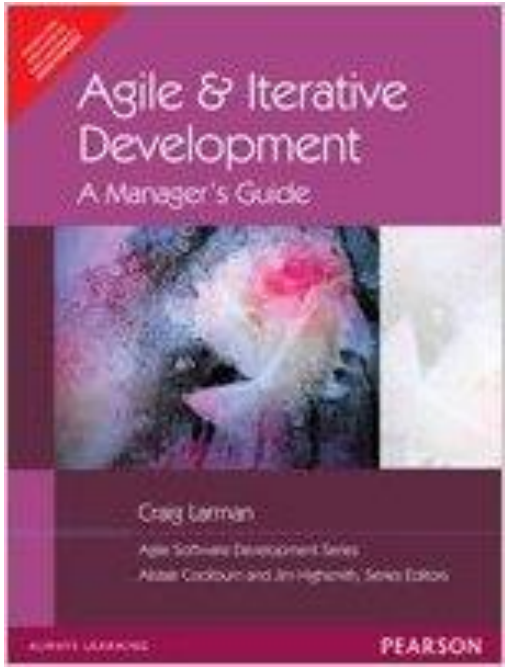
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Requirements Management in Agile

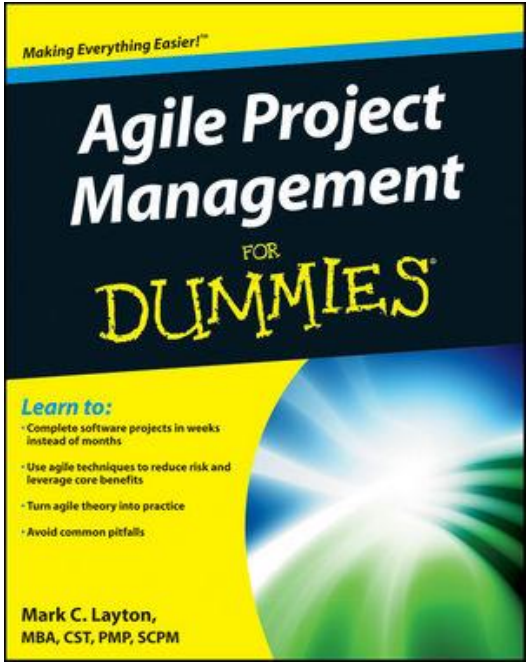
- Prof K G Krishna

Text/Reference Books

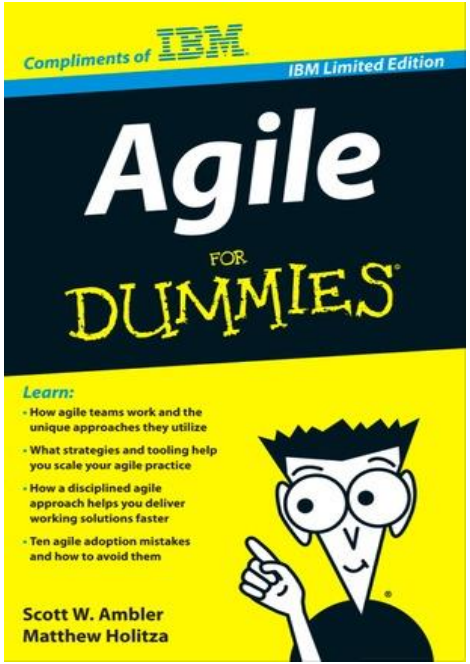
T1


Agile & Iterative Development
A Manager's Guide
Craig Larman
Agile Software Development Series
Alistair Cockburn and Jim Highsmith, Series Editors
ALWAYS LEARNING
PEARSON

T2


Making Everything Easier!™
Agile Project Management
FOR
DUMMIES®
Learn to:
· Complete software projects in weeks instead of months
· Use agile techniques to reduce risk and leverage core benefits
· Turn agile theory into practice
· Avoid common pitfalls
Mark C. Layton,
MBA, CST, PMP, SCPM

Compliments of IBM


Compliments of IBM
IBM Limited Edition
Agile
FOR
DUMMIES®
Learn:
· How agile teams work and the unique approaches they utilize
· What strategies and tooling help you scale your agile practice
· How a disciplined agile approach helps you deliver working solutions faster
· Ten agile adoption mistakes and how to avoid them
Scott W. Ambler
Matthew Holitza

➔ As this field is evolutionary, the student is advised to stay tuned to the current and emerging practices by referring to their own organization's documentation as well as Net sources

Topics

Agile Requirements Management

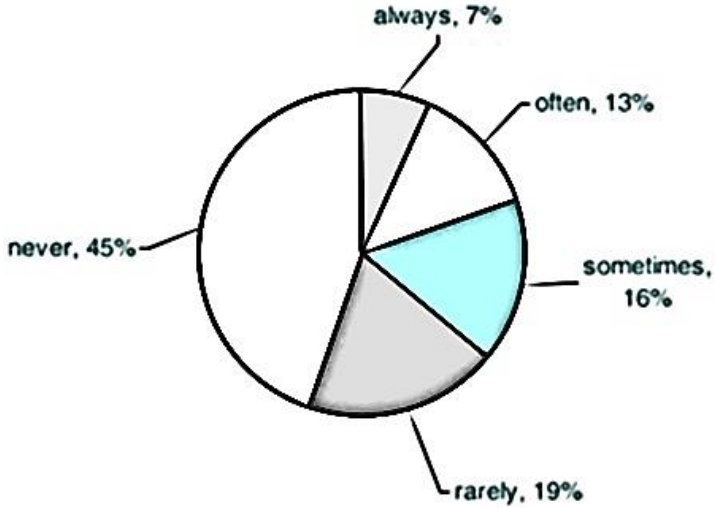
- Managing Requirements Iteratively
- User Stories as Requirements
- Size/Effort Estimation
- Prioritization Techniques
- Preparing the Product Roadmap

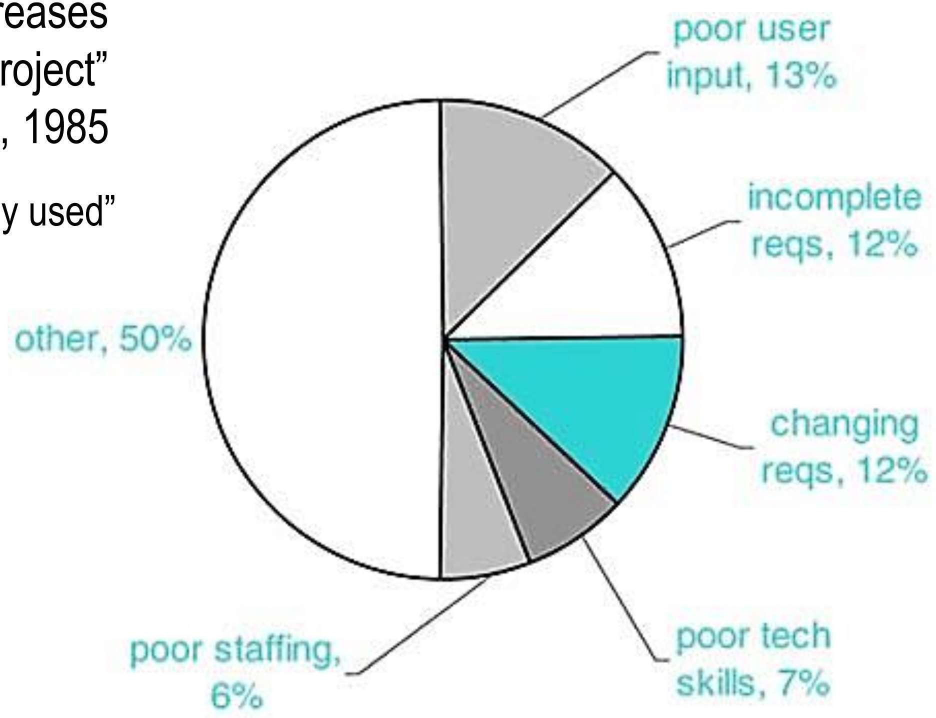
EXTREME PROGRAMMING
I CAN'T GIVE YOU ALL OF THESE FEATURES IN THE FIRST VERSION.
scottadams@aol.com
www.dilbert.com
AND EACH FEATURE NEEDS TO HAVE WHAT WE CALL A "USER STORY."
© 2002 United Feature Syndicate, Inc.
1/10/03
OKAY, HERE'S A STORY: YOU GIVE ME ALL OF MY FEATURES OR I'LL RUIN YOUR LIFE.

37% of Project Challenges are in *Requirements Management*

“Cost of fixing a Requirement defect increases non-linearly from early to late in the Project”
- B. Boehm, 1985

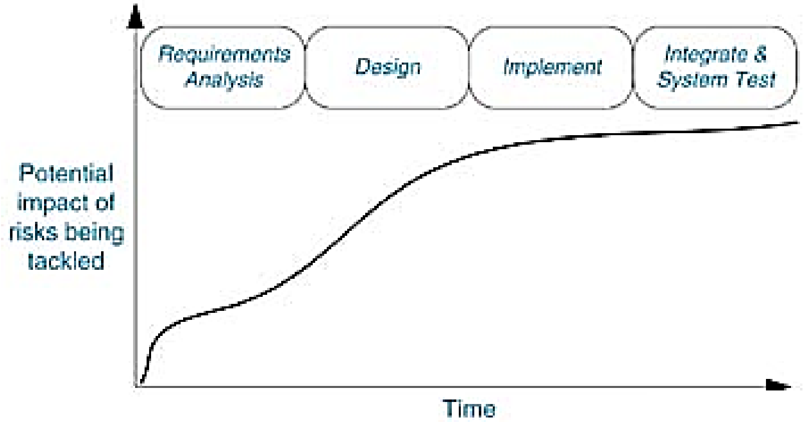
“45% of Features never used and ~20% rarely used”





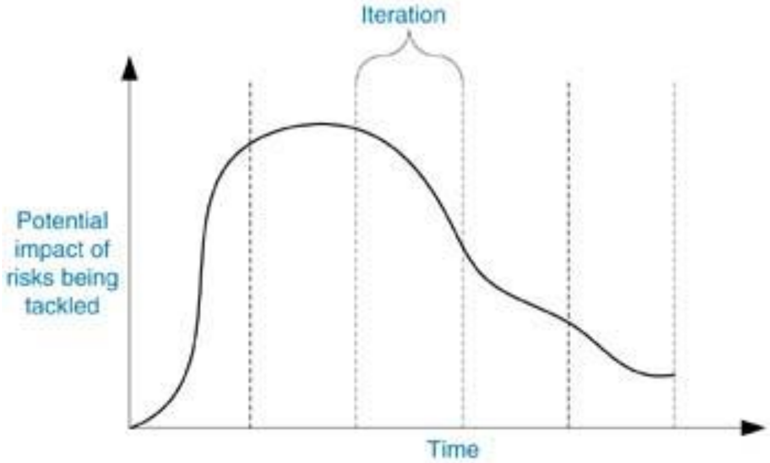
Source: (T1) / Standish Group Report, 2004

Incremental & Iterative Development (IID) is The Way To Go



In a waterfall lifecycle, high-risk issues such as integration and load test are tackled late.

Risk Profile in Waterfall Model



Risk Profile in IID

In an iterative lifecycle, high-risk issues are tackled early, to drive down the riskiest project elements.

Source: (T1-Chap 5)

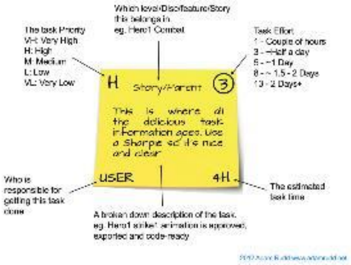
Identifying Product Requirements in Agile

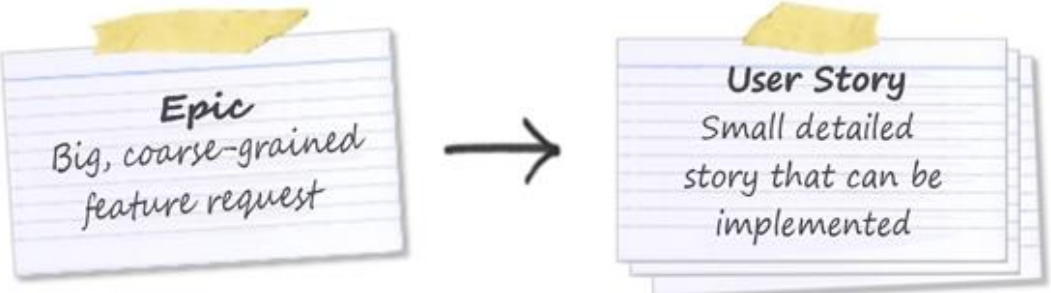
Source: http://eleventwenty.com

- From the Product Roadmap, arrive at Initial Set of High-priority Requirements
  - Product Roadmap ➔ Requirements ➔ Releases ➔ Sprints
- Requirements ➔ Logical Groups
- Decompose Requirements into: (level of detailing depends on early product definition)
  - **Themes**: logical grouping of features into Themes (Requirements at the highest level, e.g., Account Info, Transactions, Support functions,…)
  - **Features**: describe capability of the Product (part of the Product, e.g, view balance, pay bills, reset password, transfer money,…)
  - **Epic User Stories**: large set of Requirements that support a Feature containing multiple actions
  - **User Stories**: containing single action enough to start implementing (~ use-cases, scenarios)
  - **Tasks**: execution steps required to develop a story; breakdown User Story into Tasks during Sprint planning

Building *Product-backlog*

- Product-backlog comes to life soon after identification of first Requirement (Theme/Feature/Story)
- User Story - an Expression of Requirement of Business Value
- Group Features (into Themes) by Technical similarity, Usage flow, Business need, etc.
- Use Index cards / Sticky Post-it notes for easy shuffling between Themes, Sprints and Product back-logs
- A Meeting of Stakeholders (Customer) for Identifying and Grouping of Requirements


Which level/Disc/feature/Story this belongs in eg. Hero1 Combat
The task Priority
VH: Very High
H: High
M: Medium
L: Low
VL: Very Low
Task Effort
1 - Couple of hours
3 - ~Half a day
5 - ~1 Day
8 - ~ 1.5 - 2 Days
13 - 2 Days+
H
Story/Parent
3
This is where all the delicious task information goes. Use a Sharpie so it's nice and clear
USER
4H
Who is responsible for getting this task done
The estimated task time
A broken down description of the task. eg. Hero1 strike animation is approved, exported and code-ready


Epic
Big, coarse-grained feature request
User Story
Small detailed story that can be implemented

Source courtesy: http://www.bloomy.com

Size/Effort Estimation in Agile ➔ ➔

Relative Effort Estimation

- Estimation & Ordering of Requirements commence soon after they are identified and arranged into logical Groups
- *Effort* in Agile is not exact quantitative estimate, but an assessment of the *ease or difficulty* of implementing the Requirement
- Ordering and Prioritizing is about determining its *value* in relation to other Requirements
- *Value* implies how beneficial (customer value proposition) the Requirement is to the Product (when released to users)
- Ordering of Requirements considers logical dependencies

Relative Scoring of Requirements

- Relative Scoring of Requirements using “Fibonacci Sizing Sequence”

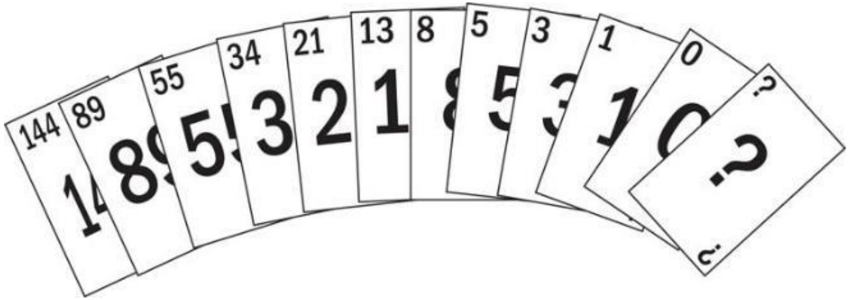
  1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

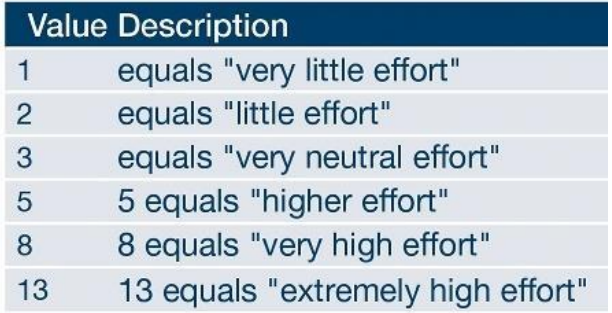
  - The effort-scores of Features (at high-level) during Product Roadmap creation will be high in the range, say 55 to 144
  - When the above are broken down into epic user stories, their scores will be in the range of 13 to 34
  - Upon further break down into low-level User Stories (ready for implementation), their scores should have effort scores between 1 and 8
- Scoring is Relative (Value & Effort)
  - Chose a Requirement that a Project Team can agree has a small value and effort and score it, and use that Requirement as a Benchmark for furthering scoring of other Requirements
  - Use two separate Benchmarks for Value and Effort to calculate Relative Priority

Effort Estimation by “Poker Estimation *Game*”

- Estimation Poker (aka Planning Poker) is a fun game to determine Story size and build consensus
- Scrum Master acts as a Facilitator and Product Owner provides reads the Story / provide details about the Feature to be estimated
- The Card deck contains cards with numbers of the Fibonacci sequence

Why Fibonacci?
“Fibonacci series represents a set of numbers that we can intuitively distinguish between them as different magnitudes…”??



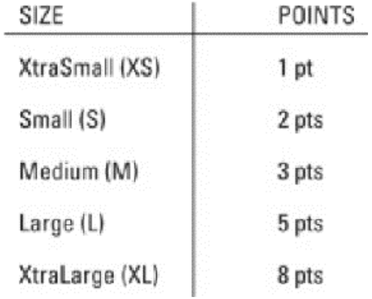


Value	Description
1	equals "very little effort"
2	equals "little effort"
3	equals "very neutral effort"
5	5 equals "higher effort"
8	8 equals "very high effort"
13	13 equals "extremely high effort"

Source: www.platinumedge.com/estimationpoker

Poker Estimation (by Consensus) contd.,

- Agree on Point-scale of about Six Numbers (representing Story Points)
- Product Owner explains the User Story and provides relevant Information
- Team (Players) briefly discusses the Story and guesses an estimate (Story Points)
- Everyone silently selects one card (they felt represent the ‘effort’) and lays the card face-down
- Once each Player selects a card, all players turn-over their cards simultaneously
- If the Players have different Story points, it’s time for discussion; if the Players do not agree on any one estimate, it’s time for Scrum Master to mediate and decide or determine that the User Story needs more detailing
- The above steps are repeated for each User Story to arrive at the collectively agreed Story Point estimates for all
- When number of Stories are large, use **Affinity Estimating** – group Stories of similar affinity (effort value) and apply Poker Estimation to these categories (e.g. *Extra-small, Small, Medium, Large, Extra-large, Epic user story* that is too large to com into the Sprint)



SIZE	POINTS
XtraSmall (XS)	1 pt
Small (S)	2 pts
Medium (M)	3 pts
Large (L)	5 pts
XtraLarge (XL)	8 pts

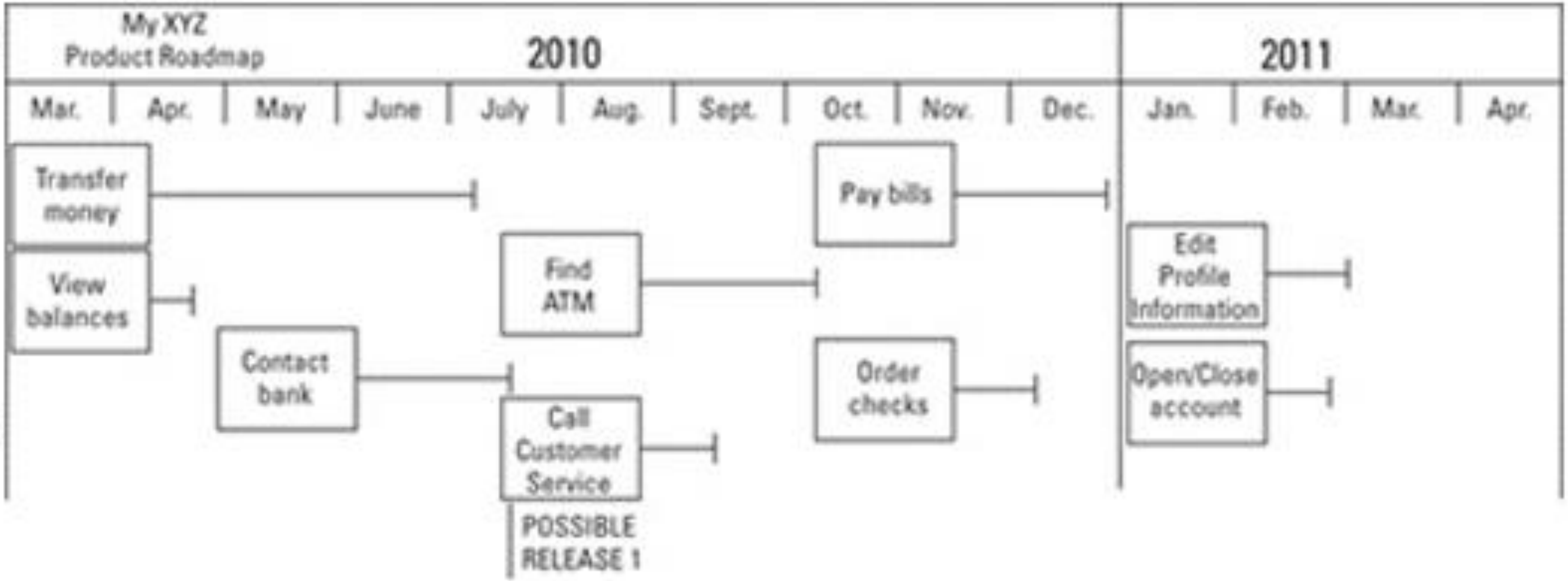
Estimates should be for the total ***Done*** – Developed, Integrated, Tested and Documented

Relative Prioritizing of Requirements

- After having Effort and Value scores for each Requirement, calculate *Relative Priority* as *Value/Effort* (and round the value to integer)
- A Requirement with High Value and Low Effort will have High Relative Priority compared to the one with Low Value and High Effort
- Relative Priority is just a mathematical idea to base decisions – however, any other equivalent technique can be used as well
- To determine the Overall Priority answer the questions:
  - What is the Relative Priority (refer the above calculation)
  - What are the Prerequisites for any Requirement
  - What set of Requirements constitute a set for Release (of relative high value)

Build Product Roadmap with *Prioritized* Requirements

- Build Product-backlog with Feature set, and start arranging as per the Relative Priority computed



Source: (T2-Chap 7)

Summary: Agile Requirements

- Continuous **Requirements Churn** is the Key Motivation for going Agile
- Requirements are **Evolutionary** in Agile Projects – they continue to Change till the Last Release/Sprint
- Requirements are organized into *Themes→Features→Epic User Stories→User Stories→Tasks*
- All Size/Effort Estimations in Agile are **Relative** with baseline Estimation of a small User Story (unit of Requirements Capture/Estimation)
- Estimations are made by **Consensus** (using *Poker Estimation, Affinity Estimation*)
- **Relative Prioritization** of Estimates (by Value) helps in building Product Roadmap (→ Product-backlog)
- Requirements are **Managed at every Planning Stage** in Agile – from Product Roadmap (Product-backlog) to Release Planning (Release-backlog) to Sprint Planning (Sprint-backlog)

Thank You