BITS Pilani

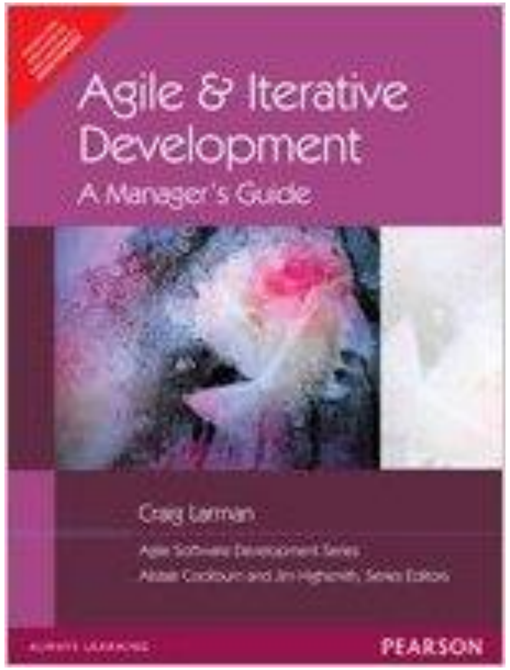
Pilani | Dubai | Goa | Hyderabad

Agile Software Development

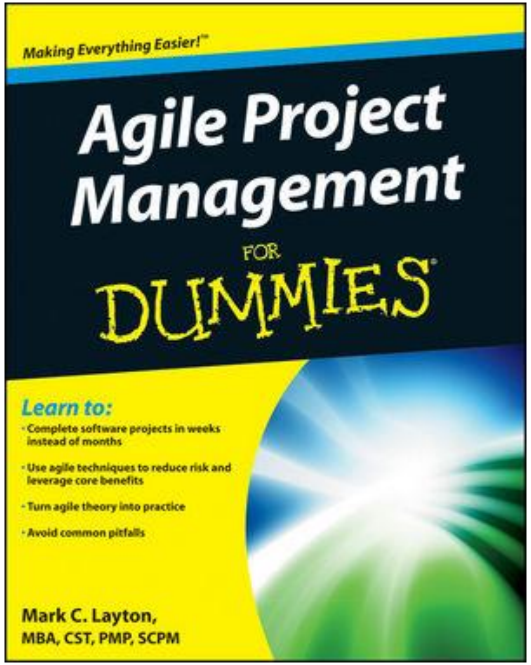
- Prof K G Krishna

Text/Reference Books

T1

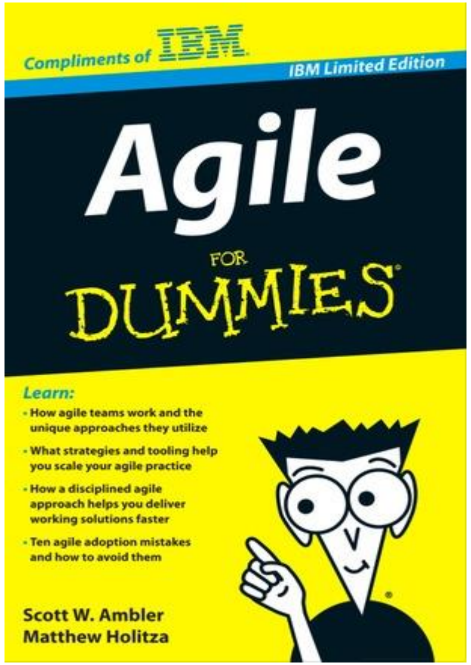

Agile & Iterative Development
A Manager's Guide
Craig Larman
Agile Software Development Series
Alistair Cockburn and Jim Highsmith, Series Editors
PEARSON

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

Basics of Agile Software Development

- Iterative and Incremental Approaches
- Risk driven and client driven development
- Time-boxed development
- Adaptive and Evolutionary development

WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.

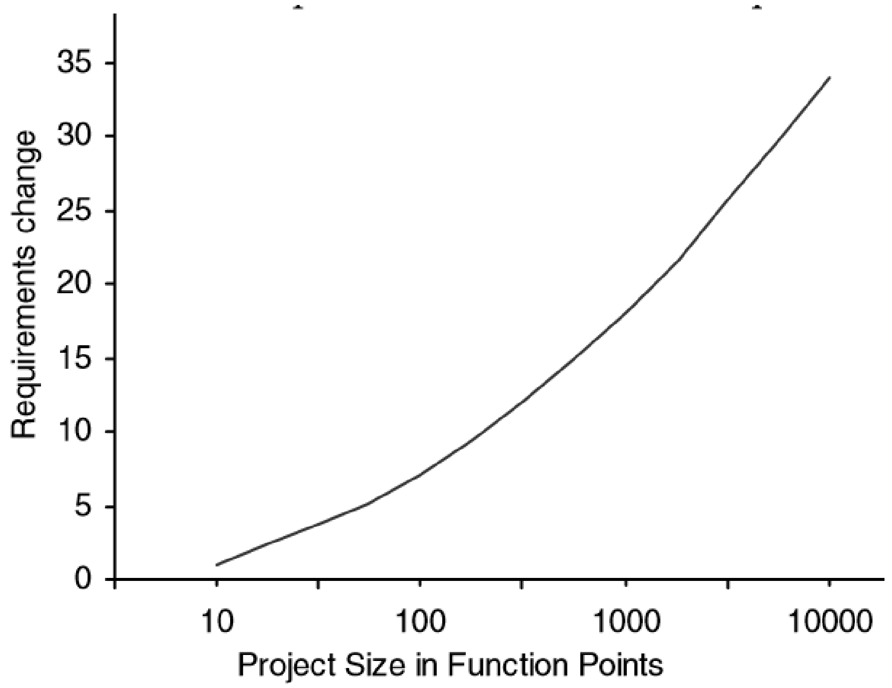
© Scott Adams

Customer Requirements? Hard To Get!

“Ours is a world where people don't know what they want and are willing to go through hell to get it.”
—Don Marquis

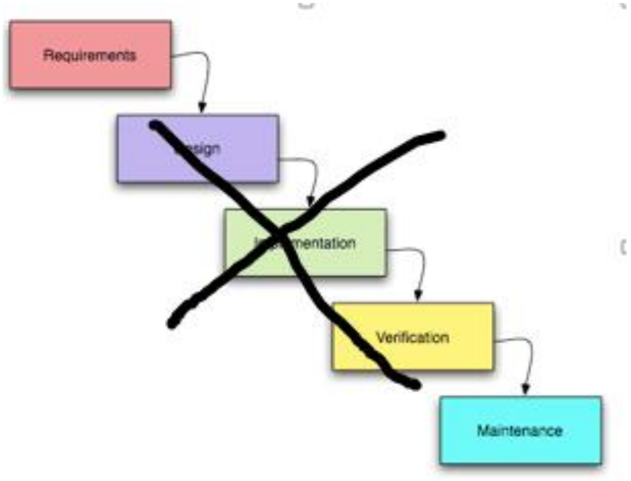
- “Requirements are capabilities and conditions to which the system—and more broadly, the project—must conform”
- “A prime challenge of requirements analysis is to find, communicate, and remember (that usually means write down) what is really needed, in a form that clearly speaks to the client and development team members.”
- More than 50% of Requirements keep changing through the Development cycle (particularly true while working with Oriental Customers like Japanese)

Waterfall is nightmare…Don’t Go For It!

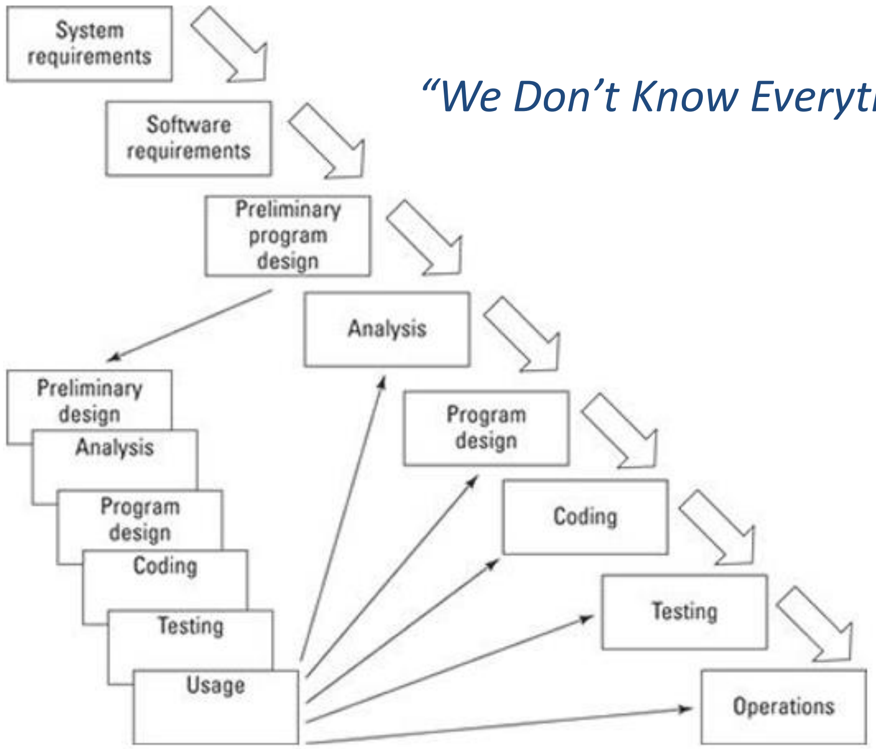


The ills of Waterfall…

- Originated in the Mainframe era (suited for Large Enterprise Projects)
- Freezing Requirements Upfront (*Reality is different*)
- Long Project Life-cycles (>>2 years)
- Lack of Transparency and Visibility to Customer
- "Last-minute surprises to Customer upon Delivery"
- Documentation overhead ("non-value-adding?")
- "Work fills available schedule"
- Hierarchical Team Structures
- …



Iterations in Waterfall? No Good Either...

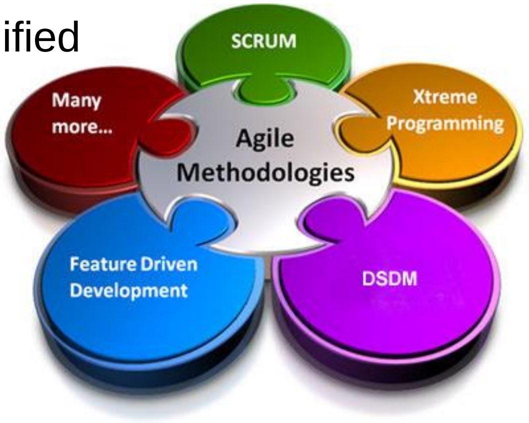


"We Don't Know Everything When We Start The Project..."

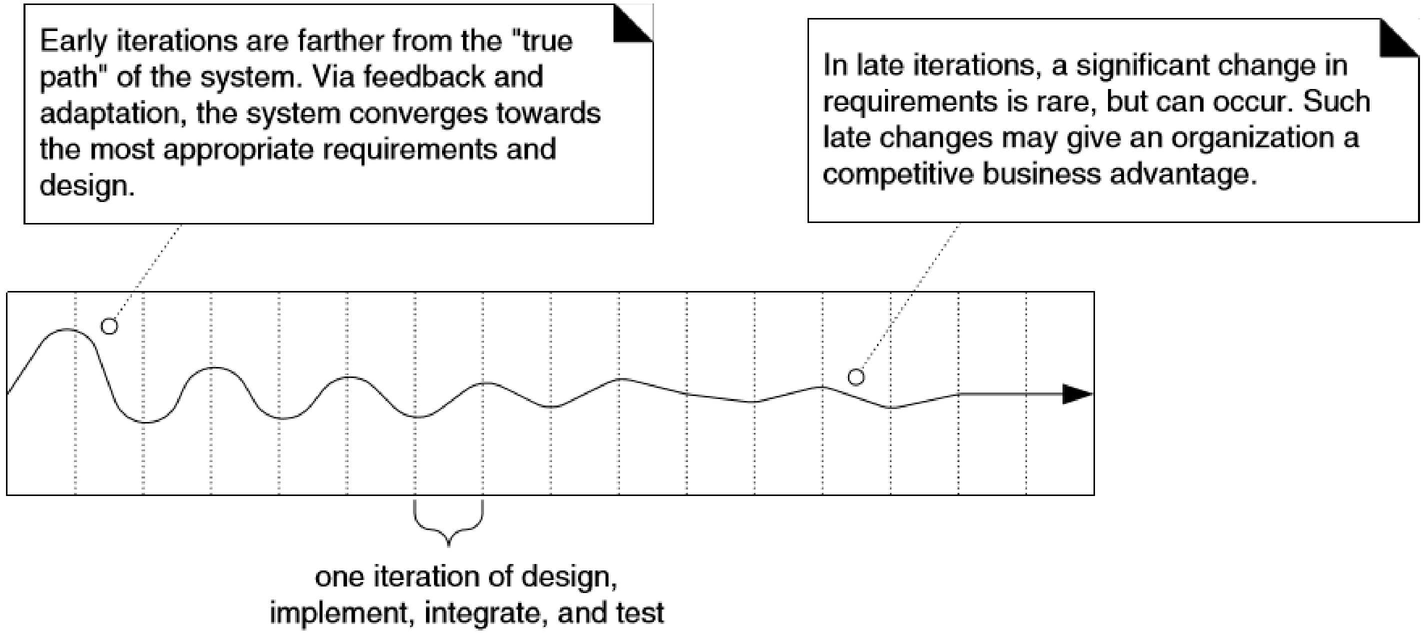
Iterative & Incremental…is the Way To Go!

You should use iterative development only on projects that you want to succeed.
—Martin Fowler

- Early Programming & Testing of Partial System, in Repeating Cycles in Agile vs. Early Upfront Speculative Requirements Freeze before Programming in Waterfall Models
- Refinement of the System through Successive Fixed-length Iterations (mini-projects or Sprints)
- Common **Agile Processes**: Scrum, Lean Development, Unified Process, Test-Driven Development (TDD), Feature-Driven Development (FDD), Adaptive Software Development, etc.



Iterations *Converge* to True Requirements!



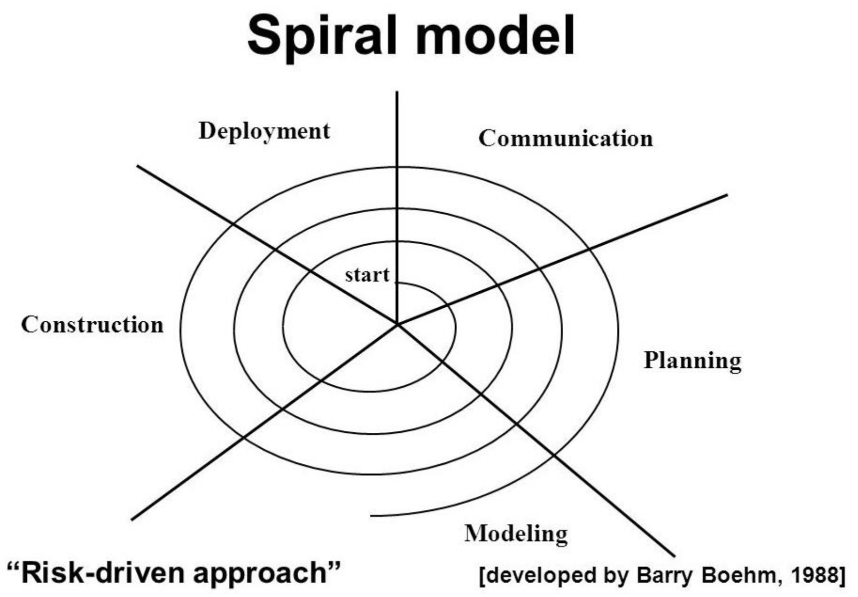
Early iterations are farther from the "true path" of the system. Via feedback and adaptation, the system converges towards the most appropriate requirements and design.

In late iterations, a significant change in requirements is rare, but can occur. Such late changes may give an organization a competitive business advantage.

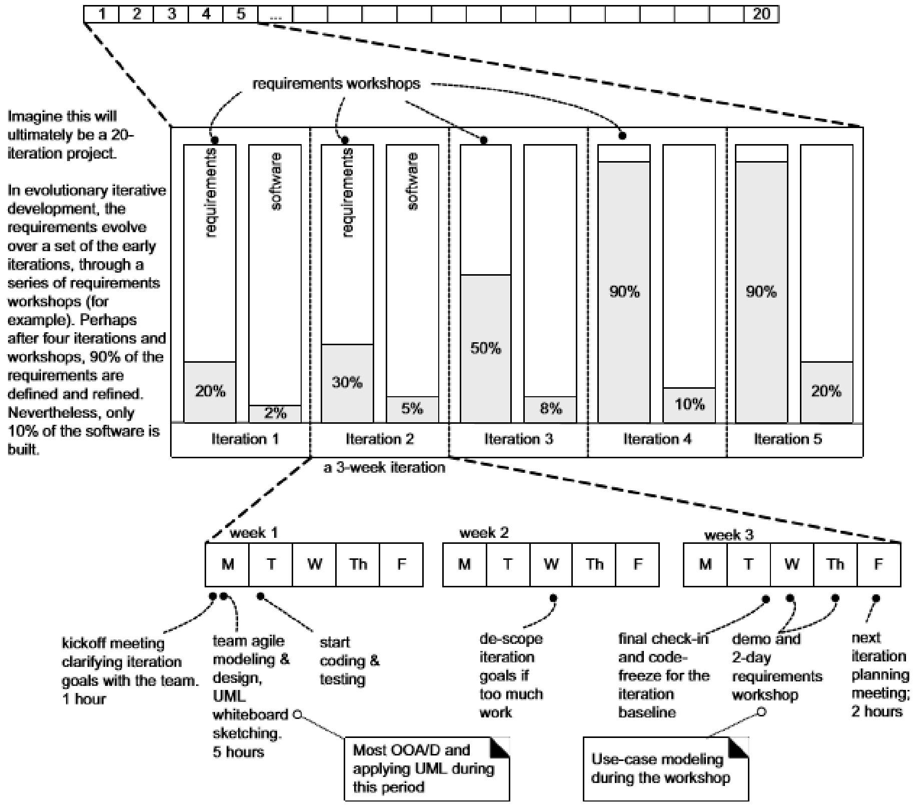
NO 'Waterfall Thinking' in Iterative Development! "…on average 45% of the features in waterfall requirements are never used, and early waterfall schedules and estimates vary up to 400% from the final actuals…"

Risk-Driven, Customer-Driven Iterative Planning

- Early Iterations with Highest Risk
- Ensure Early Visibility of Key Features
- Focus on Stabilizing Architectural Choices

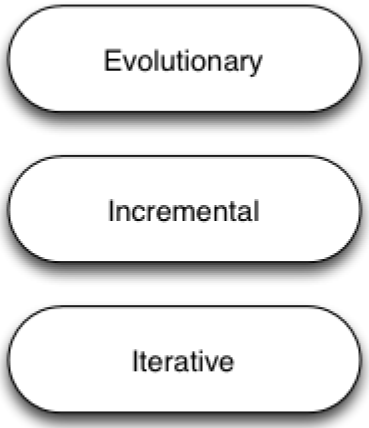


Evolutionary Analysis & Design in Early Iterations

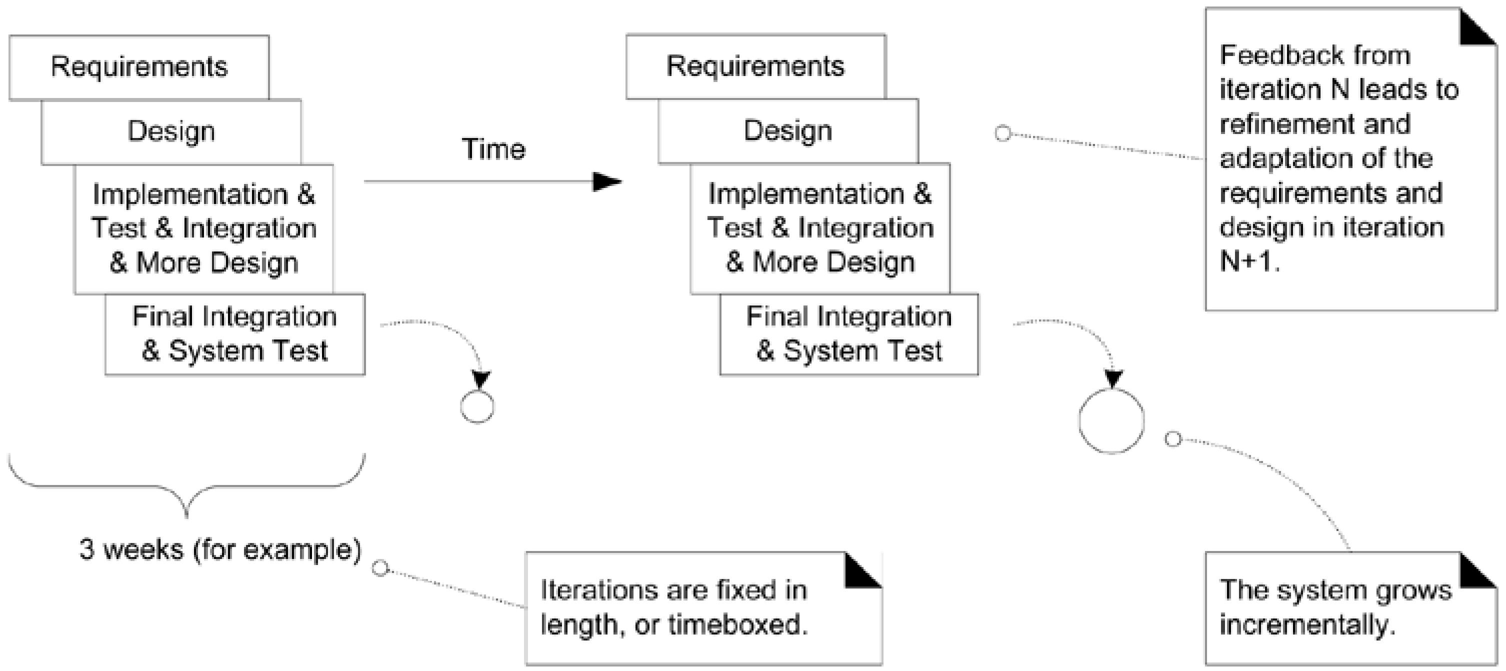

1
2
3
4
5
...
20
requirements workshops
Imagine this will ultimately be a 20-iteration project.
In evolutionary iterative development, the requirements evolve over a set of the early iterations, through a series of requirements workshops (for example). Perhaps after four iterations and workshops, 90% of the requirements are defined and refined. Nevertheless, only 10% of the software is built.
requirements
software
20%
2%
30%
5%
50%
8%
90%
10%
90%
20%
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
a 3-week iteration
week 1
week 2
week 3
M
T
W
Th
F
kickoff meeting clarifying iteration goals with the team. 1 hour
team agile modeling & design, UML whiteboard sketching. 5 hours
start coding & testing
de-scope iteration goals if too much work
final check-in and code-freeze for the iteration baseline
demo and 2-day requirements workshop
next iteration planning meeting; 2 hours
Most OOA/D and applying UML during this period
Use-case modeling during the workshop

Typical *Iteration* includes…

- Well-defined, **Prioritized** Set of Requirements
- **Time-boxed** Schedule (Deadline = ‘Dead’line!)
- Output Deliverable: Tested, Integrated and *Partial* Usable System
- Each Iteration includes its own **Requirements** *Analysis*→*Design*→*Programming*→…→*Testing* Cycles (mini-waterfall)
- Incorporates **Feedback** (from Customer and other Key Stakeholders) after every Iteration



Time-Boxed Iteration with *Feedback*


Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Time
Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Feedback from iteration N leads to refinement and adaptation of the requirements and design in iteration N+1.
3 weeks (for example)
Iterations are fixed in length, or timeboxed.
The system grows incrementally.

Let’s Review Few **Popular Agile Methods**:
(Lean, Extreme Programming, SCRUM) ➔

Agile Frameworks

- Common Frameworks (Methods & Techniques in **Practice**) that embrace characteristics of *Agile* :
  - Lean Software Development (Kanban)
  - Extreme Programming (XP)
  - SCRUM (widely adopted today)
- Common **Principles** that Govern The Agile Frameworks
  - Iterative Development: by Multiple Iterations
  - Simplicity, Transparency and Situational-strategies (being ‘street-smart’ for ‘rubber-meets-the-road’ challenges)
  - Cross-functional, Self-organizing Teams
  - Visible Progress: measured by Working Software at any instant

Projects with Agile Frameworks:
Defining Structure vs. Being Prescriptive

- Projects adopting **Lean Methods** & **SCRUM** focus on well-defined Structure and Roles
- Extreme Programming (**XP**) Techniques like *Pair-programming, Release Planning Game, Test-driven Development (TDD)*, etc., are more prescriptive in the nature of project activities
- While SCRUM is the most popular Methods practiced in organizations, individual developers/entrepreneurial setups adopt a creative combination of the methods to meet the project challenges

Lean Methods – Inspiration for Agile

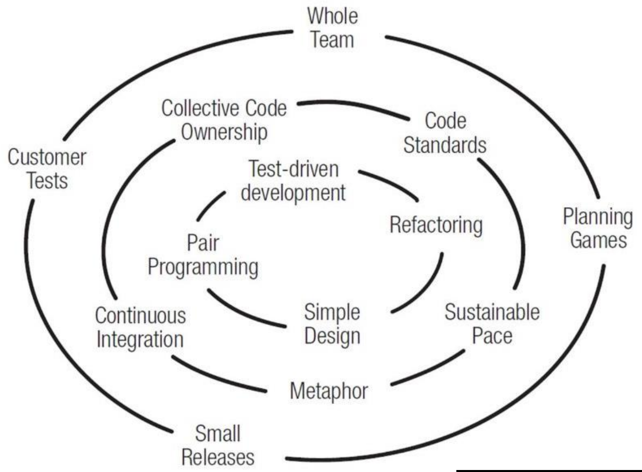
- Originated in Japanese Manufacturing Organizations (Toyota's *Kanban* system), Lean Methods **Focus** on:
  - Eliminating Wastage in mass-manufacturing processes (Just-In-Time Production System)
  - Focus on Humans in the Decision-making in the Production Process (vs. Expensive Machines)
  - Ownership by Shop-floor Workers (vs. Supervisors/Managers)
- **Principles** of Lean:
  - Optimize the Whole, Build Quality, Learn Constantly, Deliver Fast, Engage Everyone, Continuous Improvement (*Kaizen*)
- Lean applied to **Software Product Development**:
  - Avoid '**Unnecessary**' features
  - **People** (not Machines) are central to Project – they add real value
  - Involve Customers early-on and **Prioritize** Requirements
  - Constant **Communication** among All Stakeholders (using Tools)

Extreme Programming (XP)

- Guiding Spirit: *Extreme Focus on Customer and Projects are like War-rooms*
  - Features to be developed when Customer needs them
  - New Requests (or Change-requests) accepted as part of daily routine
  - Dynamic Self-organization of Teams around Customer Problems or Issues as and when they surface
- Principles of XP:
  - Coding is the Language of the Product and Communication
  - Extensive Testing: Coding doesn't start unless Success-criteria is defined;

    "A bug is not a failure of code, it's a failure to define the right test"
  - Direct Communication between Programmer and Customer
  - Design during *Refactoring* to reduce Complexity and Maintainability

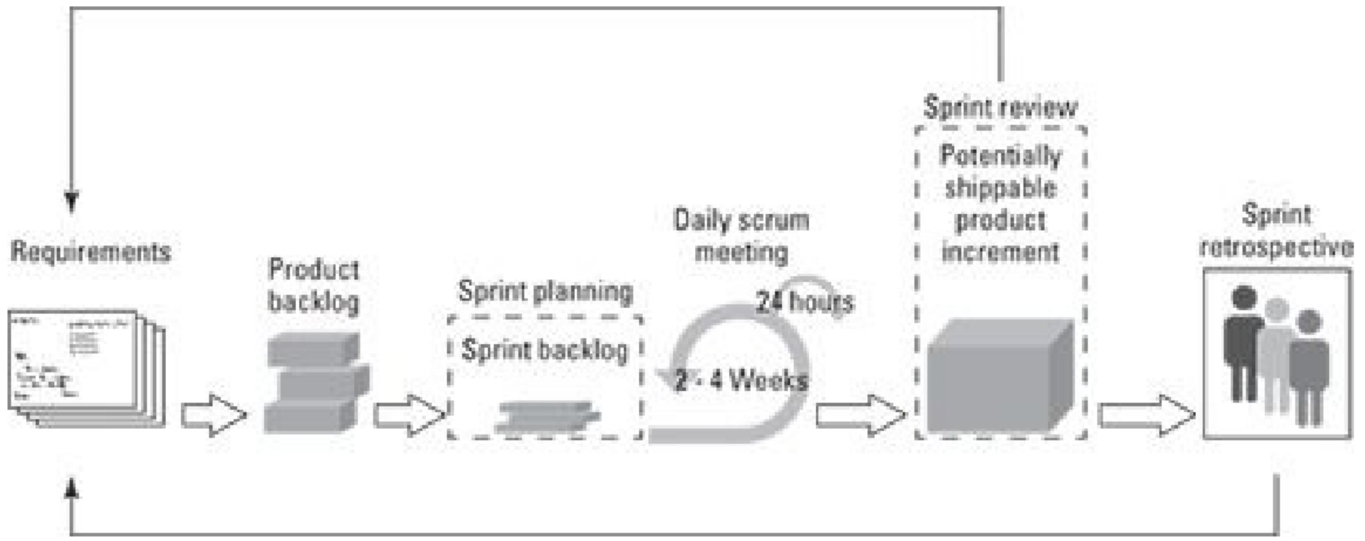
XP – Key Practices



Planning Game: All Members of the Team Should Participate in Planning – No Disconnect between Business and Technical People

WE NEED TO TALK ABOUT MY WORKLOAD.
OKAY. I JUST EMAILED YOU TWO MORE ASSIGNMENTS THAT I NEED FINISHED BY TOMORROW.
YOU ARE LITERALLY KILLING ME.
I CALL IT EXTREME MANAG-ING.
Dilbert.com DilbertCartoonist@gmail.com
© 2013 Scott Adams, Inc.

SCRUM – The Common Agile Project Management Framework



Product Owner: Responsible for End-to-end Product Development
SCRUM Master: Manages the SCRUM Process (Not a *Manager* of Teams)
Cross-functional Teams: Involving Developers, Designers, Testers, and Operations Teams

Agile Software Development - Summary

- Customer and the Developer (Programmer) is at the Centre of any Agile Project Management Framework
- Core Characteristics of Agile: Time-Boxed and Iterative Development (via Short Iterations or Sprints); Continuous Feedback; Direct Involvement of all Key Stakeholders; Constant Communication; Transparency; Cross-functional and Self-organizing Teams,..
- Agile Methods: Lean (Kanban), XP and SCRUM
- SCRUM is the Common Agile Framework adopted in most Software Product Organizations

Thank You