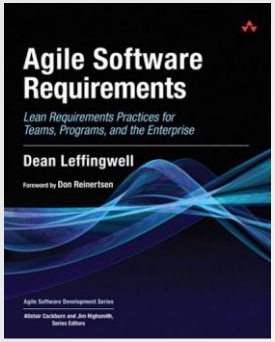


Agile Software Requirements

Software Requirements Engineering – 40688

Computer Engineering department

Sharif university of technology

Fall 402

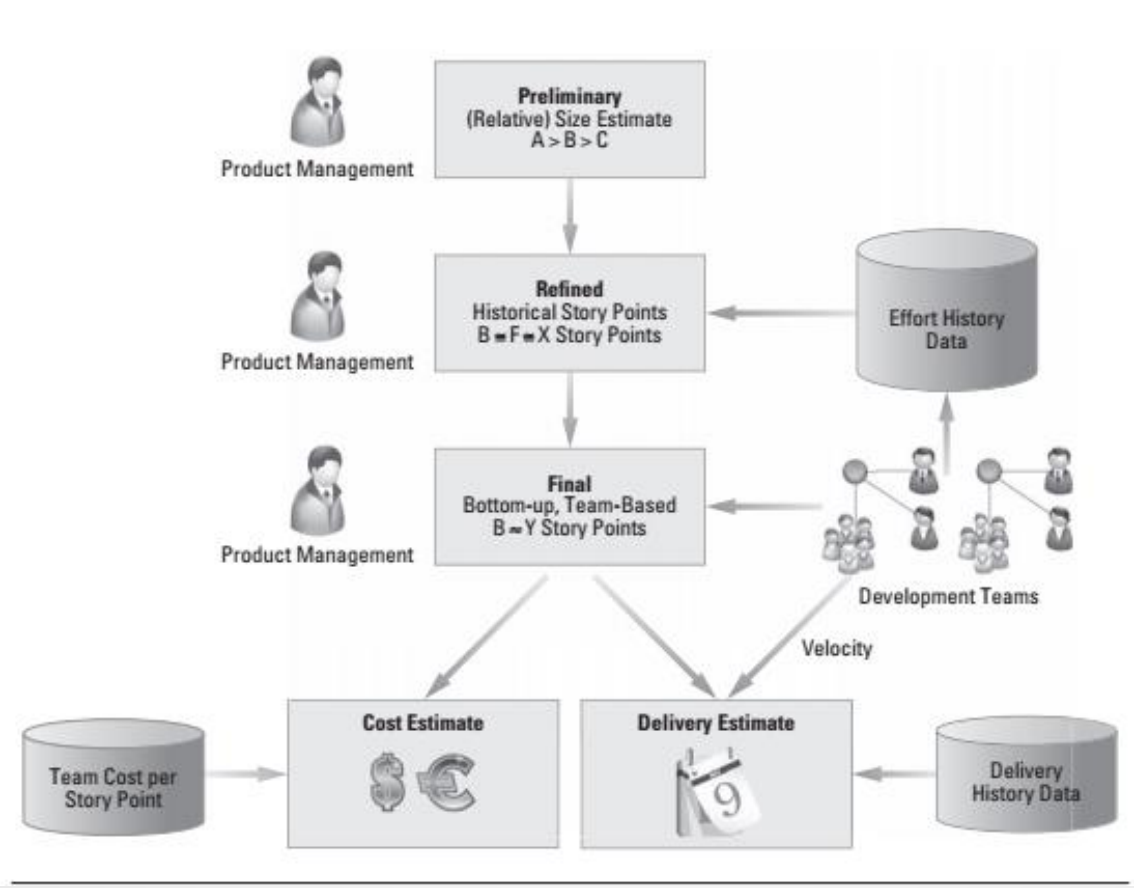
Chapter 13:

Vision, Features, and RoadMap

Expressing Features in User Voice Form

- An agilist want to express a feature in user story voice form.
- As a writer, I can get automatic notification of spelling errors as I write so that I can correct them immediately.
- There is nothing right or wrong about this.
- There is certainly an advantage in this approach, because the user role and benefit are more clearly described.
- They are written at a higher level of abstraction.

Estimating Features



Estimating Effort

Estimating the amount of effort needed to implement a feature typically goes through a series of successive refinements.

- ✓ Preliminary: Gross, Relative Estimating
- ✓ Refined: Estimate in Story Points
- ✓ Final: Bottom-up, Team-Based Estimating

Preliminary:
Gross, Relative Estimating

- For initial backlog prioritization
- Simply need a rough estimate of the effort to implement a feature.
- Even before discussion with the teams.
- They can simply use the same, relative estimating model.
- This provides a simple and fast estimating heuristic.

Refined:
Estimate in Story Points

- With story points, the teams can start to reason about potential cost and schedule impact.
- For this, we need tracking information that allows us to convert the effort estimate for features into story points.
- Many agile project management tools support the feature-to-story hierarchy, and the teams can leverage this to build some historical data.
- This is still a fairly gross estimate, because it depends on comparing the new feature to like features for which there is historical data.

Final:
Bottom-up, Team-Based Estimating

- The fidelity of the estimate can be significantly improved by having the estimating done by the teams.
- They will typically have their own history of like features.
- Ad hoc requests for estimating interrupts the team from their daily iteration activities.
- The estimating task is most efficiently done on a cadence.
- Semiweekly **feature preview** meeting for just this purpose

Estimating Cost

- Once the feature has been estimated in the currency of story points, a cost estimate can be quickly derived.
- Simply look at the burdened cost per iteration time box for the teams that provided the estimates.
- And divide that by their velocity.
- This gives an estimate of the *cost per story point* for the subject teams affected by the new feature.
- Additional work may be necessary when teams are distributed in differing geographic locations.

Estimating Development Time

- Given an understanding of what percentage of the team's time the program is willing to devote to the new feature, we can also use historical data to predict how long it will take to deliver.
- For example, if feature A was implemented in about 1,000 story points and took three months and feature B is a little bigger, then feature B will take a little more than three months.
- Assuming similar resource allocations and availability.

Testing Features

- Feature testing is referenced in quadrant 2 of the agile testing matrix.
- Features cannot be considered *done* until they pass an acceptance test.
- For teams that are organized by feature, this testing can be done by the team that implements the feature.
- If the teams are organized around components, then much of this testing is likely to be done by the system team.
- In either case, however, there are likely to be some spanning features that touch multiple feature and component teams.

Prioritizing Features

- Customers are seemingly reluctant to prioritize features. Perhaps this is because they simply "want them all," which is understandable.
- Product managers are often even more reluctant. Perhaps this is because they would be assured of receiving all the ultimate value.
- Quantifying value is extremely difficult.
- It is often necessary to compare and prioritize very unlike things.

Provide a ROI per feature (1)

- We often attempt to provide a return on investment (ROI) per feature, by predicting the likely increase in revenue if a feature is available.
- Of course, determining feature ROI is most likely a false science, because no one's crystal ball is an adequate predictor of future revenue.
- Especially when you attempt to allocate revenue on a per-feature basis.

Provide a ROI per feature (2)

If we could simply establish value and cost (if not in absolute terms, then at least relative to other features), then we have a way to prioritize based on economics.

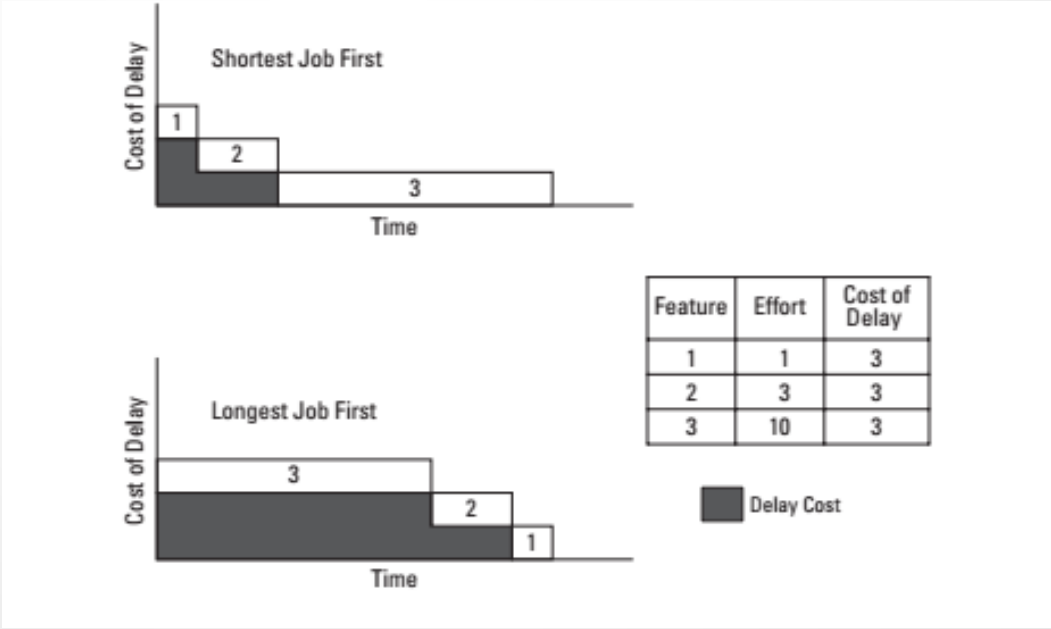
$$\textbf{Relative Priority} = \textbf{Relative ROI} = \textbf{Relative } \frac{\textbf{Value}}{\textbf{Cost}}$$

Prioritizing Features Based on the Cost of Delay

- As Reinertsen points out, “If you only quantify one thing, quantify the cost of delay.
- Fortunately, however, we don’t have to quantify only one thing.
- we’ll actually be able to quantify two things
- The feature effort estimate, as we have described earlier, and the cost of delay.
- Reinertsen describes three methods for prioritizing work based on the economics of CoD.

Shortest Job First

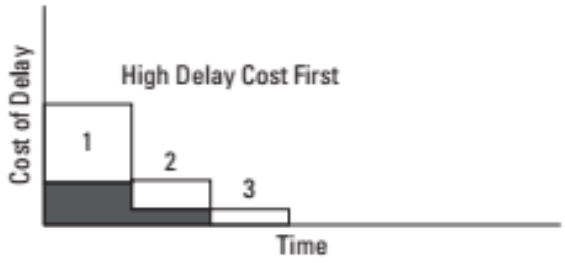
- When the cost of delay for two features is equal, doing the Shortest Job First.
- If two features have the same CoD, do the smallest feature first.

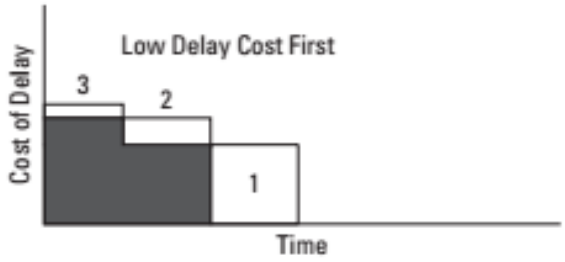


Feature	Effort	Cost of Delay
1	1	3
2	3	3
3	10	3

High Delay Cost First

If two features have the same effort, do the feature with the highest CoD first.



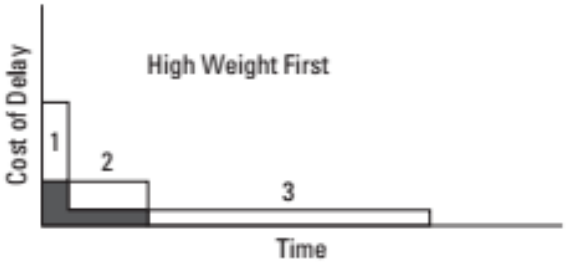


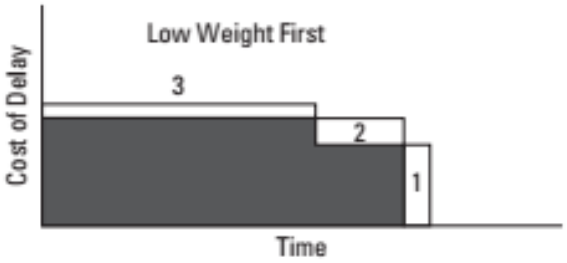
Feature	Effort	Cost of Delay
1	3	10
2	3	3
3	3	1

Delay Cost

Weighted Shortest Job First

If two features have different efforts and CoD (and they almost always do), do the weighted, smallest effort feature first.





Feature	Effort	Cost of Delay	Weight = CoD/Effort
1	1	10	10
2	3	3	1
3	10	1	0.1

Delay Cost

Estimating the Cost of Delay

CoD is an aggregation of three attributes of a feature.

1. **User value**

   The potential value of the feature in the eyes of the user

2. **Time value**

   Based on how the user value decays over time.

3. **Risk reduction/opportunity enablement value**

   Our world is laden with both risk and opportunity

An Example

	Cost of Delay				Effort	WSJF
	User	*Time*	*Risk Red.*	*Total*		
Feature A	4	9	8	21	4	5.3
Feature B	8	4	3	15	6	2.5
Feature C	6	6	6	18	5	3.6

Legend:

Scale: 10 is highest, 1 is lowest.

Total is sum of individual CoD.

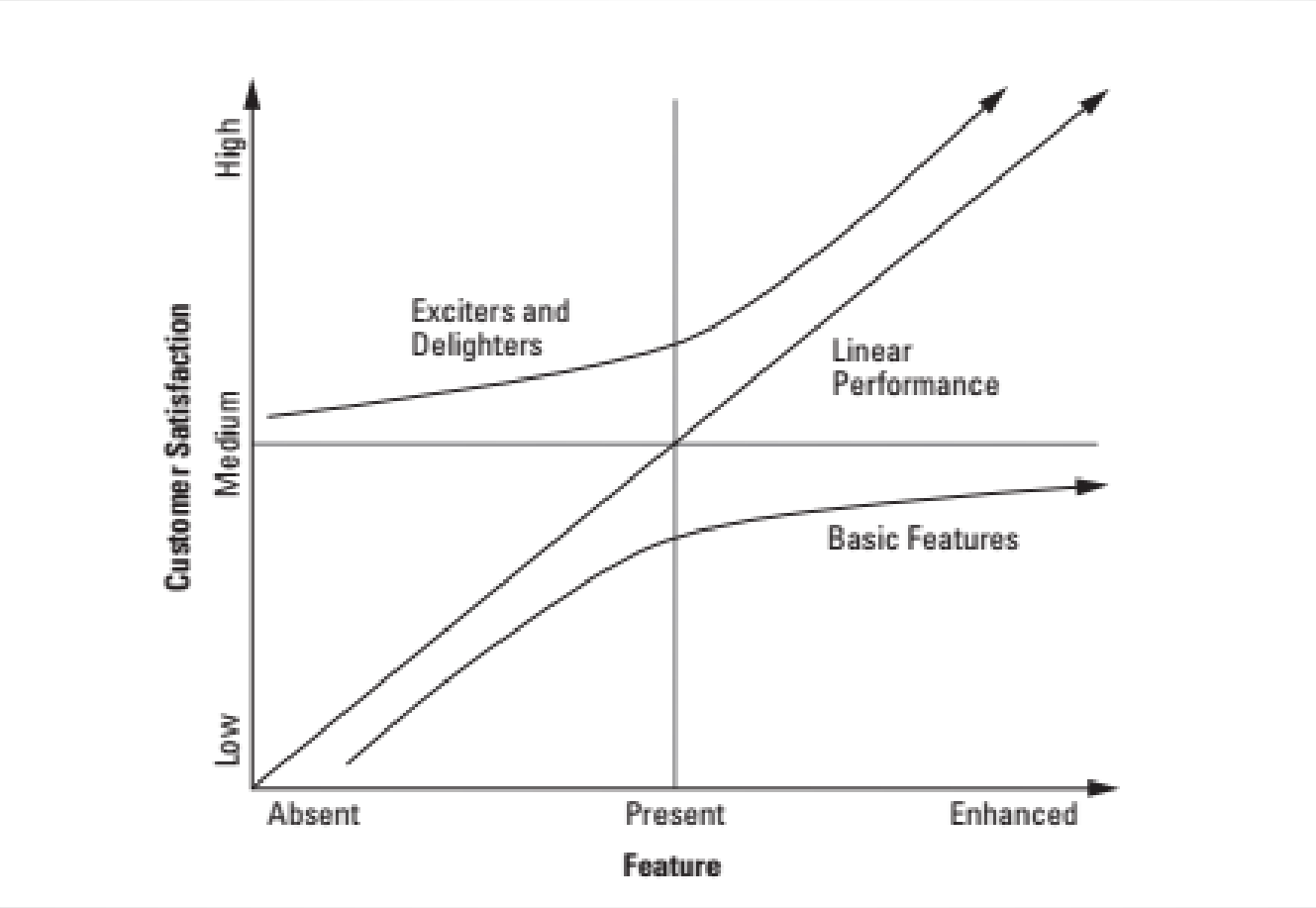
WSJF (weighted result) is calculated as Total (Cost of Delay) divided by Effort.

The Kano Model of Customer Satisfaction (1)

Specifically, the Kano model challenges the assumption that customer satisfaction is achieved by balancing investment across the various attributes of a product or service.

Rather, customer satisfaction can be optimized by focusing on *differential features*, those "exciters" and "delighters" that increase customer satisfaction and loyalty beyond that which a proportional investment would otherwise merit.

The Kano Model of Customer Satisfaction (2)



The Kano Model of Customer Satisfaction (3)

The model illustrates three types of features

1) ***Basic (must-have) features***

   Features that must be present to have a viable solution.

2) **Linear features**

   Features for which the capability of the feature is directly proportional to the result.

3) **Exciters and delighters**

   These are the features that differentiate the solution from the competition.

Prioritizing Features for Differential Value

1) **Differential value rule #1:**

   Invest in MMFs, but never overinvest in a feature that is already commoditized.

2) **Differential value rule #2:**

   Drive innovation by having the courage to invest in exciters.

3) **Differential value rule #3:**

   If resources do not allow you to compete on the current playing field, change the playing field.