Software Development Lifecycles, Agile and Scrum

3 day workshop

vedavit

Trainer : Hari Prasad Thapliyal

(PMP, PMI-ACP, CSM, MBA, MCA, PGDOM, PGDFM, CIC, PRINCE2-Practitioner)

[**PMI-ACP :** #318050**]**

[PMP : #1563385]

Hari.prasad@vedavit-ps.com
+91-9535999336

This content is legal only for participants of Toshiba, Bangalore Dec'16 Training

Faculty & Participants
An Introduction

Name, Role, Experience (Total, Agile),
Hobby, Expectations

Workshop Ground Rules

- ✓ Please keep your mobile on the silent mode. Always take your calls outside the training room.
- ✓ No corner talk! Discussions only when group discussion is allowed
- ✓ Keep your focus on the ongoing topic. Await your turn during the questionnaire round.
- ✓ Strictly follow the workshop schedule for the management of the time.
- ✓ There is parking lot. Write you questions and post with your name on parking lot.
- ✓ Breaks only on agreed time
  - ✓ Tea
  - ✓ Lunch
  - ✓ Tea
- ✓ Everybody need to contribute
- ✓ Use your experience only for relating the processes and best practices. To avoid confusion keep it outside of the class. Unlearning is first and biggest learning to learn something new.
- ✓ Two Bowls

Workshop Objective

- ✓ Learn Various Software Development Lifecycles
- ✓ Learning Agile Project Management
- ✓ How to Apply Scrum Methodology to my Project

SDLC (Software Development Lifecycles)

What is SDLC?

- A systematic way of understanding problem and requirement, designing solution, developing product, testing product and deploying software product is SDLC

Why Various SDLC?

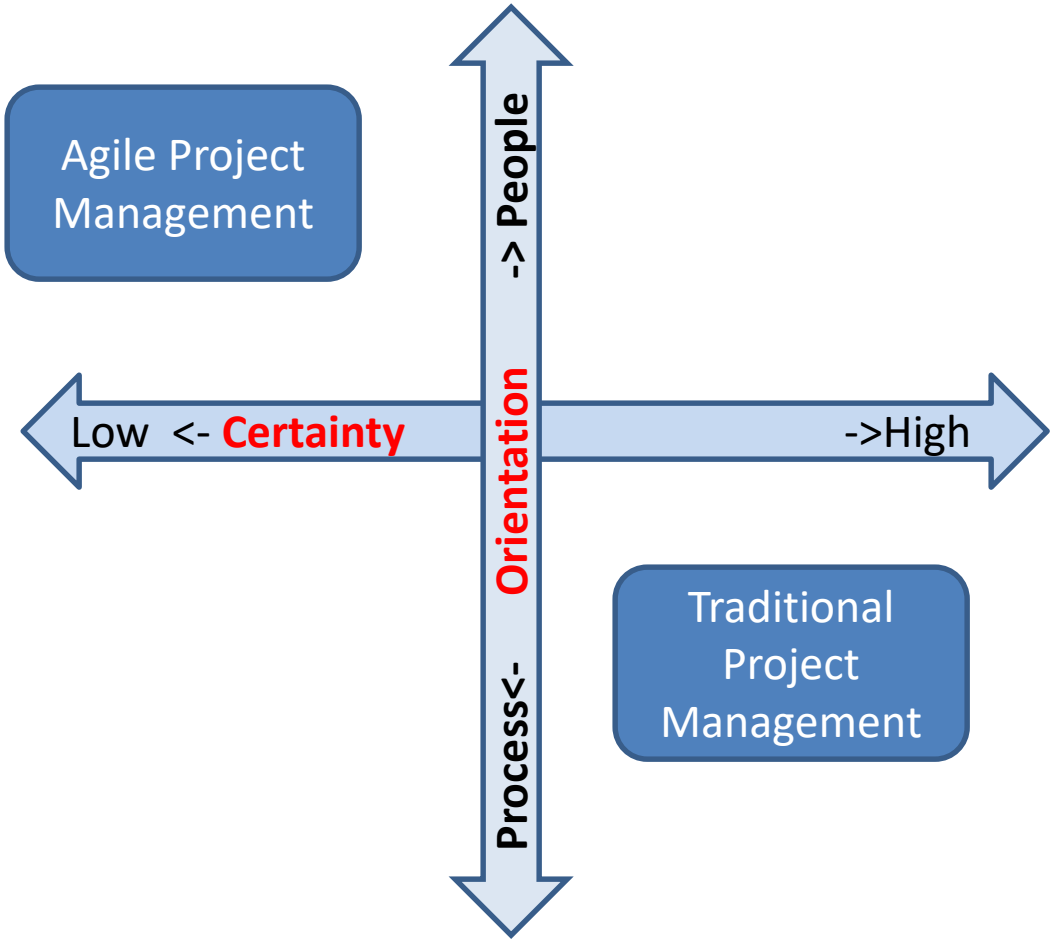
From one project to another project customer, user, businesses, organizations, technologies, team competencies, priorities are different. So SDLC depends upon…..

- What is important? Responding to change, Quality, Cost, Scope
- Team Capability: High to Low
- Requirements: Stable to Extremely Agile
- Technology: Stable to Extremely Complex or New
- Number of Developers: Low to High
- Criticality of output (quality): Low to Extreme

SDLC Depends upon

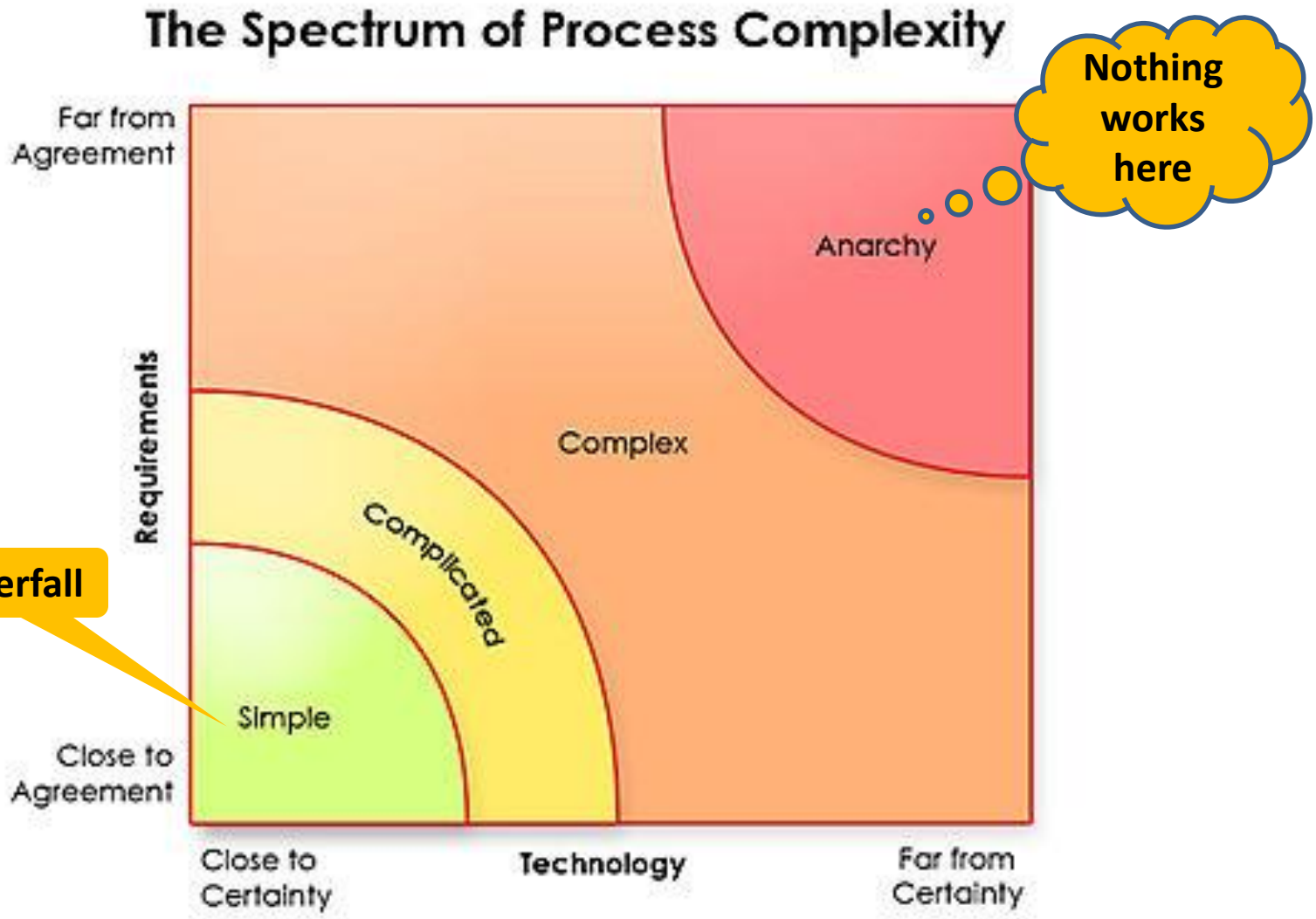
- Small Team or large team
- Greenfield Projects to enhancement project
- Unknown technologies to widely known technologies
- Simple technology to complex stack of technologies
- Project is addressing change to Large organization or Small Organization
- Project is addressing Local and small change or global and huge change
- Revamp of legacy system to enable certain feature in legacy system
- Documentation of legacy is extensively available or not available at all
- Team is distributed or co-located

Which PM Methodology is best?



Which PM Methodology is best?

Ralph Stacey's complexity matrix

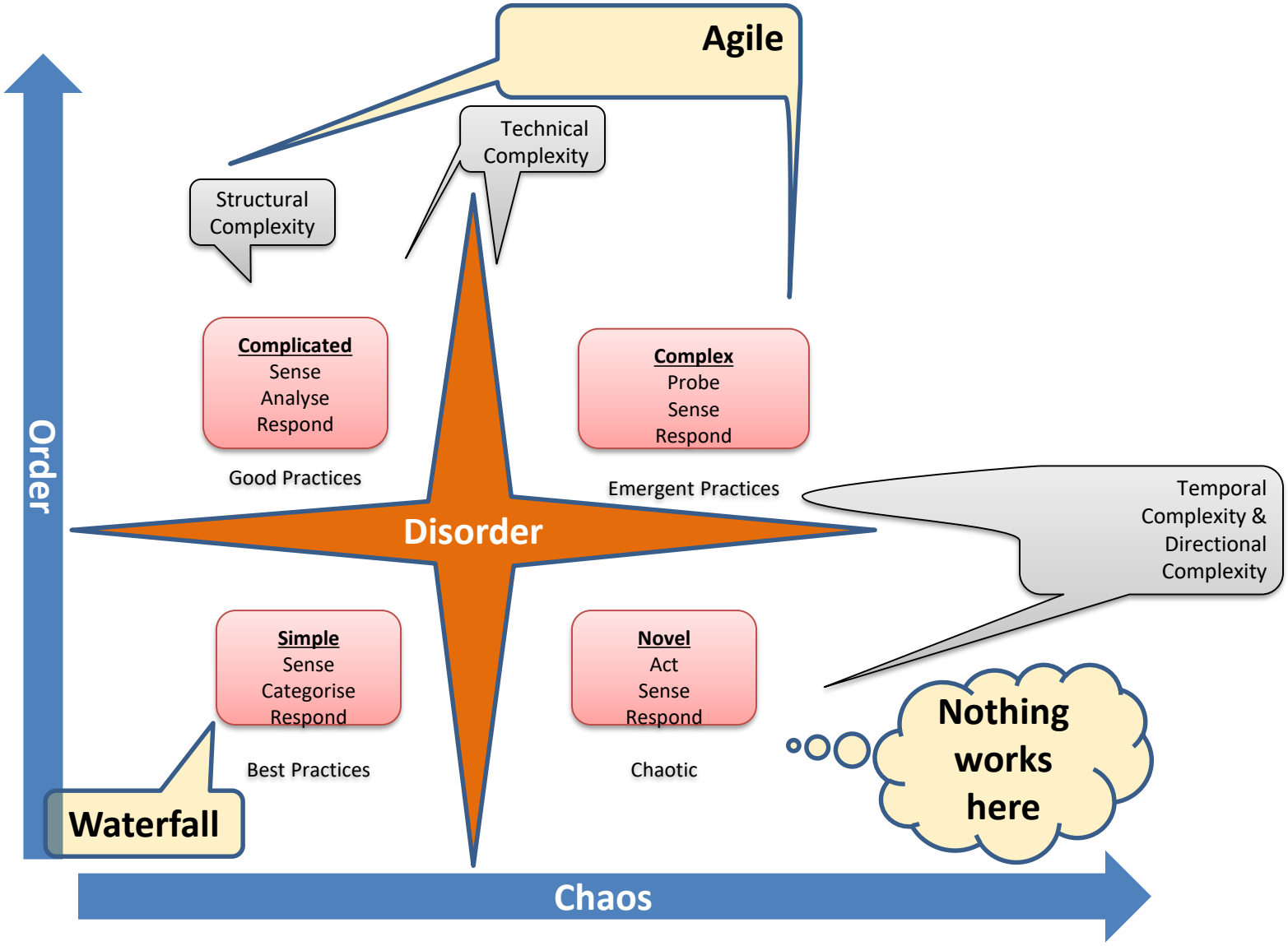


Types of Systems

Act	Probe	Sense	Analyze	Categorize	Respond	
✓		✓			✓	Chaotic
	✓	✓			✓	Complex
		✓	✓		✓	Complicated
		✓		✓	✓	Simple

- **Chaos:** Bus, Scooter, Cycle, Tram, Car, Pedestrian using the same road to commute is chaos. First you need to act to bring little order.
  - No practice helps here.
- **Complex:** As a head of department you want to remove corruption from your department. First you need to probe problem and its reasons.
  - Practices keep emerging. No standard practices helps.
- **Complicated:** One day your school going child is not interested in going school but you want him to go. First you need to sense the problem, its gravity.
  - Good practices learned from your elders and friends will help you here. Little tailoring may be required.
- **Simple:** In your kitchen you need to make tea but you realized there is no water, milk, tea, sugar! First you need to sense the problem then categories reasons and solutions.
  - Best practices will help here. No change is required in practices..

Types of Systems



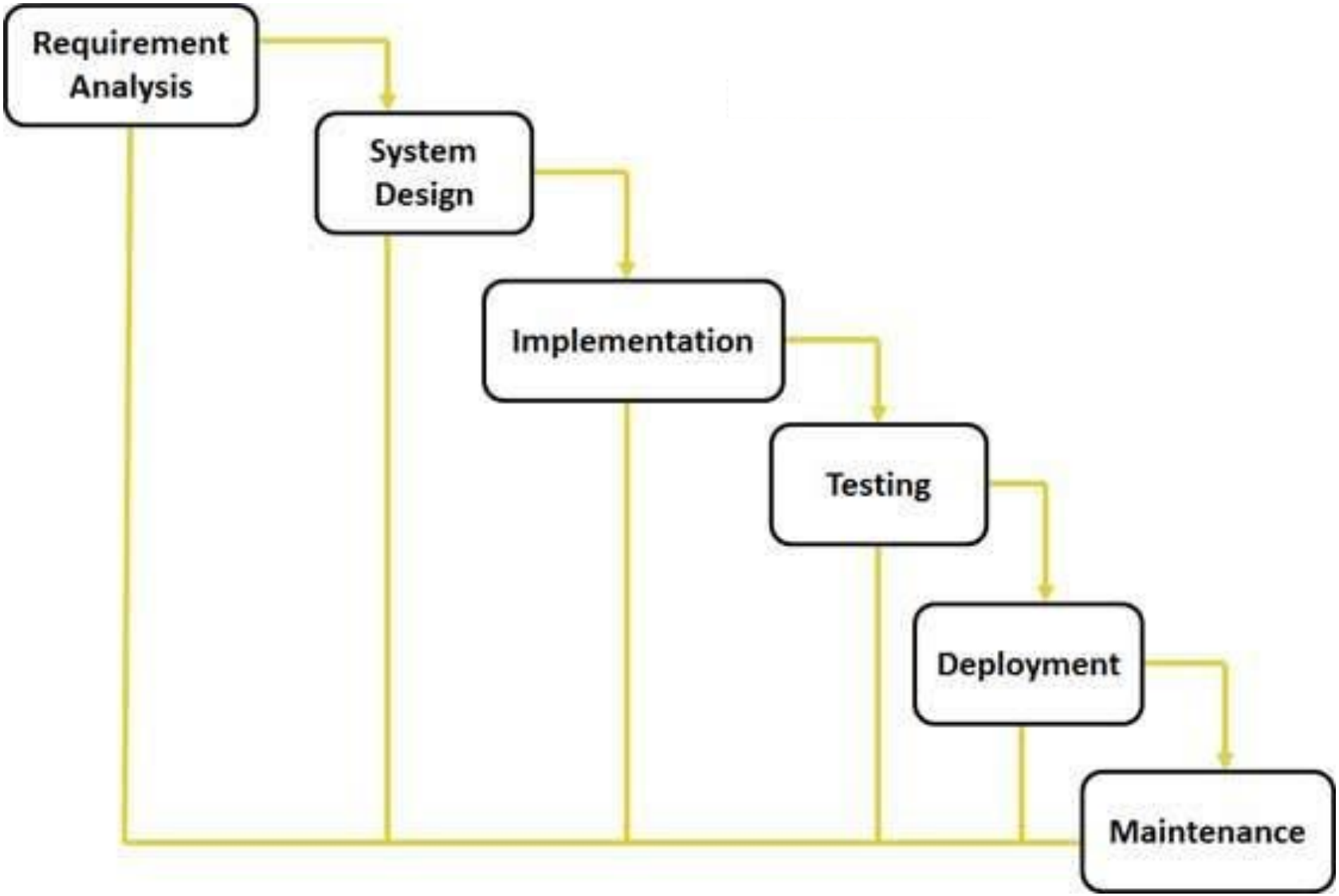
Types

- Linear
  - Waterfall
  - V Model
- Incremental & Iterative
  - Delivery of End Product Value Can Wait
    - Spiral
    - RAD
    - Prototyping
  - Delivery of End Product Value Can't Wait
    - Agile

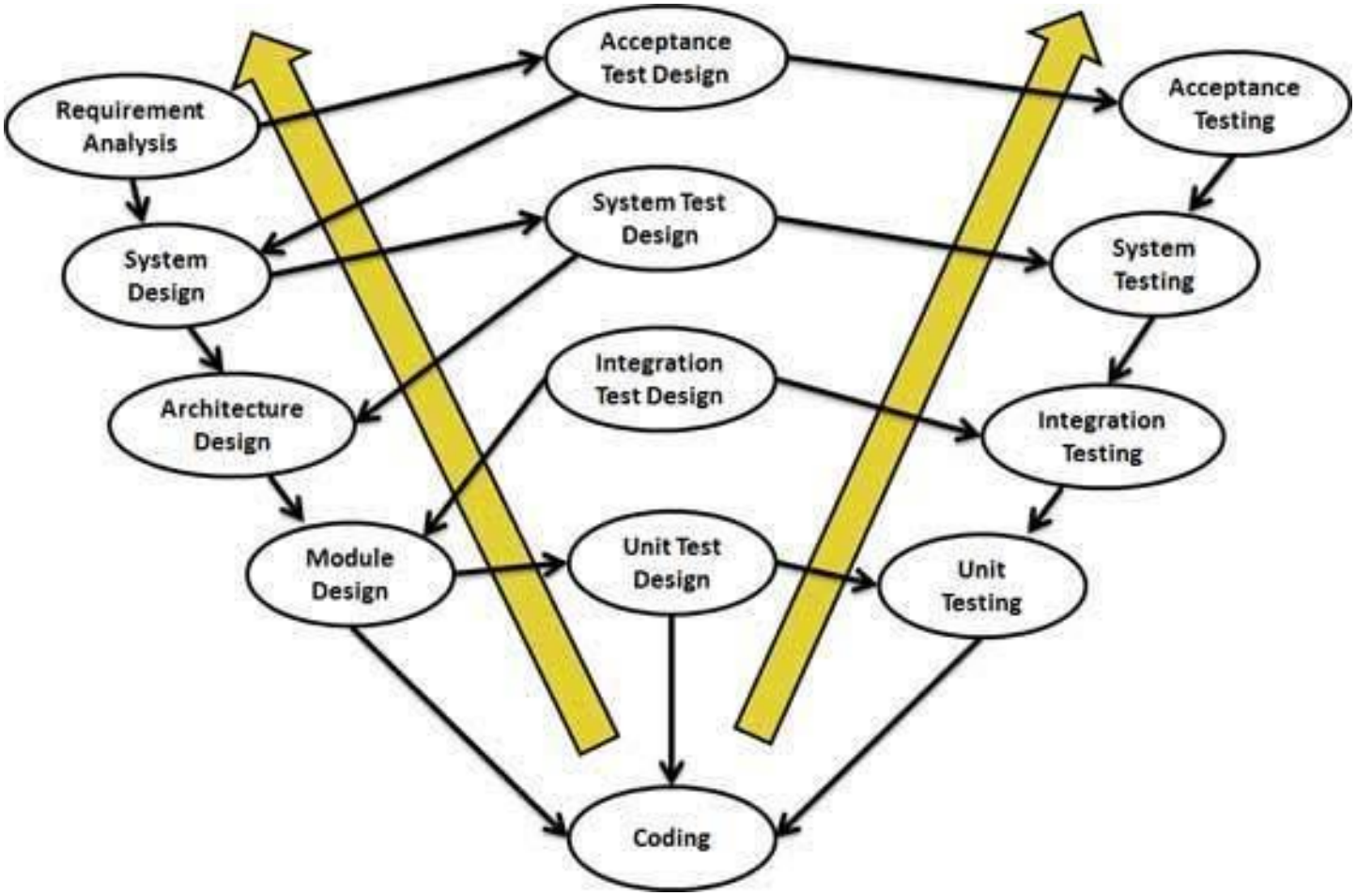
Agile

1. Requirements and solutions evolve through
2. The collaborative effort of self-organizing cross-functional teams
3. Adaptive planning,
4. Evolutionary development,
5. Early delivery, and
6. Continuous improvement,
7. Rapid and flexible response to change

Waterfall

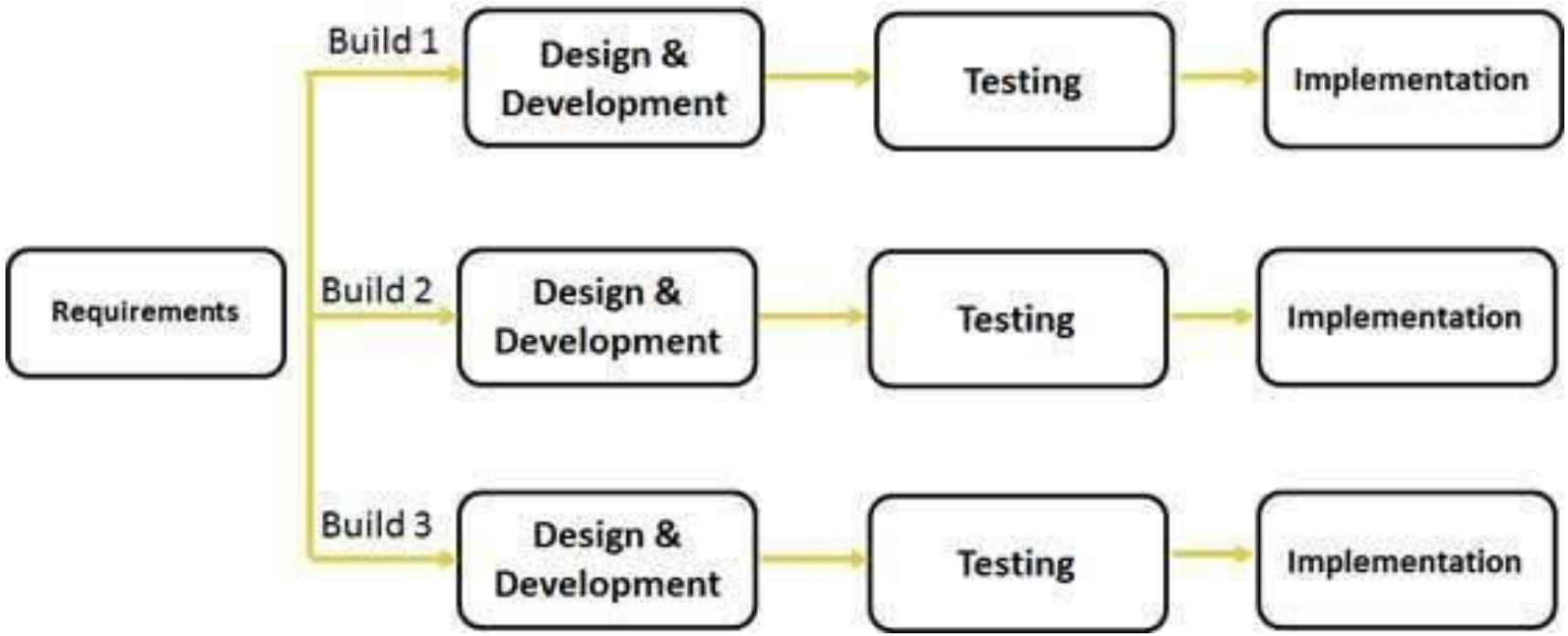


Verification and Validation model



Extension of the waterfall model and is based on association of a testing phase for each corresponding development stage

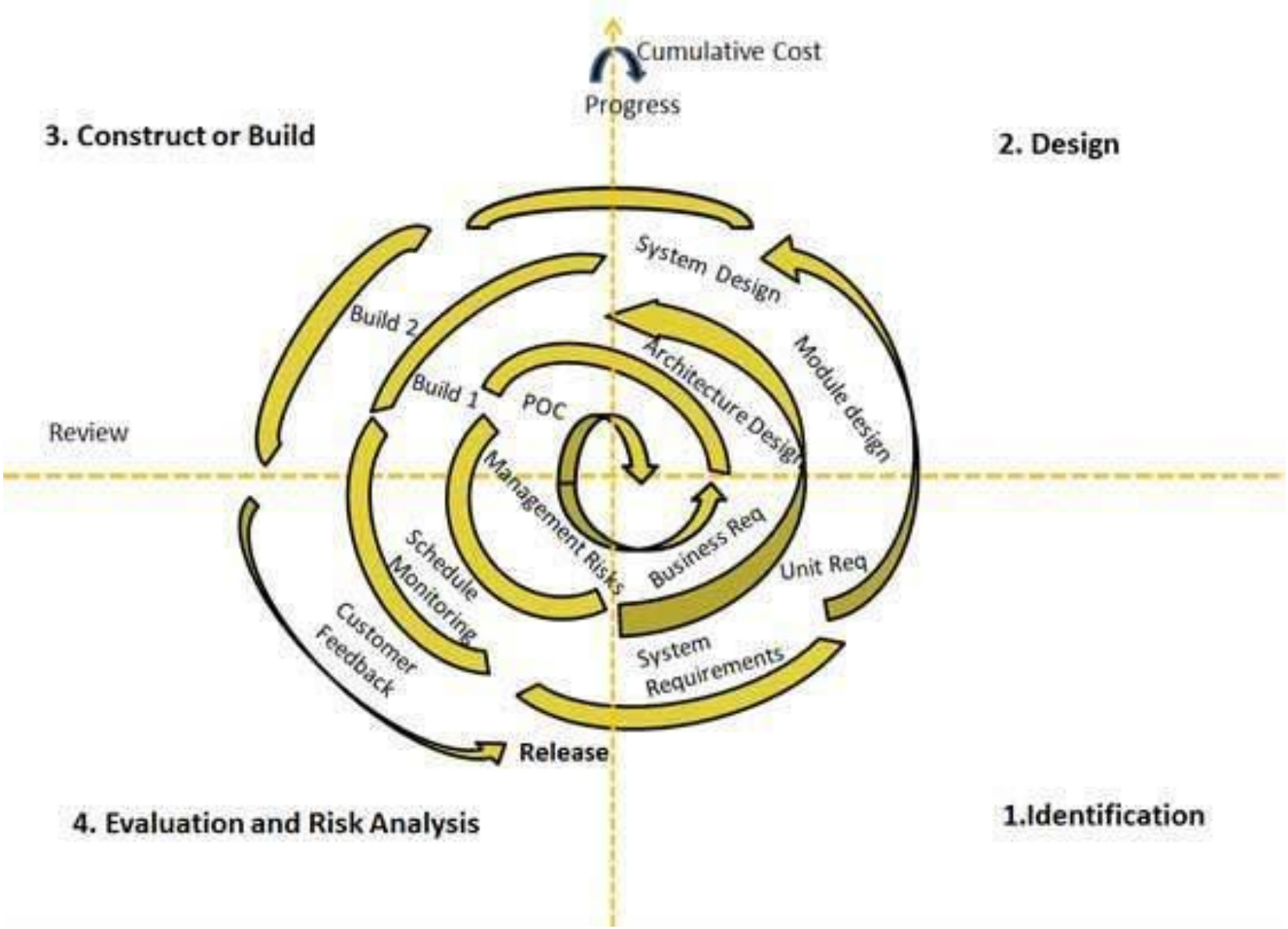
Iterative Model



Iterative Model

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill set are not available OR planned to be used on contract basis for specific period.
- There are some high risk features and goals which may change in the future.

Spiral Model

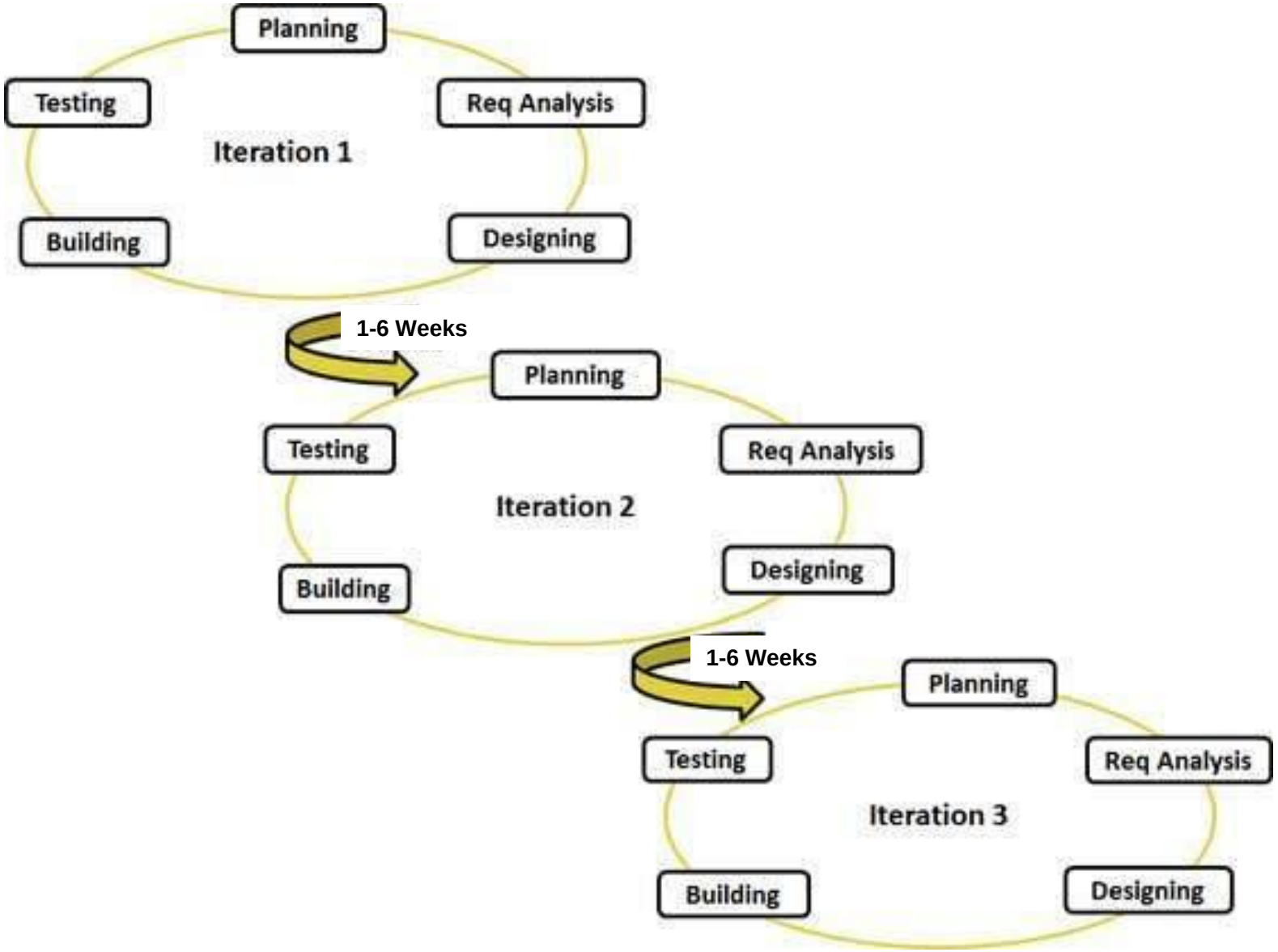


4 Phases in Spiral

- Identification
  - identification of system requirements,
  - subsystem requirements
  - unit requirements
- Design
  - conceptual design = baseline spiral
  - architectural design,
  - logical design of modules,
  - physical product design
  - final design.
- Construct & Built
  - POC = baseline spiral
  - working model of the software = build
- Evaluation & Risk Analysis
  - Risk Analysis includes identifying, estimating, and monitoring technical feasibility and management risks, such as schedule slippage and cost overrun.
  - After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

<u>Overlap of these phases is possible</u>

Agile Development

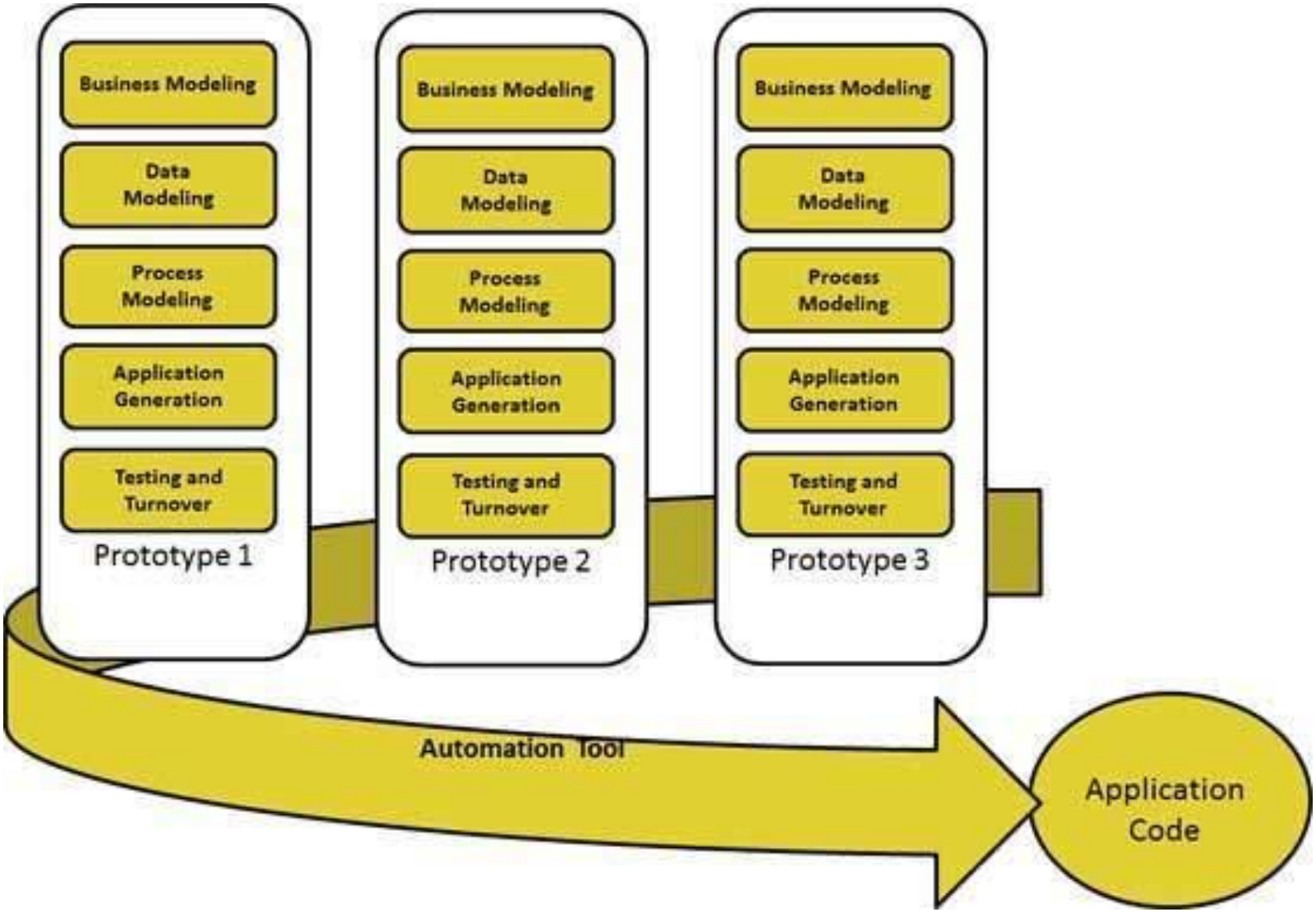


Agile Methods

1. Rational Unified Process (1994),
2. Scrum (1995),
3. Crystal (Clear, Maroon, ,
4. Extreme Programming (1996),
5. Adaptive Software Development,
6. Feature Driven Development, and
7. Dynamic Systems Development Method (DSDM) (1995).

- These are now collectively referred to as agile methodologies, after the Agile Manifesto was published in 2001.

RAD


Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 1
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 2
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 3
Automation Tool
Application Code

- RAD should be used only when a system can be modularized to be delivered in incremental manner.
- It should be used if there is high availability of designers for modelling.
- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the course of the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

Prototyping

- **Basic Requirement Identification:** This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.
- **Developing the initial Prototype:** The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed and the workarounds are used to give the same look and feel to the customer in the prototype developed.
- **Review of the Prototype:**The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.
- **Revise and enhance the Prototype:** The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like , time and budget constraints and technical feasibility of actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until customer expectations are met.

Agile Project Management

Why Different Agile Methods

- Some focus on the practices (XP, Agile Modeling)
- Some focus on managing the flow of work (Scrum, Kanban)
- Some support activities for requirements specification and development (FDD)
- Some seek to cover the full development life cycle (DSDM, RUP)

Agile anti-patterns or Agile Smells

1. Lack of overall product design
2. Adding stories to an iteration in progress
3. Lack of sponsor support
4. Insufficient training
5. Product owner role is not properly filled
6. Teams are not focused
7. Excessive preparation/planning
8. Problem-solving in the daily standup
9. Assigning tasks
10. Scrum master as a contributor
11. Lack of test automation
12. Allowing technical debt to build up
13. Attempting to take on too much in an iteration
14. Fixed time, resources, scope, and quality

vedavit

Large-scale, offshore and distributed

1. Scaled Agile Framework (SAFe), Dean Leffingwell *inter alia*
2. Disciplined agile delivery (DAD), Scott Ambler *inter alia*
3. Large-scale scrum (LeSS), Craig Larman and Bas Vodde
4. Nexus (scaled professional Scrum), Ken Schwaber
5. Scrum at Scale, Jeff Sutherland, Alex Brown
6. Enterprise Scrum, Mike Beedle
7. Setchu (Scrum-based lightweight framework), Michael Ebbage
8. Xscale
9. Agile path
10. Holistic Software Development

Requirement Prioritization

- Business value, risk, due date, dependencies, or any combination thereof

How to Avoid Fail in Agile?

- Scrum master should have authority to resolve all impediments else complex dependencies would kill the project.
- Periodically spend time on refactoring else there would be high risk of sustainability, maintainability and extensibility of the product.
- Strike a balance using level of plan. Must have an overall plan.
- Do not negotiate on quality and time. Do not force team to include more requirements else agile team will collapse quickly.
- Product's success depends upon authority, knowledge and engagement of product owner. Chose a right person else team can be driven in the wrong direction.
- Balance the documentation otherwise transfer of technology to new team members would be challenging.

Topics

1. Modern Project Management Challenges
2. How to address modern day PM challenges?
3. Challenges in Starting Agile Project Management
4. What is Agility?
5. Which Project Management Methodology is best?
6. Type of Systems
7. Complex Adaptive Systems (CAS)
8. Defined vs Empirical Processes
9. Values in Agile Manifesto
10. Agile Principles
11. Agile Methodologies
12. When Does Agile Methodologies Works Best?
13. Agile Project Lifecycle
14. Value Delivery & Project Life Cycle
15. Agile practices vs PMBoK Processes

Modern Project Management Challenges

- Uncounted uncertainties
- Very tough to negotiate with stakeholders a change in baselined plan
- Product value realization at the end of project lifecycle
- Huge difference between expectations of end user, customer and sponsor
- Typically development team is isolated from business scenario and it becomes very difficult to implement change request
- Technology and project environment changes during project execution
- Customer does not want to hear about new timelines even after requirement changes. Because project is baselined!
- Customer end up paying more because critical dates missed, non-usable product features, less-value product
- Not enough decentralized power stations to make decisions during project execution
- Execution team and project management teams are different. Execution team does not have power and execution does not happen as per initial plan
- Work product delivered at the end of every phase is not usable work product. Typical it is some paper prototype, document or some other thing.

How to address modern day PM challenges?

- Involve end user
- Engage relevant stakeholders
- Produce in increment
- Deliver high value feature first
- Take frequent feedback and allow customer to change original requirements
- Allow customer to prioritize
- Get commitment from team for valuable product not of activities
- Employ the power of level of planning
- Involve team in risk identification and responding to risk
- Transparency in project management
- Continuous improvement
- Learn quickly

Challenges in Starting Agile Project Management

- Team
  - Getting experienced team member
  - Getting 100% committed team
  - Getting collocated team in early agile adoption
  - Team's mindset shifting from action to delivery
  - A cross disciplined team with generic skills
- Environment
  - Trust building
  - Open communication with customer
  - Keeping politics out
- Infrastructure & Support
  - Active risk management framework
  - Robust, flexible and adaptable configuration and data management systems
  - You need a variety of **communication**, **collaboration**, **management** and **development tools**. Therefore a culture is required to support and facilitate this endeavors.
  - Management need a framework from where they get to know what is happening in the project.
  - Upfront training
  - Supporting in early period when velocity is less

What is Agility?

- Agility is about quickly responding to changes
- Learning quickly from mistakes and incorporate lessons learned
- Being proactive
- It helps in all aspects of success- Personal, Technical and Organizational

What is Agility?

- How agility is possible with complex, bureaucratic, gigantic processes.
- Traditional processes are reactive in nature, they learn very late, respond very slow

What is Agility?

- This refinery along with township in Jamnagar, Gujarat, India was completed in 1999 in 36 months.
- There was the time when over 100,000 people working on the project on any particular day and sponsor was paying rental for a crane Rs 10 million every day.

Complex Adaptive Systems

- Think about how newly born child is going to learn about talking, walking, learning, eating etc.
- What process you use to climb mountain?

- In the last couple of decades scientists and managers have articulated a profound shift in their view about how organisms and organizations evolve, respond to change, and manage their growth.
- Complex Adaptive Systems theory is one of the root threads of agile development. The concepts about how biological systems evolve and adapt have relevance, if only metaphorically, to organizations and how they evolve and adapt.
- Creativity and innovation are the emergent results of well-functioning agile teams.
- In complex system things get done because people adapt not because they blindly follow
- Former Visa International CEO **Dee Hock** coined the word **"chaordic"** to describe both the world where Order exists in Chaos.

Defined vs Empirical Processes

Defined Processes

- Assumes that every piece of work is completely understood
- Input is well-defined
- A set of well-defined input produces same output every time within known variance limit
- Has tightly coupled steps
- No checkpoint and feedback steps

Defined vs Empirical Processes

Empirical Processes

- Relies on frequent inspections and adaption
- Applies to those process which are loosely defined because of their complexity
- Understand that output of a process can be unpredictable and unrepeatable

Value in Agile Manifesto

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to **value**"

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

www.agilemanifesto.org

Agile Principles

1. Satisfy customer by continuous delivery of valuable product
2. Welcome Change even at late stage
3. Deliver working software Frequently
4. Business and Developer work together
5. Build around Motivated Individuals- Give team opportunity, trust them
6. Face-to-face communication
7. Working software is the primary measure of progress.
8. Sustainable Development
9. Continuous Technical Excellence
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. Self Organizing Team
12. Team reflection at regular intervals

www.agilemanifesto.org

Agile Core Practices

1. Active Stakeholder Participation
2. Apply the Right Artifacts
3. Collective Ownership
4. Create Several Models in Parallel
5. Create Simple Content
6. Depict Models Simply
7. Display Models Publicly
8. Iterate to Another Artifact
9. Model in Small Increments
10. Model with Others
11. Prove it with Code
12. Single Source Information
13. User the Simplest Tools

Agile Characteristics

- Adaptability not Predictability
- Accepting that outcomes are not predictable and process are not repeatable
- Values and Principles of Collaboration
- The conventions which we agree we define those
- Processes are in manuals; practices are in field.

Agile Methodologies

1. Scrum
2. Extreme Programming (XP)
3. Lean Software Development (LSD)
4. Feature Driven Development (FDD)
5. Dynamic System Development Methods (DSDM)
6. Kanban
7. Scrum-Ban
8. Crystal Methods

Agile Methodologies- Scrum

- Scrum is widely accepted agile project management methodology among dozen of methodologies.
- Scrum is **lightweight management framework** with broad applicability for managing iterative and incremental projects of all type of projects
- Typical iteration (also called **"Sprint")** length varies between 2-4 weeks.

Scrum Values

1. Commitment
2. Focus
3. Openness
4. Respect
5. Courage

Agile Methodologies- Scrum

Scrum Practices

- Backlog Grooming
- Sprint Planning
- Daily Standup
- Sprint Review
- Sprint Retrospective

Agile Methodologies-XP

Originally devised by Kent Beck as an agile method focused on engineering practices. Typical iteration length varies between 1-3 weeks.

XP Values

1. Communication
2. Simplicity
3. Feedback
4. Respect
5. Courage

Agile Methodologies- XP

XP Practices

Fine scale feedback

1. Test Driven Development via Programmer Tests and Customer Tests (Unit Tests & Acceptance Tests)
2. Planning Game
3. Whole Team (including on-site customer)
4. Pair Programming (2 people sitting on one work station, one writing test case on notes and another writing code)

Continuous process rather than batch

5. Continuous Integration
6. Design Improvement / Refactoring
7. Small Releases

Agile Methodologies- XP

XP Practices

Shared understanding

8. Simple Design (Do Simple Things, You Aren't Gonna Need It **(YAGNI)**, Once And Only Once **(DRY)**, Simplify Vigorously)
9. System Metaphor
10. Collective Code Ownership
11. Coding Standard or Coding Conventions

Programmer welfare

12. Sustainable Pace

Agile Methodologies- LSD

Much of its principles and practices are from Lean Enterprise Movement and companies like Toyota

Lean Software Development (LSD) 7-Principles

1. Eliminate Waste (Just in Time –JIT, Kanban)
2. Amplify Learning
3. Decide as late as possible (Make a decision when not making it means leaving some important option)
4. Deliver as fast as possible
5. Empower the team
6. Build Integrity In (overall experience of system quality, product quality should be dependable)
7. See the whole (see big picture)

Agile Methodologies- FDD

- FDD is a model-driven, short-iteration method
- First step is to establish an overall model shape
- Followed by 2 week “design by feature, build by feature” iterations.

Agile Methodologies- FDD

Eight practices of FDD

1. Domain Object Modeling
2. Develop by Feature
3. Component/class ownership
4. Feature Teams
5. Inspections
6. Configuration Management
7. Regular Builds
8. Visibility of Progress and Results

Agile Methodologies- DSDM

- Earlier in 1994 it was known as RAD method
- Previous version of DSDM is 4.2, that was released in 2003. Latest version of DSDM released in 2007 is also called **DSDM Atern**
- DSDM Atern used MoSCoW principle to prioritize requirements
- It is compatible with ISO9000 and PRINCE2
- “Fitness for business purpose” is primary criteria for delivery and acceptance of system
- It focuses that 80% of the system can be developed in 20% of the time

Agile Methodologies- DSDM

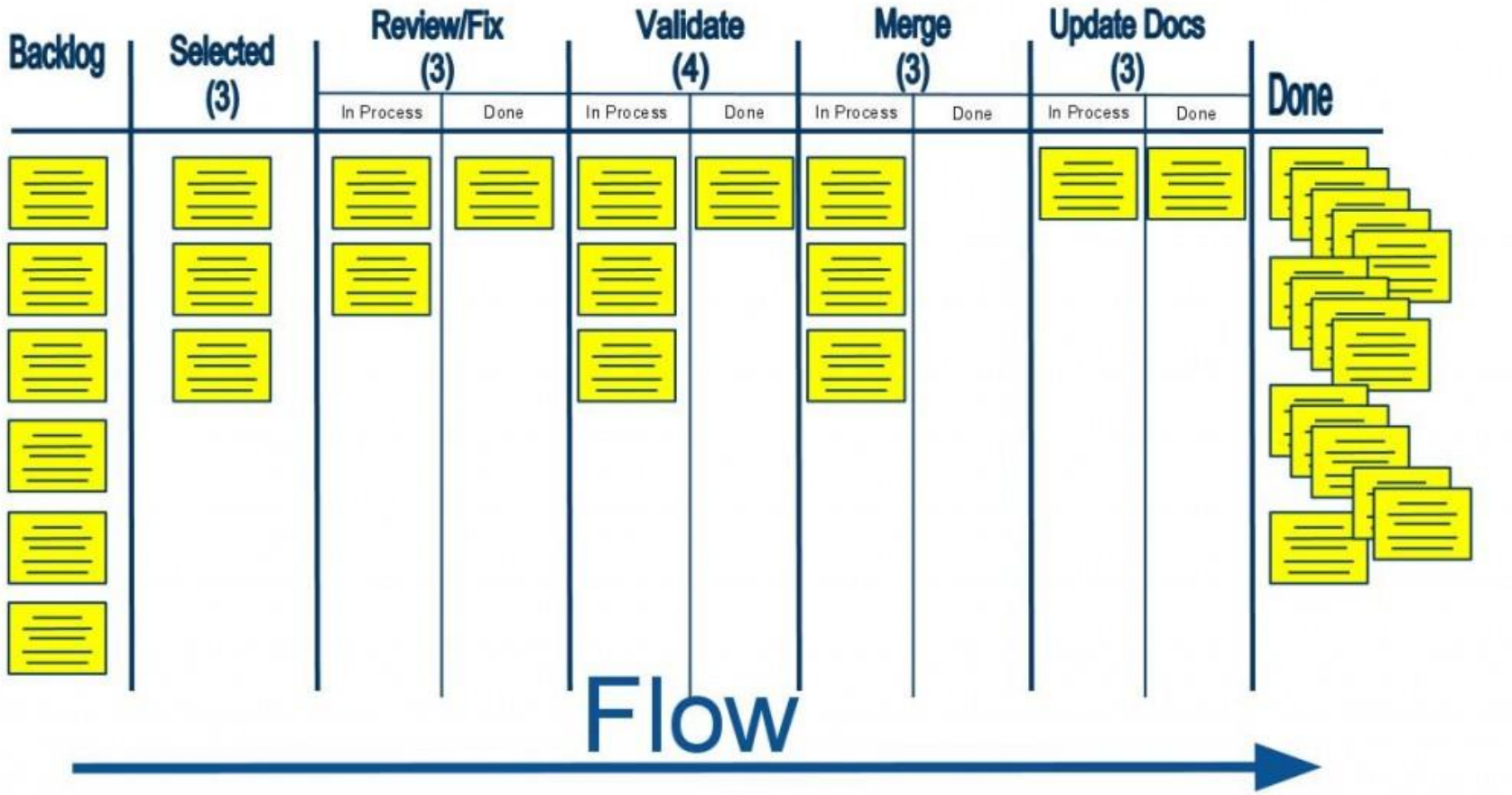
8 Principles of DSDM

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

Agile Methodologies- Kanban

- In Japanese Kanban means Sign-Board
- Kanban was developed by Taiichi Ohno, at Toyota, to find a system to improve and maintain a high level of production.
- Kanban is scheduling system for Lean and JIT production.
- Kanban is one method through which JIT is achieved.
- Kanban is pull based planning and execution method. Work is not planned and pushed into work queue of team. Team signals when they are ready to take more work and pulls into their queue. Uses cards to signal the need of items
- Under Kanban or Lean approach queues or inventory of work in any state is seen as waste.
- **WIP limits**: helps team in optimal flow of work items, minimizing any associated waste
- Allow team to achieve process optimization while respecting and maintaining sustainable pace.

Kanban Board

- 

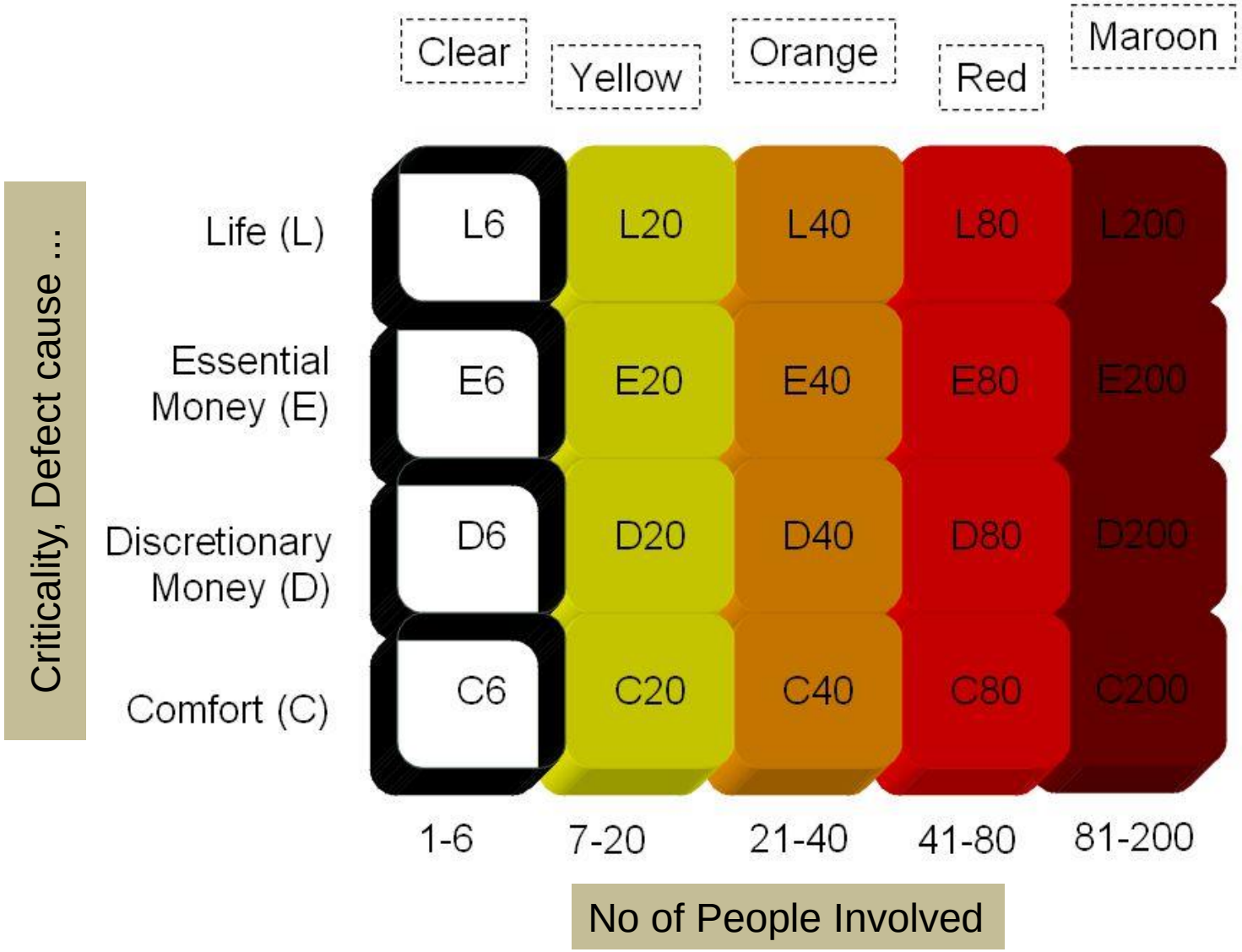
Agile Methodologies- Scrum-ban

- Scrum-ban is a software production model based on Scrum and Kanban
- Scrum-ban is especially suited for maintenance projects or (system) projects with frequent and unexpected user stories or programming errors. Thus time-limited sprints of the Scrum model are of no appreciable use.
- Scrum's daily meetings and other practices are highly useful
- Visualization of the work stages(1) and limitations for simultaneous unfinished user stories(2) and defects(3) are part of the Kanban model.
- The team's workflow is directed in a way that allows for minimum completion time for each user story or programming error, and on the other hand ensures each team member is constantly engaged.

Agile Methodologies- Crystal

- Crystal is one of the most light-weighted software development methodology
- Crystal is a family of methodology. Family members are Crystal Clear, Crystal Yellow, Crystal Orange etc.
- Which type of crystal methodology should be used for a particular project depends upon **criticality of the project** and **number of people involved**
- This methodology relies on the fact the processes can be tailored based on unique characteristics of the project

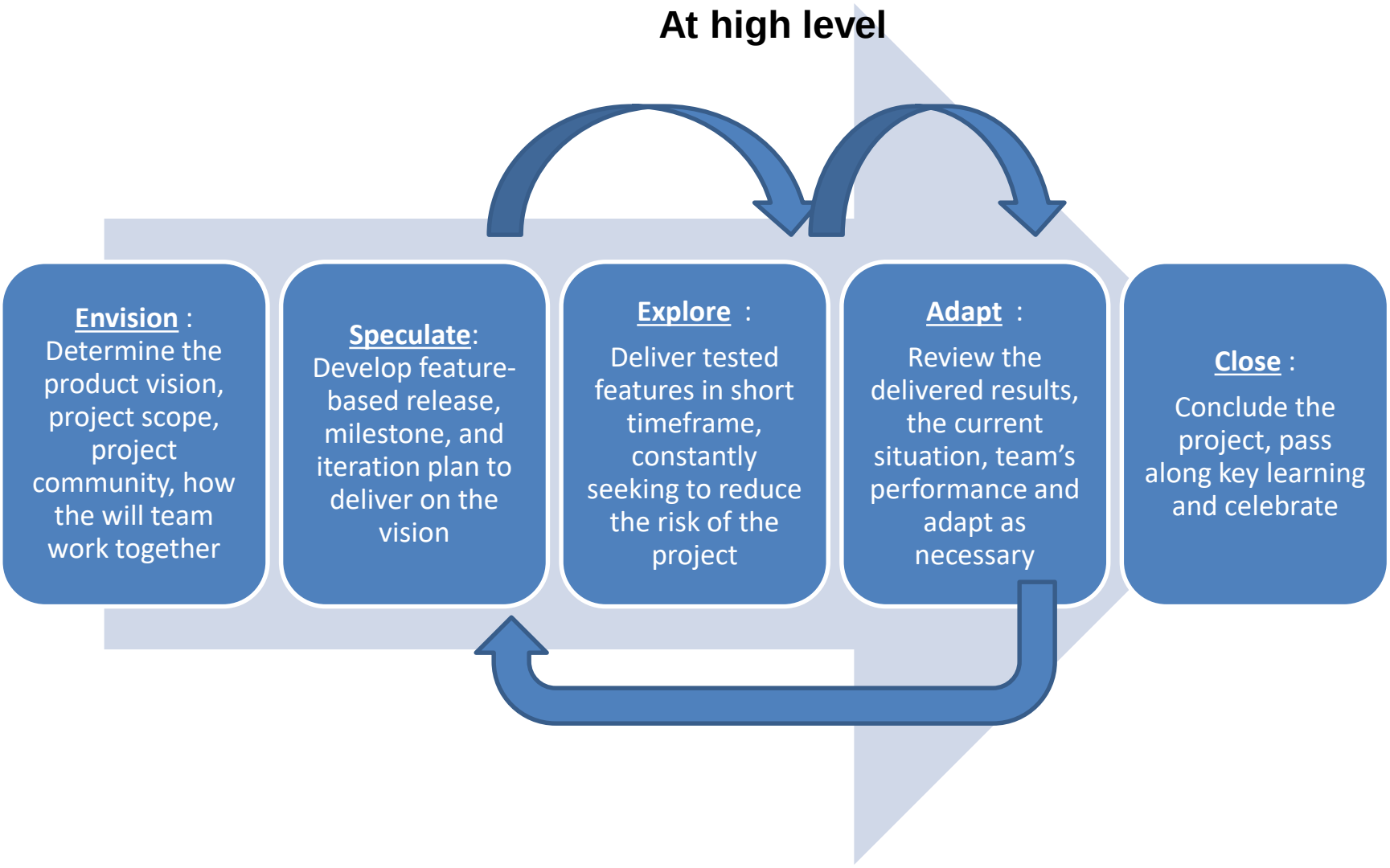
Agile Methodologies- Crystal



When does Agile Methodologies Works Best?

- When problems you are solving have following characteristics
  - Going to **change** while solving
  - **Speed** of development cannot be determined
  - **Turbulence** in environment
  - Customer **doesn't know** how exactly it will look like

Agile Project Phases

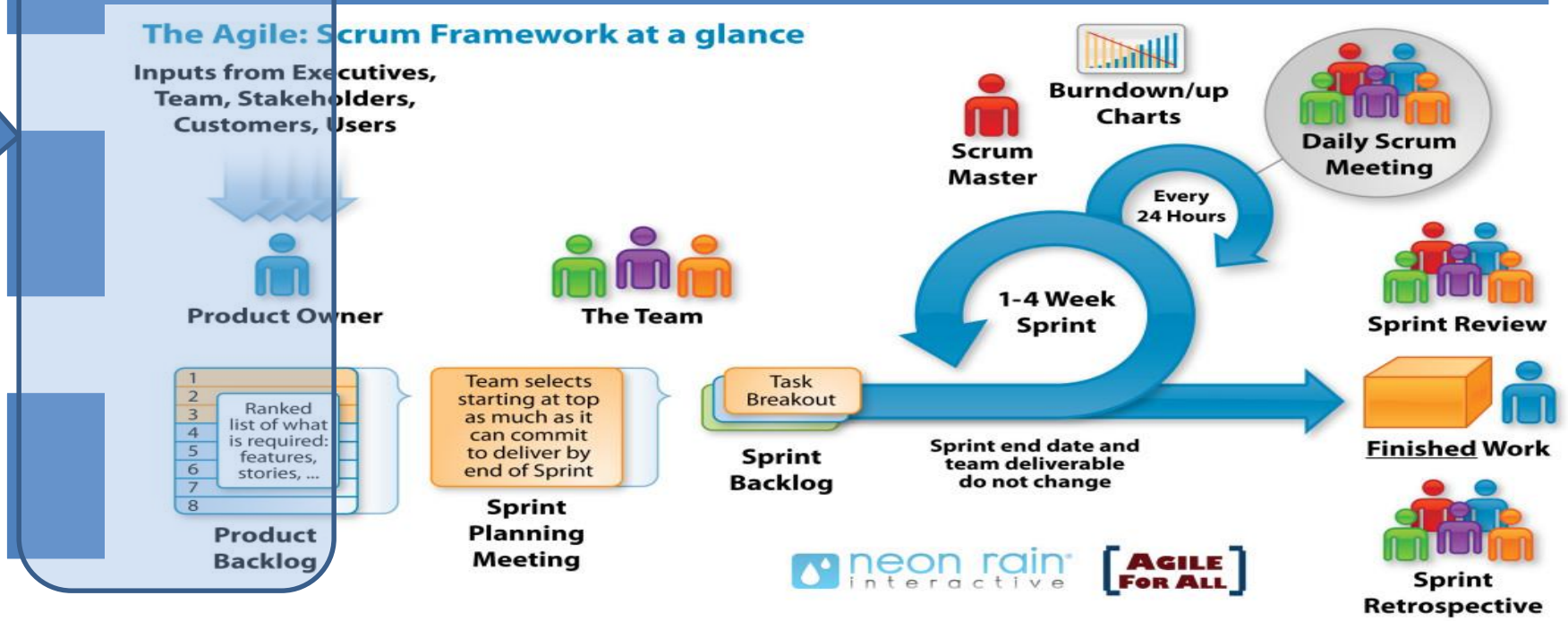


Agile Project Life-cycle

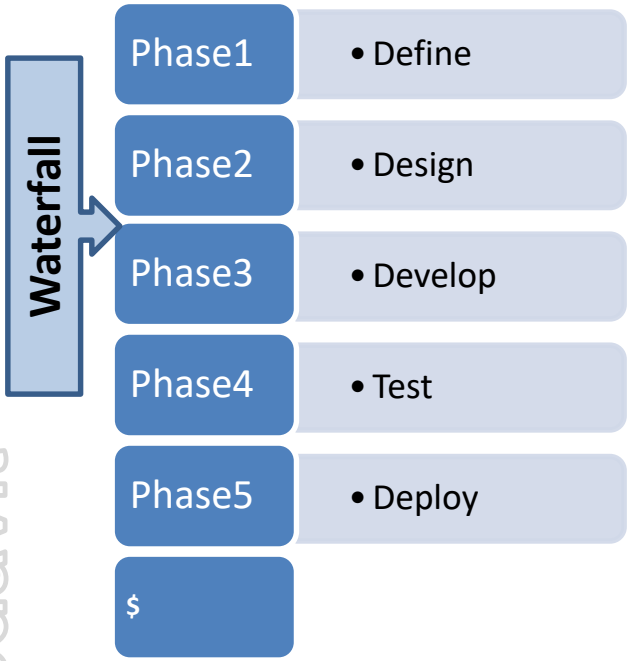
Inside Story

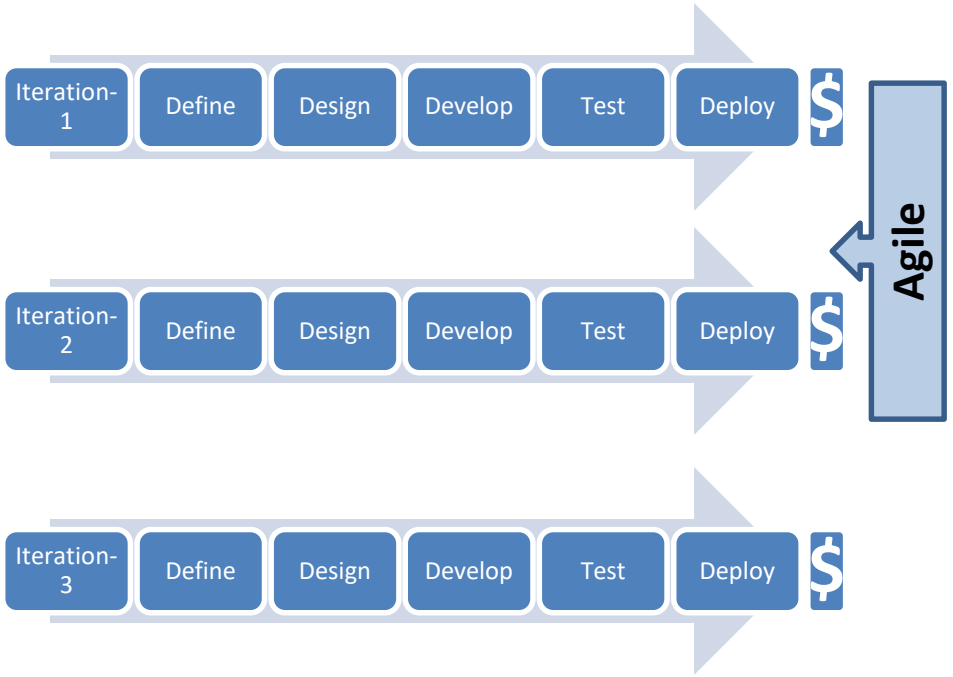
Product Vision

Product Backlog



Value Delivery & Project Life Cycle





Iterative

Iteration	Iteration1	Iteration2	Iteration3
Define	Define	Define	Define
Design	Design	Design	Design
Develop	Develop	Develop	Develop
Test	Test	Test	Test
Deploy	Deploy	Deploy	Deploy

Introduction:
Agile Mindset

Topics

1. Iteration
2. Incremental Delivery
3. Travel Light
4. Agile Product Building
5. Agile Documentation
6. Agile Architecture
7. Agile Testing
8. Last Responsible Moment
9. Refactoring
10. Technical Debt
11. Mindfulness
12. Energized Team
13. Poka Yoka
14. Agile Roles
15. Agile Health Checkup

Iteration / Sprint

- Agile project management relies on planning, developing and delivering product features in iterations
- Iteration or sprint are time-boxed and duration can not changed from one iteration to another
- Iteration is fixed time period in which team need to deliver some valuable product to customer
- Prescription for duration length varies from one method to another in agile. In XP iteration length varies 1-3 week (typically one week), scrum and other methods it varies between 2-4 weeks (typically 4 weeks)
- Iteration is also known as Sprint in Scrum Methodology

Incremental Delivery

Delivering the complete product of a project in iterations. It helps in

- Learning from previous iteration
- Delivering high business value earlier
- Adapt to change
- Delivering more business value
- Removing the waste by not doing those things which are not needed

Travel Light

- You need far less than what you think. Therefore carry on only those things which are most important and urgent.
- YAGNI (You aren’t gonna need it)
- TAGRI (They aren't gonna Read It)

This is different type of mountaineering. You may need to come back, you may need to change your path, trust that you will get stuff you need on the way etc factor exists here.

Agile Product Building

- Build complete product, all the time
- One button should produce needed documentation, build the product executables, create installation materials, produce test results and tested components
- Build should also work from command line
- Everyone in team should use the same build process

Agile Documentation

- Maximize stakeholder investment. Produce document only when
  - It is needed by a stakeholder
  - Needed to define contract model
  - You need to think something in many iterations involving multiple groups
  - It is needed for external communication
- Document only those things which are least likely to change
- First identify the specific customer of the document
- The document facilitates in estimating
- Sufficiently index, details, accurate and consistent

Agile Documentation

Strategies for reducing documentation CRUFT

- C- How correct is document?
- R- Will document be read?
- U- Will document be understood?
- F- Will document be followed?
- T- Will document be trusted?

Agile Architecture

- Do the simplest thing possible which makes future changes/additions easier
- No up-front high-level system architecture
- No up-front high-level component architecture
- No up-front high-level data model
- Determine the details of technology when building functionality

Agile Testing

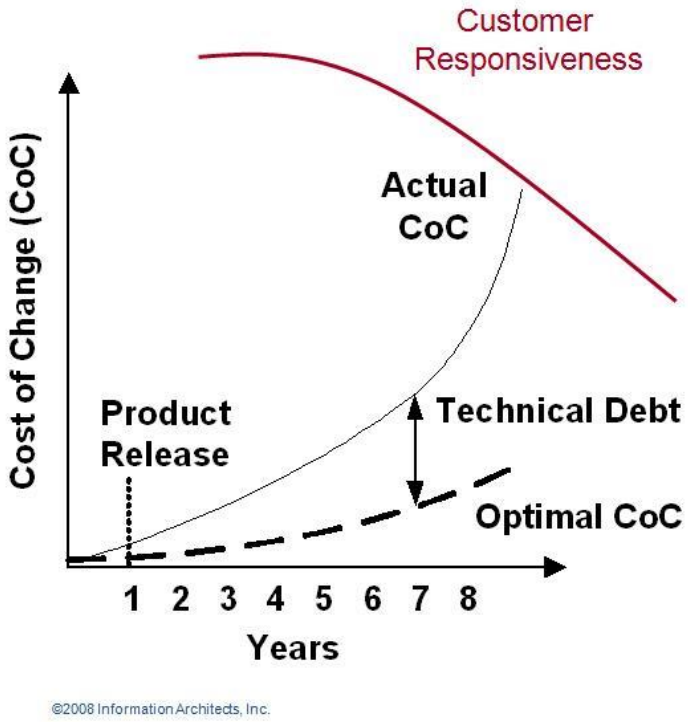
- All code must have test cases, ideally they should be created earlier
- Unit tests should be executed during automated build
- A build should be performed many times a time, ideally whenever anything is checked-in or committed to configuration server
- If anybody's code causing crash he should be informed immediately and that person should fix that problem first
- All unit test must pass before code can be released

Last Responsible Moments

- Delay the decisions till the moment you have option to exercise because after this point one important options may be eliminated
- It helps you in delivering more value in less time by doing only those things which are of high priority
- It reduces inventory carrying cost
- Getting more information and making more informed decision at last moment

Refactoring

- Agile programmer writes simple and bare minimum code they do not complicate the code
- Down the line structure the code without changing its behavior. It helps in improving the quality (maint



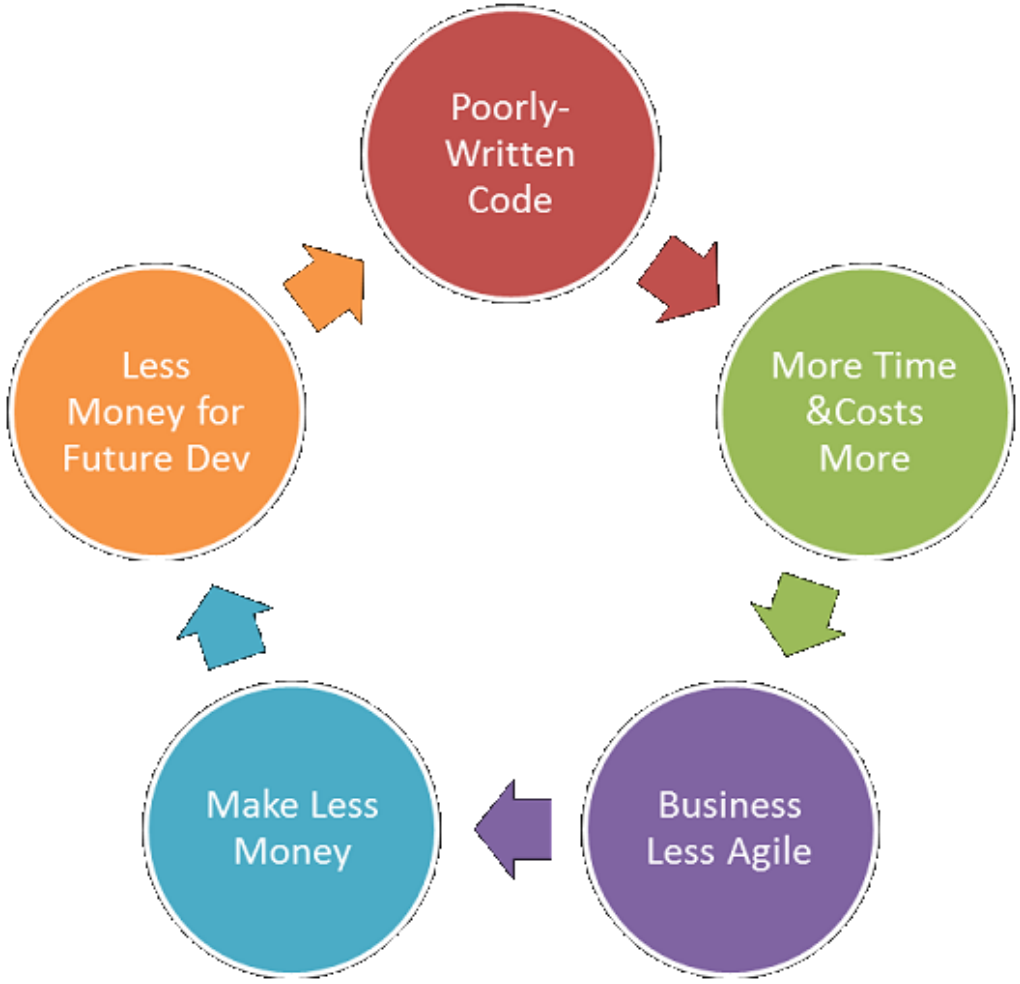
- Once on far right of curve, all choices are hard
- If nothing is done, it just gets worse
- In applications with high technical debt, estimating is nearly impossible
- Only 3 strategies
  - Do nothing, it gets worse
  - Replace, high cost/risk
  - Incremental refactoring, commitment to invest

Technical Debt

- While writing code agile programmer do not pay attention to structure, duplication etc but to functionality and make sure that code is passing all unit test cases
- In this process if they do not clean the code by structuring (refactoring) it will become unreadable and un-maintainable and over a period of time this cause increases response time to fix the problems, adding new feature and deteriorate the quality.
- The concept of unclean code is called technical debt. Technical debt keep increasing over the period of time therefore Agile team need to pay this debt back by putting efforts in refactoring.

Technical Debt

Vicious Cycle of Technical Debt



Mindfulness

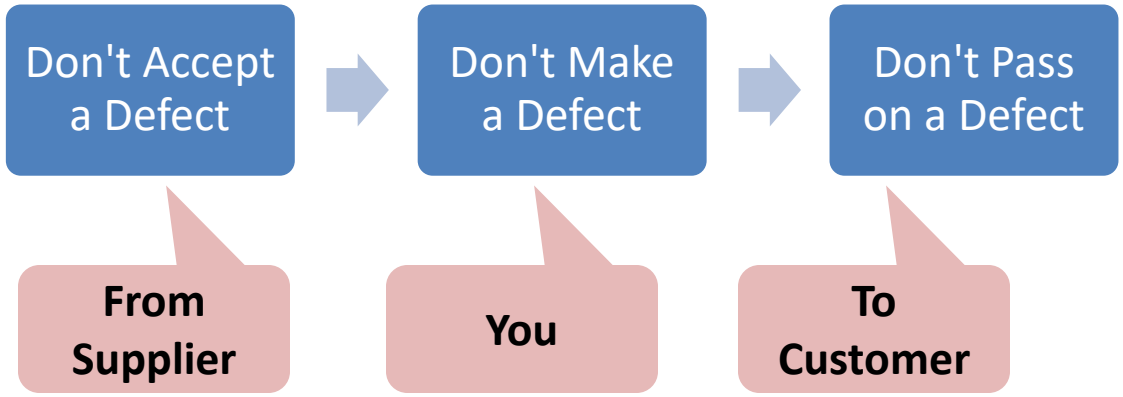
- Agile team need to be conscious all the time, they need to keep observing patterns and proactively resolving the issues at root.
- **For example** if they know that now it is taking more time to add even simple thing in existing code then it means time has come and they need to pay technical debt by refactoring

Energized Work

- Love your work: When you feel that I enjoy programming. I enjoy solving problems, writing good code, watching tests pass, and especially removing code while refactoring. I love to program in my spare time and sometimes even think about work in the shower. You love your work
- Do not work overtime, enjoy balanced life, remain healthy, energetic and excited about work
- When at work completely cut off from interruption like phone, email, IM etc. Pay 100% attention to work at hand
- If you are making more mistake than progress then that is the time for break.

Poka Yoka

Mistake (Error) Proofing



Ideally, design the product so that it can't be assembled incorrectly

Error is deviation from process. Defect is in product who is deviating from specification or don’t meet customer expectations. Defect are introduced by errors

Agile Roles

- Pigs vs Chickens
  - Pigs are those roles which are committed
  - Chickens are those roles which are involved
- Product Owner
  - Grooms product backlog, interface between product user and team
  - Prioritizes requirements based on value
  - Has authority to change requirement, reject product
  - Justify the importance of product and its features
- Scrum Master
  - Process owner, ensures that stakeholders follow the processes
  - Removes impediments
  - Works as a servant leader
- The Team
  - Includes architect, developer, tester, UI Desiger
  - Cross functional multi skilled team
  - Responsible for quality of the product

Agile Roles

- **Agile Tester**
  - Try to identify if a user does not do what he is suppose to do then what happens to the software? Report it.
  - Helps customer, product owner in writing acceptance test cases
- **Coach**: Identify and Improve work habits
- **Interaction Designer**: Learns and decides how users will interact with system.
- **SME** (Domain Expert): Understands the business domain, regulatory requirements and defines business rule for the product

Agile Health Checkup

To know the agile maturity in your project you can perform following Agile Health Checkup. Discuss in group and ask team to rate these parameters on the scale of 1-5

1. **Frequent Delivery**
2. **Reflective Improvement**
3. **Close Communication** (does it take less than 1 min to get you question answered by a person who know the answer?)
4. **Focus** (everybody understand the goal and desired outcome of the delivered software?)
5. **Personal Safety** (can you give bad news to your boss?)
6. **Easy Access to Outside Experts**
7. **Strong Technical Environment** (SVN used? Test Automation?)
8. **Sunny Day Visibility** (Does everyone on the team understand the rate of progress being made on the product?)
9. **Regular Cadence or Rhythm** (Is heartbeat of the system on?)

Communication

Team Space

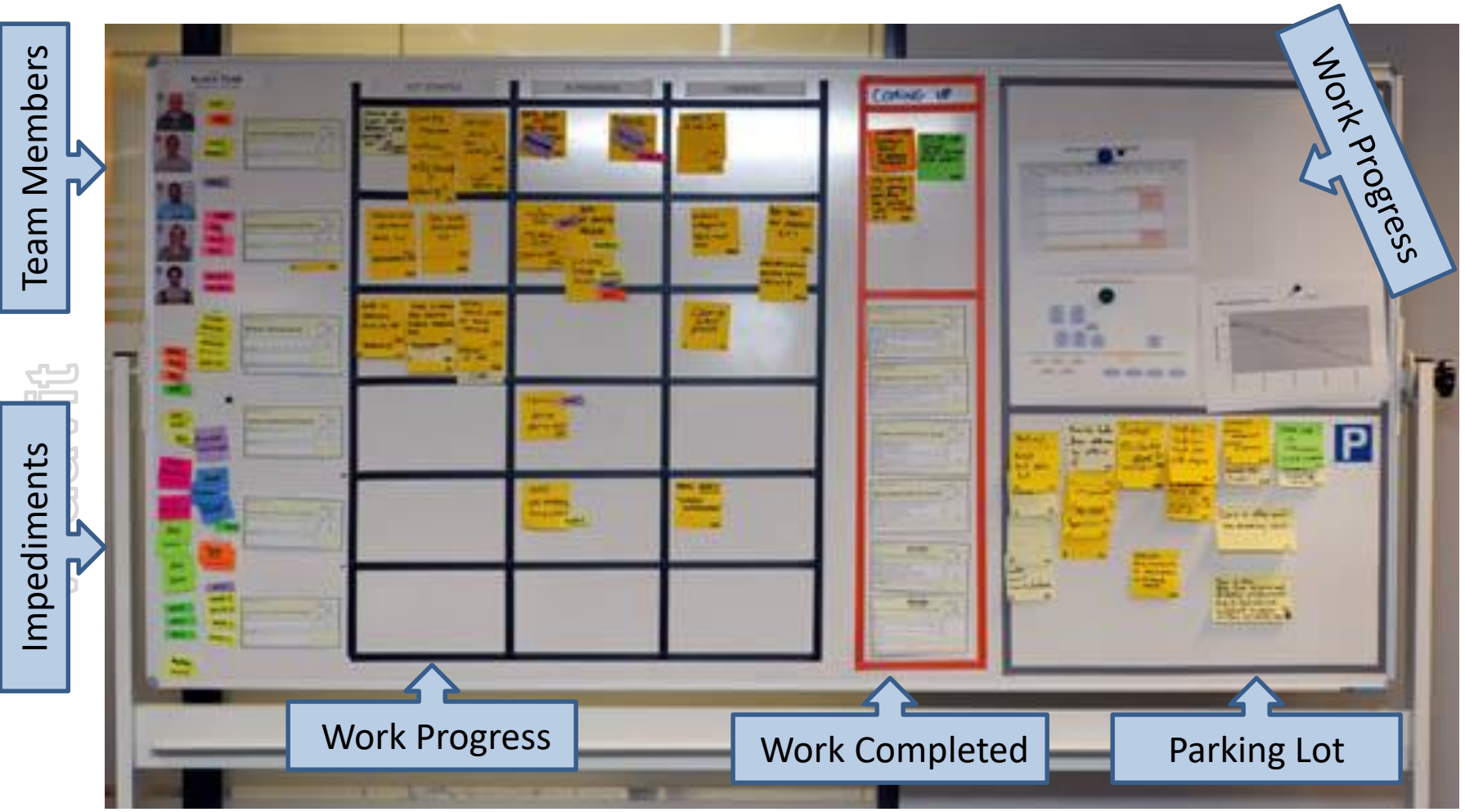
Team space is working area of agile project team. Only those people who are contributing to project directly should be allowed to sit, stay and talk in that area

This area should be

- **Informative**
- **Spacious**
- **Comfortable to work**
- **Shielded from external noise**
- **Easy to collaborate**

Having expert in the same room where rest of the team is sitting is called "Expert in earshot"

Information Radiator



Osmotic Communication

- Agile project relies on collocation, minimum documentation, least reporting, maximum constructive engagement. This can be achieved if frequently sought important information is available in published form in team space
- Whoever need the information can go and get the information without wasting time in requesting, making, sending, receiving the information
- Benefits
  - Least cost, effort and time waste in communication
  - Updated information is always available without making new reports
  - People can get whatever particular information they are looking for
  - Happens at the same time
  - Feedback loop is quick
  - Those people who are left in regular reporting also get benefitted
  - No junk, old, repetitive information, but fresh and useful.
- Dis-benefits
  - Some people get extra information which they do not need
  - It is left to individual's interpretation

Daily Stand-ups

- Daily stand-up is heart beat of agile project management
- Team meets daily (typically in working area, war room) at fixed time (time should not be changed) preferable first thing in morning
- This is not reporting session but information sharing among team members
- Only “Pigs” allowed to speak, “chicken” should listen (they are not allowed to interfere or ask while stand up meeting is in progress)
- It is 15-20 min meeting, conducted while everybody is standing (showing the sense of urgency). A person should not take more than 2 min to update this work status.
- Any one in the team can facilitate this meeting
- Project manager notes the impediments and start working on those immediately after the meeting is over.

Daily Stand-ups

3 Questions of standup meeting which every team member must address are

1. What did they do yesterday?
2. What are they planning to do today?
3. Any impediments on the way today?

Level of Listening

- Level 1: Internal Listening (interpret in own language)
- Level 2: Focused Listening (Responding without interpreting in own language)
- Level 3: Focused but understand global environment (team, organization, customer etc)

Communication Skills

- Active Listening
- Probing Ideas for Better Understanding
- Fact Finding
- Expectation Management
- Conflict Resolution
- Next Step Identification
- Making Commitment
- Contribute Effectively

Communication Barriers

- Physical Distance => Physical Proximity
- Working in Different Shifts => Temporal Proximity
- Human Language
- Technical Knowledge
- Detrimental Attitude
- Difference in Culture
- Difference in Project Environment
- Attitude, willingness => Amicability
- Communication Channels

Communication Methods

- Push Communication
- Pull Communication
- Interactive Communication

Planning, Monitoring & Adapting

Adaptive Planning

- Adaptive planning is about incorporating lessons learned. Adjusting your future pace, processes, resources etc based on previous cycle of delivery.
- Adaptive planning is required because what we want to develop may not be completely known or other factors like processes, tools, technologies, skills availability, business environment, market condition etc are not fully known or reliable and they all affect your planning.
- It is complete opposite of predictive planning where everything is known at the time of planning.
- In modern time it is extremely difficult to know about all the factors, which are affecting project success, in advance
- Adaptive planning and empirical processes are the truth of 21st century! Days of definitive processes and predictive planning are not many.

User Story

- A user story concept is kernel of Agile Project Management
- A user story is work which a user want system to accomplish because it meets some of his objectives.
- A user story is not functional specification document. It is a promise of product owner to the team that he will explain the requirements in details when the team is working on this
- User story template
  - "As a user I want to accomplish something so that business value"

"There have been great societies that did not use the wheel, but there have been no societies that did not tell stories." Unknown

Types of User Stories

- Business user story
  - "As a class teacher I want to mark attendance of student so that we can issue them certificate"
- Bug user story
  - "An error message is displayed whenever I try to save file in pdf format"
- Technical user / Technical Spikes story
  - "Research a search component in .NET3.5 which is fit for our application"
- Non-functional user story
  - "The Student Affairs Information System is up and running 99.9% during the registration time period defined in the Academic Calendar."
- Documentation user story
  - "Develop a user manual for teachers to use teacher module"

3Cs of User Story

- Card
- Conversation
- Confirmation

INVEST Model of User Story

INVEST model defines following characteristics of a user story

- Independent
- Negotiable
- Verifiable
- Estimatable
- Sized Appropriately
- Testable

Epic, Feature, Story, Task

- Epic is a collection of features. An epic is typically 1-3 months in duration
- Feature is collection of stories. A feature is typically 2-4 weeks in duration
- User-story is smallest unit of requirement created from features. A user-story is typically less than a week in duration
- Task are smallest unit of executable items which team members assign to themselves to complete a user story. A task is typically of 8 hours in duration

Agile Planning

- Agile project management does not rely on big bang planning rather it believes in level of planning.
- Three level of planning in agile are
  - Release Planning
  - Iteration Planning
  - Daily Planning

Agile Planning

Release Planning

- Creating a release plan is responsibility of product owner
- If some user story cannot be estimated due to technical complexity then technical spikes are created
- Typical length of a release varies between 3-6 months. For every month of work you can spend max one day for release planning. Thus 3 month release you can spend max 3 days.
- Release planning should not be done without knowing velocity
- It takes 3-4 uninterrupted iterations to benchmark velocity for a project team
- Release planning depends upon
  - Deadline from competitor
  - Supporting the contract
  - To meet predetermined schedule
  - Supporting financial deadline
  - When there is enough value
  - To test the product
- Two types of release plan
  - Scope-boxed release plan
  - Time-boxed release plan

Agile Planning

Release Planning

Product owner is ready with prioritized product backlog (prioritization is done based on business value)

Product owner defined release goals

During release planning product owner picks up only those user stories and features which helps him achieving release goals

All the selected user stories are pushed into release backlog

User stories in release backlog are size estimated using agile estimation techniques like story points

User stories in release backlog are again prioritized. It helps in creating number of iterations and iteration plan.

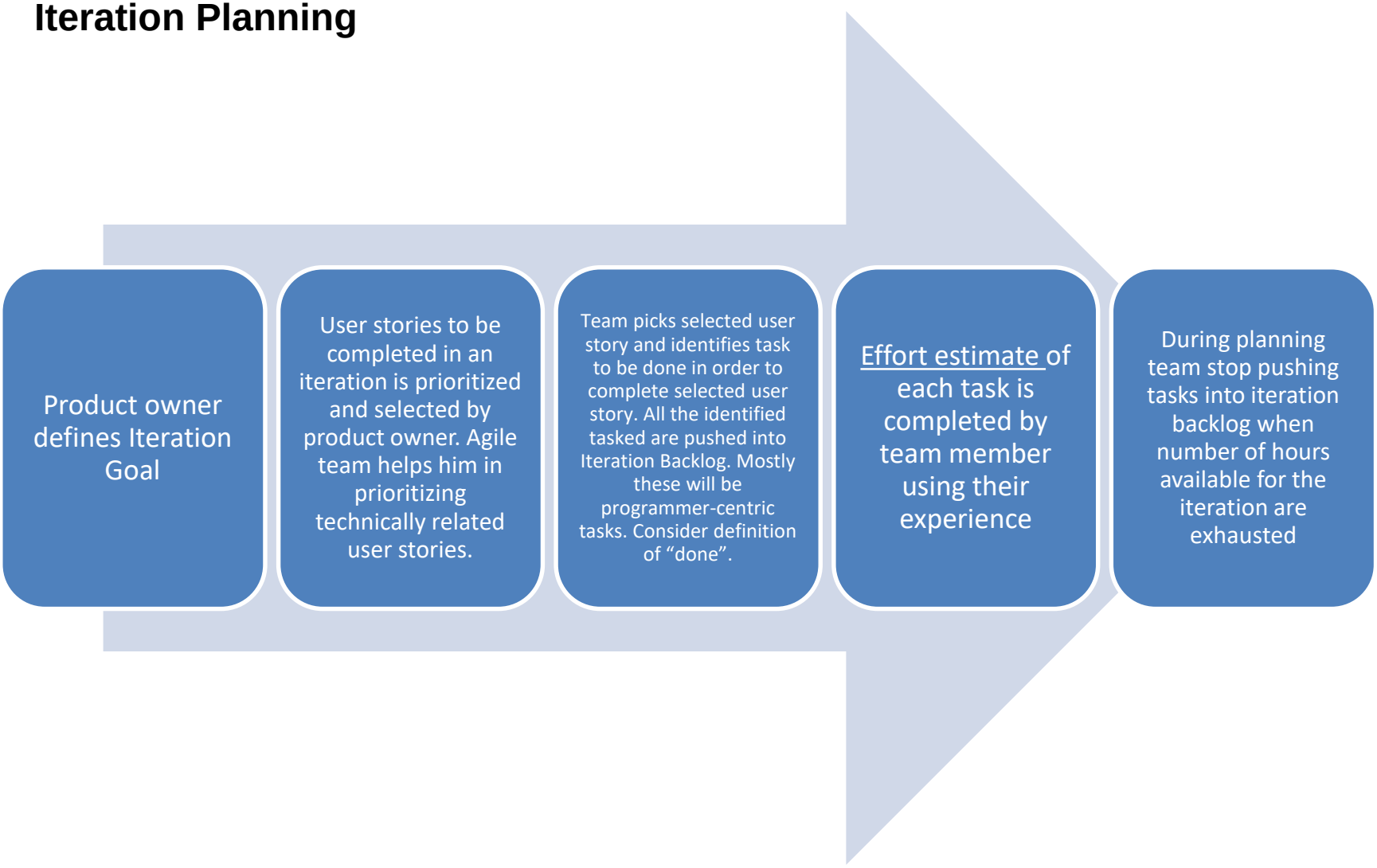
Agile Planning

Iteration/Sprint Planning

- In iteration planning team identifies the task to be performed for each user story.
- Task are pushed into iteration backlog
- Iteration planning is held after retrospective meetings of last iteration
- Typical iteration length varies from 1-4 weeks.
- Iteration planning duration (1 hour for every week of iteration)
- Iteration planning identifies iteration backlog items, assumptions, risks, actions, dependencies
- Some team start counting iteration from zero so the initial iteration is called **Sprint 0** or Iteration 0. In this iteration they take all technical spike which will help them in estimating size of complex user stories. Preparing initial architecture, solving infrastructure setup and configuration issues. Following Sprint 0 they start release planning
- Tasks in iteration backlog are not assigned to any team member. Agile teams are self-organized and they pickup the task on their base.
- Number of hours available in any iteration are calculated as Ideal engineering hours.

Agile Planning

Iteration Planning



Agile Planning

Daily Planning

- Immediately after or before daily standup meeting sessions and the team updates the kanban board.
- This reflects the progress and planning of the day based on
  - The work progress of previous day
  - Any internal or external dependency not met
  - Something critical comes up
  - Client want to drop some user story

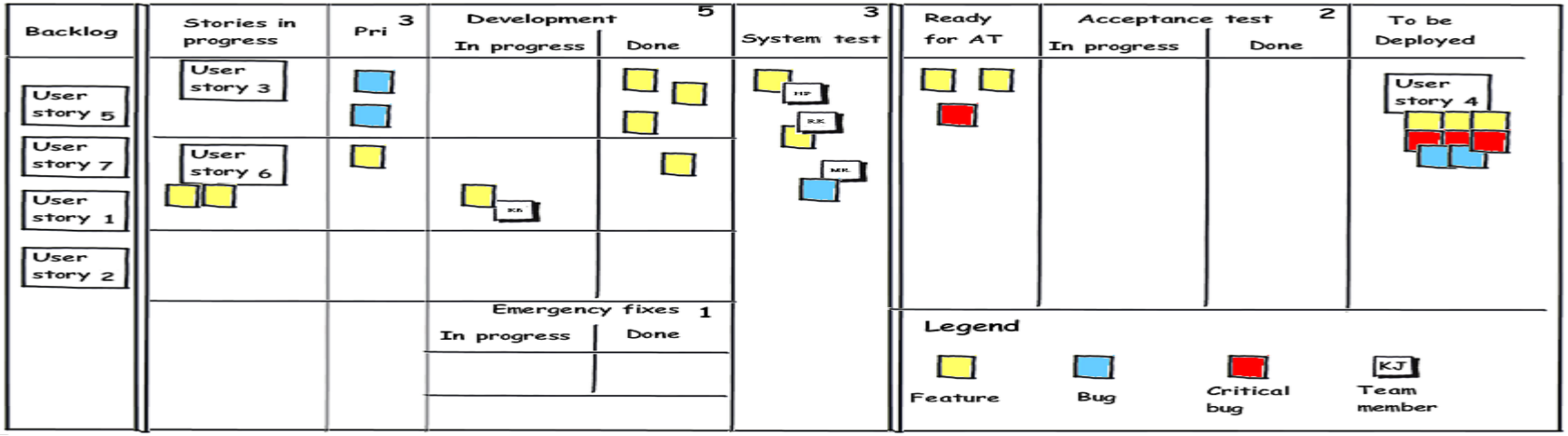
Agile Project Tracking

Kanban Board

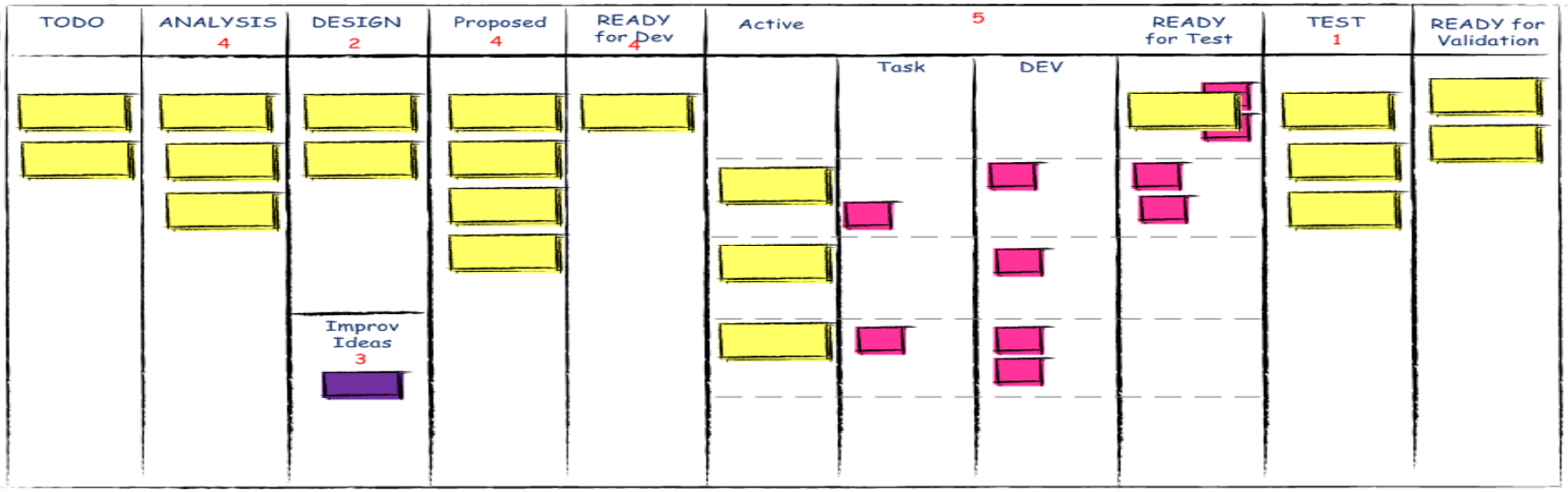
- After iteration planning is complete and before agile team starts working on task of the iteration, agile team need to pull the task from sprint backlog and put on kanban board
- Kanban board has several columns. You can customize these columns based on your project characteristics. Typical columns are “User Story”, “To be done”, “Doing”, “Done”, ”Accepted”
- You keep moving tasks from of any user story from left to right based on the progress. Finally when all the work is done all the cards will move under “Done” and user story will move user “Accepted” column

Kanban Board

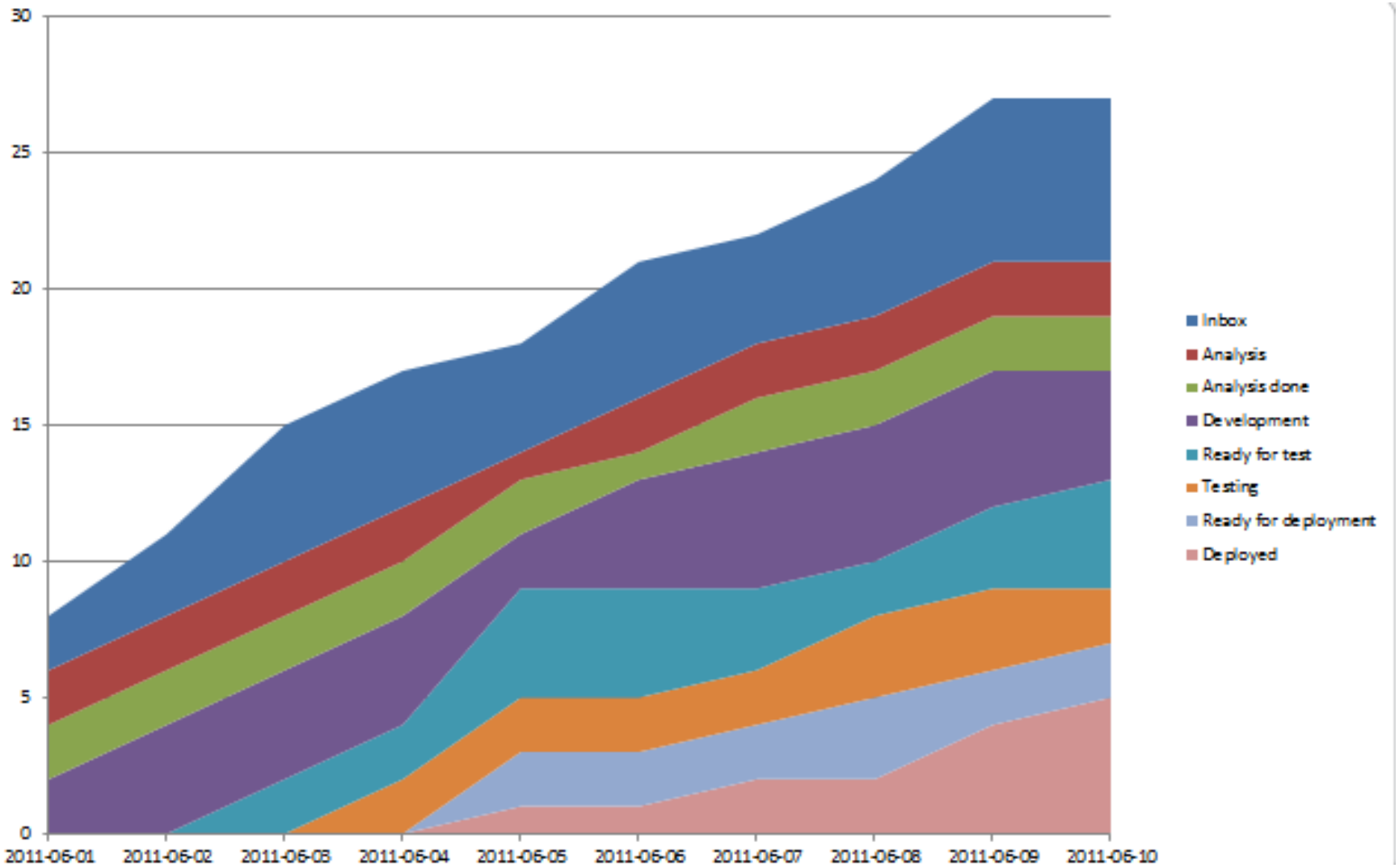
Story Kanban Board



Task Kanban Board



Cumulative Flow Diagram (CFD)

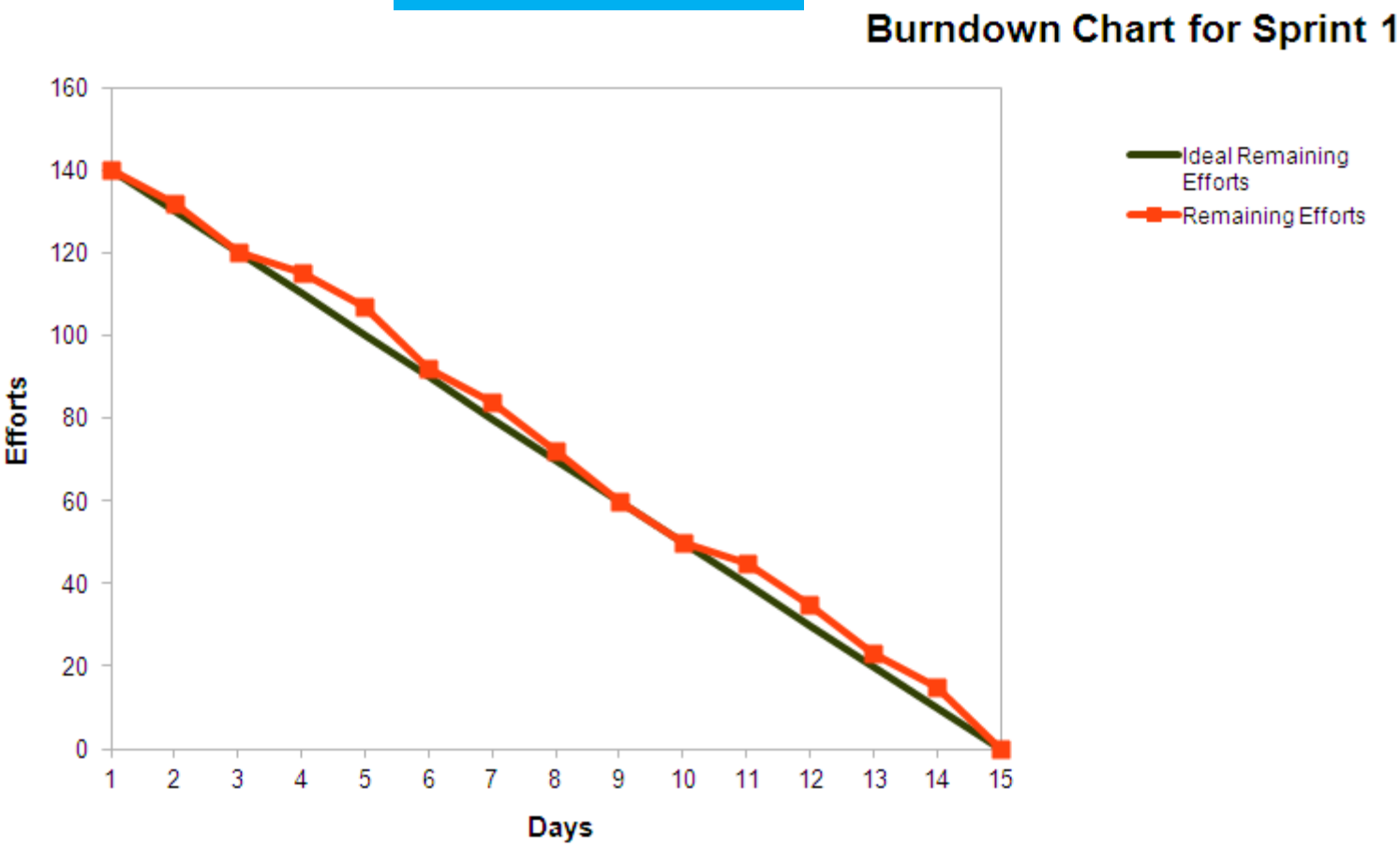


WIP Limit

- Tools & Techniques / Planning, Monitoring & Adapting
  - WIP limit is maximum number of work in progress items
  - In Kanban concept agile team should ensure that during execution at no point of time items in “in-progress” column should cross a defined WIP limit
  - “In-progress” work is treated as of no-value thus it is treated as waste. Therefore to maximize the value focus should be to convert WIP items into completed
  - Just-in-time (JIT) concept should be utilized to keep WIP limit smallest possible
  - Low WIP limit is also a challenge as it creates wastage of machine and workforce time

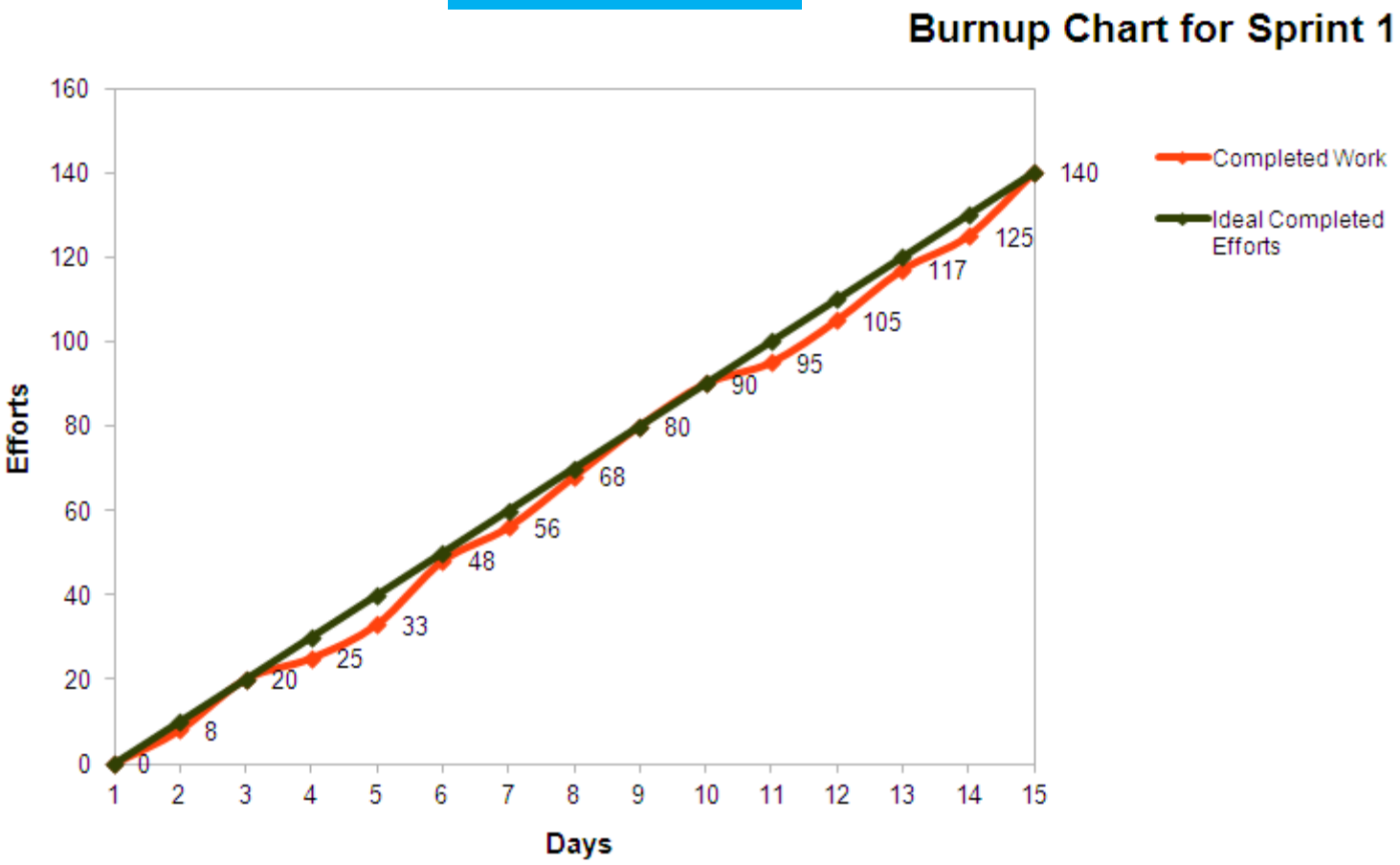
Burn Chart

Burn Down Chart



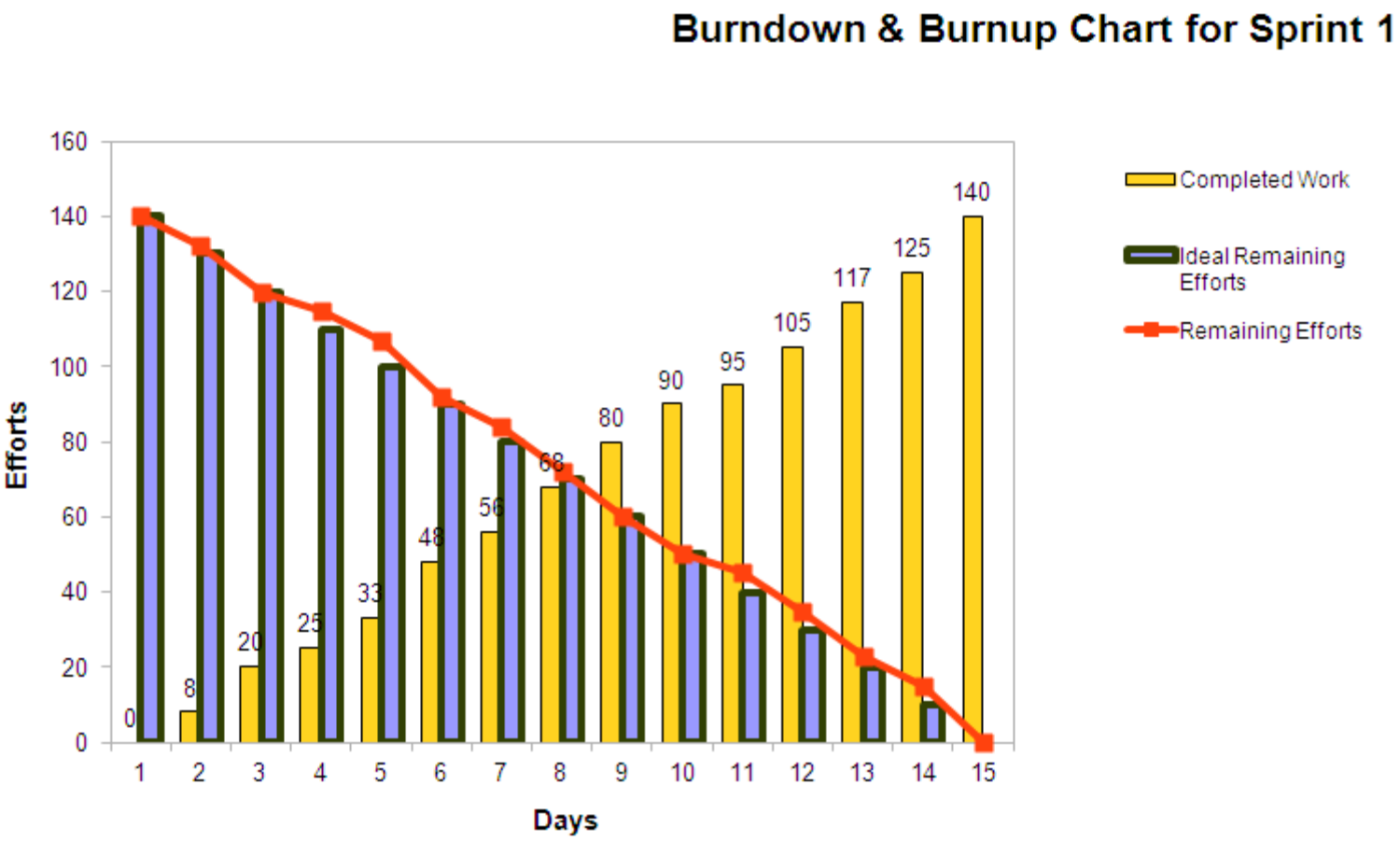
Burn Chart

Burn Up Chart



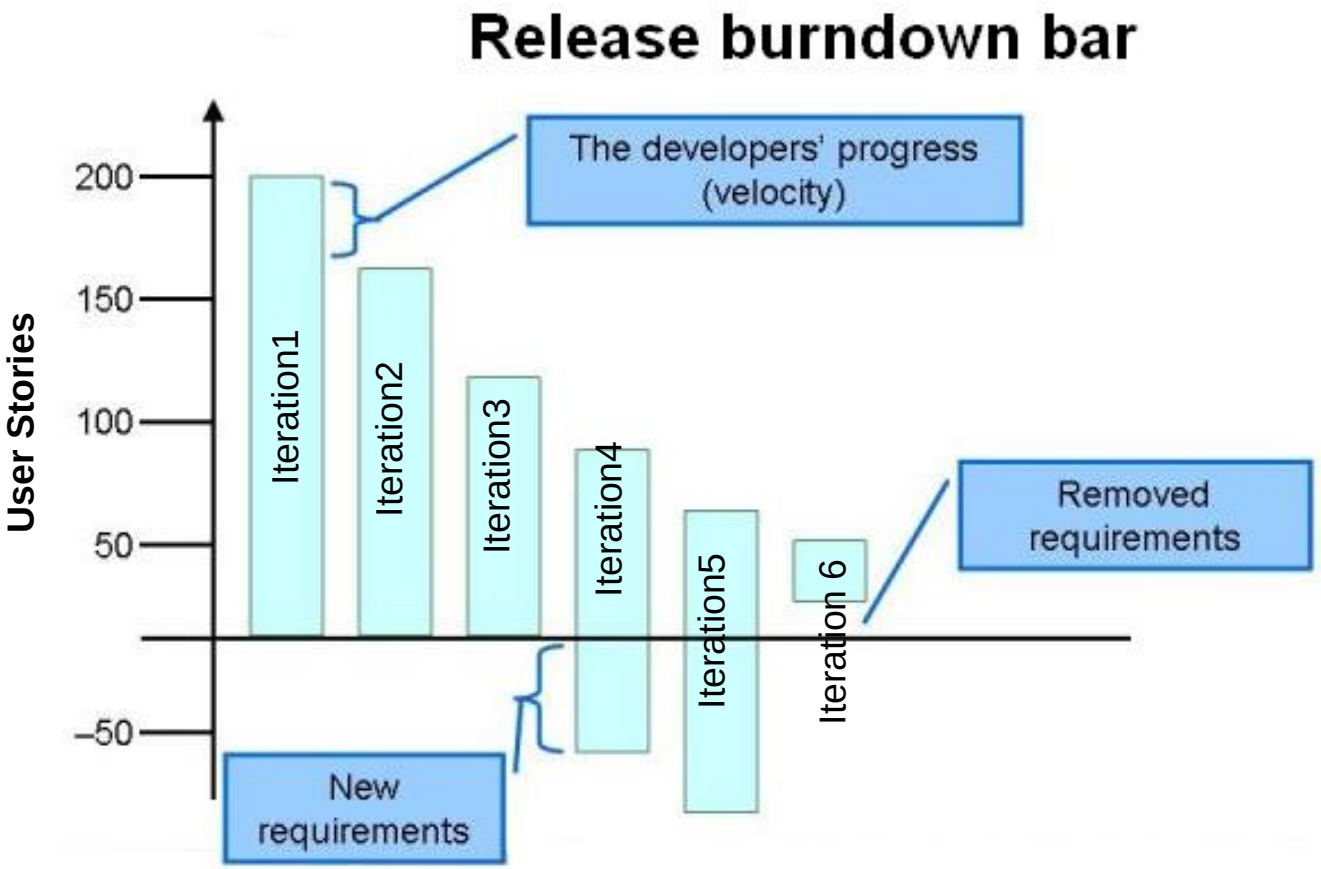
Burn chart

Burn-down & Burn-up combine chart



Burn Chart

Burn Down Bar Chart



Retrospectives

- Retrospective is conducted by agile team at the end of each sprint and each release
- Retrospective sessions are self-reflection sessions, to understand how we are doing and how can we improve it further. Kaizen is at the heart of every retrospective session.
- Retrospective leads to adaptive planning
- Depending upon the need, confidence of team, time of retrospective, nature of team members structure of retrospective can be adopted

Retrospectives

Structure of retrospective

- Set the stage (set ground rule/ working agreement)
- Gather Data (Share relevant data)
- Generate Insights (ask why it happened and how it happened)
- Decide what to do (Action plan)
- Close Retrospective (Appreciate participation)

Types of Retrospectives

- **Start, Stop, Continue** : The facilitator captures team open-ended feedback using a wheel that encourages team members to assess an iteration or milestone. What should be start new, what should we stop, what should be continue from last iterations. **Dur: 10 to 25**
- **Force Field Analysis** : A plan designed around the force field analysis technique. A retrospective for your whole company/department or to analyse a particular topic. **Dur : 60**
- **Pomodoro Retrospective** : Focused and time-constrained by using the Pomodoro technique. Useful for determining a single action to improve the work of a small team. **Dur : 25**
- **Four L's Retrospective** : Liked – Learned – Lacked – Longed For. Iteration and project retrospectives as well as for retrospection of training and conference events.. **Dur : 90 - 120**
- **Sailboat** : What anchors slow the team down, what wind propels it forward?. Good for the "gather data" and "generate insights" portions of a retrospective**.** **Dur : 90 - 120**

Temperature Reading

This retrospective can be conducted, to know the feeling, sentiments of the team. This is conducted by a facilitator and it follows following structure. Adequate time should be give to each part.

- Specific Appreciation
- New Information
- Puzzles
- Complaints with Recommendations
- Goals, Hopes and Wishes

Retrospectives

ROTI (Return on Time Invested)

- Feedback and Adaptation are part of Agile Project Management. Feedback starts with meeting and specifically retrospective meetings.
- ROTI (Return on Time Investment) is a quick and easy method to gauge the time spent on meetings or workshops, and to improve their effectiveness.
- Take 5-15 minutes at the end of the meeting to ask participants to rate their return on time invested, using the **Fist of Five technique**
- If most of number you are getting is 1 or 2 then it means something is serious and you are wasting time and something different need to be done
- ROTI is quick, easy, sometimes funny, and works very well, even with top management.

Agile Estimations & Metrics

Relative Sizing

- We know at the start of project not enough detail is available to estimate the size of work. But high level requirements are available in the form of epic or module
- Best way to estimate in this situation is relative sizing method
- Based on current understanding of system and modules team picks up smallest size epic and then start comparing that with other modules.
- Down the line at the time of release planning features or user stories are identified in epic and finer estimation is possible but again at this level preferred method of estimation is Relative Sizing
- Hours, duration, cost based quantitative estimation makes sense only when you know the task to be done. But even for user story level estimation in release planning neither you have time not it is worth to identify task to do quantitative estimation

Story Points

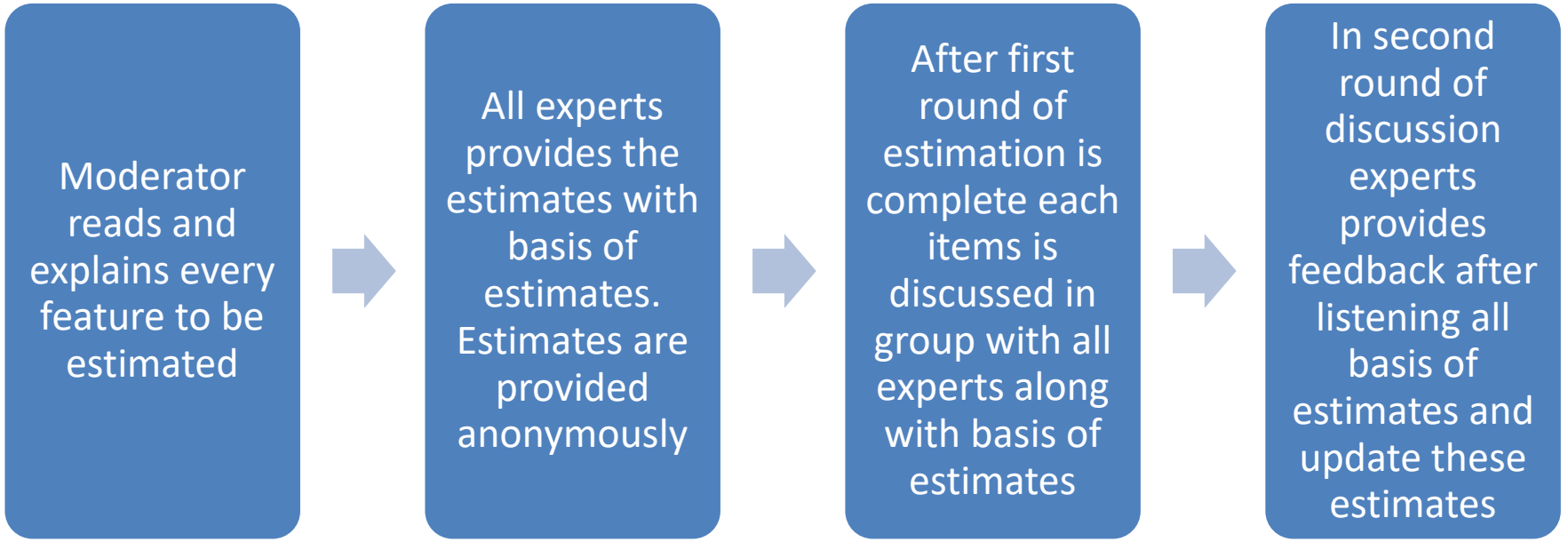
- User story is the functionality of the system which has value in the eyes of customer
- Count of user stories is called story points this is used to size user stories.
- Because all stories are not same in efforts, complexity and risk therefore different methods are used to normalize this factor
- Size is function of relative efforts, complexity and risk not of duration or IEH etc. Therefore when estimating size take away duration, number of people etc from your mind.
- Some story point estimation methods are
  - T-shirt size – Story Point Estimation
  - Fibonacci Series – Story Point Estimation
  - Fruit size – Story Point Estimation

Story Points

- Let us say our **product back log** has
  8 x bananas (5 points each) = 40 story points
  10 x pineapple (20 points each) = 200 story points
  6 x oranges (8 points each) = 48 story points
  30 x bunches of grapes (2 points each) = 60 story points
- **Total story points** in product backlog is 40+200+48+60 = 348
- Les us assume that based on guess or some previous release teams knows that their velocity is 30 story points
- Total iterations required = 348 / 30 = **11 iterations**
- If iteration is of 2 weeks then **22 weeks** required to finish the work in product backlog

Wideband Delphi

- Wideband Delphi is consensus based estimation method. A moderator facilitates the session.
- Delphi technique keep names anonymous and helps avoiding biases and politics



Planning Poker

- Planning poker is widely used estimation technique in agile projects. This is a variation of Wideband Delphi.
- Logistic Required for Planning Poker Estimation
  - As many set of deck cards as many estimators in the room
  - All the team members with a deck of cards inside the room
  - A coordinator with laptop
  - A facilitator
  - Product owner
- Planner Poker Process
  1. Facilitator projects the product backlog on wall using a projector
  2. He reads loudly a user story at a time so that everybody in the room understand what they need to estimate
  3. Estimator estimates it silently and shows the card only to the coordinator
  4. Coordinator documents all the estimates provided by all estimator
  5. If there is difference between estimates then estimators who provided two extreme estimates need to talk the basis of estimates.
  6. After a short discussion. Next round of estimation starts the same way as before and coordinator document the estimate provided by estimator
  7. Keep repeating steps 3,4,5,6 till estimates provided by estimators are not converged
  8. Read next user story

Affinity Estimation

- If product backlog has more than 20 items then implementing planning poker method of estimation is too much time consuming. In that case affinity estimation technique is most suitable
- This helps in providing coarse estimates and good enough for release planning but for iteration planning you should use Planning Poker.
- Logistics for Affinity Estimation
  - As may set of cards with printed user story. A story on a card.
  - A wide magnetic board with magnetic balls
  - All the team members inside the room
  - A coordinator with laptop
  - A facilitator
  - Product owner

Affinity Estimation

Facilitator writes “smallest” at extreme left end of magnetic board and “largest” at extreme right of magnetic board

Set the ground rule of estimation session. Like it is mute session so we should not talk at all. All cards which are put on table need to be arranged on wall from left to right in terms of their relative size. You can shift position of cards from left to right or vice versa put by other member but **without** talking, convincing anybody, arguing or any kind of body language.

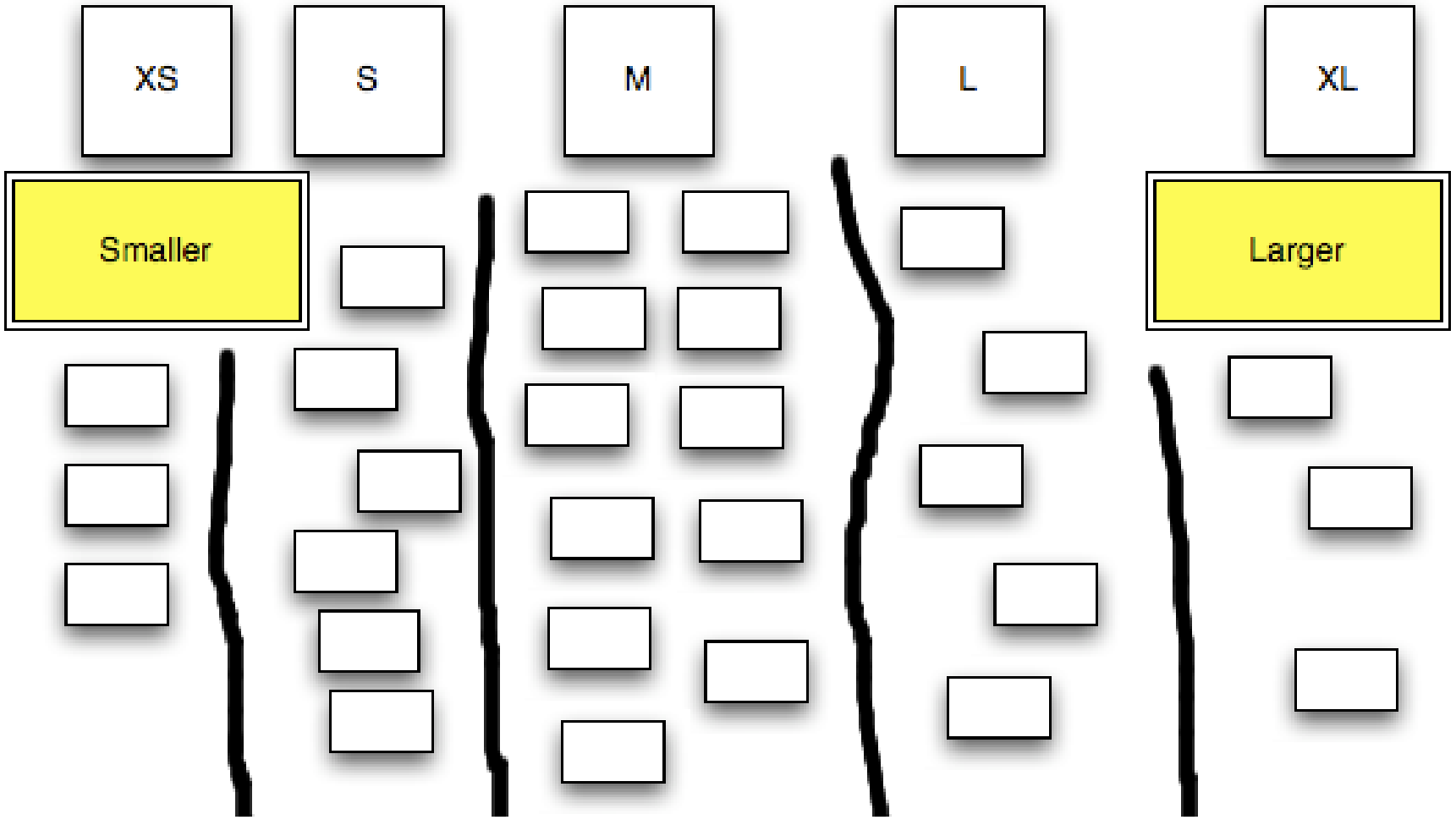
Estimators starts reading card and arranging them on the wall. If required they can discuss in private with product owner

If they are not able to understand any card ever after discussion with product owner they can move the card into Questionable column

After the session is over they need to group these cards in 5 or more groups like smallest, minor, medium, larger, largest

Coordinator need to type all these estimates into computer and share with project manager

Affinity Estimation



Velocity

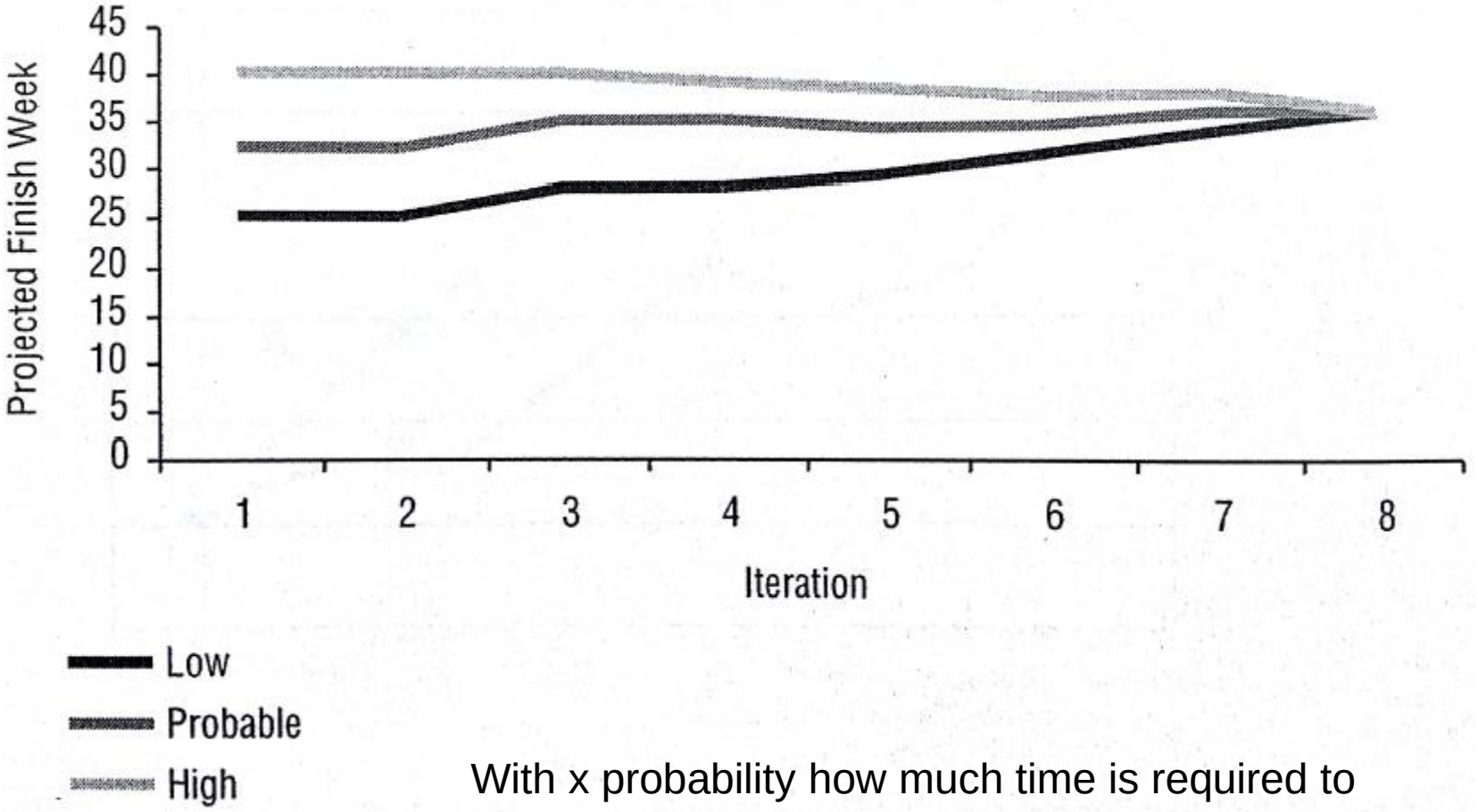
- Velocity is number of story a team can do in an iteration.
- When you are starting a new project that point of time best way to know the productivity is guess based on previous experience. But never benchmark velocity of team based on previous project
- It takes 3-4 iteration to stabilize velocity of team
- Team members should not be added or removed from development team otherwise it destabilized the velocity
- **Methods to improve velocity**
  - Periodically pay down technical debt
  - Improve customer involvement
  - Support energized work environment
  - Provided needed resources (people, equipment, software etc)
  - Offload admin duties of programmer
  - Add experienced programmers

Escaped Defects

- In Agile escaped defects mean those defects which are leaked to next iteration or sprint.
- **Escaped Defect Rate** = Total number of weighted defects for a sprint / number of total stories delivered in that sprint
- Tracking the trend of this metrics helps you understanding the quality of product and maturity of testing process

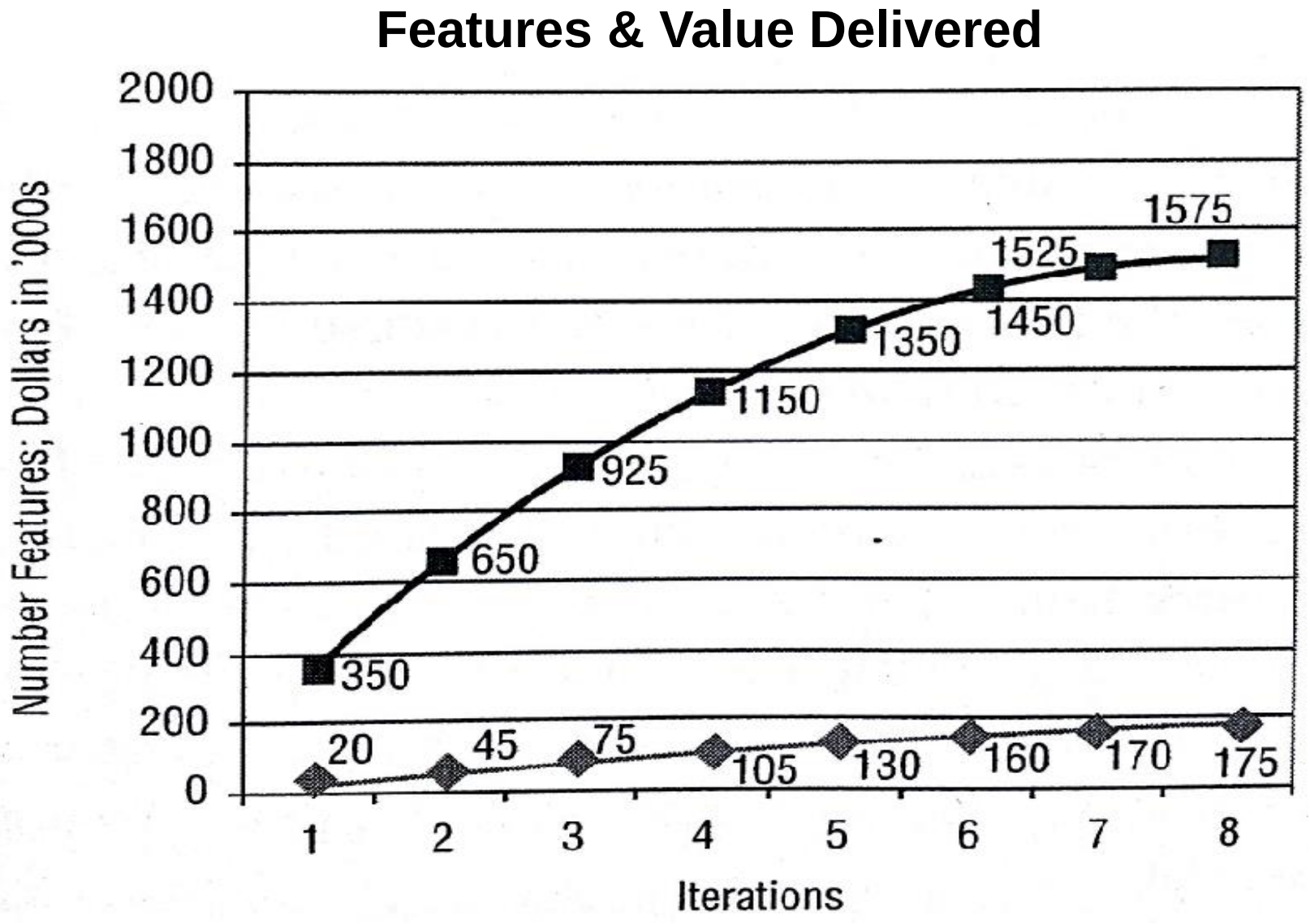
Metrics

Projected Scheduled

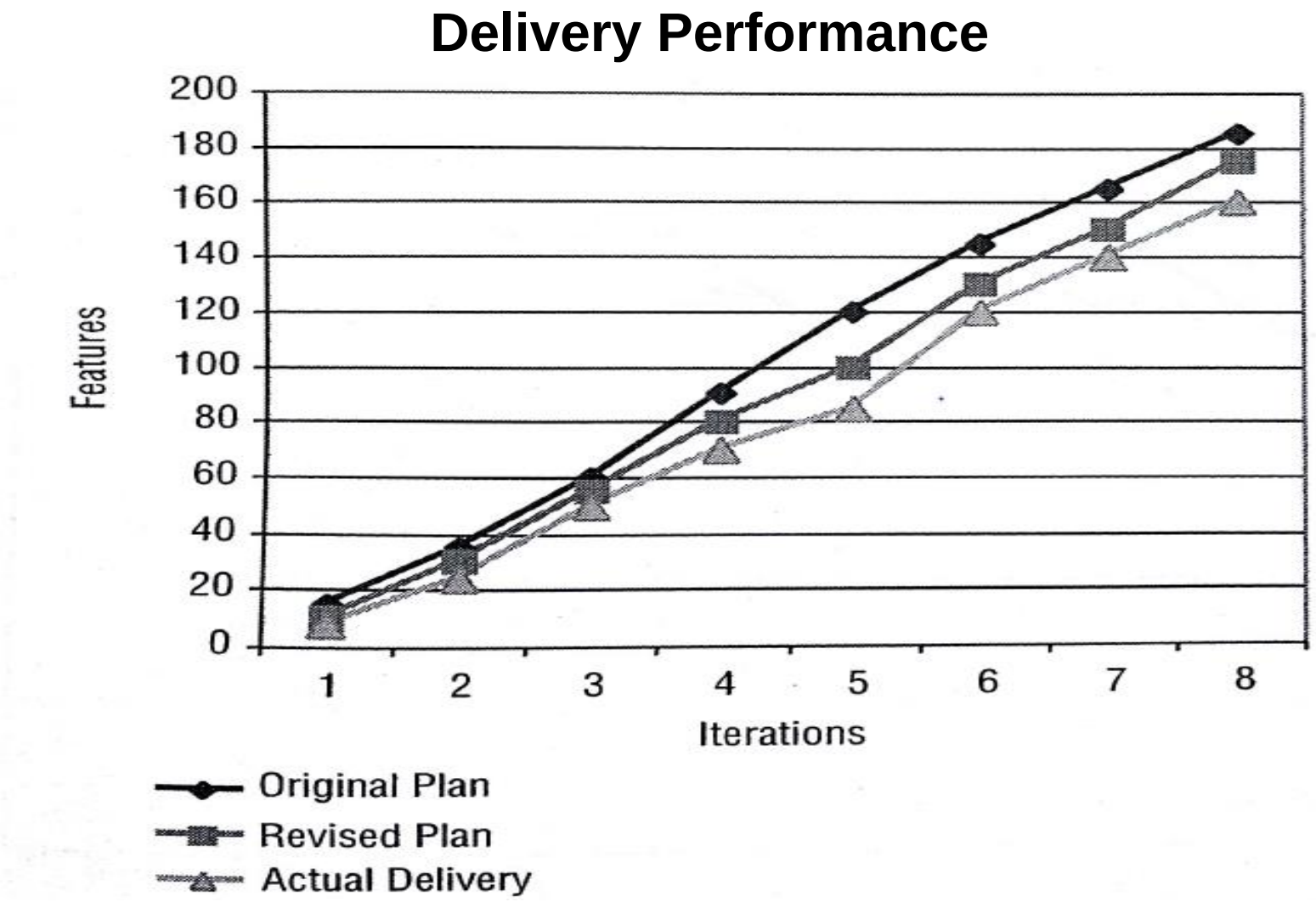


With x probability how much time is required to complete the project. Track this information for every iteration

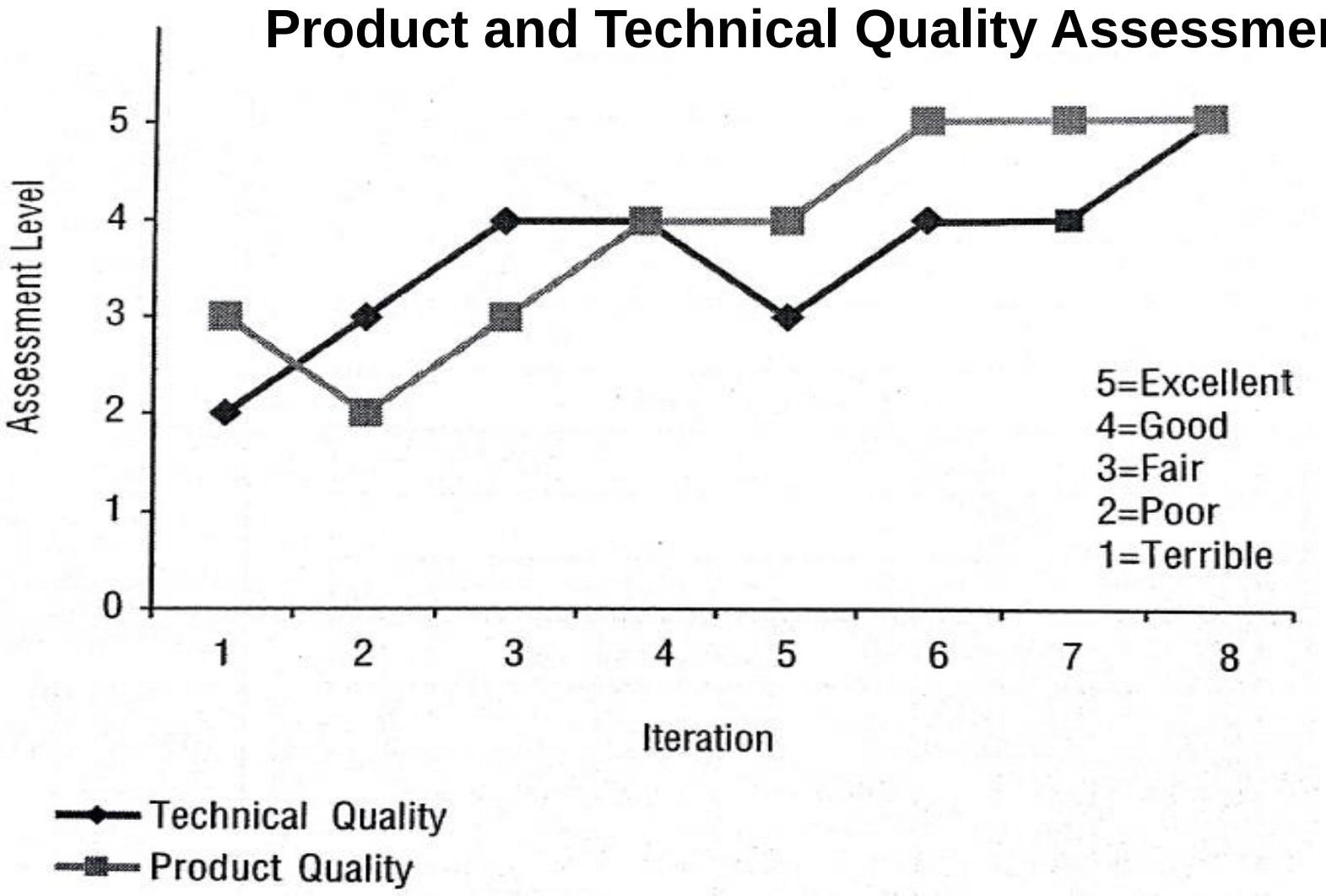
Metrics



Metrics



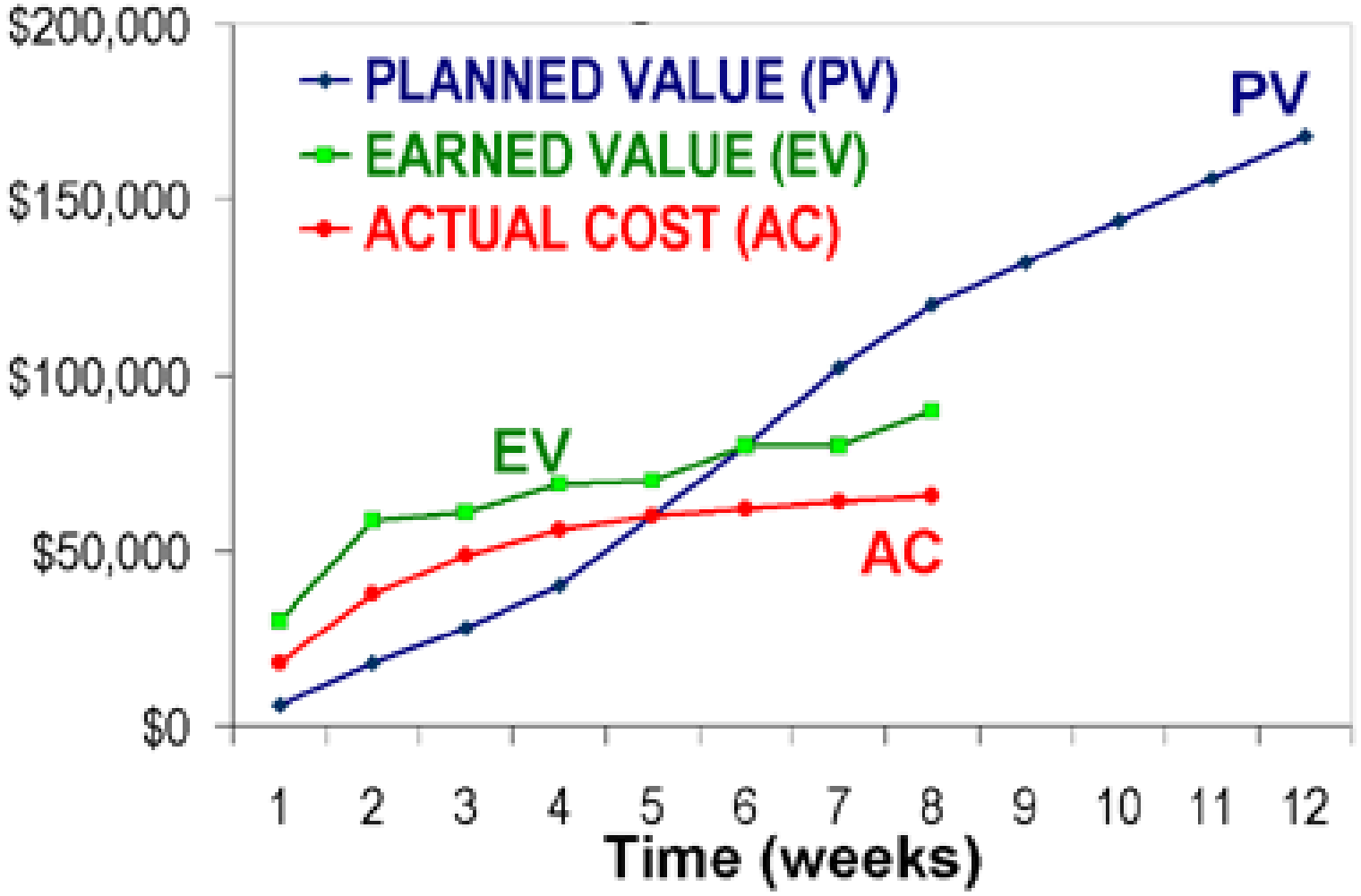
Metrics



Earn Value Management

- CV = EV- AC
- CPI = EV/ AC
- SV = EV –PV
- SPI = EV/PV
- EV = BAC - ETC
- EAC = AC + ETC
- EAC = BAC / ETC
- EAC = BAC / (SPI * CPI)
- VAC = BAC –EAC
- TCPI = (BAC-EV) / (EAC-AC)

S Curve

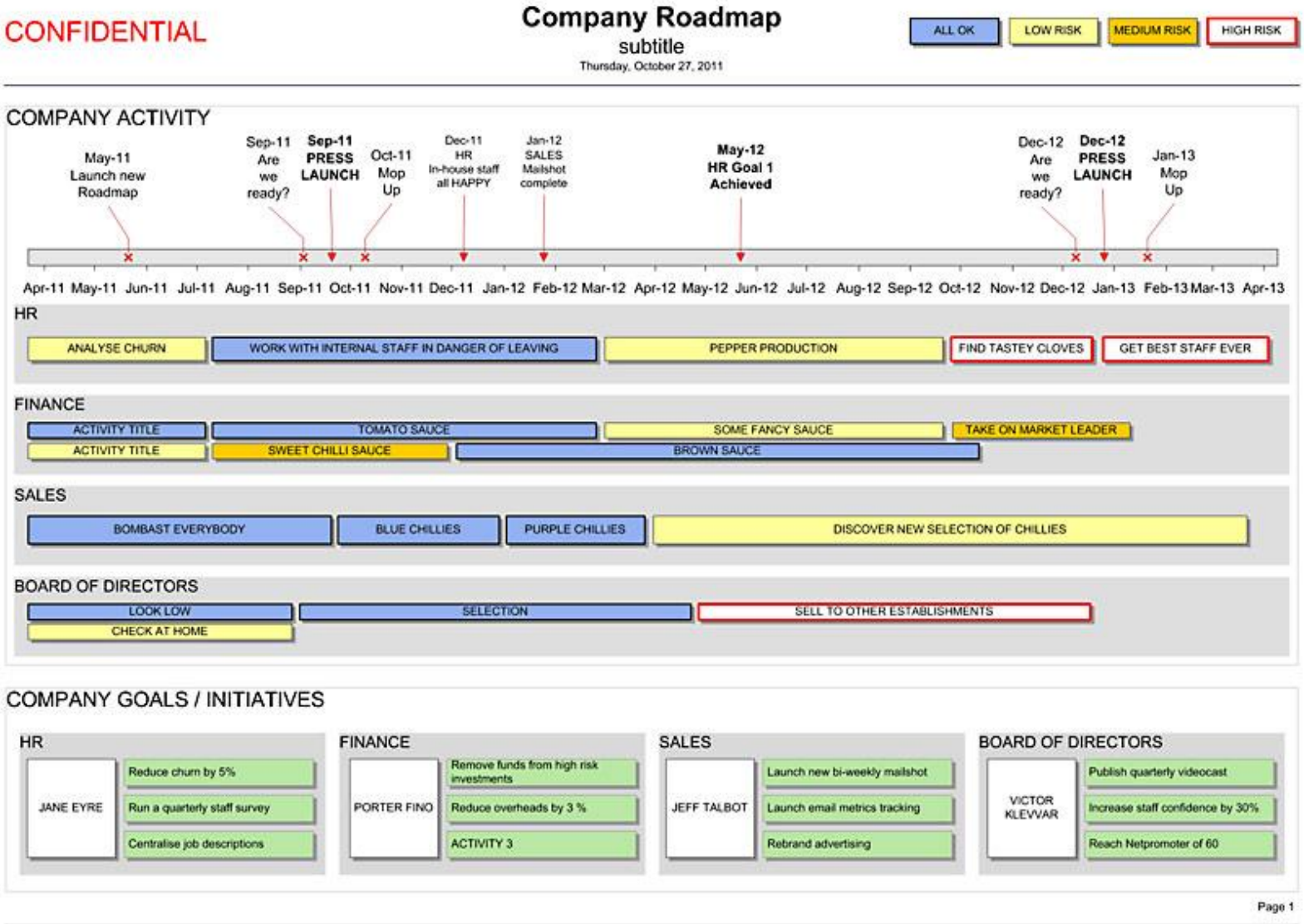


Tools & Techniques
Agile Analysis & Design

Product Roadmap

- A product roadmap is a plan where the features are planned for release on a timescale
- It talks about business value addition to the customer by each release
- Date, features and objective of each release
- Overall product features (product backlog and business story)
- Technical backlog (non-functional requirements, technical story)

Product Roadmap



Backlog

Product Backlog

Priority	Estimate	Sprint	User Type	Story	Story Type
1	1	1	Customer	I can see when the next show will begin for the show page I am on	Story
2	2	1	Editor	I can select what I want to display for each "section" within the editorial content section of the page. My options include last episode, next episode, selected forum posts, selected editorial articles (tvg generated), no selection and free form text	Story
3	2	1	Editor	I can select what picture (if any) I want to display for the corresponding content section	Story
4	5	1	Editor	I can select the default tab for the user to see upon visit to the page, for each show	Story
5	5	1	Customer	I can roll over the fields in the media player and see the various tabs change	Story
6	13	2	Editor	I can modify the existing headline for any show page	Story
7	1	2	Customer	I can select anohter show page in the drop down list next to the countdown clock	Story
8	1	2	Customer	I can click "remote record" and have the show for the show page I am on record on my tivo device	Story
9	1	2	Customer	I can click "join the discussion" button (or link) on the show page which takes me to the appropriate forum page for that show	Story
10	1	2	Customer	I can see how many recent posts have been posted in the forum for the show page I am on	Story
11	3	2	Customer	I can see how many recent replies have been posted in the forum for the show page I am on	Story
12	5	2	Customer	I can blog about the show for the show page I am on (I need to be signed in to see this)	Story
13	13	3	Customer	If I am not signed in, I can see a link to sign in	Story
14	13	3	Customer	If I am logged in, I can click "favorites" and have the show page added to my favorites menu on the site	Story
15	13	4	Customer	If I have not contributed to the poll, I can see the poll questions and submit to the poll	Epic
16	20	5	Editor	I can create a new poll for a specific show	Epic
17	20	6	Editor	I can close an existing poll for a specific show	Epic

Product backlog grooming is continuous activity. PO is responsible for this.

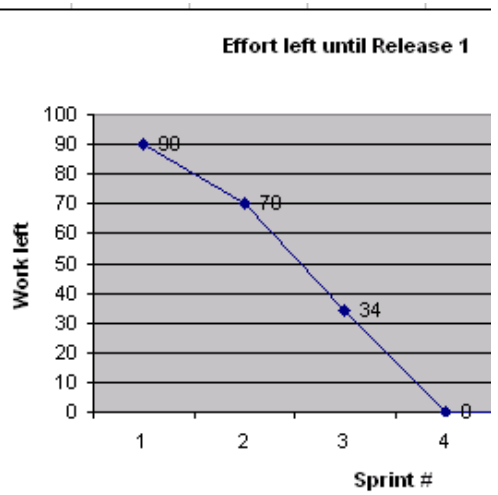
Backlog

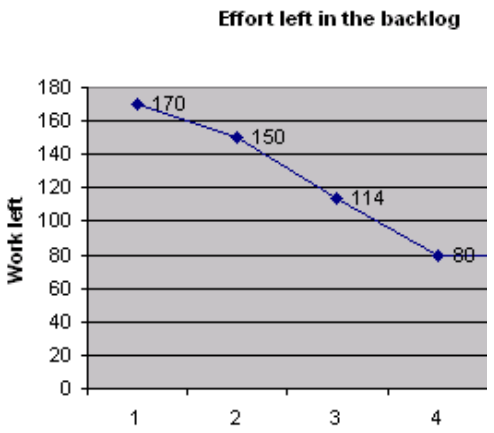
Release Backlog

ID	Description	Sprint #	1	2	3	4	5	6
	Effort needed for Release 1 as in the beginning of the sprint		90	70	34	0	0	0
1	Set up continuous integration system		5	0	0	0	0	0
2	Create compilable application skeleton		5	0	0	0	0	0
3	Display current temperature in a simplest possible way		13	0	0	0	0	0
4	Set up the web server for serving weather data		3	0	0	0	0	0
5	Implement stubby WeatherML support on the server side		13	0	0	0	0	0
Sprint 1	*Make sample data go from server to device*							
6	Graphics support on the client side		20	0	0	0	0	0
16	Make the graphics library draw some icon and sample temperature text		-	13	0	0	0	0
17	Draw the real weather screen		-	8	0	0	0	0
7	Implement support for several days		8	8	0	0	0	0
8	Implement support for rain, snow, etc. icons		2	2	0	0	0	0
9	City changing support		-	5	0	0	0	0
Sprint 2	*Minimal working version*							
10	Fetch one day temperature data from the weather provider system		?					
11	Fetch rain, snow, etc details from the provider							
12	Fetch several days data from the provider							
13	Auto-refresh feature							
Sprint 3	*Plug in the real weather data*							
Release 1	**Sellable version**							
14	Inject simulated ads from the test server							
15	Plug real ads in							
18	Change current city automatically according to the cell info		40	40	40	40	40	40
Sprint 4	*Advertisements support*							
Release 2	**Ad-supported version**							
	Effort in the whole backlog		170	150	114	80	80	80

In a real product backlog, items are more likely to be in the form of user stories. E.g. this item could be stated as "As a user I want to see the real weather data so that I could know what to wear before leaving home"

Backlog state taken after the end of sprint 3 = after release 1





Release backlog is created during release planning meetings. PO is responsible for this

Backlog

Sprint Backlog

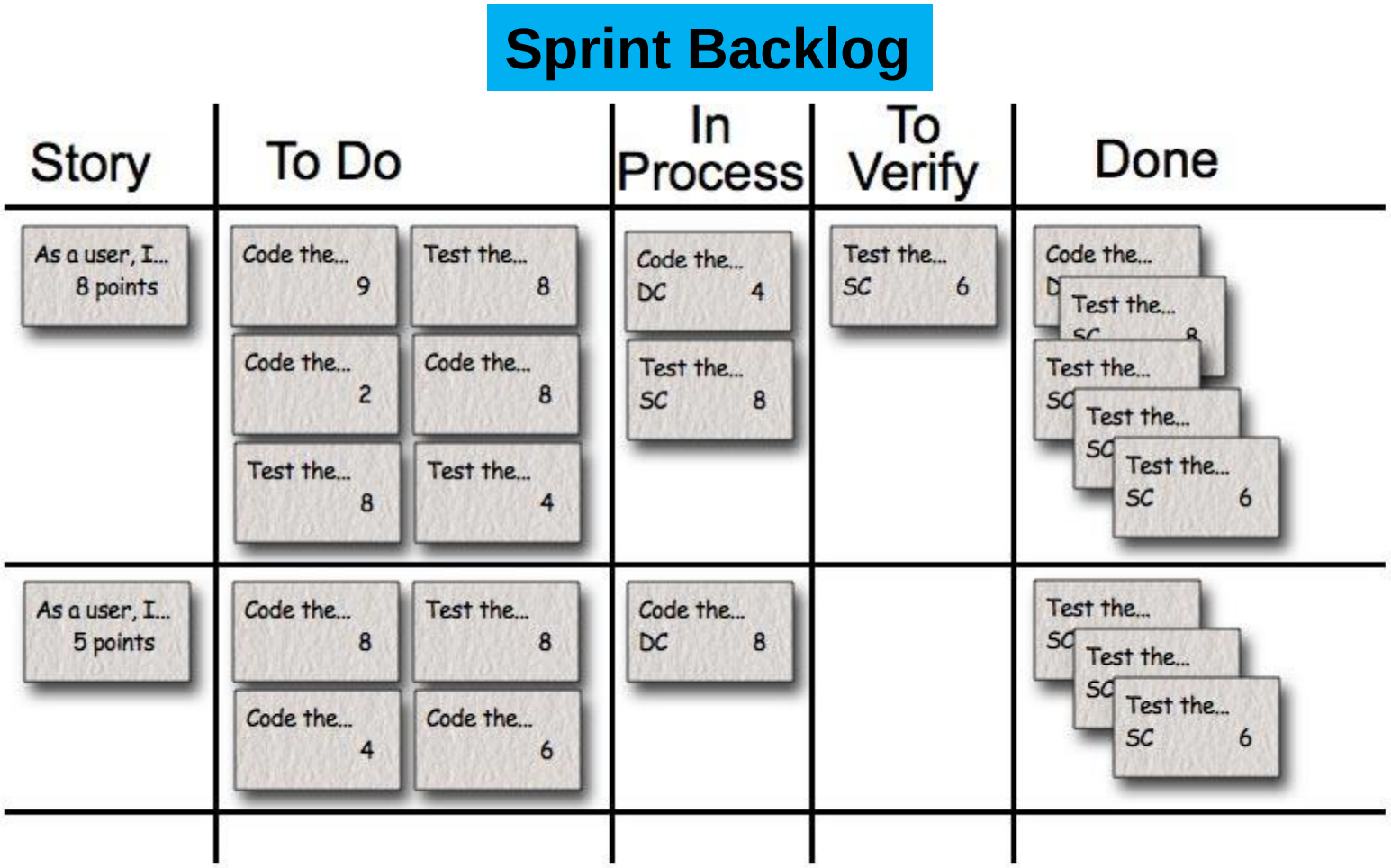
- Unlike product backlog and release backlog which is comprised by user stories, sprint backlog is comprised of task
- Sprint Backlog can have following types of tasks
  - Design Tasks
  - Coding Tasks
  - Testing Task
  - Documentation Tasks
- No one is allowed to add user story or task in Sprint backlog but team can delete task if they feel out of time
- Sprint backlog is developed at the time of sprint planning

Backlog

Sprint Backlog

- Generally every task in sprint backlog should be related to some user story
- To identify task for user story programmers need to do the detail discussion about user story with PO
- Each task in sprint back log should have following information in the form of Task card
  - Task description
  - Estimated hours
  - Owner name (if taken by any team member)
  - Exit criteria
  - Verification method (test or inspection)
  - Who will perform verification

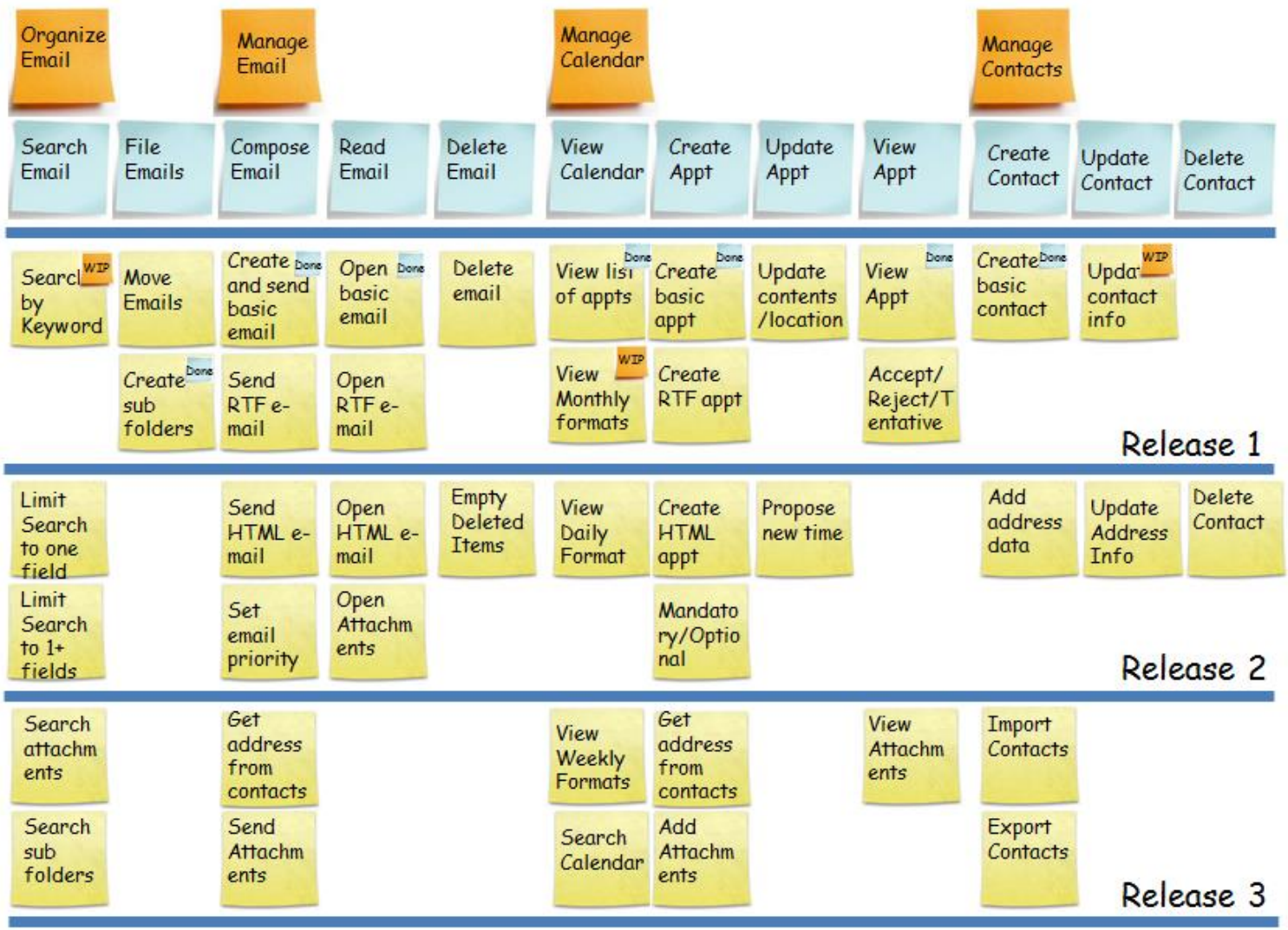
Backlog



Story-maps

- The main purpose of story mapping is to understand the end-to-end user requirements and stitch them in a logical thread so that users get maximum return on the business value
- Benefits of Story-maps
  - Overall workflow of the system help in understanding value chain.
  - Relationship between large stories and child stories get established
  - Completeness of backlog can be validated
  - A basis of prioritization
  - Helps in release planning

Storymaps



Tools & Techniques
Product Quality

Cost of Quality

Cost of conformance	Cost of nonconformance

• Appraisal Costs
  • checking & Testing Purchased Goods
  • In-process and Final Inspection Testing
  • Field Testing
  • Product, Process and Service Audits
  • Calibration of Measuring and Test Equipment
• Prevention Costs
  • Capability Evaluation
  • Error Proofing
  • Quality Planning
  • New Product Review
  • Quality Improvement Project
  • Quality Education and Training
  • Supplier Evaluation
	• Internal Failure Costs
	  • Re-design
	  • Downtime
	  • Re-testing
	  • Shortages
	  • Delays
	  • Rework
	  • Lack of flexibility
	• External Failure Costs
	  • Environmental Costs
	  • Loss due to Sales Reductions
	  • Warranties
	  • Repairing Goods
	  • Complaints

Definition of Done (DoD)

- The DoD changes over time. Organizational support and the team's ability to remove impediments may enable the inclusion of additional activities into the DoD for features or sprints.
- Continuous Integration (CI) helps you validating the "Doneness"
- There are 3 level of DoD
  - Story DoD
  - Iteration DoD
  - Release DoD

Definition of Done (DoD)

Story "Done"

- Unit test should provide 60-70% test coverage
- Story is either written in pair or reviewed by peer
- All code checked in
- All unit code passed
- All acceptance test case passed
- Story accepted by owner

Definition of Done (DoD)

Iteration "Done"

- Iteration should have defined Iteration Goal
- All acceptance test cases should run for all user stories in Iteration
- All stories completed must be accepted by the product owner
- Defects identified are fixed or planned for future
- Code performance is tested and accepted
- If database is involved then database script should be available, automated and tested
- Backup of iteration work product is taken

Definition of Done (DoD)

Release "Done"

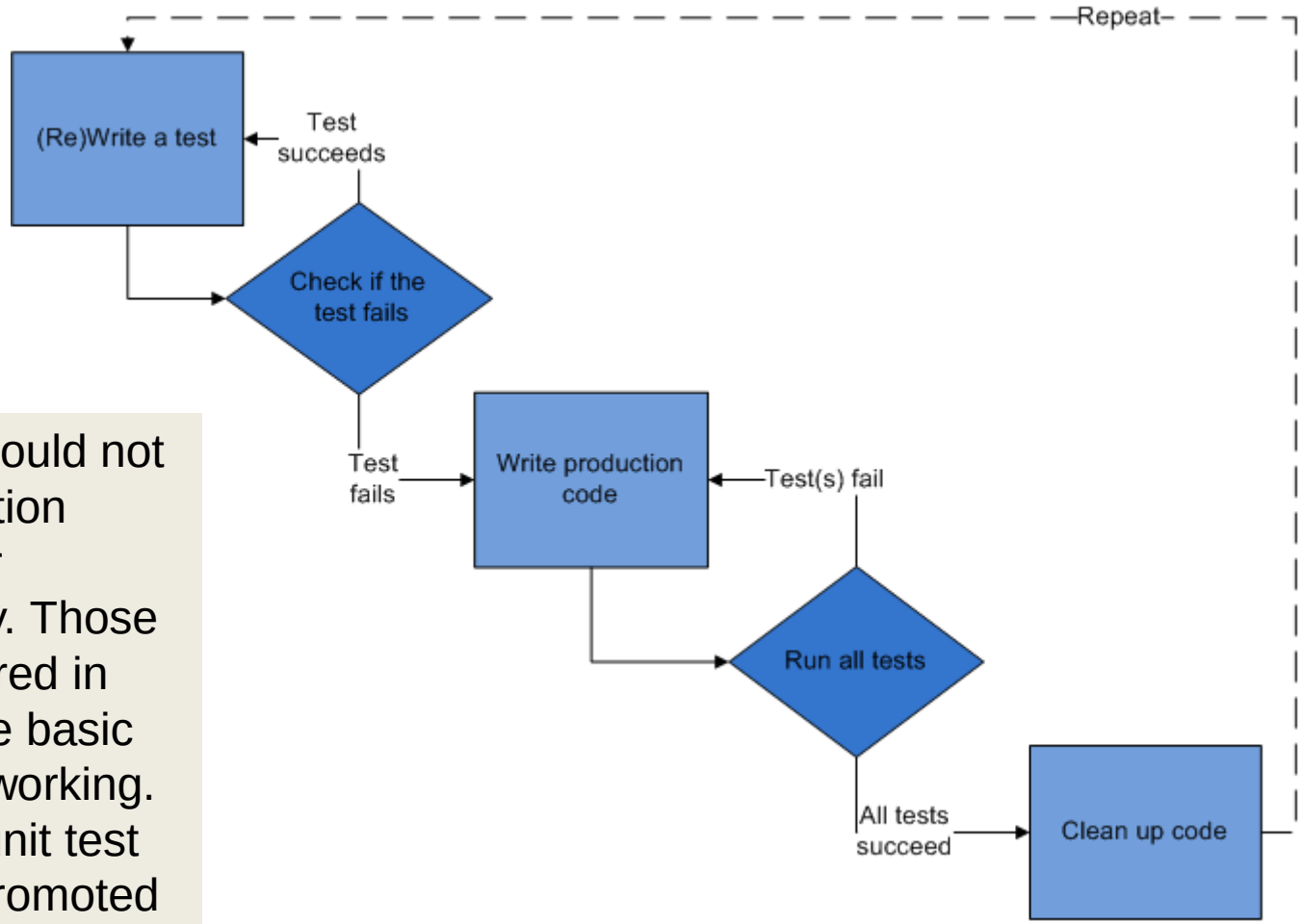
- Release should have defined Release Goal
- Product has formal release date
- Product is deployed on staging area
- Stress testing done and results accepted
- All non-functional requirements are tested and results accepted
- Required documentation is available
- Release should not have any known bug

Feedback Techniques of Agile Project Management

Timely & regular feedback is heart beat for producing a good quality product and managing risks. Following techniques are used to get quick feedback.

- Release planning
- Sprint planning
- Daily stand-up
- Sprint review
- Sprint retrospective
- Pair programming
- Peer review
- Customer involvement throughout PLC

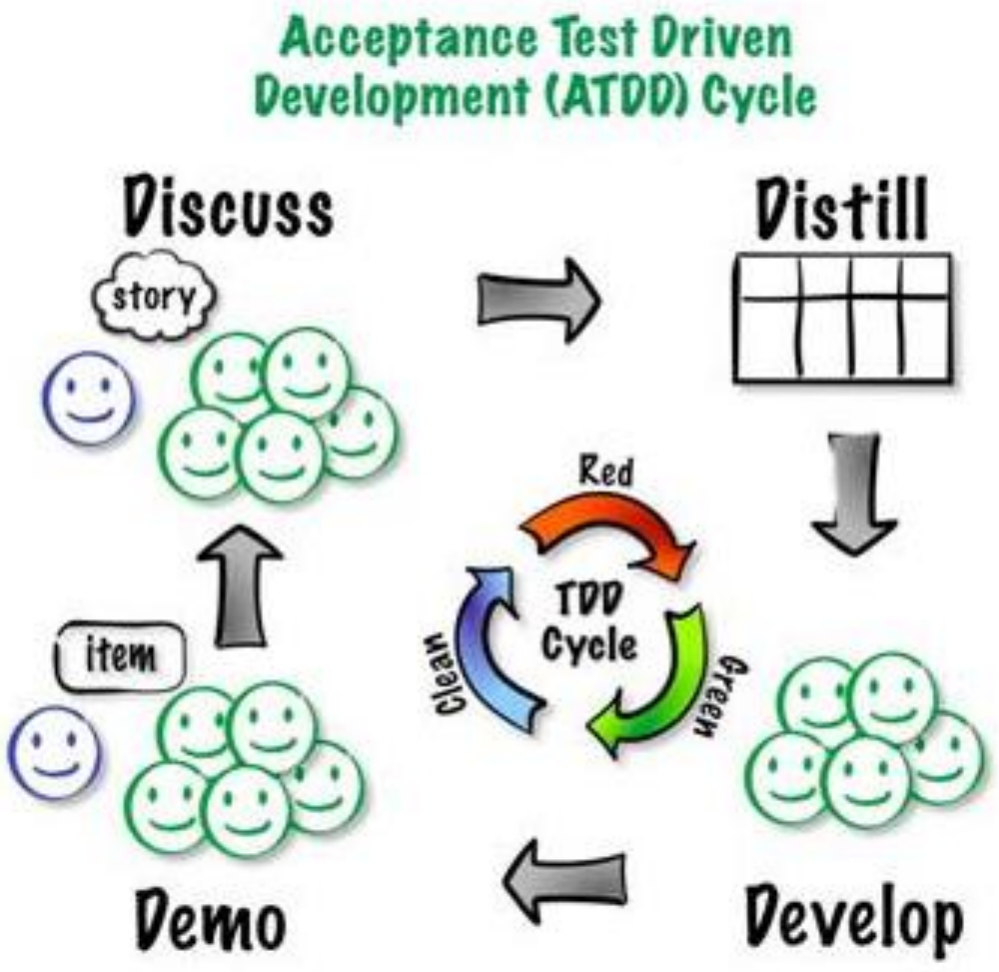
Test Driven Development (TDD)



- Programmer should not consider exception cases and error handling initially. Those can be considered in next cycle, once basic functionality is working.
- If built passes unit test then only it is promoted to the next step as a candidate for acceptance test build

Acceptance Test Driven Development (ATDD)

- Major difference between ATDD and TDD is in ATDD user story or **functionality is tested** not the task like in TDD.
- Using ATDD practice you are not allowed to add more functionality until earlier user stories are successful.
- Collection of acceptance test cases forms regression test and it should run automatically on built server

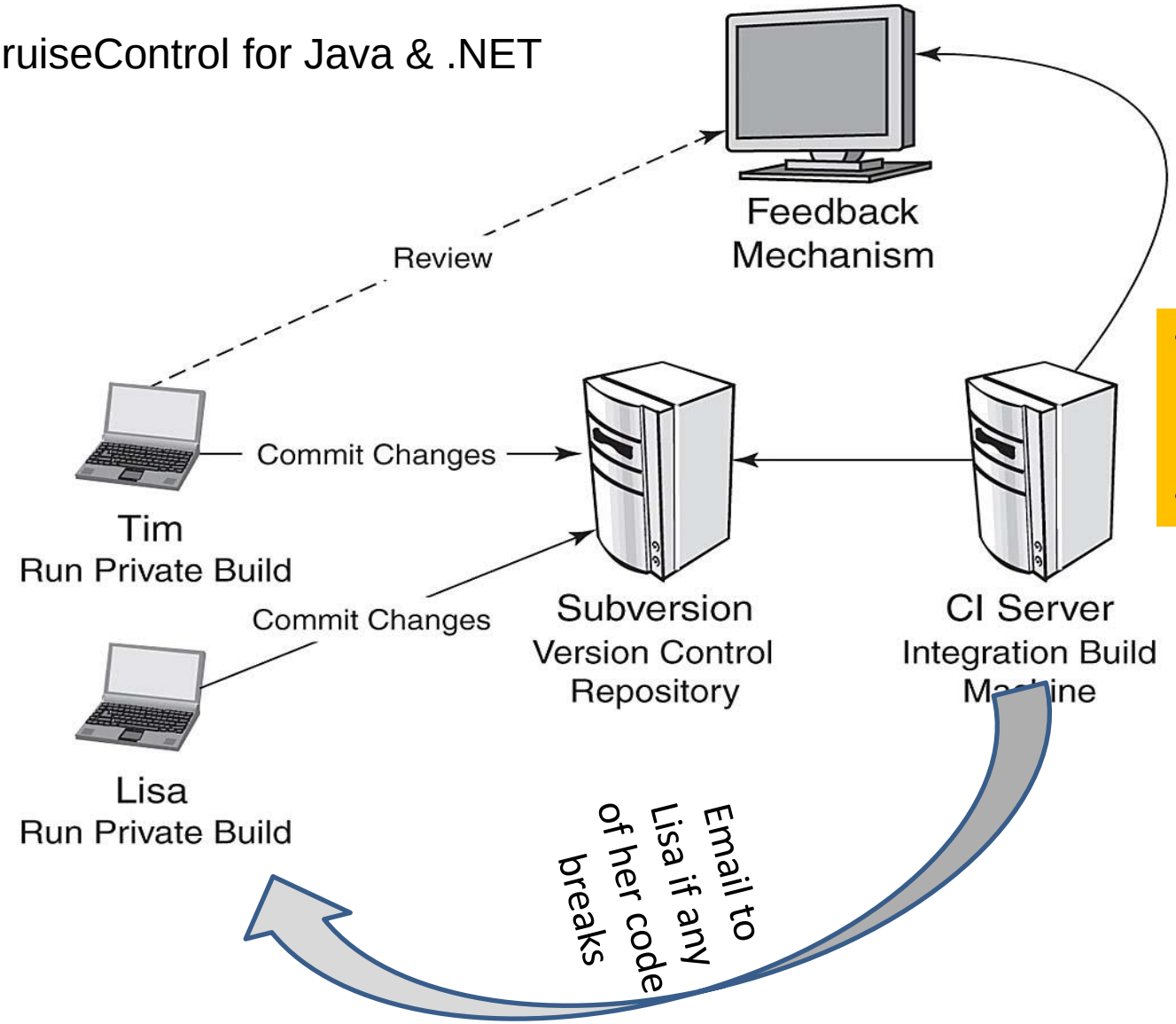


Exploratory Testing

- After acceptance test cases are passed then Agile tester may choose to perform exploratory testing
- In exploratory testing agile tester tests those scenario which system user is not supposed to perform but if he perform in that case system should give proper message.

Continuous Integration

Tool: CruiseControl for Java & .NET



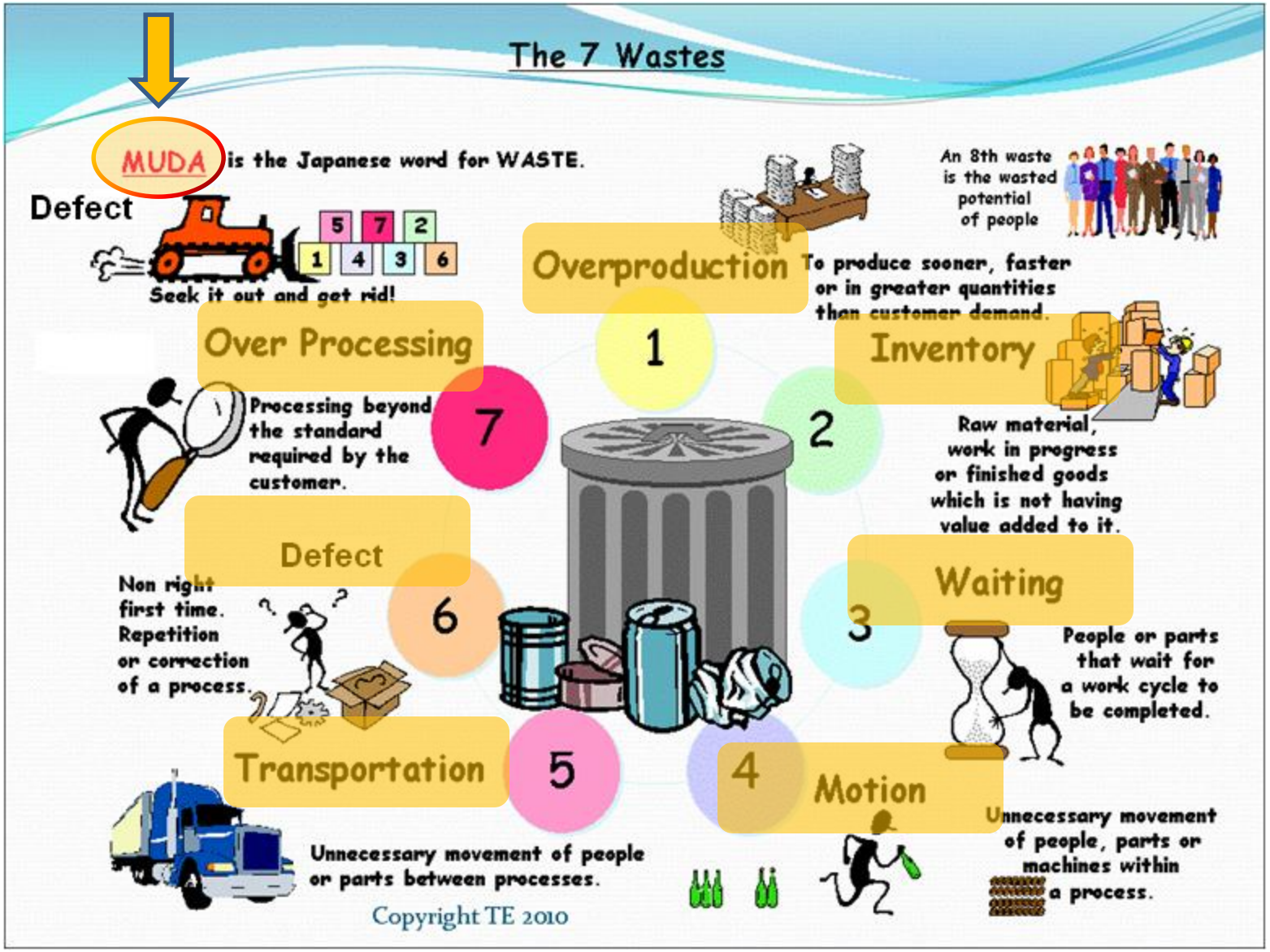
- Build should happen within 10 min, else change server
- No partial build

Continuous Process Improvement

Models for Continuous Process Improvement

- PDCA Cycle (by Juran - “Fintness for Use”)
- Score Card
  - Escaped defects, Overtime, Velocity, Cost, Time
- Balance Score Card (BSC)
  - Four aspects of BSC are Financial, Customer, Internal Business Proceeds, Learning and Growth
- Six Sigma (DMAIC)
  - Motorola in 1985
  - Used widely by Jack Welch @ GE in 1995
- Kaizen
- Eliminating the waste
- Kanban

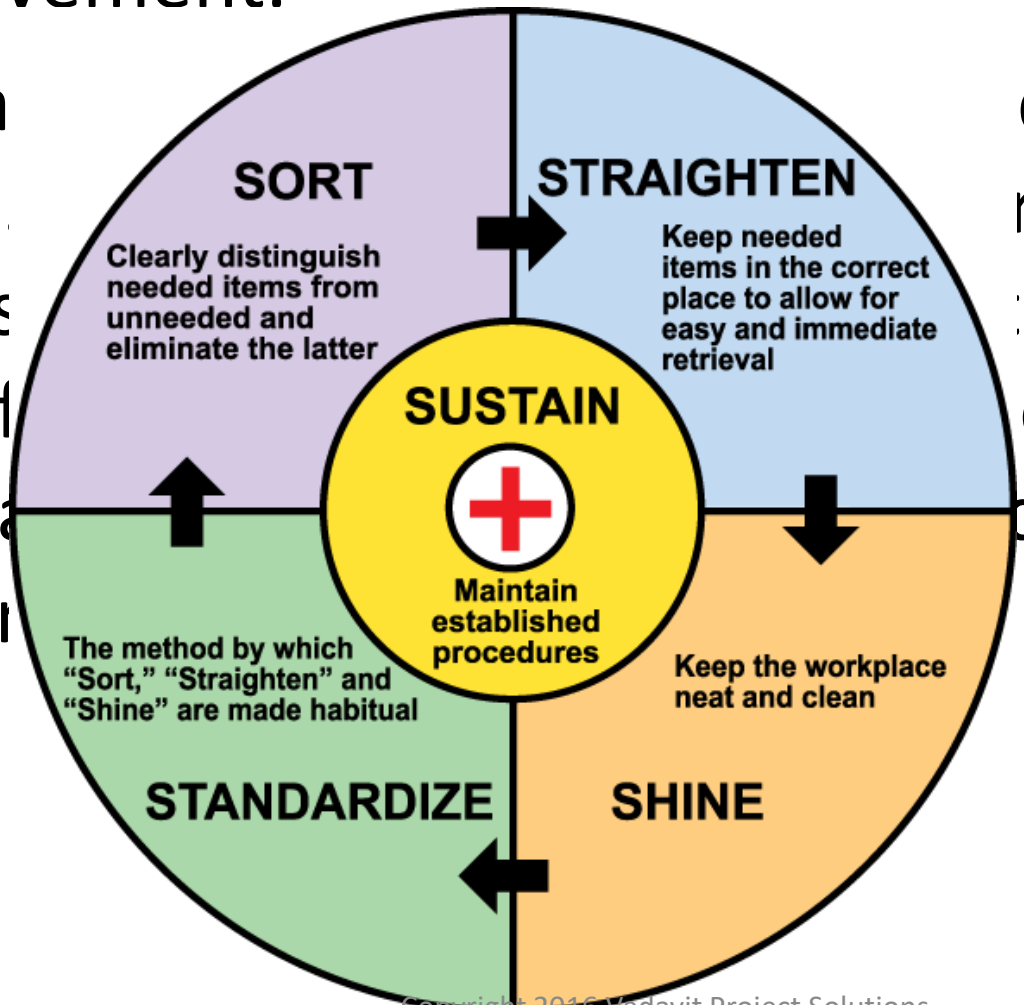
7 Wastes



TIM WOOD

House Keeping Standard (5S)

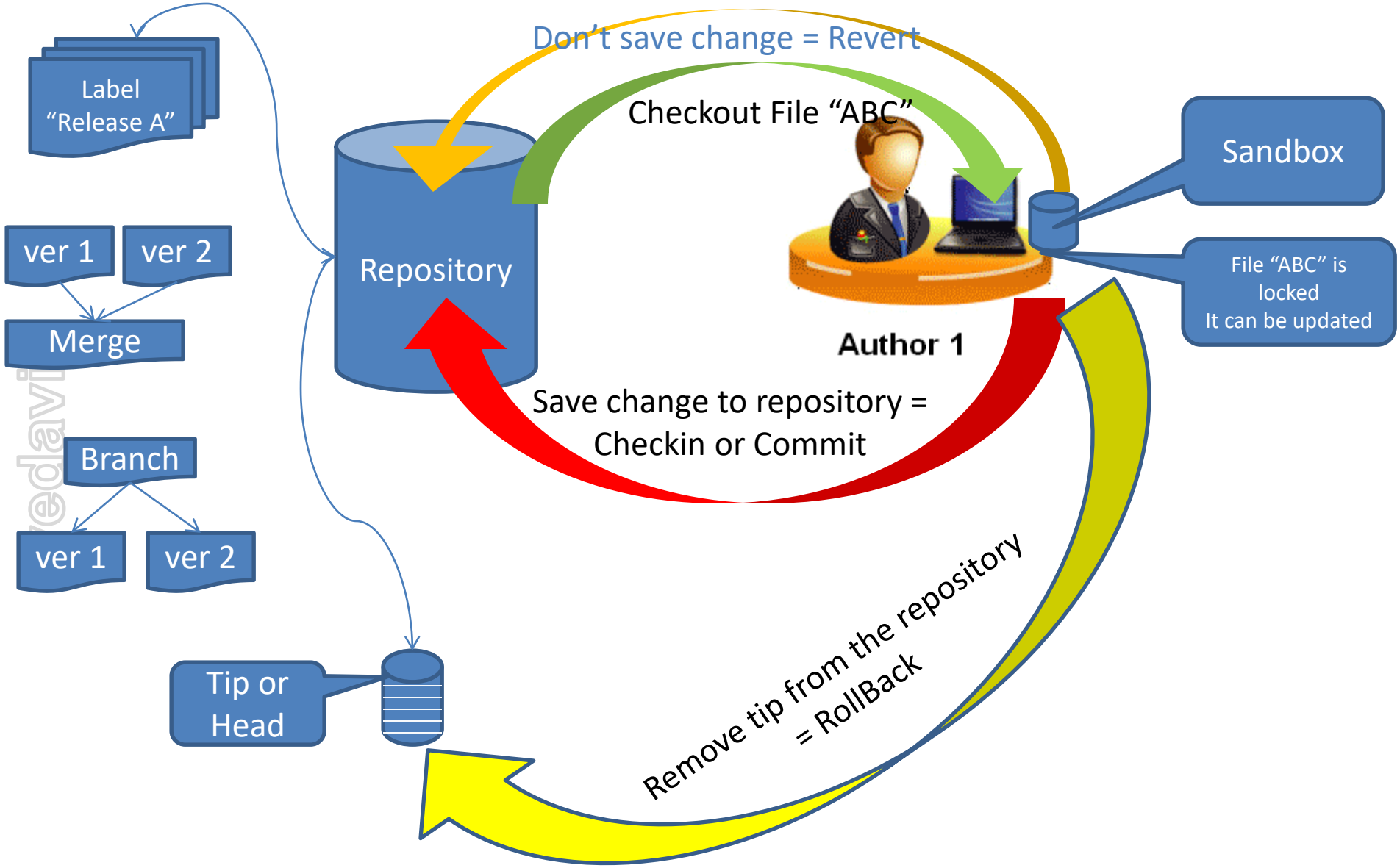
- In Lean 5S is foundation tool for continuous improvement.
- 5S is a dard becau rganize a work s iveness by identif ed, maint d sustai



Version Control

- A ***version control system*** is a repository of files and source code, with monitored access. Every change made to the source is tracked, along with who made the change, why they made it, and references to problems fixed, or enhancements introduced, by the change. This is very important tool to ensure quality product.
- **Collective code ownership** means whole team takes responsibility of the quality of the code therefore anybody can modify the code, if needed. (Encouraged in XP Agile Method)
- **Strong code ownership** means each person owns his module. Only he can make changes to his module.
- **Weak Code Ownership** means one person write the module but another person can make changes as long as owner helps them or they coordinate with owner
- When two users can update the same code at same time is called **Concurrent editing**. Conflict is resolved using merging method.

Version Control



Agile Tooling Mistake

For producing a good quality product agile project management relies heavily on automation. Therefore not investing in tools or using legacy tools for agile project management affects the moral of the team and quality of the product

Generally following mistakes are made by organization in agile tooling

- Using old version of tool which were working for waterfall
- Not investing on automation tool still continuing manual testing, documentation etc
- Working on age old slow hardware, while team is wasting time
- Not investing time on continuous-integration tool
- Using traditional waterfall metrics to manage the project
- Performing continuous integration only on mainline not on branch so branch code becomes buggy and unreliable.

Typical Software Tools in Agile Projects

- Agile project management systems (APMS) for effective communication and providing bird's eye view of the project
  - Rally, VersionOne, Jira
- Testing Tools to achieve 100% code coverage by auto unit testing.
  - xUnit
- Regression Testing Tool
  - CruiseControl, TFS
- Version Control to mange versions of code, product, test-case, tested product
  - SVN, SourceSafe, TFS

Tools & Techniques
Soft Skills

Shu-Ha-Ri

Shu-Ha-Ri is Japanese term according to this a student or team goes through three stages of learning. Whatever you teach them it takes time to master the art. Three stages are

- Shu: This is the initial obedient stage, when the student just follows the rules.
- Ha: The student questions the rules, understands their importance, and makes innovations within the rules.
- Ri: The student breaks the rules and creates his own rules, thereby escalating to the master level.

Collaboration

- Collaboration is animated two way conversation with real understanding and progress happening. Collaboration means whole is more than the sum.
- Eight agile practices which helps team and stakeholders in effective collaboration are
  - **Trust** => to thrive on relations
  - **Sitting together** => for quick and accurate communication
  - **Real customer involvement** => to make sure that team understand what to build
  - **A ubiquitous language** => to help team members, customer understand each other
  - **Stand up meeting** => to keep team member informed
  - **Coding standards** => to seamless integrate the work of individual's work
  - **Iteration demos** => to keep the team's efforts aligned with stakeholder goals
  - **Reports** => to help reassure the organization that the team is doing well

Adaptive Leadership

- Organization does not work in incubator but in volatile, fragile, competitive world. It is not possible to deliver results if we do not learn from mistakes, accommodate change request or changed condition. It is impossible to know everything before we start.
- Adaptive leadership pays attention to **value delivery, flexibility, fluidity, cooperation, reprioritization, simplicity, self-organizing, local decision making, innovation, lifelong learning, discuss diverse and divergent views**.

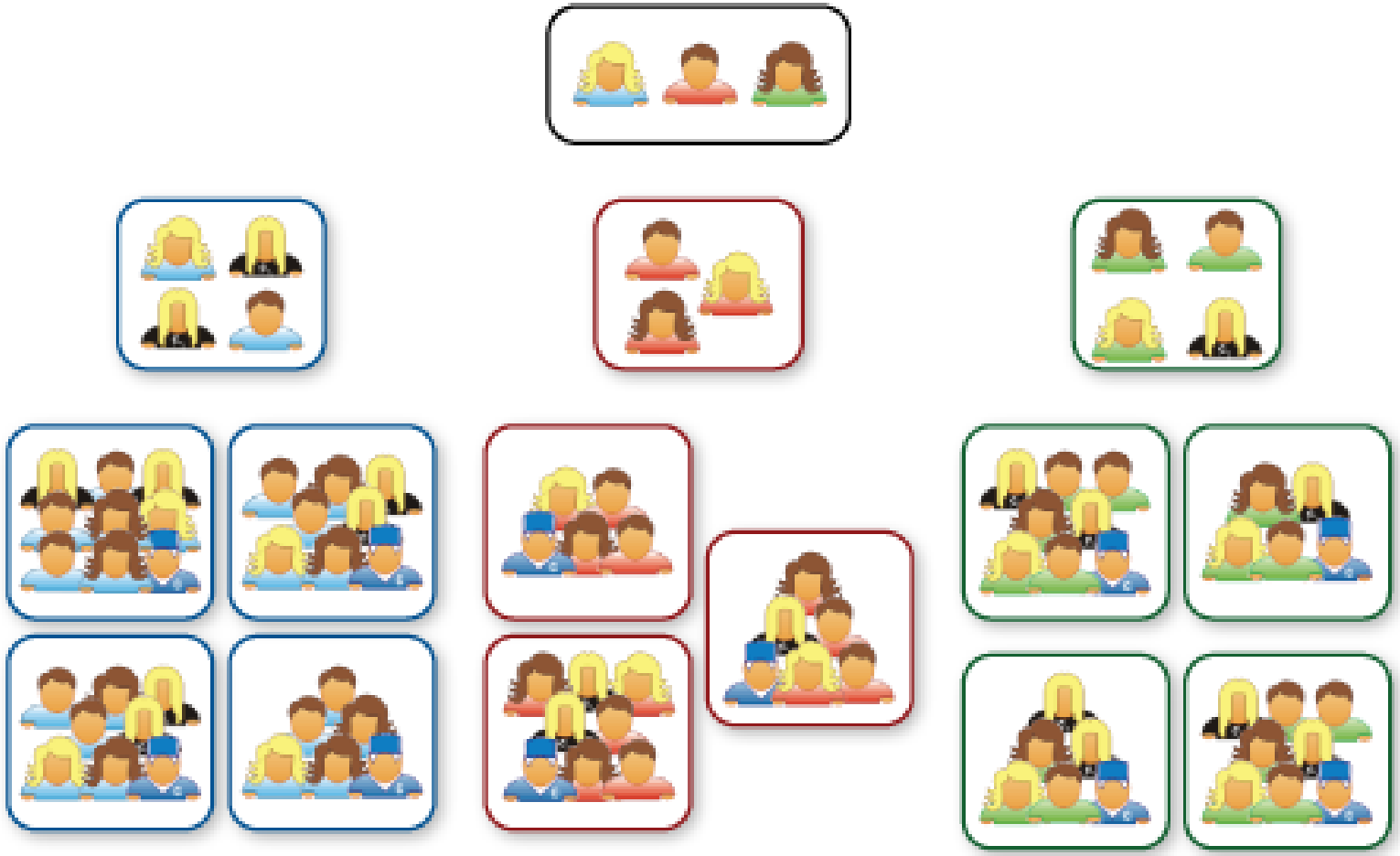
Agile Team

- Agile team is smaller compare to Waterfall team size
- Scrum suggests team size of 7 ± 2. But maximum team size recommended in Agile is 20
- If you need to add more resource than add more experienced people rather adding junior people
- If you further need to add more team member then use agile scaling principle

Agile Team Scaling

- Create feature team (vertical team) do not create skill wise (horizontally).
- Create high level scrum of scrum teams
- Everyday standup meeting is joined by one designated person from each team
- Extra four questions for Scrum of Scrum Daily Standup meeting
  - What has your team done since we last met?
  - What will your team do before we meet again?
  - Is anything slowing your team down or getting in their way?
  - Are you about to put something in another team's way?

Agile Team Scaling



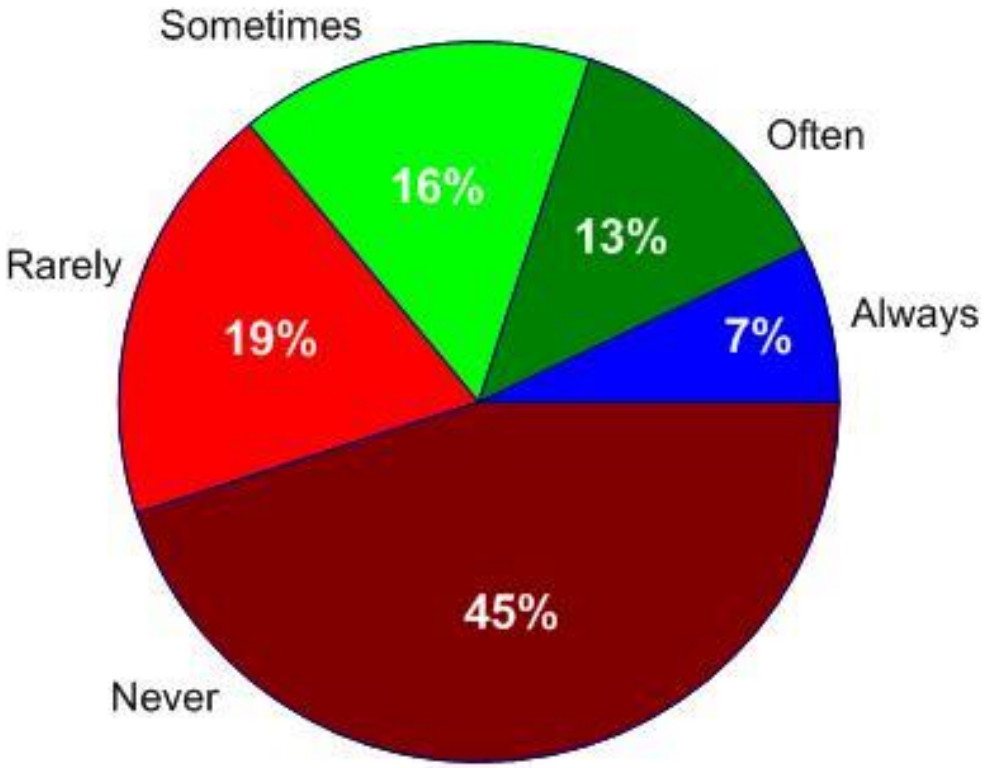
Type of Teams

	Manager lead team	Self-managing team	Self-organizing team	Self-governing team
Setting overall direction for the team				
Designing the team and its context				
Managing work processes and monitoring progress				
Executing the team task				

Value Based Prioritization

Value of Delivered and non-used Features

- Average percentage of delivered functionality actually used when a serial approach to requirements elicitation and documentation is taken on a "successful" information technology project.



Source: Chaos Report v3, Standish Group.

Copyright 2005-2006 Scott W. Ambler

JIT in Value Delivery

Your customer asks you to develop an application with 50 features, out of these 10 are business critical. Application will be ready in 12 months. You are in 3rd week in the project and ask these questions to yourself

- ✓ What is the value of detail plan of this whole application?
- ✓ What is the value of hiring those resources/expertise which you need after 6 month (that too not sure)?
- ✓ What is the value of detail design, SRS, Functional specification (especially when you know they are going to change)?
- ✓ What is the value of writing test cases for complete application?
- ✓ What is the value of hardware, software, infrastructure which you do not need today?

JIT (Just In Time) principle helps controlling our greed and maximizing value.

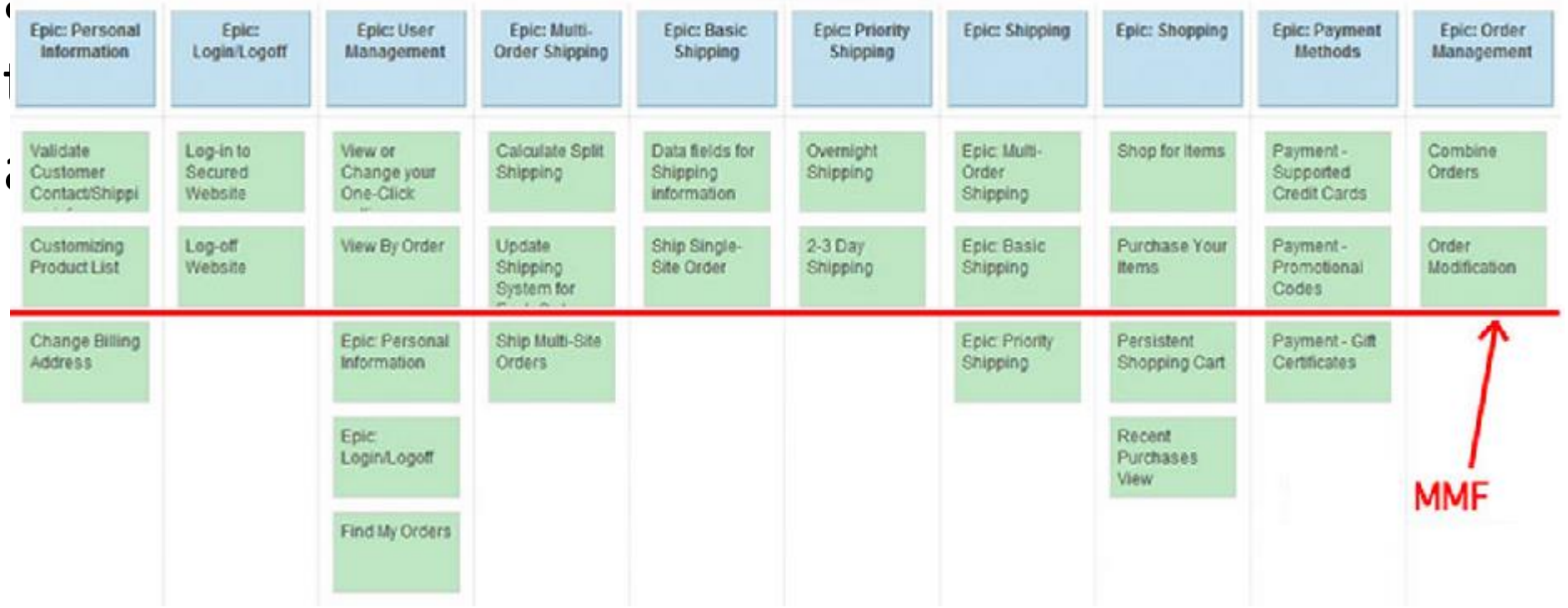
Companies like Toyota, Dell start using this concept early in the industry.

Sources of Revenue

- Incremental Revenue
  - Comes from existing customer
- New Revenue
  - Comes from new customer
- Retained Revenue
  - Comes by enhancing the existing feature. But if you do not enhance features then customer will buy other product.
- Operational Efficiency Revenue
  - Comes by improving operational process

Minimum Marketable Features (MMF)

Product is saleable or usable or viable only when a set of minimum number of features is built



Prioritization

Agile project management is based on principle of **incremental delivery** and **deliver fast**. Thus we must know the prioritization methods so that we can deliver high business values items earlier.

Possible Factors of prioritization are as below. Each factor may have different weightage (business value has highest weightage) when you prioritize your product backlog items.

- Business Value
- Cost of Implementation
- Prioritization Scenario
- Implementation Time
- Frequency of usage
- Safety
- Implementation Difficulty
- Stability
- Re-usability

Relative Prioritization & Ranking Methods

1 Customer Value Prioritization

– Customer based on input from internal stakeholders assigns some priority to each requirement

2 –

Requirements	Stakeholder1	Stakeholder2	Stakheolder3	Total	Ranking
Req#1	1	5	4	10	2
Req#2	2	4	5	11	1
Req#3	3	3	3	9	3
Req#4	4	1	2	7	5
Req#5	5	2	1	8	4

Relative Prioritization & Ranking Methods

3 Analytical Hierarchical Process (AHP)

– Relative prioritization comparing feature pair wise

4 Paired Comparison Techniques

–

Requirements	Req#1	Req#2	Req#3	Req#4	Req#5	Total	Ranking
Req#1	0	1	1	1	1	4	1
Req#2	0	0	1	1	1	3	2
Req#3	0	0	0	1	1	2	3
Req#4	0	0	0	0	1	1	4
Req#5	0	0	0	0	0	0	5

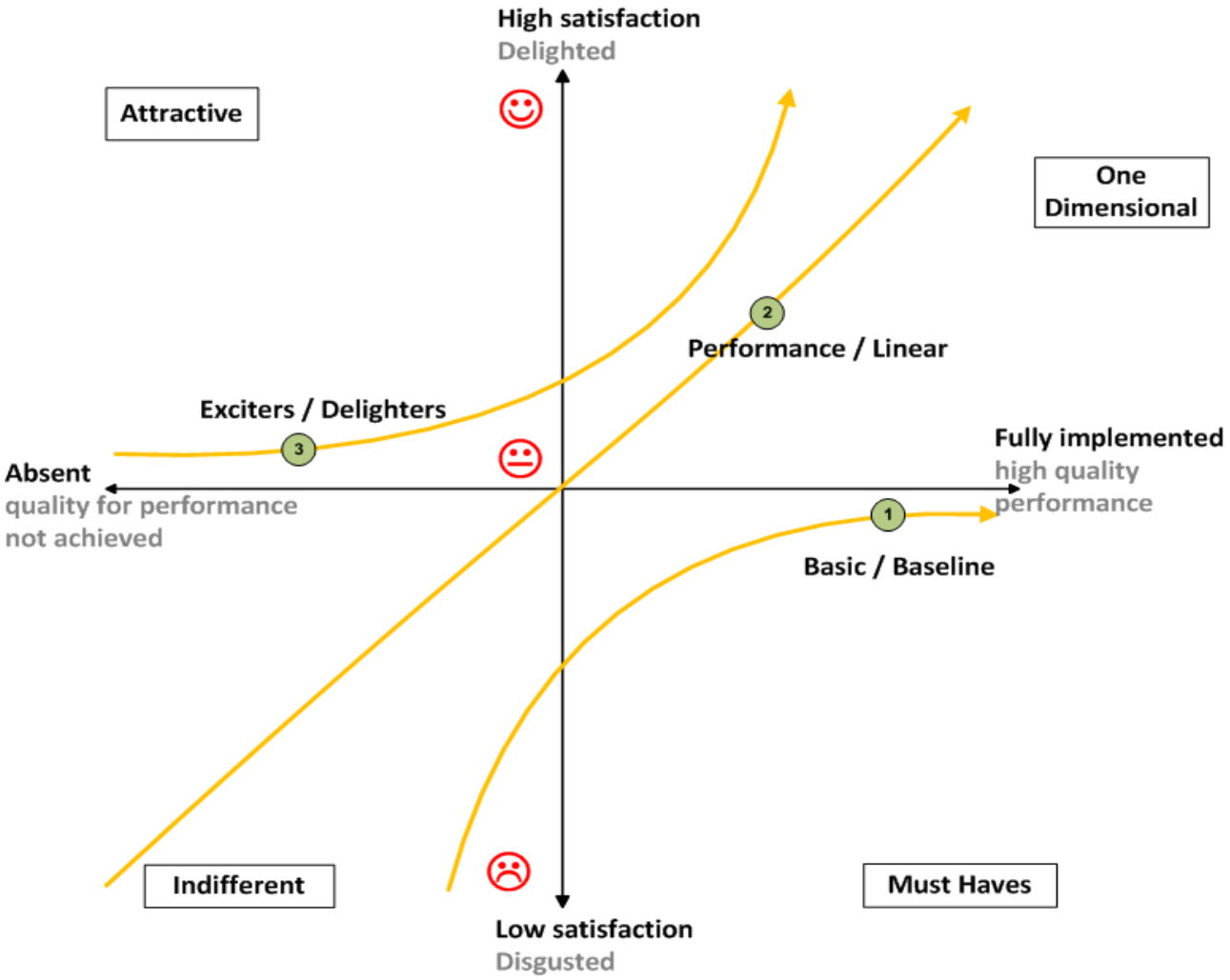
Relative Prioritization & Ranking Methods

5 Weight-based Prioritization Techniques

Requirements	Raw Scoring	Scaled Scoring	Ranking
Req#1	4	.22	2
Req#2	4	.22	2
Req#3	5	.27	1
Req#4	2	.11	4
Req#5	3	.16	3
	18		

Relative Prioritization & Ranking Methods

6 Kano Analysis



Relative Prioritization & Ranking Methods

7 100 Point Method

Each stakeholder is given 100 points and they can use these points to buy the requirements they want

Requirements	Stakeholder1	Stakeholder2	Stakheolder3	Total	Ranking
Req#1	0	20	30	50	3
Req#2	30	10	40	80	2
Req#3	50	20	20	90	1
Req#4	20		10	30	4
Req#5	0	50	0	50	3

Multiple Parameter Based Prioritization

Done by Customer

Done by Customer

Done by Developer

value percentage / (cost percentage * cost weight) + (Risk percentage * Risk weight)

Relative Weight	2	1			1		.5		
Feature	Relative Benefit	Relative Penalty	Total Value	Value %	Relative Cost	Cost %	Relative Risk	Risk %	Priority
1. Query status of a vendor order	5	3	13	8.4	2	4.8	1	3	1.345
2. Generate a Chemical Stockroom inventory report	9	7	25	16.2	5	11.9	3	9.1	0.987
3. See history of a specific chemical container	5	5	15	9.7	3	7.1	2	6.1	0.957
4. Print a chemical safety datasheet	2	1	5	3.2	1	2.4	1	3	0.833
5. Maintain a list of hazardous chemicals	4	9	17	11	4	9.5	4	12.1	0.708
6. Modify a pending chemical request	4	3	11	7.1	3	7.1	2	6.1	0.702
7. Generate an individual laboratory inventory report	6	2	14	9.1	4	9.5	3	9.1	0.646
8. Search vendor catalogs for a specific chemical	9	8	26	16.9	7	16.7	8	24.2	0.586
9. Check training database for hazardous chemical training record	3	4	10	6.5	4	9.5	2	6.1	0.517
10. Import chemical structures from structure drawing tools	7	4	18	11.7	9	21.4	7	21.2	0.365
Totals	**54**	**46**	**154**	**100**	**42**	**100**	**33**	**100**	--

Potential negative impact of not doing the feature

Multiple Parameter Based Prioritization

MoSCoW Prioritization

Product Catalog	Hours	Priority
Product Catalog	100	Must
Product Categories	160	Must
Browse Products by Age	120	Should
Search	100	Must
Product Catalog Admin	80	Could
Product Category Admin	200	Won't
Product Sale Pricing	80	Could
Payment		
Credit Card Payment	160	Must
Paypal	100	Should
CVV2 - Security Code	20	Should
SSL	80	Must
Account		
Save Credit Card	100	Could
Multiple Address Shipments	350	Won't
Order History	120	Could
Wish List Feature	300	Won't
Shopping Cart		
Shopping Cart	100	Must
Gift Wrapping	120	Could
Coupons	120	Could
Tax Calculation	120	Must
Cart Progress Bar	40	Could
Order Maintenance		
Backend System Integration	1000	Won't
Order Data Export	160	Must
Reports and Statistics	1000	Won't
Shipping		
Shipping Rate Lookup	120	Must
Shipping Confirmation E-mail	80	Should
Site Framework		
Page Framework	80	Must
About, Home, Terms, Privacy	80	Must
Estimated Total Project Hours	5090	

	Hours	% of Total Hours
Must Have	1200	56%
Should Have	320	14%
Could Have	660	29%
Tota Hours	2240	
Total Hours/Time box		101%
Must have hours/Time Box		57%

Mo = Must Have

S = Should Have

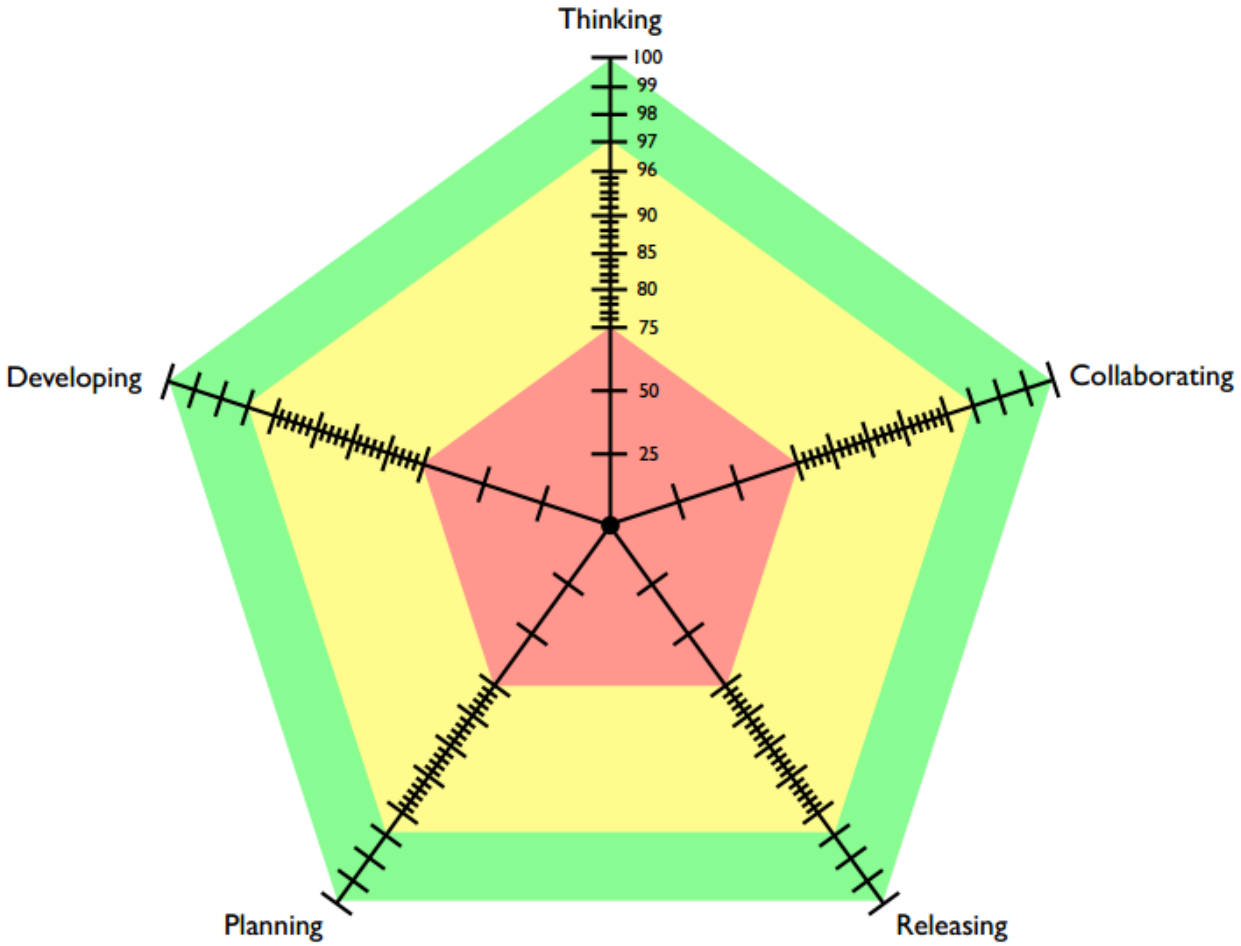
Co = Could Have

W = Wouldn't have now

Regulatory Compliance

- Agile regulatory compliance are taken care by Agile PMO
- Agile PMO conducts training for Agile Teams and ensure that they implement agile project management principles and the same time comply regulatory requirements
- Example of regulatory compliance
  - CMMI
  - ISO
  - Sarbanes Oxley
  - Basel II
  - HIPAA
  - ISO 27001

Agile Self-Assessment Chart



The Art of Agile Development
Self-Assessment Chart
http://www.jamesshore.com/Agile-Book/assess_your_agility.html

Tools & Techniques
Agile Project Risk Management

What is Risk?

- Risk is an event which has probability of happening in future and it can have negative or positive impact on the objective of the project
- Risk itself is not negative or positive. Risk is perceived as negative or positive based on its impact on the project's objectives.

Risk Attitude

- Risk Seeker
  - They look for opportunities to get more return
- Risk Neutral
  - They do not look for opportunity but if something comes on the way and they are getting corresponding returns for this then they take risk
- Risk Averse
  - They are very conscious about the risk and they do not want to take risk.

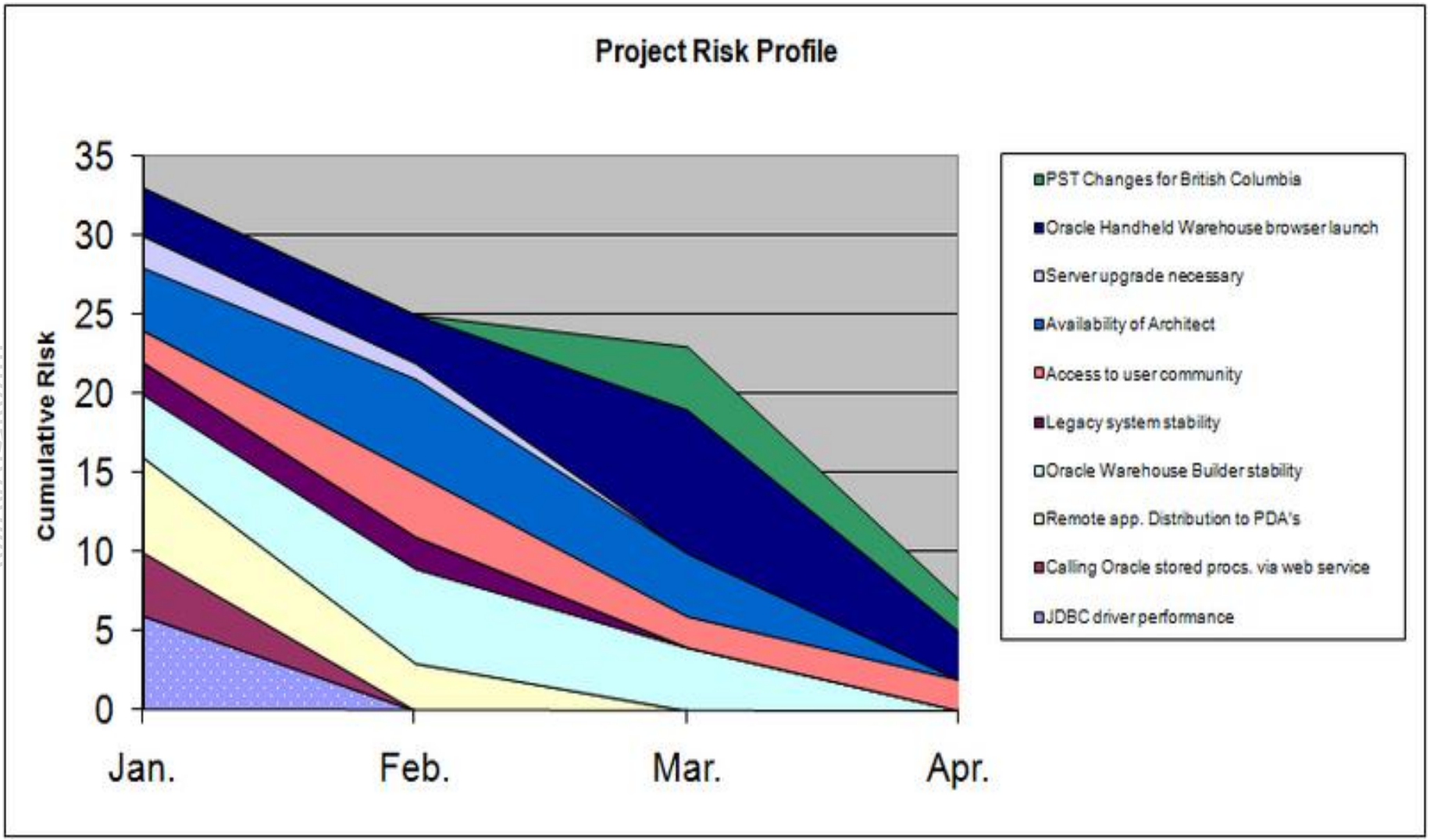
Risk Management Terminologies

- Assumptions, Dependencies, Constraints are **root cause of risk**
- **Sources of risks :** Depending upon business domain of the project its sources may be technology, skills, resource availability, project management practices etc.
- **Proximity of risk** : In future how soon risk is going to occur
- **Risk Area**: What are geographies, department, components, services, server etc. will be affected by the risk
- **Risk Urgency** : If it occurs how soon it will impact the project objective: immediately, in short run, in long run

Plan Risk Management

- Setup Risk Management Framework
- Identify tools & templates
- Conduct awareness training for risk management activities
- Establish responsibilities for Risk Management Activities
- Establish risk reporting formats and frequency
- Determine who will conduct risk audit and when
- Define risk attitude and risk appetite of key stakeholders

Project Risk Profile

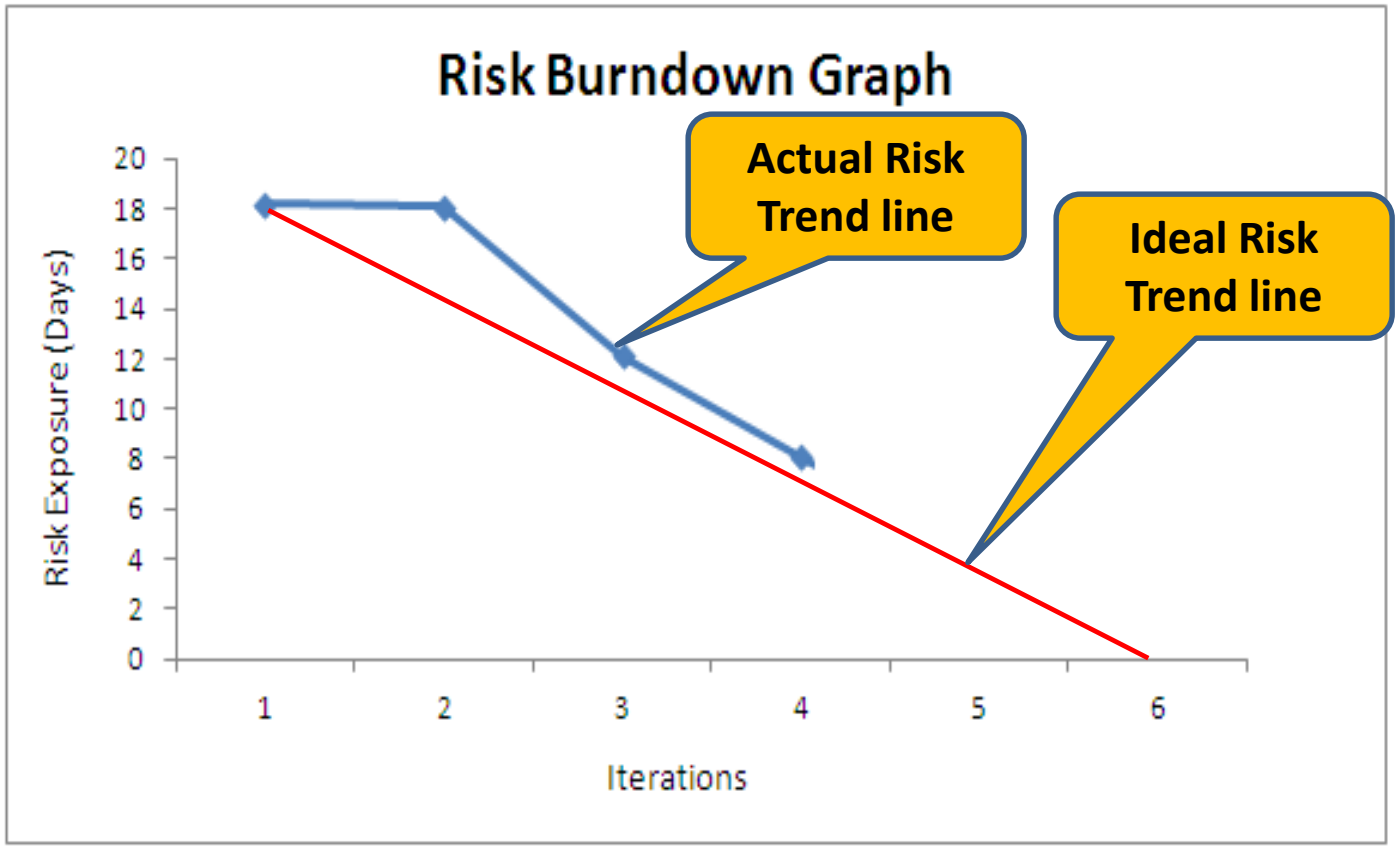


Risk Adjusted Backlog

Risk Score

Risk	Probability	Impact(Days)	Exposure
Internet may be down while working	10%	1	0.1
Lack of experience may take time in implementing functionality	60%	10	6
Backup/store may required additional hardware	80%	5	4
Resource R may not join to support on nn date	50%	8	4
Partner P employee may not be available to validate new features	30%	3	0.9
Component A may not work with the product	40%	8	3.2
Total Risk Impact (Days)			18.2

Risk Burn-down Graph



If actual risk trend line is above than ideal trend line then it means risks are not coming down at the appropriate rate

Identify Risks

- Risk identification is performed for each product backlog items
- Every team member is responsible for identifying risk and updating related data/graph
- Risk Identification techniques
  - Brainstorming Technique
  - Delphi Technique
  - Checklist Analysis

Probability and Impact Matrix

Probability					
VH (0.9)	4.5%	9%	18%	36%	72%
H (0.7)	3.5%	7%	14%	28%	56%
M (0.5)	2.5%	5%	10%	20%	40%
L (0.3)	1.5%	3%	6%	12%	24%
VL (0.1)	0.5%	1%	2%	4%	8%
	VL (0.05)	L (0.1)	M (0.2)	H (0.4)	VH (0.8)
	Impact				

Which risk you would chose to respond or which you can afford to accept depends upon your risk appetite. Your risk appetite for each risk depends upon the exposure of the risk under consideration.

Qualitative Risk Analysis

- Agile project management prefers Qualitative Risk Analysis method for analysis and prioritization of the risk
- This is the quick and simplest way of risk analysis
- In this method you need to determine probability and impact. Exposure can be calculated by multiplying probability and impact. This step is called probability and impact assessment.
- Rank risks based on their exposure and pick up those top risks which you cannot afford to forego
- Even if you have 50 risks in your risk register do not choose to work upon more than 10 on any point of time.

Risk Categorization

- Risk can categorized based on
  - Ranking of risk
  - Cause of risk
  - Response type
  - Priority of risk
  - Any other specific trend

Addressing an Unknown

In the product backlog sometimes agile team may find some epic/feature/user story which they are not able to understand. Therefore it is difficult to estimate those epic/features/ stories. Keeping the stories in the product backlog as estimated and in priority order is essential, as it gives the product owner insight into approximately which features will be completed by a certain date. There are 2 ways of solving this problem

- Spike Solution
- Tracer Bullet Solution

Addressing an Unknown

- A Spike Solution
  - Spikes allows a team to set some amount of time to research and explore some unknown factor about an upcoming story.
  - Create an experimental solution that cuts through all the "layer". The code is thrown away once solution is verified. No time is estimated for this kind of solutions but typically 1-2 days time is allowed to create a spike solution. Spike solution can be developed using A technology and actual implementation can be done using B technology.
- Tracer Bullet
  - Create an experimental solution that cuts through all the "layer". The code is **not** thrown away once solution is verified but it is extended. This is not time-boxed solution and final implementation should be in the same language in which tracer bullet solution was developed.

Risk-based Spikes

- Spike is a technical investigation to research an answer to a problem
- Unlike Technical spike risk-based spikes gets the answer through analysis. Technical spike depends upon technical research and feasibility study.
- Using Risk-based spikes there are four ways to know the answer of any complex problem
  - Sensitivity Analysis
  - Decision Tree Analysis
  - Expected Monetary Value (EVM) Analysis
  - Simulation

A Glance on Risk Response Plan

Threats or Negative Risk	Opportunities or Positive Risk
Avoid	Exploit
Transfer	Share
Mitigate	Enhance
Accept	Accept

Monitoring & Controlling Risk

Tools and techniques to monitor and control risks

- Risk Audit
- Secondary Risk
- Residual Risk
- Workarounds
- Reserve Analysis

Understanding Reserves

<u>Contingency reserves</u> : Known – Unknown

–It is designed for use only if certain events occur or only under certain predefined conditions, provided there is sufficient warning to implement the response.

–Examples of events that may trigger the contingency response include missing intermediate milestones or gaining higher priority with a supplier.

–Events triggering the contingency response should be triggered and tracked.

<u>Management reserves:</u> Unknown – Unknown

It is defined for use only if 'the events that occur or only under certain conditions', where information about the event & its occurrence is absolutely NOT available.

Scrum Methodology

Scrum Values

Scrum Roles

Scrum Ceremonies

Scrum Artifacts


hari.prasad@vedavit-ps.com


+91 9535999336


http://rarementors.com


https://in.linkedin.com/in/harithapliyal


@harithapliyal


http://pmlogy.com

Hari P Thapliyal,

PMP, PMI-ACP, MCITP, Princ2 Practitioner, CSM, MBA, MCA, CIC, PGDFM

PMO Architect & Project Management Evangelist

Profile: http://in.linkedin.com/in/harithapliyal

For content related queries please contact me via hari.Prasad@vedavit-ps.com