github.com/WhiteHouse/playbook/blob/gh-pages/_includes/techfar-online.md
15. USDS playbook: https://playbook.cio.gov
16. 18F Consulting: https://18f.github.io/consulting/

18F Consulting

http://18f.github.io/consulting/

Your trusted user-centered technical advisors.

robert.read@gsa.gov | 202-809-1941

christopher.cairns@gsa.gov | 267-228-1390

GSA, 18F Headquarters, 1800 F Street, Fourth Floor “Infill Area”