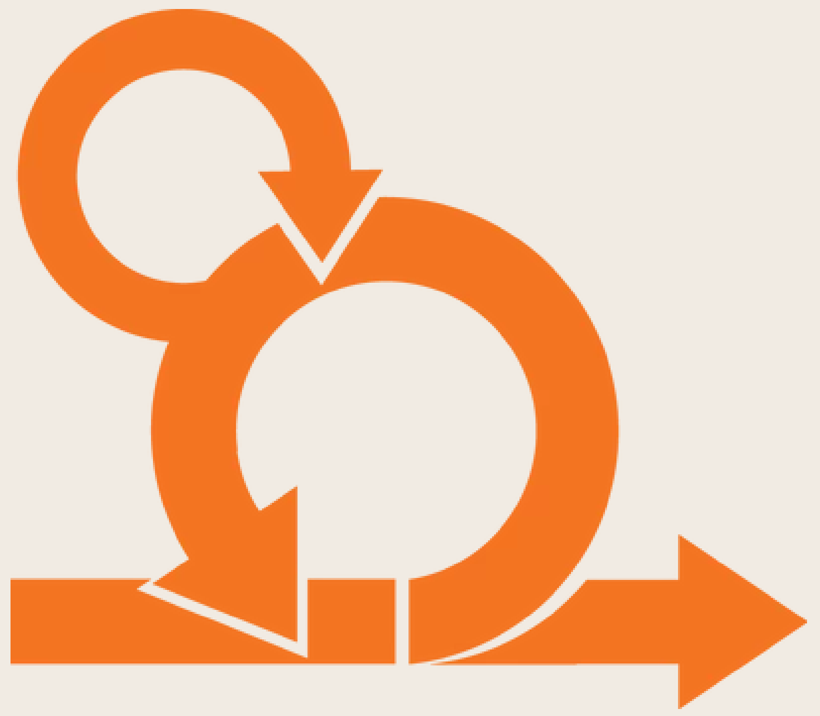


AGILE SOFTWARE DEVELOPMENT

Performance Management in Agile Teams

Project performance

Favorable conditions

Interesting project
Involved customer
Mature team

Unfavorable conditions

Unappealing project
Disengaged customer
Junior team
New technology
High-risk domain

CHALLENGES

Because of ...condition (cause)..

It will/might happen that ...trigger.........................

Leading to ...effect...

Types of challenges

Risks

Probability of condition < 100%

Probability of trigger = 100%

Strategy: mitigation, contingency, transfer

Issues

Probability of condition = 100%

Probability of trigger = 100%

Strategy: solve

Assumptions

Probability of condition < 100%

Probability of trigger < 100%

Strategy: constant checking

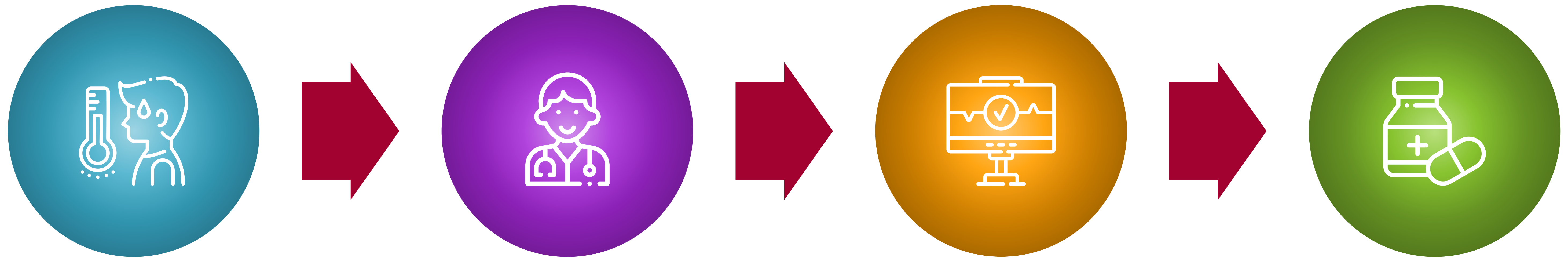
Constraints

Probability of condition = 100%

Probability of trigger < 100%

Strategy: adapt

Approach



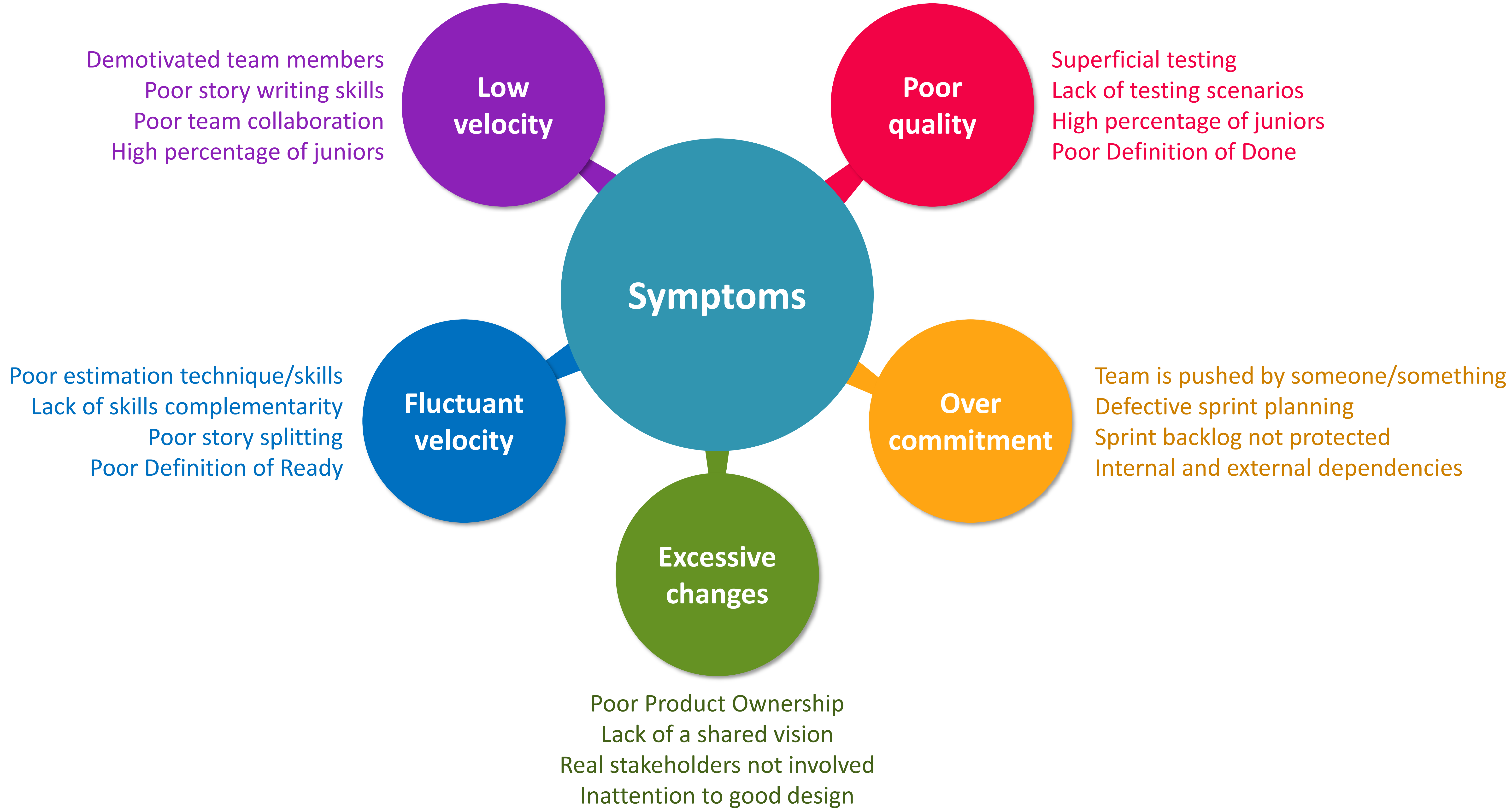
SYMPTOMS
How it manifests , what are the main perceivable effects

CAUSES
What are the most probable root causes for the symptoms

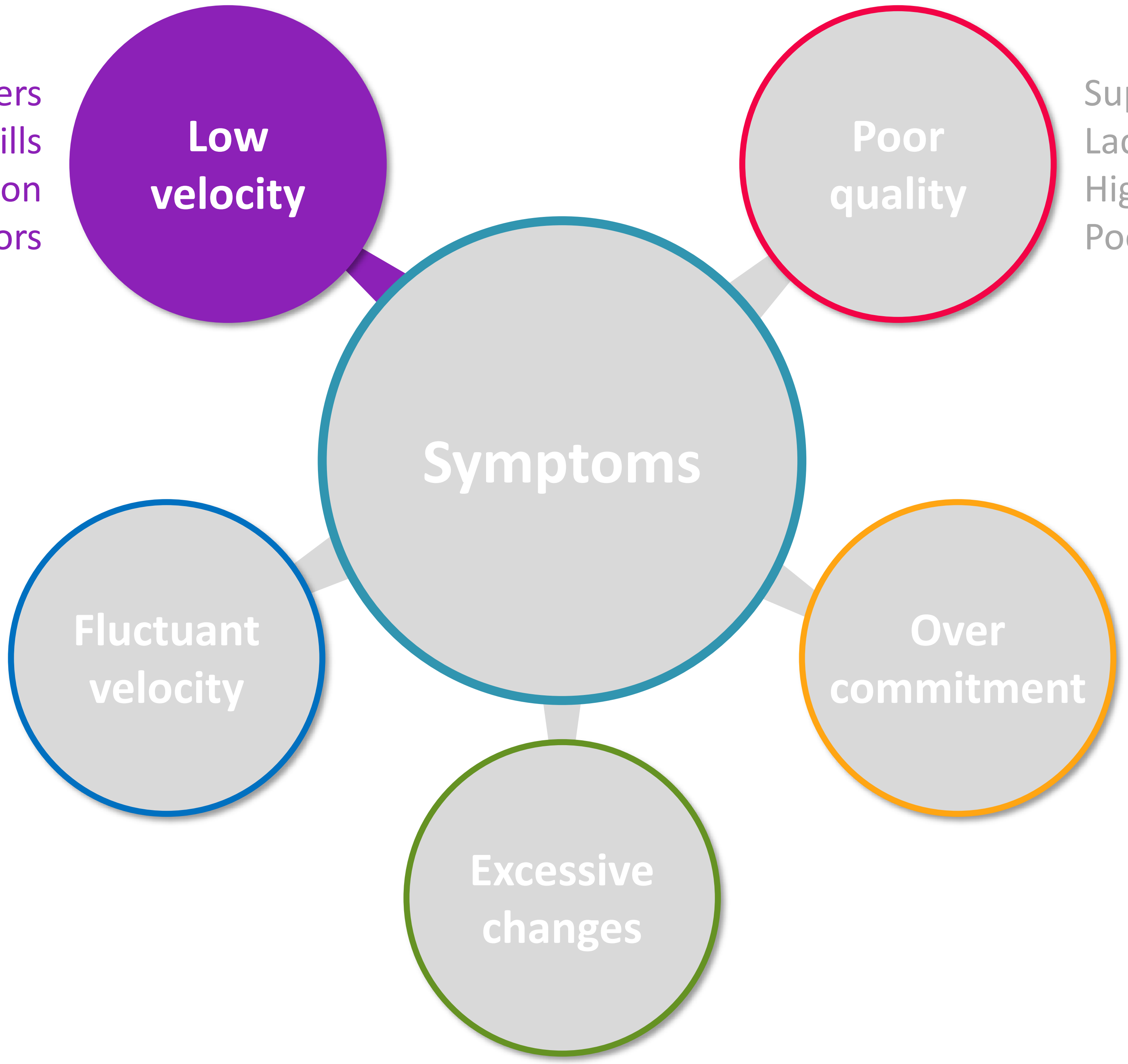
DIAGNOSTIC
How we may diagnose the nature and severity of the challenge

SOLUTIONS
What can be done to address the challenge or remove the cause

Most frequent symptoms


Symptoms
Low velocity
Demotivated team members
Poor story writing skills
Poor team collaboration
High percentage of juniors
Poor quality
Superficial testing
Lack of testing scenarios
High percentage of juniors
Poor Definition of Done
Fluctuant velocity
Poor estimation technique/skills
Lack of skills complementarity
Poor story splitting
Poor Definition of Ready
Over commitment
Team is pushed by someone/something
Defective sprint planning
Sprint backlog not protected
Internal and external dependencies
Excessive changes
Poor Product Ownership
Lack of a shared vision
Real stakeholders not involved
Inattention to good design

Most frequent symptoms



Demotivated team members
Poor story writing skills
Poor team collaboration
High percentage of juniors

Superficial testing
Lack of testing scenarios
High percentage of juniors
Poor Definition of Done

Poor estimation technique/skills
Lack of skills complementarity
Poor story splitting
Poor Definition of Ready

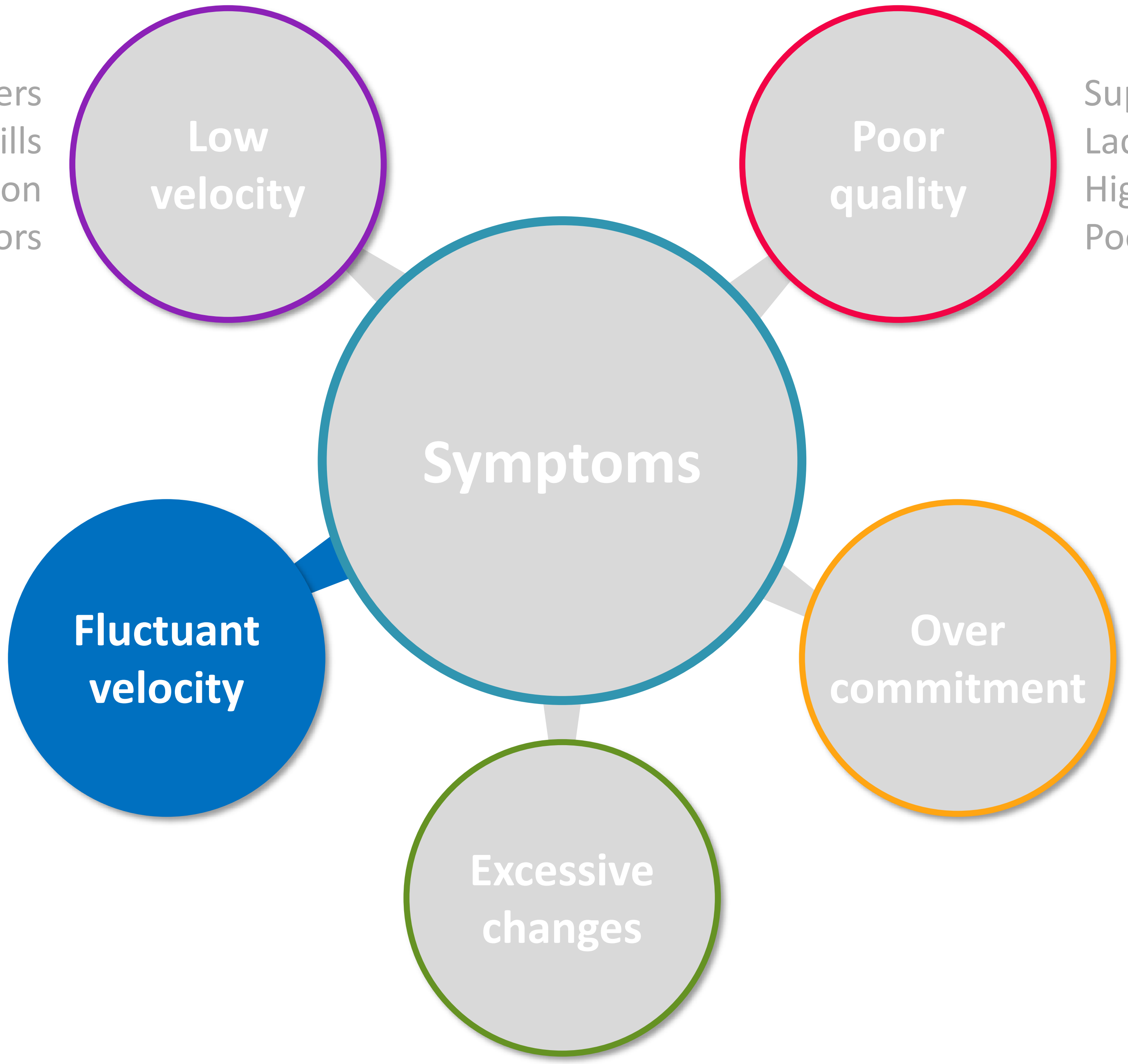
Team is pushed by someone/something
Defective sprint planning
Sprint backlog not protected
Internal and external dependencies

Poor Product Ownership
Lack of a shared vision
Real stakeholders not involved
Inattention to good design

Low velocity (compared to project complexity)

	Demotivated team members	Poor story writing skills	Poor team collaboration	High percentage of juniors
CLASSIFYING	Sometimes an issue or risk, sometimes a constraint	Always an issue	Always an issue	Usually a constraint, sometimes an issue
DIAGNOSING	Face-to-face talking Direct observation	Check INVEST rules Check acceptance criteria	Apply Gemba (mingle) Attend daily standups	Examine team CVs Direct observation
SOLVING	Seek for deeper cause Align project & team goals	Story writing meetings Use business analysis skills	Apply value stream mapping Maintain a stable team core	Get external mentoring support Replace some team members
ADAPTING	Manage stakeholders expectations	Don’t adapt, solve it!	Don’t adapt, solve it!	Manage stakeholders expectations

Most frequent symptoms



Demotivated team members
Poor story writing skills
Poor team collaboration
High percentage of juniors

Superficial testing
Lack of testing scenarios
High percentage of juniors
Poor Definition of Done

Poor estimation technique/skills
Lack of skills complementarity
Poor story splitting
Poor Definition of Ready

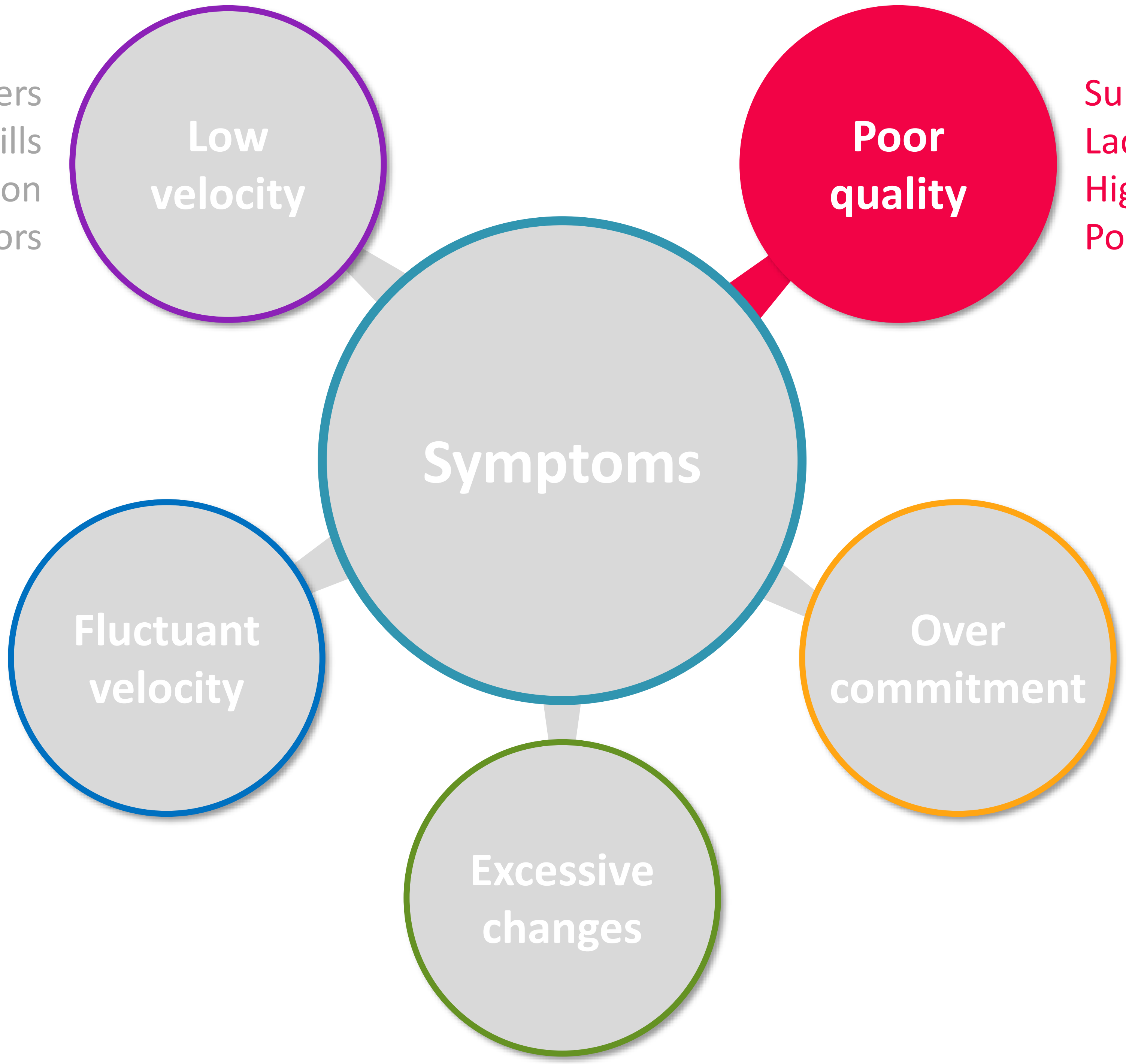
Team is pushed by someone/something
Defective sprint planning
Sprint backlog not protected
Internal and external dependencies

Poor Product Ownership
Lack of a shared vision
Real stakeholders not involved
Inattention to good design

Fluctuant velocity

	Poor estimation technique / skills	Lack of skill complementarity	Poor story splitting	Unclear Definition of Ready
CLASSIFYING	Always an issue	Sometimes an issue or risk, sometimes a constraint	Always an issue	Always an issue
DIAGNOSING	Analyze effort / SP Test previous estimations	Analyze effort / team member Look for bottlenecks	Monitor unfinished stories	Monitor sprint plannings Ask team which is the DoR
SOLVING	Review current SP system Move to a different technique	Pair working Knowledge sharing strategy	Apply splitting techniques Adopt a SP threshold	Run a clarification session Review periodically DoR
ADAPTING	Don’t adapt, solve it!	Match stories to skills	Don’t adapt, solve it!	Don’t adapt, solve it!

Most frequent symptoms



Demotivated team members
Poor story writing skills
Poor team collaboration
High percentage of juniors

Superficial testing
Lack of testing scenarios
High percentage of juniors
Poor Definition of Done

Poor estimation technique/skills
Lack of skills complementarity
Poor story splitting
Poor Definition of Ready

Team is pushed by someone/something
Defective sprint planning
Sprint backlog not protected
Internal and external dependencies

Poor Product Ownership
Lack of a shared vision
Real stakeholders not involved
Inattention to good design

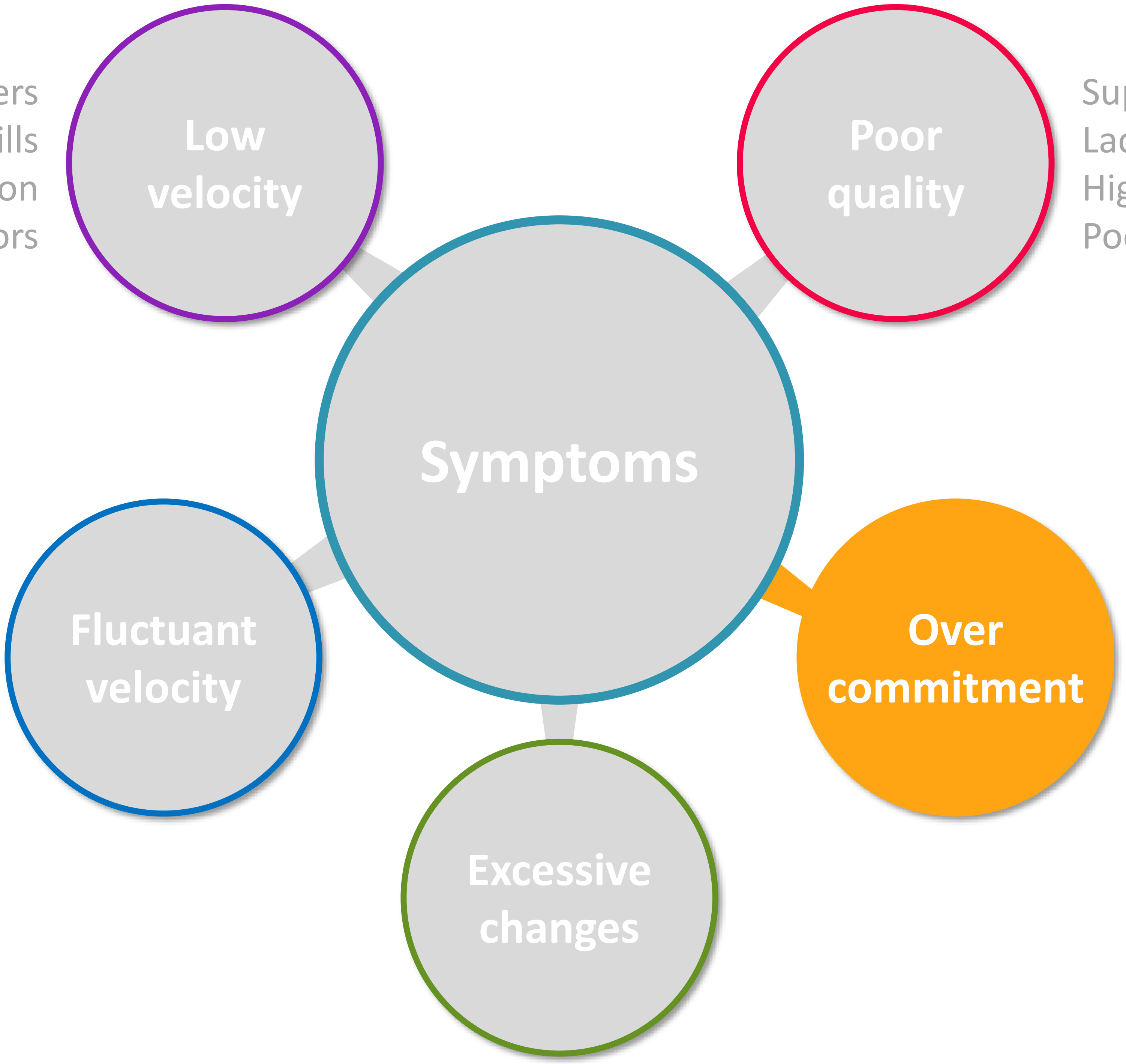
Poor quality of deliverables

	Superficial testing	Lack of testing scenarios	High percentage of juniors	Poor Definition of Done
CLASSIFYING	Always an issue	Always an issue	Usually a constraint, sometimes an issue	Always an issue
DIAGNOSING	Analyze QA effort / SP Monitor escaped defects	Inspect testing practice Examine acceptance criteria (*Given… When… Then…*)	Monitor bugs by seniority	Monitor sprint reviews Ask team which is the DoD
SOLVING	Increase test automation Introduce QA metrics	Adopt AC format Include test scenarios in DoD	Implement code review Implement unit testing	Run a clarification session Review periodically DoD
ADAPTING	Don't adapt, solve it!	Don't adapt, solve it!	Create a bug fixing squad Accept workarounds	Don't adapt, solve it!

Most frequent symptoms

Demotivated team members
Poor story writing skills
Poor team collaboration
High percentage of juniors

Superficial testing
Lack of testing scenarios
High percentage of juniors
Poor Definition of Done



Poor estimation technique/skills
Lack of skills complementarity
Poor story splitting
Poor Definition of Ready

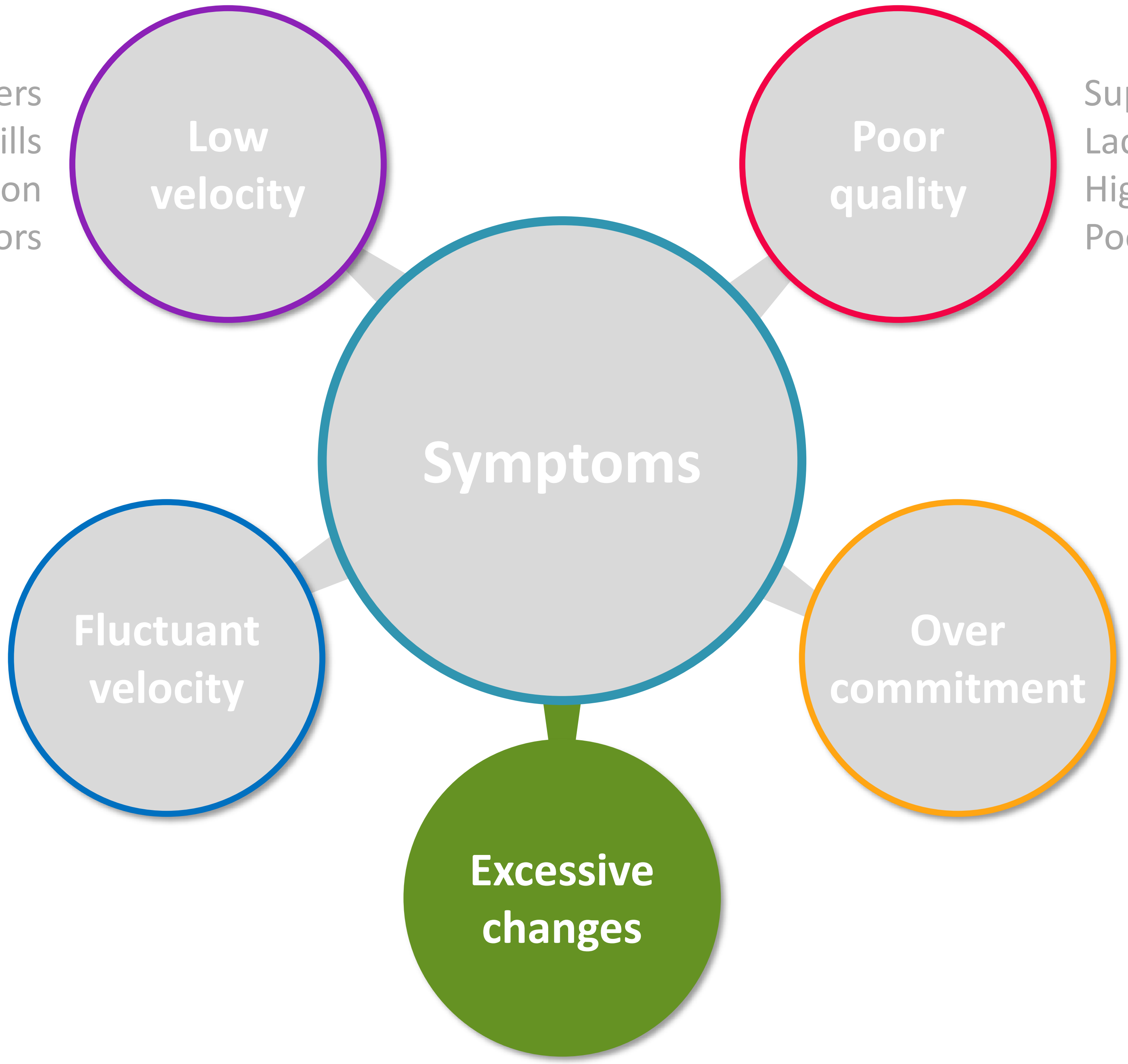
Team is pushed by someone/something
Defective sprint planning
Sprint backlog not protected
Internal and external dependencies

Poor Product Ownership
Lack of a shared vision
Real stakeholders not involved
Inattention to good design

Over commitement (constant or frequent)

	Team is pushed by someone/something	Defective sprint planning	Sprint backlog not protected	Internal & external dependencies
CLASSIFYING	Always an issue	Always an issue	Always an issue	Usually an issue, sometimes a constraint
DIAGNOSING	Monitor communication Discuss with informal leaders	Inspect planning practice Examine task allocation	Monitor changes of sprint backlog Daily Scrum/Standup	Monitor for waitings & approvals (process waste)
SOLVING	Coach the pushing person Coach team to commit	Split stories in subtasks Introduce WIP limits in sprint	Coach PO/stakeholders Coach team to discipline	Include dependency in DoR Remove dependency from DoD
ADAPTING	Don’t adapt, solve it!	Don’t adapt, solve it!	Don’t adapt, solve it!	Improve availability of external resources

Most frequent symptoms



Demotivated team members
Poor story writing skills
Poor team collaboration
High percentage of juniors

Superficial testing
Lack of testing scenarios
High percentage of juniors
Poor Definition of Done

Poor estimation technique/skills
Lack of skills complementarity
Poor story splitting
Poor Definition of Ready

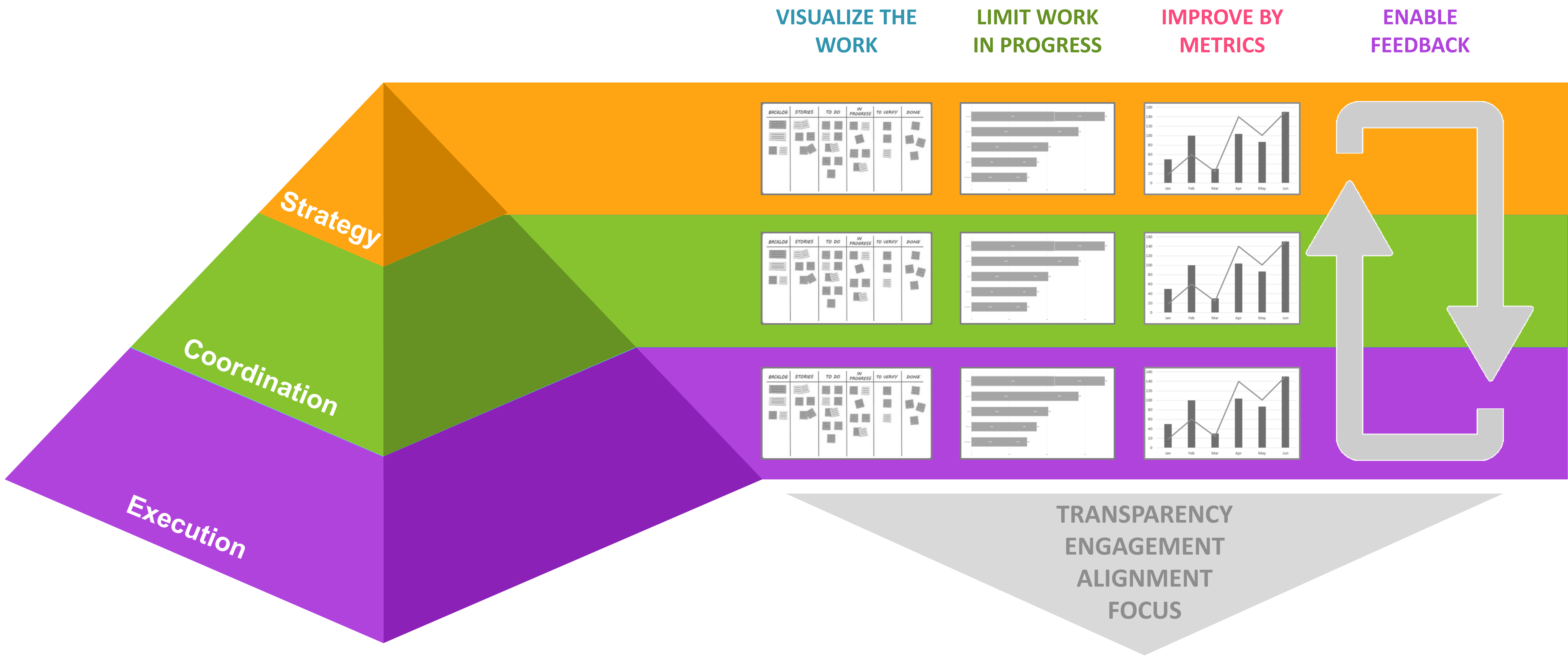
Team is pushed by someone/something
Defective sprint planning
Sprint backlog not protected
Internal and external dependencies

Poor Product Ownership
Lack of a shared vision
Real stakeholders not involved
Inattention to good design

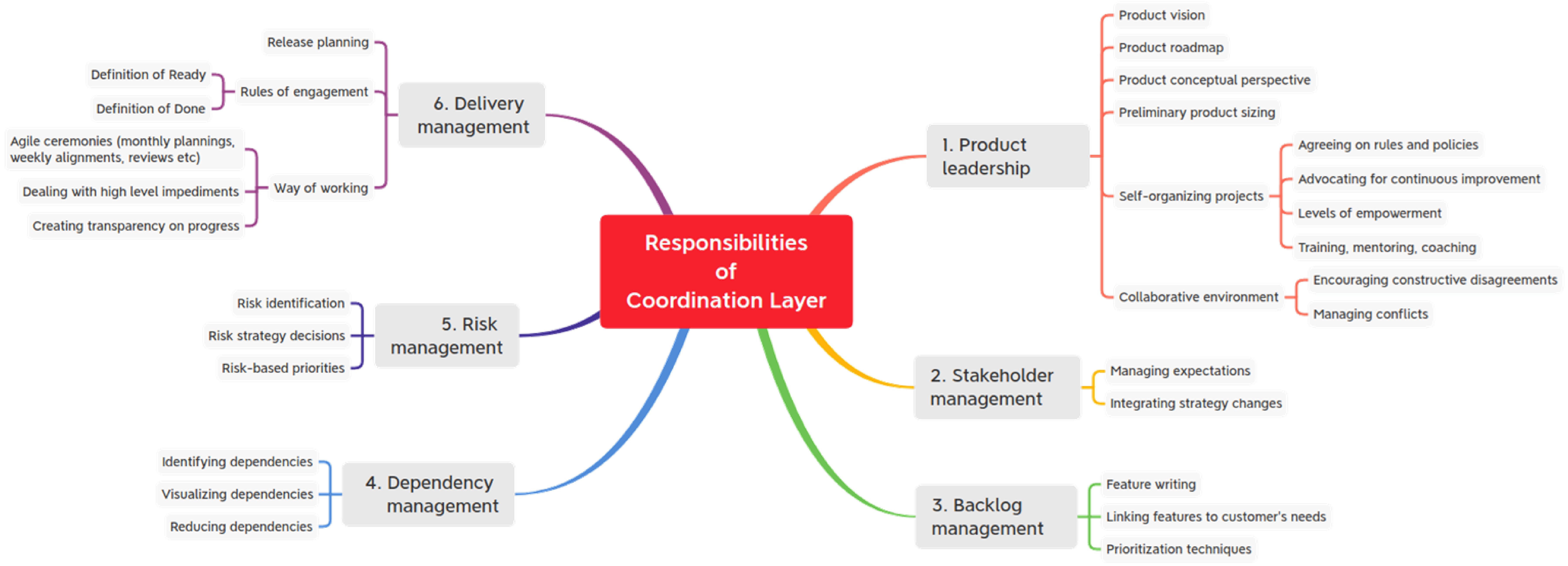
Excessive changes (affecting budget and time)

	Poor product ownership	Lack of a shared vision	Real stakeholders not involved	Inattention to good design
CLASSIFYING	Always an issue	Always an issue	Usually an issue, sometimes a constraint	Always an issue
DIAGNOSING	Inspect project backlog Discuss with stakeholders	Inquire team members Examine PO-team alignment	Monitor decision making process	Create a refactoring backlog Monitor refactoring needs
SOLVING	Coach the PO Get support for PO	Reiterate project goals Create project visual maps	Get real decision makers on board	Get support from architects Create solution architecture
ADAPTING	Don't adapt, solve it!	Don't adapt, solve it!	Implement pseudo dual track (prototype-develop)	Don't adapt, solve it!

Project governance

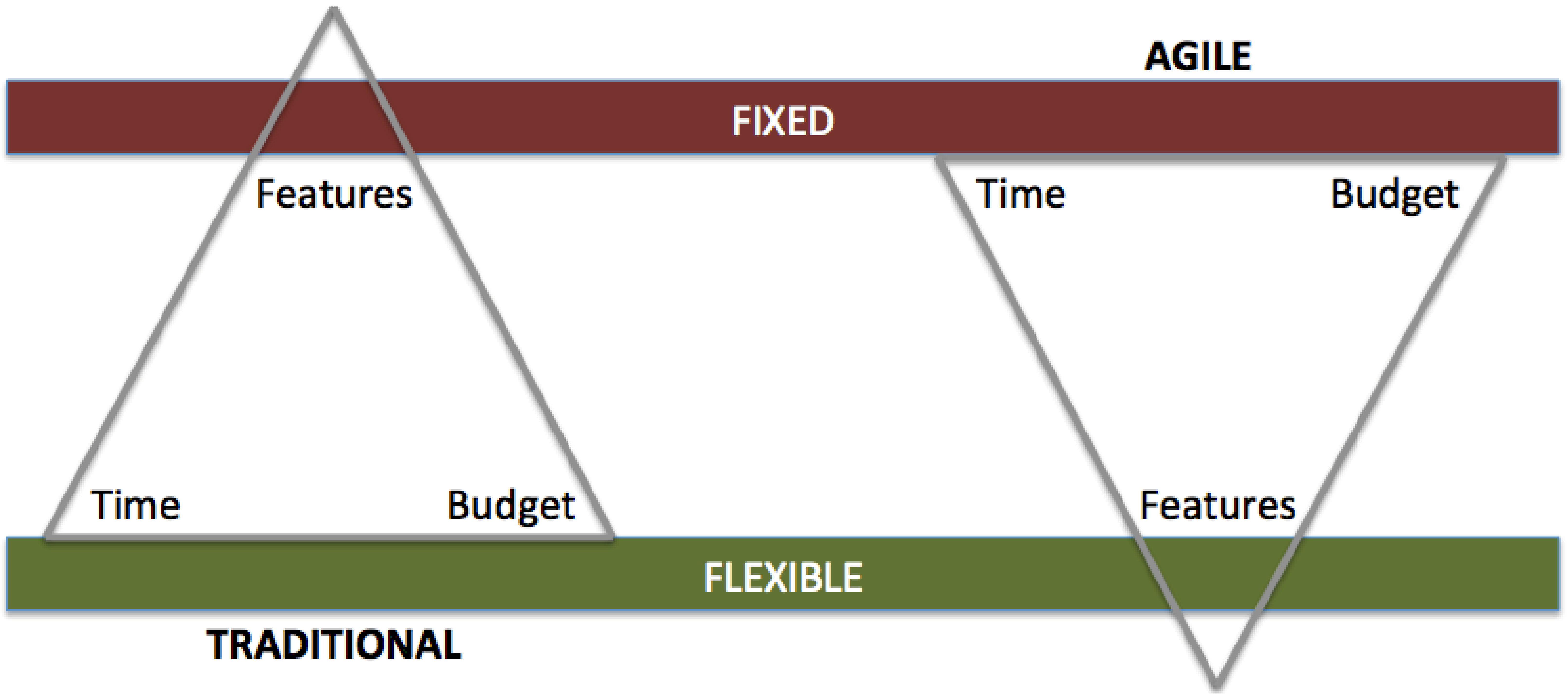


Coordination performance


Responsibilities of Coordination Layer
1. Product leadership
Product vision
Product roadmap
Product conceptual perspective
Preliminary product sizing
Self-organizing projects
Agreeing on rules and policies
Advocating for continuous improvement
Levels of empowerment
Training, mentoring, coaching
Collaborative environment
Encouraging constructive disagreements
Managing conflicts
2. Stakeholder management
Managing expectations
Integrating strategy changes
3. Backlog management
Feature writing
Linking features to customer's needs
Prioritization techniques
4. Dependency management
Identifying dependencies
Visualizing dependencies
Reducing dependencies
5. Risk management
Risk identification
Risk strategy decisions
Risk-based priorities
6. Delivery management
Release planning
Rules of engagement
Definition of Ready
Definition of Done
Way of working
Agile ceremonies (monthly plannings, weekly alignments, reviews etc)
Dealing with high level impediments
Creating transparency on progress

Agile Contracting

Agile Contracting



Agile Contracting

- Traditional contracts contains:
  - Fixed scope
  - Firm estimates

↓

  - Inflated estimates
  - Not all specs bring value

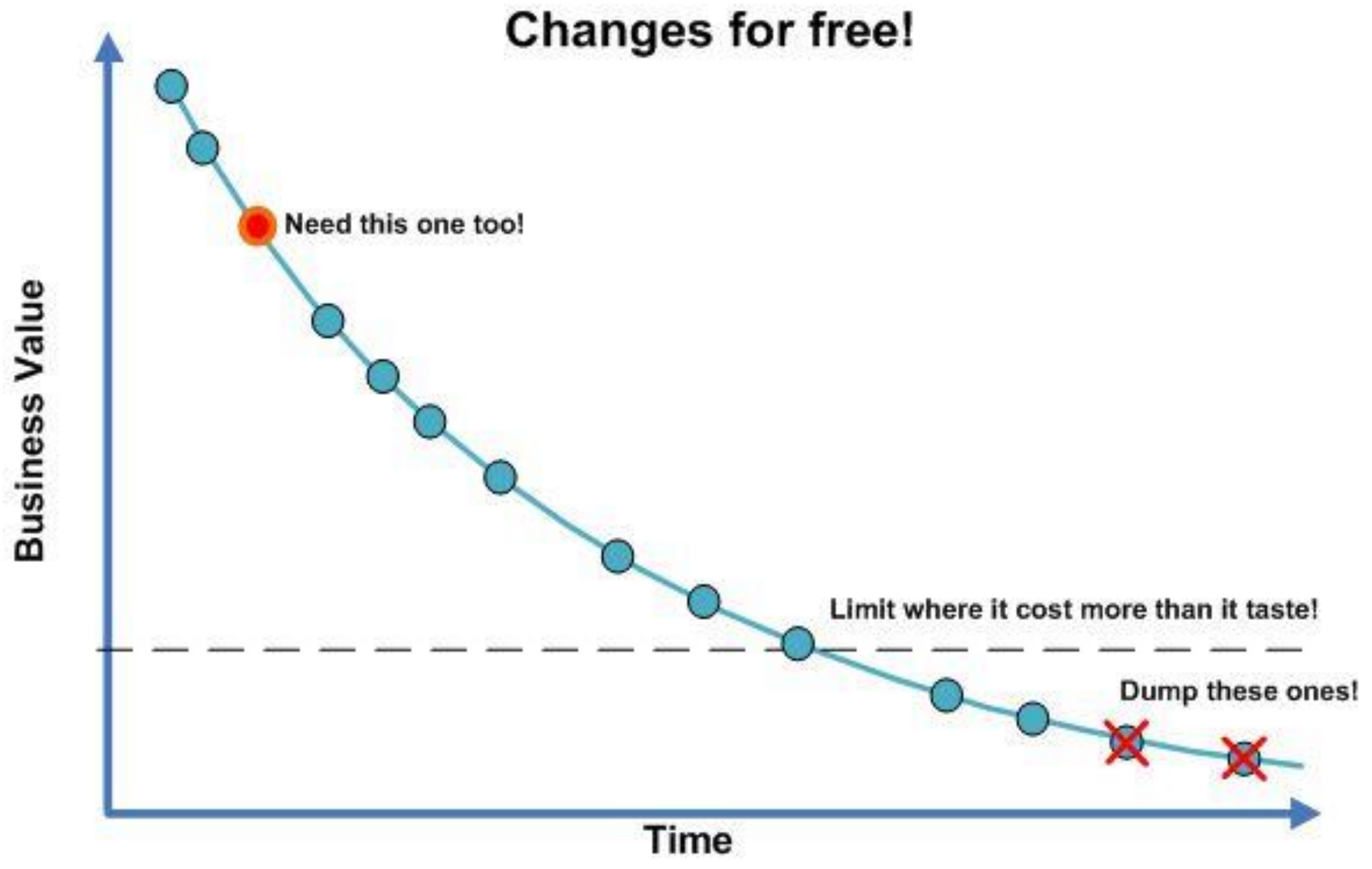
Agile Contracting

- DSDM Contract
- *Money for Nothing* and *Change for Free*
- Graduated Fixed Price Contract
- Fixed Price Work Packages
- Customized Contracts

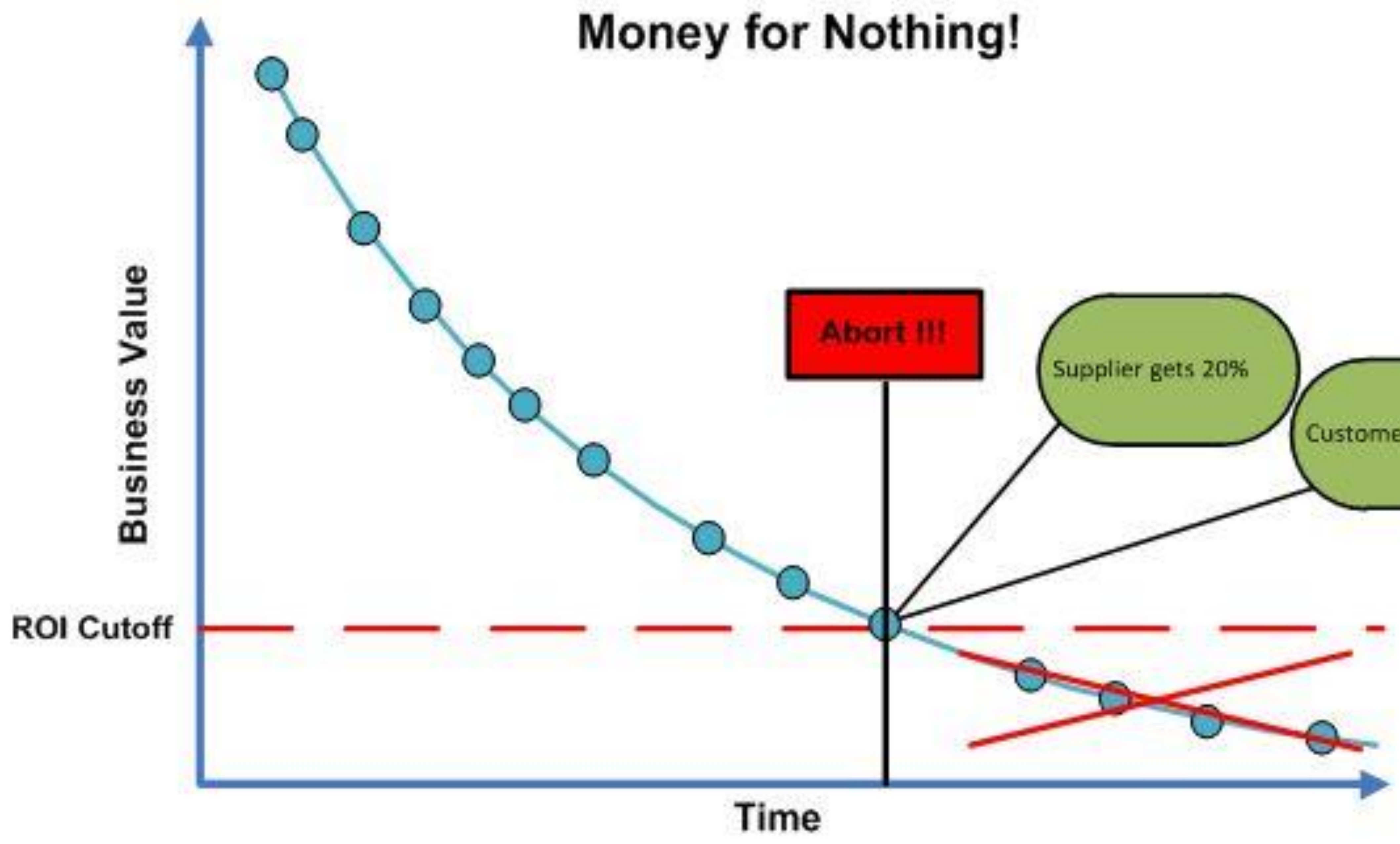
DSDM Contract

- focused on work being “*fit for business purpose*” and passing tests rather than matching a specification

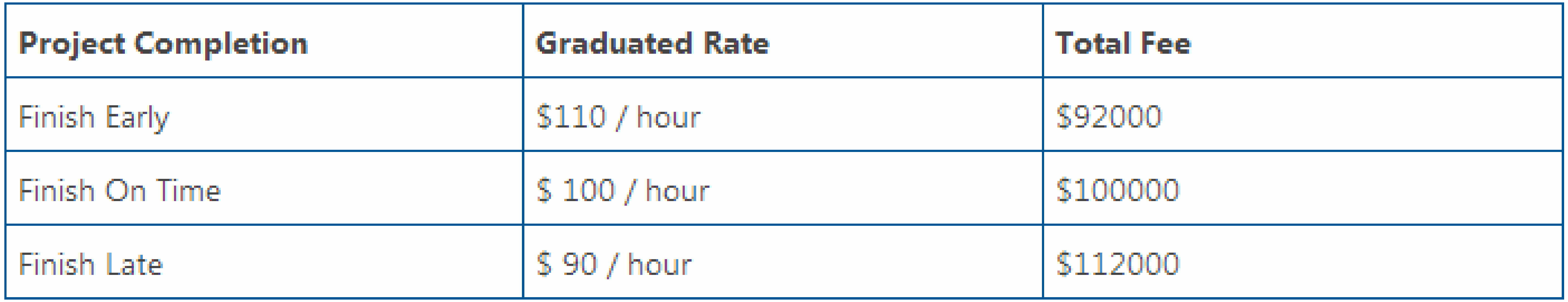
Money for Nothing and Change for Free



Money for Nothing and Change for Free

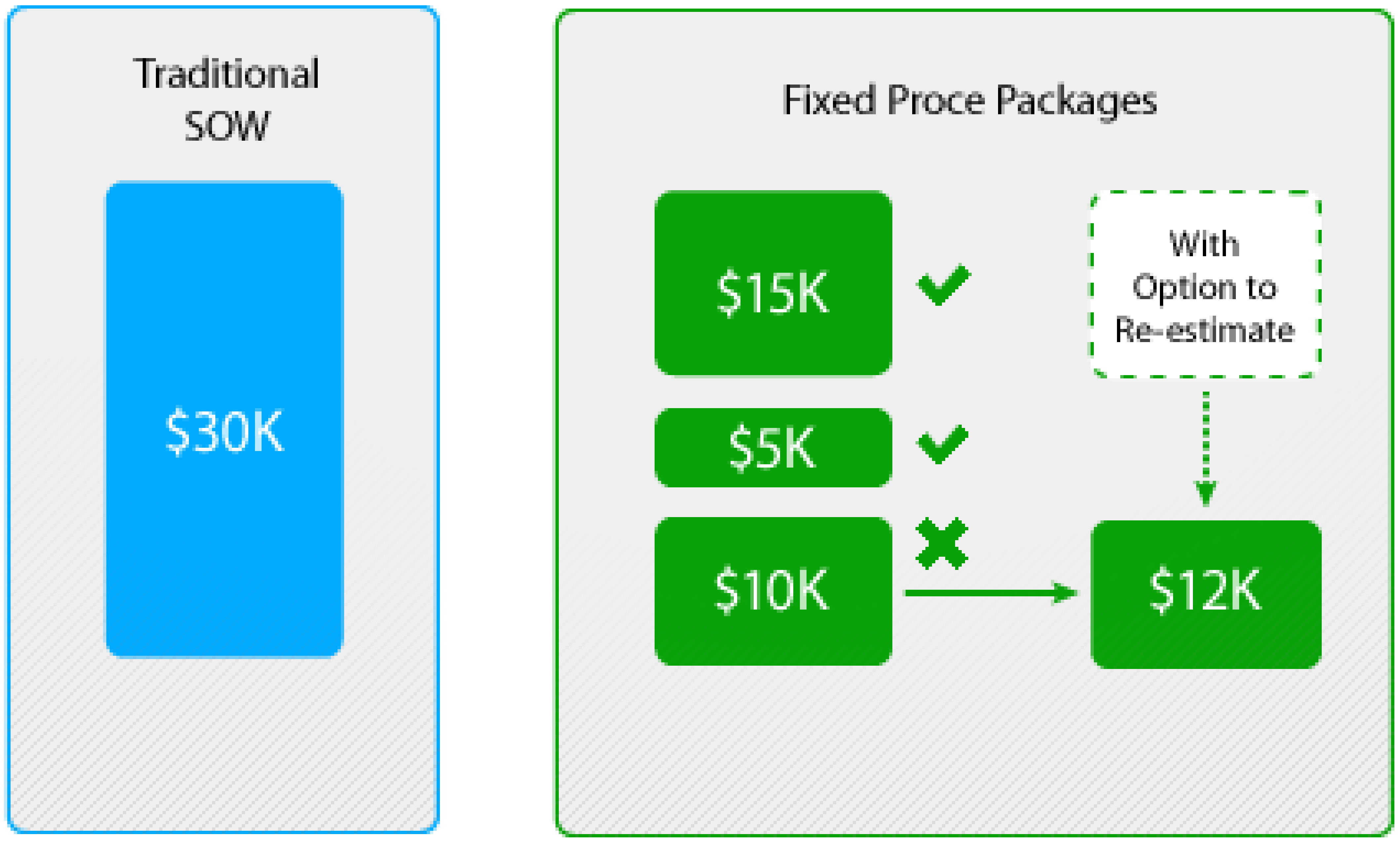


Graduated Fixed Price Contract



Project Completion	Graduated Rate	Total Fee
Finish Early	$110 / hour	$92000
Finish On Time	$ 100 / hour	$100000
Finish Late	$ 90 / hour	$112000

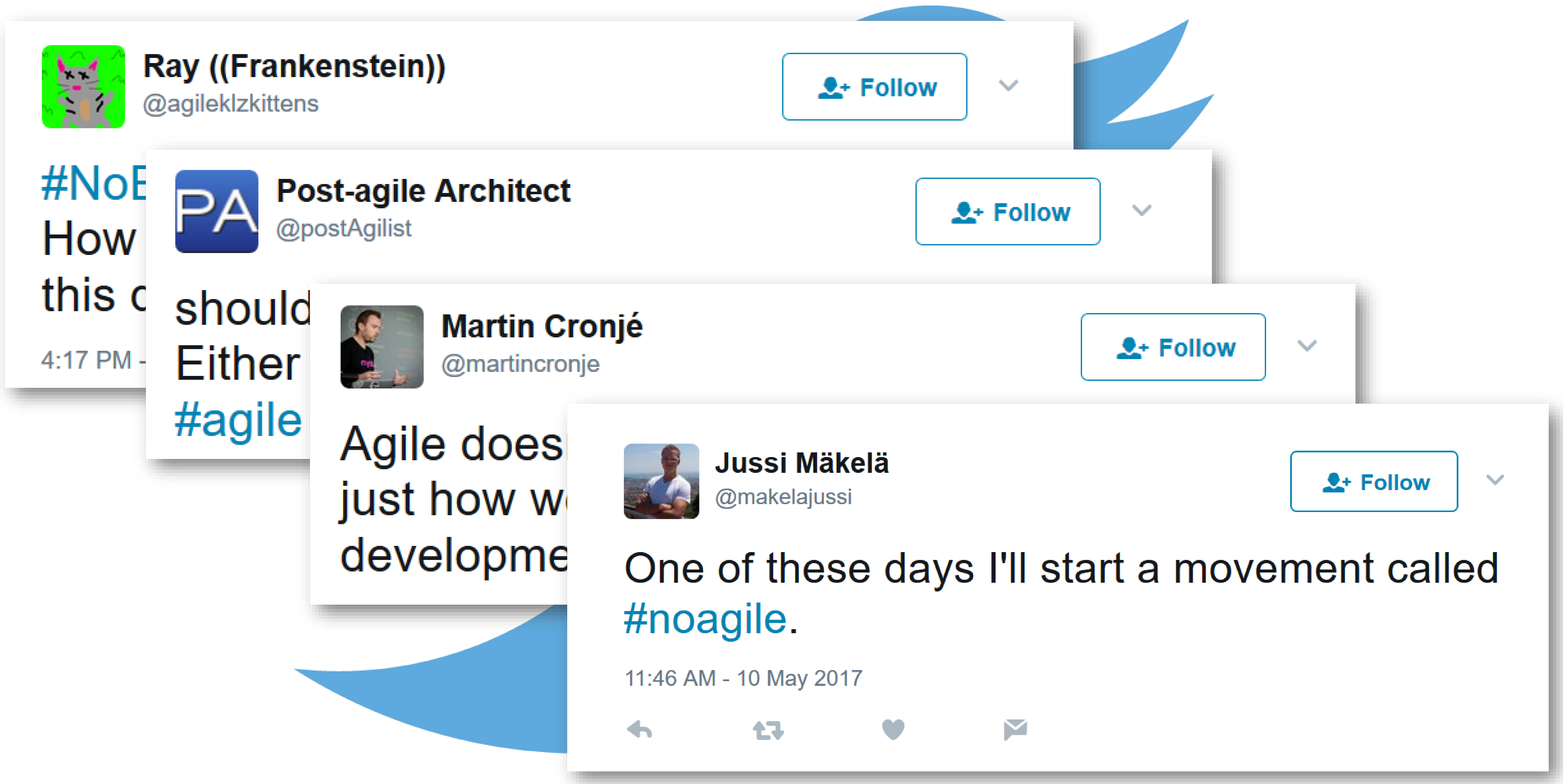
Fixed Price Work Packages



#NoAgile

2017

Estimates
Projects
Backlog
Iteration
Release



Ray ((Frankenstein))
@agileklzkittens

#NoE
How
this c

4:17 PM -

Post-agile Architect
@postAgilist

should
Either
#agile

Martin Cronjé
@martincronje

Agile does
just how w
developme

Jussi Mäkelä
@makelajussi

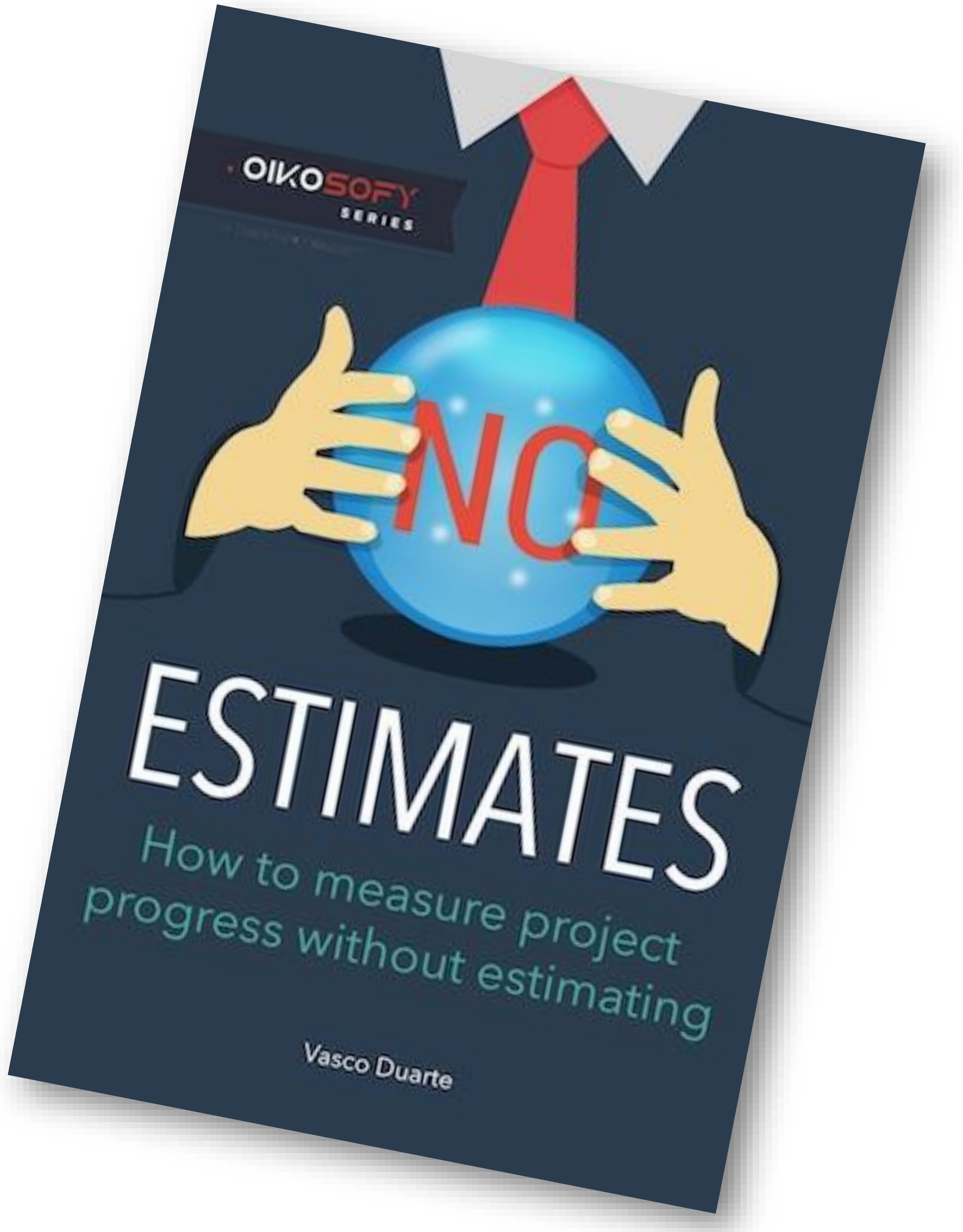
One of these days I'll start a movement called #noagile.

11:46 AM - 10 May 2017

#NoEstimates

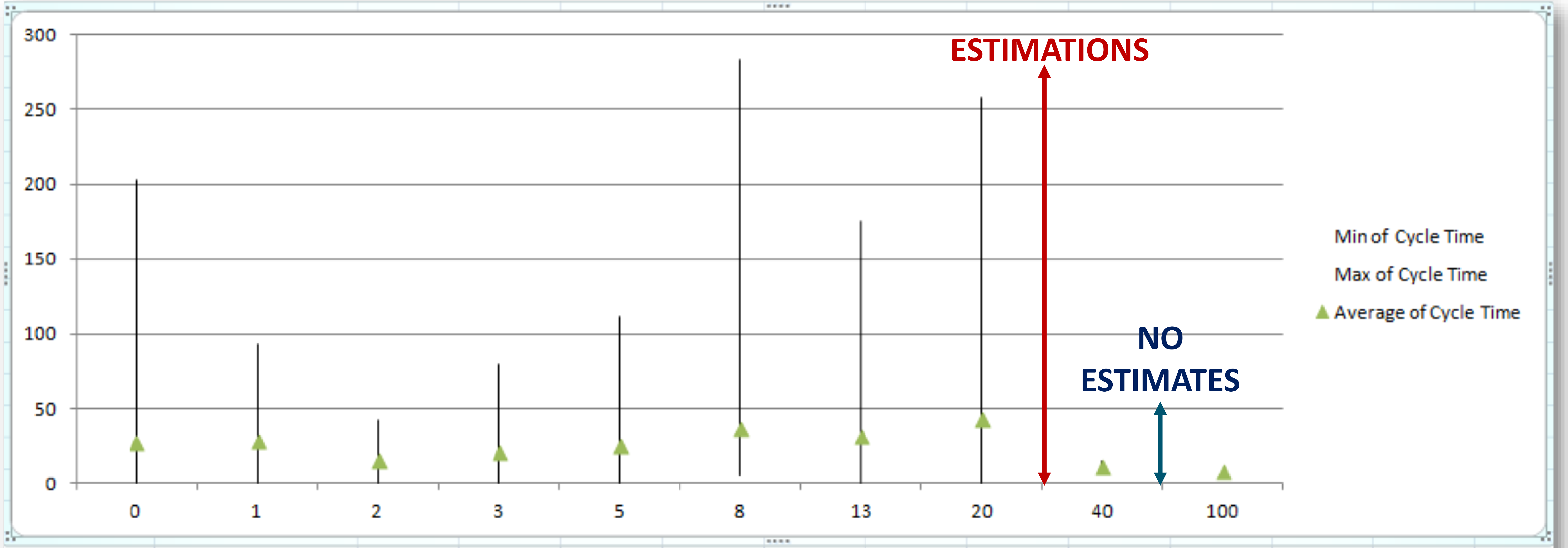
Woody Zuill

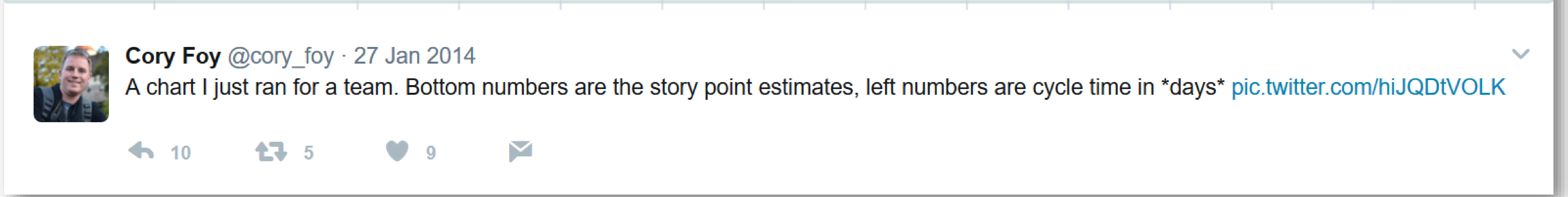
Vasco Duarte



This guy is a software engineer, you can tell by his awesome estimation skills

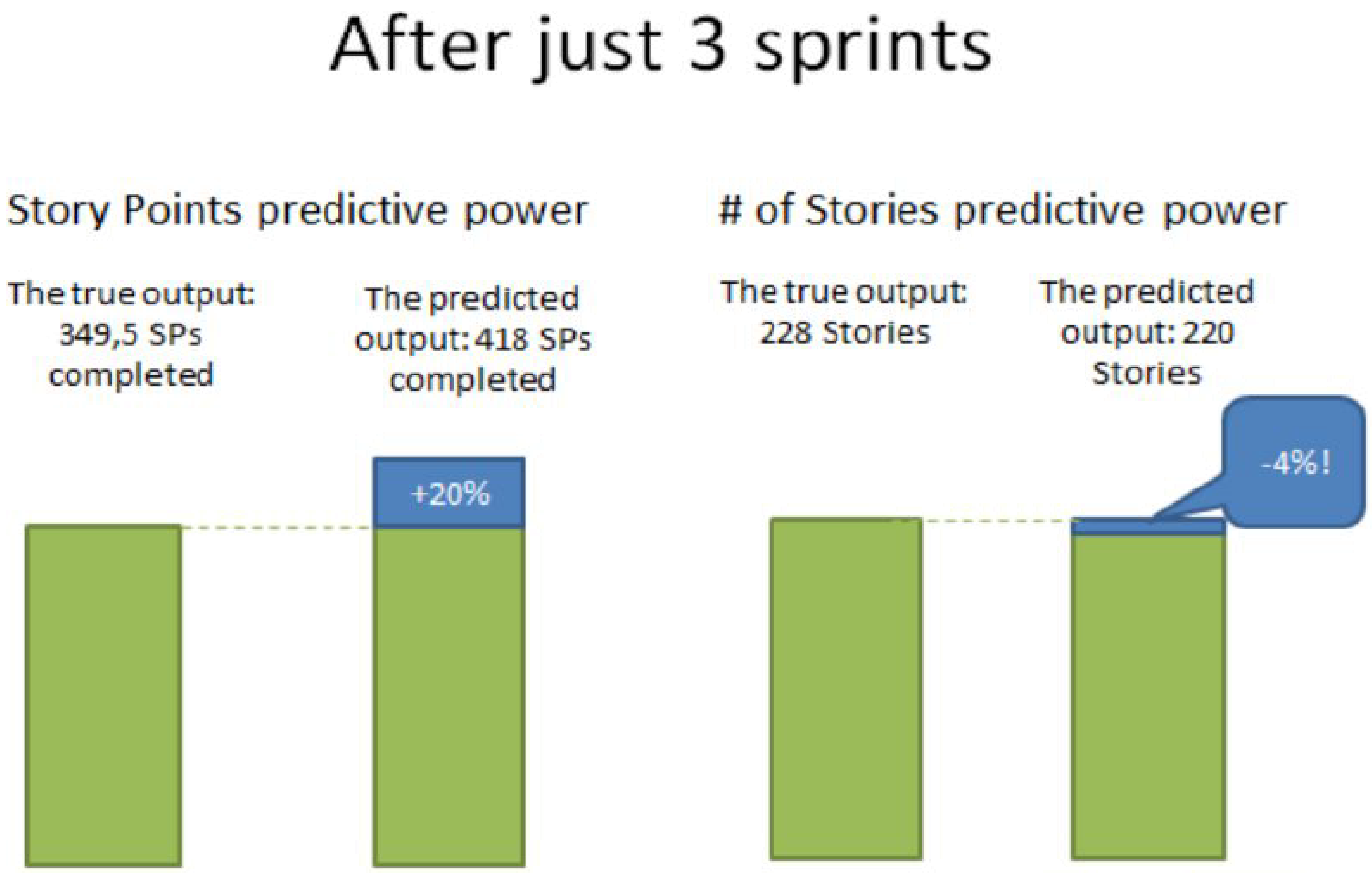
#NoEstimates




Cory Foy @cory_foy · 27 Jan 2014

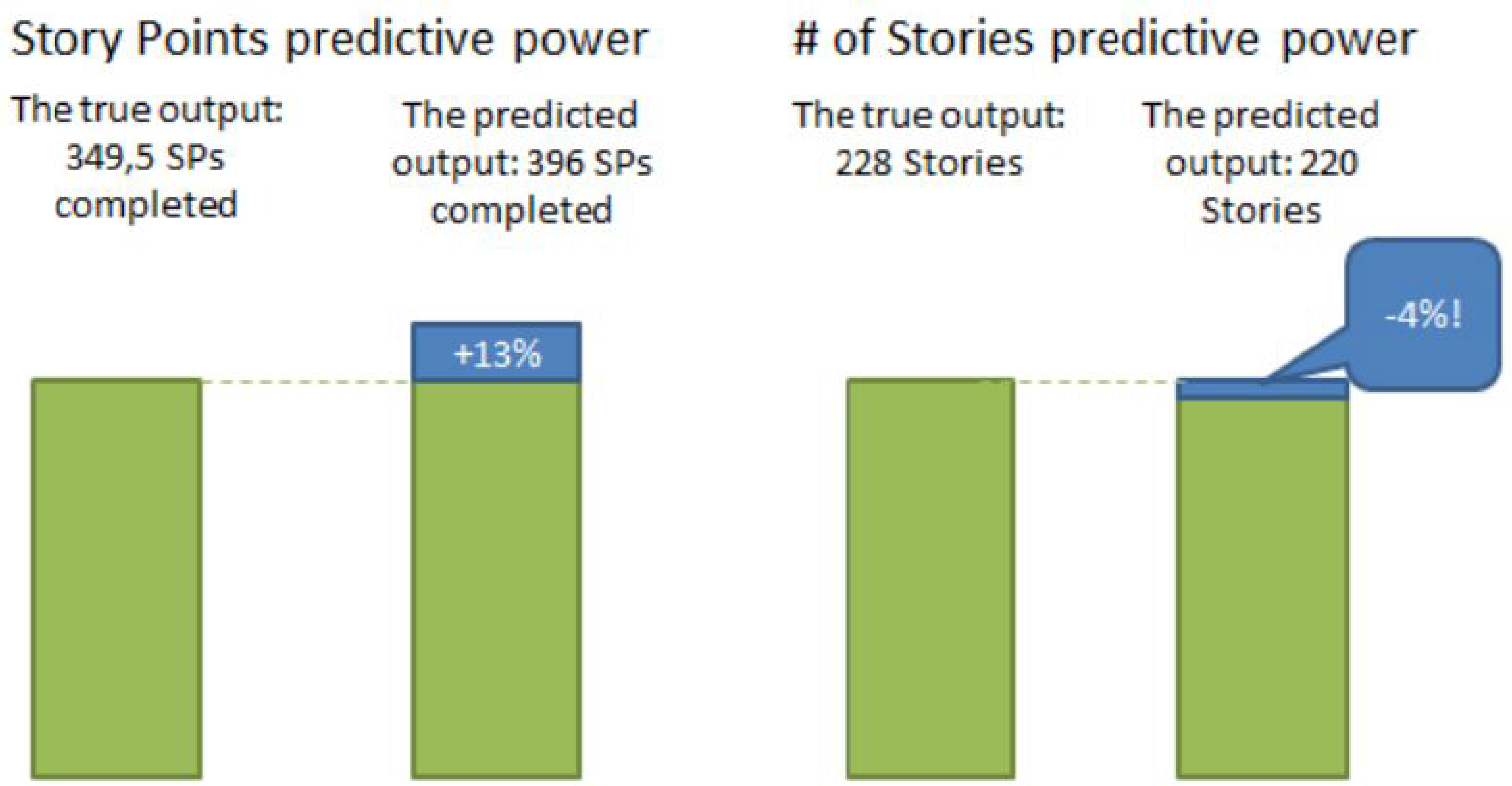
A chart I just ran for a team. Bottom numbers are the story point estimates, left numbers are cycle time in *days* pic.twitter.com/hiJQDtVOLK

10 5 9



#NoEstimates Whitepaper by Vasco Duarte

After just 5 sprints



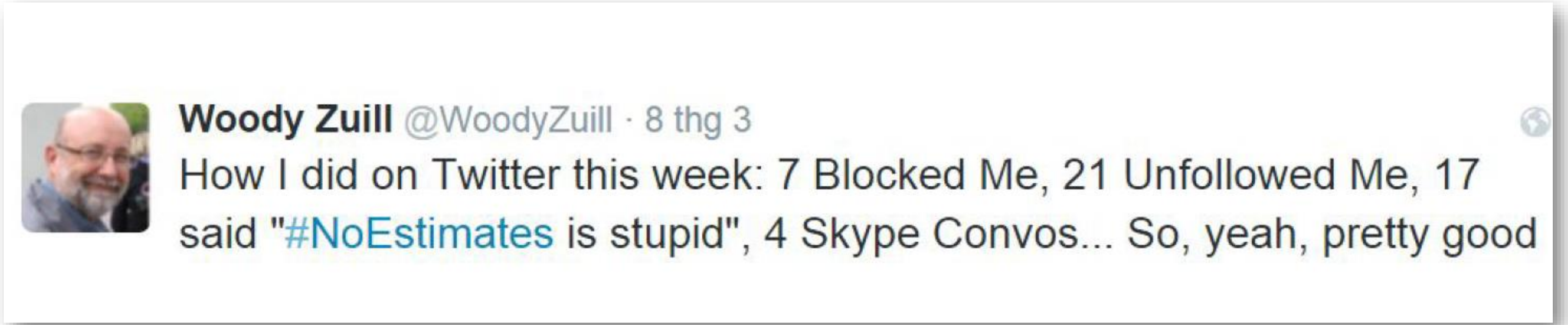
#NoEstimates Whitepaper by Vasco Duarte

#NoEstimates

“How to stop having consistently late projects?”

“*Start them sooner!*” (Jim Highsmith)

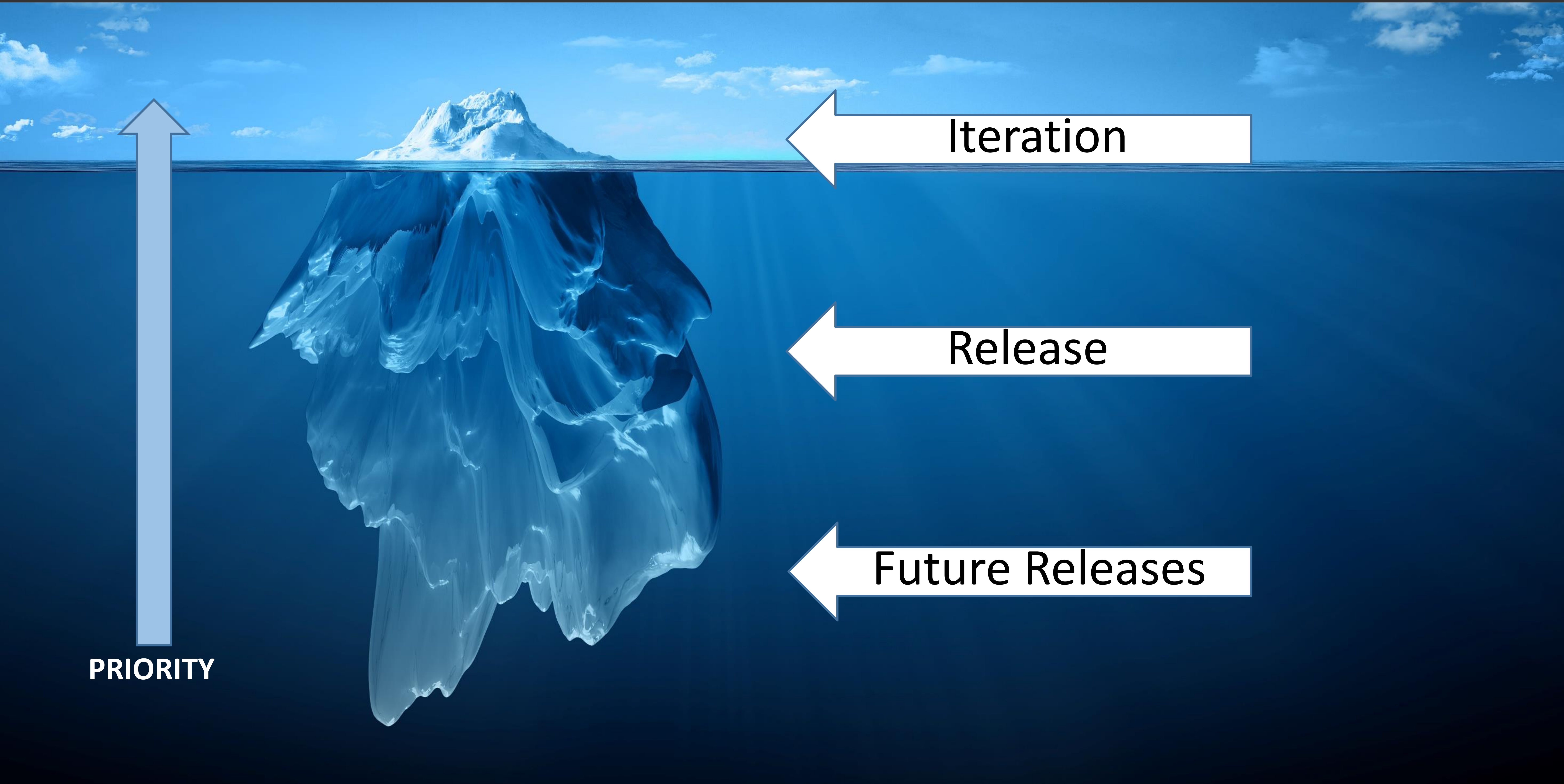
#NoEstimates



Woody Zuill @WoodyZuill · 8 thg 3

How I did on Twitter this week: 7 Blocked Me, 21 Unfollowed Me, 17 said "#NoEstimates is stupid", 4 Skype Convos... So, yeah, pretty good

#NoBacklog



#NoBacklog

“Most backlogs are waste.
Estimating backlog items is therefore super-waste.
Revisiting backlog estimates are in
mentally-deranged territory”

Paul Klipp