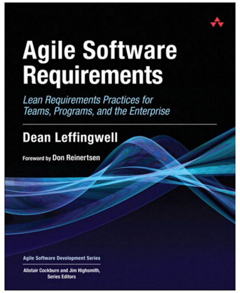
CE Department

Software Requirements Engineering

40688

These slides are designed to accompany Agile Software Requirements (2011) by Dean Leffingwell and support the university course Software Requirements Engineering, instructed by Mehran Rivadeh. Created and designed by Mahnaz Rasekhi.


Agile Software Requirements
Lean Requirements Practices for Teams, Programs, and the Enterprise
Dean Leffingwell
Foreword by Don Reinertsen
Agile Software Development Series
Alistair Cockburn and Jim Highsmith, Series Editors

Agile Software Requirements (2011)
Dean Leffingwell

Vision, Feature, And Roadmap

Chapter 13

Mehran Rivadeh
mrivadeh@sharif.edu
Software Requirements Engineering
October 2025 - Fall 1404 - SUT

Contents

Introduction

Introduction

Vision

- ➔ How to continuously communicate the strategic intent of the program?
- ➔ What are some approaches for communicating and documenting this critical information?

Feature

- ➔ The primary content of the Vision is a set of features.
- ➔ How to formulate, estimate, and prioritize features to deliver the maximum value to our users?

Roadmap

- ➔ How we can see the future of the program unfolding, at least insofar as we can predict it?

Contents

1. Vision

Vision

- ➔ **Traditionally**, the intended **requirements** for a product, system, or application were **captured** and **communicated** in **document form**.
- ➔ When properly applied, **documents** still **work great**, **even in agile**, and we can continue to use them for this purpose.
- ➔ However, the **investment** in **up-front requirements analysis** is **greatly reduced in agile**, and therefore the traditional MRDs, PRDs, SRSs, and the like are unlikely to appear.
  - ◆ Marketing Requirements Documents (MRDs)
  - ◆ Product Requirements Documents (PRDs)
  - ◆ Software Requirements Specifications (SRSs)

Vision

- **Agile** enterprises **take** a **leaner approach** better suited to
  - The last-responsible-moment
  - Delayed decision-making
  - and Artifact-light development practices.
- This **prevents overinvestment** in things we are **unlikely** to **understand very well** anyway and **prevents** the **too-early binding of resources** to **a set of fixed commitments** that are likely to haunt us later.
- This keeps the program agile and light on its feet.

Vision

What is the key factor that ensures agile teams understand what to build when formal documentation is missing?

➔ Since the MRD, PRD, and SRS documents may no longer exist to specify intended system behavior, **communicating** the **Vision directly to** the **agile development teams is even more critical**.

➔ Otherwise, how would they know what it is they are supposed to build?

➔ This is generally **executive's** and **product management's responsibility**, because the **Vision is** an **outcome** of the **company's business** and **portfolio investment strategy**.

Vision

Generally, the **Vision communicates** the **strategic intent** for the **program** and **answers** some of the **big questions**.

1. **Why** are we **building this product**, system, or application?
2. What **problem** will it **solve**?
3. What **features** and **benefits** will it provide?
4. **For whom** does it provide these features and benefits?
5. What **performance**, **reliability**, and **scalability** must it deliver?
6. What **platforms**, **standards**, **applications**, and so on, will it **support**?

Contents

2. Expressing The Vision

Expressing The Vision

How the **Vision** is **communicated to** the **teams** is a matter of the **organization's preference**, and the mechanisms vary greatly.

- ➔ A vision document
- ➔ The advanced data sheet approach
- ➔ The preliminary press release approach
- ➔ The feature backlog with briefing approach
- ➔ Communicating nonfunctional requirements (System qualities)

Contents

2. Expressing The Vision

A Vision Document

- Writing things down is still the best way to communicate
  - when face-to-face is impractical
  - when key decisions and discoveries need to persist over time.
- For example, in RUP, the Vision document is a key, well defined artifact that teams used to communicate the components of the Vision.
  - It is an easy carryover to a team’s agile practices.
  - This approach used to **good effect** on **relatively small teams** of **10** to **15 team members**.
  - Their belief is: “We document the Vision to test our own understanding of what we think we know.”

A Vision Document

- ➔ In **larger programs**, the **Vision** document serves as the “**umbrella**” document for a **large system initiative**.
- ➔ **With respect to agility**, it is important to note that typically only one such **document** of **5** to **10 pages** (**20 maximum** for a large system) is needed, even for a **large green-field program**.
- ➔ So developing and updating this document is not a burdensome overhead.

Template → Book - Appendix B

Contents

2. Expressing The Vision

The Advanced Data Sheet

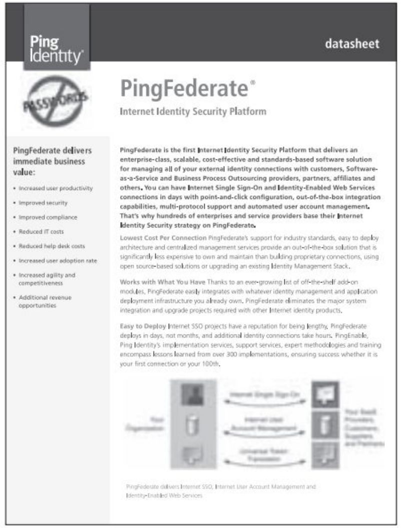
- For product-oriented companies, the **development** of the business case for a new product **requires** an **understanding** of
  - The user’s needs
  - The key benefits a proposed solution is to provide
  - The platforms and operating environments that must be supported in the user’s environment
  - and The key labeling claims for performance, compatibility, and so on.

The Advanced Data Sheet

- ➔ Moreover, the need for the **product team** to **be able** to **articulate** the **business case in** a **concise manner to prospective buyers** is also imperative:
  - ◆ If this cannot be done effectively, the product value proposition is likely to be lost on the marketplace, regardless of the team's ability to produce a worthy offering.
- ➔ Also, since the role of product management is well understood by these teams, one or more individuals on that team will eventually need to **communicate** the **product boundaries** and **features**.
- ➔ One way to do that is with the **development** of a very **preliminary**, **advanced data sheet**.

The Advanced Data Sheet

- ➔ Development of a very preliminary, advanced data sheet
- ➔ It isn’t an actual data sheet per se, but it uses a data sheet template to start to define, at a high level:
  - ◆ **What** the product **does**?
  - ◆ **For whom** it is intended to do it?
- ➔ It is an extremely concise document.
- ➔ Two pages front and back are typical.
- ➔ It must focus on what is critical to communicate.



Ping Identity

datasheet

PingFederate®

Internet Identity Security Platform

PingFederate delivers immediate business value:

- Increased user productivity
- Improved security
- Improved compliance
- Reduced IT costs
- Reduced help desk costs
- Increased user adoption rate
- Increased agility and competitiveness
- Additional revenue opportunities

PingFederate is the first Internet Identity Security Platform that delivers an enterprise-class, scalable, cost-effective and standards-based software solution for managing all of your external identity connections with customers, Software-as-a-Service and Business Process Outsourcing providers, partners, affiliates and others. You can have Internet Single Sign-On and Identity-Enabled Web Services connections in days with point-and-click configuration, out-of-the-box integration capabilities, multi-protocol support and automated user account management. That’s why hundreds of enterprises and service providers base their Internet Identity Security strategy on PingFederate.

Lowest Cost Per Connection PingFederate’s support for industry standards, easy to deploy architecture and centralized management services provide an out-of-the-box solution that is significantly less expensive to own and maintain than building proprietary connections, using open source-based solutions or upgrading an existing Identity Management Stack.

Works with What You Have Thanks to an ever-growing list of off-the-shelf add-on modules, PingFederate easily integrates with whatever identity management and application deployment infrastructure you already own. PingFederate eliminates the major system integration and upgrade projects required with other Internet identity products.

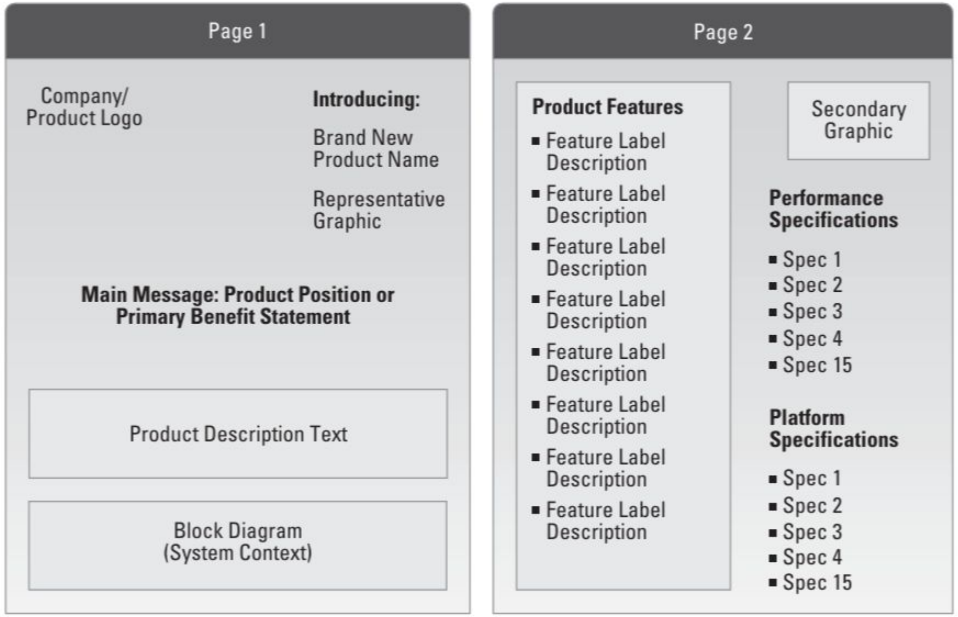
Easy to Deploy Internet SSO projects have a reputation for being lengthy. PingFederate deploys in days, not months, and additional identity connections take hours. PingEnable, Ping Identity’s implementation services, support services, expert methodologies and training encompass lessons learned from over 300 implementations, ensuring success whether it is your first connection or your 100th.

Your Organization
Internet Single Sign-On
Internet User Account Management
Universal Token Translation
Your SaaS Providers, Customers, Suppliers and Partners

PingFederate delivers Internet SSO, Internet User Account Management and Identity-Enabled Web Services

Source: Ping Identity, Ping Federate Data Sheet

The Advanced Data Sheet

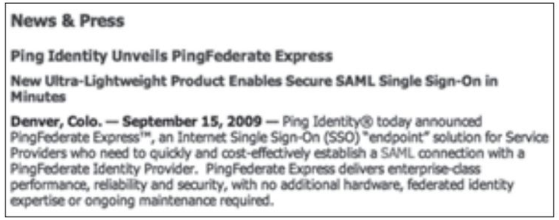

Page 1
Company/ Product Logo
Introducing:
Brand New Product Name
Representative Graphic
Main Message: Product Position or Primary Benefit Statement
Product Description Text
Block Diagram (System Context)
Page 2
Product Features
▪ Feature Label Description
▪ Feature Label Description
▪ Feature Label Description
▪ Feature Label Description
▪ Feature Label Description
▪ Feature Label Description
▪ Feature Label Description
▪ Feature Label Description
Secondary Graphic
Performance Specifications
▪ Spec 1
▪ Spec 2
▪ Spec 3
▪ Spec 4
▪ Spec 15
Platform Specifications
▪ Spec 1
▪ Spec 2
▪ Spec 3
▪ Spec 4
▪ Spec 15

➔ Although this data sheet appears to be very simple on the surface, teams will quickly discover that drafting the data sheet is, in fact, a fairly difficult exercise but one that forces development of an early and concise common Vision for the team.

Contents

The Preliminary Press Release

- ➔ Some teams have found that drafting a preliminary, hypothetical press release causes them to **think through** the **Vision** from the **standpoint** of the **way** the **solution** will **be described to** the **market**.
- ➔ A press release has to tell **a complex story** in a **simple** and **compelling** way, a way that **clearly articulates** the **benefits** to the **prospective customers**, and they also know that they have only **two pages** to do it in.



News & Press

Ping Identity Unveils PingFederate Express

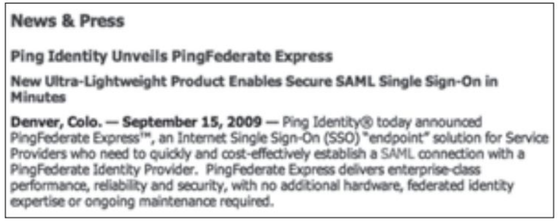
New Ultra-Lightweight Product Enables Secure SAML Single Sign-On in Minutes

Denver, Colo. — September 15, 2009 — Ping Identity® today announced PingFederate Express™, an Internet Single Sign-On (SSO) "endpoint" solution for Service Providers who need to quickly and cost-effectively establish a SAML connection with a PingFederate Identity Provider. PingFederate Express delivers enterprise-class performance, reliability and security, with no additional hardware, federated identity expertise or ongoing maintenance required.

Source: Ping Identity, Ping Federate Data Sheet

The Preliminary Press Release

- Having the team work with their marketing partners to draft a preliminary press release is a way to:
  - Foster early collaboration.
  - Illustrate that the development team can speak in the language of the customer community.
  - Paint the Vision in the minds of those key internal stakeholders who will become involved as the product approaches market readiness.



News & Press

Ping Identity Unveils PingFederate Express

New Ultra-Lightweight Product Enables Secure SAML Single Sign-On in Minutes

Denver, Colo. — September 15, 2009 — Ping Identity® today announced PingFederate Express™, an Internet Single Sign-On (SSO) "endpoint" solution for Service Providers who need to quickly and cost-effectively establish a SAML connection with a PingFederate Identity Provider. PingFederate Express delivers enterprise-class performance, reliability and security, with no additional hardware, federated identity expertise or ongoing maintenance required.

Source: Ping Identity, Ping Federate Data Sheet

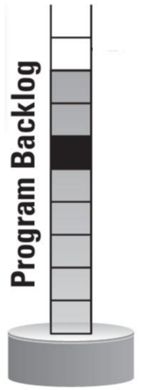
Contents

The Feature Backlog With Briefing

- In even **lighter-weight approaches**, the **backlog** alone may be **sufficient** to **communicate much** or **all** of the **Vision** to the team.

- In this case, **product managers elaborate** far enough ahead to show the team **where** the **project is headed**, while **simultaneously laying in priorities** and **estimates** for **future scoping** of **work**.

- A reasonably well-formed and maintained **backlog**—in conjunction **with** a **face-to face Vision briefing** by the product managers or other business stakeholders to the development team—can be an **adequate** way to **communicate** the **Vision**.

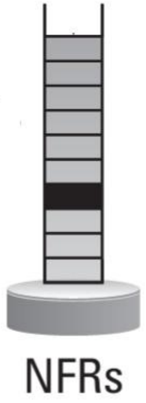


Contents

Communicating NFRs

Nonfunctional Requirements (System Qualities)

- **Many nonfunctional requirements**, which are qualities of the system in use—as opposed to specific functional behaviors—**may be equally important to the Vision**.
- These NFRs communicate attributes such as the **usability**, **performance**, **reliability** and **supportability** requirements, imposed **standards**, **compatibility** requirements, and so on.
- In some fashion, these items must also be communicated to the teams as part of the Vision and captured for future consideration.



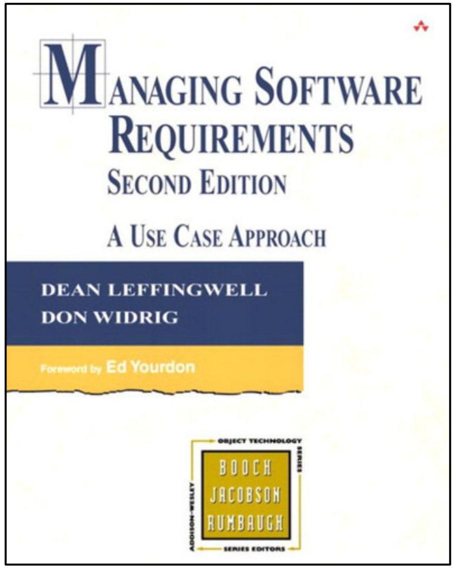
Contents

3. Features

Feature

- ➔ The **primary content** of the **Vision** is a **set** of **prioritized features**.
- ➔ They describe **what new things the system will do for its users.**
- ➔ **What benefits** the user will derive from them.
- ➔ It is more **abstract** and **higher level view** of the **system** of **interest**.

"Services provided by the system that fulfill one or more stakeholder needs."


MANAGING SOFTWARE REQUIREMENTS
SECOND EDITION
A USE CASE APPROACH
DEAN LEFFINGWELL
DON WIDRIG
Foreword by Ed Yourdon
OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH
SERIES EDITORS
ADDISON-WESLEY

Feature

- ➔ They **lived** at a **level above software requirements** and bridged the gap from the problem domain to the solution domain
- ➔ **Problem domain:** understanding the needs of the users and stakeholders in the target market
- ➔ **Solution domain:** specific requirements intended to address the user needs

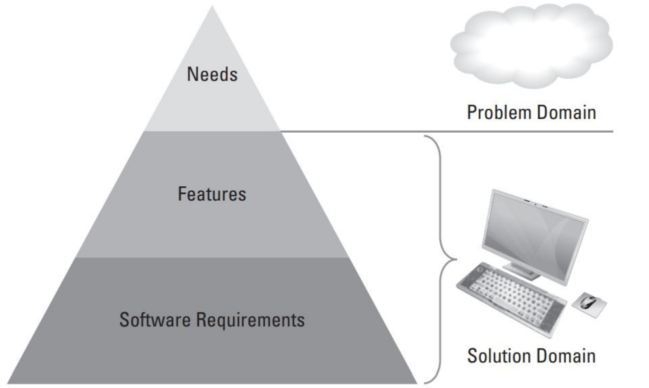


Figure 13–2 Requirements pyramid

Feature

- ➔ The word “feature or benefits” is:
  - Industry-standard norms to describe products.
  - The **language** typically used by **marketing** to describe the **capabilities** and **benefits provided** by a **new system.**
- ➔ Familiar construct in agile development bridge the language gap from the agile project team/product owner to the system/program/product manager level and give those who have traditionally operated outside our agile teams a traditional label (feature) to use to do their traditional work (describe the thing they’d like us to build).

Feature

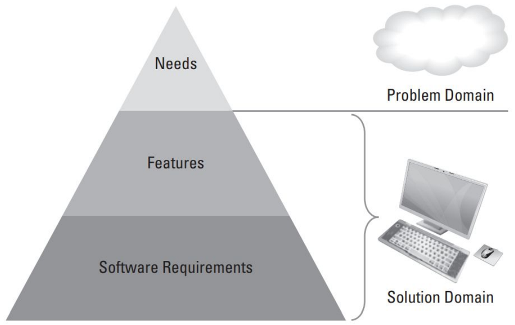
Example:

A feature of a word processor is → spell checking as you type

Expressing Features in User Story Form:

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

- ➔ The user role and benefit are more clearly described.
- ➔ However, they do look just like user stories, although they are written at a higher level of abstraction.



Contents

4. Estimating Features

a. Estimating Effort

b. Estimating Cost

c. Estimating Development Time

Estimating Features

User Story Estimation → Review

- ➔ Teams typically estimate user stories using an **abstract**, **relative** estimating **model** based on **story points**.
- ➔ Story points can be measured in the abstract (unit-less but numerically relevant) or as ideal developer days (**IDDs**), with the abstract measure being the most common.
- ➔ The aggregate amount of story points that a team can deliver in the course of an iteration is the team's **velocity**.
- ➔ When a team's size changes or vacations or holidays occur, the team adjusts the expected velocity accordingly.

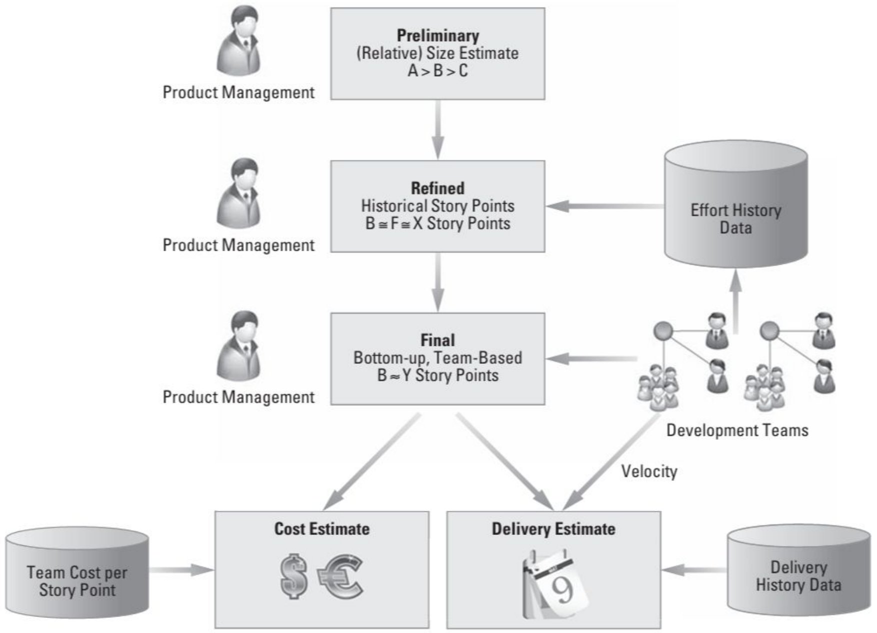
Estimating Features

User Story Estimation → Review

- ➔ If the system and organizations are largely stable, after some number of iterations, teams will generally have a fairly reliable velocity.
- ➔ That allows them to make intelligent commitments for each iteration.
- ➔ It also provides the basic mechanism we need for estimating at the program/release level.

Estimating Features

- Depending on where the item is in the program backlog and how important the estimate is, the **estimate** for a **feature** may go through a **series** of **preliminary**, **refined**, and **final estimates**.



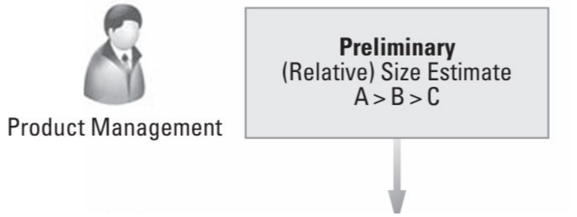
Contents

4. Estimating Features

Estimating Effort

Preliminary: Gross, Relative Estimating

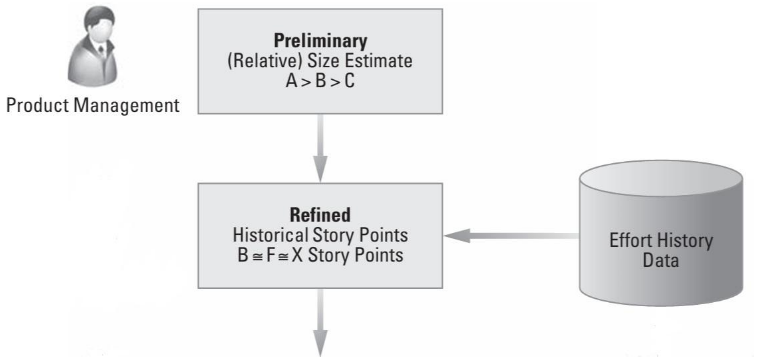
- The product management team may simply need a rough estimate of the effort to implement a feature, even before discussion with the teams.
- They can simply use **relative estimating** mode.
  - **The "bigness" of each backlog item is simply estimated relative to others of the same type.**
- It is a **simple** and **fast** estimating **heuristic.**
- It can be used for the **initial scoping** and **prioritization** of **work**.
- (But again, only relative)



Estimating Effort

Refined: Estimate In Story Points

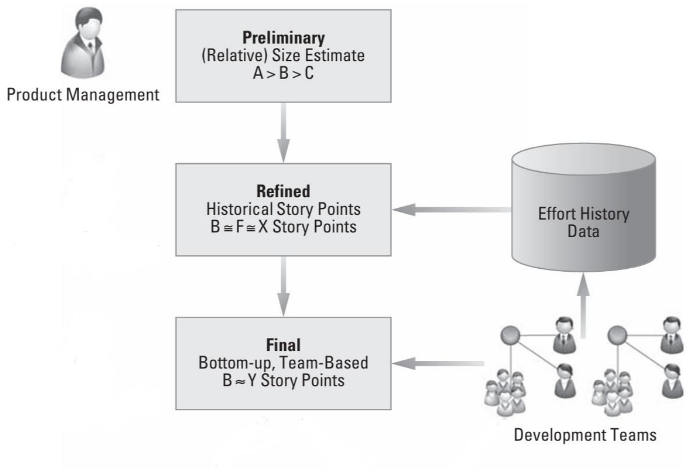
- ➔ We need **tracking** information.
- ➔ Many agile project management tools support the **feature-to-story hierarchy.**
- ➔ A simple **comparison** of the **new feature** to the **expended story points** for **a similar size feature provides** this **first refinement**.



Estimating Effort

Final: Bottom-Up, Team-Based Estimating

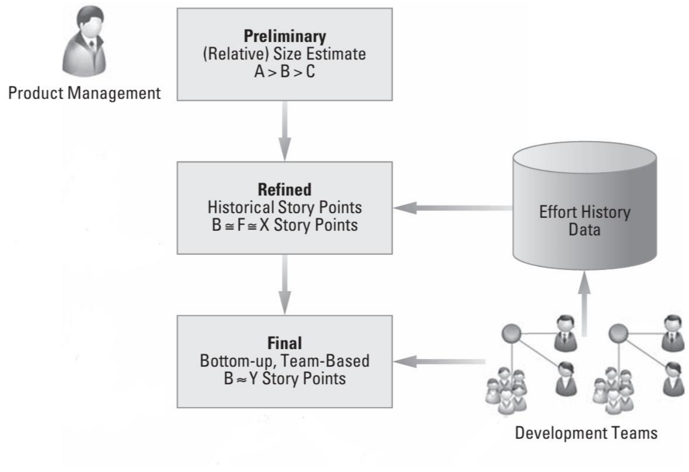
- ➔ The estimates so far require only a minor time investment and can be done by the product management team in isolation.
- ➔ That can be appropriate, based on the stage of the feature.
- ➔ However, for any meaningful estimate, the fidelity of the estimate can be significantly improved by having the estimating done by the teams.



Estimating Effort

Final: Bottom-Up, Team-Based Estimating (Cont.)

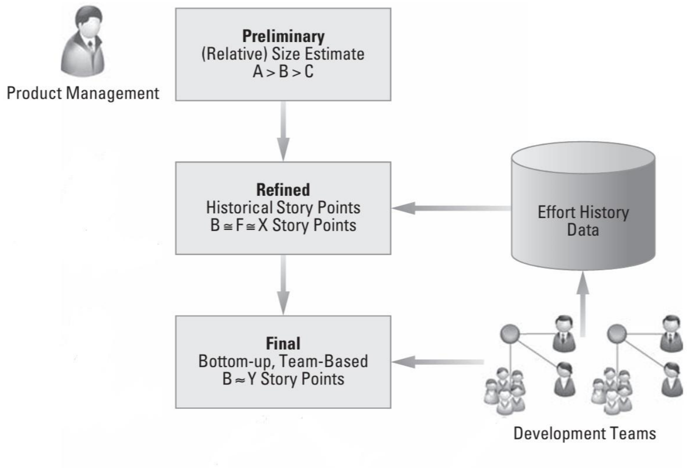
- In any material program, there will be multiple teams, and they may or may not be affected by the new feature.
- Sometimes, only they know whether their module, feature, or component is impacted.
- Therefore, only they can actually determine a more responsible estimate.
- They will typically have their own history of like features—and the story points required to complete them—in their project management repository.



Estimating Effort

Final: Bottom-Up, Team-Based Estimating (Cont.)

- ➔ However, since ad hoc requests for estimating interrupts the team from their daily iteration activities, the estimating task is most efficiently done on a cadence.
- ➔ We described the semiweekly feature preview meeting, which is designed, in part, for just this purpose.



Contents

Estimating Cost

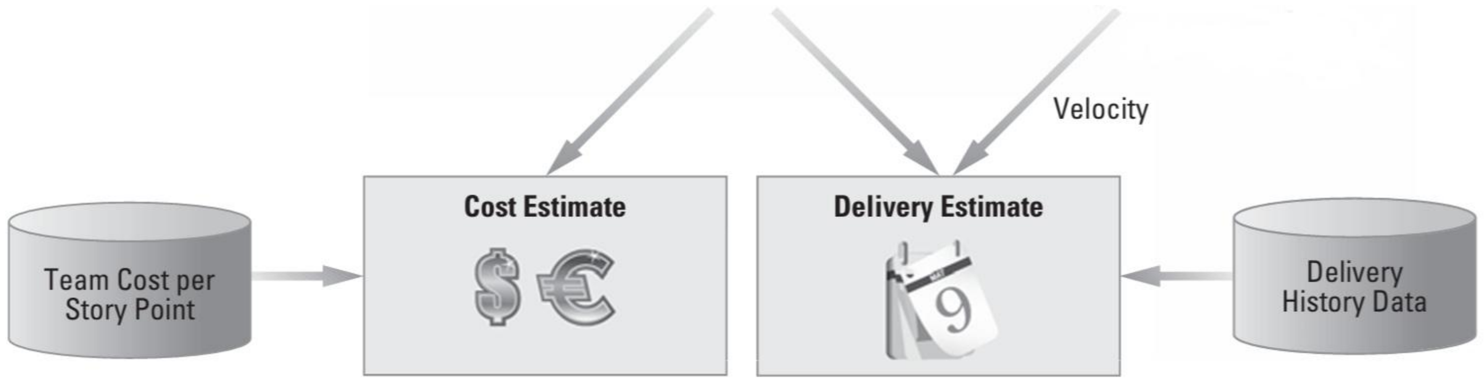
➔ **Once** the **feature** has been **estimated** in the **currency** of **story points**, a **cost estimate** can be quickly **derived**.

➔ Although the development teams themselves may not have ready knowledge of the cost of a story point, at the Program level it is fairly straightforward to calculate one.

➔ Simply look at the **burdened cost per iteration timebox 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**.

Estimating Cost



Contents

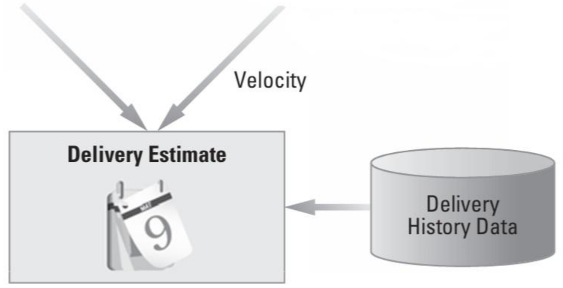
4. Estimating Features

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.

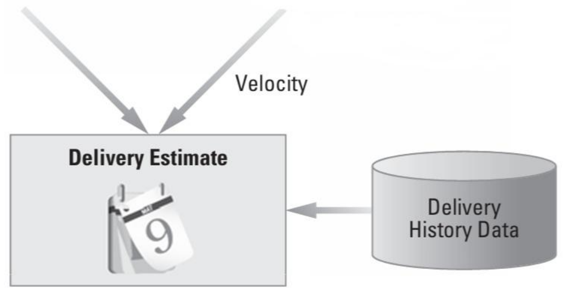
Example: Let's assume:

- ➔ **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.



Estimating Development Time

➔ As a further refinement, the **program** can also look at the **current** available **velocity** of the **affected teams**, **make some assumptions** about percentage time allocation, and **derive a better time** and **schedule estimate** from there.



Contents

5. Testing Features

Testing Features

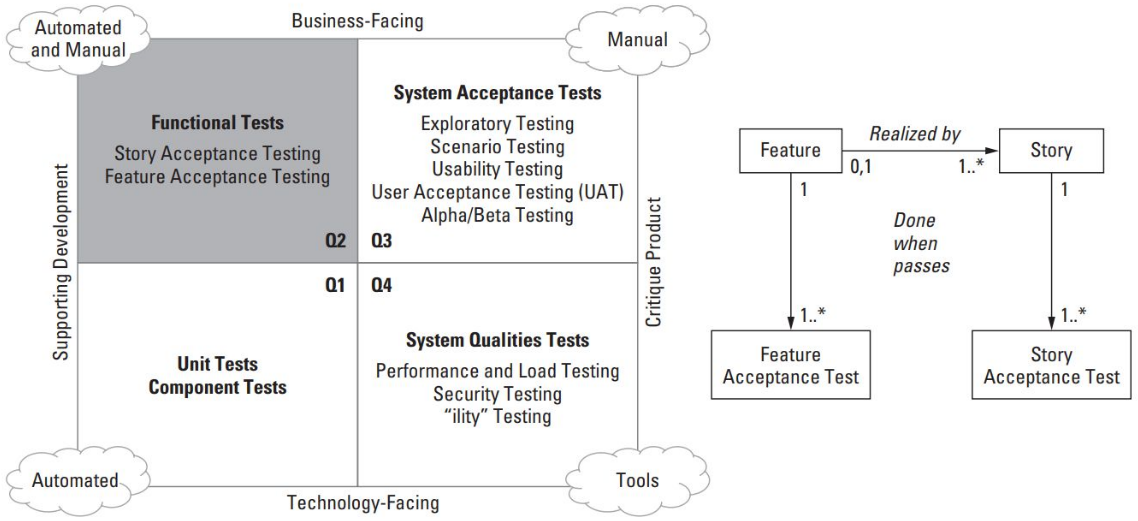


Figure 13–4 Feature testing in the agile testing matrix and the agile requirements model

Testing Features

- ➔ 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; testing of those features is often done under the purview of the **system team**.
- ➔ As with stories, feature tests may be **manual** or **automated**, with **automation** being the **preferred** approach.

Contents

6. Prioritizing Features

Prioritizing Features

- ➔ "Please tell me what your priorities are so I'll know what I don't have to work on."
  - ◆ Developer says to the product manager.
- ➔ One of the biggest challenges that all software teams face is **prioritizing requirements** for **implementation** and **delivery** to the **customers**.
- ➔ **Small decisions** can have **big impacts** on **implementation cost** and **timeliness** of **value delivery**.

Prioritizing Features

Why Prioritization Is Such A Hard Problem?

- ➔ **Customers** are seemingly **reluctant** to prioritize features.
- ➔ **Product managers** are often even **more reluctant**.
- ➔ **Quantifying value** is extremely **difficult**.
- ➔ We admit up front that we **can't implement** (nor even **discover**) **all potential requirements**.
  - ◆ After all, we typically have fixed quality, resources, and delivery schedules.
  - ◆ Therefore, the only variable we have is scope.
- ➔ **Effective prioritization** becomes a mandatory **practice** and **art**—one that must be mastered by every agile team and program.

Prioritizing Features

- There is certainly **no one right way to prioritize**, and teams will benefit from differing perspectives on this unique problem.

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

- 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**.
- After all, who wouldn’t want to deliver a higher ROI feature before a lower ROI feature?
- That seemed to make totally intuitive and (apparently) economical common sense.

Contents

Prioritizing Features Based On The Cost Of Delay

Reinertsen: *"If you only quantify one thing, quantify the cost of delay."*

➔ Because we have already outlined an estimating strategy, we'll actually be able to quantify two things:

1. The feature effort estimate
2. The cost of delay

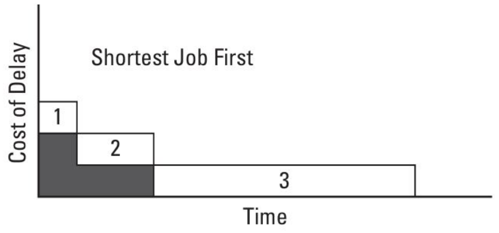
➔ Reinertsen describes three methods for prioritizing work based on the economics of CoD:

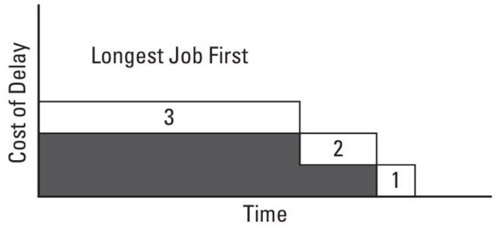
1. Shortest job first
2. High delay cost first
3. Weighted shortest job first

Prioritizing Features Based On The Cost Of Delay

Shortest Job First

➔ **When** the cost of delay (**CoD**) for **two features** is **equal**, doing the **Shortest** (in our case, smallest) **Job First**, produces the **best economic returns**.





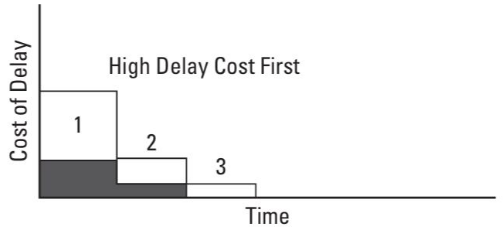
Feature	Effort	Cost of Delay
1	1	3
2	3	3
3	10	3

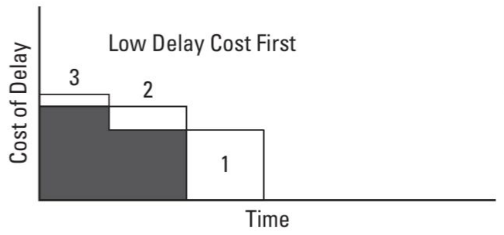
▆ Delay Cost

Prioritizing Features Based On The Cost Of Delay

High Delay Cost First

- If **two features** have the **same CoD**, do the **smallest feature first**.





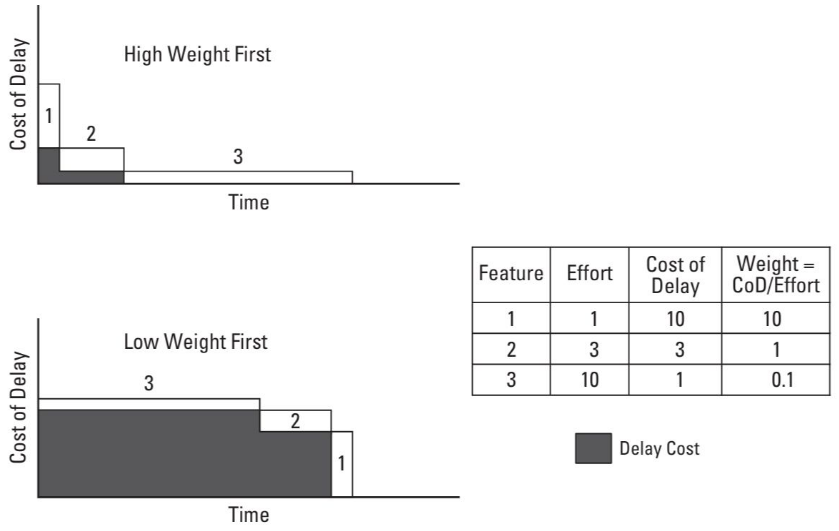
Feature	Effort	Cost of Delay
1	3	10
2	3	3
3	3	1

▇ Delay Cost

Prioritizing Features Based On The Cost Of Delay

Weighted Shortest Job First

- ➔ 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

Contents

Estimating The Cost Of Delay

How Does One Go About Calculating The Cost of Delay For A Feature?

➔ We suggest that **CoD is** an **aggregation** of **three attributes** of a **feature**, each of which can be estimated fairly readily, when compared to other features.

- User value
- Time value
- Risk reduction value

Estimating The Cost Of Delay

How Does One Go About Calculating The Cost of Delay For A Feature?

- **User Value:**
  - It is simply the potential **value** of the **feature** in the **eyes** of the **user**.
  - **Product managers** often have **a good sense** of the **relative value** of a **feature** (“they prefer this over that”), even when it is impossible to determine the absolute value.
  - And since we are prioritizing like things, relative user value is all we need.

Estimating The Cost Of Delay

How Does One Go About Calculating The Cost of Delay For A Feature?

- **Time Value**
  - It is another relative estimate, one based on **how** the **user value decays over time**.
  - Many features provide higher value when they are delivered early and differentiated in the market and provide lower value as features become commoditized.
  - In some cases, time value is modest at best (implement the new UI standard with new corporate branding).
  - In other cases, time value is extreme (implement the new testing protocol prior to the school year buying season), and of course there are in-between cases as well (support 64-bit architectures as soon as our competitors do).

Estimating The Cost Of Delay

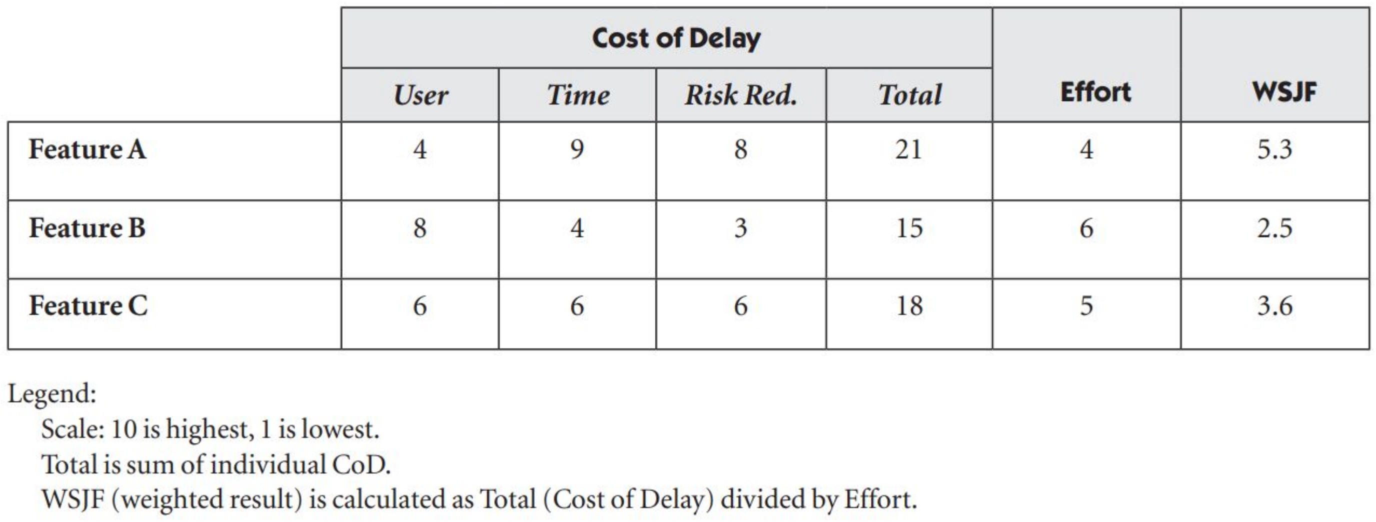
How Does One Go About Calculating The Cost of Delay For A Feature?

- ➔ **Risk Reduction**
  - ◆ One that acknowledges that what we are really doing is software research and development.
  - ◆ Our world is laden with both risk and opportunity. Some features are more or less valuable to us based on how they help us unlock these mysteries, mitigate risk, and help us exploit new opportunities.
  - ◆ For example, move user authentication to a new web service could be a risky effort for a shrink wrapped software provider that has done nothing like that in the past, but imagine the opportunities that such a new feature could engender.

Contents

6. Prioritizing Features

Feature Prioritization 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.

Contents

Achieving Differential Value

The Kano Model of Customer Satisfaction

➔ Noriaki Kano, an expert on the field of quality management and customer satisfaction, developed a model for customer satisfaction that also challenged some traditional beliefs.

➔ 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.

Achieving Differential Value

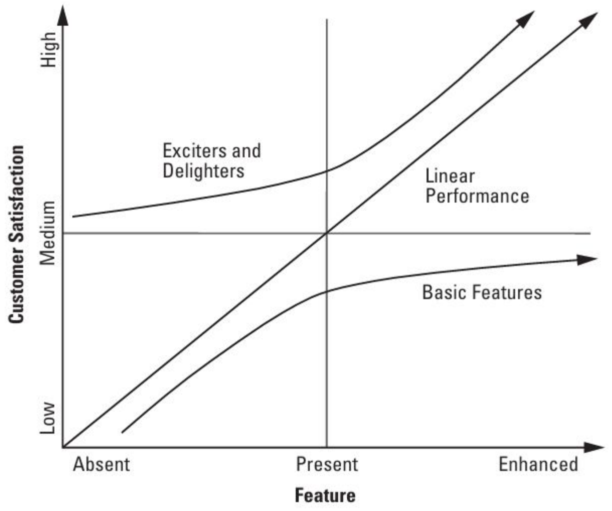


Figure 13–8 Kano model of customer satisfaction

Achieving Differential Value

Prioritizing Features For Differential Value

Given Weighted Shortest Job First prioritization model, what additional benefit we can derive from Kano's thinking?

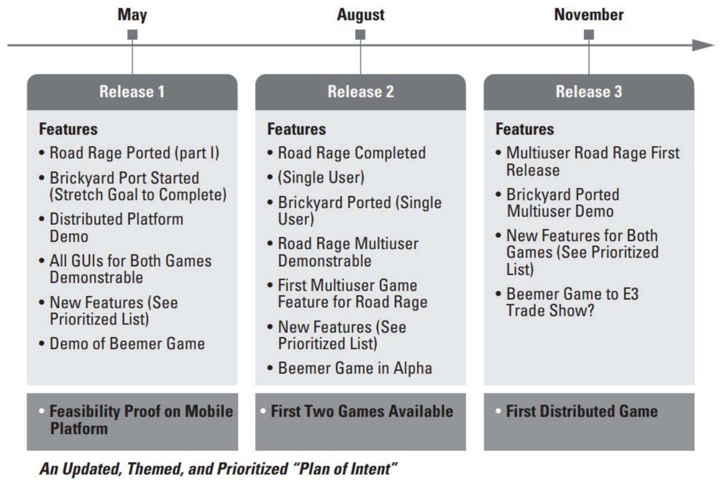
- Differential value rule #1:
  - **Invest in MMFs**, but never overinvest in a feature that is already commoditized.
- Differential value rule #2:
  - **Drive innovation** by having the courage to invest in exciters.
- Differential value rule #3:
  - If **resources** do **not allow** you to **compete** on the current playing field, **change** the **playing field**.

Contents

7. The Roadmap

The Roadmap

- ➔ In order to **set priorities** and **plan** for **implementation**, we need an additional perspective, a product Roadmap that provides **a view** we can use to **communicate future objectives** to **our outside stakeholders**.

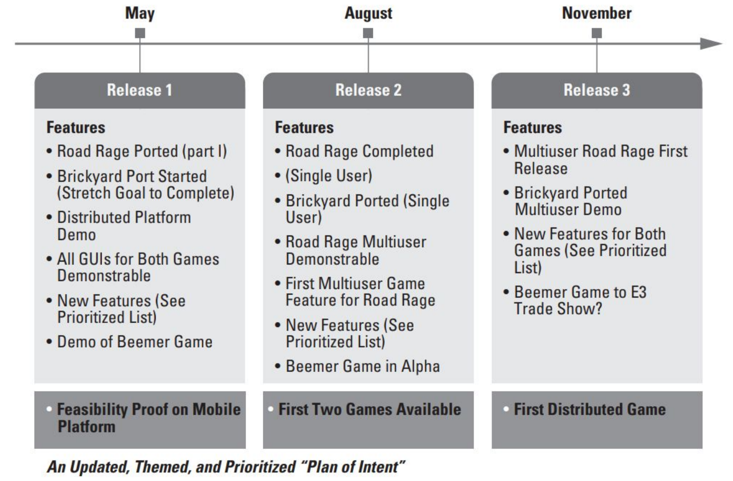


May
August
November
Release 1
Features
• Road Rage Ported (part I)
• Brickyard Port Started (Stretch Goal to Complete)
• Distributed Platform Demo
• All GUIs for Both Games Demonstrable
• New Features (See Prioritized List)
• Demo of Beemer Game
Release 2
Features
• Road Rage Completed
• (Single User)
• 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"

Figure 13–9 Roadmap—a themed, prioritized "plan of intent"

The Roadmap

- ➔ Each **vertical box** represents an **upcoming release** (or PSI).
- ➔ The **label** at the **bottom represents** the **theme** or **primary objective** of the **release**.
- ➔ The **features** are **listed** in **prioritized order**.
- ➔ The **dates**, **themes**, and **quality** for the **next release** are **fixed**.
- ➔ The **features** are **prioritized** and **variable**.



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"

Figure 13–9 Roadmap—a themed, prioritized "plan of intent"

The Roadmap

- ➔ The **teams** can **commit only** to the **features** in the **next upcoming release**.
- ➔ **Releases beyond** the **next** represent only a **best estimate**.
- ➔ The **Roadmap**, then, is a "**plan of intent**" and **is subject to change** as development facts, business context, and customer needs change.

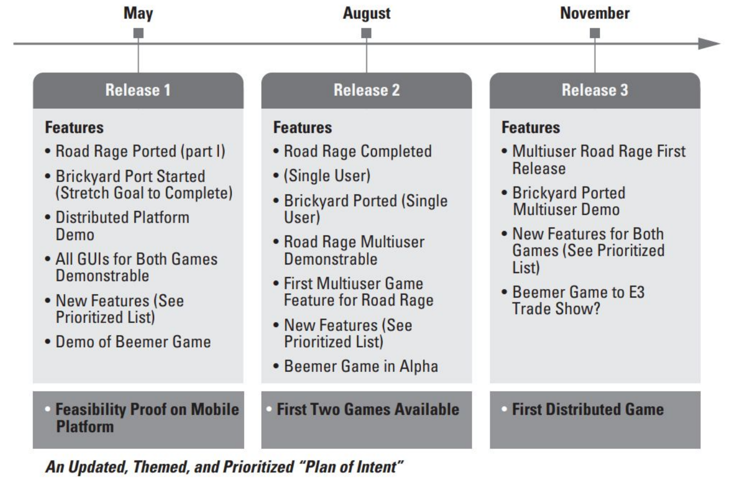

May
August
November
Release 1
Features
• Road Rage Ported (part I)
• Brickyard Port Started (Stretch Goal to Complete)
• Distributed Platform Demo
• All GUIs for Both Games Demonstrable
• New Features (See Prioritized List)
• Demo of Beemer Game
• Feasibility Proof on Mobile Platform
Release 2
Features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

Figure 13–9 Roadmap—a themed, prioritized "plan of intent"

End of Chapter 13

Contributions

- ➔ Author of Reference Book: **Dean Leffingwell**
- ➔ Course Instructor: **Mehran Rivadeh**
- ➔ Slide Creator: **Mahnaz Rasekhi**
    - ◆ These slides are primarily based on Agile Software Requirements by Dean Leffingwell, with occasional adaptations to enhance clarity and engagement.