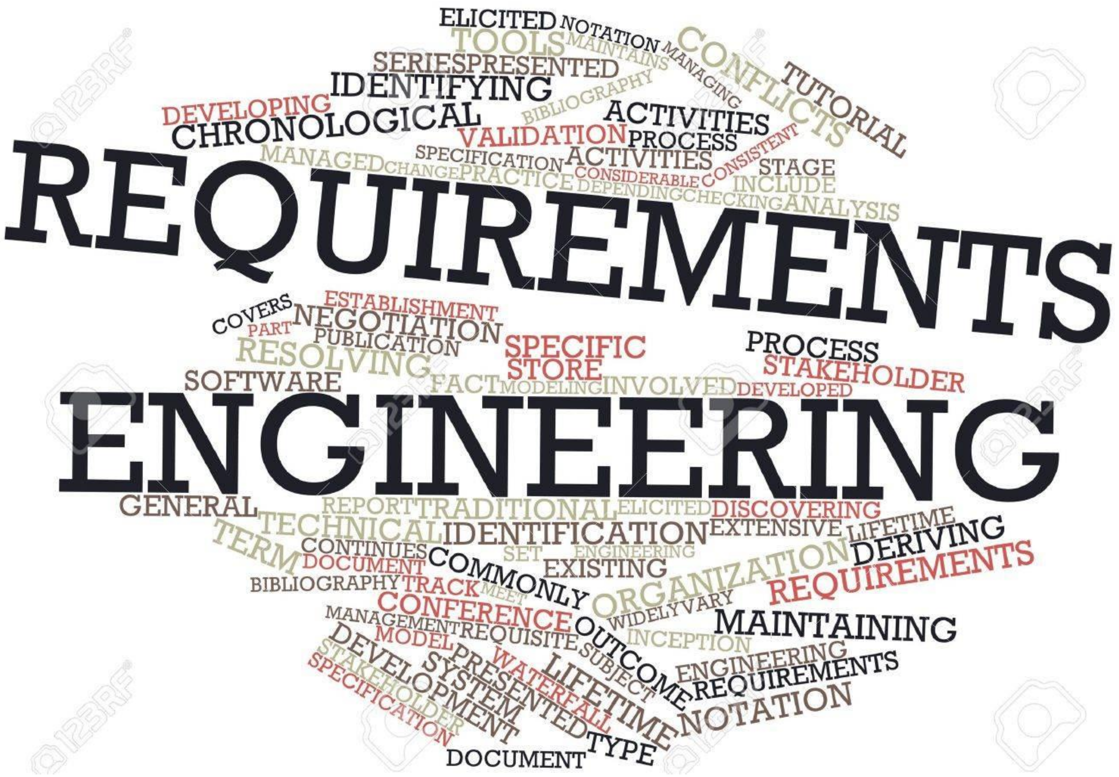


SWE 215: Software Requirements Engineering

Lecture 9

Features Prioritization

Course Topics

- ~~Why Requirements Engineering?~~
- ~~Introduction to Requirements~~
- ~~RE in Software Development Life Cycles~~
- ~~System Vision, Context, and RE Framework~~
- ~~Fundamentals of Goal Orientation~~
- ~~Fundamentals of Scenarios~~
- ~~Requirements Discovery~~
- ~~User Stories and Agile Estimation~~
- Features Prioritization
- Requirements Negotiation
- Requirements Validation
- Fundamentals of Requirements Management

Lecture Objectives

- Learn how to formulate, estimate, and prioritize features to deliver the maximum value to users
- Learn about product Roadmap

Features

Features can be described as follows:
"Services provided by the system that fulfill one or more stakeholder needs."

- ➢ The language typically used by **marketing** to describe the capabilities and benefits provided by a new system.

- ➢ The primary content of the Vision is **a set of prioritized features**, which describe what new things the system will do for its users, and the benefits the user will derive from them.

- ➢ Features also provide a focus to **organize agile teams around** (as the feature team).

Expressing Features in User Voice Form

- It is also natural for an agilist to want to express a feature in **user story voice form**, so a feature such as ”*automatic spell checking*” becomes the following:

“As a writer, I can get automatic notification of spelling errors as I write so that I can correct them immediately.”

- The advantage in this approach is that **the user role and benefit are more clearly described.**

The Problem of Feature Prioritization

There are a number of reasons why prioritization is such a **hard problem**:

- **Customers are seemingly reluctant to prioritize features. Product managers are often even more reluctant.**
- **If they could only *get them all*, they wouldn't have to prioritize anything.**
- **They cannot gain internal agreement.**
- **They are uncertain as to what the relative priorities are.**
- **Quantifying value is extremely difficult**. Some features are simple "must haves" to remain competitive or keep market share. **How does one quantify the impact of keeping market share, one feature at a time?**

Prioritizing Features

- It is often necessary to **compare** and **prioritize** **very unlike things**. For example, how does one prioritize an entirely **new feature** that could take many months against a **minor feature** that can be delivered in just a few weeks?

- **Return on investment (ROI) per feature, by predicting the likely increase in revenue if a feature is available.**

- Determining feature ROI is most likely **a false science.**

- Any product manager or business analyst can probably make a case for a great ROI for their feature; otherwise, they wouldn’t have worked on it to begin with.

Value/Effort as an ROI Proxy: A First Approximation

The relative ROI is the relationship between **potential return** (**value**) divided by the **effort** (cost to implement) for a feature.

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

Deliver a **higher ROI** feature before a **lower ROI** feature

What's Wrong with Value/Effort ROI?

- Based on a more complete economic framework, the assumption that a high relative ROI feature should naturally have precedence over a lower ROI feature is **not Correct.**

- **The potential profit for a particular high ROI feature could be less sensitive to a schedule delay than a lower ROI feature.** In this case, *the lower ROI feature should be implemented first*, followed by the higher ROI feature.

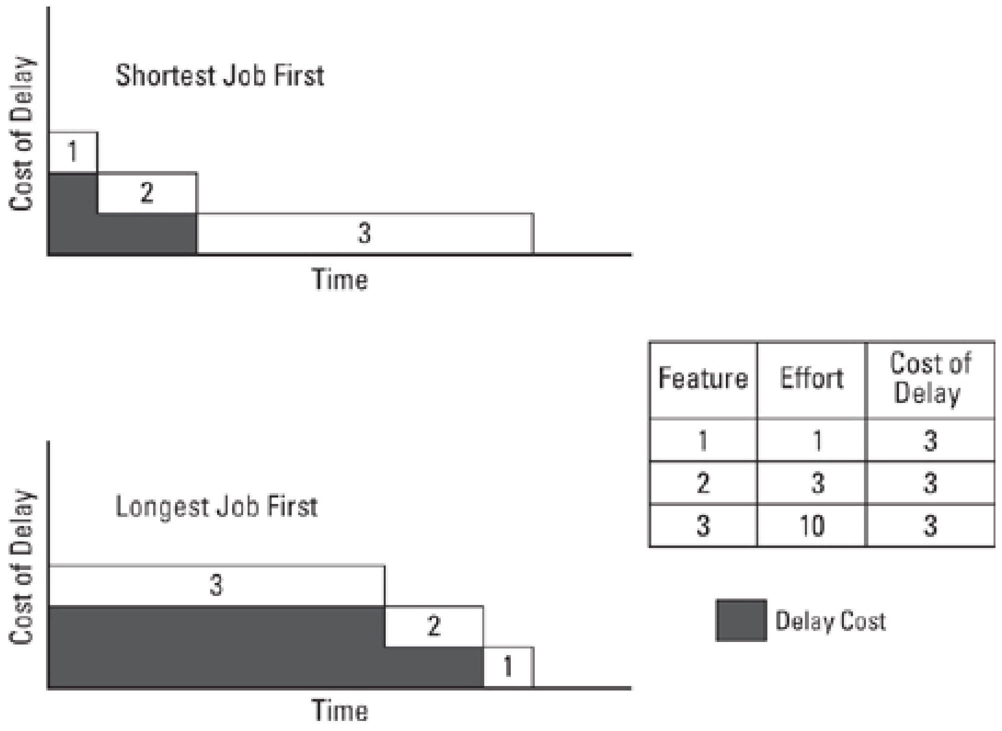
Prioritizing Features Based on the Cost of Delay

Reinertsen describes three methods for prioritizing work based on the economics of CoD:

1. **Shortest Job first**
2. **High Delay Cost First**
3. **Weighted Shortest Job First**

Prioritizing Features Based on the Cost of Delay: Shortest Job First

When the cost of delay for two features is equal, doing the *Shortest Job First*, produces the best economic returns. Delivering the smallest feature first substantially decreases the overall cost of delay in this case.

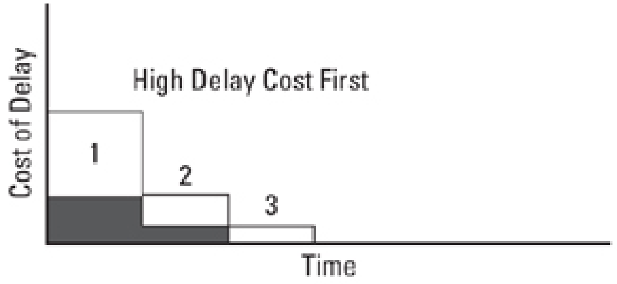


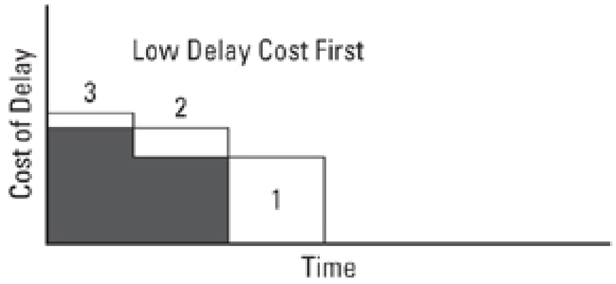
Feature	Effort	Cost of Delay
1	1	3
2	3	3
3	10	3

Prioritizing Features Based on the Cost of Delay: High Delay Cost First

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

In other words, if CoD is a proxy for value and if one feature has more value than another and if it's the same effort/time, we do the higher value feature first; we knew that already from our ROI value/effort proxy.



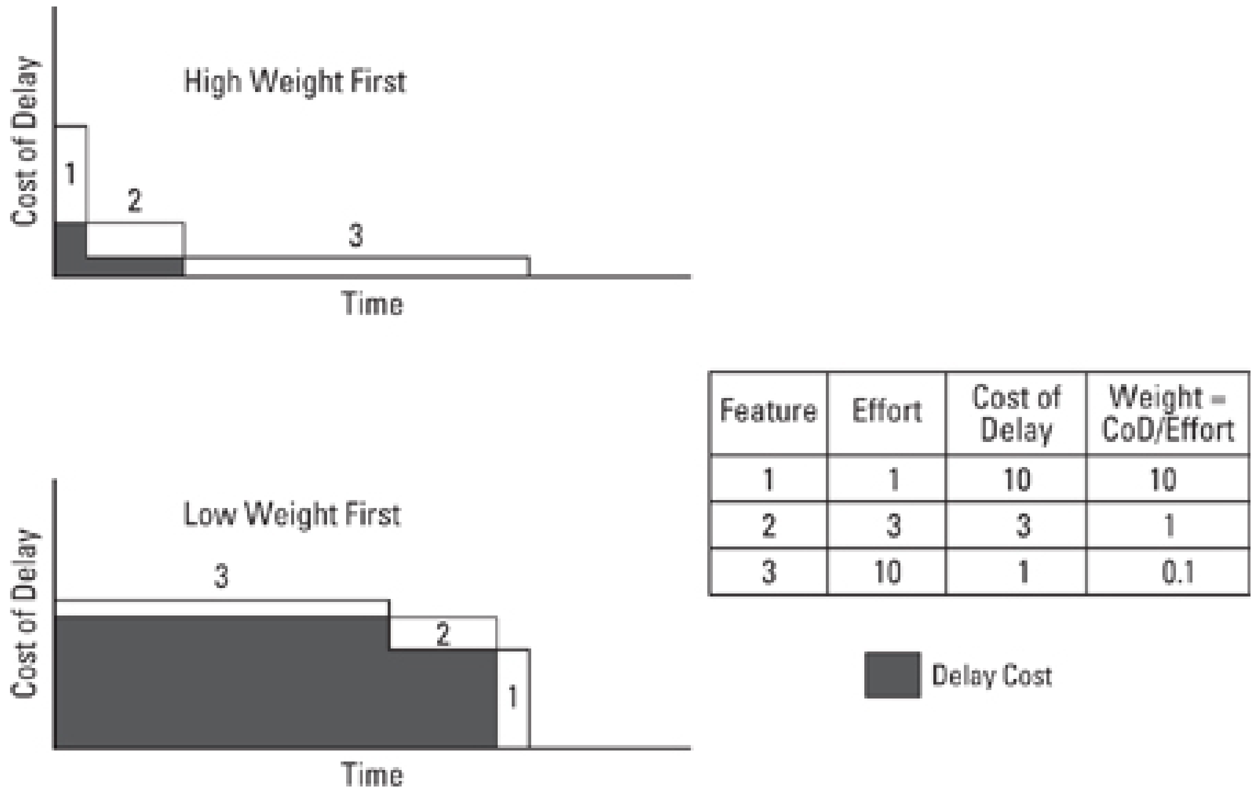


Feature	Effort	Cost of Delay
1	3	10
2	3	3
3	3	1

▇ Delay Cost

Prioritizing Features Based on the Cost of Delay: Weighted Shortest Job First

The CoD and implementation effort for different software features are likely to be highly variable.



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

Estimating the Cost of Delay

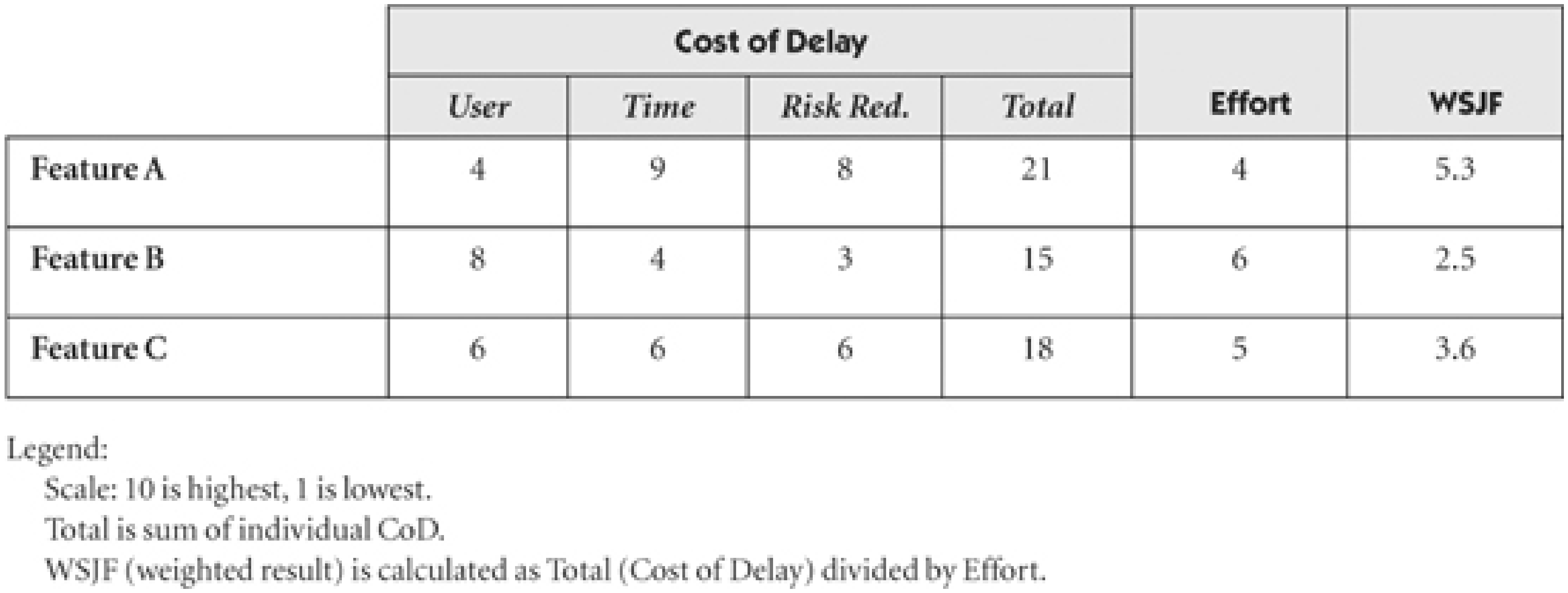
CoD is an aggregation of three attributes of a feature:

- ***User value***: is simply **the potential *value* of the feature in the eyes of the user** (relative estimate).

- ***Time value***: is another relative estimate, one **based on how the user value *decays* over time**.

- ***Risk reduction/opportunity enablement* value**: acknowledges that some features **are more or less valuable to us** based on how they help us mitigate risk, and help us **exploit new opportunities**.

Feature Prioritization Evaluation Matrix



	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.

Prioritization is Local, Gobal, and Temporal

Priority is based on:

- **Cost of Delay,** which is **a *global* property** of the feature,
- **Effort**, which is a ***local* property** of the **team** that is implementing the feature.

The presented model is highly **sensitive** to **the time element**—***priorities change rapidly as deadlines approach.***

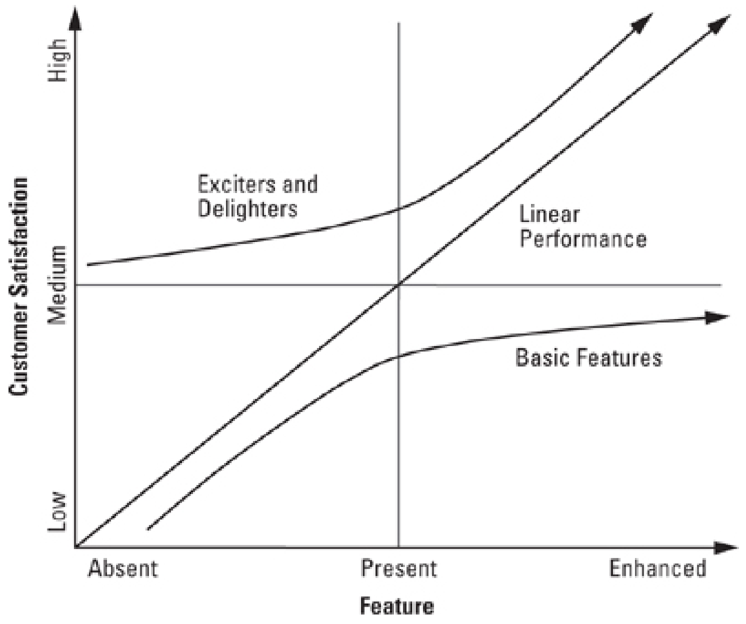
For example, *implement the new student registration system in time for the next academic year* could have a time value of "1 to 2" in January (prior to the next school year start) but could easily be a "10" in May.

Conclusion: **Priorities have to be determined locally and at the last responsible moment. That is the time when we can best assess the CoD and the resources available to work on the feature.**

Achieving Differential Value: The Kano Model of Customer Satisfaction

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

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.



Achieving Differential Value: The Kano Model of Customer Satisfaction

The model illustrates three types of features.

1.Basic (must-have) features: Features that must be present to have a viable solution. Without them, your solution cannot compete in the marketplace.

2.Linear features: Features for which the capability of the feature is directly proportional to the result. Generally, the more you invest in those features, the higher the satisfaction.

3.Exciters and delighters: These are the features that differentiate the solution from the competition. They provide the highest opportunity for customer satisfaction and loyalty.

Achieving Differential Value: The Kano Model of Customer Satisfaction

- The shape of the ***basic* curve** is telling that: Until a feature is simply “present,” satisfaction remains low until a threshold is achieved. **Enhancing the feature produces a *less* than proportional reward.**

- The position and shape of the *exciters and delighters* **curve tells the opposite story**. Because these features are **unique**, **compelling**, and **differentiated**, even a small investment (the area on the left) still **produces high customer interest and potential satisfaction.**

Achieving Differential Value: Prioritizing Features for Differential Value

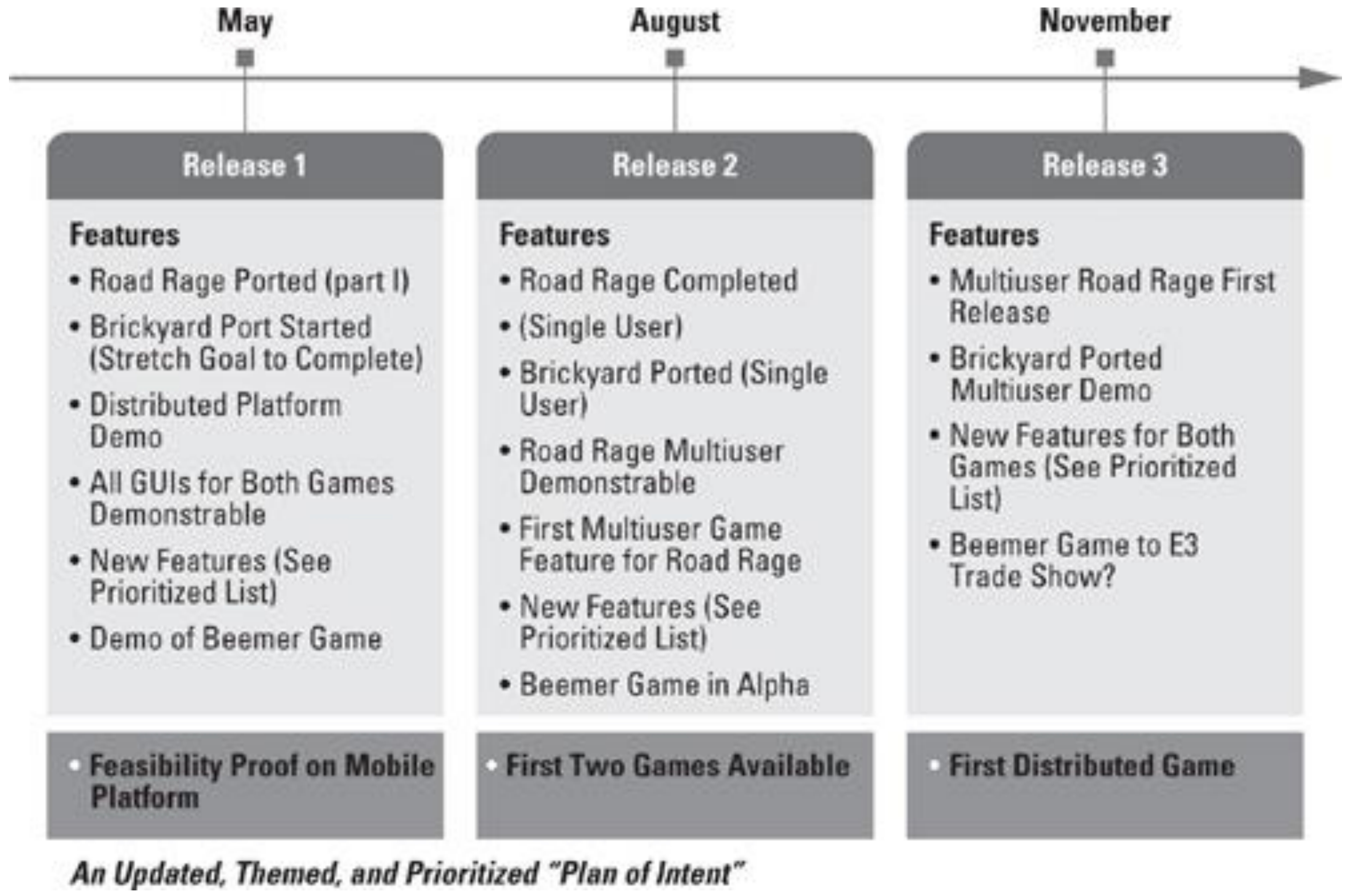
Differential value rule #1: *Invest in MMFs (minimum marketable feature), but **never overinvest in a feature that is already commoditized.***

Differential value rule #2: *Drive innovation by having the courage to **invest in exciters**.*

Differential value rule #3: *If resources do not allow you to compete on the current playing field, change the playing field.*

The Product Roadmap

Communicates *future* objectives to our outside stakeholders.


May
August
November
Release 1
Features
• Road Rage Ported (part I)
• Brickyard Port Started (Stretch Goal to Complete)
• Distributed Platform Demo
• All GUIs for Both Games Demonstrable
• New Features (See Prioritized List)
• Demo of Beemer Game
Release 2
Features
• Road Rage Completed
• (Single User)
• Brickyard Ported (Single User)
• Road Rage Multiuser Demonstrable
• First Multiuser Game Feature for Road Rage
• New Features (See Prioritized List)
• Beemer Game in Alpha
Release 3
Features
• Multiuser Road Rage First Release
• Brickyard Ported Multiuser Demo
• New Features for Both Games (See Prioritized List)
• Beemer Game to E3 Trade Show?
• Feasibility Proof on Mobile Platform
• First Two Games Available
• First Distributed Game
An Updated, Themed, and Prioritized "Plan of Intent"

The Product Roadmap

- Each vertical box represents an upcoming release. The label at the bottom represents **the theme or primary objective of the release**. The features are listed in **prioritized order**.

- The teams can commit only to the **features** in the **next upcoming release**. Releases beyond the next represent only a **best estimate**.