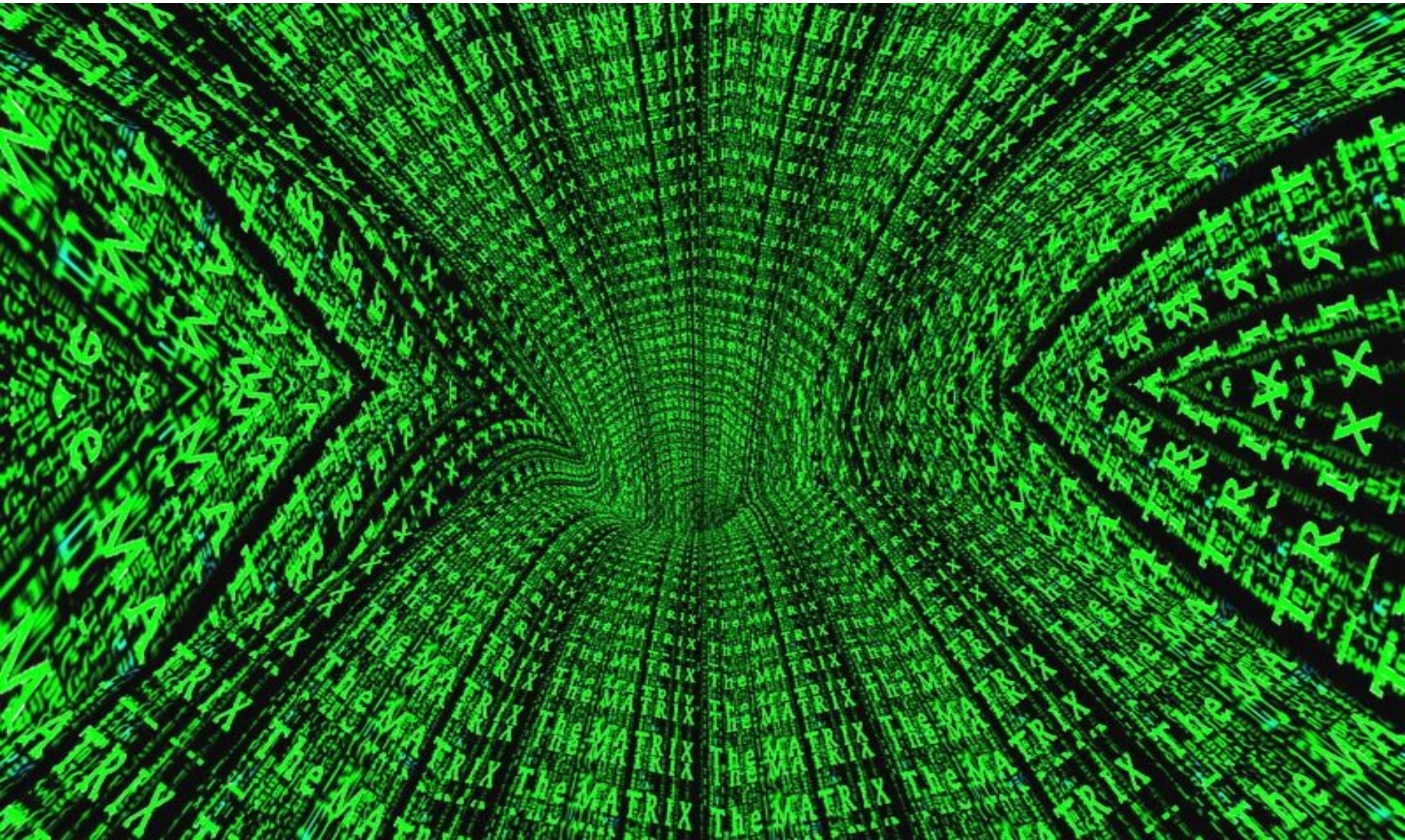
Best Practices in Software Development
~ Agile ~

Henriette Koning, Director IT PMO
Stemcell Technologies

(in 20 minutes… ☺)



- Not tools, algorithms, software engineering techniques…
- But process of organizing the work

1. Time

- Once upon a time…
- We had a Brilliant idea
- And a Talented team
- And…
- “Oh dear! Oh dear! I shall be **late**!”

And we rush to follow the white rabbit

We don’t have time for process

1. Time

- Time estimation is hard – esp for innovation or research
- Planning is hard – esp for innovation or research

- Especially when you have no time, you have NO TIME TO WASTE!!

➢ Agile process helps you stay organized

2. Unknowns

- Scope
- Algorithms
- Problem
- Solution
- Etc.

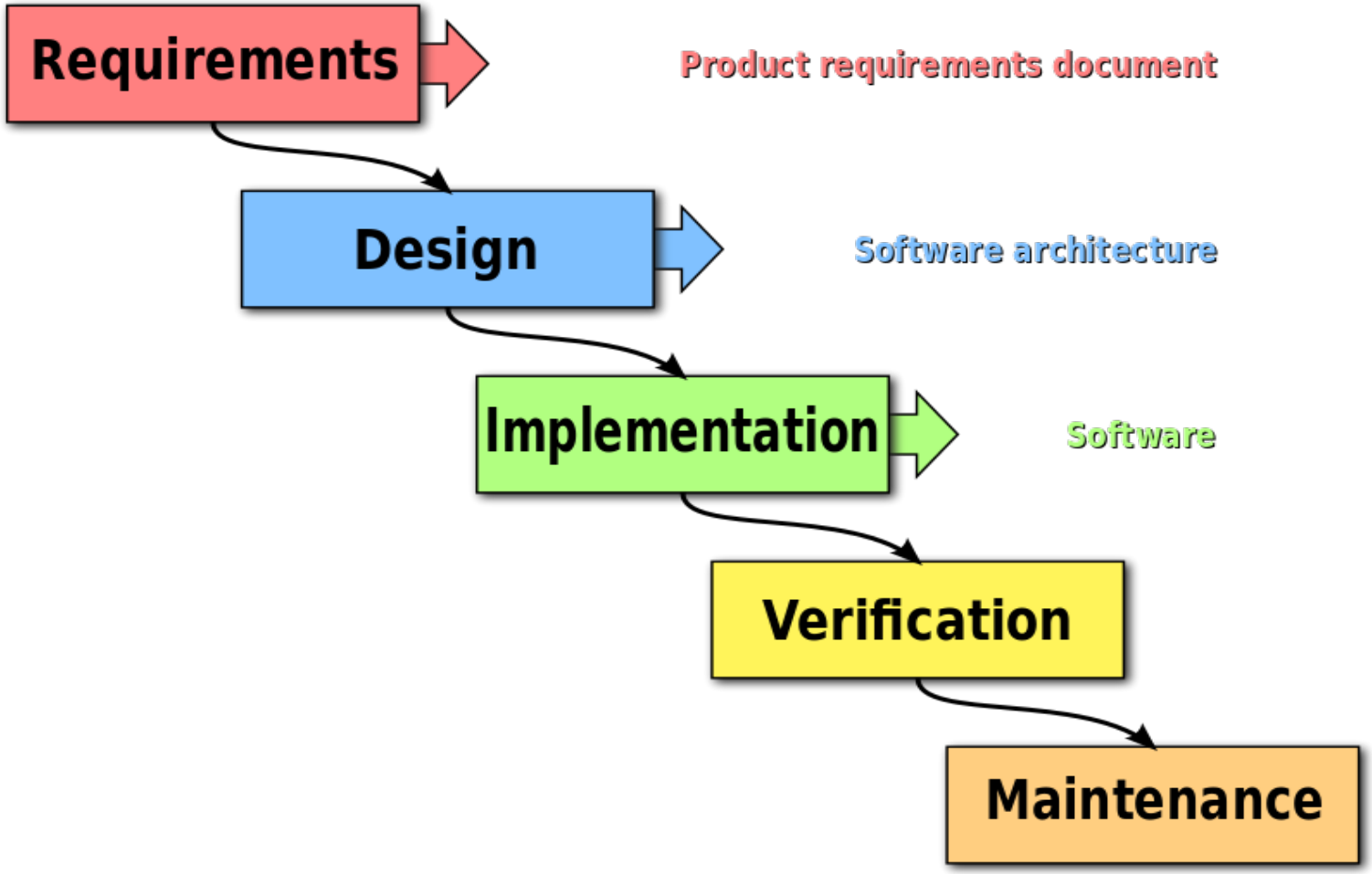
➢ Agile process does not require definition up front

Three best practices

1. Scrum
2. Test driven development
3. Definition of Done

1. Use the Agile “Scrum” approach

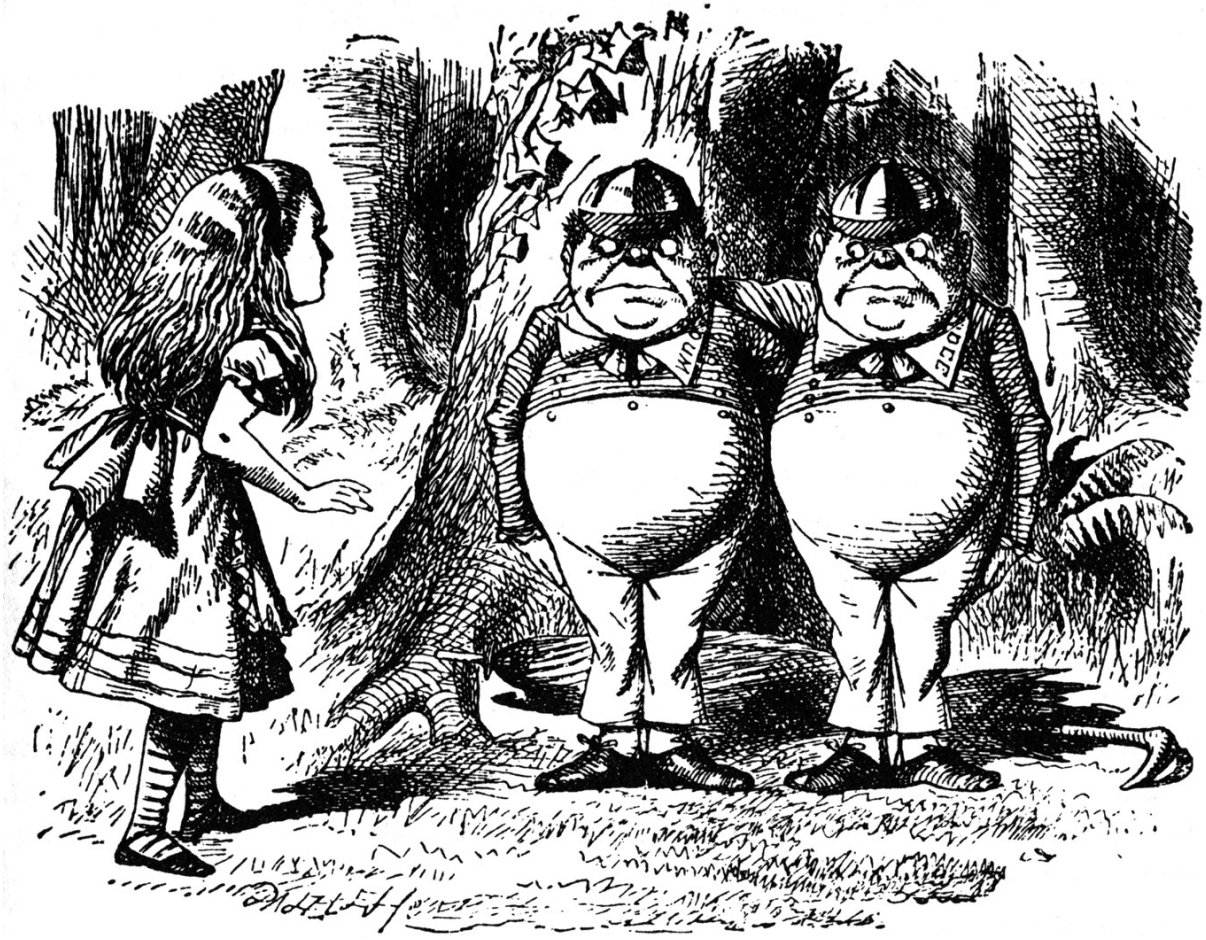
Waterfall


Requirements
Product requirements document
Design
Software architecture
Implementation
Software
Verification
Maintenance

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!
- But … research?

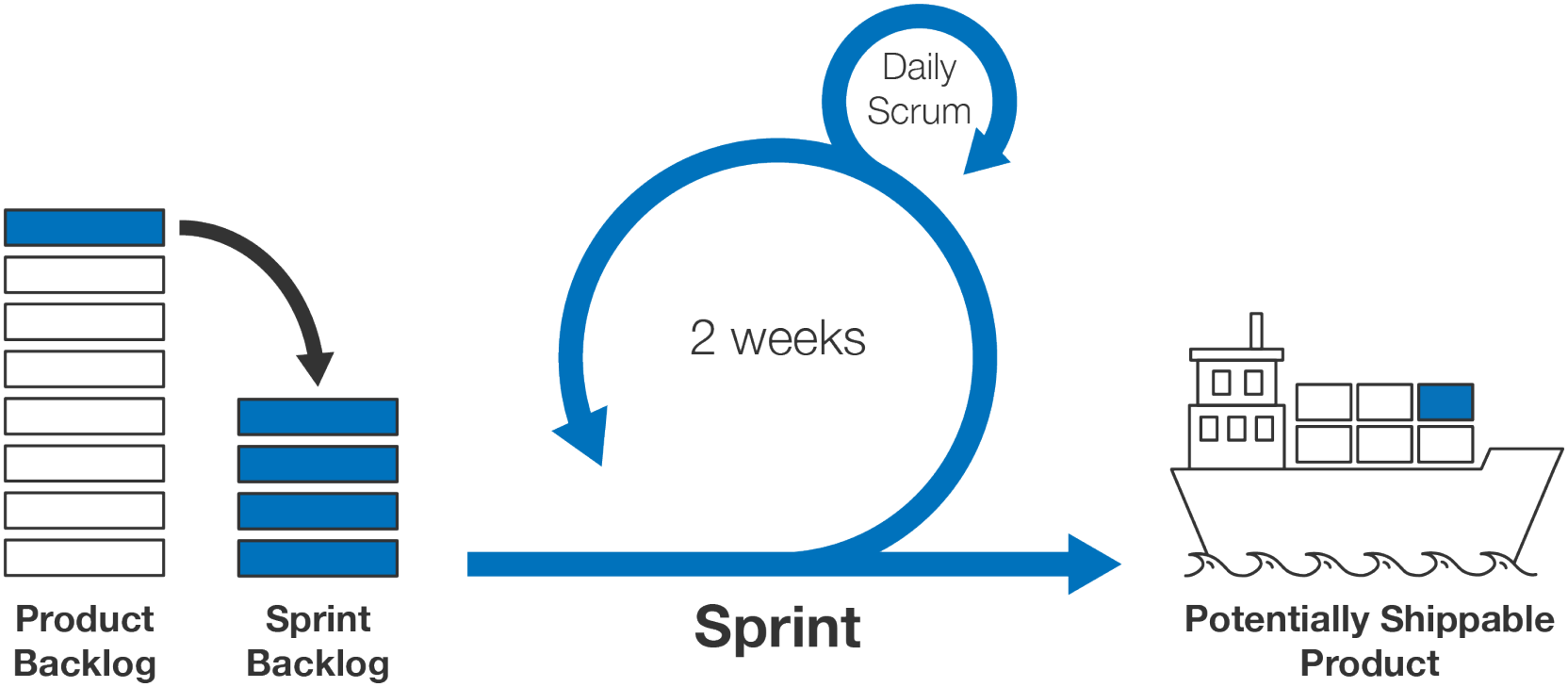
A Daily Stand-up does not an Agile Project make



Scrum

1. **Product Backlog**
   - Features
   - Acceptance criteria
2. Fixed time (typically 2 weeks) – "**Sprint**"
3. **Definition of Done** for each feature
4. Team commits to selecting doable features and getting them to Done within the Sprint ("**sprint backlog**")
5. But… Something almost done is not done and moves to the next sprint

Sprint

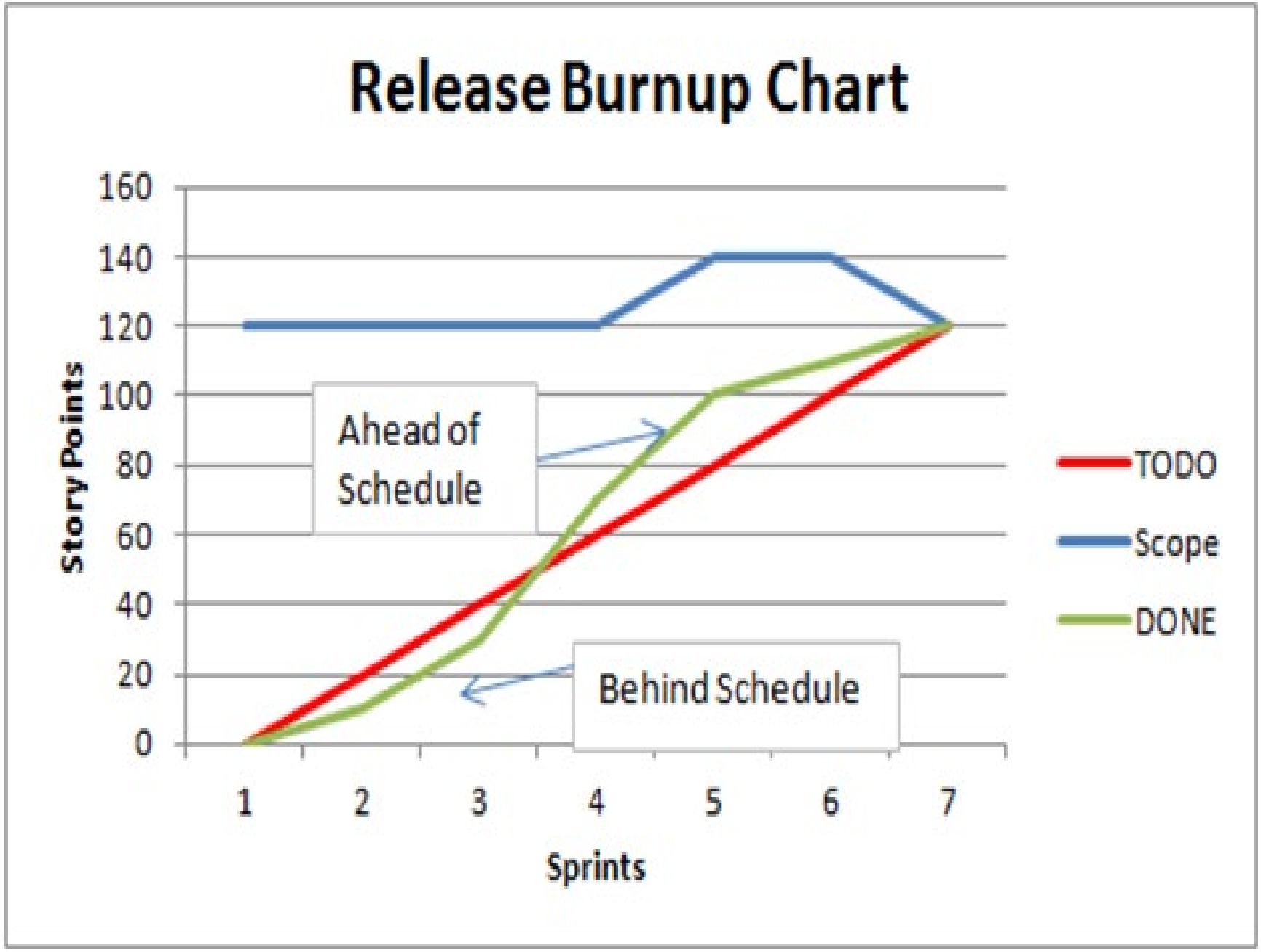

Daily
Scrum
2 weeks
Product
Backlog
Sprint
Backlog
Sprint
Potentially Shippable
Product

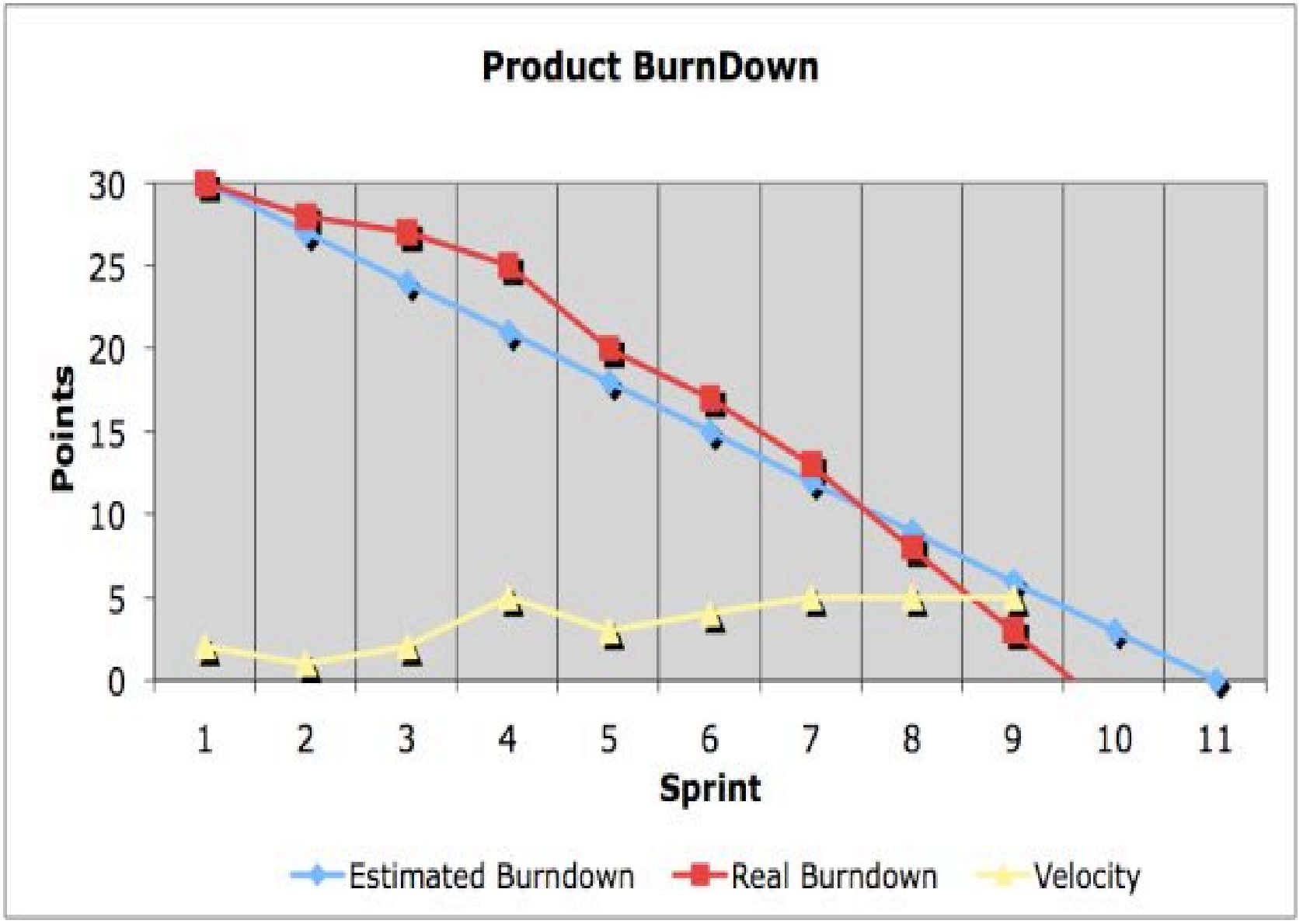
Scrum

- Build, Test and Demo Code
  - ”shippable” : meets acceptance criteria
- Review, adjust & plan next sprint
  - Every sprint: re-prioritization (“grooming”)
- Rinse and Repeat
  - Careful! don’t keep reworking the first bit and never get to the end bit

Scrum techniques

- Clear & frequent metrics on progress – burnup/down charts
- Problem decomposition (features or “user stories”)
- Retrospective
- Daily standup


Release Burnup Chart
160
140
120
100
80
60
40
20
0
Story Points
Ahead of Schedule
Behind Schedule
1
2
3
4
5
6
7
Sprints
TODO
Scope
DONE


Product BurnDown
Points
30
25
20
15
10
5
0
1
2
3
4
5
6
7
8
9
10
11
Sprint
Estimated Burndown
Real Burndown
Velocity

Every sprint

Backlog

QUICK FILTERS: Only My Issues Recently Updated

SINT Sprint 24 23 issues ACTIVE

22/Apr/19 3:46 PM • 03/May/19 7:46 PM View linked pages

Execute Improvements, Prepare for TC3

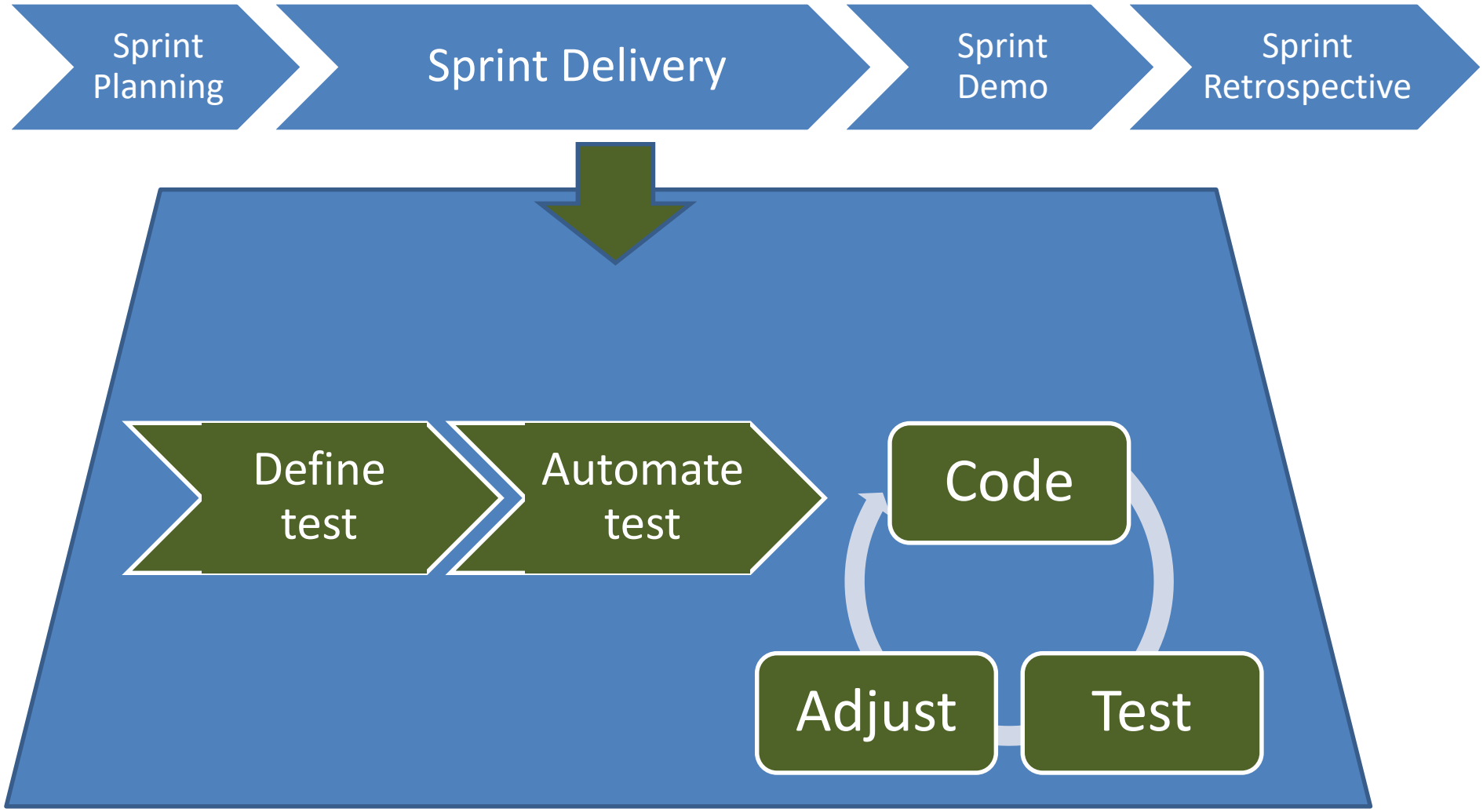
- ~~SINT-3395~~ AWS - Setting up performance logging for APIs
- SINT-3397 xe.com
- ~~SINT-3398~~ Implement MaSS Dashboard
- ~~SINT-3400~~ AWS - API Gateway Performance Fix

Backlog 23 issues

- SINT-1303 106.2: Create Bank Payment API - US Process API
- SINT-2924 I42: Create eCommerce -Materials from SAP
- SINT-2978 I46:Create eCommerce - Checkout to Order Integration

2. Test Driven Development

Every sprint



For each feature

1. Acceptance criteria to agree on “Done”
2. Test cases to show “Done” & quality
3. Code
4. Prove Done

Test driven development

- What data will you need
- How can you predict outcome by another means - baseline
- What should stay constant
- How many scenarios/use cases, etc.
- Error conditions
- Use cases

3. Definition of Done

DoD

- What does success look like
- What is included in “done”
  - For the project
  - For a feature
- Agree on what Done means!

SAP Integrations / SINT-3316

SAP Workday Integration - Elapsed Time Logging

Estimate: Unestimated

Smart Checklist

Add a checklist item

- [ ] Mule Unit Functioning
- [ ] Mavenized Build
- [ ] Jenkins Configured
- [ ] Robot Test Executed
- [ ] Data Details Documented
- [ ] Security Rules Set
- [ ] All functional spec test

Example DoD

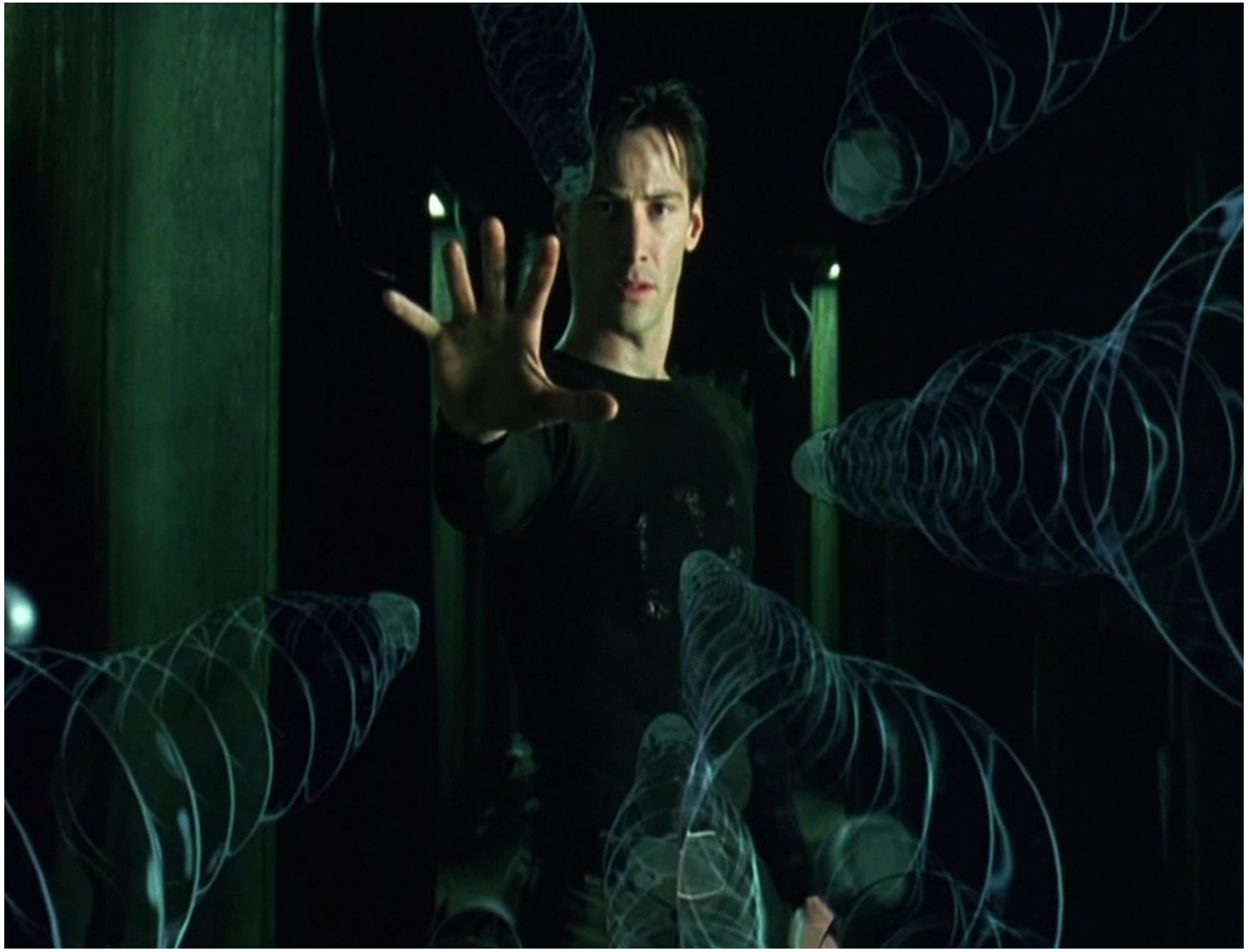
Once the project is done, what will happen to

- Data
- Documents
- Software
- Users
- Team

Backlog

- Fill your backlog not just with the algorithms but also, based on Project DoD:
  - “plumbing” (e.g. data pipeline)
  - UI – can be CLI!
  - Environment
  - Deploy
  - Security
  - Etc.
- Your first sprint goal could be to build a backlog

SUMMING IT UP



Best practices

1. Scrum cadence avoids the rabbit hole of ‘no time”

- Some planning to create stories for what you do know
- Deliver working & validated features each sprint
- Review, refine & replan each sprint & incorporate learnings

Best practices

2. Test driven development

- Helps clarify problem & solution
- Helps validate solution & contain scope

3. Definition of Done helps clarity for a feature

- Focus on Done helps backlog prioritization
- Thinking about ‘after done’ helps sustainability

Questions?