## Introduction to Agile

* 1. Software development as a complex adaptive system.
  2. Waterfall – main issues and root causes
  3. Agile – main advantages over Waterfall
  4. The Manifesto for Agile Software Development – goal, values, principles
  5. Modern Agile – core tenets and what they mean

## Agile in Practice

* 1. Teams – what makes a good team for Agile software development?
  2. Planning – how do we ensure realistic commitments and achievable plan
  3. Iterations and flow – main differences
  4. Feedback – importance of, examples of, frequency

## Agile Product Management

* 1. Software Economics – understand the four laws
  2. Knowing your users – personas, experiments, dogfooding, early involvement, adoption curve
  3. Focusing on working software – definition of done, technical debt
  4. User stories – essence, purpose, what they are not,
  5. Other forms of work – features, spikes, etc.
  6. The Product Backlog – what makes a good backlog
  7. Story Mapping – how it works, benefits over a flat backlog
  8. Slicing user stories – purpose, techniques, success criteria
  9. Managing multiple backlogs – when and why would we organize backlogs a certain way?

## Estimation, Planning and Tracking

* 1. Targets, estimates, commitments
  2. The Cone of uncertainty – the effect of planning horizon distance on estimation accuracy
  3. Story points – what they represent and what they are not. Key properties – subjectivity, affinity
     1. Components of size – effort, complexity, risk
  4. Fixed scales – why use them. The Fibonacci scale.
  5. Ideal days – what they are and what they are not. Difference with Story Points.
  6. Estimation rituals: planning poker, reference items, buckets. What to do, what not to do. Main benefits and drawbacks of.
  7. Re-estimation – when to do it and when not to do it
  8. Overestimation vs. underestimation
  9. Tracking
     1. Burndown charts – work done, scope added, projections
     2. Cumulative flow diagrams – cycle time, throughput, projections, bottlenecks

## Inspect, Adapt, Retrospect

* 1. The importance of feedback in a complex adaptive system
  2. Effects of feedback loop length
  3. Nesting feedback loops – benefits, examples
  4. Other forms of feedback
  5. Retrospectives – essence, the Prime Directive, essence, stages
  6. Good vs. bad retrospectives

## Agile Metrics

* 1. Rationale – why use metrics, key benefits and pitfalls
  2. Metrics, Insights, Decisions and Outcomes
  3. Metrics quality – what makes a good vs. bad metric
  4. Dimensionality and balance
  5. The six main dimensions of agility
  6. Key metrics: throughput, velocity, cycle time, lead time, queue length, predictability
  7. Insights – trends, correlations, cause-effect
  8. Decisions – teams, processes, practices
  9. Outcomes – real life examples (team size, WIP size, branch lifetime)

## TPS, Lean, Kanban

* 1. Overburden and inconsistency
  2. Types of waste. How do they translate to software development?
  3. Just-in-Time decision making. Examples in Agile software development?
  4. Intelligent automation – examples in software development? (hint: CI/CD, DevOps)
  5. Continuous improvement, respect for people – Agile perspective?
  6. Quality and integrity. Examples in software development.
  7. Local (sub)optimization – examples, dangers
  8. Value streams – value-adding activities, waste
  9. Continuous flow. Pull systems. WIP limits
  10. Kanban – boards (columns, lanes, WIP limits, policies), flow (priority of execution)

## Scaling Agile

* 1. Start with why. The right reason to consider Agile
  2. Big Bang vs. Iterative/Pilot approach
  3. Whom to involve?
  4. What we need to put in place?
  5. Prerequisites for Agile to even exist
  6. Common mistakes
  7. Common challenges (top 5) – culture, resistance, support, skills, training/coaching
  8. Success tips – training/coaching, consistency, sponsorship
  9. Scaling frameworks common grounds - number of teams, shared backlog, cadence, synchronization, shared events (planning, demo, retrospective)