

Agile Software Development Techniques for Small Scale Research Projects

“how to not go down the rabbit hole”

Henriette Koning

Senior Manager Software Delivery

But first....

- Henriette Koning (me)
- We will talk about
 - Success through process & approach
 - Focused on IT/software projects
 - Right sized
- But not
 - Dev technology or tools
 - Architecture
 - People / Skills

Why some process is a good thing

- Saves time
- Saves frustration
- Avoids panic
- Creates clarity
- MVP
 - Minimum Viable Product
 - Minimum Viable Process



AND A LONG TALE

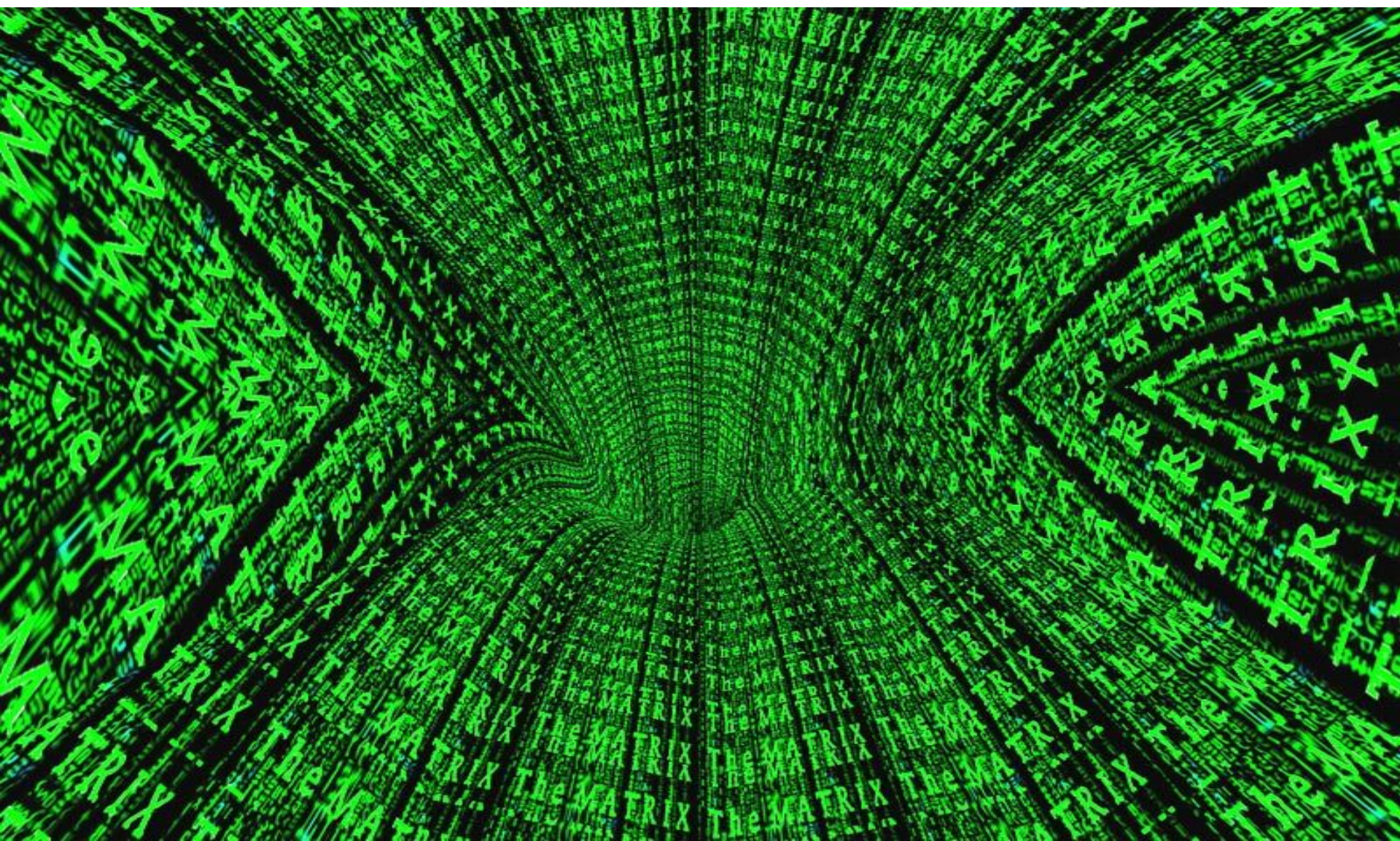
"It is a long tail, certainly," said Alice, looking down with wonder at the Mouse's tail; "but why do you call it sad?" And she kept on puzzling about it while the Mouse was speaking, so that her idea of the tale was something like this:—"Fury said to

a mouse, That
he met in the
house, ' Let
us both go
to law: I
will prosec-
ute you.
Come, I'll
take no de-
nial: We
must have
the trial;
For really
this morn-
ing I've
nothing to do.'
Said the
mouse to
the cat,
' Such a
trial, dear
Sir, With
no jury
or judge,
would
be want-
ing our
breath.'
' I'll be
judge,
I'll be
jury,'
said
creas-
ing
and
Purr-
' I'll
try
the
whole
case,
and
the
law
too.'
And
the
mouse
said
to
the
cat,
' I'll
be
judge,
I'll
be
jury,'
said
creas-
ing
and
Purr-
' I'll
try
the
whole
case,
and
the
law
too.'

Consider a project a story

- A **starting** bit – introducing the cast and the plot
- A **middle** bit – all the excitement
- An **end** bit – happily ever after
- And a good start is critical. Puts everything in place for the happy ending

Ready?



And the story starts....

- Once up on a time...
- We had a Brilliant idea
- And a Talented team
- And...
- “Oh dear! Oh dear! I shall be too **late!**”





And we rush to follow the white rabbit

We don't have time for process



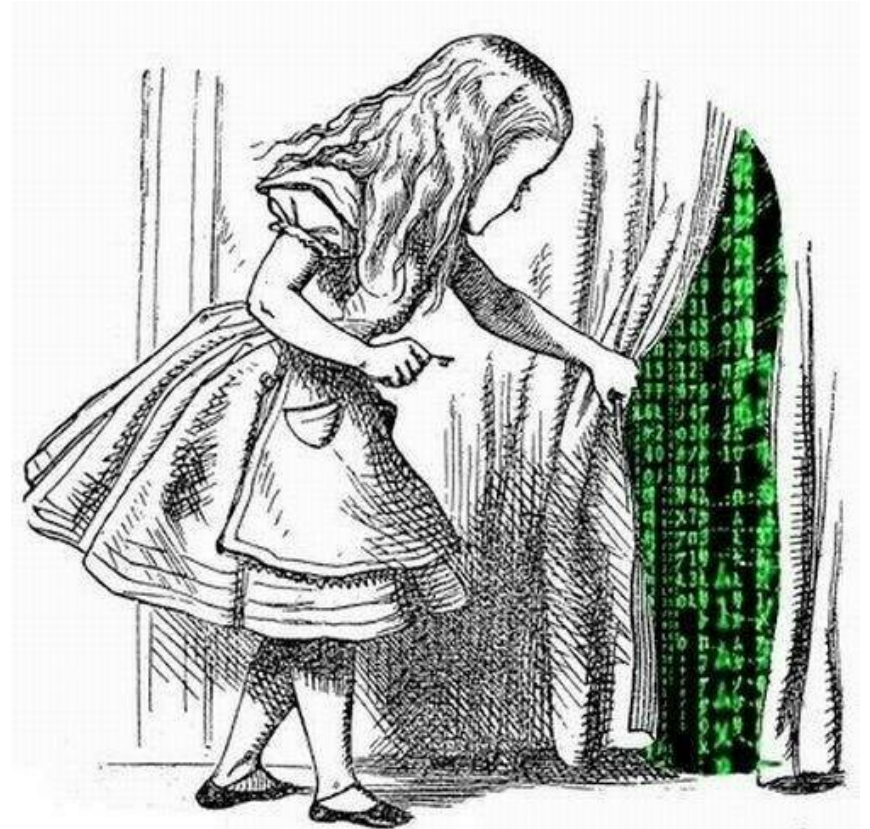
The adversaries & pitfalls

- Unknown target – hope we'll figure it out along the way
- Bringing it together
 - Sum of three brilliant parts may not make one diamond
- Getting to 100%
 - How far along are we and how do we know when we're done?
- Leaving the best for last
 - The last 10% can take forever
- We created it – but how do we see if it is good?
- How do we make decisions along the way
- We built it and OMG now they're coming



The Curtain opens

- Our project usually has
 - Intended outcome (“deliverables”) or goal
 - Start
 - Finish
 - Schedule
 - Budget

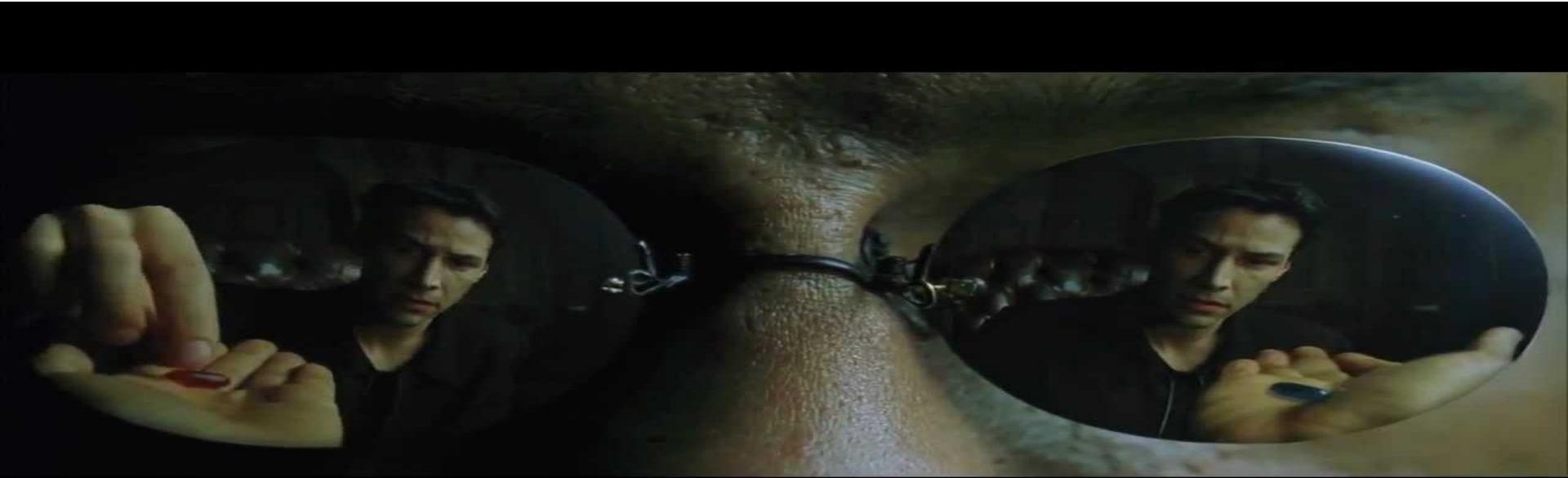


And the project is kicked off



In the beginning, Agree on the happily ever after

- How will we know when we're 'done'
- What does success look like
- And how will we measure or validate 'done'



“Done” includes Quality Assurance & Validation

- Especially for algorithms
 - What data will you need
 - How can you predict outcome by another means
 - What should stay constant
 - How many scenarios, etc.
- Error conditions



In the beginning, Agree on the happily ever after

Once the project is done,
what will happen to

- Data
- Documents
- Software
- Users
- Team



In the beginning, Agree on the happily ever after

Thinking about how the software will be used

- Where will it live
- User access
- Sustainment
- Funding
- Security



In the beginning... Agree on

Governance & decision making

- Which types of decisions
- Who makes them
- How will we make them

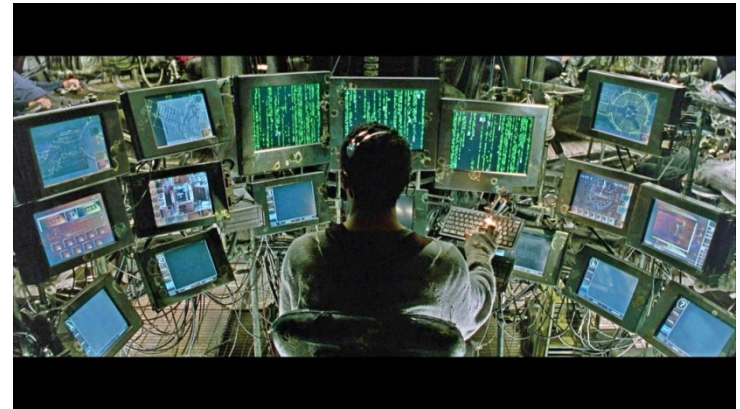


In the beginning... Agree on

- Scope – in big huge terms, what's in, what's out
- Schedule – how long do we have
- Budget – how much money do we have (and for which part)
- Process:
 - how will we run the project
 - what do we need to report/track progress (funding agencies?)

Consider the infrastructure

- DEV – QA – PROD
- What works on your DEV station – does it work somewhere else?
- What works standalone on your DEV stations – does it work with the other team member's code?
- Version control – to deal with “it worked yesterday”
- Be aware of dependencies
 - File system structure
 - Data or data structure
 - Libraries
 - OS
 - Tools
 - Etc.
- A separate QA environment helps prove and demo
- A separate PROD for more secure and stable environment



Project Charter (or Terms of Reference)



- Clarity
- Agreement
- Sorting this out at the start saves SO much time later
- Can be a few pages, but is vital
- There are templates

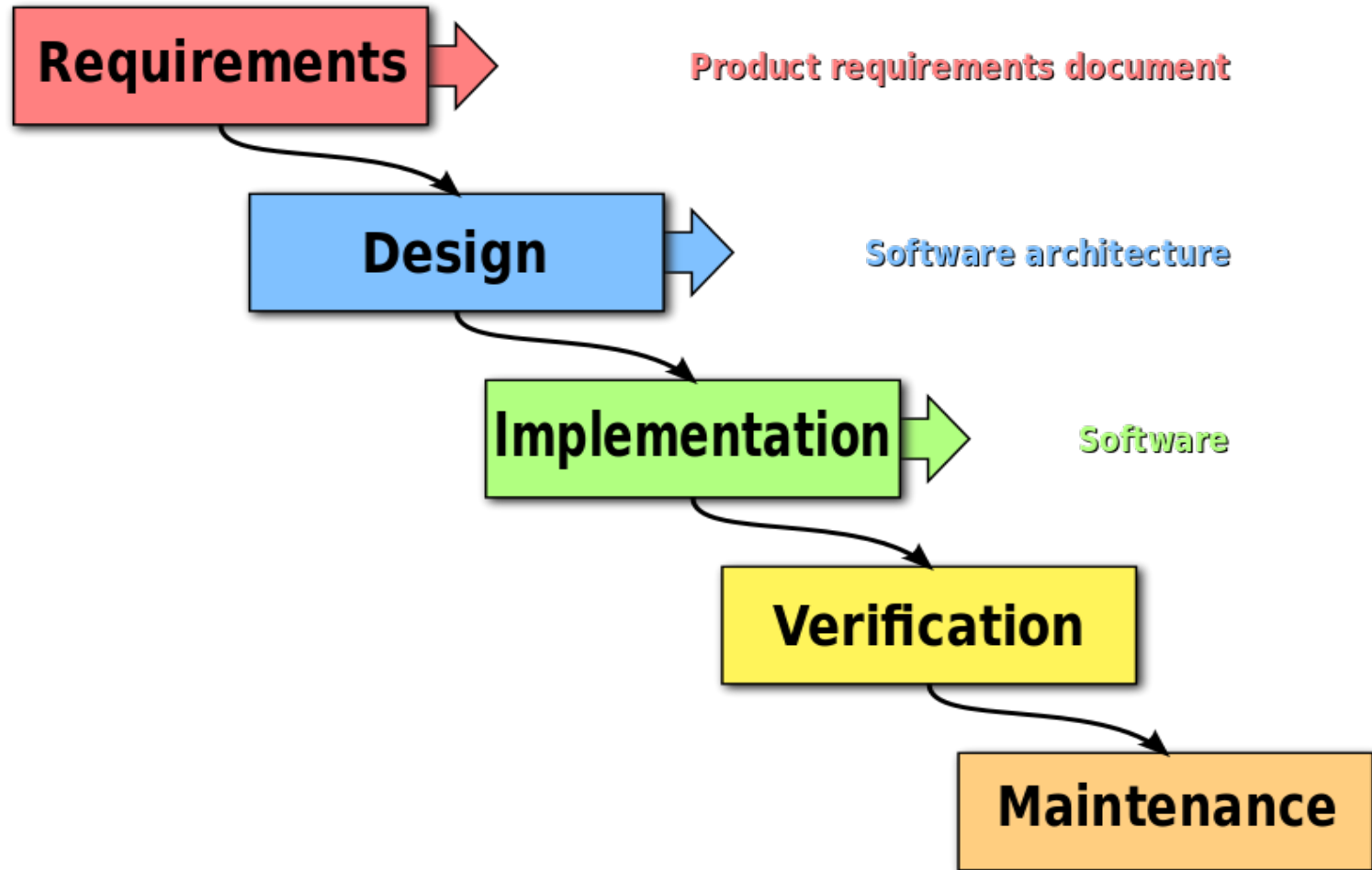
Plot thickens



Running a project

- Methodology
 - Waterfall
 - Agile
 - => see it as a toolbox, pick & choose, and much of it is common sense
- Progress tracking
 - Creates clarity
 - Allows decisions

Waterfall



Waterfall

- Traditionally, IT projects were based on an engineering approach
 - Large cost of change
 - Ability to specify outcome
 - Language for specification
- Everything designed and defined at the beginning of the project
- Change is controlled, schedule is committed to
- And for some IT projects, this is the right approach!

Incremental and Iterative
Change is embraced (so has to be cheap) Agile
Less definition and specification, more delivery

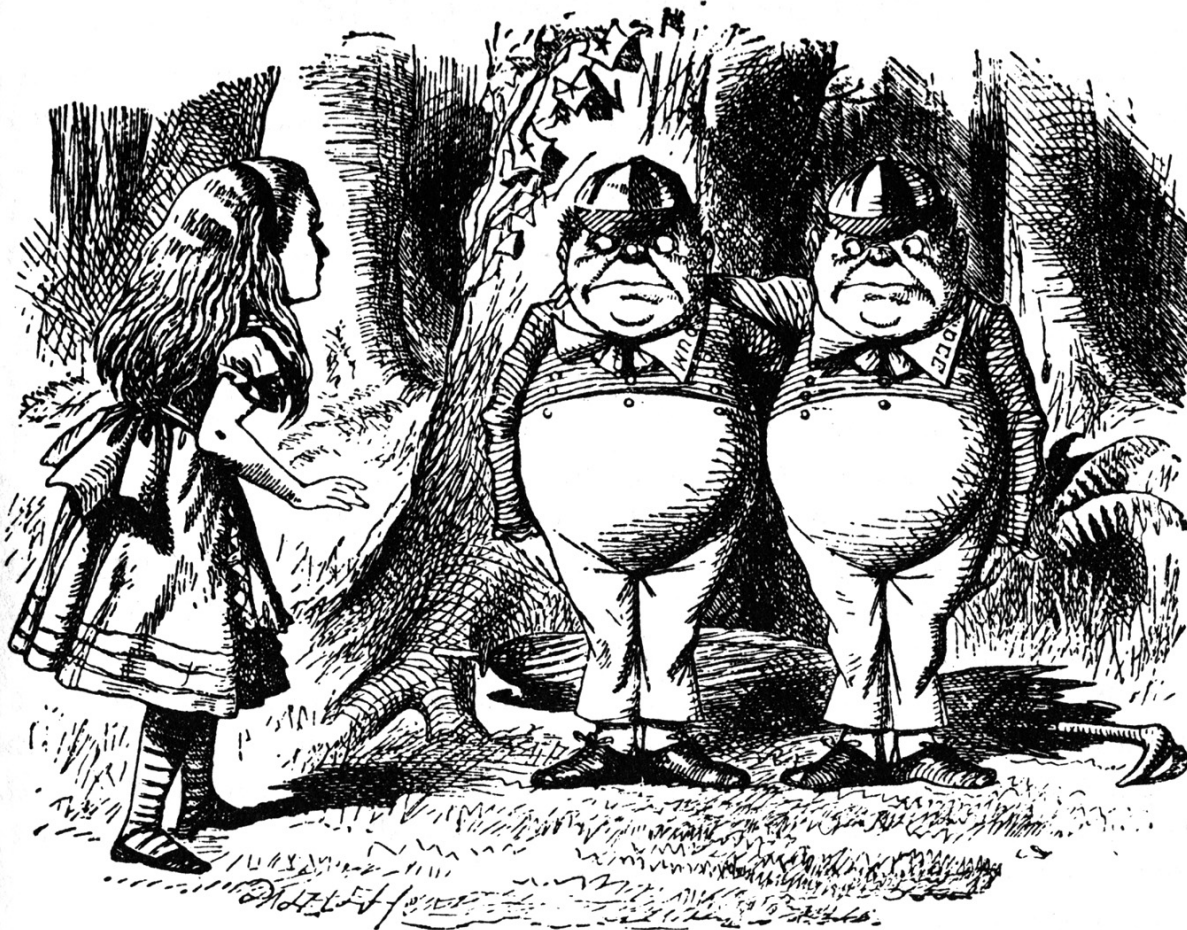


Why Agile for research?



- Early validation
- Lots of adjustment
- MVP approach
- Small teams

A Daily Stand-up does not an Agile Project make



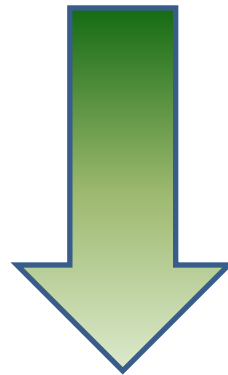
Agile Software Development

- Backlog of small pieces (features)
 - decomposition
- Prioritization (“grooming”)
- Deliver and Demo
- Review and **adjust – next iteration**
- Done
- Rinse and Repeat
 - Danger: don’t keep reworking the first bit and never get to the end bit

Agile approach variants

- Scaled Agile
- Scrum
- Kanban

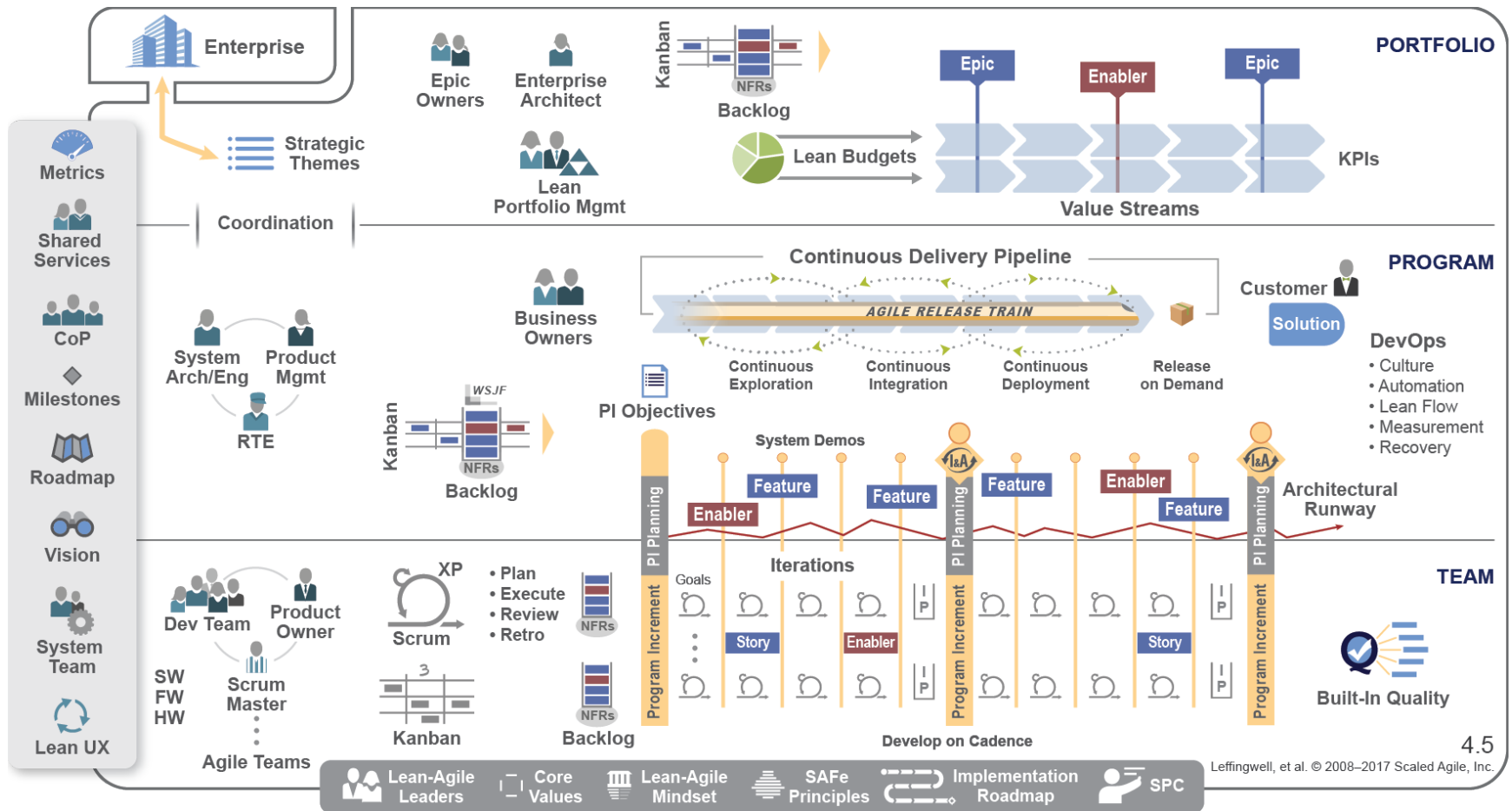
More complex



Less complex



Large projects – Scaled Agile



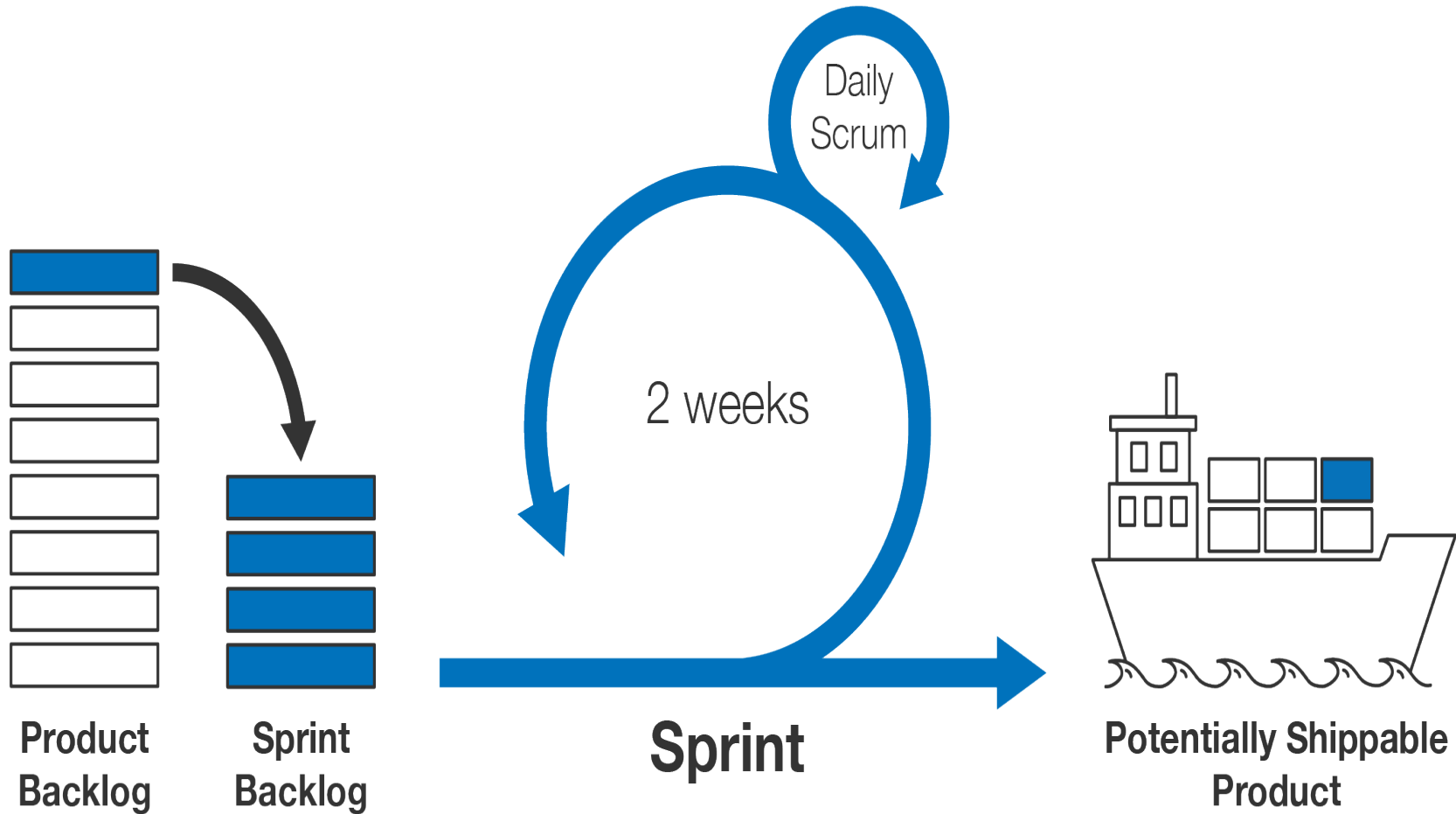


NOT SUGGESTING YOU DO THIS 😊

Scrum

- Prioritized backlog
 - Features
 - Acceptance criteria
- Fixed time (typically 2 weeks) – “Sprint”
- Definition of Done
- Team commits to selecting doable priorities and getting them to done
- Something almost done is not done and moves to the next sprint

Sprint



Other typical aspects of agile

- Collaboration – self organized teams
- Colocation
- Frequent quick group meets (daily stand-ups, 5-10 mins)
- Roles
 - Product manager -> product focus
 - Team (dev & test) -> delivery focus
 - Scrum master -> process focus
- Cross functional
- Plan & retrospective – continuous improvement

Kanban

- Work management approach
- Work is pulled when capacity becomes available
- From prioritized list (“backlog”)
- Progress (next bit) shown on kanban board

Progress tracking

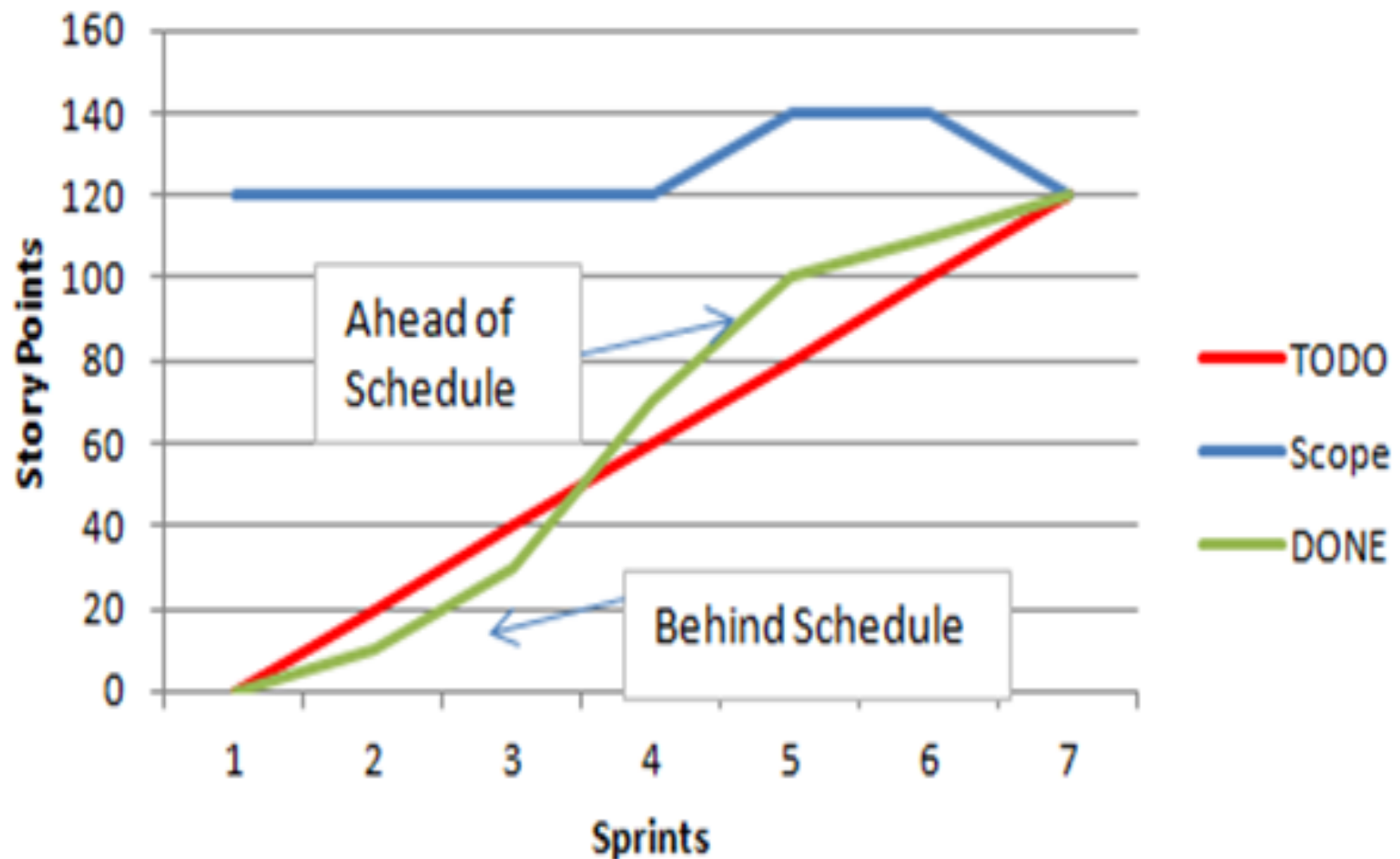
- Helps communication in the team
 - Uses schedule and budget responsibly
 - Allows for adjustment
 - Great for PR
-
- Does NOT have to take a lot of time!



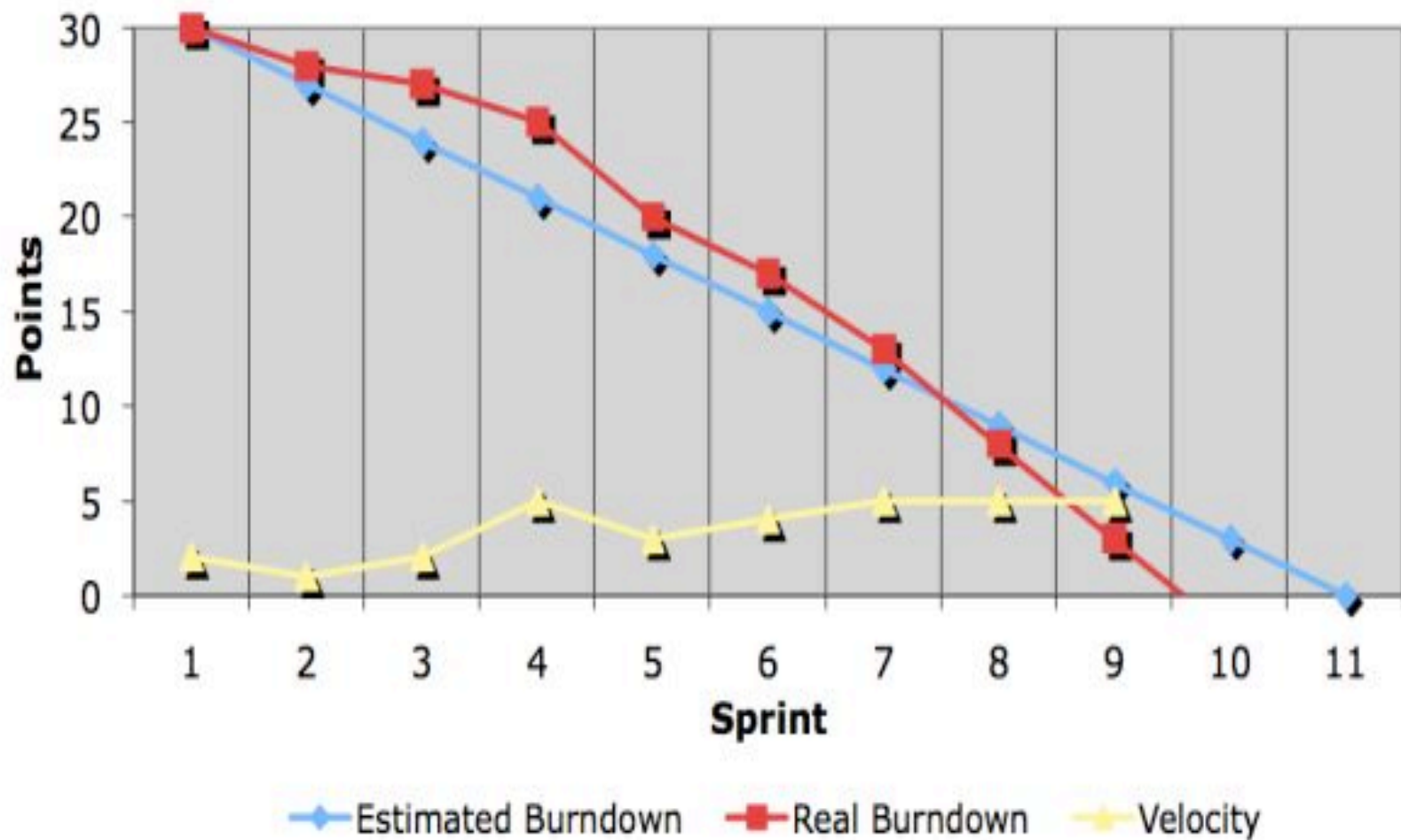
Burn up/down charts

- Getting to 100% done (or 0% left)
- How far along are we?
- Are we on track?
- Total items to be done, curve that would get everything done at the end of project, and team's progress by period

Release Burnup Chart























Product BurnDown






Power of Dashboards

- Progress
- Readiness
- Helps discussion and decision

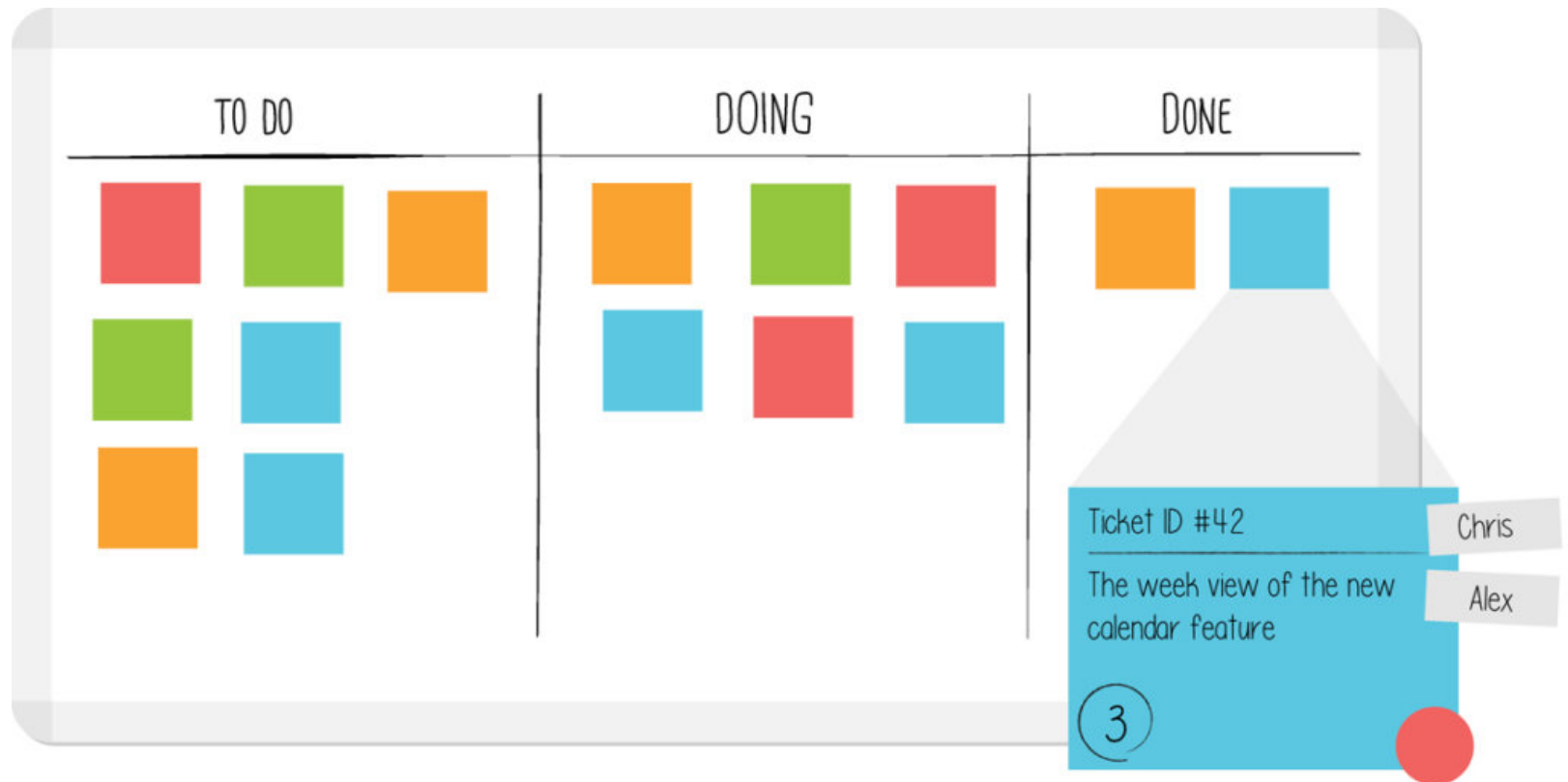
Change Readiness – Dashboard

	Organization	People	Process	Reality
Practice A				
Practice B				
Practice C				
Practice D				
Practice E				

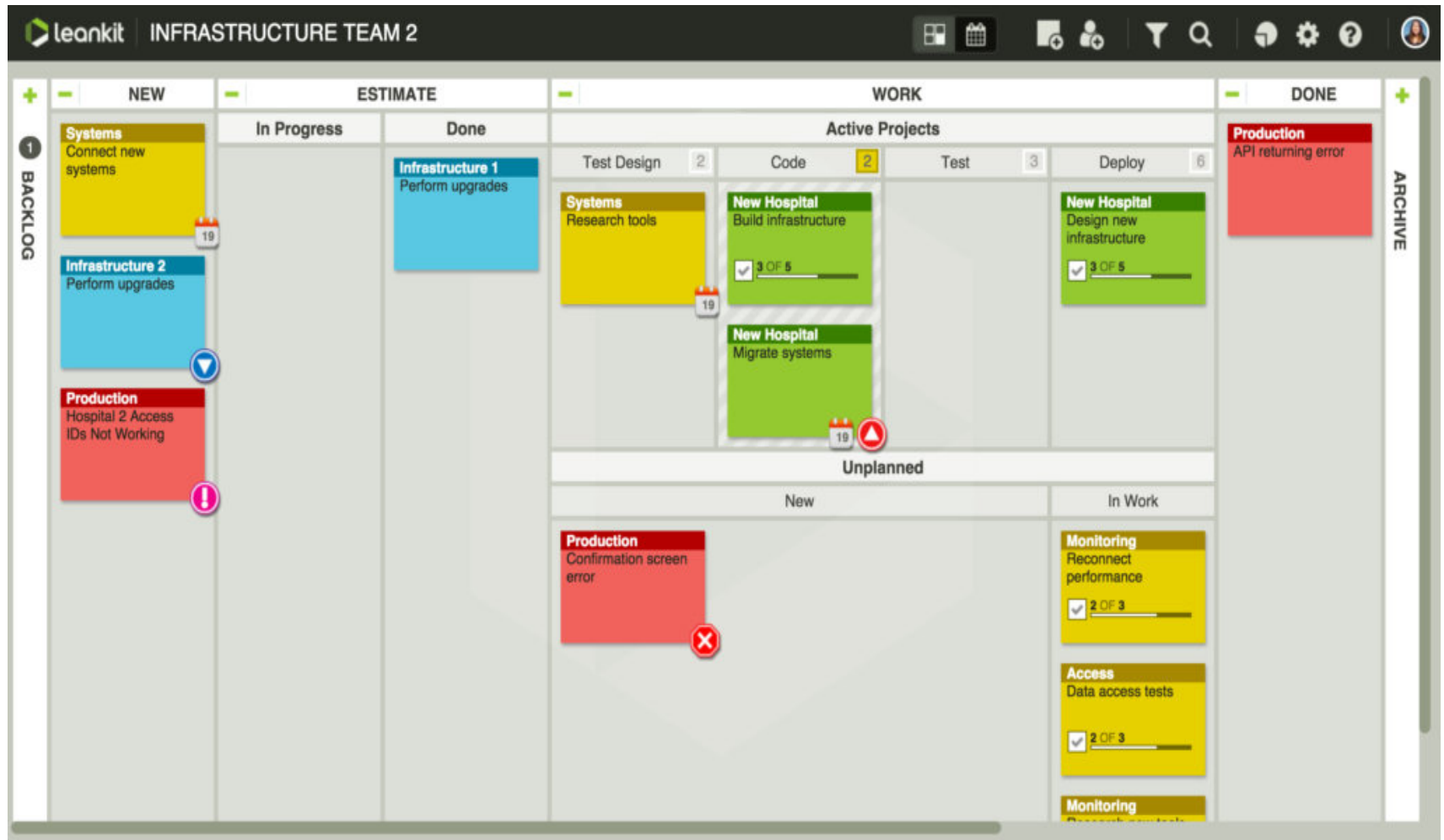
-  Ready to Proceed
-  Some Risk: Additional Preparation
-  High Risk: Significant Preparation

Kanban board

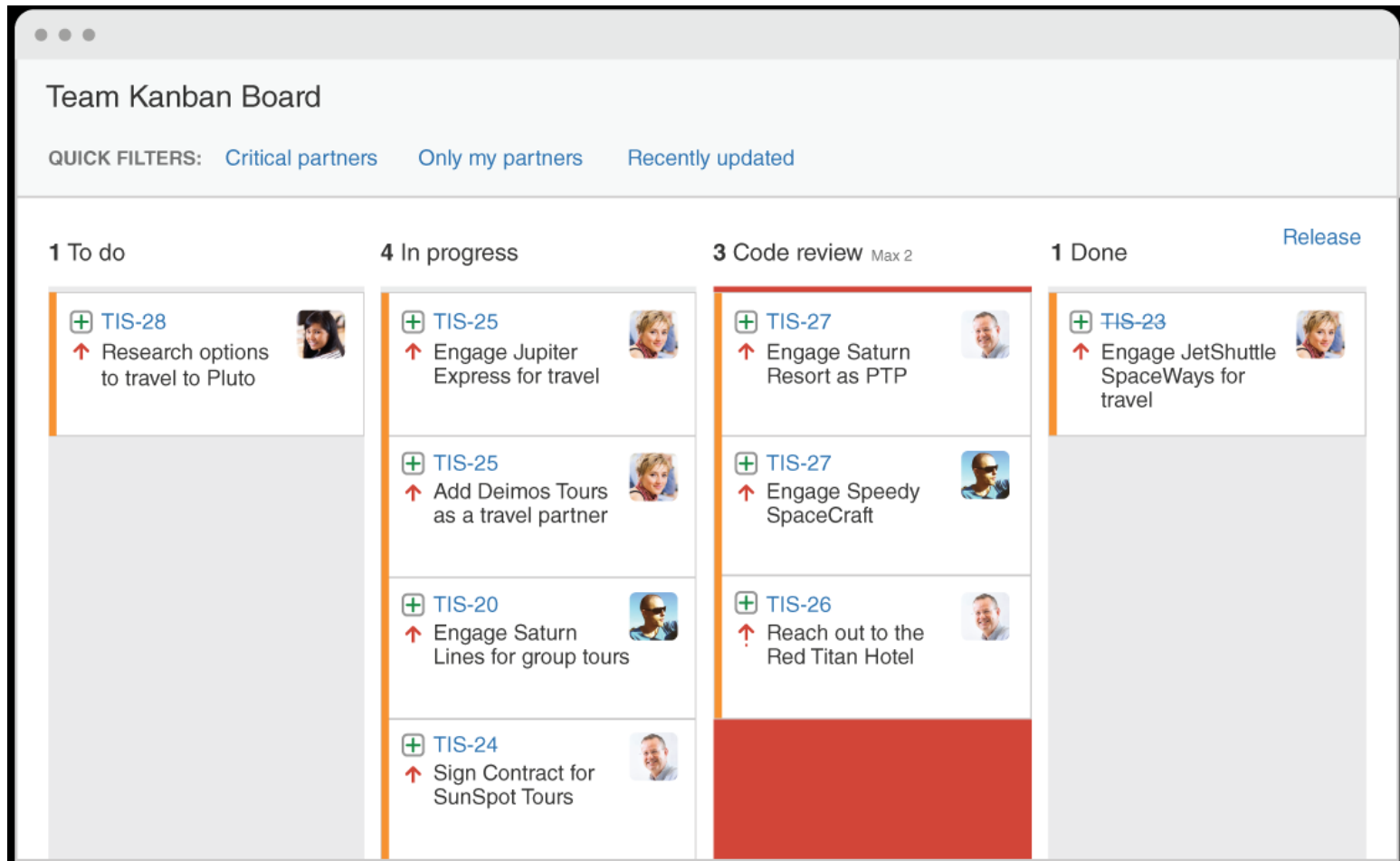
- Public presentation
- Tool or stickies



Kanban



Kanban



So how are we doing on our plot

And our mighty
adversaries...



Unknown target

(This is research, after all....)

- Roadmap / backlog
- Review & Adjust frequent and early
- Spikes
- Next sprint may be a full redo
- Prototype or concept can be “done”
- Prove out small pieces
- Demo allows for team review and critique

Bringing it together

(Sum of three brilliant parts may not make one diamond)

- “Done” means built and tested
- “Done” means Demo’d and accepted
- Infrastructure environment
 - DEV-QA-PROD
 - Avoid dependencies
- SAVES YOU SOOOO MUCH TIME

Getting to 100%

- Definition of Done
- Dashboarding and progress tracking
- Standups
- Reprioritize when things change



Leaving the best for last

- The last 10% can take forever
- Prioritize your backlog
 - Prerequisites first
 - Complex first
 - Must haves first
- You'll have a lot of things that are done (rather than a lot that is almost done)



We created it.. (but will it hold up?)

- Demo's
- Validation, testing, quality assurance
 - (even better if you can define a feature to automate this!)
- Acceptance criteria



How do we make decisions

- Backlog grooming
- Governance (charter)
- People change, people's minds change – process helps
- Preset your decisions
 - Acceptance criteria
 - Methodology
 - Cadence

We built it and OMG now they're coming

Once the project is done, what will happen to

- Data
- Documents
- Software
- Users
- Team
- Sustainment
- Security



And... It's a wrap ("The End" bit)

- Lessons learned
- Publish method
- Store stuff
- Hand over to a sustainment or Ops team

- So... to sum up



Agile for Research Software Dev

- Start with the end especially
 - Definition of DONE
 - Validation
 - What does your happily ever after look like
- Create your backlog
- What is your MVP – prioritize
- Try, review, improve, try again
- In a fixed time (scrum) or purely prioritized (kanban)

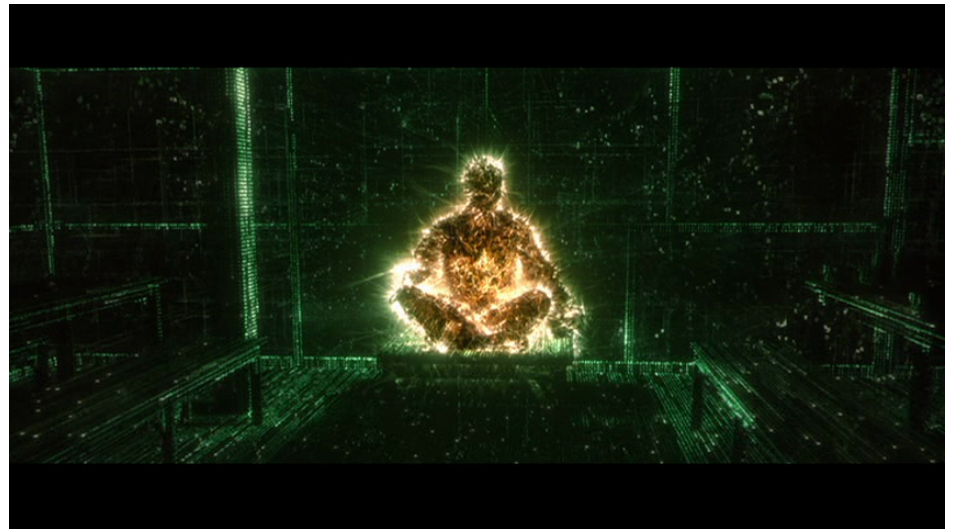
In the beginning, Agree on the happily ever after

- Write a charter
- Decide your process & tools & environments
- If scrum, then define your sprint length
- Decide how you track progress and when to do standups



In the beginning, Agree on the happily ever after

- Thinking about how the software will be used
 - User access
 - Sustainment
 - Security
 - Data management



Agile for research Software Dev

- Trust that it IS faster and less wasteful than just going at it
- When you hear “Oh dear! Oh dear! I shall be too **late!**”
- Don’t follow the white rabbit into the rabbit hole
- **Plan**



The Happy End





Questions?