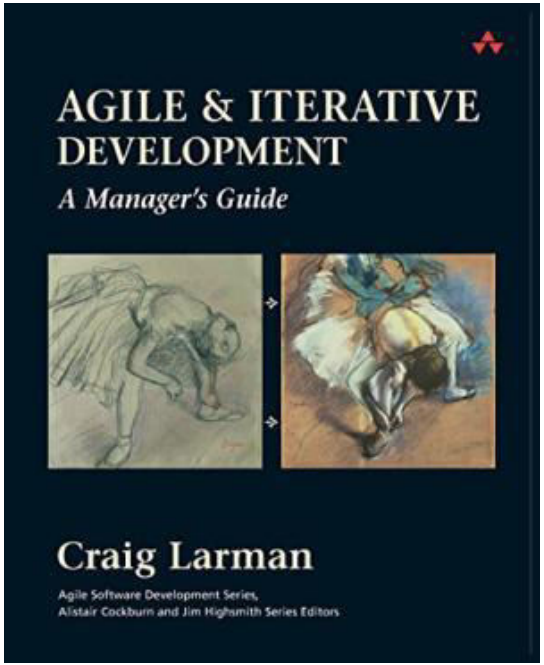
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Agile Methods – An Introduction

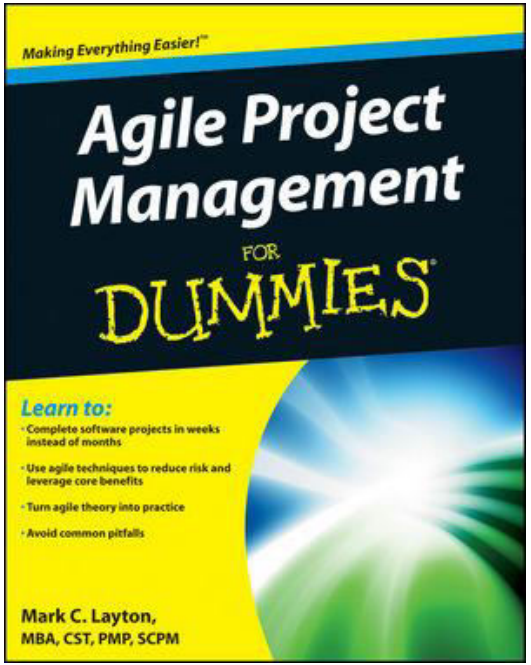
- Prof K G Krishna

Text/Reference Books

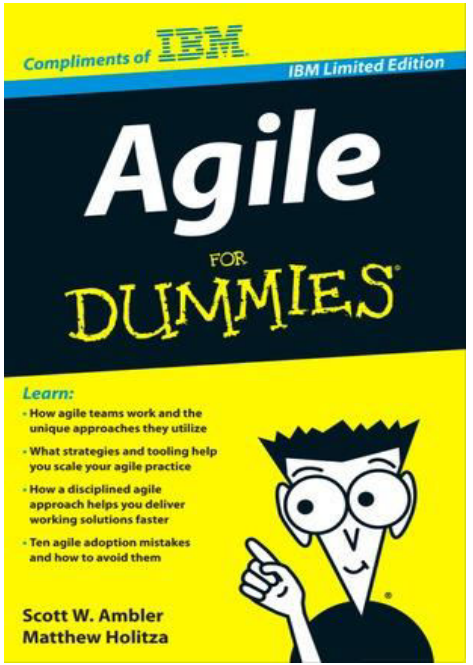
T1



T2



Compliments of IBM



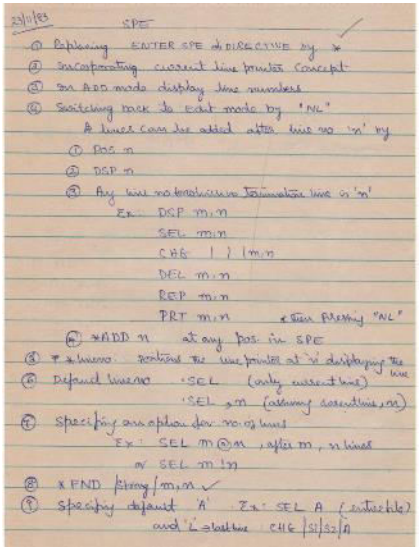
➔ As this field is evolutionary, the student is advised to stay tuned to the current and emerging practices by referring to their own organization's documentation as well as Net sources

Topics

- Traditional software development practices
- Need for Agile Methods
- Benefits of Agile Methods

Traditional Software Development – The Backstory

- Large Software Projects (Mainframe era)
- Development Models based on Linear Models (Sequential/Waterfall)
- No Time-to-Market Constraints (Software is Expensive)
- Mostly Custom/Bespoke Software Development
- Programmer-Intensive Manual Activities

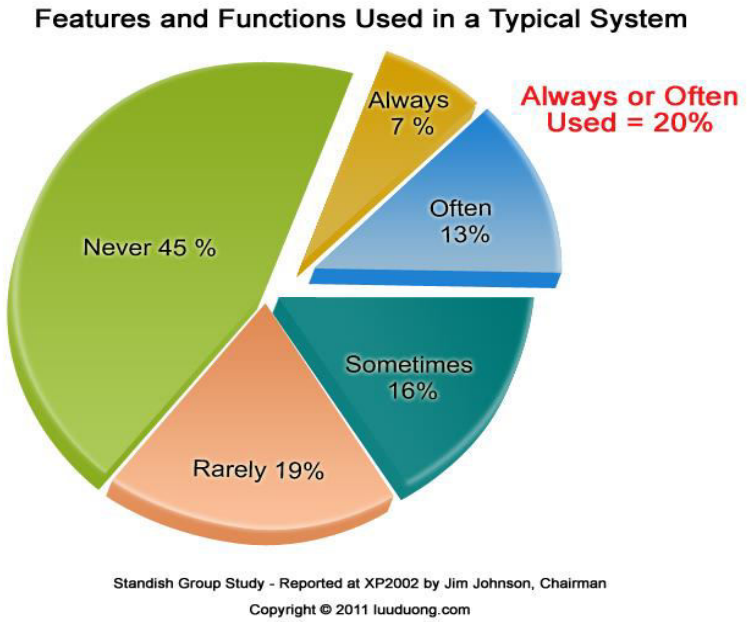


K G Krishna

K G Krishna

"*A temporary endeavour undertaken to create a unique product, service, or result.*" - A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 4th Edition (Project Management Institute, 2008)

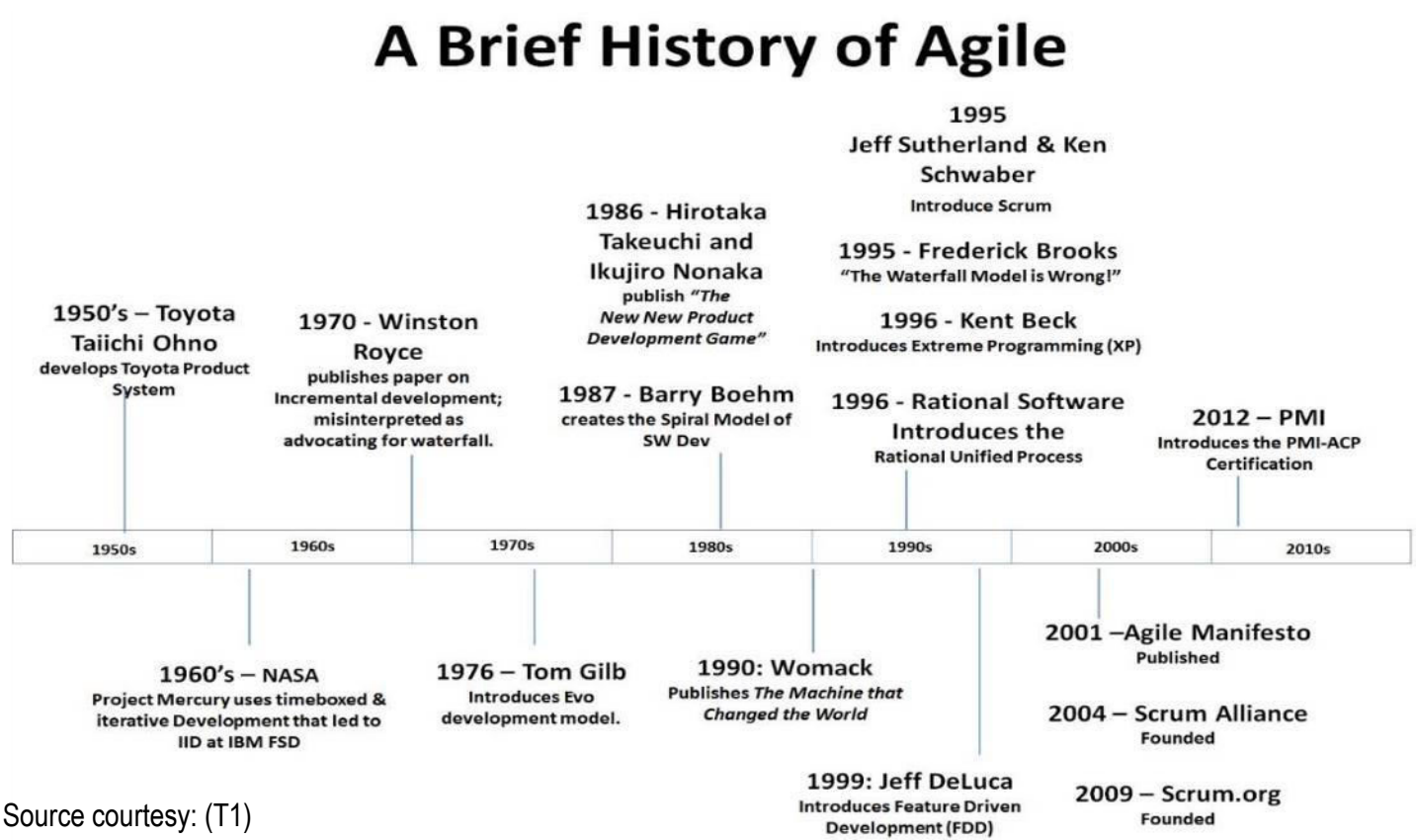
Traditional Software Projects (circa 1990 ~ 2000)



In 2009, companies and organizations in the U.S. spent $491.2billion on application development. That means that more than $103billion was wasted on failed projects.

Agile Turns Upside-down Project Constraints

Production-Line ➔ Agile System – An Evolution



Source courtesy: (T1)

Being Agile Means…

agile
/ˈadʒʌɪl/

adjective

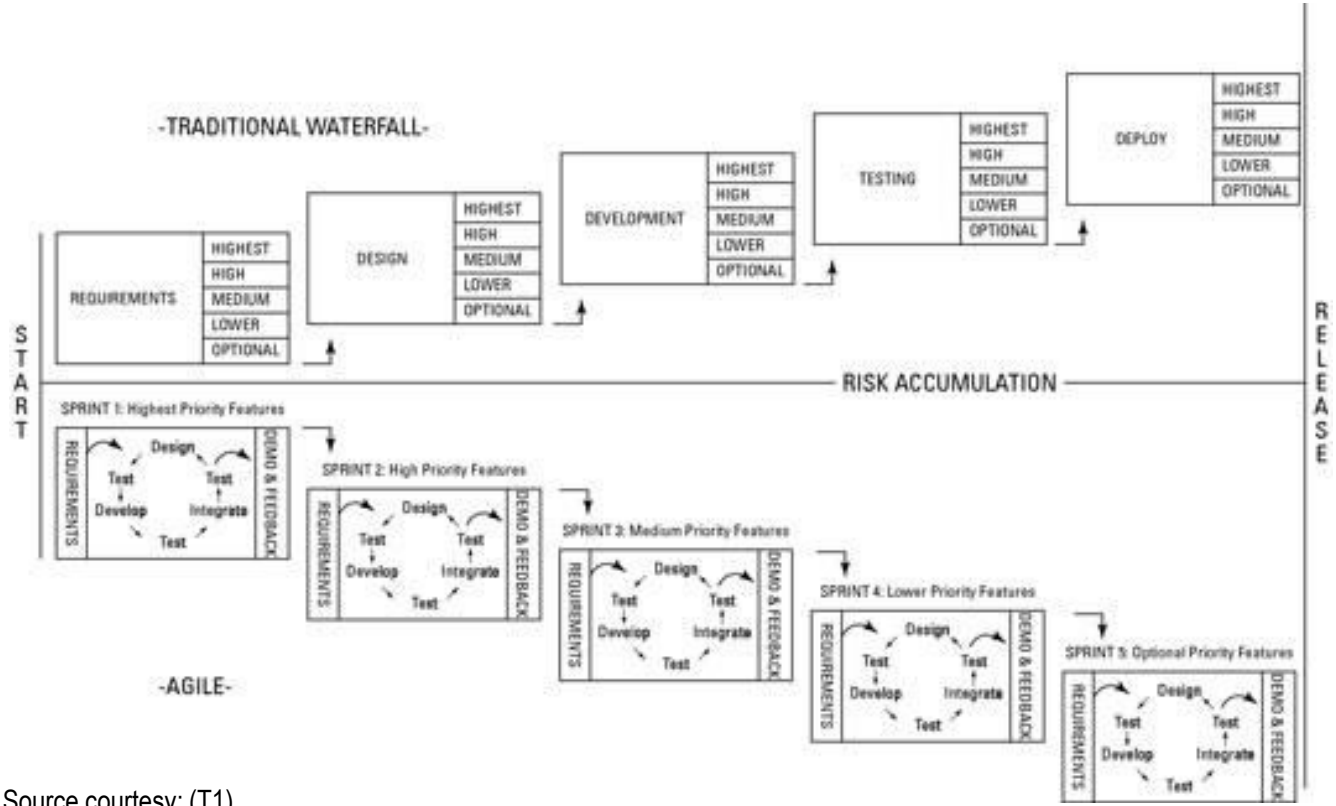
1. able to move quickly and easily.
   "Ruth was as agile as a monkey"
   synonyms: nimble, lithe, spry, supple, limber, sprightly, acrobatic, dexterous, deft, willowy, graceful, light-footed, nimble-footed, light on one's feet, fleet-footed; More

- Agility

  The ability to both create and respond to change in order to profit in a turbulent business environment

  - Rigid Processes vs. Agile Frameworks
  - Being Agile = Competitive, Responsive, Flexible,…

Agile Development – A Series of Short Sprints



Source courtesy: (T1)

Agile Development = Iterative Releases!

- **Agile software development** is a conceptual framework for software engineering that promotes development **iterations** throughout the life-cycle of the project.

- Software developed during one unit of time is referred to as an iteration (**sprint**), which may last from one to four weeks.

- Agile methods also emphasize **working software** as the primary measure of progress

Need for Agile Methods ➔

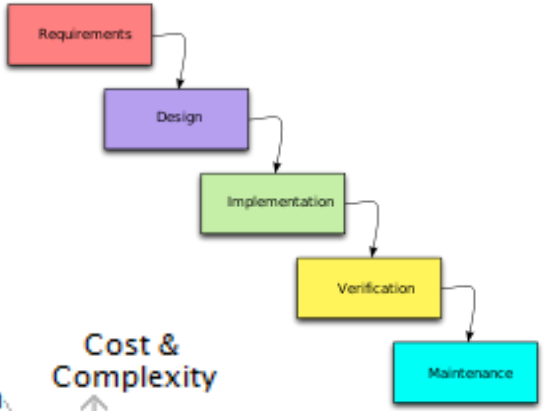
Software Systems Today…

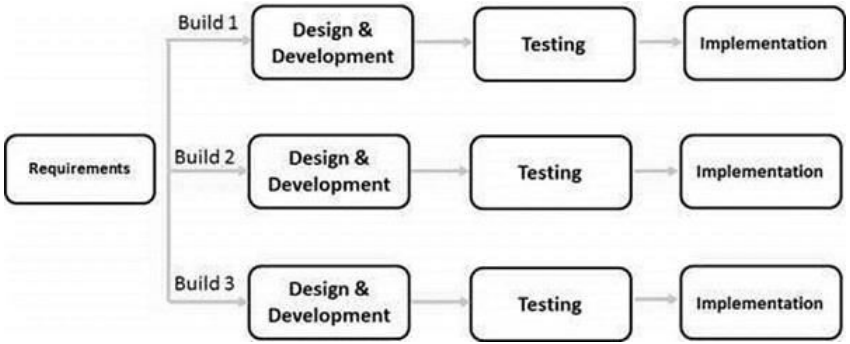
- Changing (‘Unclear’) Requirements / Customer Demands

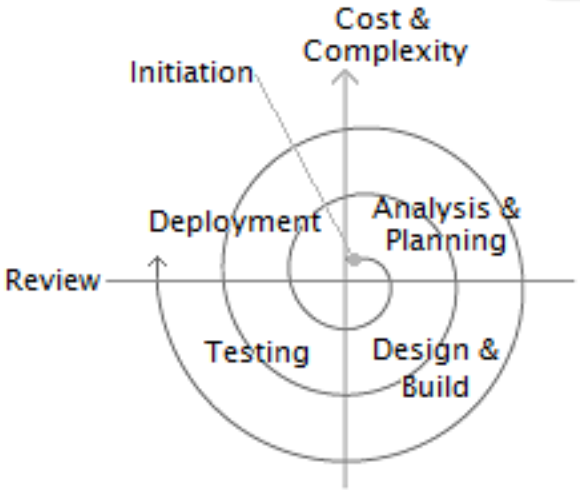
  “Requirements will be clear only at the end of the Project”
- Shrinking Development Cycles (Time-to-Market Releases)
- Entrepreneurial, Innovative Apps
- Shorter Shelf-life

Addressing Challenges of *Waterfall* Model

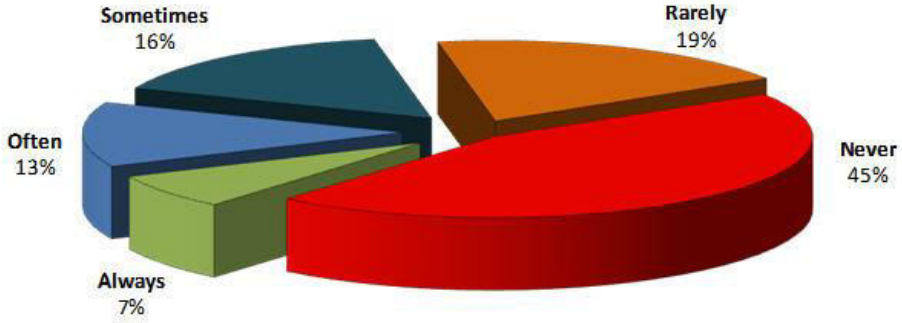
- Traditional Waterfall ➔ Adaptations of Waterfall (Incremental, Iterative Prototyping,…)
  - Focus on Requirements Management (Breaking down into manageable ‘Increments’)
  - De-risking Large Development Effort
  - Capturing Requirements (‘show-and-tell’, ‘prototype’ as basis of discussion around Requirements)
  - “Quick-and-dirty” Working Prototype to Start with







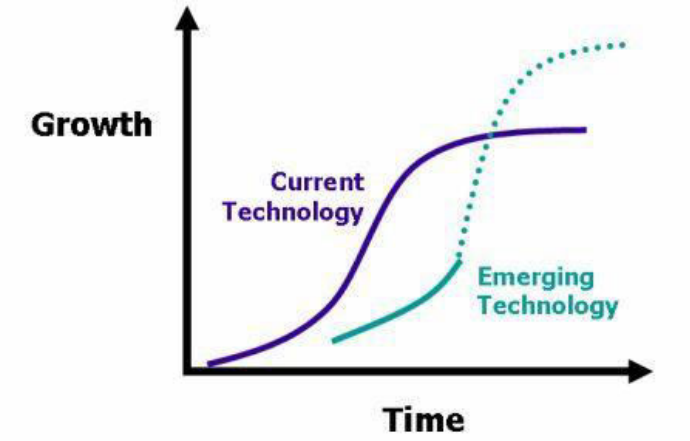
Software Feature Overload



Source: Standish Group Study, 2002

Pace of Technology *vs* Project Duration

- Technology Life-cycles have shrunk (decade ➔ 1..2..3 years)
- Bespoke Software ➔ Product Customization
- Line-by-line Coding ➔ Integrated Development Frameworks



De-risking Future Investments Upfront…

- “Fail-Safe” early rather than Failure at the end
- “Proof-of-concept” when adopting new Technologies
- “Testing the waters” before launching Big in the Marketplace
- Today’s Software Products are strategic levers--“Idea-driven” rather than mere automation of manual process

Benefits of Agile Methods ➔

Controlled Development

- Agile Products are based on empirical control method – decisions based on reality
- Adjustments on-the-go by Frequent Inspections
- **Transparency**: Everyone involved knows what is going in the project
- **Frequent Inspection**: Regular evaluation of the Product
- **Adaptation**: Make quick adjustments to minimize problems later

Agile Development ➔ Agile Project Management

- Traditional Project Management Turned Upside-down!
- "*Let's wait till the Project completes to see the Product*" ➔ "*Several Min-projects with little visible successes*"
- Transformation of Project Management by actively involving ALL the Stakeholders; Organization Structures & Communication; Time-Boxing of Deliveries

"*Standish Group Study on Software project success and failure: In 2009, 26 percent of projects failed outright — but in 2011, that number fell by 5 percent. The decrease in failure has, in part, been attributed to wider adoption of agile approaches*

Benefits of Agile Project Management

- *Almost Zero* Risk of Catastrophic Project Failure
- Prioritization of Business Value over ‘Good or Nice-to-have’ features
- Agile Testing (Continuous Testing) ensures Problems are discovered early
- Down-plays ‘Scope-creep’ as Requirement Changes are managed throughout Product Development Life-cycle
  - Prioritizing Features in early Iterations
  - Managing Evolving Requirements
- Continuous Inspection and Adaptation: Improvement of Processes and Products based on Prior Experience of the ‘Completed’ Product

Agile Methods - Summary

- Traditional Software Development Methods involving Large One-time Projects are yielding to Low-risk High-turnaround Incremental Deliverables in Agile Methods
- Involvement of All Stakeholders (including Customer) early on in the Process enhances Collaboration and minimizes Scope-creep
- Full Transparency and High-visibility of Deliverables in Short Iterations (Sprints)
- Continuous Quality Monitoring with embedded Agile Testing across all Iterations

“Agile is the Great Leap Forward in Software Development Methodology with its associated Transformation in Agile Project Management”

Thank You

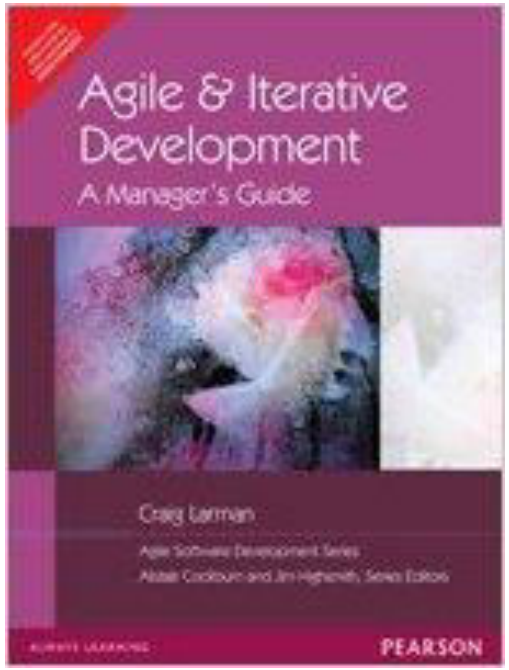
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Agile Software Development

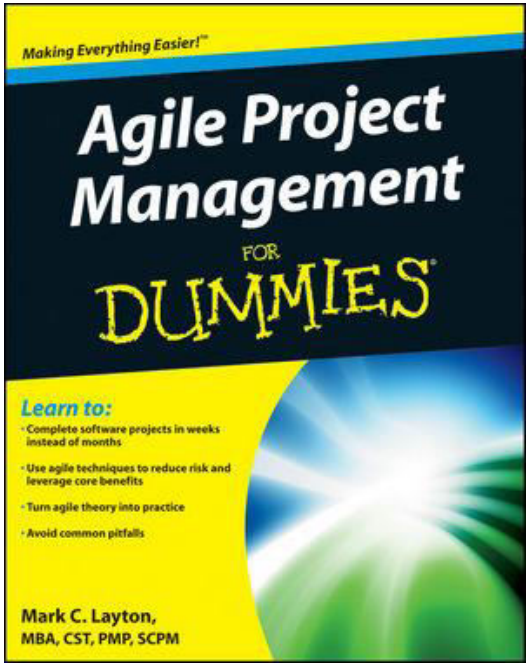
- Prof K G Krishna

Text/Reference Books

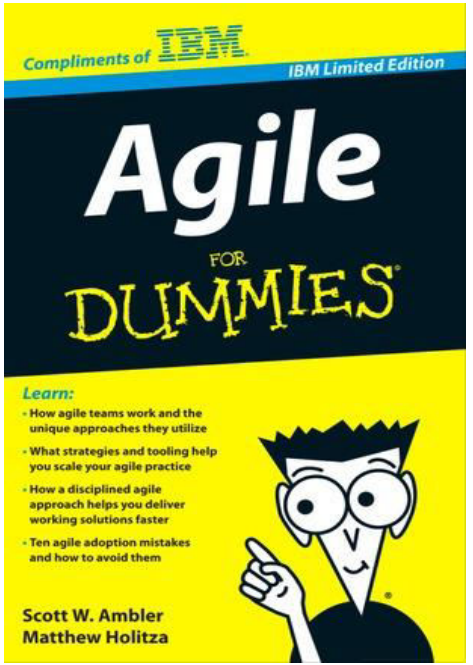
T1



T2



Compliments of IBM



➔ 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

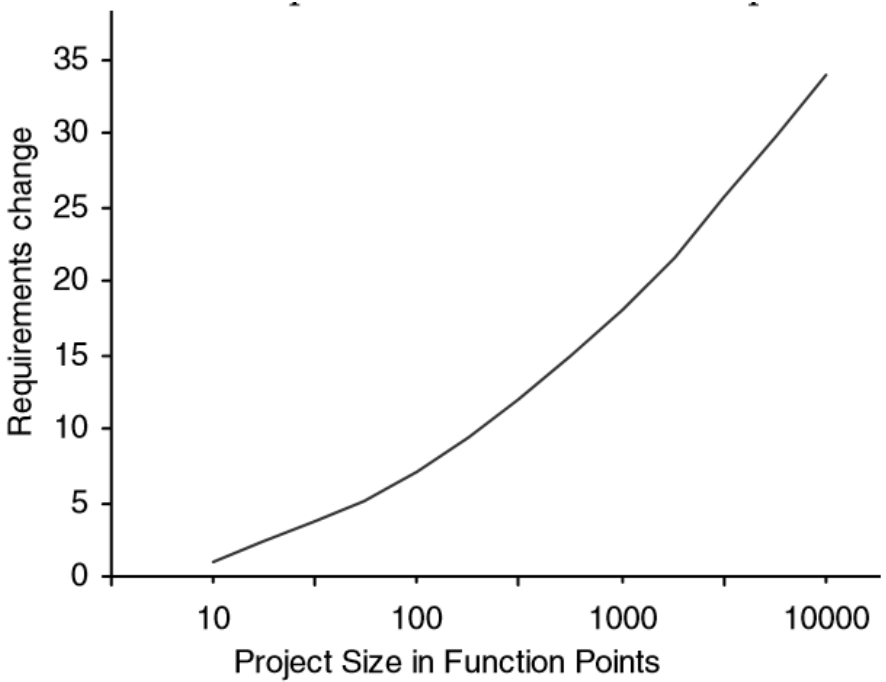
- Iterative and Incremental Approaches
- Risk driven and client driven development
- Time-boxed development
- Adaptive and Evolutionary development

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

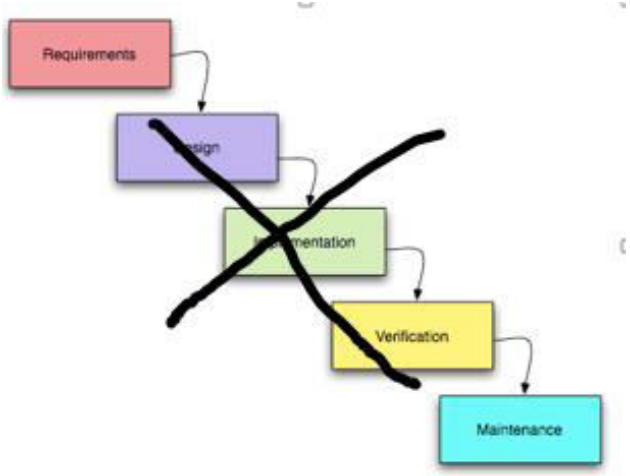
- "Requirements are capabilities and conditions to which the system—and more broadly, the project—must conform"
- "A prime challenge of requirements analysis is to find, communicate, and remember (that usually means write down) what is really needed, in a form that clearly speaks to the client and development 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!

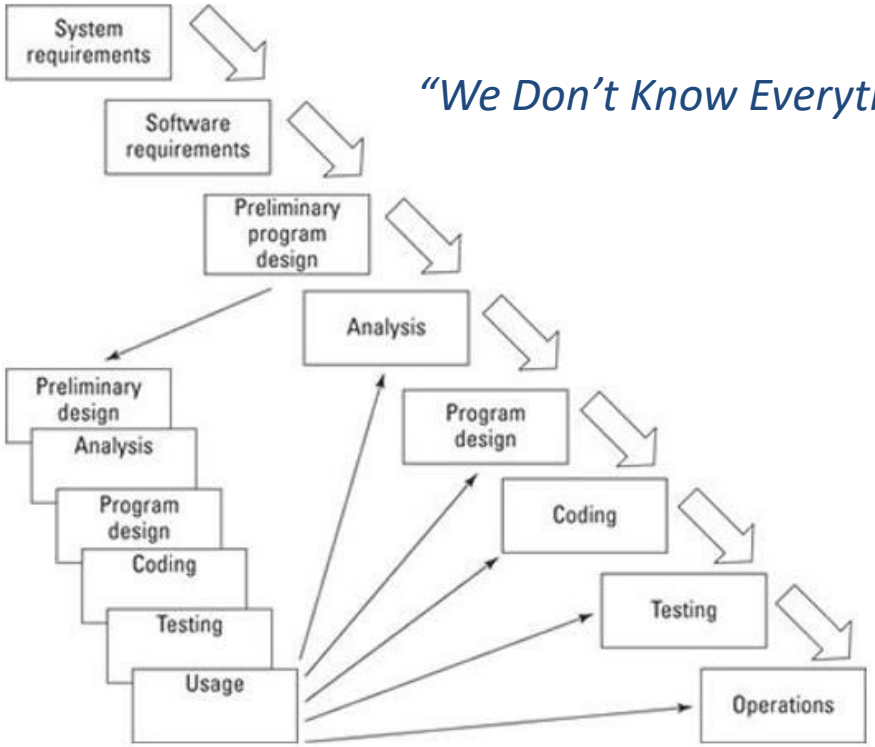


The ills of Waterfall…

- 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…

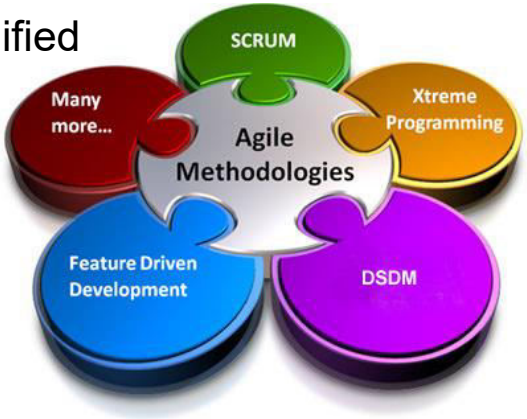


"We Don't Know Everything When We Start The Project…"

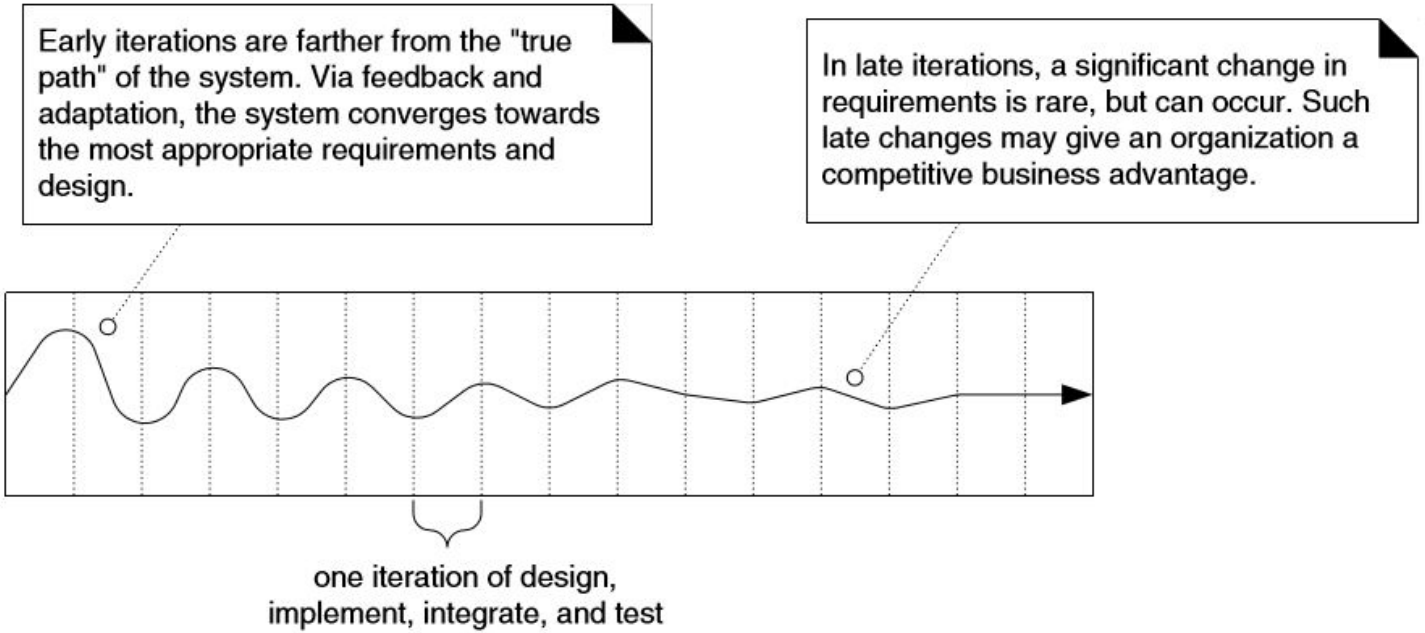
Iterative & Incremental...is the Way To Go!

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.



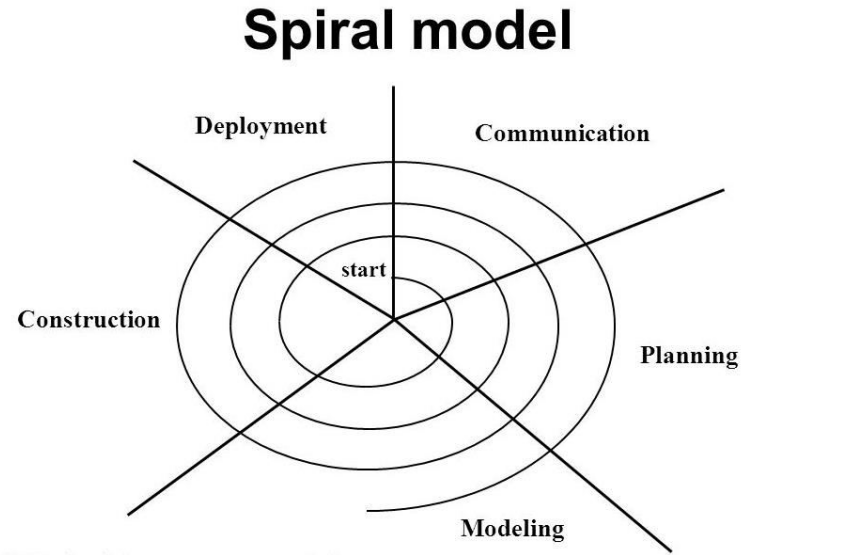
Iterations *Converge* to True Requirements!



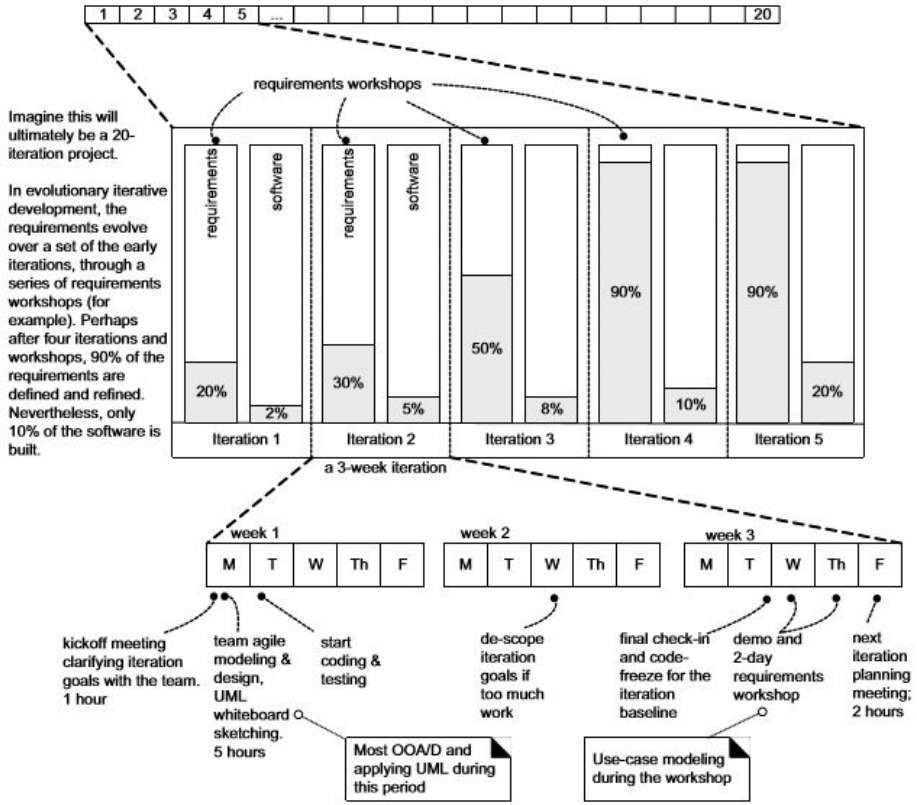
NO 'Waterfall Thinking' in Iterative Development! "…on average 45% of the features in waterfall requirements are never used, and early waterfall schedules and estimates vary up to 400% from the final actuals…"

Risk-Driven, Customer-Driven Iterative Planning

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

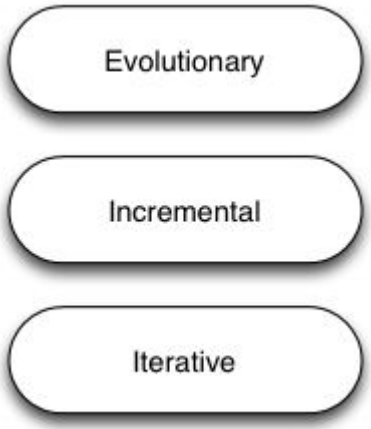


Evolutionary Analysis & Design in Early Iterations

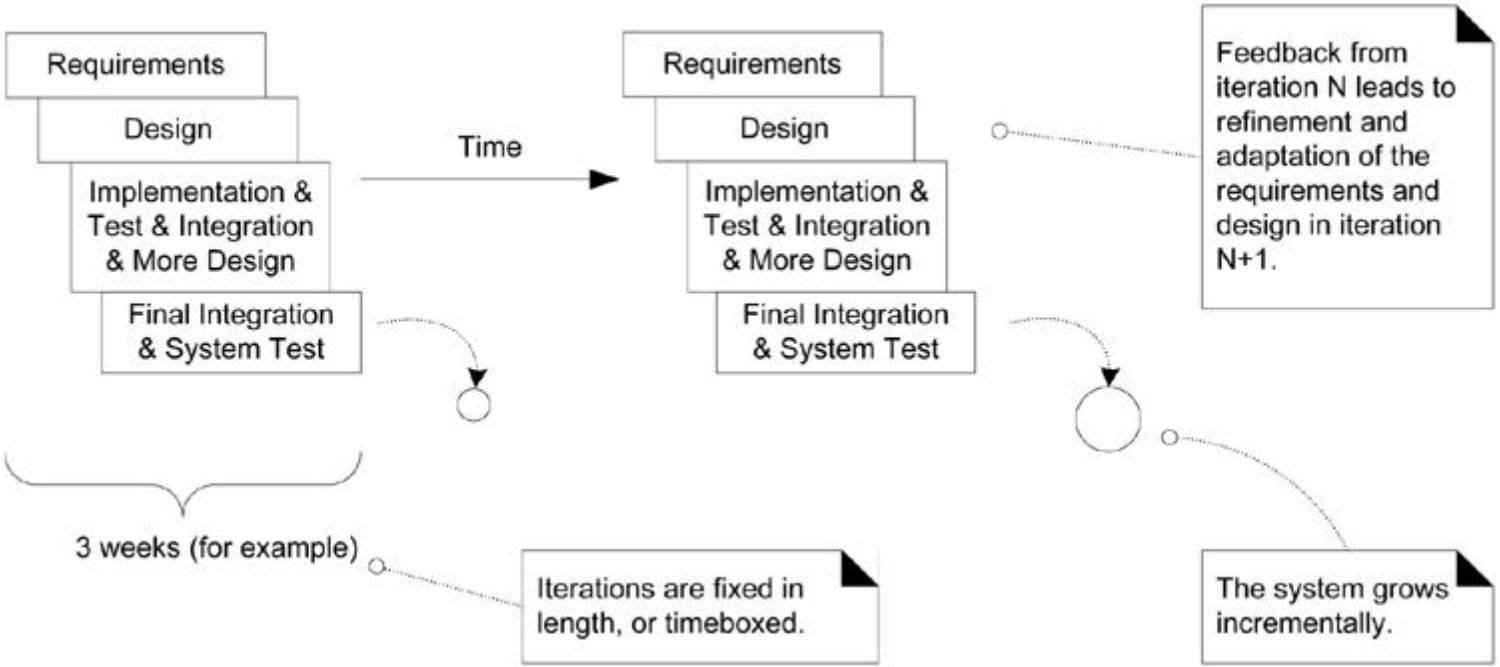


Typical *Iteration* includes…

- 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*



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

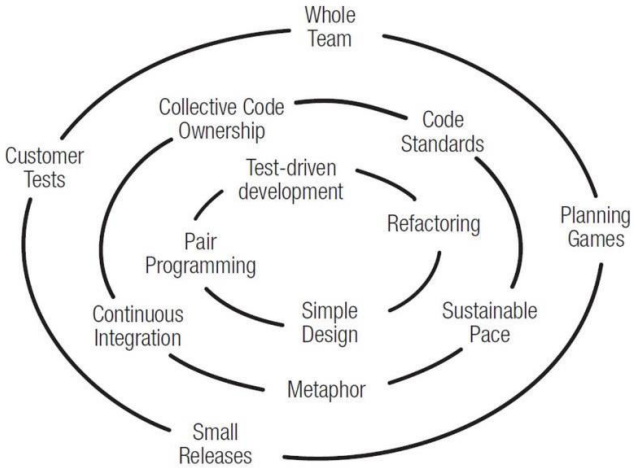
- Originated in Japanese Manufacturing Organizations (Toyota's *Kanban* system), Lean Methods **Focus** on:
  - Eliminating Wastage in mass-manufacturing processes (Just-In-Time Production System)
  - Focus on Humans in the Decision-making in the Production Process (vs. Expensive Machines)
  - Ownership 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;

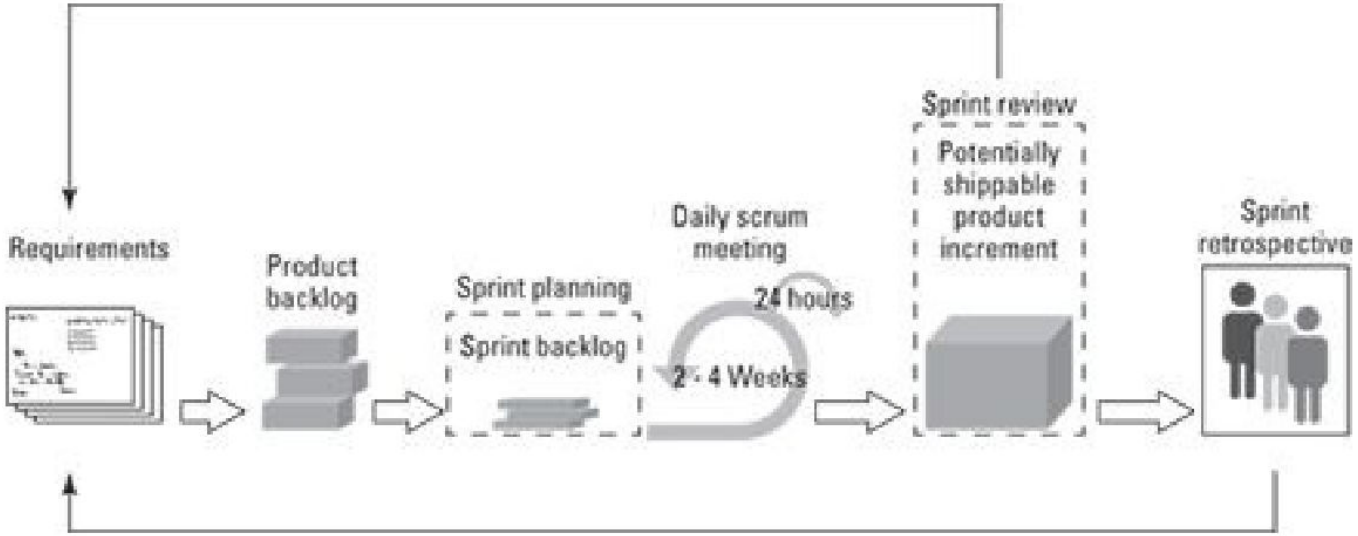
    "A bug is not a failure of code, it's a failure to define the right test"
  - Direct Communication between Programmer and Customer
  - Design during *Refactoring* to reduce Complexity and Maintainability

XP – Key Practices



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

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

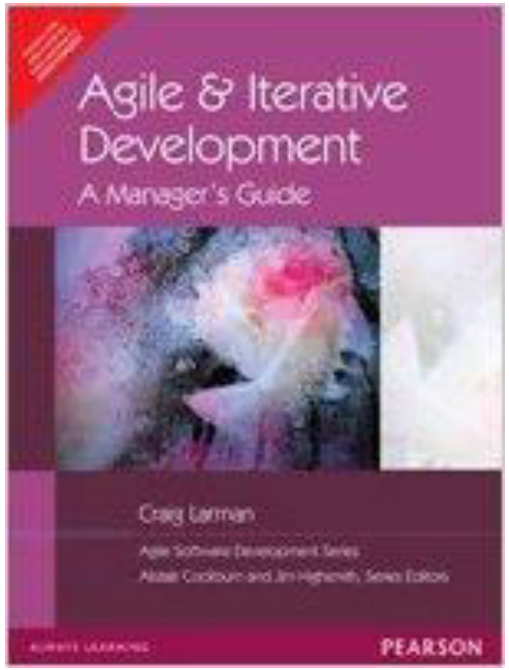
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Agile Principles & Manifesto

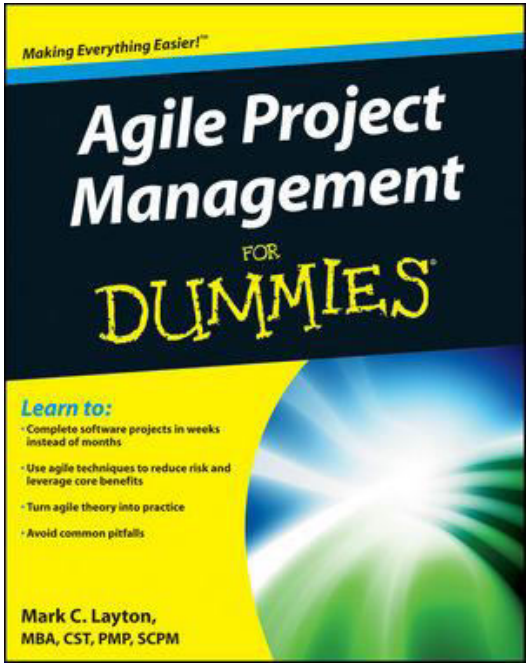
- Prof K G Krishna

Text/Reference Books

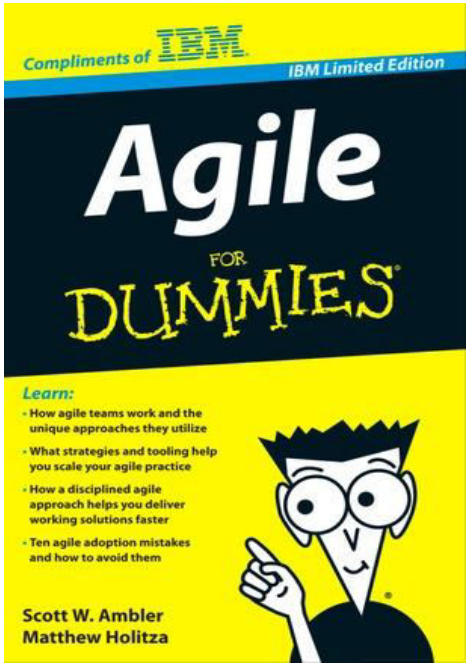
T1



T2



Compliments of IBM

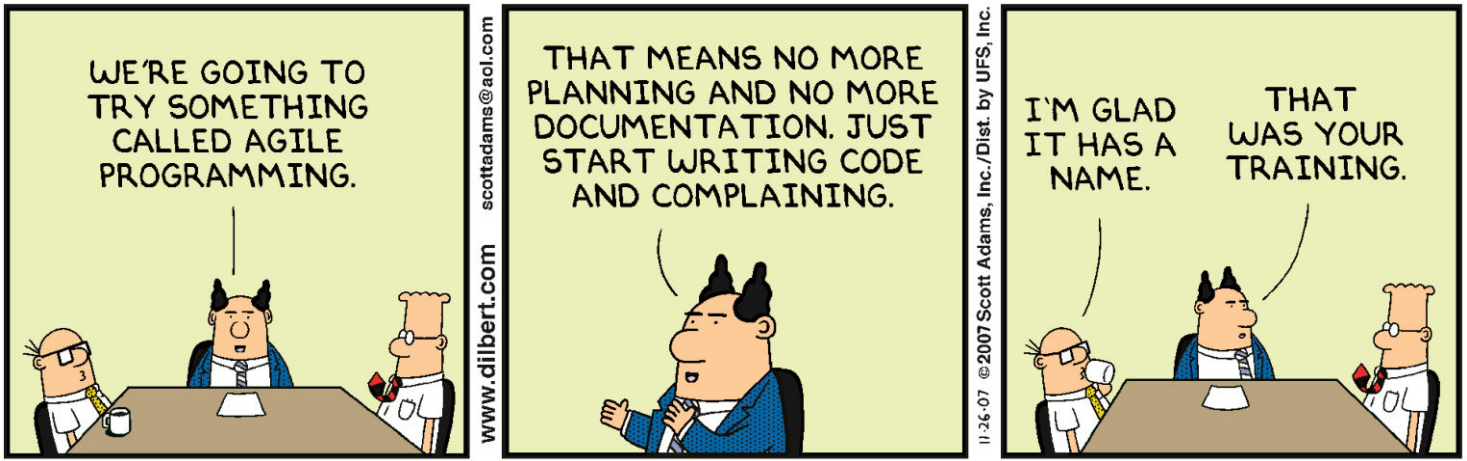


➔ As this field is evolutionary, the student is advised to stay tuned to the current and emerging practices by referring to their own organization's documentation as well as Net sources

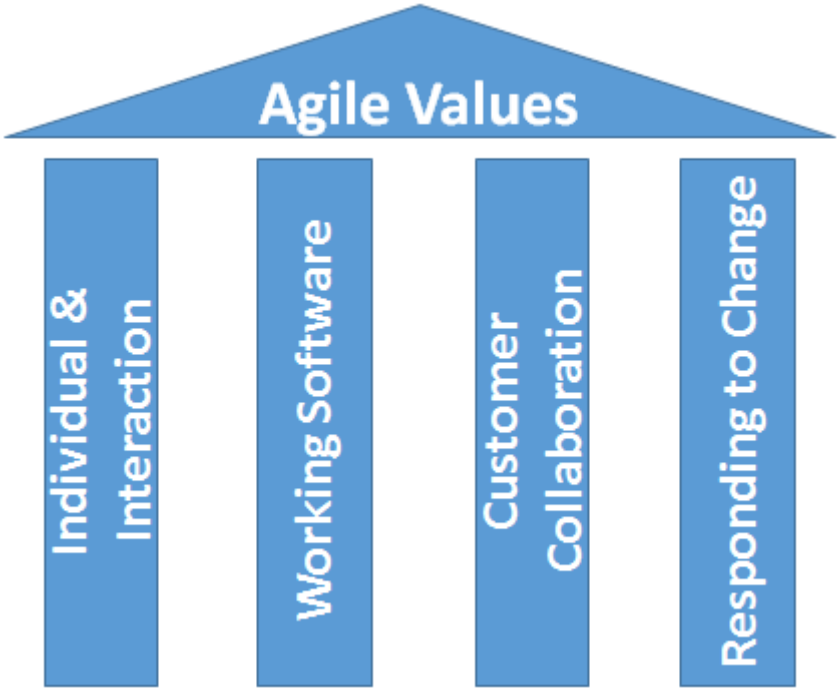
Topics

Principles of Agile

- Agile Values
- Agile Manifesto



Agile Core Values...



Agile Principles…

Principles behind the Agile Manifesto

We follow these principles:

- ✓ Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- ✓ Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- ✓ Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- ✓ Business people and developers must work together daily throughout the project.
- ✓ Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- ✓ The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- ✓ Working software is the primary measure of progress.
- ✓ Agile processes promote sustainable development.
- ✓ The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- ✓ Continuous attention to technical excellence and good design enhances agility.
- ✓ Simplicity--the art of maximizing the amount of work not done--is essential.
- ✓ The best architectures, requirements, and designs emerge from self-organizing teams.
- ✓ At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Source courtesy: agilemanifesto.org

Agile Manifesto (2001)

Source courtesy: agilemanifesto.org

Agile Manifesto (adapted to times)

Source: https://www.ncsc.gov.uk/blog-post/securing-agile-delivery-collaboration-crucial

9 Principles *Agile Project Manager*

1. Deliver something useful to the client; check what they value
2. Cultivate committed stakeholders
3. Employ a leadership-collaboration style
4. Build competent, collaborative teams
5. Enable team decision making
6. Use short time-boxed iterations to quickly deliver features
7. Encourage adaptability
8. Champion technical excellence
9. Focus on delivery activities, not process-compliance activities

The Agile Project Manager

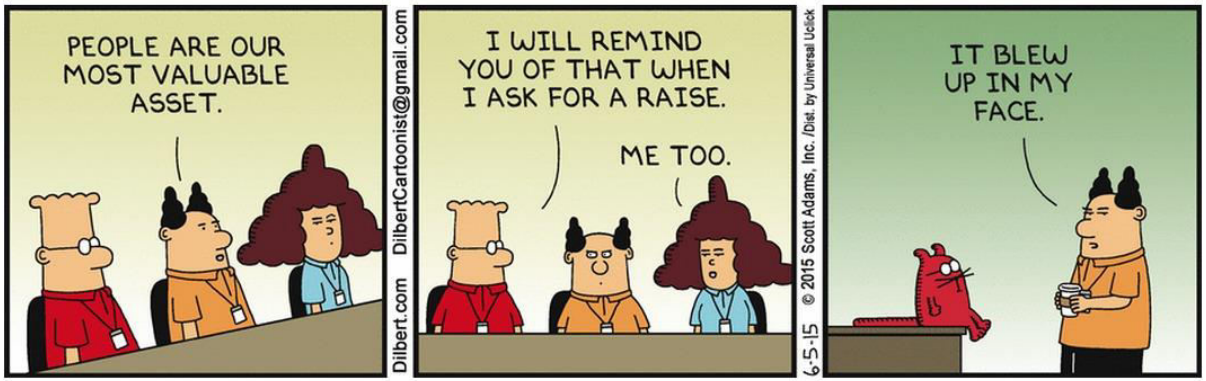
- Devolve of both control and planning to the entire team, not the manager
- The manager does not create WBS, schedule, estimates or tell people what to do
- The manager does not define and assign detailed team roles and responsibilities

Source: (T1-Chap3) Jim Highsmith, Agile Alliance Founder

The *Human Touch* in Agile

"People are more important than any process. Good people with a good process will outperform good people with no process every time" – Grady Booch

- Programming is an intense Human Activity - People matter much more than Machines
- Individuals and Interactions over Processes and Tools
- Sustainable Pace – Programmers to maintain a healthy social and family life
- Respect Diversity of Individual Contributions – Skill Transfer through *Pair Programming*
- Preference for Direct Face-to-Face Communication over Virtual Meetings or Remote Teams
- …



Agile Principles & Manifesto - Summary

- Programming as if People Matter Most
- Customer is at the Centre – The Key Stakeholder
- Adapt to the Reality: Requirements keep Changing till the End of the Project
- Continuous and Incremental Delivery with a series of Working Product Releases
- Involve All Stakeholders Early-on
- Focus on Delivery over Process-compliance
- Close-knit Communication and Collaboration among Teams
- Agile Project Manager is the Leader and Motivator – Not the Boss of Manager of People
- Collective and Collaborative Decision-making
- …

Thank You

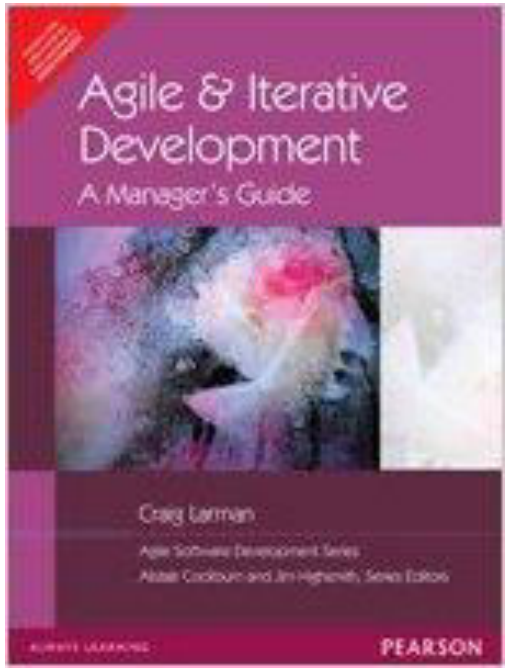
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Agile Methodologies

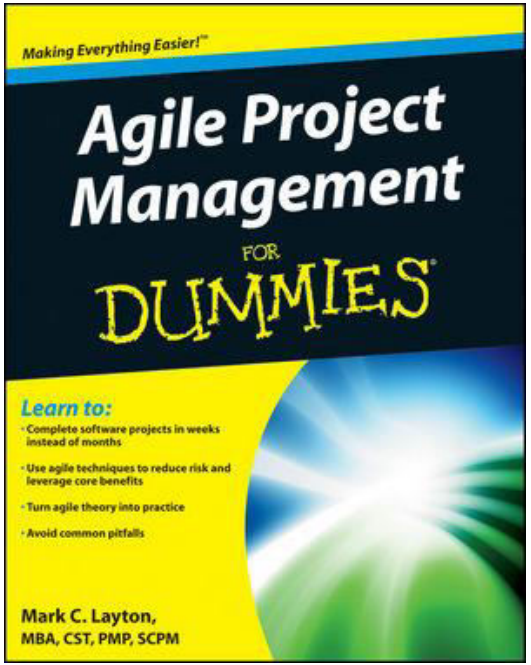
- Prof K G Krishna

Text/Reference Books

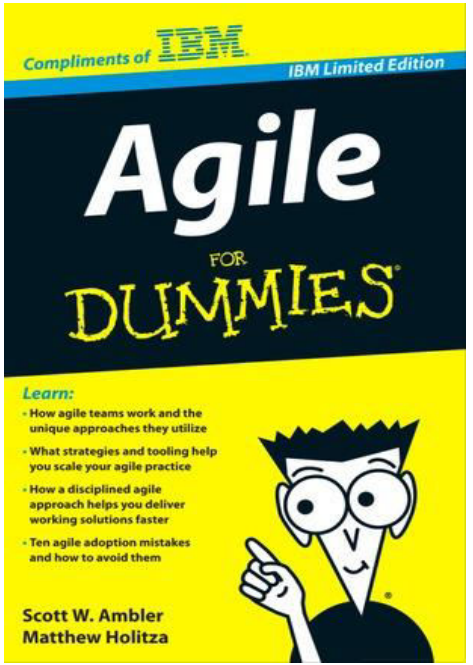
T1



T2



Compliments of IBM



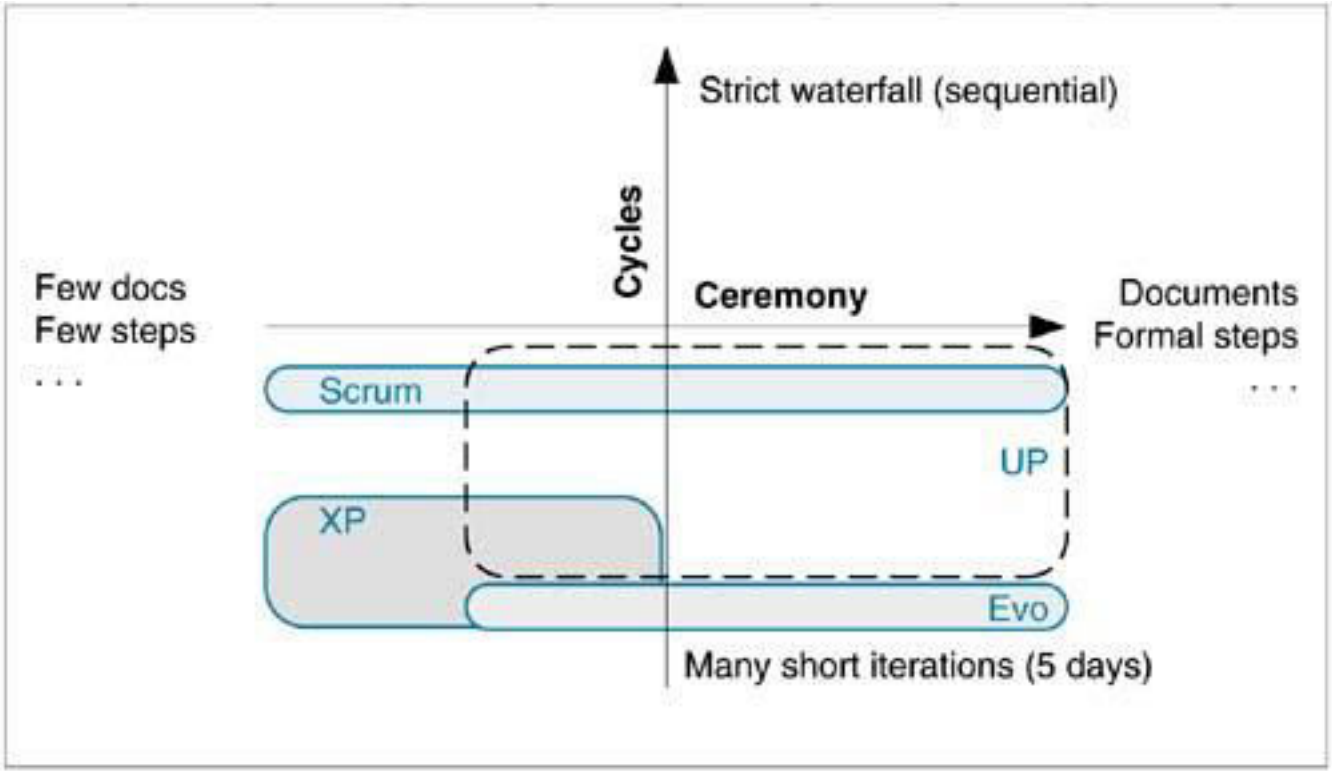
➔ As this field is evolutionary, the student is advised to stay tuned to the current and emerging practices by referring to their own organization's documentation as well as Net sources

Topics

Overview of Agile Methodologies

- SCRUM
- Extreme Programming (XP)
- Test-Driven Development (TDD)

SCRUM on '*Ceremony – Cycles*' Scale



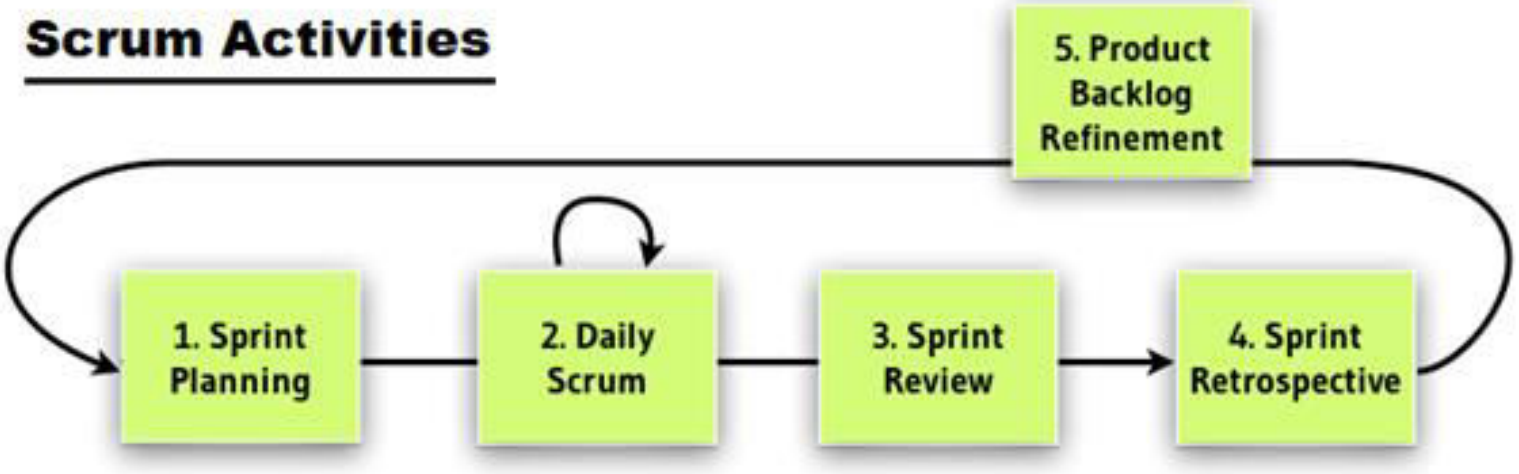
Source: T1-Chap7

SCRUM Lifecycle

PRE-GAME		DEVELOPMENT	RELEASE
PLANNING	**STAGING**		
Purpose: - establish the vision, set expectations, and secure funding	**Purpose**: - identify more requirements and prioritize enough for first iteration	**Purpose**: - implement a system ready for release in a series of 30-day iterations (Sprints)	**Purpose**: - operational deployment
Activities: - write vision, budget, initial Product Backlog and estimate items - exploratory design and prototypes	**Activities**: - planning - exploratory design and prototypes	**Activities**: - Sprint planning meeting each iteration, defining the Sprint Backlog and estimates - daily Scrum meetings - Sprint Review	**Activities**: - documentation - training - marketing & sales - . . .

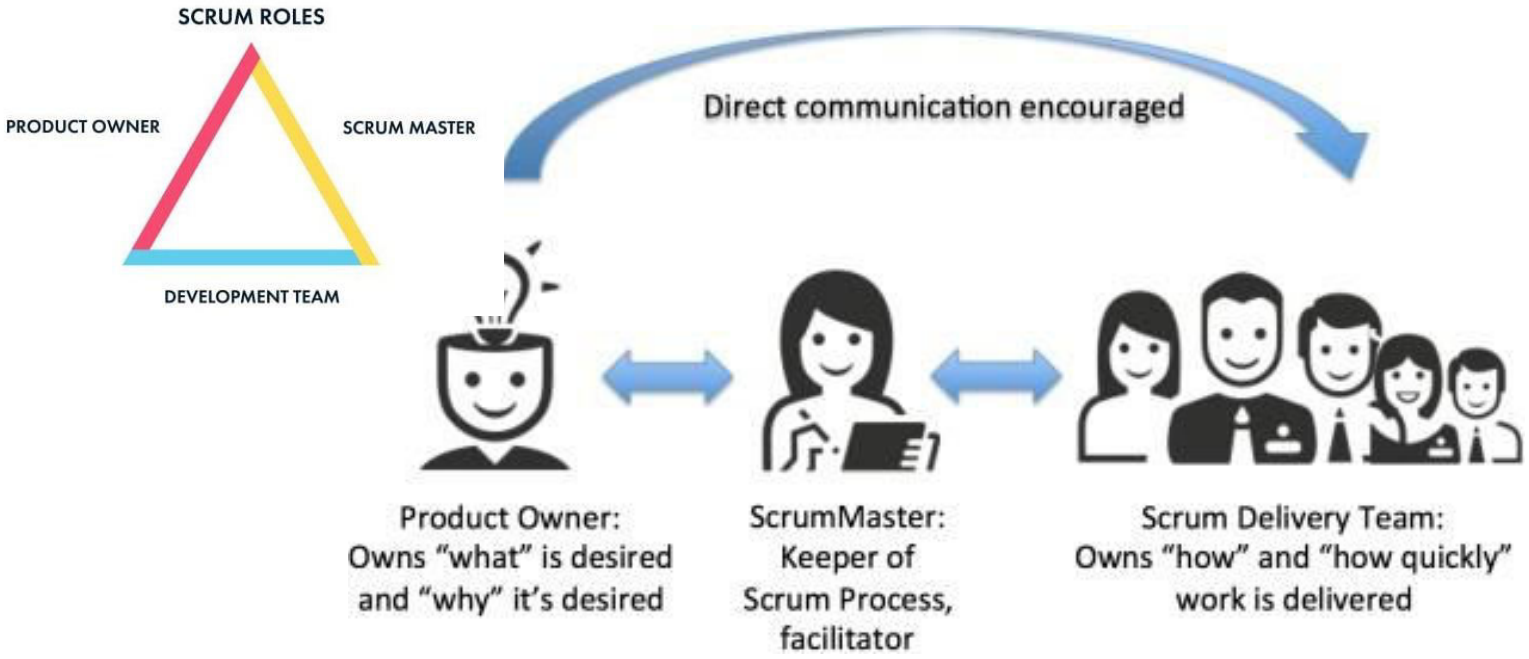
Source: T1-Chap7

SCRUM Activities

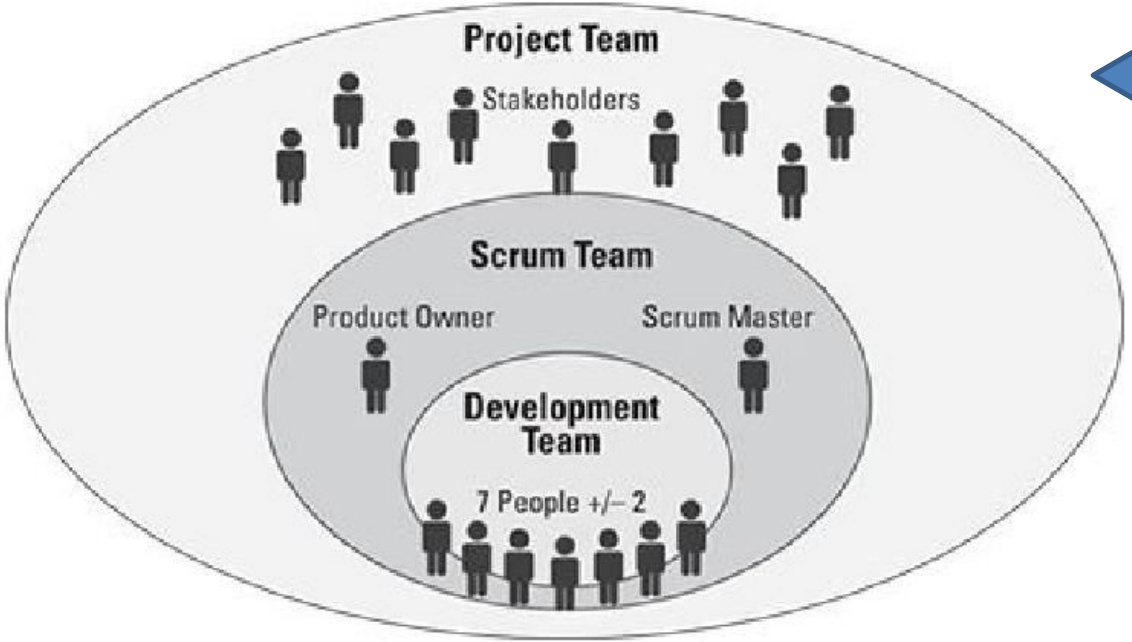


Source courtesy: www.snyxius.com/how-to-run-scrum-planning-meeting-like-pro

SCRUM – The *Key Players*



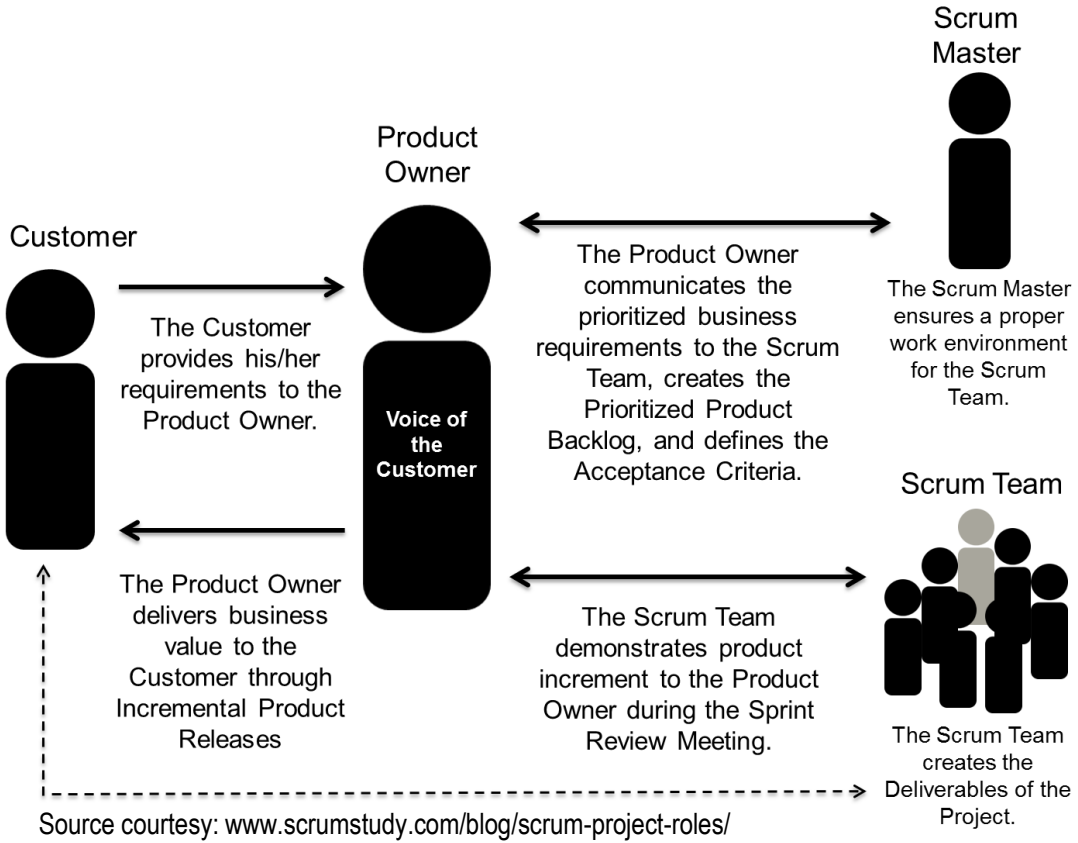
The ‘Extended’ SCRUM Team



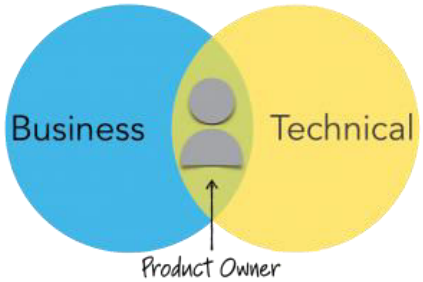
+ Agile Coach

Source: (T2)

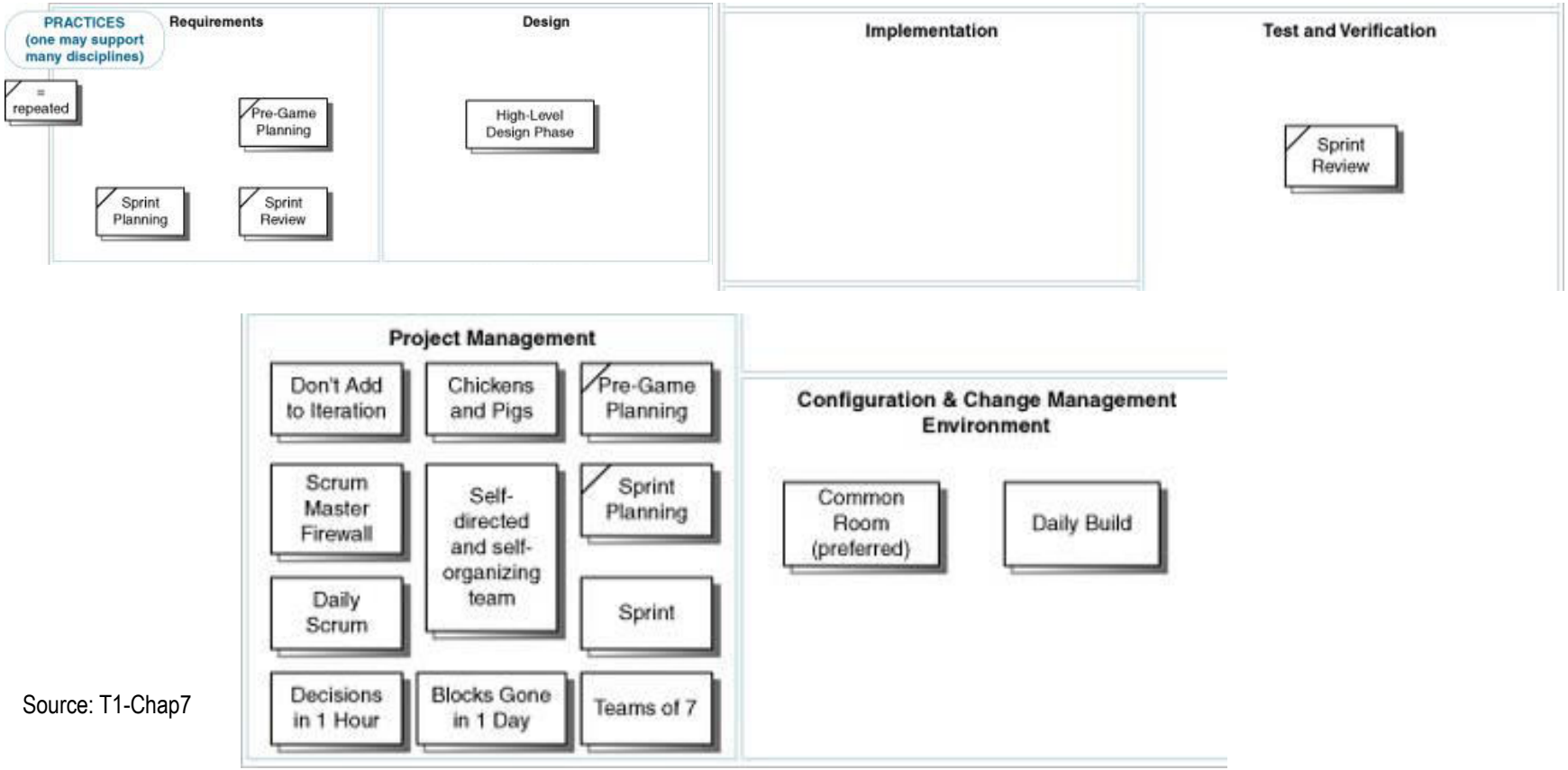
SCRUM Project Roles



Source courtesy: www.scrumstudy.com/blog/scrum-project-roles/



SCRUM Practices



Source: T1-Chap7

The Daily SCRUM Meeting –
The Heartbeat of the Project

- Daily Standup Meetings (1 – 7 Members, 15 – 20 minutes)
- Typical Questions Answered:
  - What have you done since the last Scrum?
  - What will you do between now and the next Scrum?
  - What is getting in the way (blocks) of meeting the iteration goals?
  - Any tasks to add to the Sprint Backlog? (missed tasks, not new requirements)
  - Have you learned or decided anything new, of relevance to some of the team members? (technical, requirements, …)
  - …
- Non-Team Members (‘chickens’) Can’t Ask Questions – they just listen
- White-board Meeting / Tele-Conf Call
- Shared Responsibility and Team Cohesiveness is maintained

SCRUM Artifacts

Product Backlog

Requirement	Num	Category	Status	Pri	Estimate
log credit payments to AR	17	feature	underway	5	2
process sale-simple cash scenario	232	use case	underway	5	60
slow credit payment approval	12	issue	not started	4	10
sales commission calculation	43	defect			
lay-away plan payments	321	enhance			
PDA sale capture	53	technology			
process sale-credit pmt scenario	235	use case			

Sprint Backlog

Task Description	Originator	Responsible	Status	Hours of work remaining				
				6	7	8	9	10
				362	322	317	317	306
Meet to discuss the goals and	JM	JM/SR	Completed	20	10	0	0	0
Move Calculations out of	TL	AW	Not Started	8	8	8	8	8
Get GEK Data		TN	Completed	12	0	0	0	0
Analyse GEK Data - Title		GP	In Progress	24	20	30	25	20
Analyse GEK Data - Parcel		TK	Completed	12	12	12	12	12
Define & build Database		BR/DS	In Progress	80	80	75	60	52

Source: T1-Chap7

SCRUM Values

("At the end, It's the People that Matters the Most")

- **Commitment**: The Team given Authority and Autonomy to decide; The Product Owners commits to Product Backlog; The Scrum Master commits not to introduce new work items till Iteration is complete
- **Focus**: The Scrum Master ensures the Team is not distracted; commits Resources and removes Roadblocks if any
- **Openness**: Product Backlogs and Daily Progress of Work Items are Visible to the entire Team
- **Respect**: Diversity of Individual Strengths and Weaknesses; facilitate Self-directed Teams; Teams empowered to seek/hire resources
- **Courage**: The Team has the courage to take Decisions adaptively; Management is supportive by empowering Teams

SCRUM Drawbacks (discountable however!)

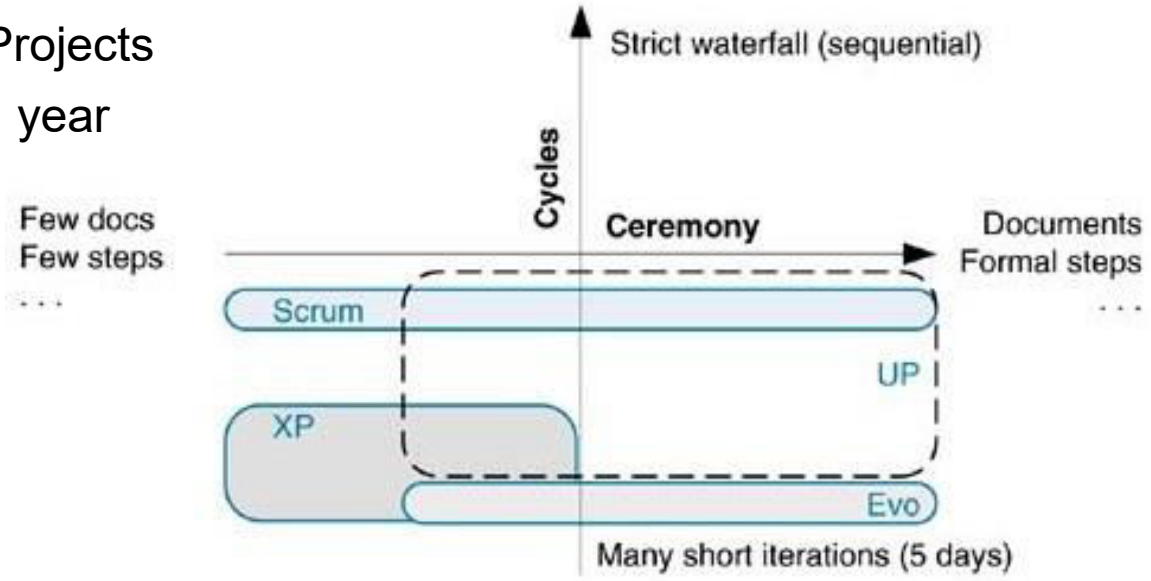
- **Active Engagement of the Customer**: the customer has to be continually involved in the project; however, while this can be seen as an advantage, it requires a lot of time and effort to manage the process
- **High Visibility may lead to Scope Creep**: as the Scrum methodology makes problems more visible, any problems arising at the end of each phase can also lead to "scope creep".
- **Small Teams losing Focus on the Big Picture**: while Scrum encourages small teams, it implies that larger teams may have to be broken down into smaller sizes which could result in the smaller teams loosing sight of the overall project focus.
- **Increase of Project Cost**: Scrum requires a certain level of training for all users which could increase the overall cost of the project.
- …

Extreme Programming (XP) – A Set of Skillful Practices:

➔➔

Extreme Programming (XP) – Core Values

- Communication, Simplicity, Feedback, Courage
- “Informal” (low on ‘ceremony’, usage of *Story Cards*)
- Small Teams (<10)
- Non-mission-critical Projects
- Delivery Schedule <1 year



Source: T1-Chap8

XP Lifecycle

EXPLORATION	PLANNING	ITERATIONS TO FIRST RELEASE	PRODUCTIONIZING	MAINTENANCE
Purpose: - Enough well-estimated story cards for first release. - Feasibility ensured.	**Purpose**: - Agree on date and stories for first release.	**Purpose**: - Implement a tested system ready for release.	**Purpose**: - Operational deployment	**Purpose**: - Enhance, fix. - Build major releases
Activities: - prototypes - exploratory proof of technology programming - story card writing and estimating	**Activities**: - Release Planning Game - story card writing and estimating	**Activities**: - testing and programming - Iteration Planning Game - task writing and estimating	**Activities**: - documentation - training - marketing - . . .	**Activities**: - May include these phases again, for incremental releases.

Source: T1-Chap8

XP Core Practices (12)

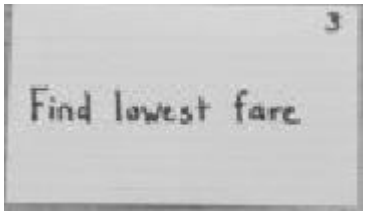
1. Planning Game
2. Small, Frequent Releases
3. System Metaphors
4. Simple Design
5. Testing
6. Frequent Refactoring
7. Pair Programming
8. Team Code Ownership
9. Continuous Integration
10. Sustainable Pace
11. Whole Team Work together
12. Coding Standards

Pitfalls of XP

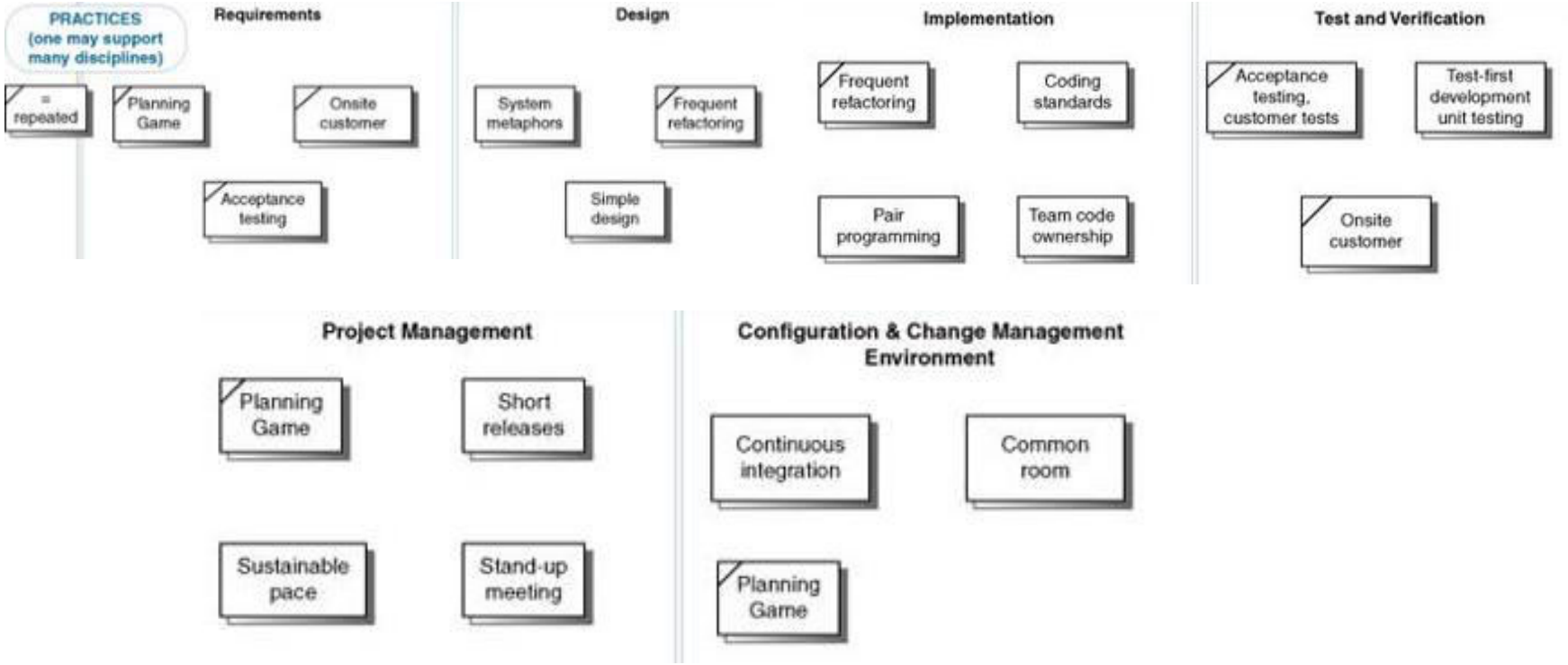
- Requires presence of Onsite Customer or his/her Proxy
- Relies on Informal oral understanding – difficult to ramp-up to speed new members or in large projects
- XP practices are highly interdependent and tightly coupled—we can’t be selective in any
- No ‘Standard’ means of documenting design
- Pair programming may not be favoured by all – *programmers are loners!*
- Lack of Focus on Architecture (relies on simple, if not fragile design and refactoring)

Extreme Programming (XP) Work Products

- Minimalist approach towards Requirements Gathering: Story Cards (sketches, visuals, post-it notes,…) representing *Features (not Use-cases/Scenarios)* – just cue-cards for discussion
- Task-list: containing Stories gathered for the Iteration (Task-card); Task-effort ~ 1-2 days
- Visible Wall Graphs for all to communicate with each other—progress of Stories, Test-cases, etc. using Simple Metrics



XP Practices



Source: T1-Chap8

XP – Other Common Practices/Values

- Embrace Change with Onsite Customer
- Volunteering rather than by Assignment
- Light Modeling (Just enough to get stared)
- Minimal Documentation
- Daily Metrics (few vitals) for Progress Tracking
- Incremental Infrastructure (no upfront investment)
- Daily Standup Meetings
- Simplicity – “Do the simplest thing that could possibly work.”
- Communication promoted through Pair-programming

“The practices are what you do. The values are how you decide if you are doing it right.”

Test Driven Development (TDD):
A Test-before-Code XP Practice

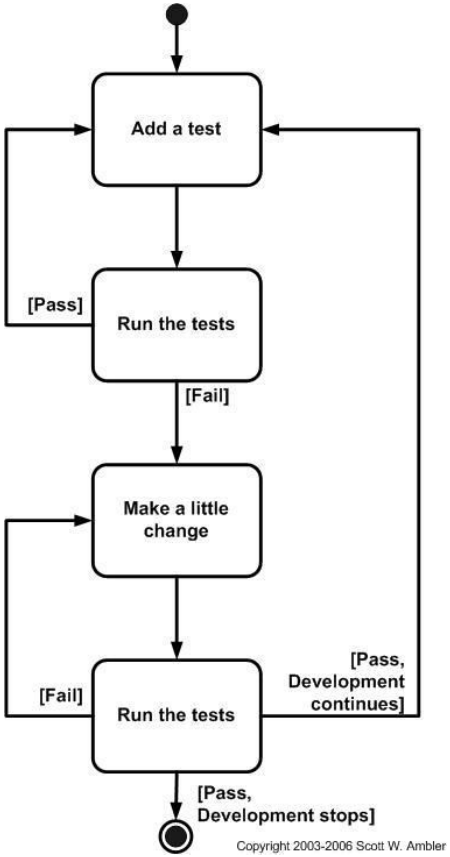
➔➔

Test-Driven Development

- Writing Test-cases before Coding (Unit Tests are written before writing the Code to be tested)
- Adoption of Short Iterative Development Cycle & Automated-Test Suites
- Eliminates “coder bias”

- “test with a purpose" - know why you are testing something and to what level it needs to be tested
- “achieve 100% coverage test” – every single line of code is tested
- “well-written unit-tests provide working specification of functional code” (code ~ documentation, tests ~ specifications)
- “proxies”: (unit-tests ~ design specifications, acceptance-tests ~ requirements)

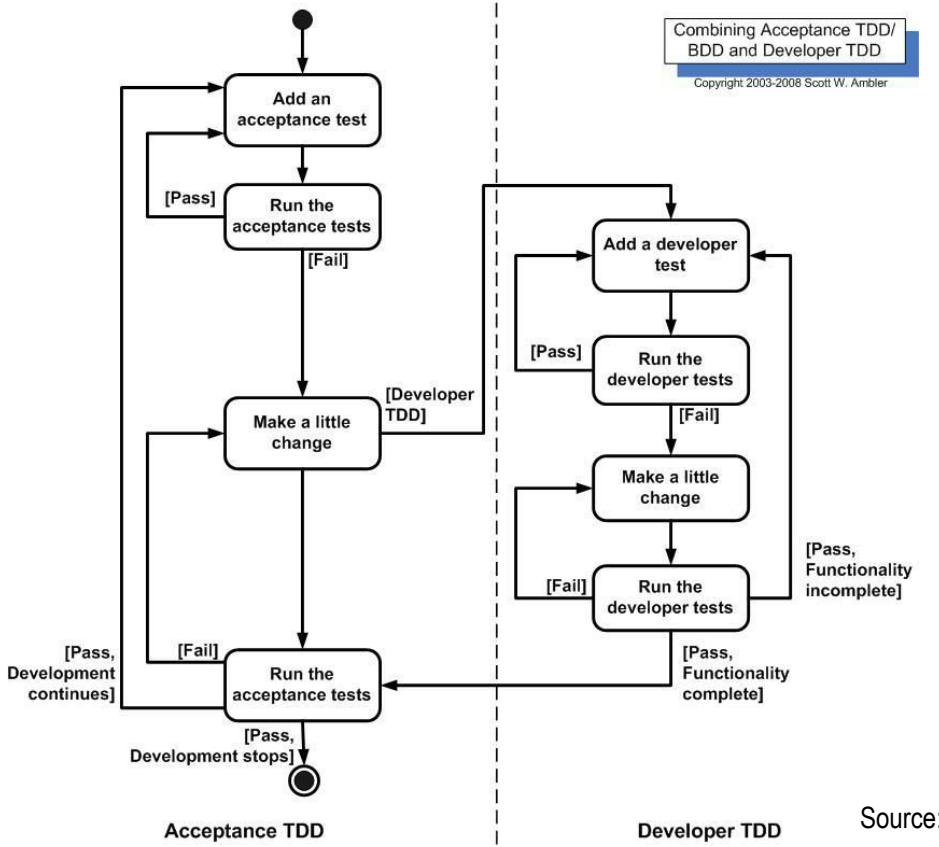
Test-Driven Development (TDD) =
Test-First Development (TFD) + Refactoring



- Evolutionary Approach to Development
- Write a Test-case first that fails before you write new functional Code; Coding evolves to fulfil those Test-cases, and then Refactor the Code
- Is TDD an Agile Requirements Capture technique or Programming Technique (than a Validation technique)?
- A Way to arrive at Clean Specifications!
- Does NOT replace traditional Testing, ensures effective Unit Testing

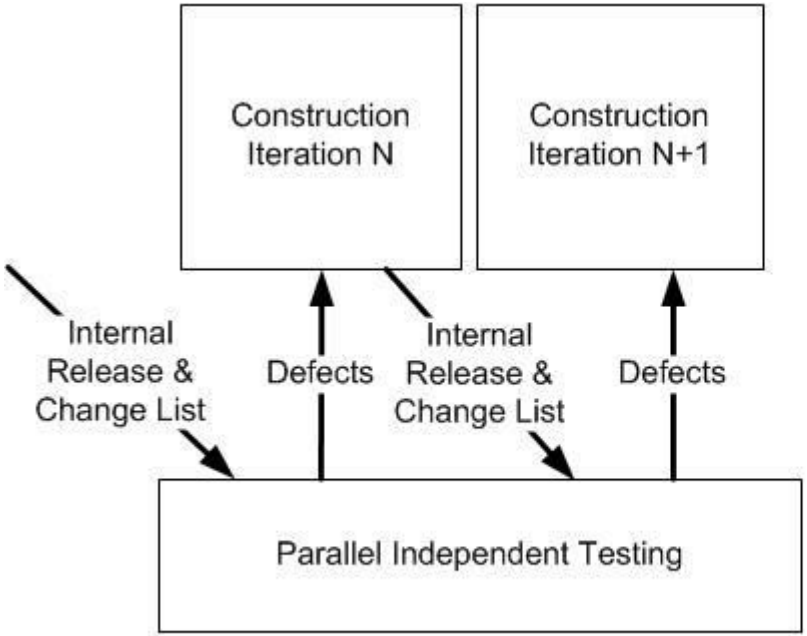
CUnit
DUnit (Delphi)
DBFit
DBUnit
DocTest (Python)
Googletest
HTMLUnit
HTTPUnit
JMock
JUnit
Moq
NDbUnit
NUnit
OUnit
PHPUnit
PyUnit (Python)
SimpleTest
TestNG
TestOoB (Python)
Test::Unit (Ruby)
VBUnit
XTUnit
xUnit.net

TDD: Acceptance TDD, Developer TDD



Source: http://agiledata.org/essays/tdd.html

TDD is NOT an End to Itself...



Confirmatory Testing
- Acceptance TDD
- Developer TDD
- Developer regression testing

Independent Testing:
- Pre-production integration testing
- Other expensive/skillful testing

Agile Methodologies - Summary

- SCRUM is a Process in Agile Methodology which is a creative combination of an Iterative and Incremental Methods; it is prescriptive in nature and has well-defined Roles
- XP – set of Practices that take Programming to the Extreme, i.e. just enough, just-in-time with minimalist approach; relies heavily on people
- TDD – an XP technique turns traditional Code Development upside-down, i.e. write test first and then write (just enough) code to fulfil that test and move-on to write the next test, then code, etc.; relies on availability of automated test tools

Thank You

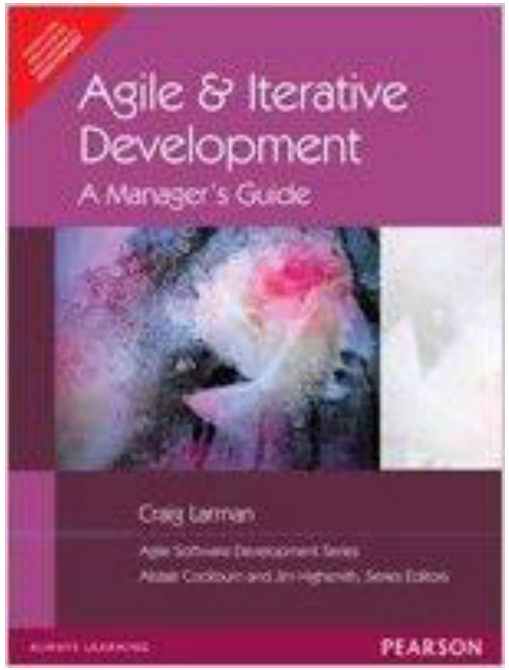
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Requirements Management in Agile

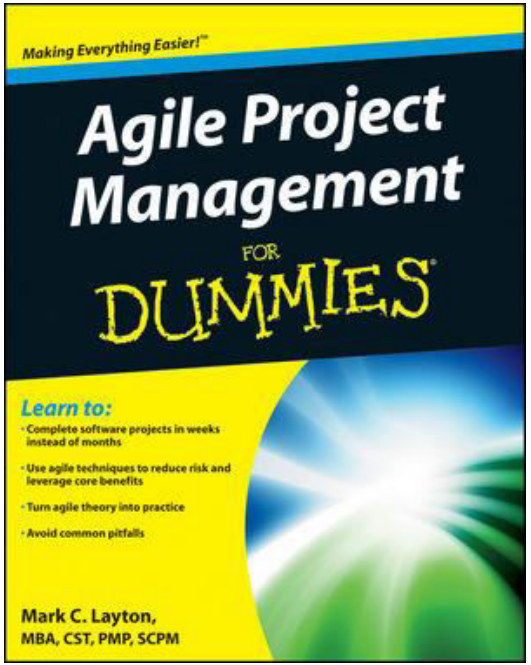
- Prof K G Krishna

Text/Reference Books

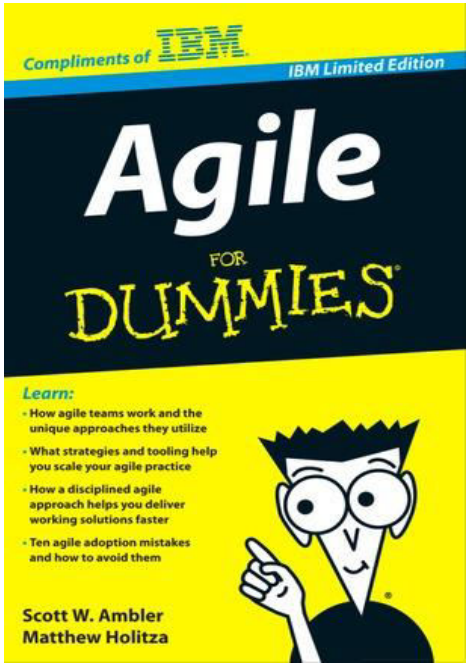
T1



T2



Compliments of IBM



➔ 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

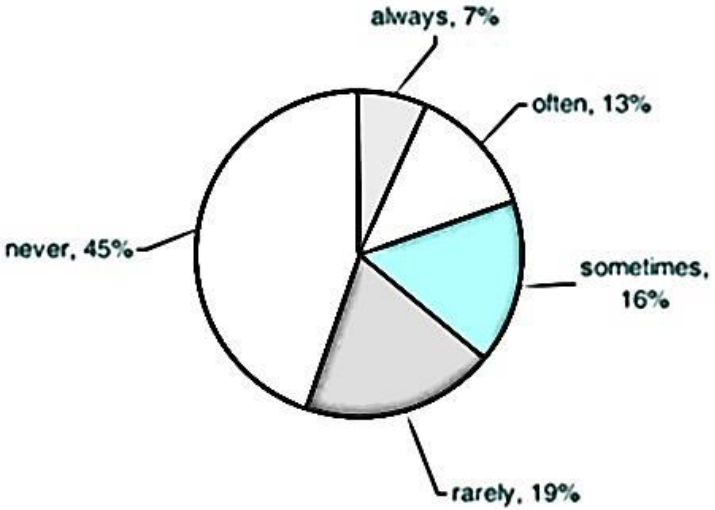
Agile Requirements Management

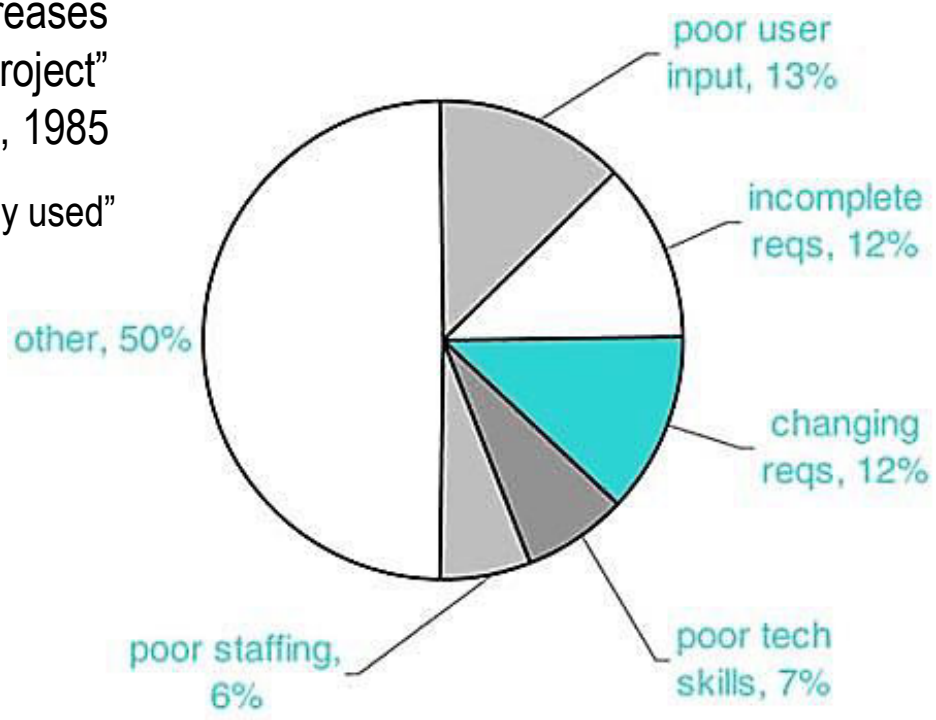
- Managing Requirements Iteratively
- User Stories as Requirements
- Size/Effort Estimation
- Prioritization Techniques
- Preparing the Product Roadmap

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

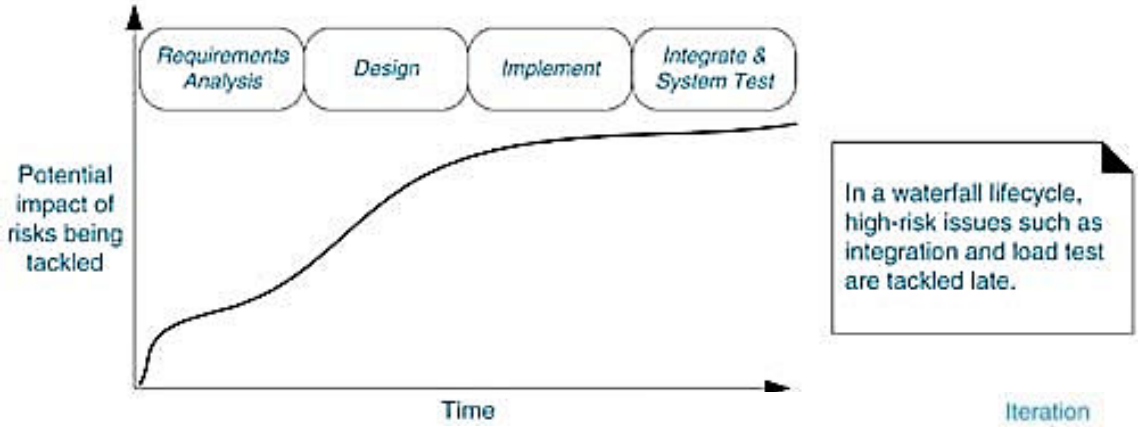
"45% of Features never used and ~20% rarely used"





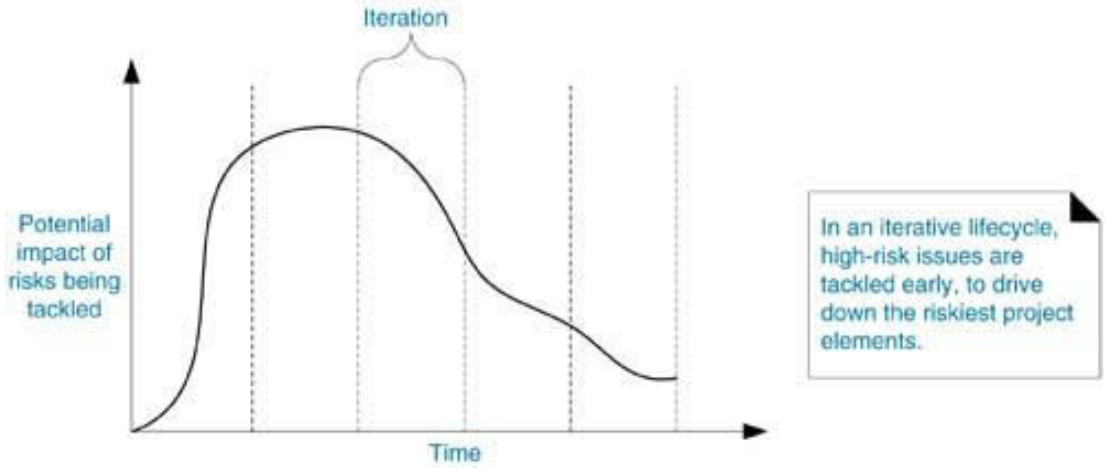
Source: (T1) / Standish Group Report, 2004

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



Risk Profile in Waterfall Model

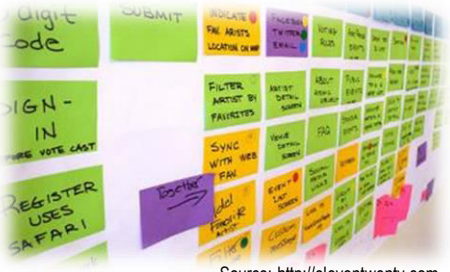
Risk Profile in IID



Source: (T1-Chap 5)

Identifying Product Requirements in Agile

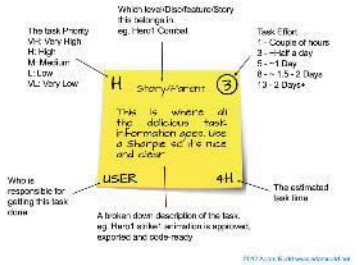
- From the Product Roadmap, arrive at Initial Set of High-priority Requirements
  - Product Roadmap ➔ Requirements ➔ Releases ➔ Sprints
- Requirements ➔ Logical Groups
- Decompose Requirements into: (level of detailing depends on early product definition)
  - **Themes**: logical grouping of features 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

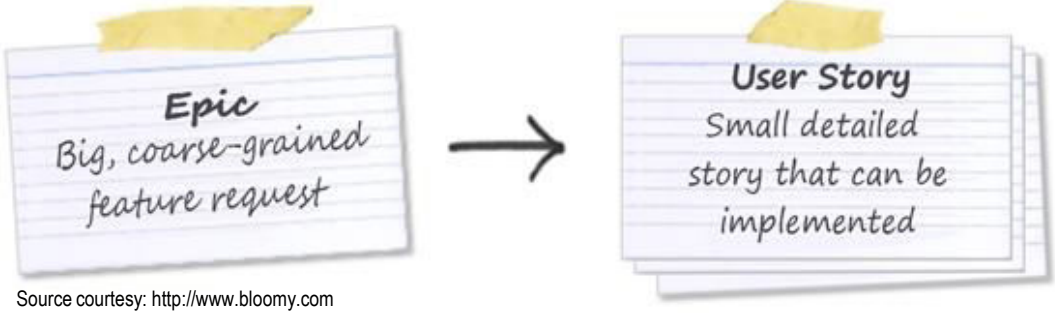


Source: http://eleventwenty.com

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





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”

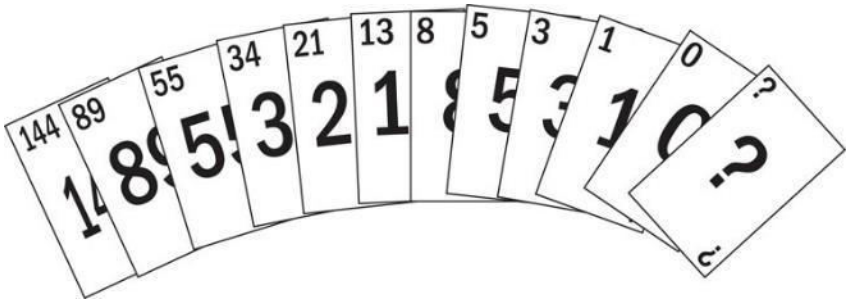
  1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

  - 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

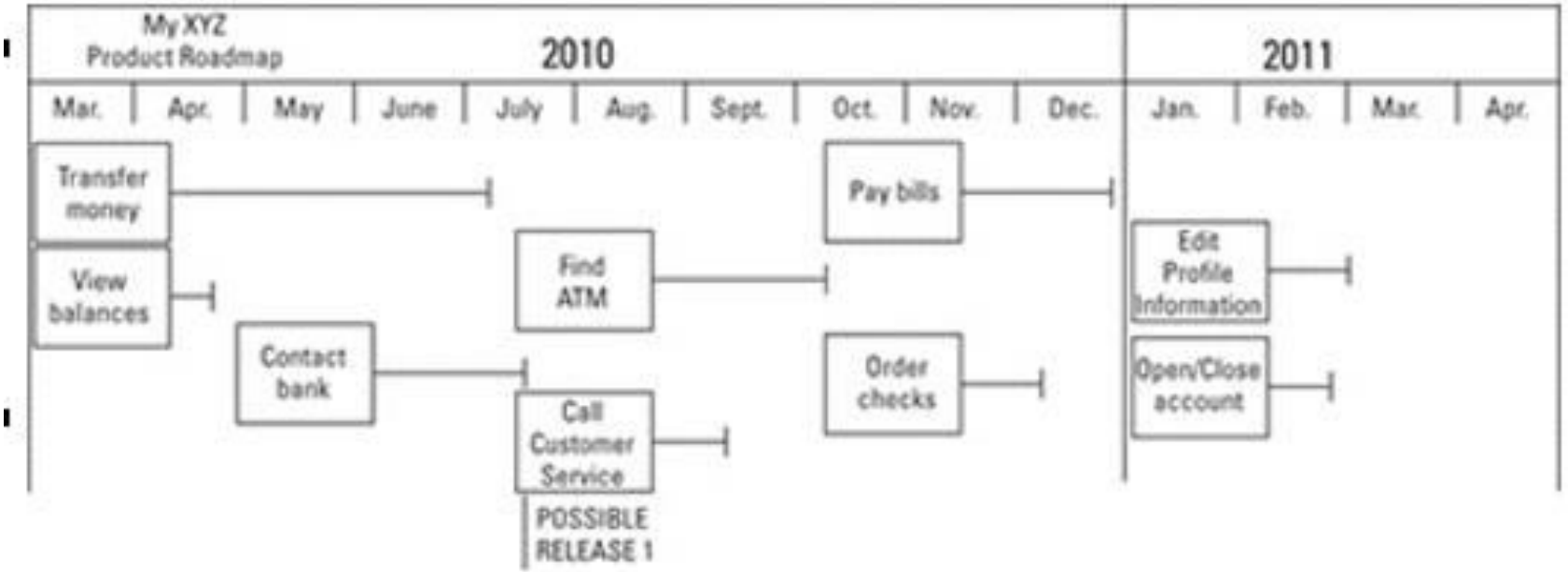
Estimates should be for the total ***Done*** – Developed, Integrated, Tested and Documented

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

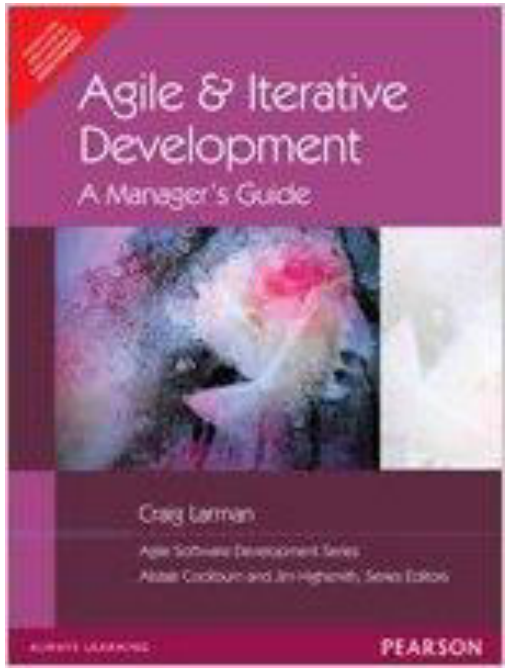
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Release Planning in Agile

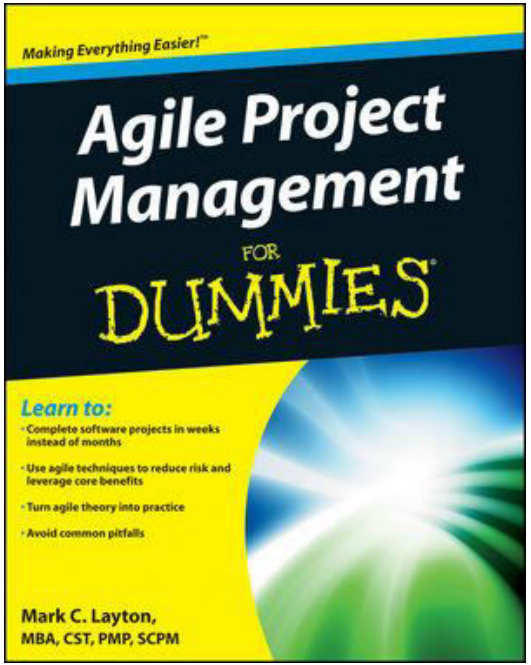
- Prof K G Krishna

Text/Reference Books

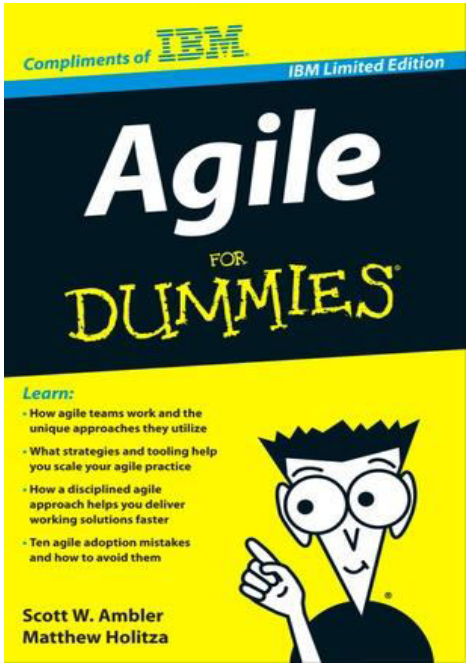
T1



T2



Compliments of IBM



➔ 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

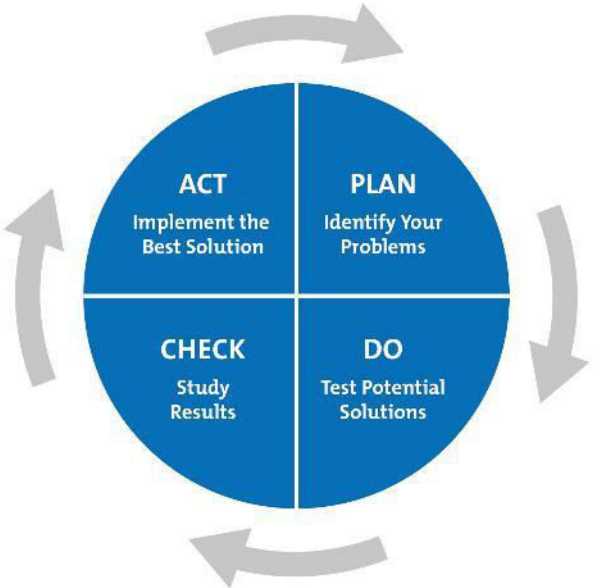
Release Planning in Agile Methods

- Characteristics of Agile Planning
- Stages of Agile Planning
- Release Planning

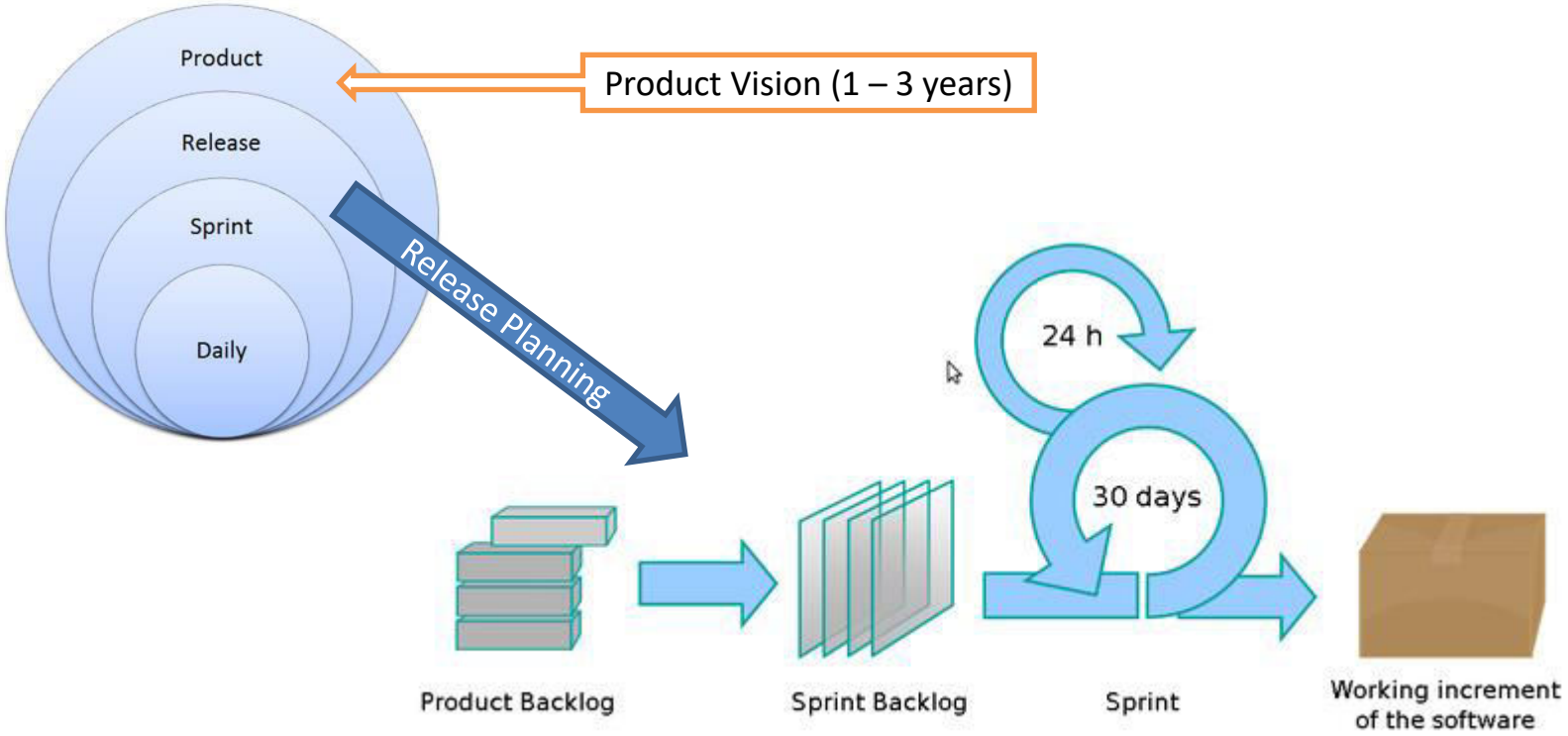
Planning is *Continuous*!
(Deming's Continuous Improvement Cycle)

- Planning is NOT a One-time Activity at the Beginning of a Project
- Part of PDCA (Deming's Cycle of Continuous Improvement)
- PDCA vs. "Plan-the-Work, Work-the-Plan" (Waterfall Model)

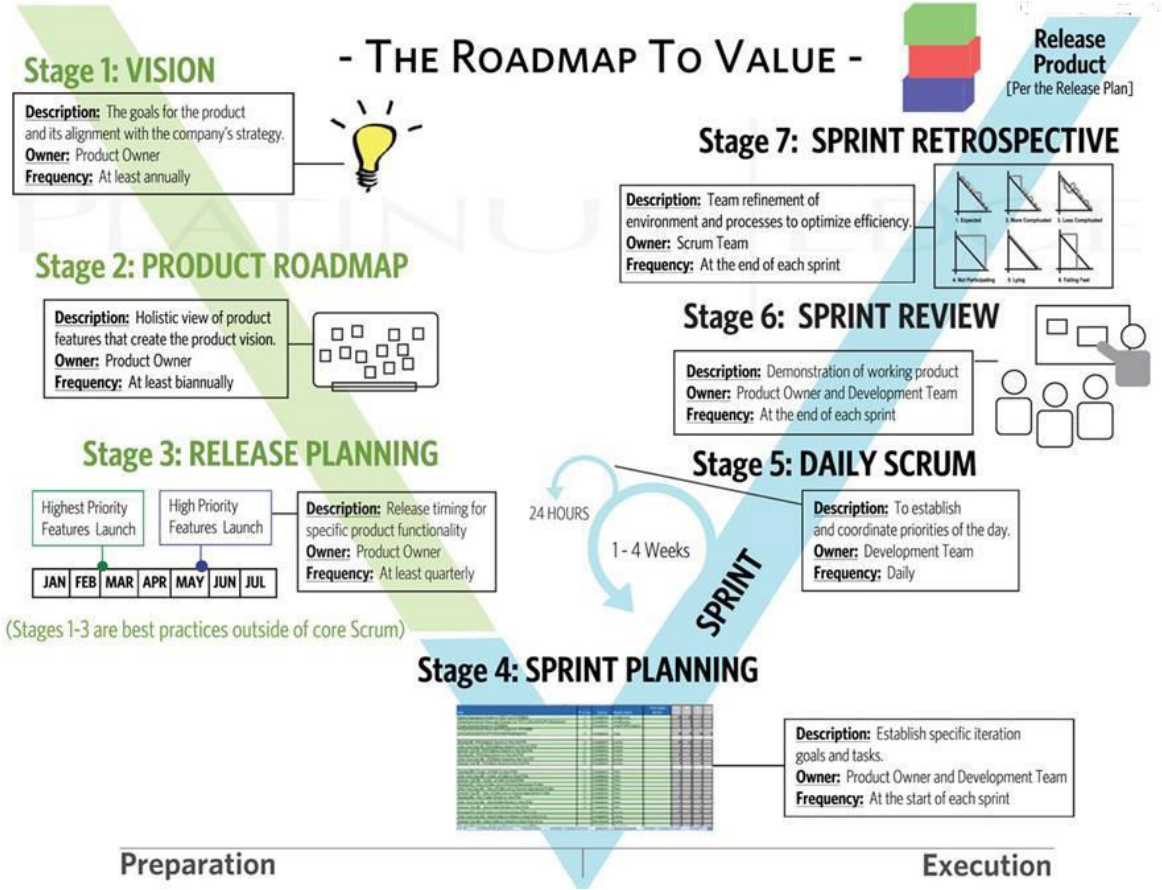
- Agile Planning
  - Just-In-Time Planning / "Situational Planning"
  - Agile Planning is Continuous throughout the Project



Levels of Planning in Agile



Planning at *Every* Stage in Agile (SCRUM)



Source: (T2)

Key Characteristics of Agile Planning

- Planning occurs at every Stage
- Planning, like Development is Iterative
- Just-enough and Just-in-Time Planning at every Stage
- Progressive Detailing - start with a broad plan and narrow it progressively
- Prioritizing Value at every Stage: Add High-value Requirements first
- Adapt the Plan after Feedback at every Stage

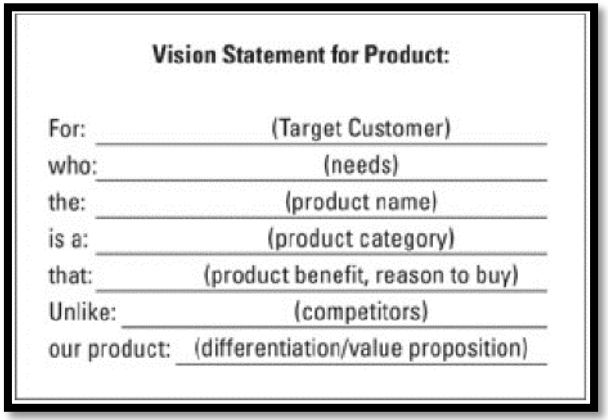
Plan-**D**o-**I**nspect-**A**dapt

Agile Release Planning Stages in Detail

➔➔

Stage-1: Defining Product Vision

- The Vision Statement (Summary) communicating Product Strategy
  - Product Goals aligned with Strategy
  - Owned by the Product Owner
  - Frequency: annually (minimum)
- Developing the Vision Statement
  - Develop the Product Objective
  - Draft the Vision Statement
  - Validate the Statement with Stakeholders and Revise after Feedback
  - Finalize the Vision Statement
- The Vision Statement must be:
  - Clear with Simple language
  - Non-technical (everyone can understand)
  - Brief (in one or two lines)
  - Internally focused (at development, not a sales pitch)



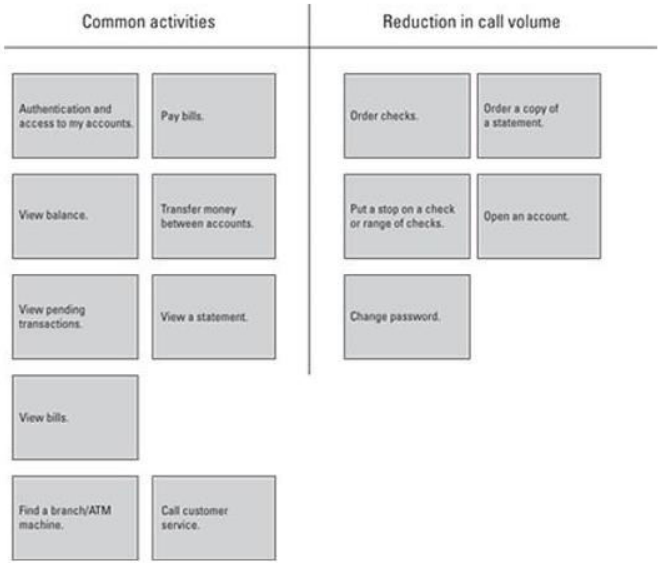
Vision Statement for Product:

For: (Target Customer)
who: (needs)
the: (product name)
is a: (product category)
that: (product benefit, reason to buy)
Unlike: (competitors)
our product: (differentiation/value proposition)

Stage-2: Holistic Product Roadmap

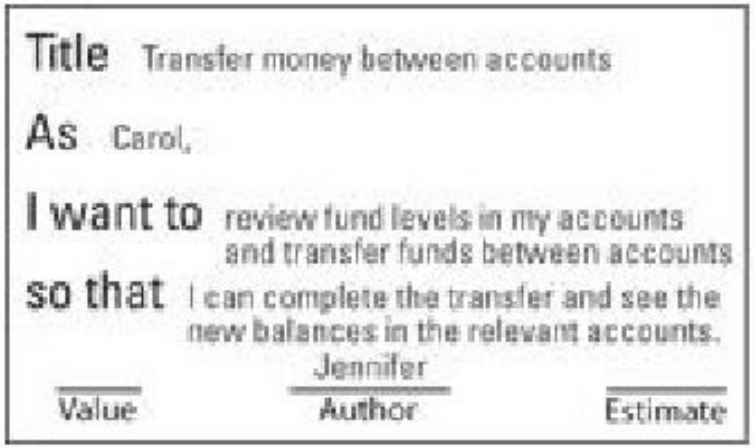
- Derived from Project Vision
- Identify Requirements that Define Product Roadmap
- Arrange Requirements into Logical Groups
- Estimate Effort and Prioritize Requirements
- Set High-level Time-frame for each Group of Requirements
- Beginning of the Creation of Product-Backlog

➔ Update the Product Roadmap throughout the Project (unlike Project Vision)



Stage-3: Release Planning

- Elaboration of Project Roadmap into Details
- A Release is a *Minimum set* of Marketable Requirements
- Requirements Breakdown Structure (WBS) by granular Decomposition
- “Epic-story ..→ User-stories”
- User-story: Simple Description of Requirements (user-walkthrough or benefit statement)
- Create *Personas* for each Class of Users
- Identify each Story by ID and set a Value (in terms of benefits or priority)
- Estimate Effort for each Story (“story points”)
- Document on Index-cards or Post-it Notes
- Product Owner manages the Stories
- Break Stories further into detailed Features/Tasks for Sprint Planning,

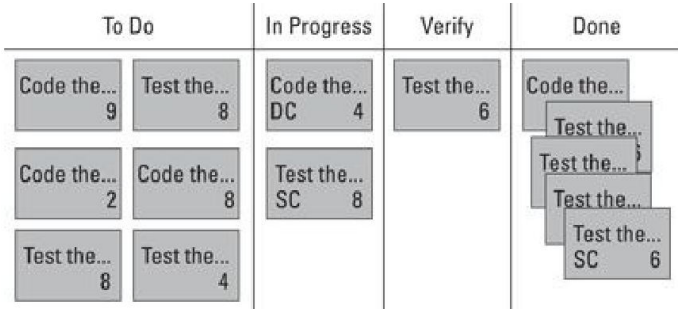


Stage-4: Sprint Planning

- A Sprint is a consistent (fixed-length ~ 1-4weeks) Iteration of Time in which Product takes Demonstrable Shape
- Sprint Backlog: List of User-stories (prioritized) with detailed WBS and Time-estimates
- Each Task to be completed in 1-2 days max (no over-committing, reduce Scope if necessary)
- Task Done ➔ Developed, Integrated, Tested and Documented
- Development Team work only on one Requirement (Story/Tasks) at a time
- Only the Development Team can modify the Sprint Backlog
- Each Sprint includes:
  - Sprint Planning (max. 2 hours at the beginning of every week)
  - Daily Scrum Meeting (standup meeting for 15-20mins)
  - Development (bulk of the effort in Sprint)
  - Sprint Review
  - Sprint Retrospective

Stage-5: Daily Scrum

- Each Day can be a Planning/Replanning Day (Daily Scrum meeting)
- Daily Scrum Meeting to be Brief (~15-20mins Standup)
- Scrum Master to facilitate the meeting (review progress, roadblocks,…)
- Participants: Product Owner, Development Team and Scrum Master
- Focus of Meeting: Coordinate/Prioritize (Not to solve Problems)
- Update Sprint Backlog Daily (at the end of the meeting) and make it visible to everyone in the team



Stage-6: Sprint Review

- Sprint Review Meeting at the end of each Sprint to review and demonstrate User-stories that were completed during the Sprint
- Entire Team (Product Owner, Development Team, Stakeholders and Scrum Master) participates
- Product Owner confirms Status of Completion of Sprint (ready for Release of partial-working product)
- Invites Feedback from all Stakeholders
- Scrum Master to update Product-backlog for the next Sprint Planning

Stage-7: Sprint Retrospective

- Post Sprint Meeting (Scrum Master, Development Team and Product Owner) to discuss *the experience* of the Sprint – What went right and what went wrong
- Focus is on Continuous Improvement of the *Process* to improve *Efficiency* and *Velocity* of throughput
- Adapt Scrum Processes to improve morale of Team and their Work-life balance
- Lasting for ~45mins maximum for every week of the Sprint
- Opportunity to *Inspect and Adapt* (Plan-Do-Inspect-Adapt) Scrum Process

Preparing for Release

- End of every Sprint to be *Working and Demonstrable* Product
- A Sprint outcome can be a Release Sprint meant for Customers
- Sprint-backlog Items in a Release Sprint might include:
  - Creating User Documentation for the just finished Release
  - Testing of Key Non-Functional Requirements (Performance, Security, Load balancing,…)
  - Compliance with mandatory Organizational or Regulatory Procedures
  - Integrating with existing Organization's Enterprise Systems
  - Preparing Deployment Package (Installation scripts, etc)
  - Preparing a Release Note
- Note that Development for Regular Sprint is different that of Release Sprint
- Sprint Review meeting for Release to include Customer and Key Stakeholders from Marketing and Operations as well

Summary: Release Planning

- Planning is Continuous in Agile – at every Stage in the Scrum Process (Release Planning, Daily Scrum, Sprint Review)
- Each Release may span one or more Sprints
- Agile Planning is planning for a pre-determined number of Releases to Customers
- Planning for Release involves more Tasks (Sprint-backlog) than for regular Sprint
- Sprint Retrospective Meeting identifies Opportunities for Improvement in the Scrum Process and implements them before the next Sprint cycle

More Meetings, More Sharing of Information/Feedback, Near-Real-Time Visibility into the Product, and finally 'Unsurprising' and Acceptable Product Release(s)!

Thank You