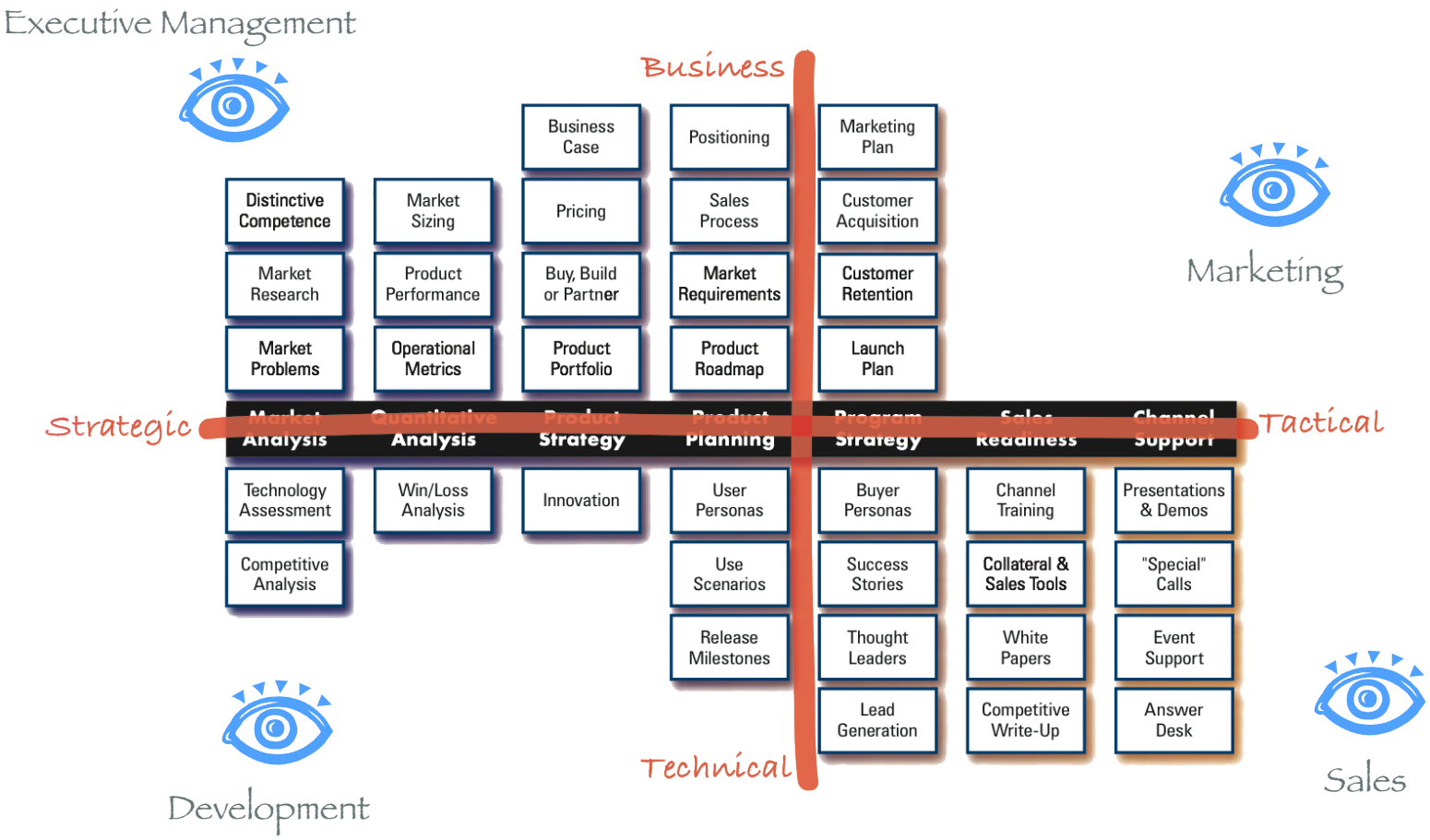
Owning An Agile Project: PO Training Day 2

Petri Heiramo
Agile Coach, CST

Product Management ≠ PO

- Product management is a larger scope than what Scrum defines as a PO
- Or rather, Scrum implicitly assumes that a PO has awareness of the other areas, but defines responsibilities only regarding the control process

Product Manager Responsibilities



Different Options

- Product manager as PO
  - Usually suitable when only one or two teams
- Product Owner pair
  - Business PO to interface with market and users
  - Technical PO to interface team and development
  - Collaborative management of the product and prioritization
  - Can probably manage a couple of teams
- Product Manager, with subordinate PO's to lead teams
  - Often necessary when more than three teams

Ultimately...

Product management is about managing the market attractiveness of the product and seeking maximum long term profits from the product.

Project as a Part of A Larger Whole

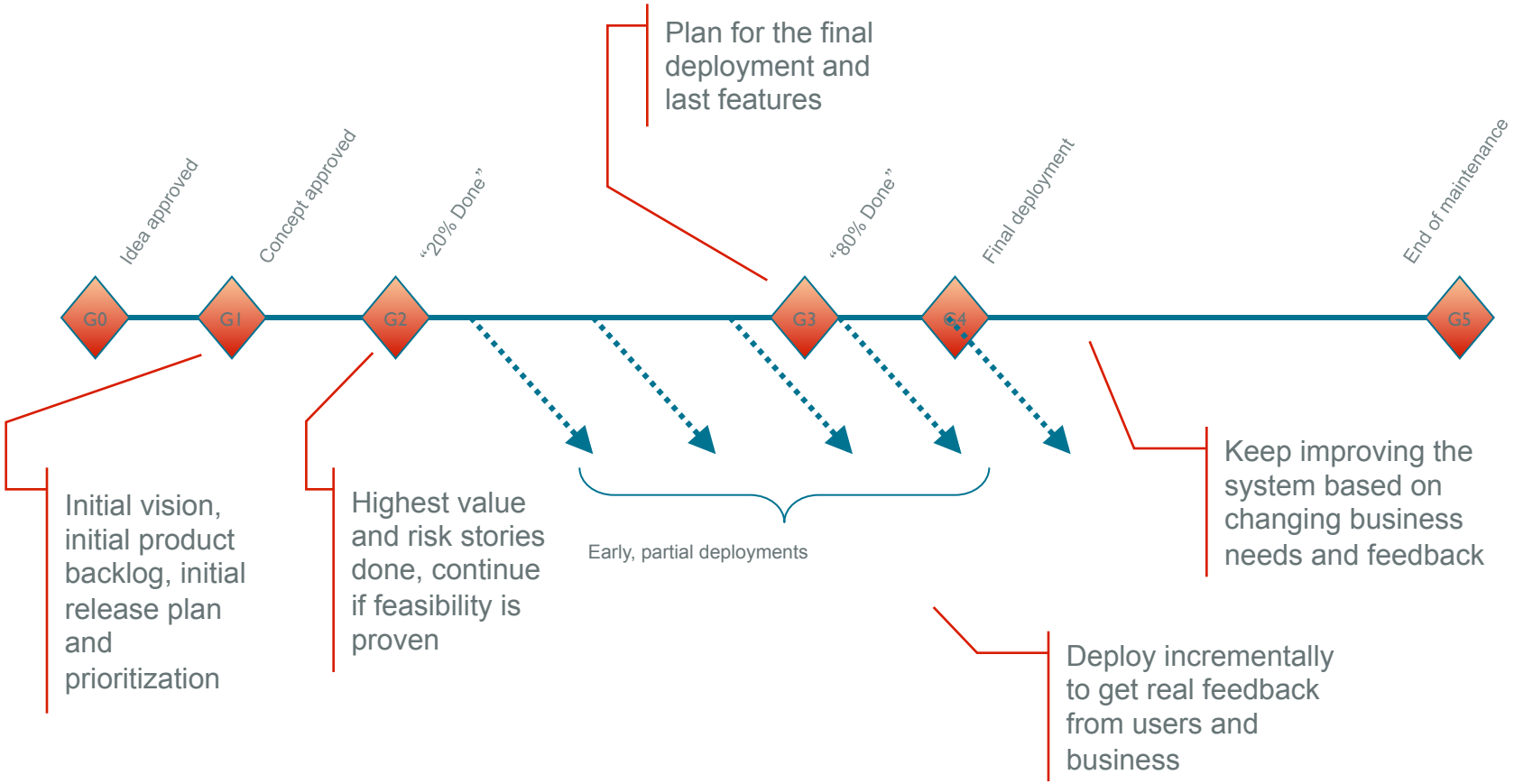
The Grander Scheme of Things

- The “Agile practices” work in the development space
  - XP, Scrum meetings, user stories, story points, etc.
- “Being Agile” affects everything
  - How do we, as an organization, work so as to enable efficient and valuable work being done in the projects?

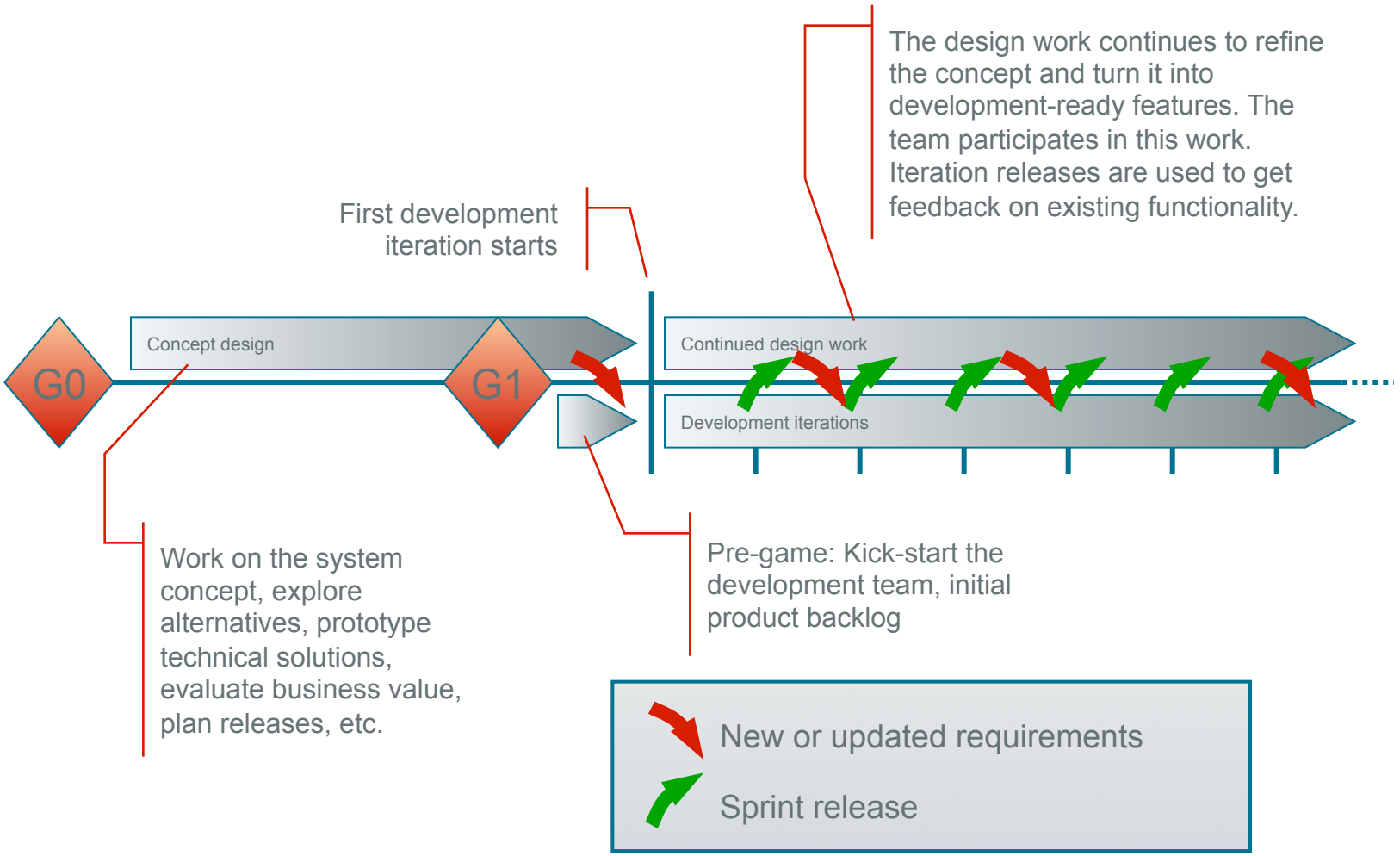
An Agile Project Tied to Larger Environment

- Can be adapted e.g. as a part of traditional stage gate model in product development
  - Some adaptations needed for the gate approval criteria
- An Agile team can be part of a continuous product improvement
  - E.g. development ideas from from ITIL change management and the PO responsible for the product portfolio prioritizes the changes

Agile Gate Model



Product Management

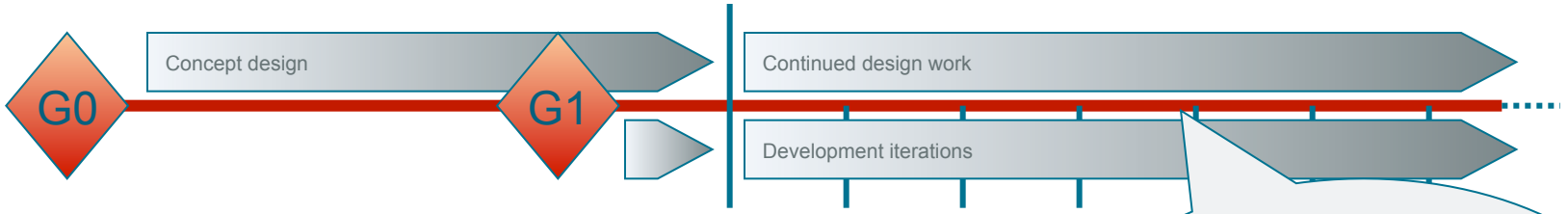


Research vs. Development

- Clear distinction
  - Research: Finds out information about something, does not deliver any system functionality
  - Development: Delivers system functionality based on defined user stories, Running Tested Features
- Don't commit to development without "ready" user stories
  - The team must have a good idea of what to do, how to do it, and be able to estimate effort

Above This Line...

Research



Development

A typical Agile team will work on both sides of the line. Additionally, there are often external specialists supporting PO in the research part.

Impact of Agility on Others

- Direct or ripple effects to all operations
  - Sales, HR, product management, IT, etc.
- Scrum reveals dysfunctions in the organization
  - The reaction to identified dysfunctions will define how Agility can succeed in the company
- For real corporate wide benefit from Agile, remove identified dysfunctions

Essential

- Quality and Agile principles must not be compromized
  - Except temporarily and then paid back
- Operation must be guided with capacity and prioritization, not in a requirements driven way
- Requirements and needs channeled to the team through one PO
  - The business (and not the team) must take the responsibility for sorting out conflicting interests

Project Vision

Project Vision

- The vision defines the purpose of the project's existence
- Provides the base for defining business value and priorities
- Communicates the purpose to the Scrum team and stakeholders
- Hopefully an uplifting purpose to create commitment for shared goal
- If the vision cannot be written down, does that mean that the project is lacking purpose?

Two Basic Elements

- Public statement or description
  - What is the project trying to achieve? What are key priorities? Who are the most important user groups or stakeholders? What do they value?
  - Often part of the product concept
- High-level project plan and goals
  - Duration of the project and important milestones/ releases
  - General goals and feature priorities at high level
  - Initial release plan
    - Key features tied to releases and schedule
    - Central dependencies to external parties

Creating a Vision

- Different tools for creation
  - Vision box exercise
  - Elevator pitch creation
  - Press release exercise
  - Magazine review exercise
  - One page data sheet
  - Personas and scenarios
- Include appropriate stakeholders, users and maybe team, too

Exploring, Writing Down and Managing Requirements

Product Backlog in Summary

- Contains all the work needed for delivering the product
- Two key differences to traditional requirements specification
  - Product Backlog is not expected to contain all planned features of the product
  - All the items recorded in the Product Backlog are not expected to be incorporated in the product
    - In an individual project such expectations can be made, but not generally
- Owned by Product Owner
  - Information in Product Backlog can come from many sources
  - Final prioritization only from PO

Collecting Requirements

- Initial Product Backlog
  - One of the purposes of the ”pre-game” phase is to define the initial Product Backlog at sufficient detail
  - Different approaches
    - Concepting projects, workshops, questionnaires, research, etc.
  - Necessary for starting the development work
- Updated during development
  - Different sources, such as
    - Feedback from users and stakeholder groups
    - Ideas or dependencies found by the team
    - Changes in business environment
  - Requires initiative, especially towards users and stakeholders
  - ”Add to Product Backlog, ask team to estimate, prioritize”

Defining Requirements

- Necessary detail is project dependent
  - Are there contractual or legal requirements to meet?
  - How good is the communication within the project?
  - Are there some particularly complex requirements?
- User story (or just ”story”) = something that someone can do with the product
  - Big stories (sometimes called ”epics”) are often similar to use cases (except for the level of formality and detail), small ones are like scenarios or exception cases
  - Can also be non-functional requirements, like performance or quality requirement
  - “User can update personal details for the account”
- Leading the definition work is one of the central responsibilities of the PO
  - Using the expertise of users, stakeholders and software development professionals is critical for real success
- Early on, stories can be high level and lack detail, but they need to be refined before the team can commit to them in an iteration

Themes

- Themes are collections of related stories
  - By feature
  - By function, e.g. security, API intensive
- Higher level planning, prioritizing and valuation is often easier on themes
- Use as you like really ☺

INVEST

- I = Independent
- N = Negotiable
- V = Valuable
- E = Estimatable
- S = Small
- T = Testable

- Epics can be anything, but you shouldn't put non-INVEST stories into iterations

Spikes

- Spike = short research on issue or technological challenge
- Very useful in…
  - Testing something new and unknown
  - Splitting up big stories
  - Evaluating and comparing alternatives
  - Estimating the size of highly complex stories
- Spike itself should be as small as possible
  - Purpose is to learn, not to specify
- Avoid multiple spikes to an iteration
  - Always some real development work is needed
- Always plan a result for a spike
  - Split up a story, estimate size, make a decision, etc.

Documenting the Requirements

- Avoid overly heavy solutions or tools
- Allow some flexibility in solution
  - Avoid defining UI or technical details; focus on what a user can achieve with the system
- The system should allow definitions at different levels of detail
- Necessary information
  - Description
  - Cost or size estimate
  - Priority (indicated by the position in the product backlog)
  - Acceptance criteria (when the story is going into an iteration)

Defining Business Value

Business Value

- Typically hard to enumerate
  - In individual cases, it may be possible to assign monetary value (gain or savings)
  - Often subjective or otherwise not measurable in money
    - E.g. happiness of own employees
- Often most useful at theme level
- Options
  - Money
  - Relative value
  - Relative comparison
  - Point systems
  - Buying games
  - Desirability (Kano model)
  - Subjective valuation
- Useful methods strongly dependent on project

Money

- Typically the most difficult, but when workable, gives clear valuation
  - Direct benefit/cost evaluation
- Often requires a financial model
  - Estimated revenue over time, estimated costs, interest
  - Effect of different release times to revenue expectations
- Works with both effort and relative estimation

Relative Values

- Often easy to use
  - Gives a relatively reliable estimate
- Typical process
  - Decide the scale (e.g. 1-5, 1-10 or fibonacci series)
  - Estimate the relative value for each feature (compared to other features)
  - Compare to cost estimates given by the team
    - Value / Cost

Relative Comparison

- Different methods for comparing features against each other
- Attempts to quantify and summarize subjective estimates to comparable values
- Examples
  - Theme screening
  - Theme scoring
  - Relative weighting

Point Methods

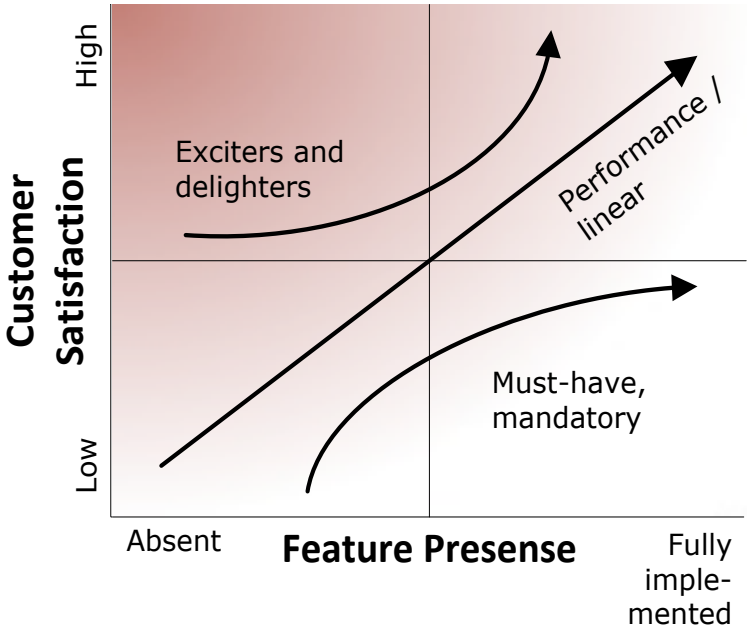
- Give each participant either points or ”money”
  - Either equally or different amount depending on stakeholder influence
- Participants can distribute their points as they wish on product features
  - More points == more value for that stakeholder
- Can be done in one session with multiple stakeholder present, or in separate sessions
- The business value equals the number of points on the item

Buying Games

- Each participant receives some ”money”
  - Either equally or different amount depending on stakeholder influence
- Participants can use their money to buy product features (against cost estimates given by the team)
  - It is allowed to collaborate on larger or more desired features
- For prioritization purposes, only give enough ”money” to buy a couple of iterations worth of work
  - Return to the game regularly to ”buy” more features
- Features purchased now have high priority, others stay at low priority

Desirability (Kano Model)

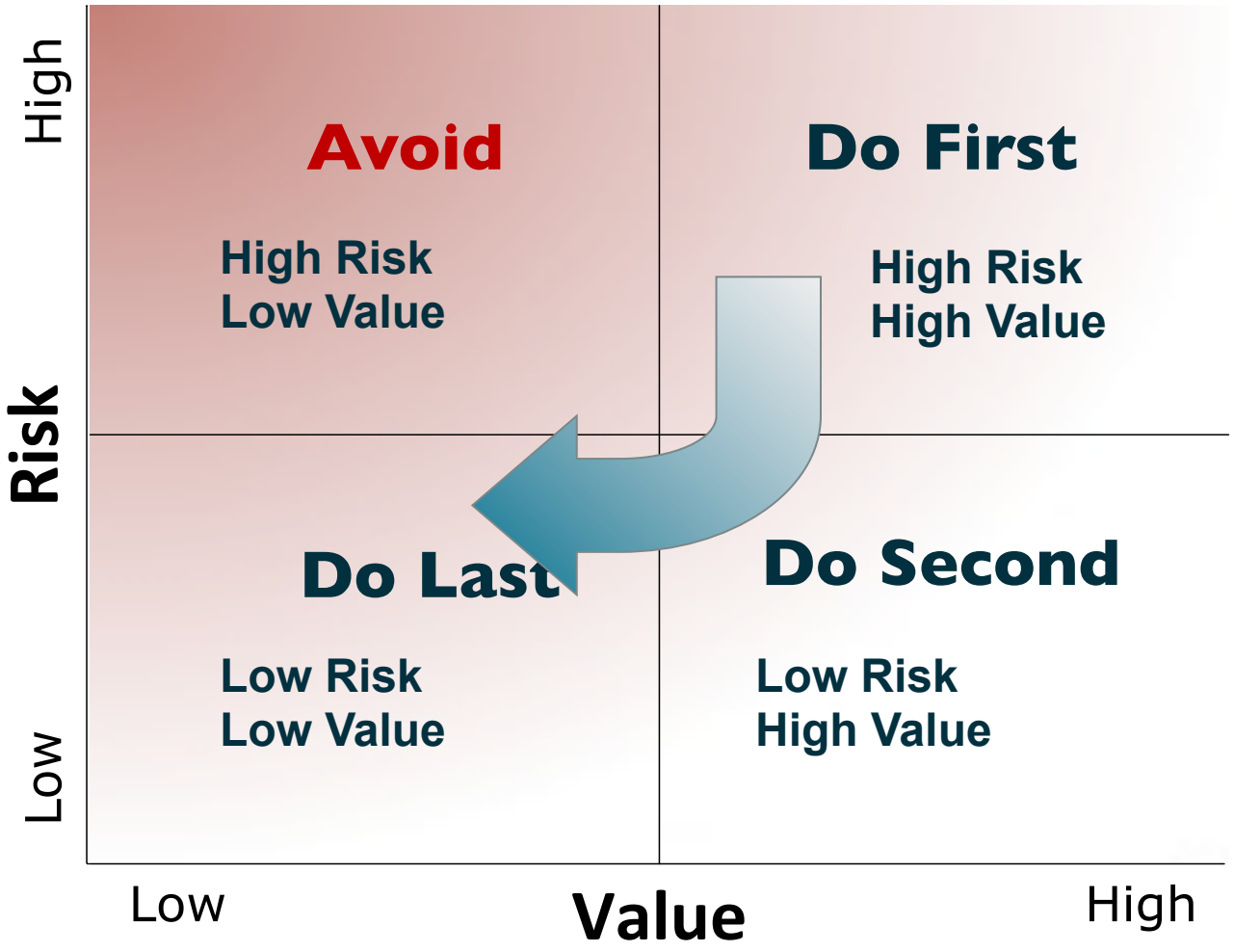
- Utilizes Kano model for feature desirability
  - Requires making of a certain type of customer questionnaire study
- Results can indicate features which belong to three interesting categories
  - Threshold features (mandatory)
    - Not having a feature can result in market rejection
  - Exciters
    - Customer does not expect these features, but their presense creates excitement even when implemented with limited functionality
    - Typically new features in the market, and costly to develop
  - Performance features
    - The better the feature is implemented the happier the customer is
- Prioritization
  - All threshold items must exist, at least at satisfactory level
  - As many performance features as possible, as well done as possible
  - A few exciters to differentiate and attract buyers



Subjective Valuation

- Subjective evaluation to value order or categories
- Prioritized into one list so that the most valuable items are at the top
- Important to factor in risks and dependencies
  - The team must contribute to this

Risk and Value in Prioritization



Important

- Business value is a model for value
  - Represents how the business values particular features
- No model gives complete answers
  - PO and stakeholders should use that information, such as risk and dependencies, to create meaningful plans

Managing Releases, Scope and Details

Focus of Agile Planning

- Strategy
- Portfolio
- Product
- **Release**
- Sprint
- Day

Agile focus is on planning at product, release and iteration levels

Daily planning is team's concern

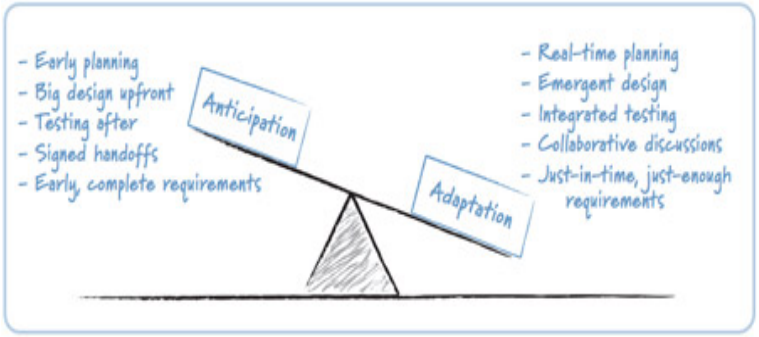
"Often detail adds no more usefulness – only a false appearance of validity."

– Edward de Bono

Why a Release Plan?

- Most stakeholders and financiers want to see (out of common sense) an overall plan
  - When this is expected to be ready
  - What does it cost
- The project needs a release plan for directing its own work
  - It's the big picture
- The first version should be part of project vision

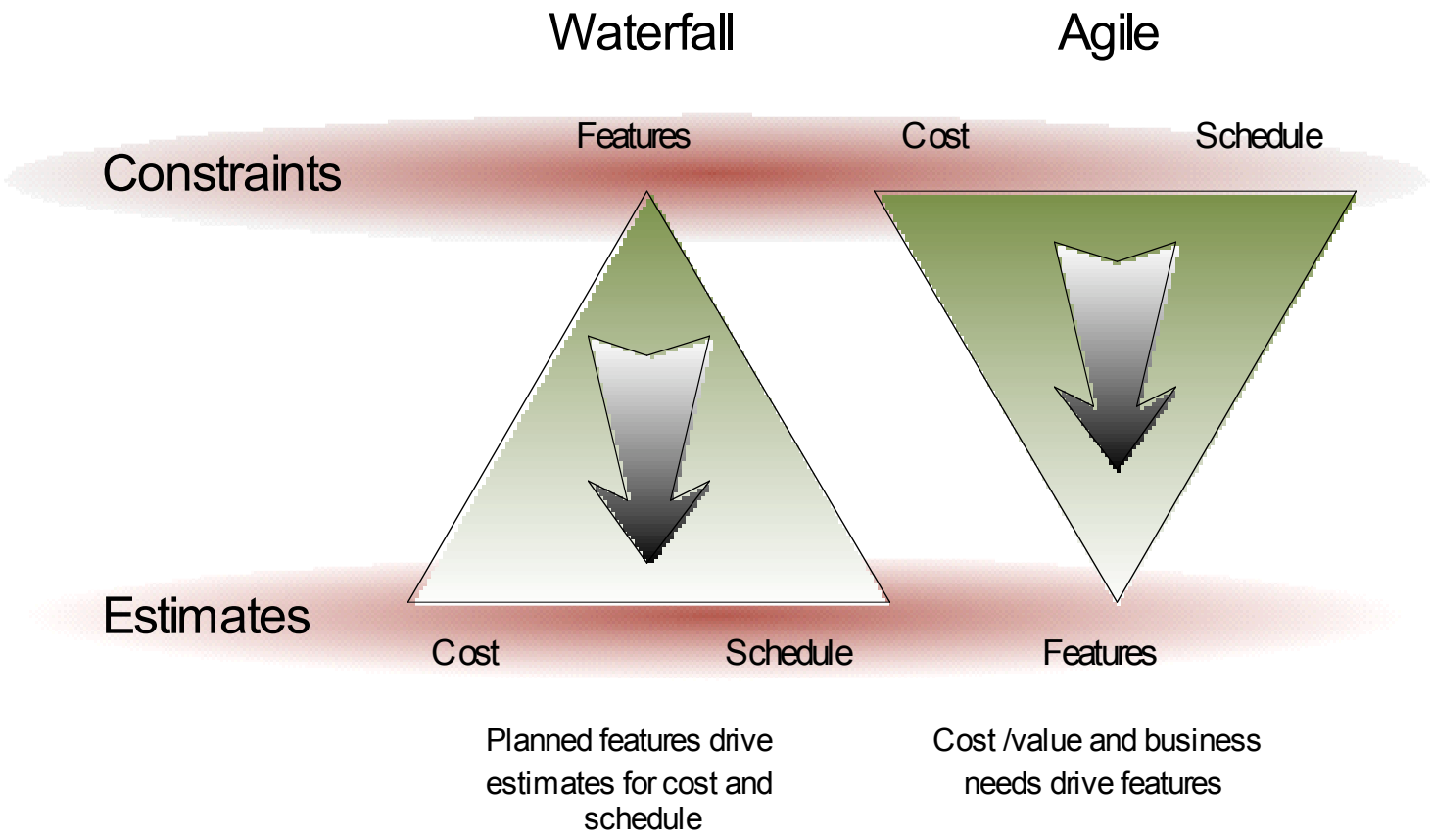
Finding Balance



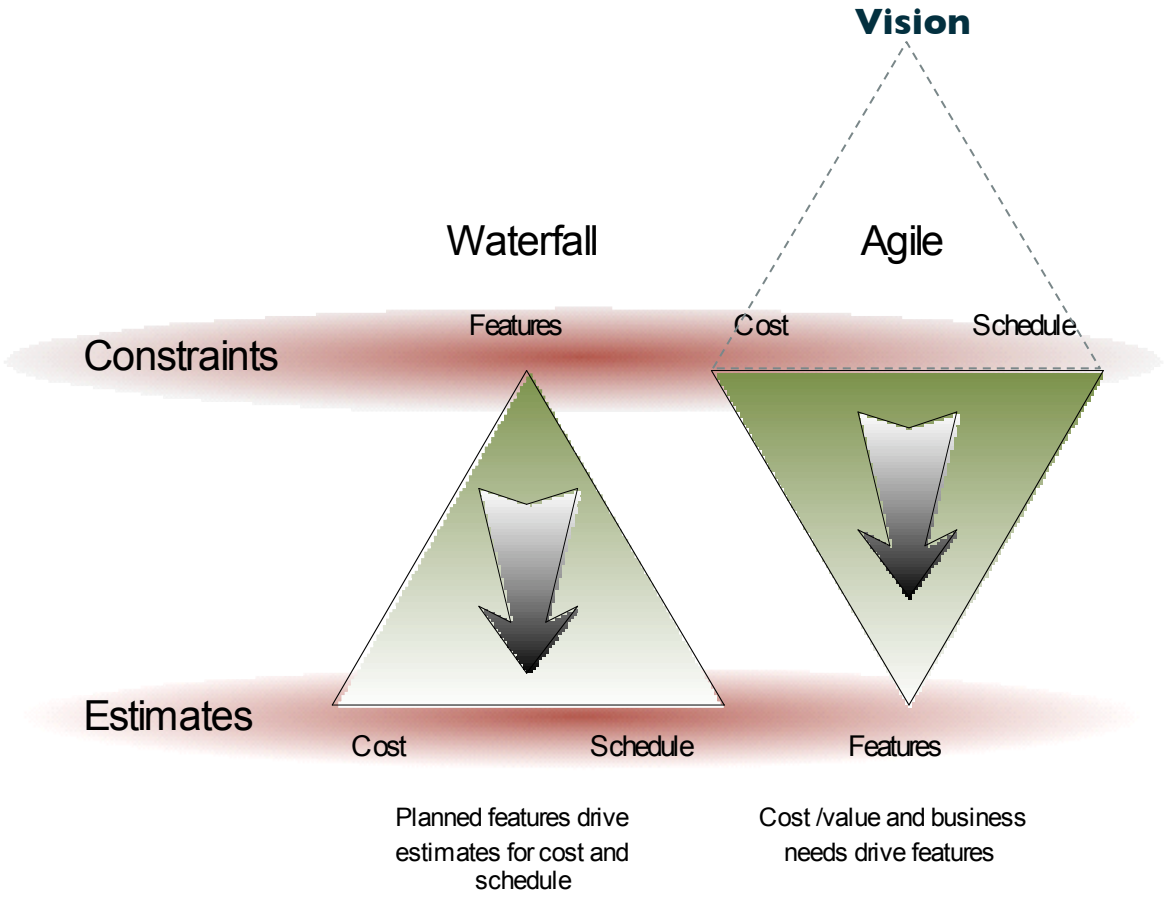
© Mike Cohn, Mountain Goat Software

- Every project has its own balance between anticipation and adaptation
- The balance can change over the course of the project
- The balance should be evaluated regularly in retrospectives
  - ”Are we planning too much? Or are we getting too many nasty surprises? Do we have sufficient ability to adapt in our current processes?”

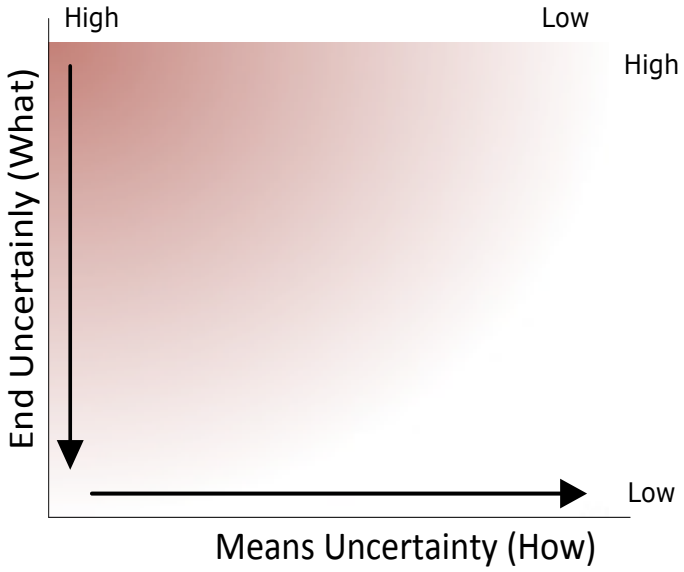
Agile Planning Mindset



… But the Starting Point is Forgotten

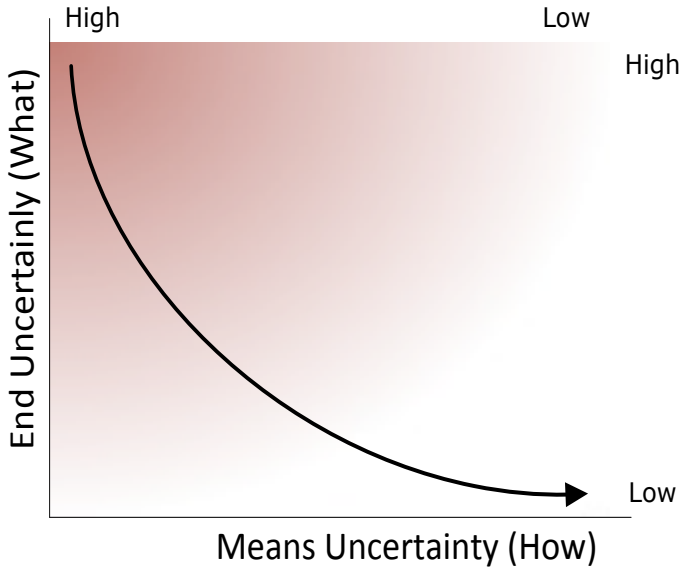


Reducing Uncertainty



Waterfall / phased

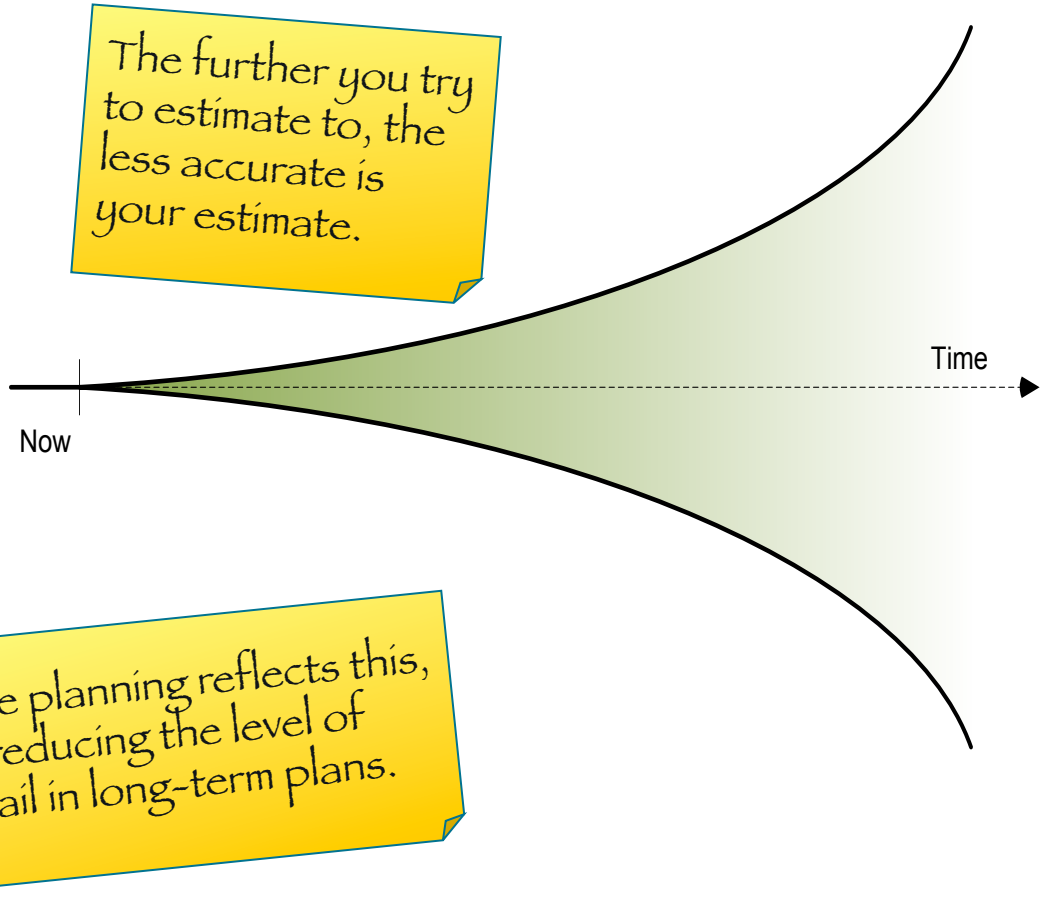
- Requirement specification defines “what” first, then design defines “how”



Agile / iterative

- Initial focus on clarifying “what” with natural transition to “how”

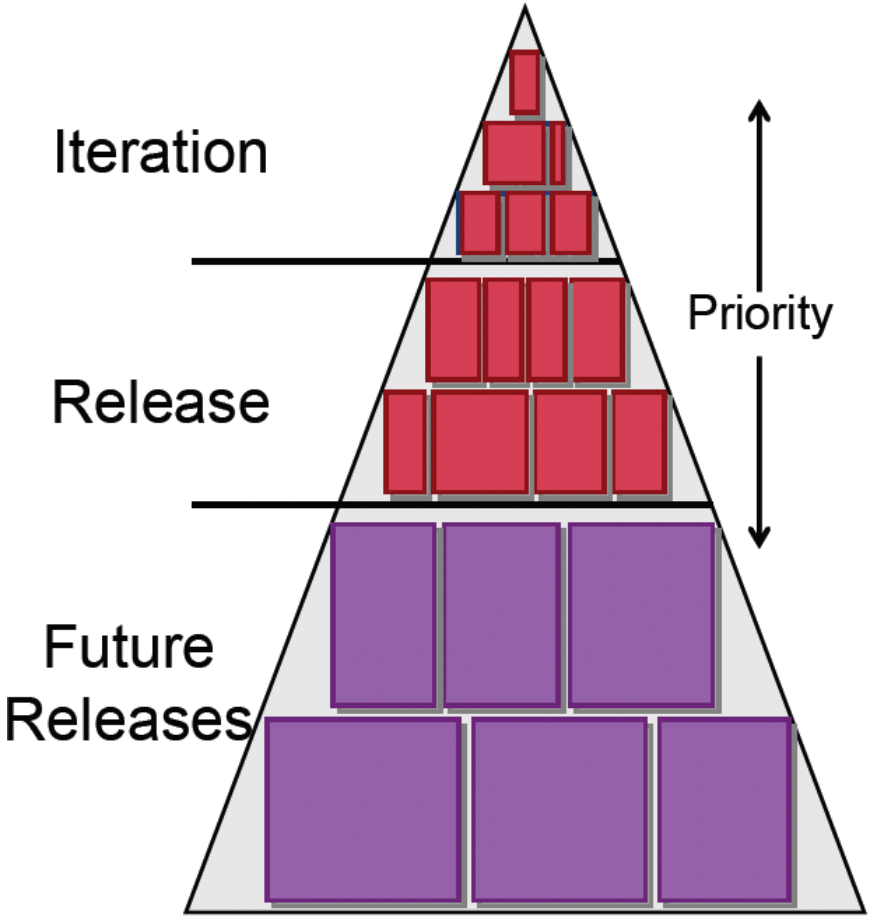
Cone of Uncertainty (Reversed)



Managing Scope

- Scope management is dependent on business objectives
  - Fixed budget → Make a cut-off point
  - Feature-driven → Adjust budget and schedule estimate to include desired stories
  - Value-driven → Develop features as long as they provide enough value-added
  - Fixed schedule → Prioritize features and adjust team size to fit desired most important features into schedule
  - Combinations of above
- Scope-creep is very possible, if not guarded against
- Having everything at top priority is not scope management
  - Results in arbitrary results as no real prioritization is made
- Most large features have different priorities for sub-features
  - Split large stories to smaller ones for meaningful prioritization

Granularity Increases

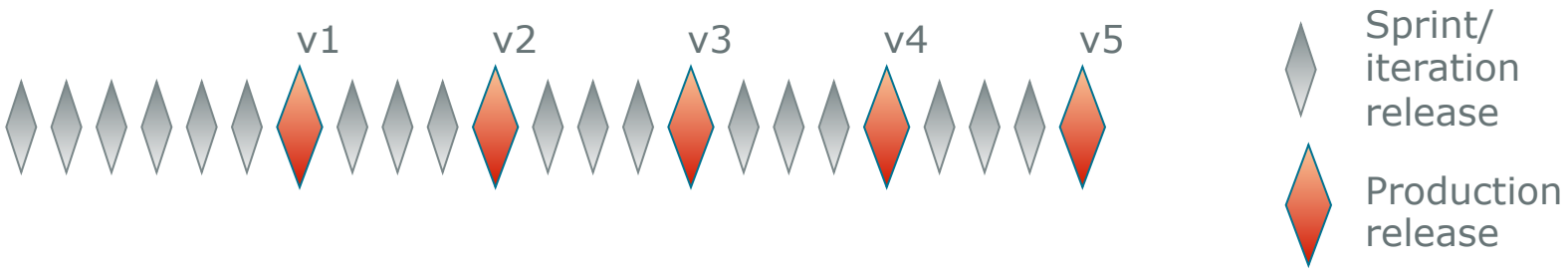


When planning releases, low granularity simplifies planning.

When planning sprints, high granularity is a necessity.

Breaking stories smaller is a continuous activity through the project.

Planning Release Schedule



- Considerations
  - Given your organization and users, how often can you do production releases?
  - How often can you expect to get feedback?
  - How big is the overhead in doing a production release?
  - Are there external parties that need releases at particular times?
- Typically, the more often you can do production releases, the more feedback you will get

Planning Release Content

- Focus on key features (epics)
  - These act as vision elements for the release
- Calculate overall expected capacity
- ”Paint with a large brush”
  - It probably isn’t useful to allocate things to individual sprints
- Identify key milestones within the release
  - External dependencies (either from or to outside)
  - Internal dependencies between teams
  - These milestones are usually quite fixed
    - Plan buffers and slack to ensure their requirements are met

PLANNING AND TRACKING PROGRESS

Estimate Is an Educated Guess

- Estimate is never “correct”
  - Inherent uncertainty
  - Can also be outright wrong
- No-one estimates purposefully wrong (if trust exists)
  - But teams can learn from their estimates
- The correctness of an estimate can be evaluated, but only over time
  - The estimate of the correctness of an estimate is also an estimate! ☺
- Reasons for poor estimates
  - Not enough information or wrong information
  - Different assumptions
  - Things have changed since making the estimate
  - Lack of competence or domain expertise
  - Estimate has been made by someone else than who is doing the work

Effort and Size Estimates

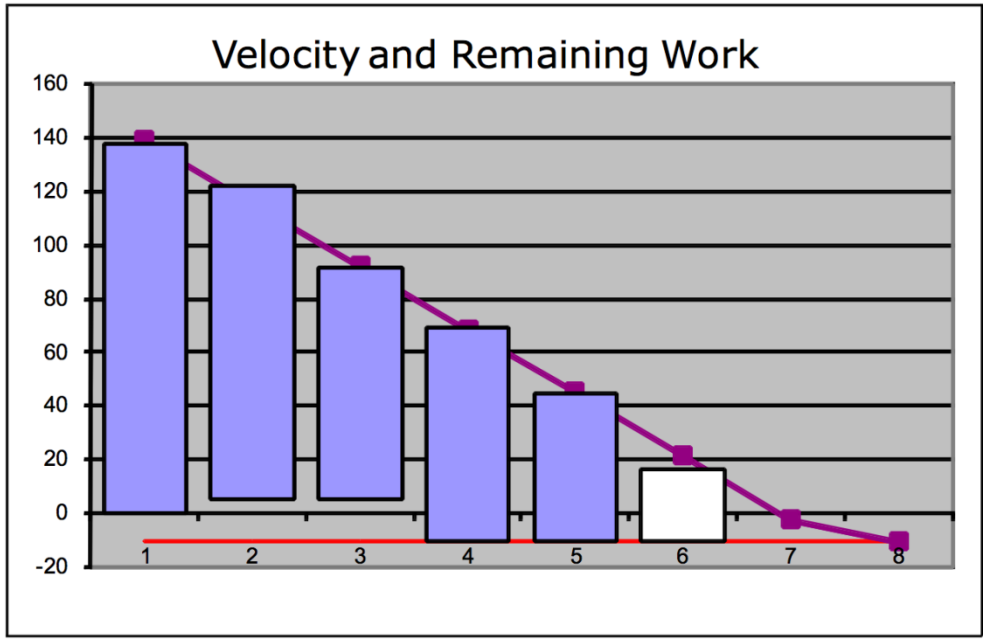
- Team always provices size estimates
- Two primary approaches
  - Cost based – estimates in work hours or money
    - Easy to undestand, easty to compare against available work time or budget
    - Doesn't scale with increasing development speed or systematic estimation errors
    - The concept of Velocity doesn't work with these
    - Difficult to estimate early on (as the developers fear, and rightfully so, that the preliminary estimates are taken as commitments)
  - Relative estimates – estimates in story points
    - Fast to estimates, scales well, independent of skill, reliable even early on
    - Estimates cannot be compared between projects
    - Recommendable approach for Agile projects
- Both can be used for estimation and tracking

Velocity

- Scrum measures development team with ”velocity”
  - Velocity equals features delivered per iteration
- Velocity requires using relative estimation
- Velocity is used to estimate future achievable scope and schedules
- Individual iteration velocity will vary, use only averages to plan future
  - Historical trends extended to future

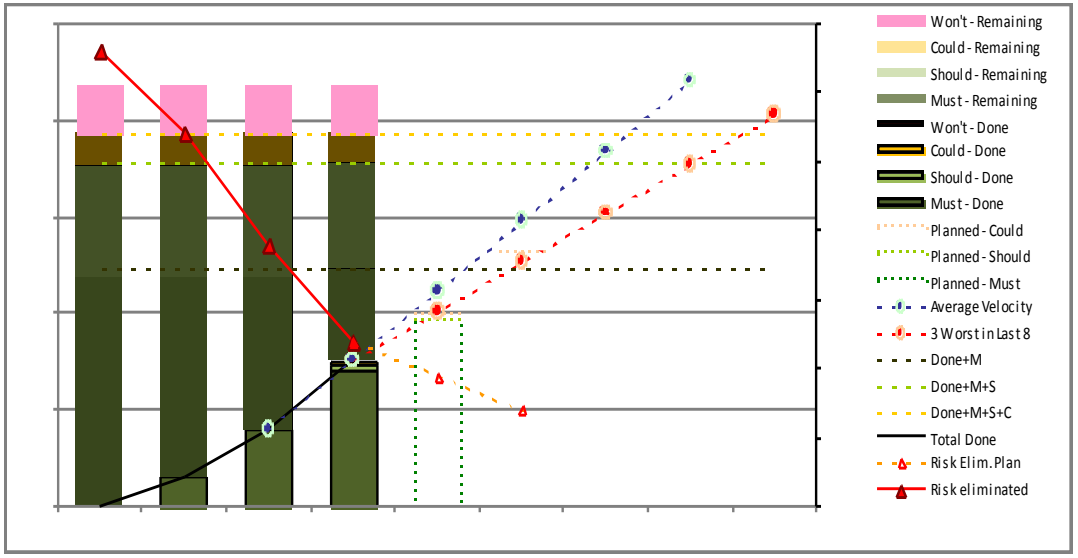
Product Burndown Chart

- Project progress is visualized in the product burndown chart
- The same chart can provide visual planning feedback



- The burndown can also show changing scope
  - Reduced scope → the bottom of a bar rises
  - Increased scope → the bottom goes down
- The effect on estimated duration can be viewed from the chart

”Burn-up”



Thank You!

For more information, email to
petri.heiramo@gmail.com