AGILE METHODOLOGY

- PRESENTED TO:
- MR. MANESSH KUMAR
- MRS. DEEPTI GUPTA

- PRESENTED BY:
- MOHIT KUMAR
- 1313310092
- CSE

Contents of the table

- Definition
- Manifesto
- Principles
- Characteristics
- Overview
- Philosophy
- Agile methods
- Agile practices
- Experience and Adoption
- Pitfalls
- Criticism
- Application outside software development

What is Agile software development?

Agile software development is a set of principles for software development in which requirements and solutions evolve through collaboration between self-organizing,cross-functional teams.

Agile Manifesto

- *Individuals and interactions*: self-organization and motivation are important, as are interactions like co-location and pair programming.
- *Working software*: working software is more useful and welcome than just presenting documents to clients in meetings.
- *Customer collaboration*: requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- *Responding to change*: agile methods are focused on quick responses to change and continuous development.

Principles

The Agile Manifesto is based on twelve principles:

- Customer satisfaction by early and continuous delivery of valuable software
- Welcome changing requirements, even in late development
- Working software is delivered frequently (weeks rather than months)
- Close, daily cooperation between business people and developers
- Projects are built around motivated individuals, who should be trusted.

Principle continued..

- Face-to-face conversation is the best form of communication (co-location)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Best architectures, requirements, and designs emerge from self-organizing teams
- Regularly, the team reflects on how to become more effective, and adjusts accordingly

Characteristics of Agile Software Development

-- Light Weighted methodology
-- Small to medium sized teams
-- vague and/or changing requirements
-- vague and/or changing techniques
-- Simple design
-- Minimal system into production

Overview

- There are many specific agile development methods. Most promote development, teamwork, collaboration, and process adaptability throughout the life-cycle of the project.

Philosophy

- Compared to traditional software engineering, agile software development mainly targets complex systems and projects with dynamic, non-deterministic and non-linear characteristics, where accurate estimates, stable plans, and predictions are often hard to get in early stages—and big up-front designs and arrangements would probably cause a lot of waste, i.e., are not economically sound.

Adaptive vs. predictive

- Agile methods lie on the *adaptive* side of this continuum. One key of adaptive development methods is a "Rolling Wave" approach to schedule planning, which identifies milestones but leaves flexibility in the path to reach them, and also allows for the milestones themselves to change.
- Adaptive methods focus on adapting quickly to changing realities. When the needs of a project change, an adaptive team changes as well.

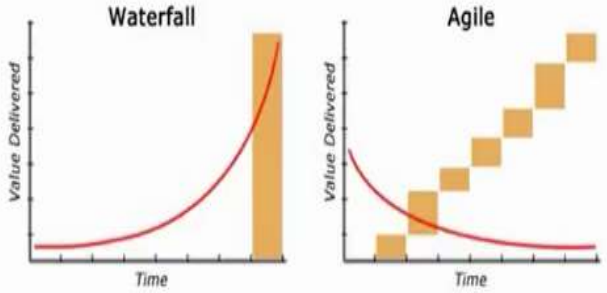
Predictive

- *Predictive* methods, focus on analysing and planning the future in detail and cater for known risks.
- In the extremes, a predictive team can report exactly what features and tasks are planned for the entire length of the development process.
- Predictive methods rely on effective early phase analysis and if this goes very wrong, the project may have difficulty changing direction.
- Predictive teams often institute a change control board to ensure they consider only the most valuable changes.

Iterative vs Waterfall

- In the waterfall model there is always a separate *testing phase* after a *build phase*.
- However, in agile development testing is usually done concurrently with, or at least in the same iteration as, programming.
- Because testing is done in every iteration—which develops a small piece of the software—users can frequently use those new pieces of software and validate the value.

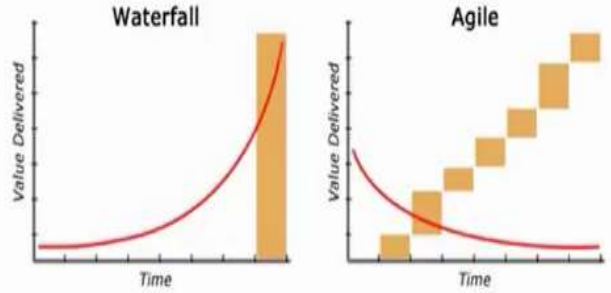
WATERFALL DEMISES



Serialized Process

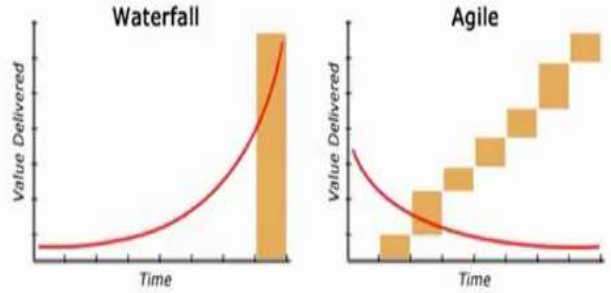
A previous phase has to be completed before starting the next phase.

WATERFALL DEMISES



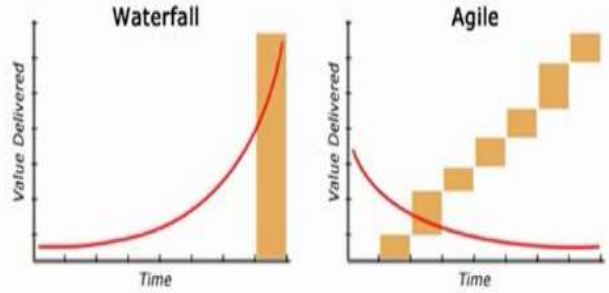
Serialized Process	A previous phase has to be completed before starting the next phase.
Planning far in advance	Products no longer match reality by the time they are released.

WATERFALL DEMISES



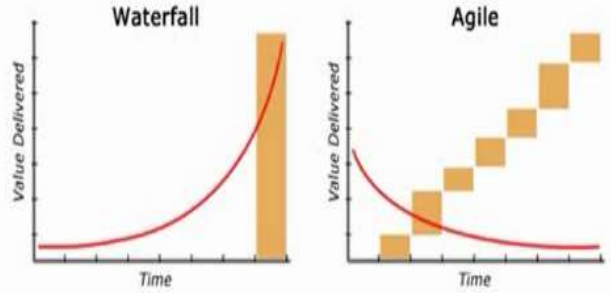
Serialized Process	A previous phase has to be completed before starting the next phase.
Planning far in advance	Products no longer match reality by the time they are released.
Lack of Visibility	Teams don't realize they are behind schedule.

WATERFALL DEMISES



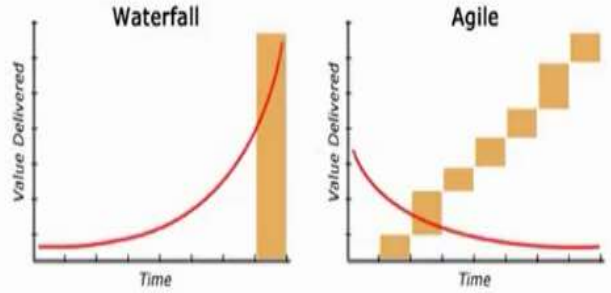
Serialized Process	A previous phase has to be completed before starting the next phase.
Planning far in advance	Products no longer match reality by the time they are released.
Lack of Visibility	Teams don't realize they are behind schedule.
Project Timeline	Planned at the start. If one phase is delayed all other phases are also delayed.

WATERFALL DEMISES



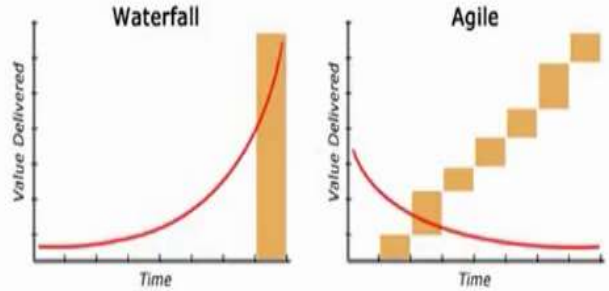
Serialized Process	A previous phase has to be completed before starting the next phase.
Planning far in advance	Products no longer match reality by the time they are released.
Lack of Visibility	Teams don't realize they are behind schedule.
Project Timeline	Planned at the start. If one phase is delayed all other phases are also delayed.
Static Requirements	Requirements cannot be c middle.

AGILE'S RISE



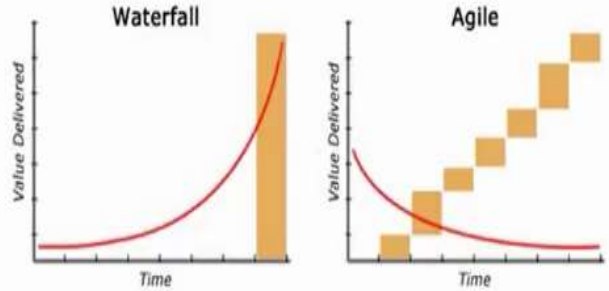
Serialized Process	Iterative approach with tasks broken into small increments.

AGILE'S RISE



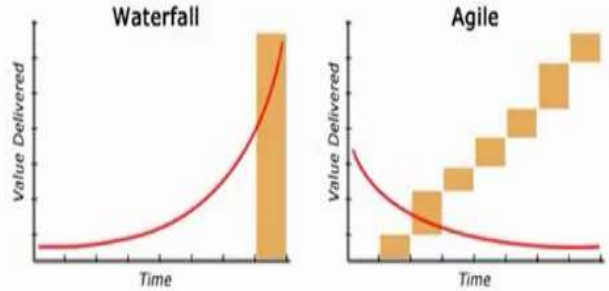
Serialized Process	Iterative approach with tasks broken into small increments.
Planning far in advance	Plan for what we know and we have left sufficient allowance in our plans for what we don't know.

AGILE'S RISE



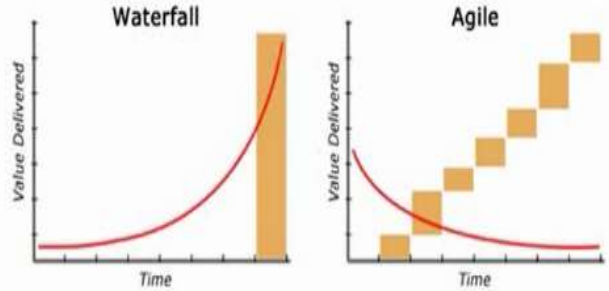
Serialized Process	Iterative approach with tasks broken into small increments.
Planning far in advance	Plan for what we know and we have left sufficient allowance in our plans for what we don't know.
Lack of Visibility	Teams don't realize they are behind schedule

AGILE'S RISE



Serialized Process	Iterative approach with tasks broken into small increments.
Planning far in advance	Plan for what we know and we have left sufficient allowance in our plans for what we don't know.
Lack of Visibility	Teams don't realize they are behind schedule
Project Timeline	Allows the development effort to get feedback from the customer throughout.

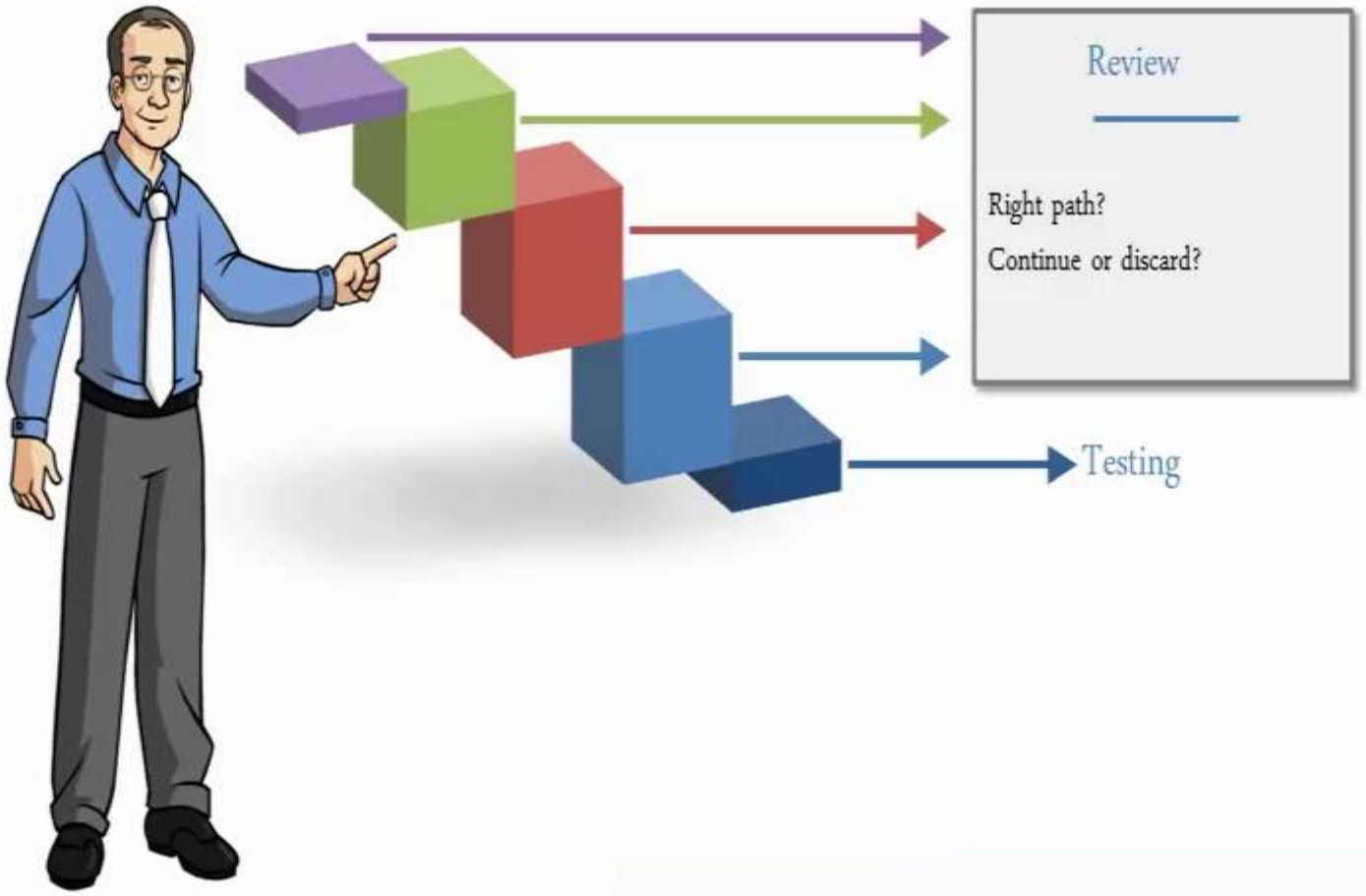
AGILE'S RISE



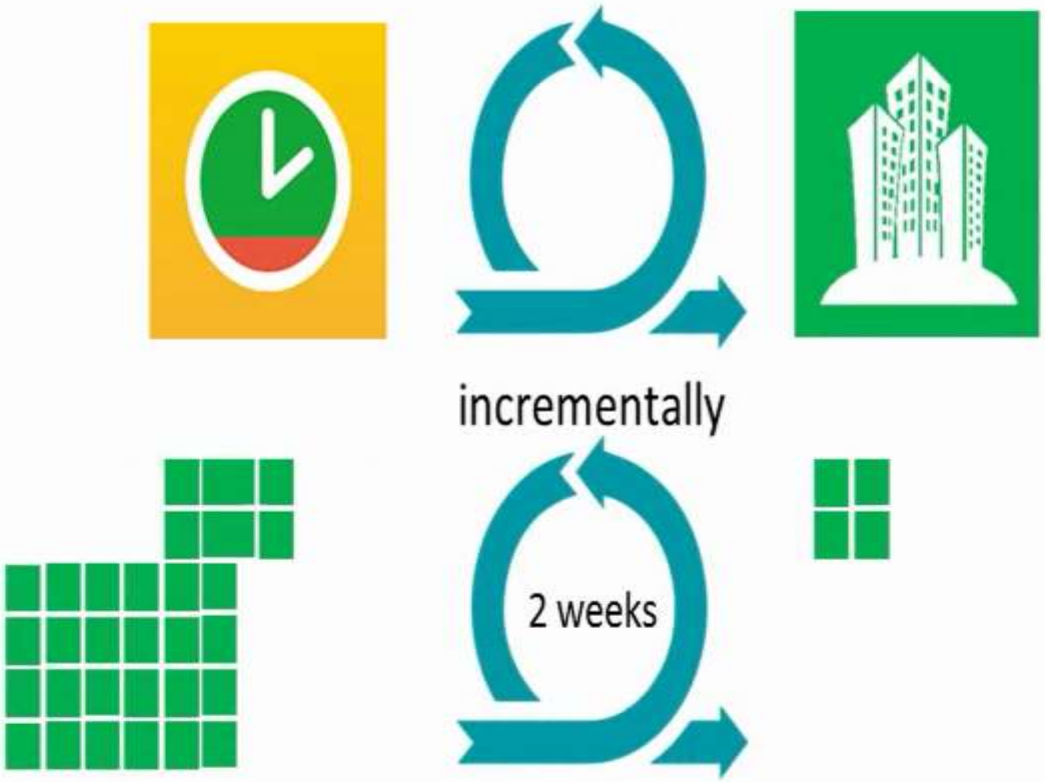
Serialized Process	Iterative approach with tasks broken into small increments.
Planning far in advance	Plan for what we know and we have left sufficient allowance in our plans for what we don't know.
Lack of Visibility	Teams don't realize they are behind schedule
Project Timeline	Allows the development effort to get feedback from the customer throughout.
Static Requirements	Scope is never closed; Con requirement priorities by the business.

HOW WATERFALL PROJECT WORKS

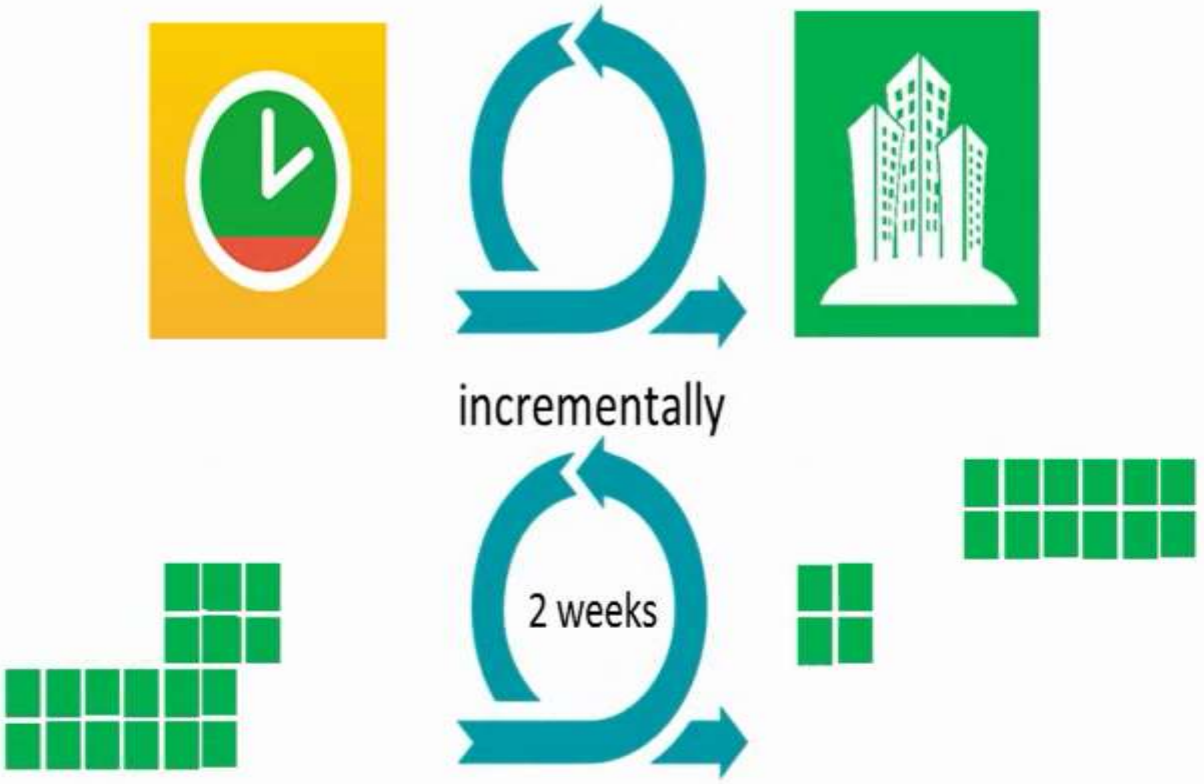
Used for Small Projects



WHAT IS AGILE?

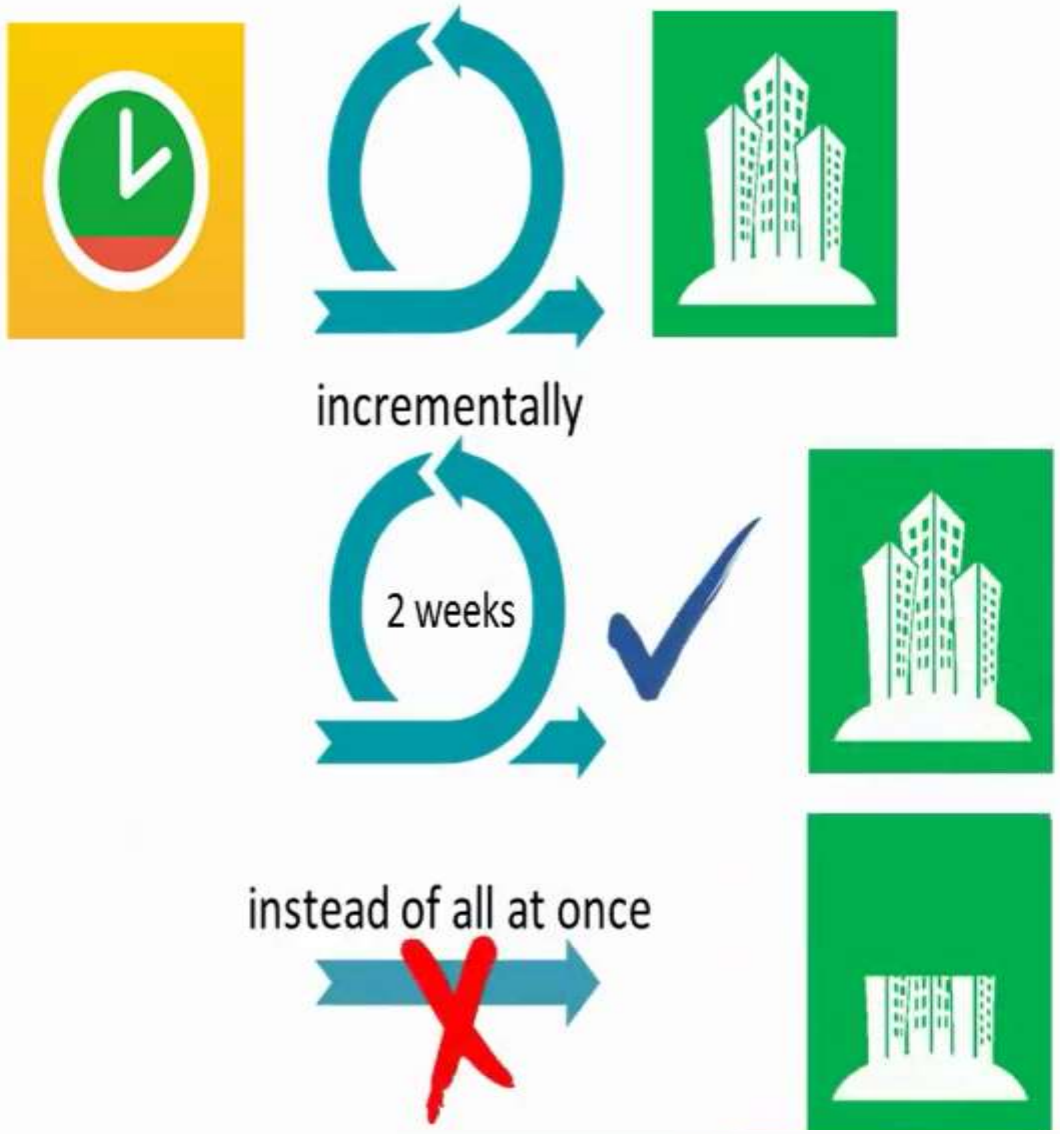


WHAT IS AGILE?



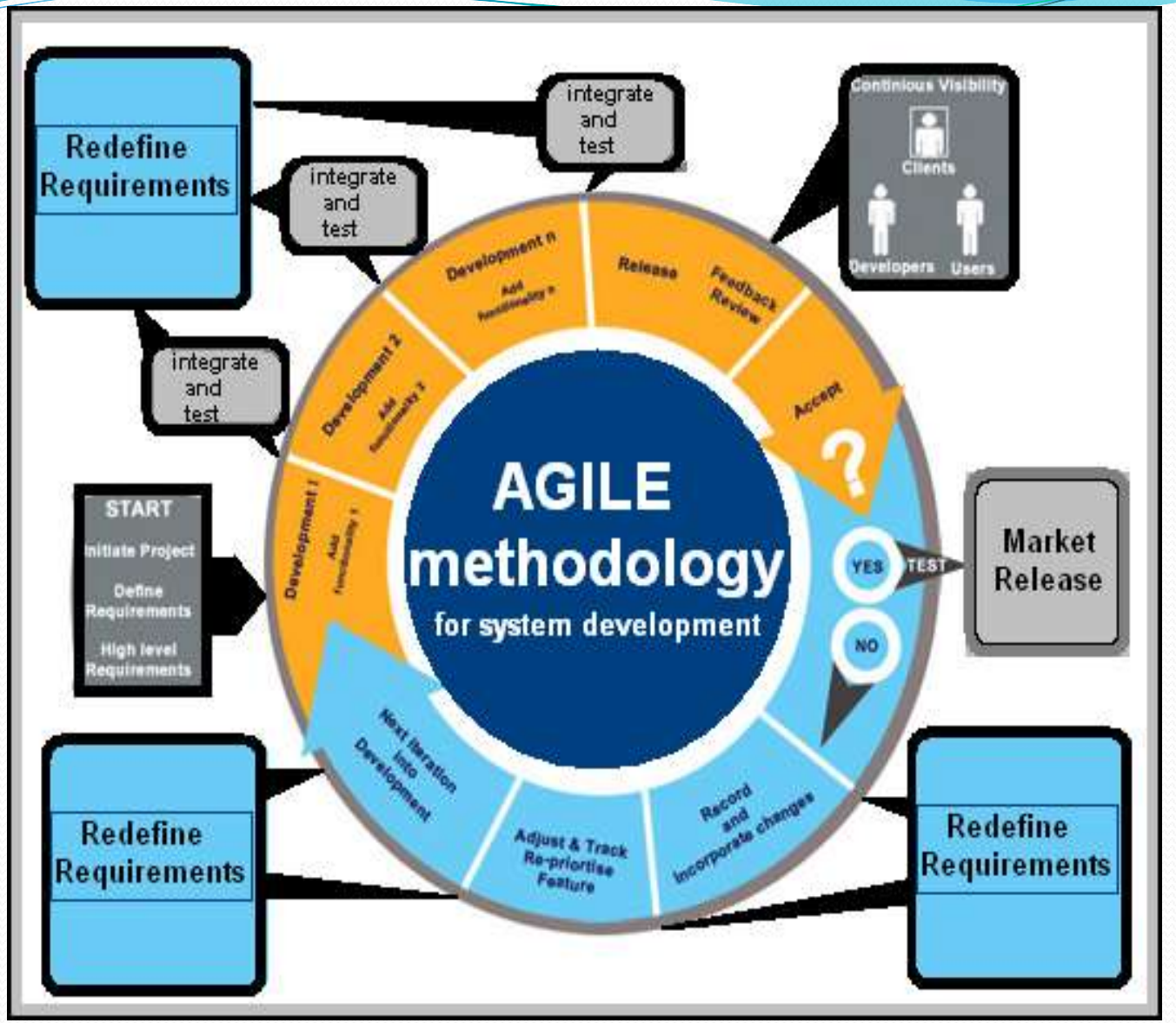
WHAT IS AGILE?

WHAT IS AGILE?



Agile Method

- Adaptive software development (ASD)
- Agile modeling
- Agile Unified Process (AUP)
- Business analyst designer method (BADM)
- Crystal Clear Methods
- Disciplined agile delivery
- Dynamic systems development method (DSDM)
- Extreme programming (XP)
- Feature-driven development (FDD)
- Lean software development


Redefine Requirements
integrate and test
integrate and test
integrate and test
Continious Visibility
Clients
Developers
Users
Development n
Release
Feedback Review
Accept
Development 2
Development 1
START
Initiate Project
Define Requirements
High level Requirements
AGILE methodology
for system development
YES
TEST
NO
Market Release
Next iteration into Development
Adjust & Track Re-prioritise Feature
Record and incorporate changes
Redefine Requirements
Redefine Requirements

Agile Practices

- Acceptance test-driven development (ATDD)
- Agile modeling
- Backlogs (Product and Sprint)
- Behavior-driven development (BDD)
- Cross-functional team
- Continuous integration (CI)
- Domain-driven design (DDD)
- Pair programming
- Planning poker
- Refactoring

Experience and Adoption

- Agile methods were first used by technology early adopters such as Tektronix.
- Agile can be used with any programming paradigm or language in practice Agile has usually been closely associated with object-oriented environments such as Smalltalk and Lisp and later Java.
- The initial adopters of Agile methods were usually small to medium-sized teams working on unprecedented systems with requirements that were difficult to finalize and likely to change as the system was being developed.

Common Pitfalls

- **Lack of overall project design**
- **Adding stories to a sprint in progress**
- **Lack of sponsor support**
- **Insufficient training**
- **Product owner role is not properly filled**
- **Excessive preparation/planning**
- **Lacking test automation**

Criticism

- *The agile movement is in some ways a bit like a teenager: very self-conscious, checking constantly its appearance in a mirror, accepting few criticisms, only interested in being with its peers, rejecting en bloc all wisdom from the past, just because it is from the past, adopting fads and new jargon, at times cocky and arrogant. But I have no doubts that it will mature further, become more open to the outside world, more reflective, and also therefore more effective.*

Aplications outside s/w development

- Integrated customer engagement - to embed customers within any delivery process to share accountability for product/service delivery.
- Facilitation-based management - adopting agile management models, like the role of Scrum Master to facilitate the day-to-day operation of teams.
- An enabling organisational structure - with a focus on staff engagement,personal autonomy and outcomes based governance.

THANKS