Incorporating Learning and Expected Cost of Change in Prioritizing Features on Agile Projects

R. Scott Harris[1] and Mike Cohn[2]
[1]Montana State University–Billings
[2]Mountain Goat Software

Business value

- Usual advice to product owners is to prioritize based on "business value"
- But what is business value?
  - Putting the competition out of business?
  - Lowering delivery cost?
  - Increasing short term revenue?
  - Achieving cash-flow breakeven?

Telling a product owner to "prioritize on business value" offers as much guidance as the president of General Motors ordering a lathe operator to "maximize corporate profits."

Traditional advice

- Saaty's Analytic Hierarchy Process is often considered "the most promising approach"
  - Involves pairwise comparison of all features
  - Perhaps feasible once at the start of a project
  - Assumes perfect knowledge
- Agile projects incorporate and acknowledge learning and feedback
  - Not feasible every iteration on an agile project

Three guidelines

1. Defer features with high expected costs of change
2. Bring forward features that generate useful knowledge
3. Incorporate new learning often, but only to decide what to do next

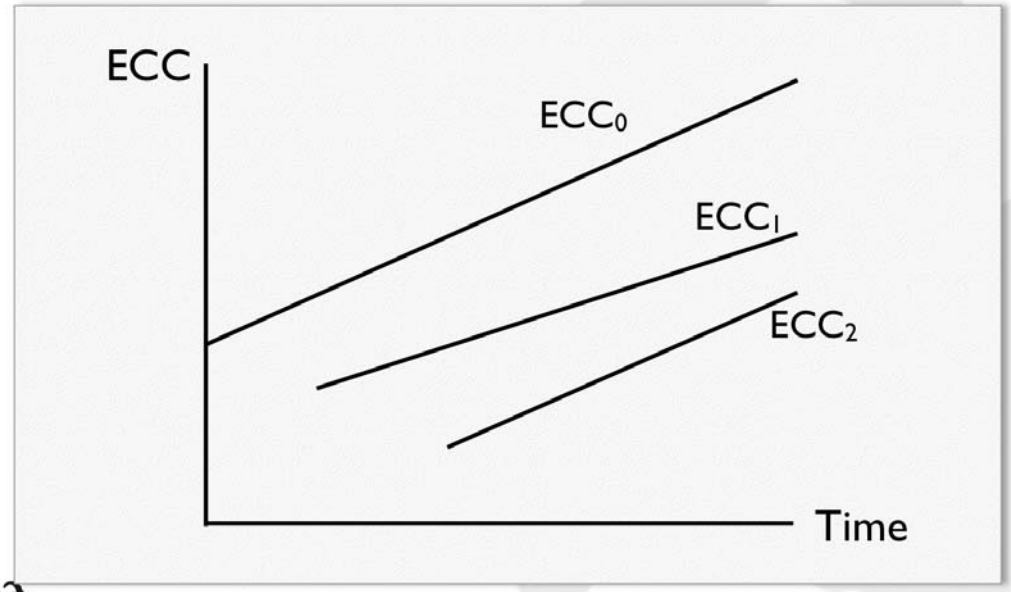
Guideline 1
Defer features with high expected costs of change

- Expected Cost of Change = ECC

ECC = (probability of change) × (cost of change)

- Overall expected cost can be lowered if features that are likely or costly to change are deferred
  - We'll know more later so deferring these means we're more likely to get them right

Expected Cost of Change



An implication

- Because of this we want to:
  - Prioritize activities that have the greatest impact on lowering the ECC curve
    - This leads to:

Guideline 2
Bring forward features that generate useful knowledge

Useful knowledge

- Comes in a variety of forms
  - About the desirability of a feature
  - About the usability of a feature
  - About the technical feasibility of a feature
- Useful knowledge is knowledge that will affect prioritization of subsequent features
  - Product owner asks herself, "If this feature had been implemented already, would I do anything differently?"

Guideline 3

Incorporate new learning often, but only to decide what to do next

- Learning is a continuous process
  - Agile projects acknowledge that all learning cannot be put upfront (as sequential projects try)
- So, decision-making about priorities is simplified
  - "Now" vs. "Not Now"
  - Those not done "Now" are reevaluated next iteration
- Supports agile preference for short iterations

Release plans still necessary

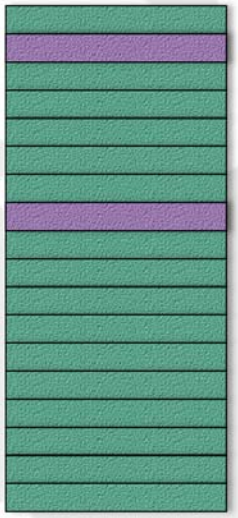
- Release plans are still useful and often necessary
  - Help establish a vision for where a project wants to end up
- But should not detail iteration by iteration sequencing details

Practical application

- Our advice to clients:
  - Perform rough, initial prioritization based on the "business value" of each feature
  - Don't bother prioritizing beyond the next 1-3 iterations
  - Think of ECC and knowledge generated as sliders
    - Move items forward or back in the prioritization

Example: architecture

- Consider a feature that:
  - Has significant architectural implications
  - Does not have an exceptionally high ECC
  - Will generate significant new knowledge
- Based on ECC, feature does not slide backward
- Based on knowledge generated, feature does slide forward



Some examples

- We've used this to support early selection of:
  - A particular application server
  - Features to test designs for a security framework
  - Features that confirm main metaphors of the user interface design
- We've used this to defer decisions with high ECC that generate little new knowledge
  - Choosing among three client technologies

Conclusions

- More useful than advice to prioritize on "business value"
- Instructing product owners to
  - consider relative changes in Expected Cost of Change (ECC)
  - amount and significance of knowledge generated

  leads to better decisions
- Guideline-based approach is easy
  - Keeps focus on "what one thing should we do next" rather than "what is full set of priorities"
- More iterative approach to prioritizing acknowledges learning and fits with agile approach better