SDLC

Sujata Batra

Sujata Batra - sujatabatra@hotmail.com

- IT Trainer Since 2000
- More than 50+ Corporate Clients

Sun Life	BOEING	accenture
Sapient	amdocs	TATA CONSULTANCY SERVICES
ICAS Practice Community Partner Royal Bank of Scotland	ERICSSON	WIPRO Applying Thought
HCL	YAMAHA	NEC

SujataBatra

SDLC

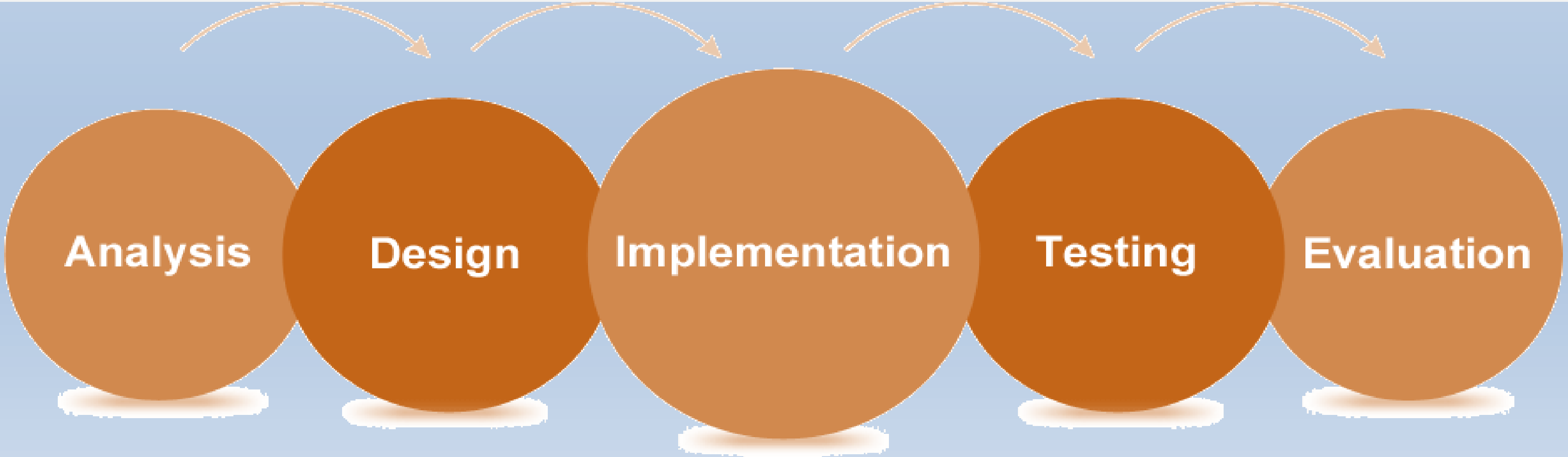
The systems development life cycle (**SDLC**) is a term used in:

Systems Engineering

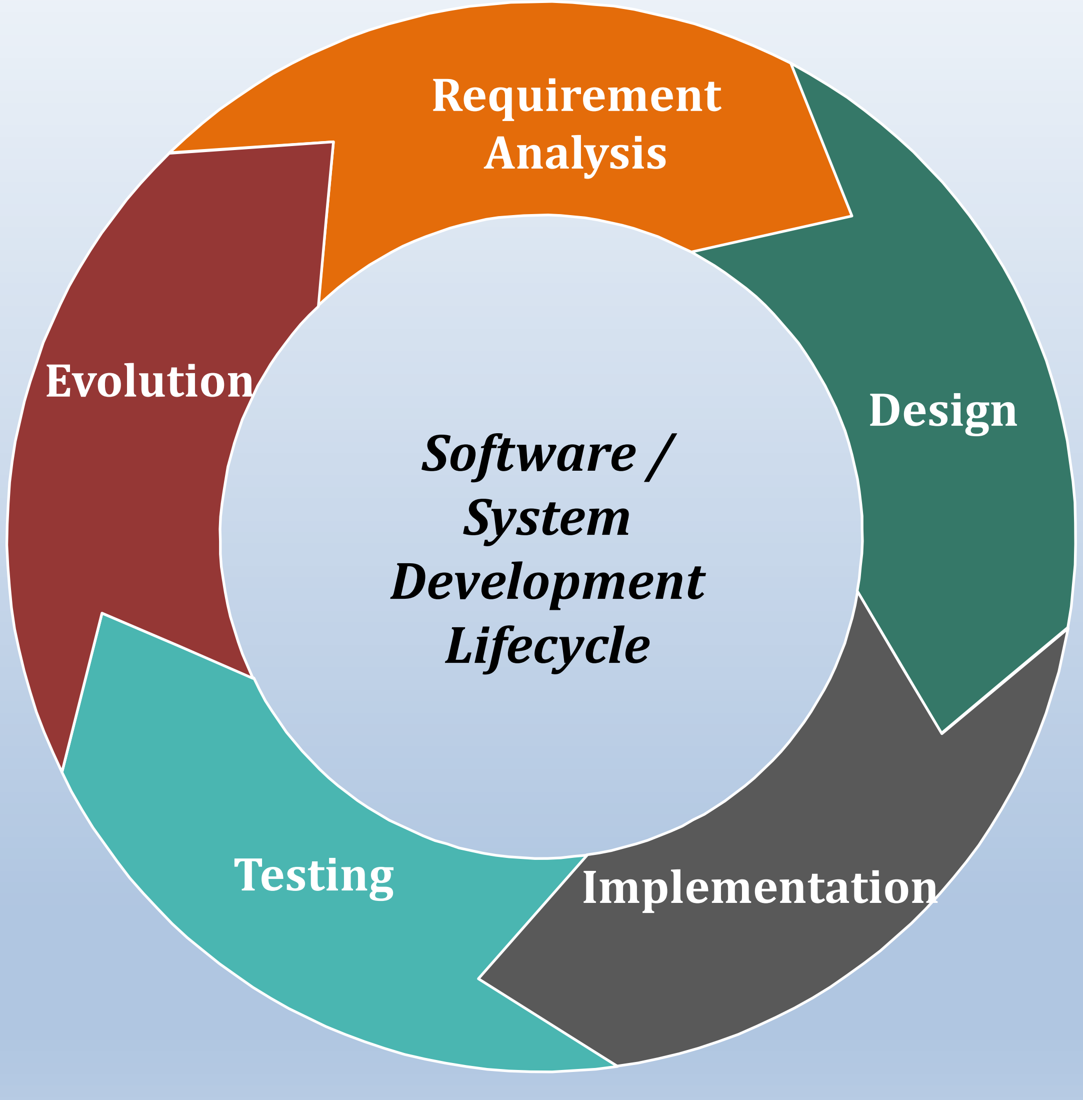
Information Systems

Software Engineering

Also called application development life-cycle.

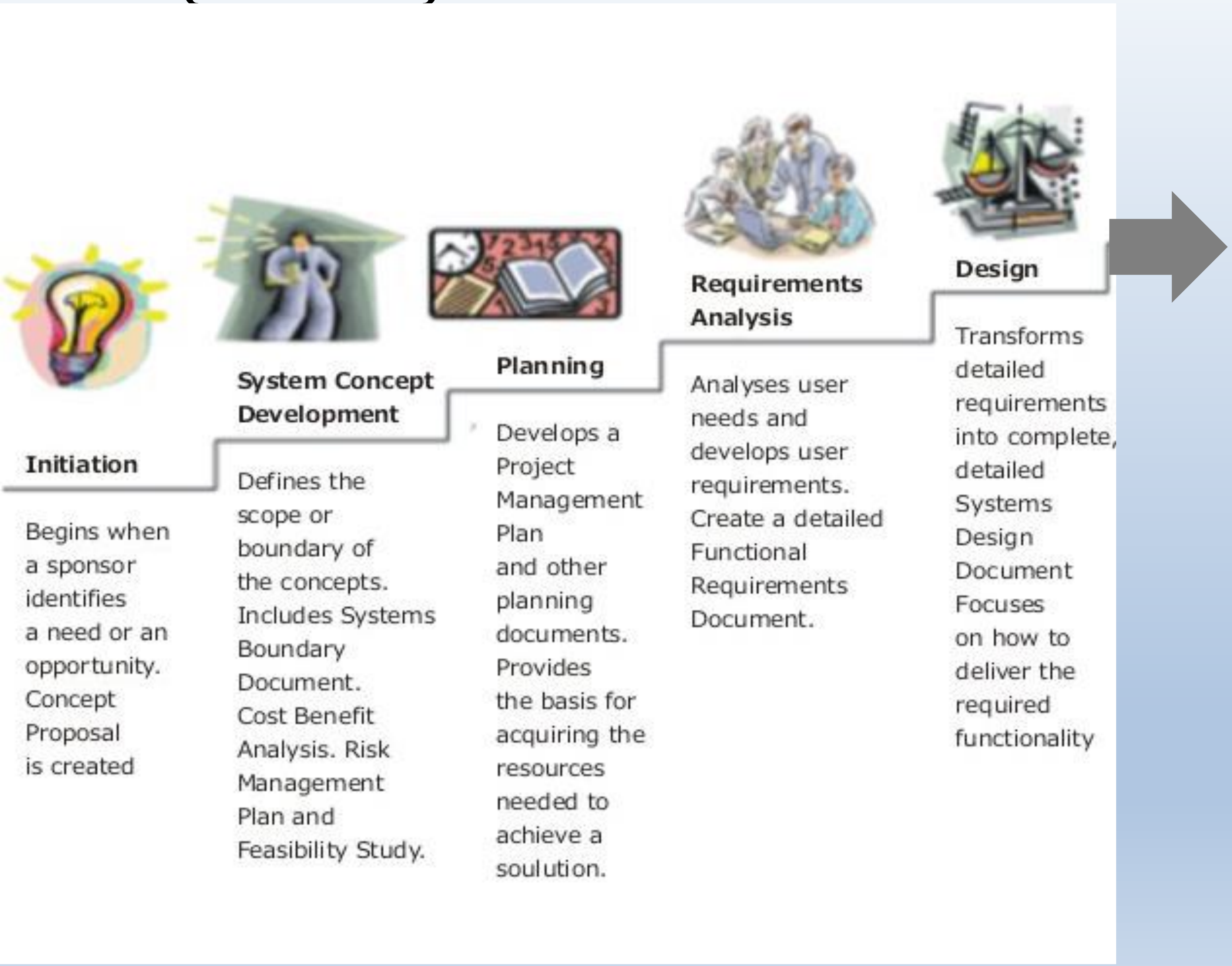


SDLC



SDLC is the motherboard for all kinds of project developments!

SDLC Phases (Part 1 of 2)



Initiation

Begins when a sponsor identifies a need or an opportunity. Concept Proposal is created

System Concept Development

Defines the scope or boundary of the concepts. Includes Systems Boundary Document. Cost Benefit Analysis. Risk Management Plan and Feasibility Study.

Planning

Develops a Project Management Plan and other planning documents. Provides the basis for acquiring the resources needed to achieve a soulution.

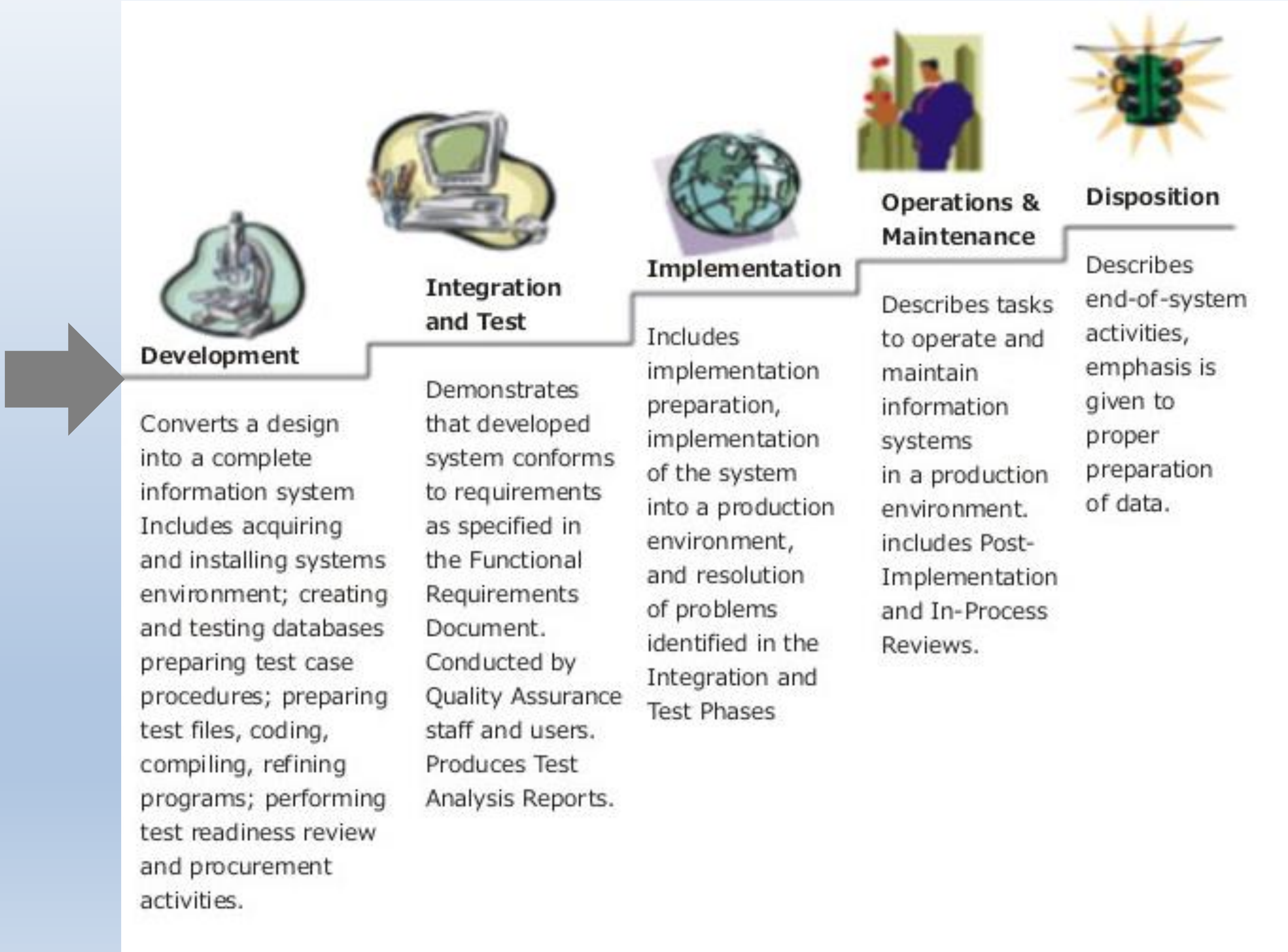
Requirements Analysis

Analyses user needs and develops user requirements. Create a detailed Functional Requirements Document.

Design

Transforms detailed requirements into complete, detailed Systems Design Document Focuses on how to deliver the required functionality

SDLC Phases (Part 2 of 2)



Development

Converts a design into a complete information system Includes acquiring and installing systems environment; creating and testing databases preparing test case procedures; preparing test files, coding, compiling, refining programs; performing test readiness review and procurement activities.

Integration and Test

Demonstrates that developed system conforms to requirements as specified in the Functional Requirements Document. Conducted by Quality Assurance staff and users. Produces Test Analysis Reports.

Implementation

Includes implementation preparation, implementation of the system into a production environment, and resolution of problems identified in the Integration and Test Phases

Operations & Maintenance

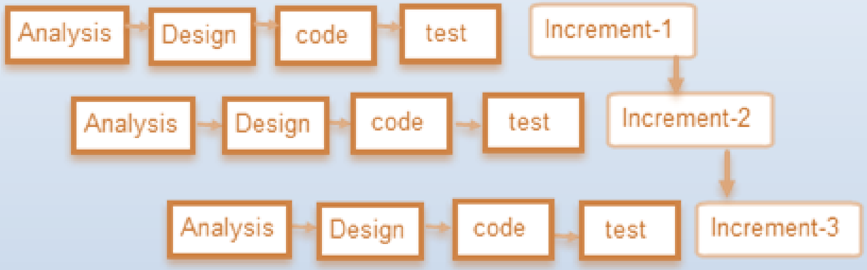
Describes tasks to operate and maintain information systems in a production environment. includes Post-Implementation and In-Process Reviews.

Disposition

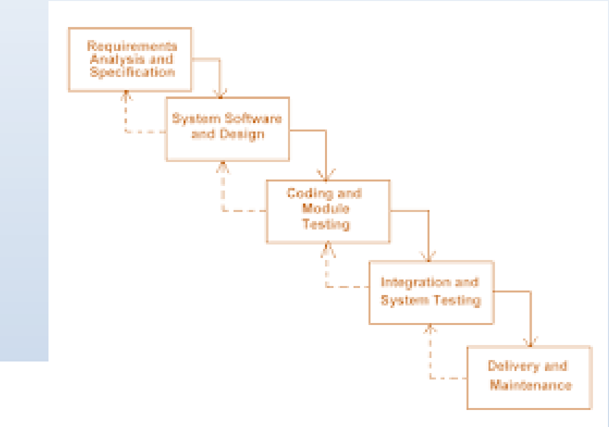
Describes end-of-system activities, emphasis is given to proper preparation of data.

Types of SDLC

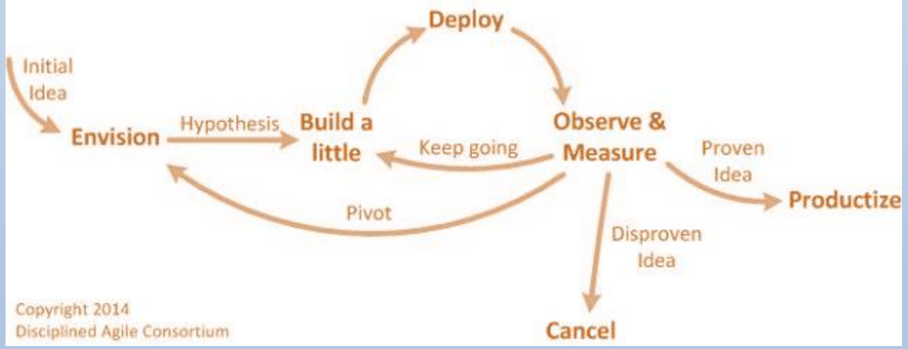
INCREMENTAL



WATERFALL

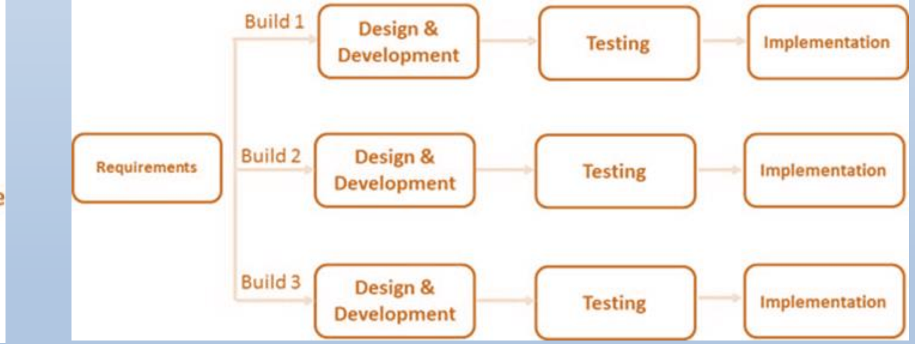

Requirements Analysis and Specification
System Software and Design
Coding and Module Testing
Integration and System Testing
Delivery and Maintenance

AGILE



Copyright 2014
Disciplined Agile Consortium

ITERATIVE

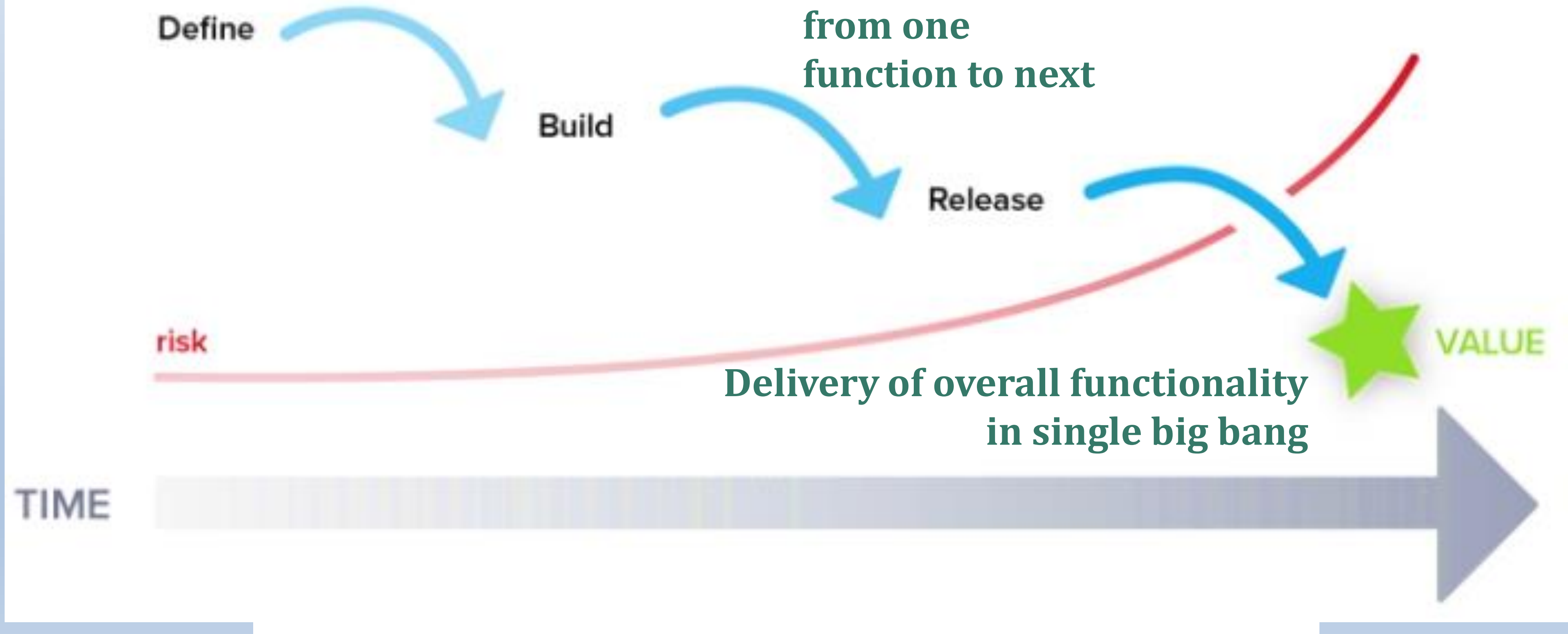


Each kind of SDLC has its individual specificity tailored to situations where it may be necessitated.

S
u
j
a
t
a
B
a
t
r
a

Waterfall Model

Sequential phase driven approach.



Focus on completing one step before moving to next step

Formal handover from one function to next

Delivery of overall functionality in single big bang

Strengths of Waterfall Model

Easy to understand, easy to use

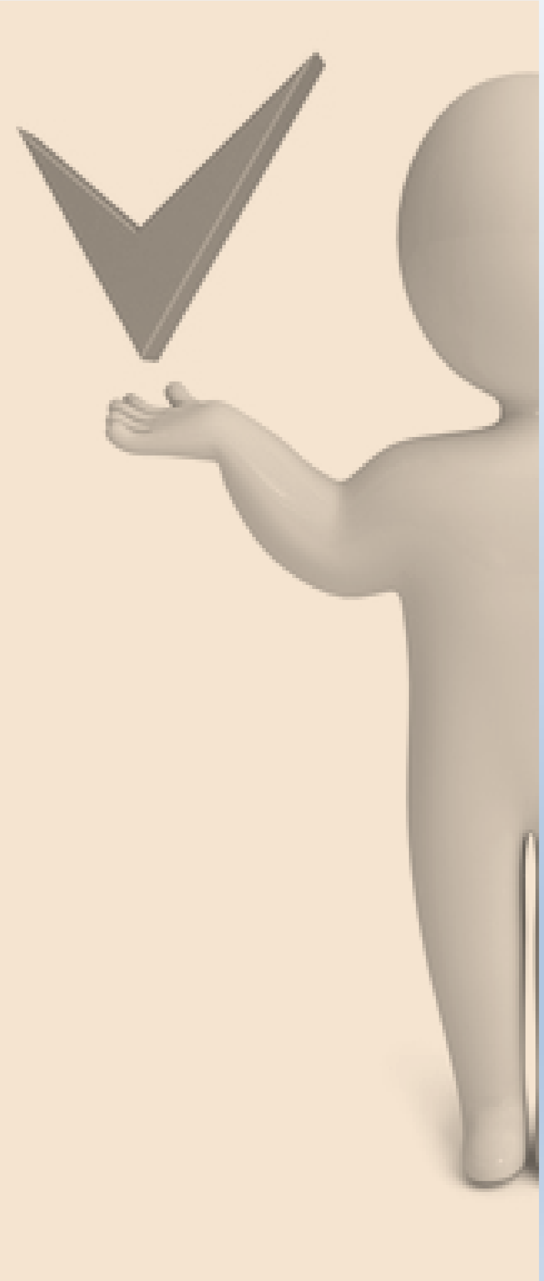
Provides structure to inexperienced staff

Milestones are well understood

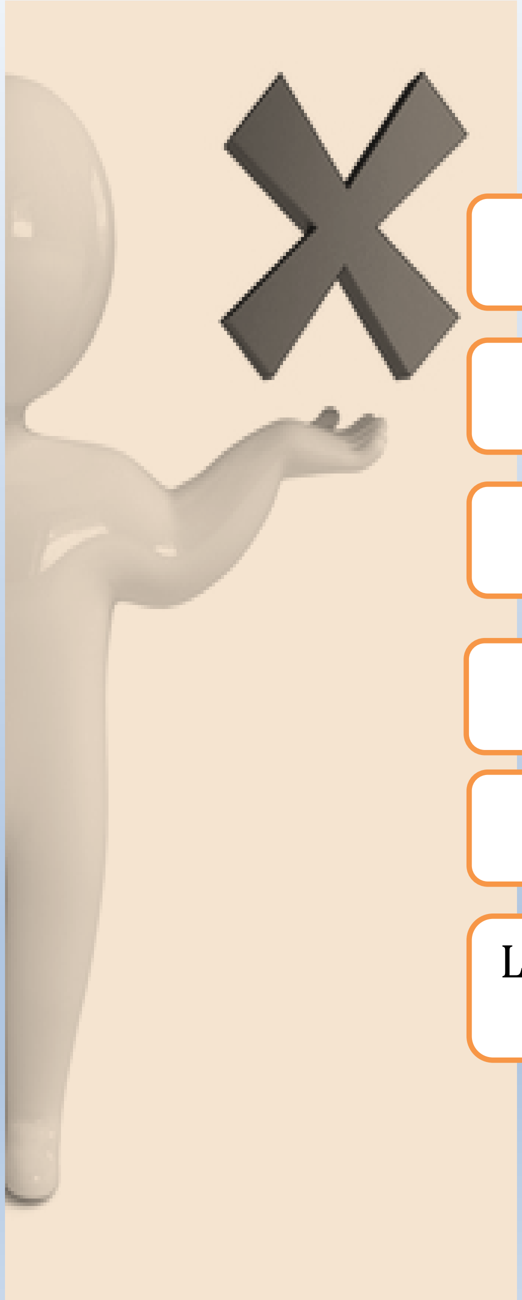
Sets requirements stability

Good for management control (plan, staff, track)

Works well when quality is more important
Than cost or schedule



Weaknesses of Waterfall Model



All requirements must be known upfront

Deliverables created for each phase are considered frozen – inhibits flexibility

Can give a false impression of progress

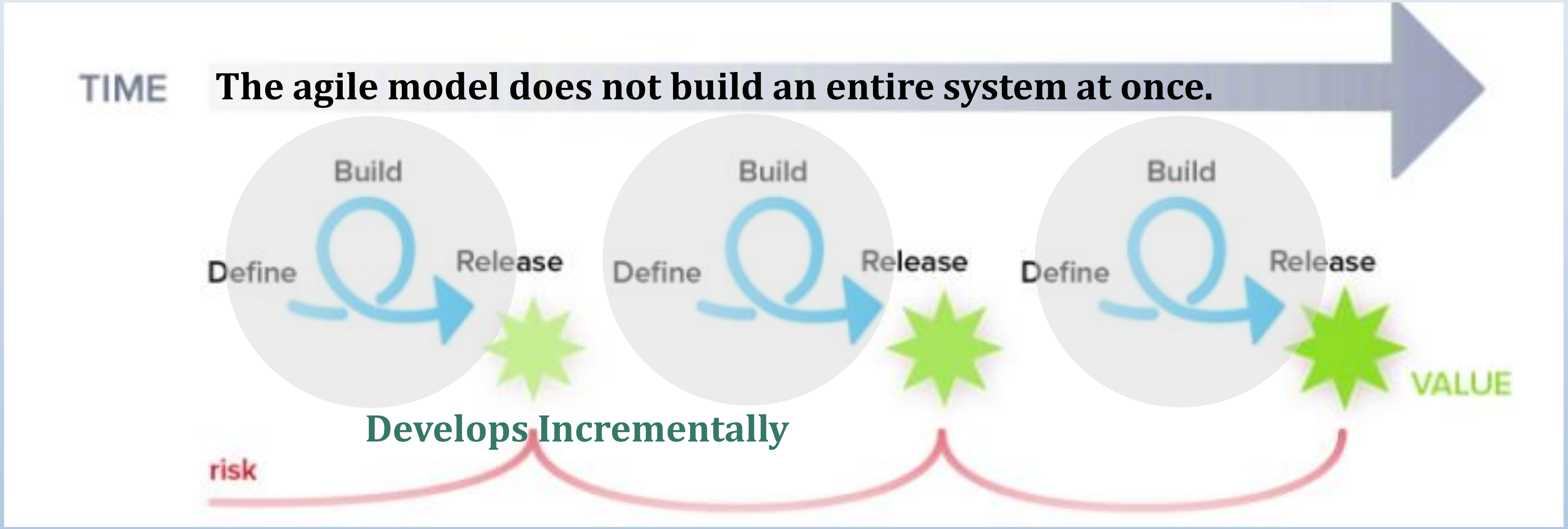
Does not reflect problem-solving nature of software development – iterations of phases

Integration is one big bang at the end

Little opportunity for customer to preview the system (until it may be too late)

Adaptive or Agile Project Life Cycle

Less time is invested upfront for documenting requirements when development is done incrementally.

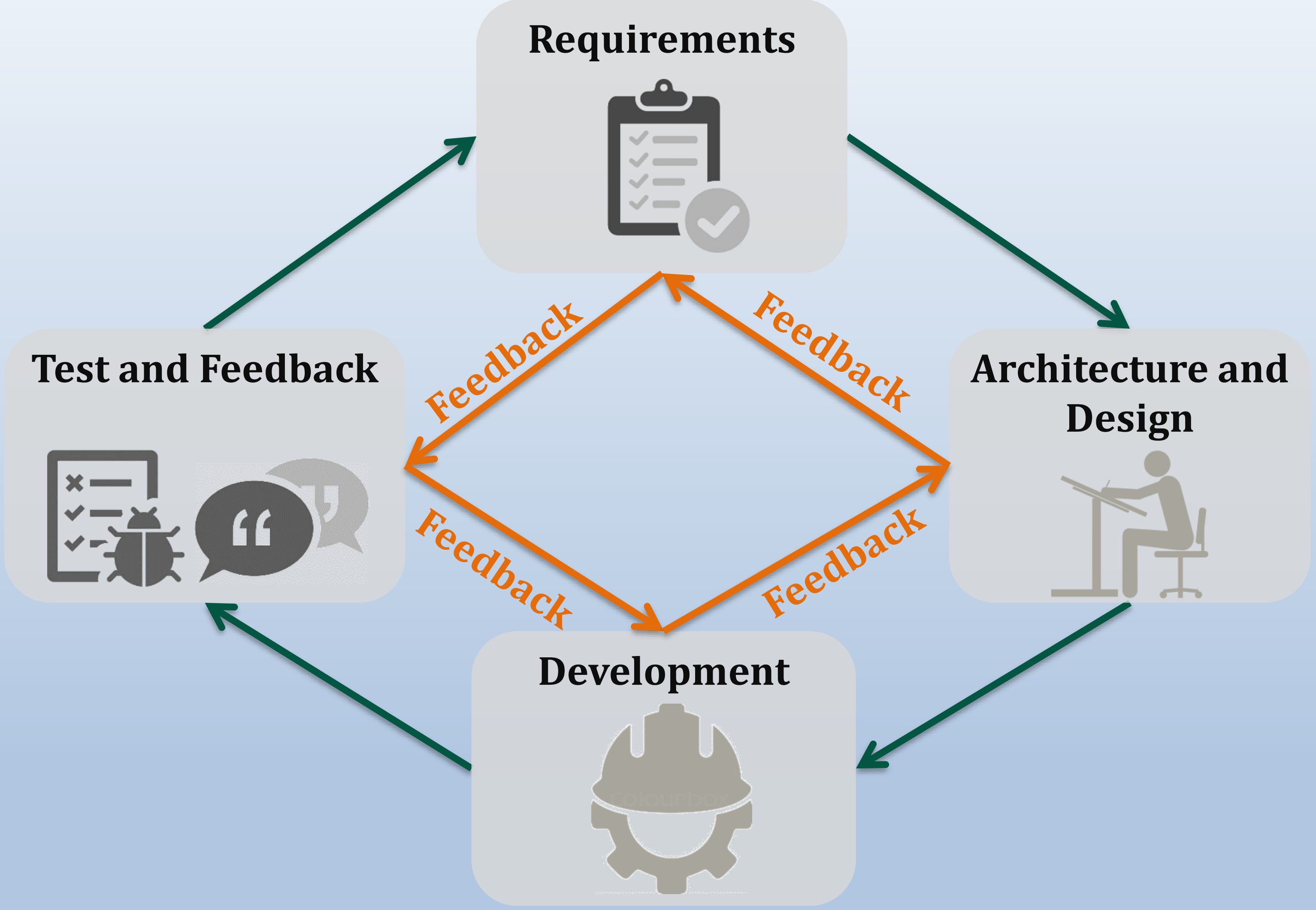


The agile model does not build an entire system at once.

Develops Incrementally

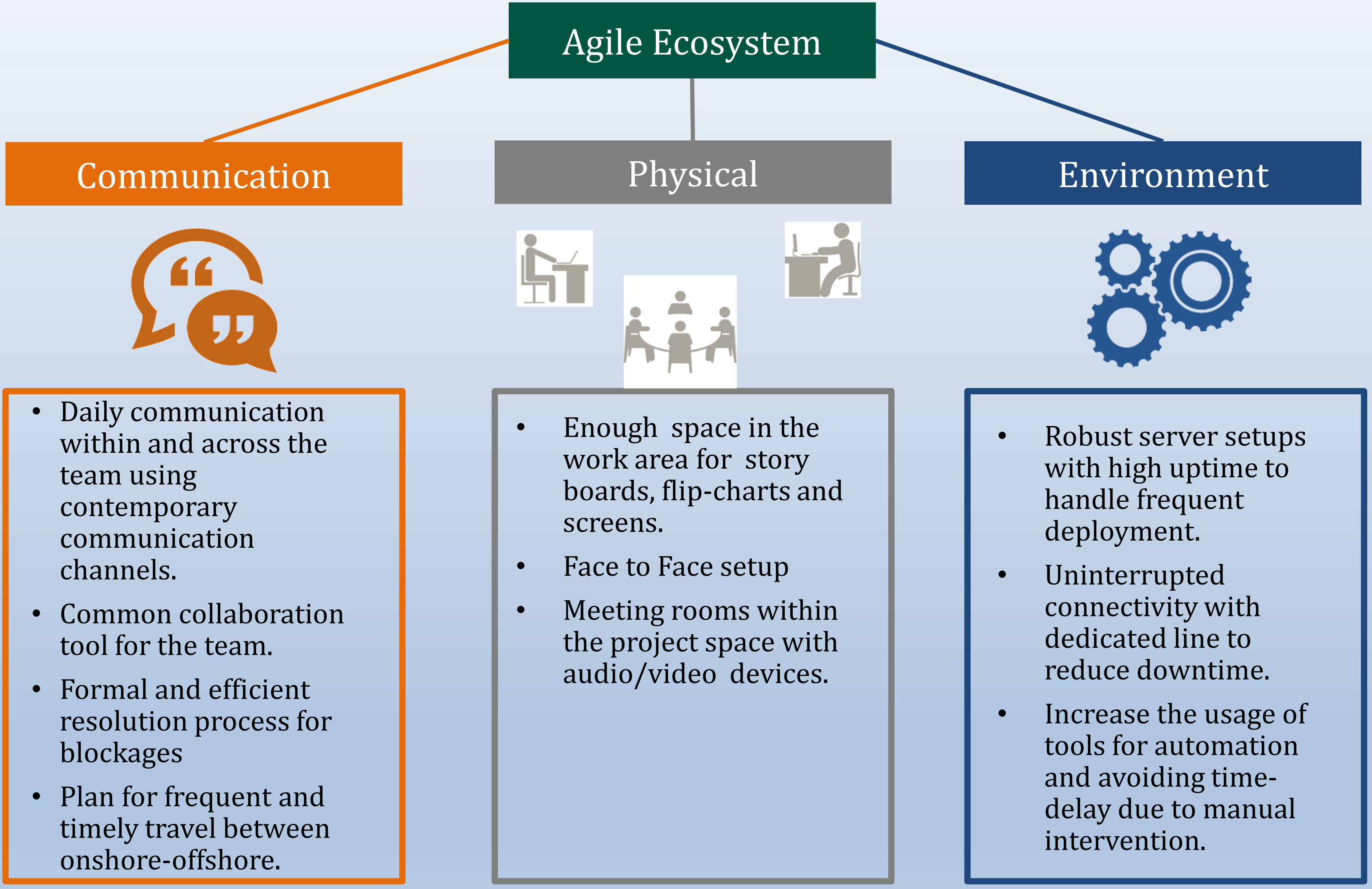
Unlike the more traditional waterfall approach, the agile development method is based on iterative and incremental development.

Adaptive or Agile Project Life Cycle



A mainline characteristic of agile software development is that customer feedback occurs simultaneously with development

Agile Ecosystem



Agile Ecosystem

Communication

- Daily communication within and across the team using contemporary communication channels.
- Common collaboration tool for the team.
- Formal and efficient resolution process for blockages
- Plan for frequent and timely travel between onshore-offshore.

Physical

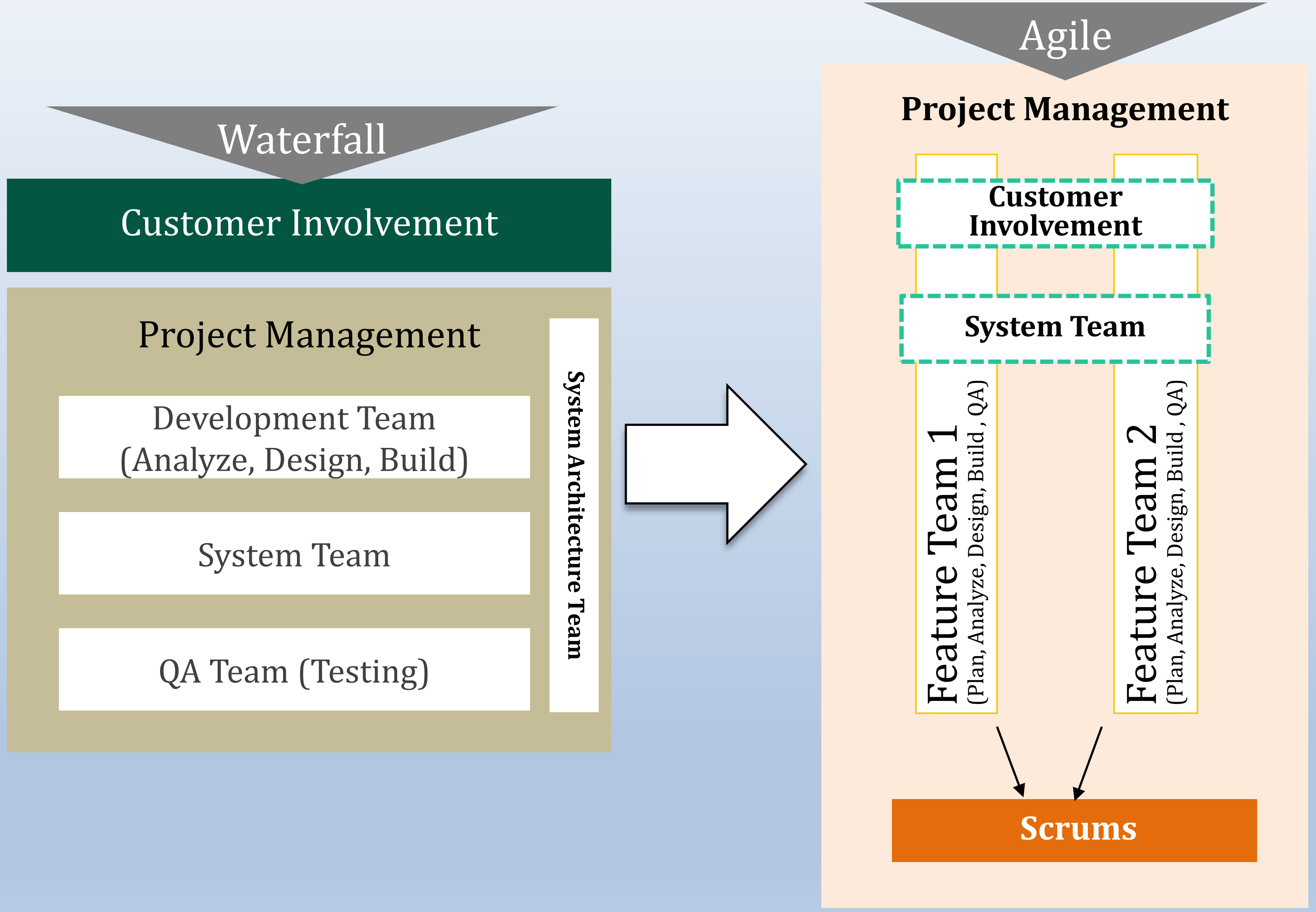
- Enough space in the work area for story boards, flip-charts and screens.
- Face to Face setup
- Meeting rooms within the project space with audio/video devices.

Environment

- Robust server setups with high uptime to handle frequent deployment.
- Uninterrupted connectivity with dedicated line to reduce downtime.
- Increase the usage of tools for automation and avoiding time-delay due to manual intervention.

Agility is speed that's what matters when you need to reduce the speed to market!

Team Transformation



Waterfall

Customer Involvement

Project Management

Development Team (Analyze, Design, Build)

System Team

QA Team (Testing)

System Architecture Team

Agile

Project Management

Customer Involvement

System Team

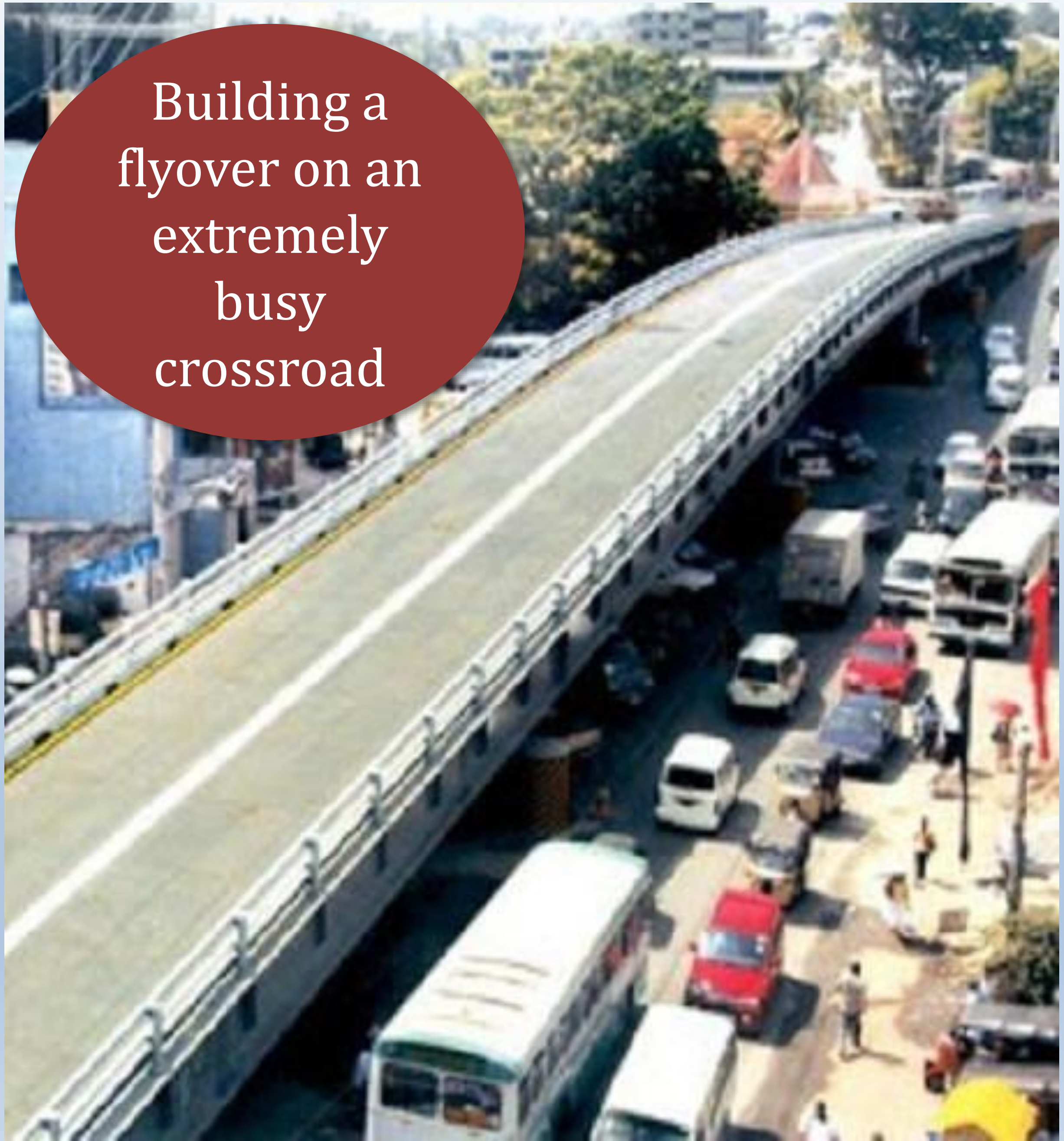
Feature Team 1 (Plan, Analyze, Design, Build , QA)

Feature Team 2 (Plan, Analyze, Design, Build , QA)

Scrums

Waterfall models gets trans mutated into Agile model as more and more traditional set ups are incorporating some or most of agile within themselves

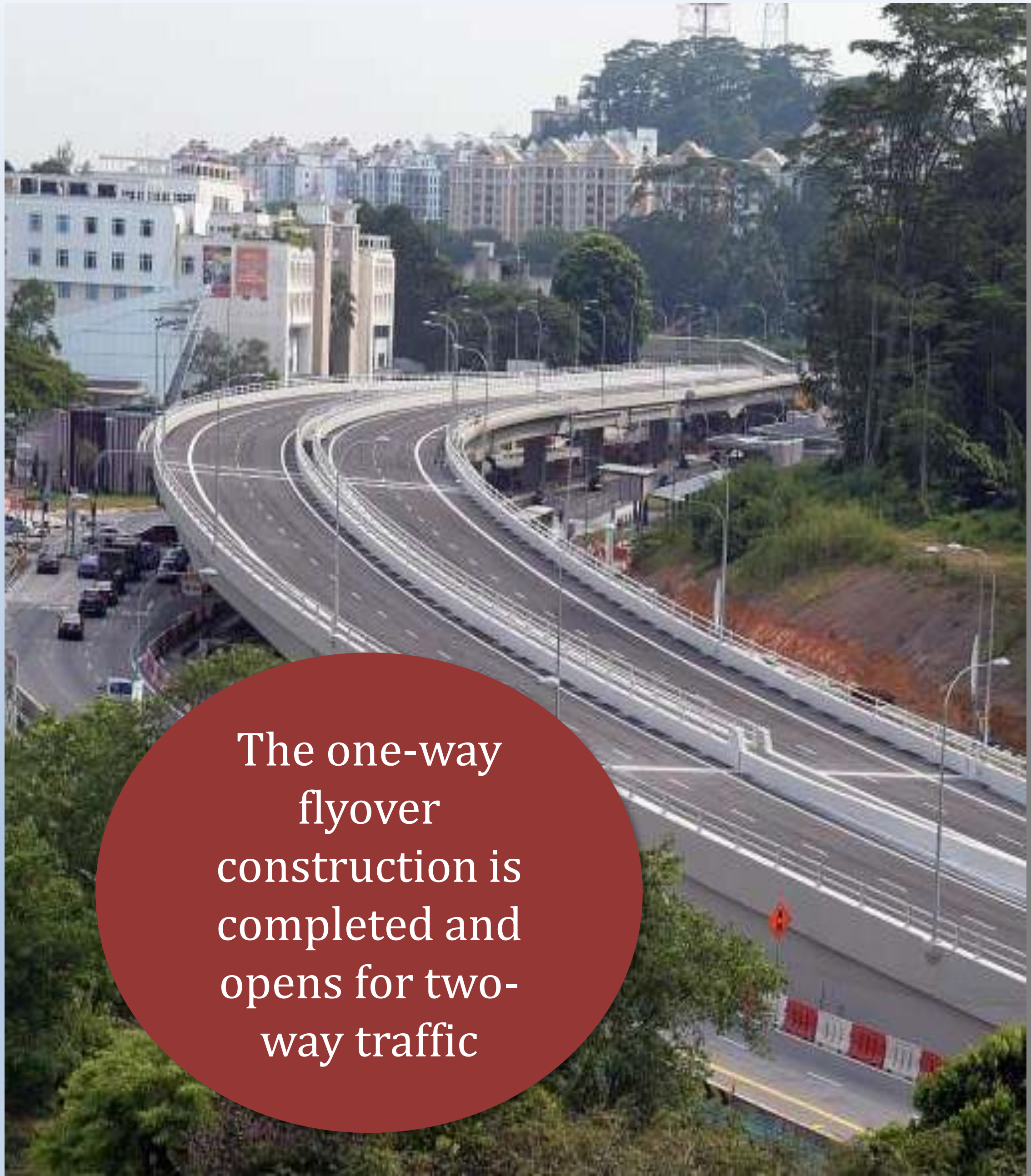
Example of Agile Approach



Building a flyover on an extremely busy crossroad

- This flyover project demonstrated how incremental delivery can indeed be extremely useful for the project as well as for the end customers.
- The construction was planned to have incremental delivery, so that one direction of the flyover would be constructed before starting the work on the second direction

Example of Agile Approach



The one-way flyover construction is completed and opens for two-way traffic

01

The overall traffic is still slow, but much better than without any flyovers.

02

Here the end customer (commuter) is using what we call a product of incremental delivery.

03

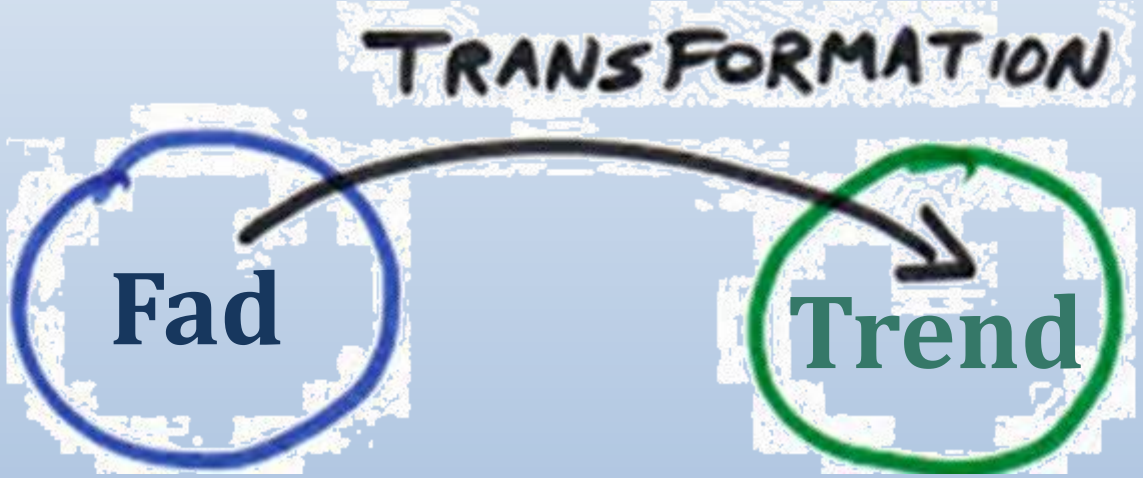
This incremental delivery helped customers use the project (the flyover) in nine months instead of waiting twice that long (plus some inevitable delays).

Agile Software Development

Agile is one of the big buzzwords of the IT development industry.

Five years ago,

agile practices transformed from the latest fad to a respectable trend.



As of **2016**, the majority of business analysts we have are experienced or are working in agile teams.

That’s because agile is much more widely accepted and adopted now as a discipline.

Agile Software Development

Agile development is a different way of managing IT development teams and projects.

The traditional approach to managing software development projects was failing far too often and there had to be a better way.

The agile manifesto describes 4 important values that are as relevant today as they were then.

It says,

“We 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”.

Agile Software Development

Over the last

10 years

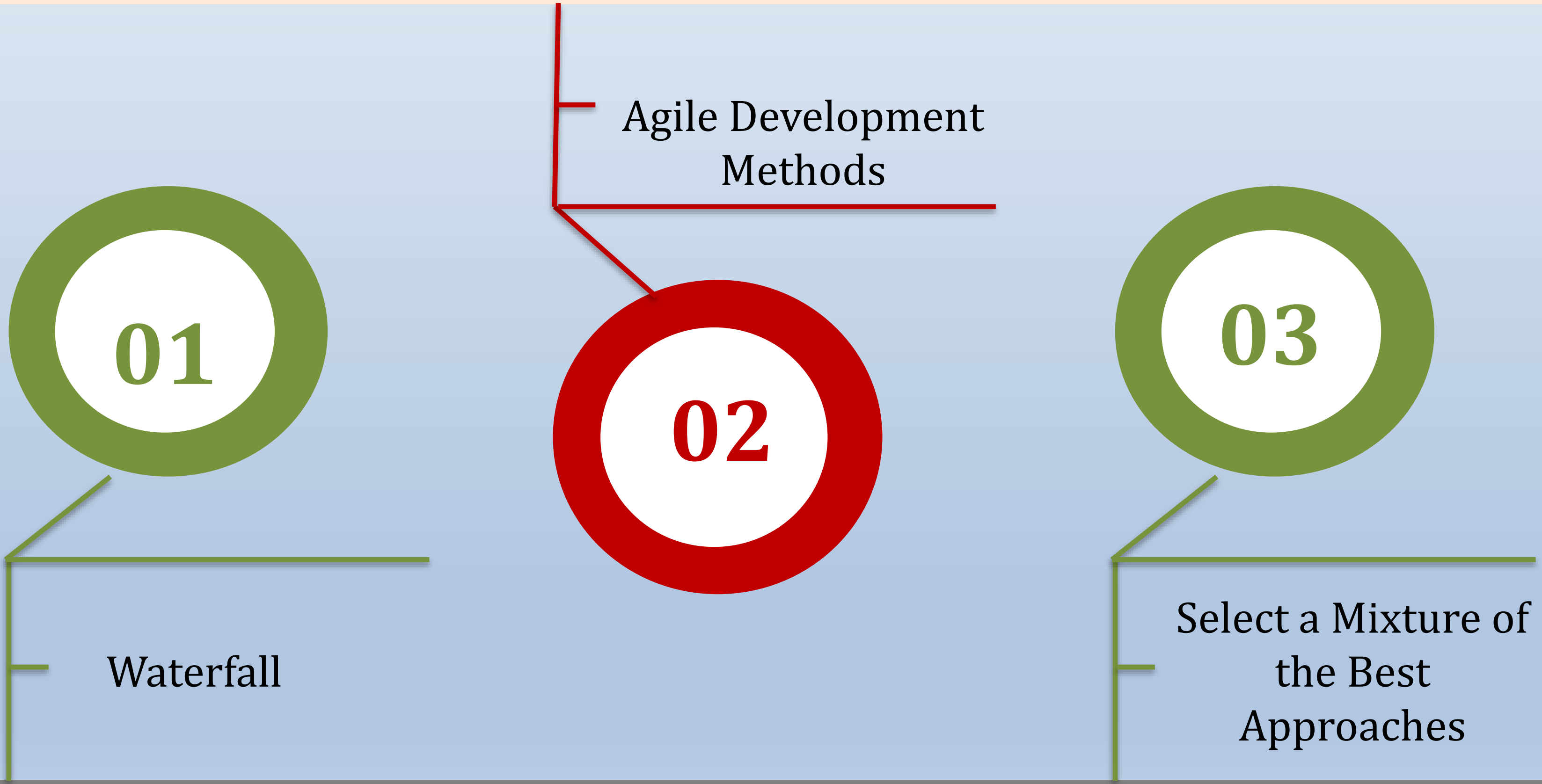
There is an ever-increasing volume of success stories, where companies have dramatically improved the success and performance of their IT development teams and projects.

SujataBatra

Agile Software Development

Agile is not a magic bullet for all software development issues.

The real trick is to know lots of techniques from various :



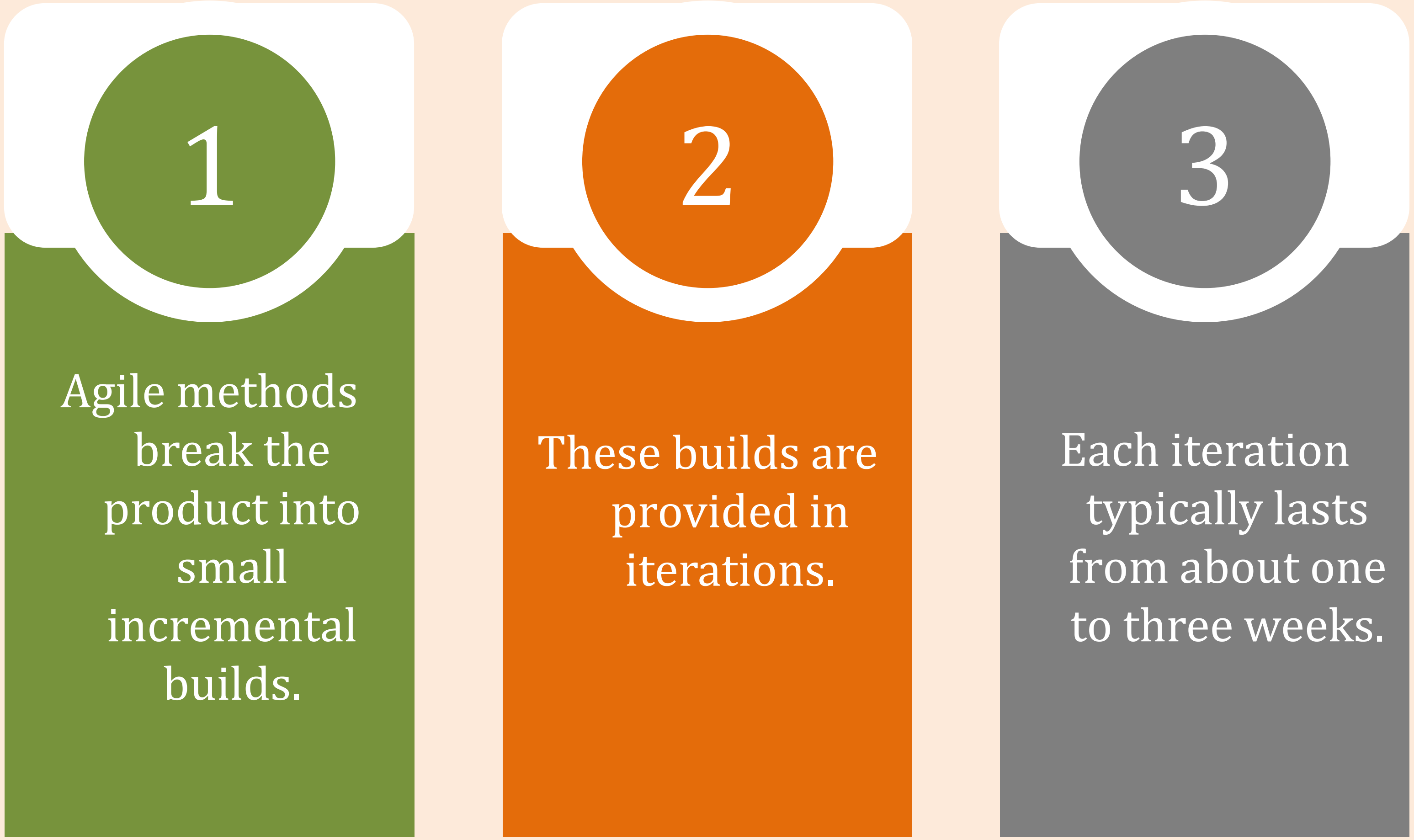
Agile Development Methods

Waterfall

Select a Mixture of the Best Approaches

To do this reliably with any degree of success really requires a lot of experience and skill.

Champion of Change - The Business Analyst



1

Agile methods break the product into small incremental builds.

2

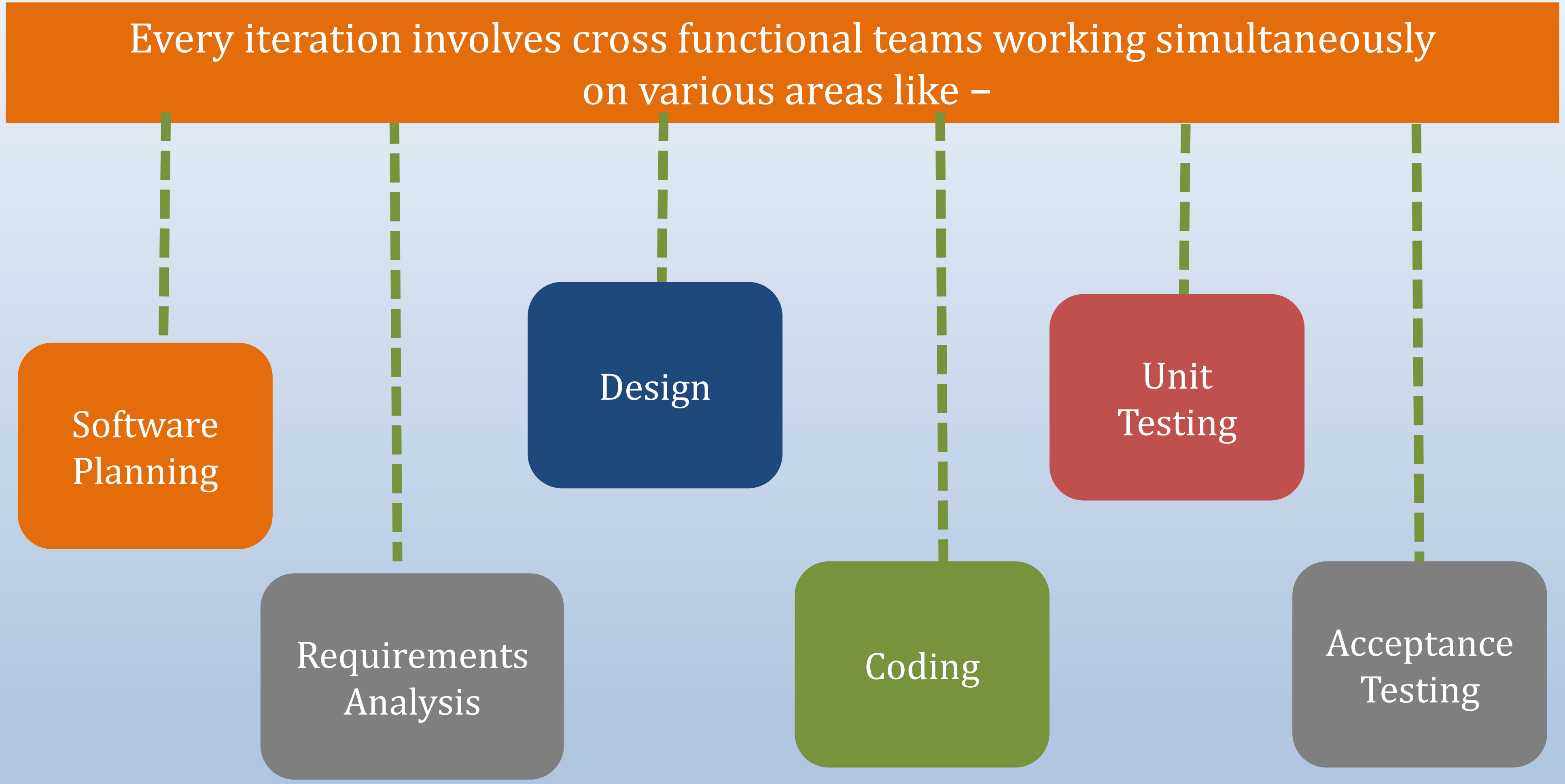
These builds are provided in iterations.

3

Each iteration typically lasts from about one to three weeks.

Sujata Bara

Agile Software Development



Every iteration involves cross functional teams working simultaneously on various areas like –

Software Planning

Requirements Analysis

Design

Coding

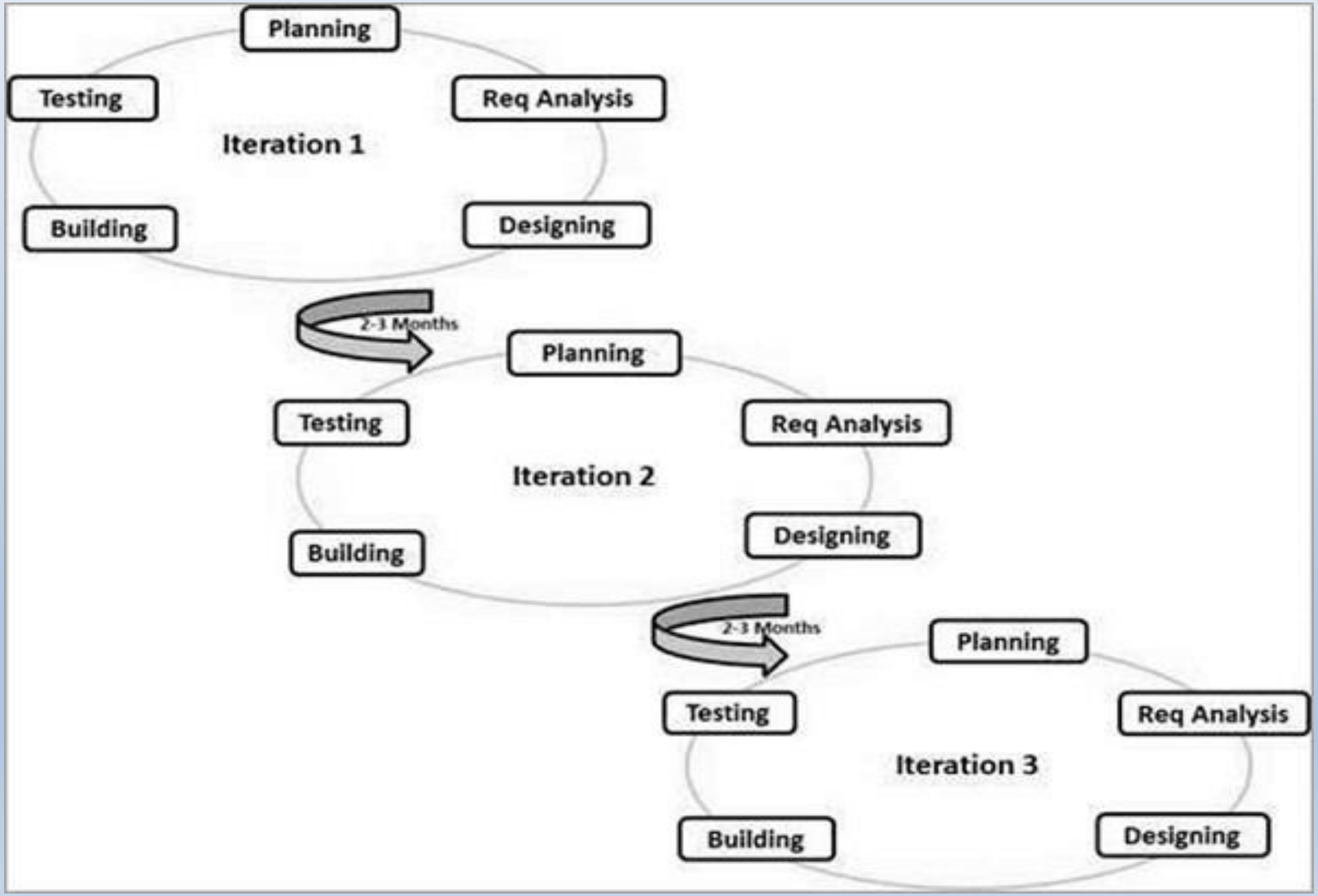
Unit Testing

Acceptance Testing

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Agile Software Development

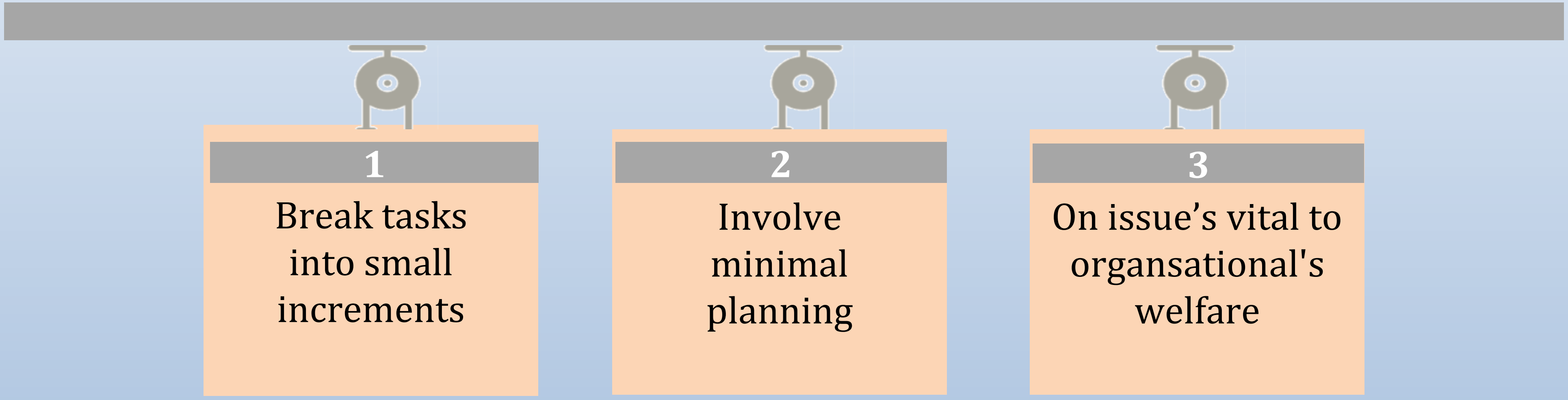
Here is a graphical illustration of the Agile Model –



Agile Software Development

The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

Agile methods :



1
Break tasks into small increments

2
Involve minimal planning

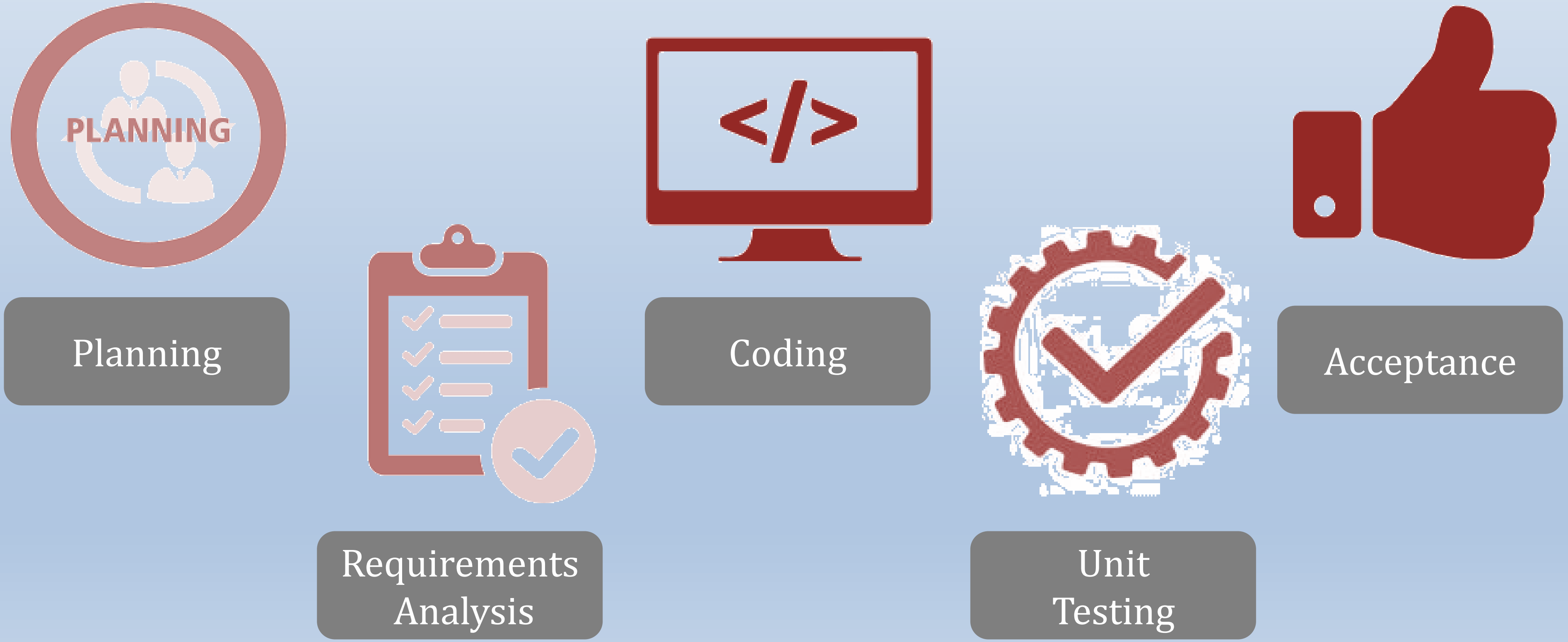
3
On issue’s vital to organsational's welfare

Do not Directly Involve Long-term Planning

Iterations are short time frames that last from one to four weeks.

Agile Software Development

Each iteration involves a team working through a full software development cycle, including:



Planning

Requirements Analysis

Coding

Unit Testing

Acceptance

Agile Software Development

This minimizes overall risk and allows the project to adapt to changes quickly.

An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal bugs) at the end of each iteration.

Multiple iterations might be required to release a product or new features.

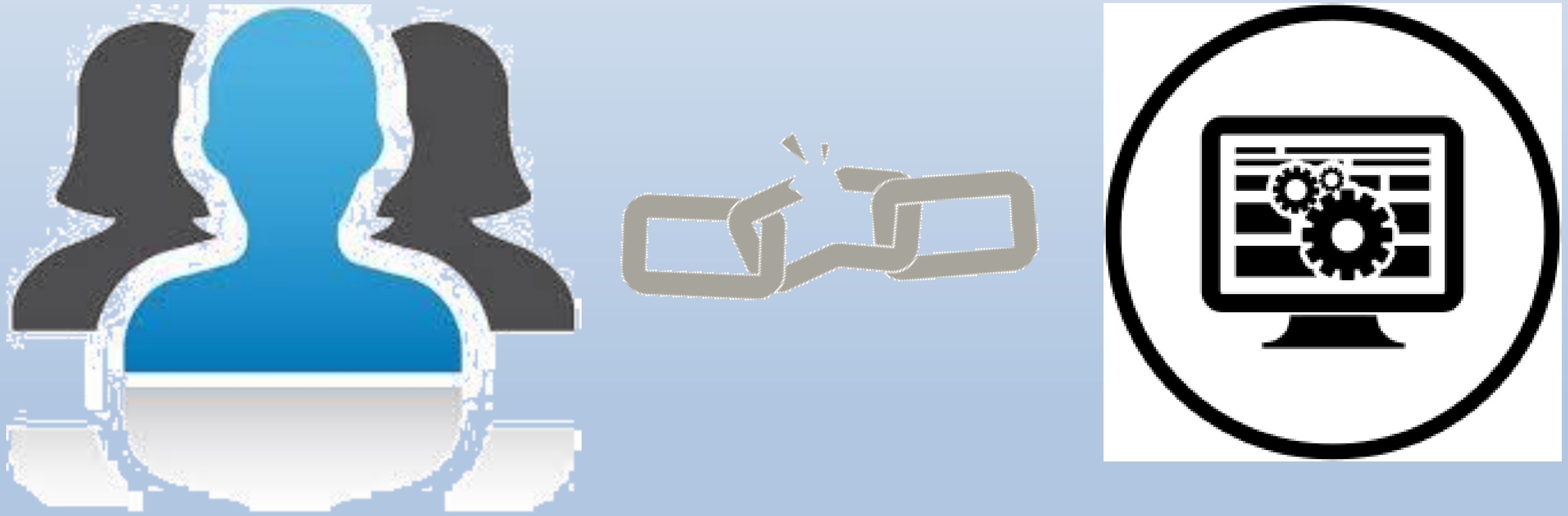
Agile methods emphasize face-to-face communication over written documents when the team is all in the same location.

Features of Agile

Principle 1: Active user involvement is imperative

Active user involvement is the first principle of agile development.

External users cannot be involved in project development projects



External Customers

Project Development

In this event it is imperative to have a senior and experienced user representative involved throughout.

Features of Agile

Principle 2: Agile Development Teams Must Be Empowered

An agile development team must include all the necessary team members to make decisions and make them on a timely basis.

The team must establish and clarify and prioritise requirements, agree to the tasks required to deliver, and estimate the effort involved.

Features of Agile

Principle 3: Requirements evolve, but timescales are fixed.

01 Agile development works on the premise that requirements emerge and evolve, and that however much you analyze and design, this will always be the case because you cannot really know for sure what you want until you see and use the software.

02 In the time, you would have spent analyzing and reviewing requirements and designing a solution, external conditions could also have changed.

03 Agile development projects accept change; in fact, they expect it.

04 In these projects, requirements are allowed to evolve, but the timescale is fixed.

Features of Agile

Principle 3: Requirements evolve, but timescales are fixed.

05 To include a new requirement, or to change a requirement, the user or product owner must remove a comparable amount of work from the project in order to accommodate the change.

06 This ensures the team remains focused on the agreed timescale and allows the product to evolve into the right solution.

07 It does, however, also pre-suppose that there’s enough non-mandatory features included in the original timeframes to allow these trade-off decisions to occur without fundamentally compromising the end product.

Features of Agile

Principle 4: Agile Requirements are barely sufficient

Agile development teams capture requirements at a high level and on a piecemeal basis, just-in-time for each feature to be developed.

Agile requirements are ideally visual and should be barely sufficient, i.e. the absolute minimum required to enable development and testing to proceed with reasonable efficiency.

The rationale for this is to minimise the time spent on anything that doesn’t actually form part of the end product.

Features of Agile

Principle 5: Done means done!

01 Features developed within iteration i.e. a sprint in scrum, should be 100% complete by the end of the sprint.

02 Too often in software development, “done” doesn’t really mean “done!”, tested, styled and accepted by the product owner. It just means developed.

03 Make sure that each feature is fully developed, tested, styled, and accepted by the product owner before counting it as “DONE!”.

04 If there is any doubt about what activities should or shouldn’t be completed within the sprint for each feature, “DONE!” should mean shippable.

Features of Agile

05 Multiple features can be developed in parallel in a team situation.

06 However, within the work of each developer, do not move on to a new feature until the last one is shippable.

07 This is important to ensure the overall product is in a shippable state at the end of the sprint, not in a state where multiple features are 90% complete or untested, as is more usual in traditional development projects.

Features of Agile

Principle 6: Agile testing is not for dummies!

Testing is integrated throughout the software development lifecycle.

Agile development does not have a separate test phase as such.

Developers are much more heavily engaged in testing, writing automated repeatable unit tests to validate their code.

Features of Agile

Principle 6: Agile testing is not for dummies!

1. With automated repeatable unit tests, testing can be done as part of the build, ensuring that all features are working correctly each time the build is produced.

2. And builds should be regular, at least daily, so integration is done as you go too.

3. The purpose of these principles is to keep the software in releasable condition throughout the development, so it can be shipped whenever it’s appropriate.

Scrum

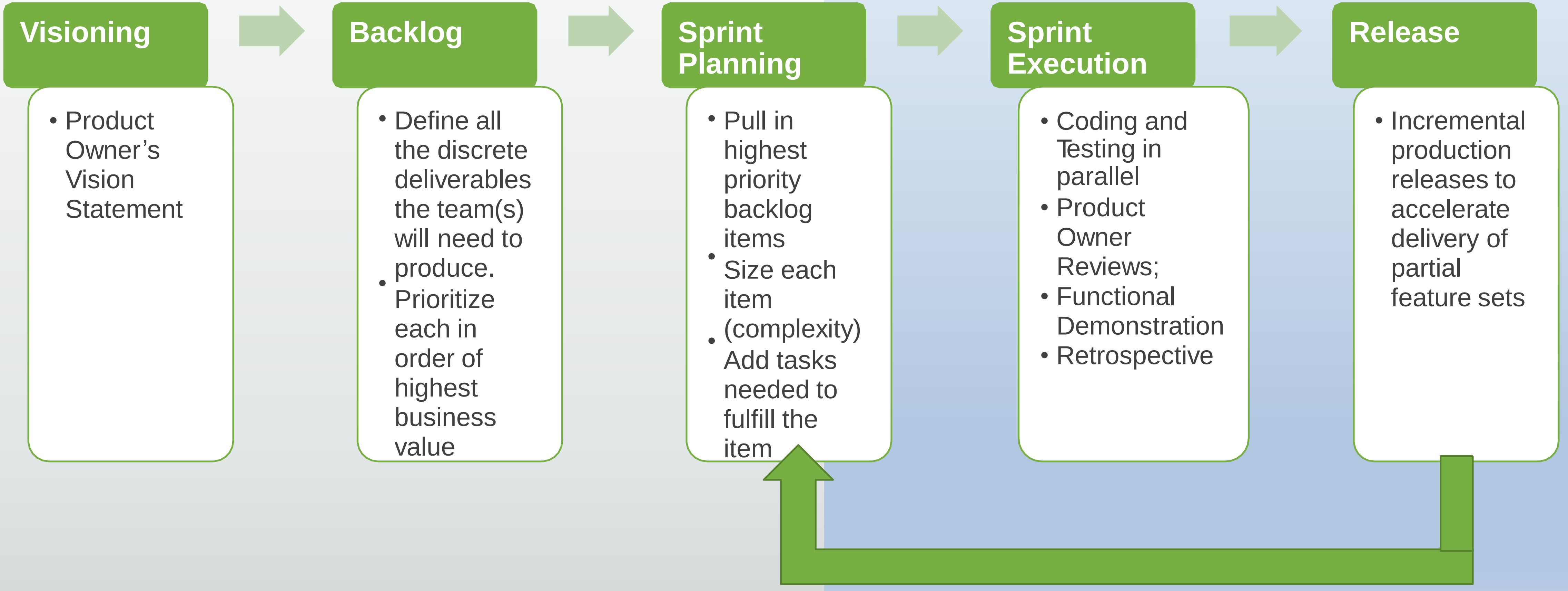
Overview of the Scrum Practice Framework

SujataBatra

Scrum

Scrum is the framework that helps teams work together.
Gets its name from a Rugby term used as a metaphor to reflect the degree of team cooperation needed to advance the football across the goal line.

How Does It Work



Visioning

- Product Owner's Vision Statement

Backlog

- Define all the discrete deliverables the team(s) will need to produce.
- Prioritize each in order of highest business value

Sprint Planning

- Pull in highest priority backlog items
- Size each item (complexity)
- Add tasks needed to fulfill the item

Sprint Execution

- Coding and Testing in parallel
- Product Owner Reviews;
- Functional Demonstration
- Retrospective

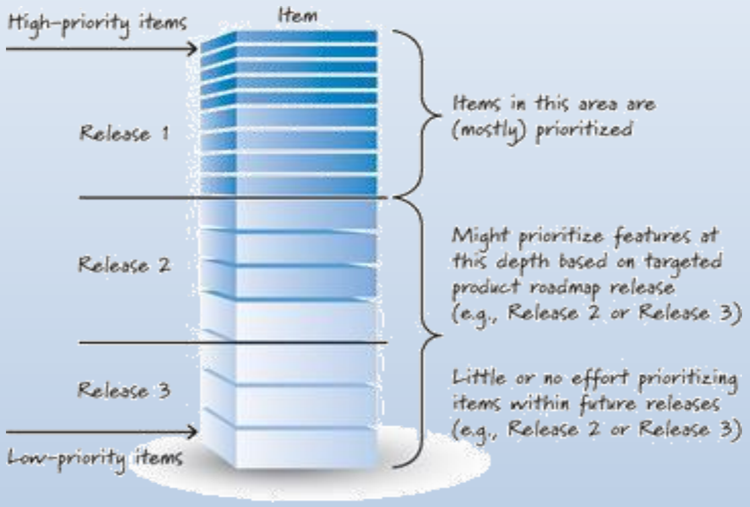
Release

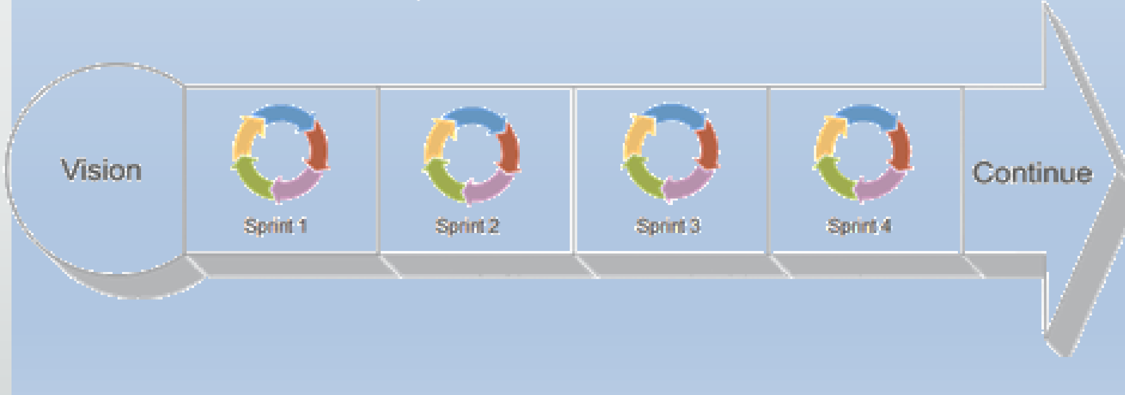
- Incremental production releases to accelerate delivery of partial feature sets

Scrum

How Does It Work

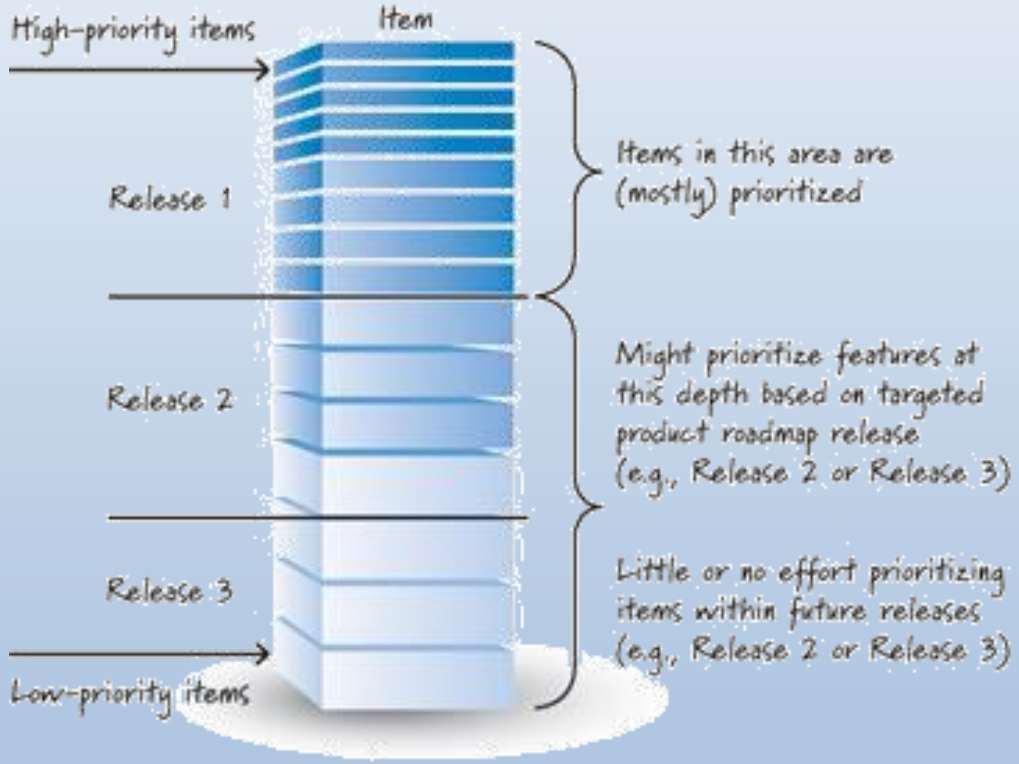
- Product Backlog: Single Source of Functional and non-functional requirements.
- Chops up the Product Backlog into a series of smaller pieces
- Each piece is worked within a time boxed period called a Sprint.
- Work is inspected, accepted or rejected each Sprint by the Product Owner (business owner).


High-priority items
Item
Release 1
Items in this area are (mostly) prioritized
Release 2
Might prioritize features at this depth based on targeted product roadmap release (e.g., Release 2 or Release 3)
Release 3
Little or no effort prioritizing items within future releases (e.g., Release 2 or Release 3)
Low-priority items



How Does It Work

- Business Value
  - Work is prioritized highest business value to lower business value.
  - Highest value items should be elaborated in detail; ready for the next Sprint Planning.
- Tactics
  - MoSCoW (must have, should have, could have, won't have)
  - REIO (Required, Essential, Important, Optional)
  - Cost – Benefit Matrix


High-priority items
Item
Release 1
Items in this area are (mostly) prioritized
Release 2
Might prioritize features at this depth based on targeted product roadmap release (e.g., Release 2 or Release 3)
Release 3
Little or no effort prioritizing items within future releases (e.g., Release 2 or Release 3)
Low-priority items

User Story

Describes a small discrete “need” from the perspective of a role or persona.

As a [call center agent] (WHO)

I need to [login with my password] (WHAT)

So that [I can access the customer’s reservation to cancel it] (WHY)

Contains acceptance criteria that defines “done” (story is done when . . .)

- ❑ a premium member can cancel same day without a fee
- ❑ a non-premium member is charged 10% for a same-day cancellation
- ❑ email confirmation is sent to the customer
- ❑ hotel is notified of the cancellation

User Story

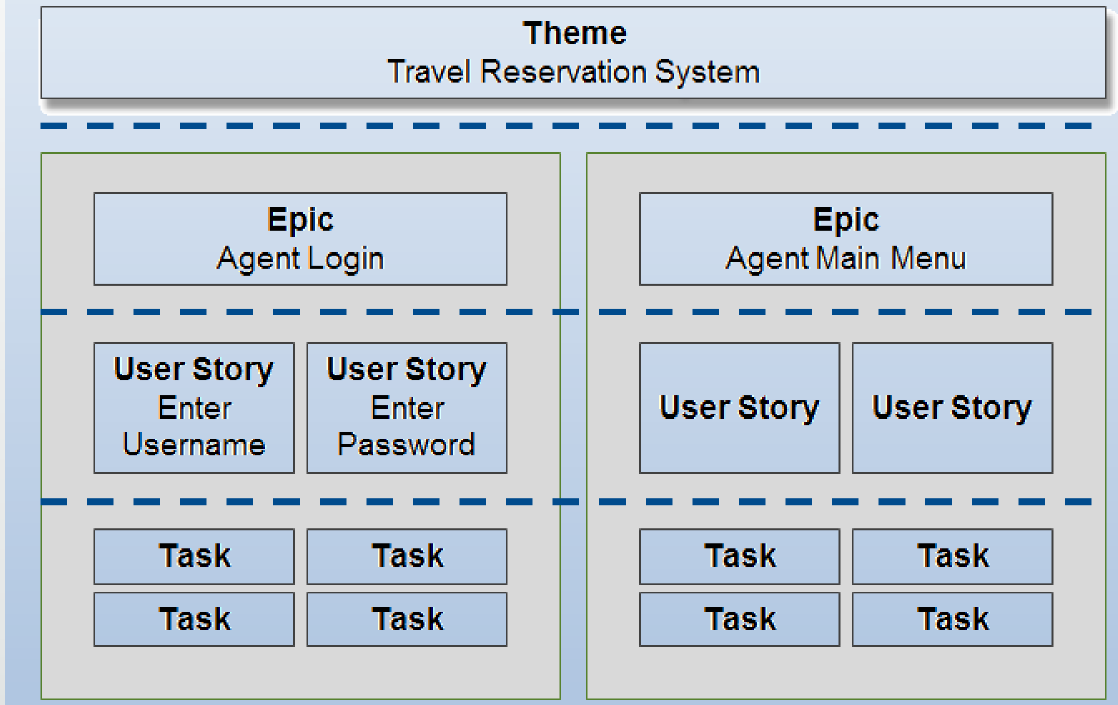
- Contains tasks that describe the actions and estimated effort required to fulfill the Story need.
  - Typically starts with a verb, concise, and self evident what the action is and an estimate of effort

○ Create User Table	(1 hr)
○ Create password encryption service	(4 hr)
○ Create login service	(4 hr)

- Is testable (functionally)
  - Well constructed acceptance criteria doubles as functional test criteria for the story (positive and negative)
    - User can login using a valid password
    - User cannot login using an invalid password

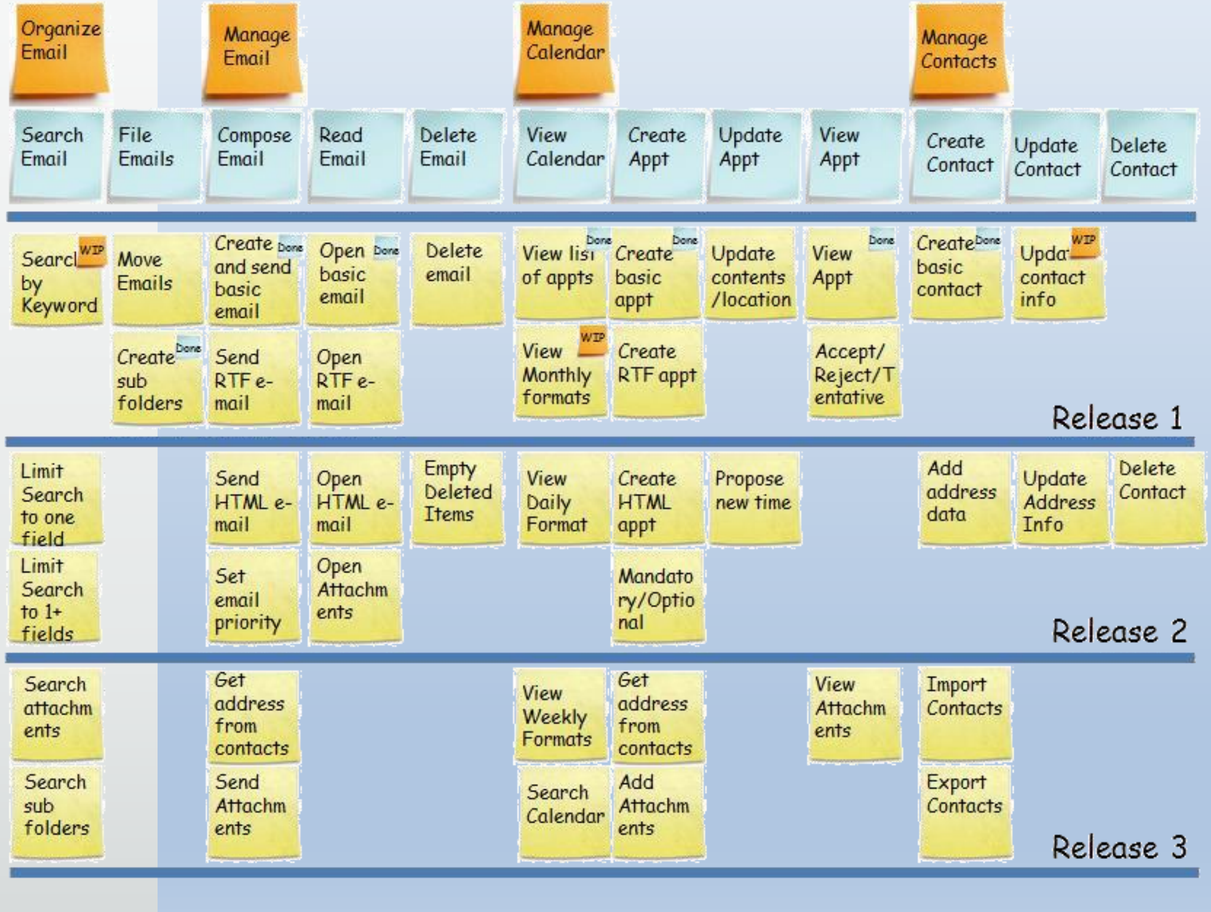
User Story Scope

- Theme
  - Very broad high level category of related Epics and Stories
- Epic
  - High level User Story; typically representing a broad functional feature
  - Epics are sometimes referred to as Feature
- User Story
  - Represents a discreet detailed functional requirement.



Story Map

- Make visible the workflow or value chain
- Show relationships of larger stories to child stories
- Help confirm the completeness of the Backlog
- Provide a useful context for prioritization
- Plan releases in complete slices of functionality



Organize Email		Manage Email			Manage Calendar				Manage Contacts		
Search Email	File Emails	Compose Email	Read Email	Delete Email	View Calendar	Create Appt	Update Appt	View Appt	Create Contact	Update Contact	Delete Contact
Search by Keyword WIP	Move Emails	Create and send basic email Done	Open basic email Done	Delete email	View list of appts Done	Create basic appt Done	Update contents /location	View Appt Done	Create basic contact Done	Update contact info WIP	
	Create sub folders Done	Send RTF e-mail	Open RTF e-mail		View Monthly formats WIP	Create RTF appt		Accept/Reject/Tentative			
											Release 1
Limit Search to one field		Send HTML e-mail	Open HTML e-mail	Empty Deleted Items	View Daily Format	Create HTML appt	Propose new time		Add address data	Update Address Info	Delete Contact
Limit Search to 1+ fields		Set email priority	Open Attachments			Mandatory/Optional					
											Release 2
Search attachments		Get address from contacts			View Weekly Formats	Get address from contacts		View Attachments	Import Contacts		
Search sub folders		Send Attachments			Search Calendar	Add Attachments			Export Contacts		
											Release 3

SujataBatra

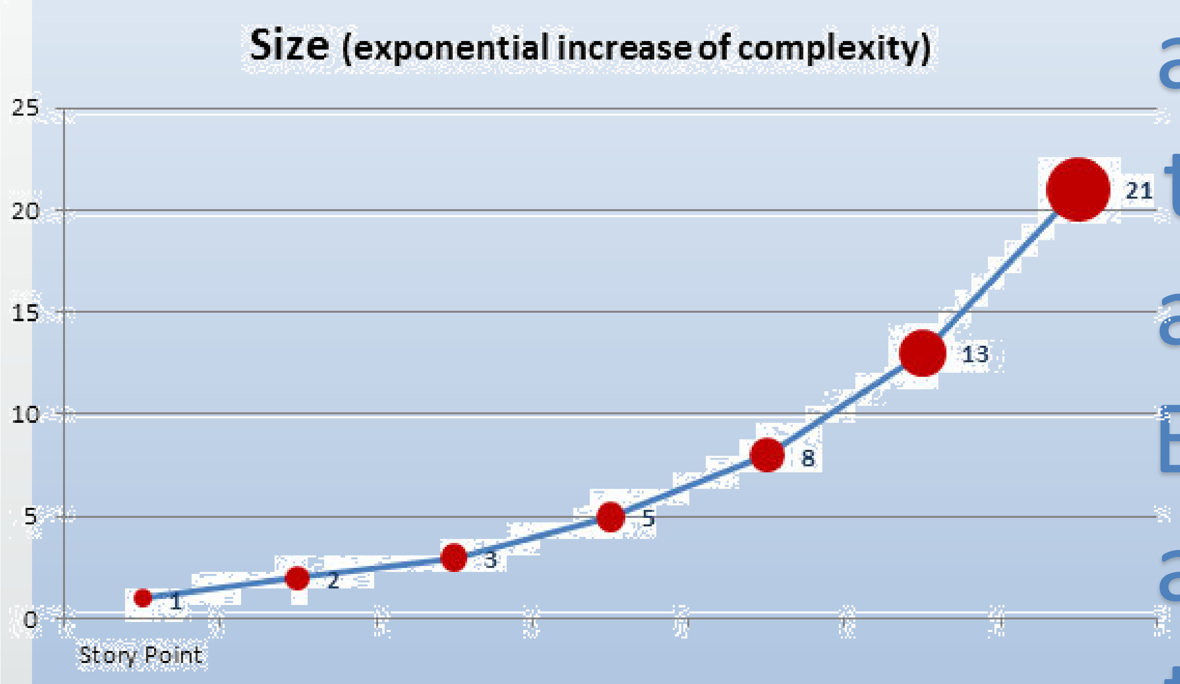
Release Roadmap

- Helps align stakeholder expectations
- List the Release Name or Version Number
- List the goals for each release
- List the Preliminary feature set for each release
- Optionally include metrics that help define if the goal(s) were met

Timeline	2016Q1	2016Q2	2016Q3	2016Q4
Rel ID	R1	R2	R3	R4
Goal	One UI; all admin systems, basic search functions	Add additional search types	Integration of IVR pop, SWAP, and CLASS	Additional Notes Functionality
Features	• Name search • Organization search • Policy number search • View Contract details (Summary) • Search usage reporting	• Customer search using last 4 of SSN • Search using FULL SSN • Adjustments to Agent Result Data • Search usage reporting adjustments	• IVR Pop integration • View note by Policy Number and Owner • SWAP Integration CLASS Integration	• Attention and Alert note handling • Copy/paste functionality • Ability to enter notes on UI and write back to source system

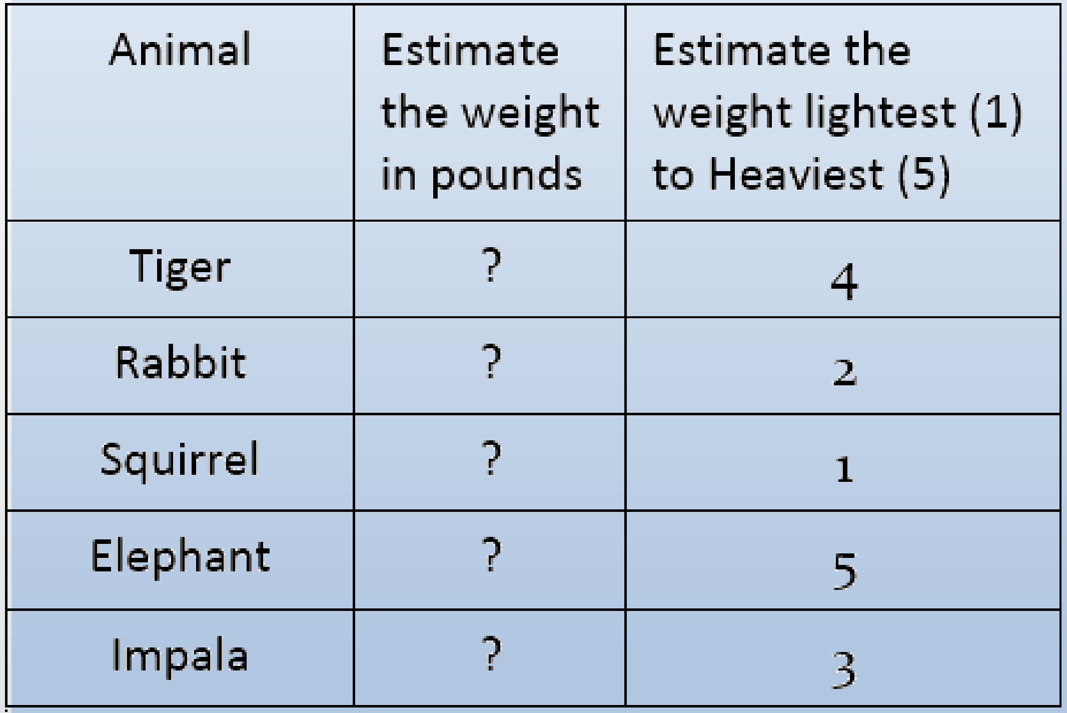
Estimating

- Story Points
  - Variation of tee-shirt sizing estimated in points relative to perceived complexity of the story (effort, complexity, and risk)
  - Much quicker and accurate than time spent ‘breaking down and measuring’
- Techniques
  - Planning poker cards
  - Reference Story. 2 story points = ‘small’, size other User Stories relative to that; smaller, larger, same.



Relative Estimation Advantage

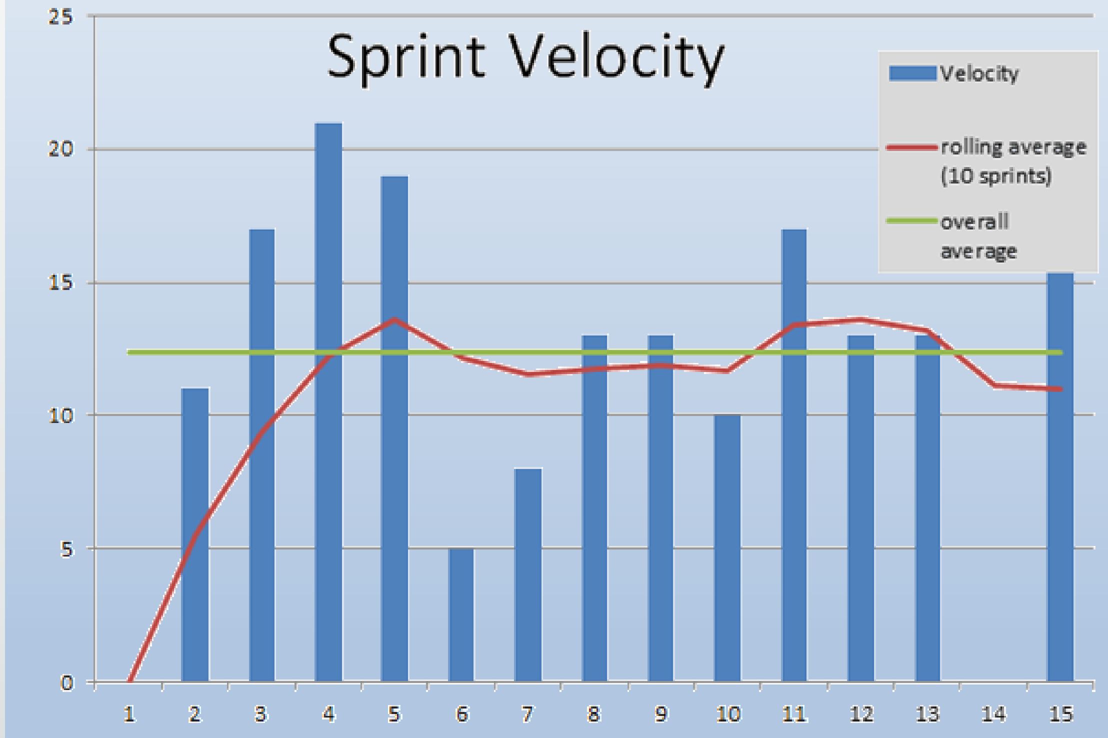
- Humans are terrible at absolute estimation but quite good at relative estimation.
- It is generally faster
- It gets a team thinking (and talking) as a group, rather than as individuals
- It encourages spending analysis time appropriately
- It is cost-effective



Animal	Estimate the weight in pounds	Estimate the weight lightest (1) to Heaviest (5)
Tiger	?	4
Rabbit	?	2
Squirrel	?	1
Elephant	?	5
Impala	?	3

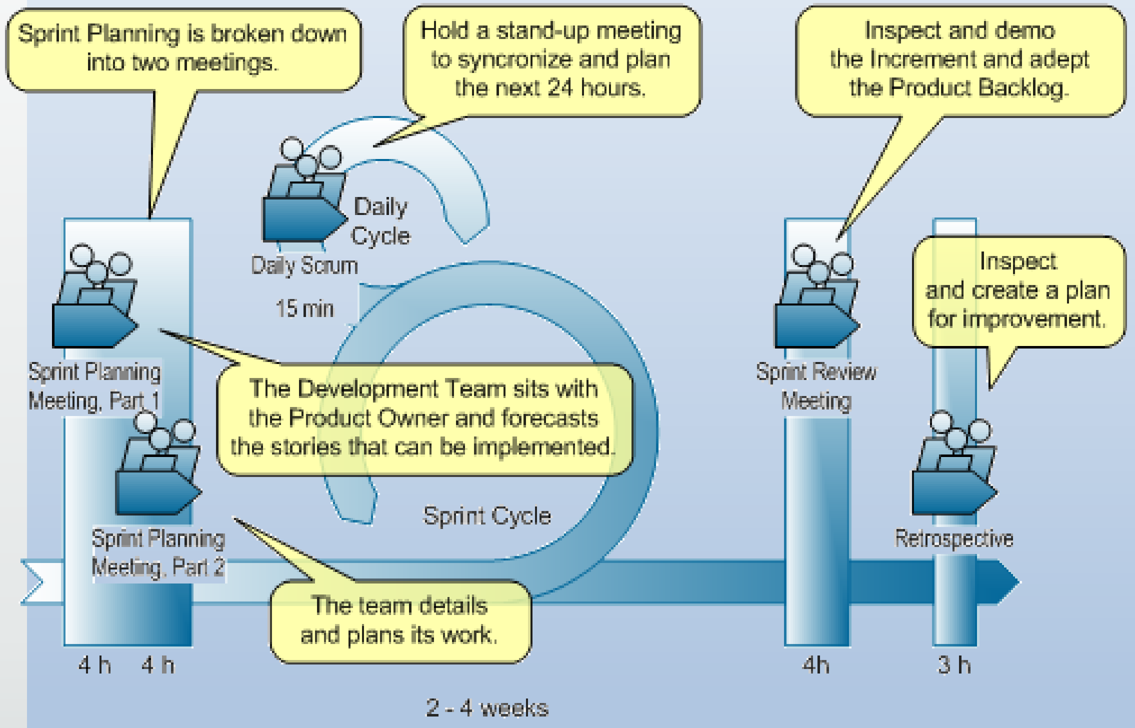
Velocity

- Points total from all completed stories is the team’s velocity for that Sprint.
- After several Sprints, velocity “norms”. Average velocity then becomes a predictor of Sprint throughput.
- The team can periodically compute estimated project completion based on backlog remaining points



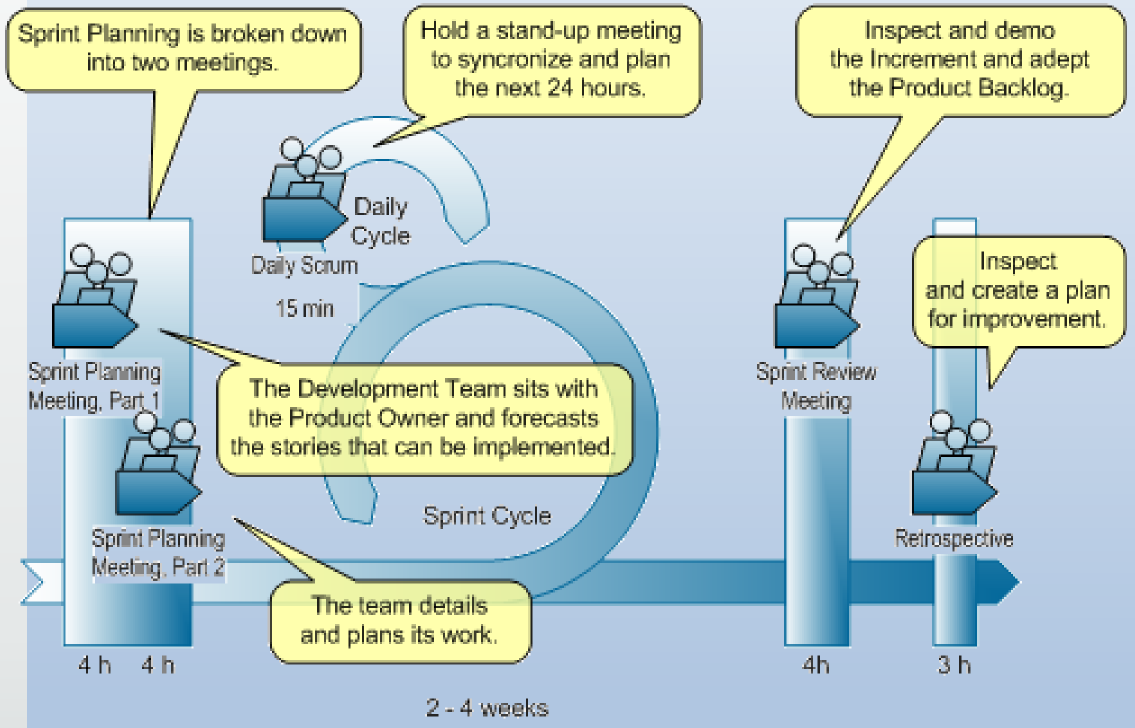
Sprints

- Time boxed
  - Typically 2-4 weeks
- Sprint Planning (Day 1)
  - Pull in the next highest priority items from the backlog.
  - First session with the Product Owner
  - Second session to work out the technical strategy for completing the work.


Sprint Planning is broken down into two meetings.
Hold a stand-up meeting to syncronize and plan the next 24 hours.
Inspect and demo the Increment and adept the Product Backlog.
Daily Cycle
Daily Scrum
15 min
Inspect and create a plan for improvement.
Sprint Planning Meeting, Part 1
The Development Team sits with the Product Owner and forecasts the stories that can be implemented.
Sprint Review Meeting
Sprint Cycle
Sprint Planning Meeting, Part 2
Retrospective
The team details and plans its work.
4 h
4 h
4h
3 h
2 - 4 weeks

Sprints

- Daily Stand-up (Each Day)
  - Each team member:
    - What they did yesterday
    - What they plan to do today
    - Any impediments blocking progress.
- Sprint Review (Final Day)
  - Product Owner reviews achievements of the Sprint with the team
  - Product Owner and team presents a demonstration or discusses latest functionality with external audience.



Sprint Planning is broken down into two meetings.

Hold a stand-up meeting to syncronize and plan the next 24 hours.

Inspect and demo the Increment and adept the Product Backlog.

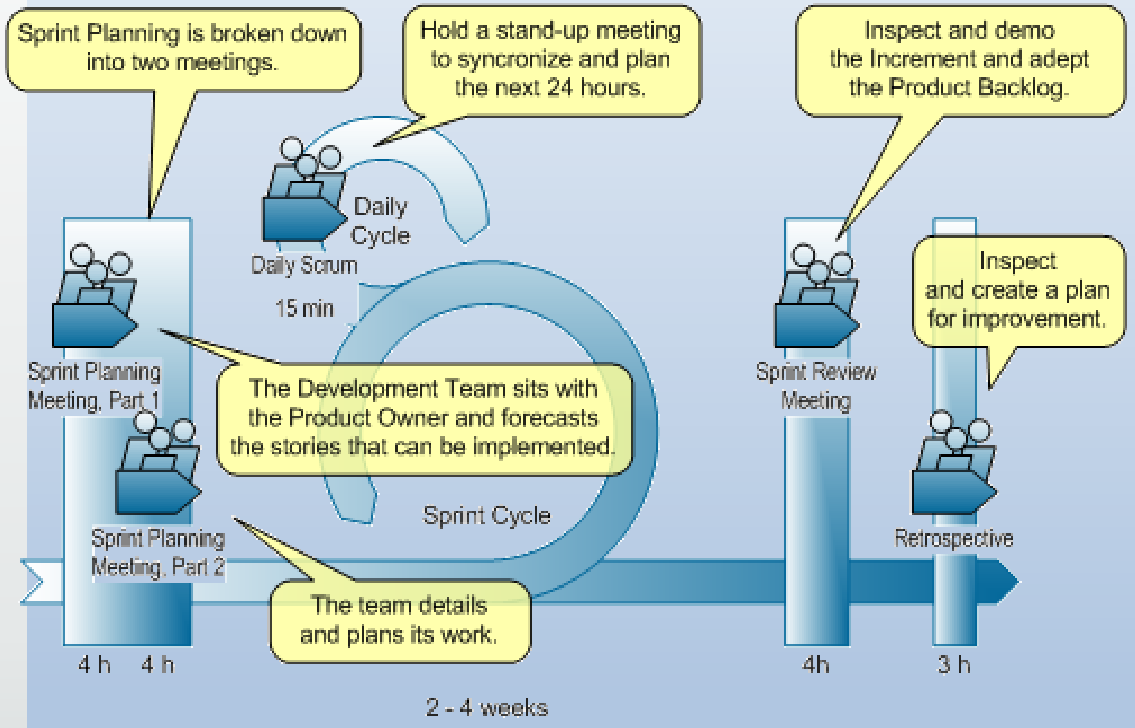
Inspect and create a plan for improvement.

The Development Team sits with the Product Owner and forecasts the stories that can be implemented.

The team details and plans its work.

How Does It Work

- Retrospective
  - The Retrospective, or ‘Retro’, is attended by the Scrum Master and the team and is the final team meeting in the Sprint.
  - The primary purpose is to determine what went well, what didn't go well, and how the team can improve in the next Sprint.
  - The Retrospective is the opportunity for the team to focus on its overall performance and identify strategies for continuous improvement on its processes


Sprint Planning is broken down into two meetings.
Hold a stand-up meeting to syncronize and plan the next 24 hours.
Inspect and demo the Increment and adept the Product Backlog.
Daily Cycle
Daily Scrum
15 min
Inspect and create a plan for improvement.
Sprint Planning Meeting, Part 1
The Development Team sits with the Product Owner and forecasts the stories that can be implemented.
Sprint Review Meeting
Sprint Cycle
Sprint Planning Meeting, Part 2
Retrospective
The team details and plans its work.
4 h
4 h
4h
3 h
2 - 4 weeks

Sujata Batra

Roles

Product Owner

- Represents the Business
- Defines requirements (the backlog)
- Accepts or rejects team output
- Makes business decisions
- Provides visibility to leadership

Scrum Master

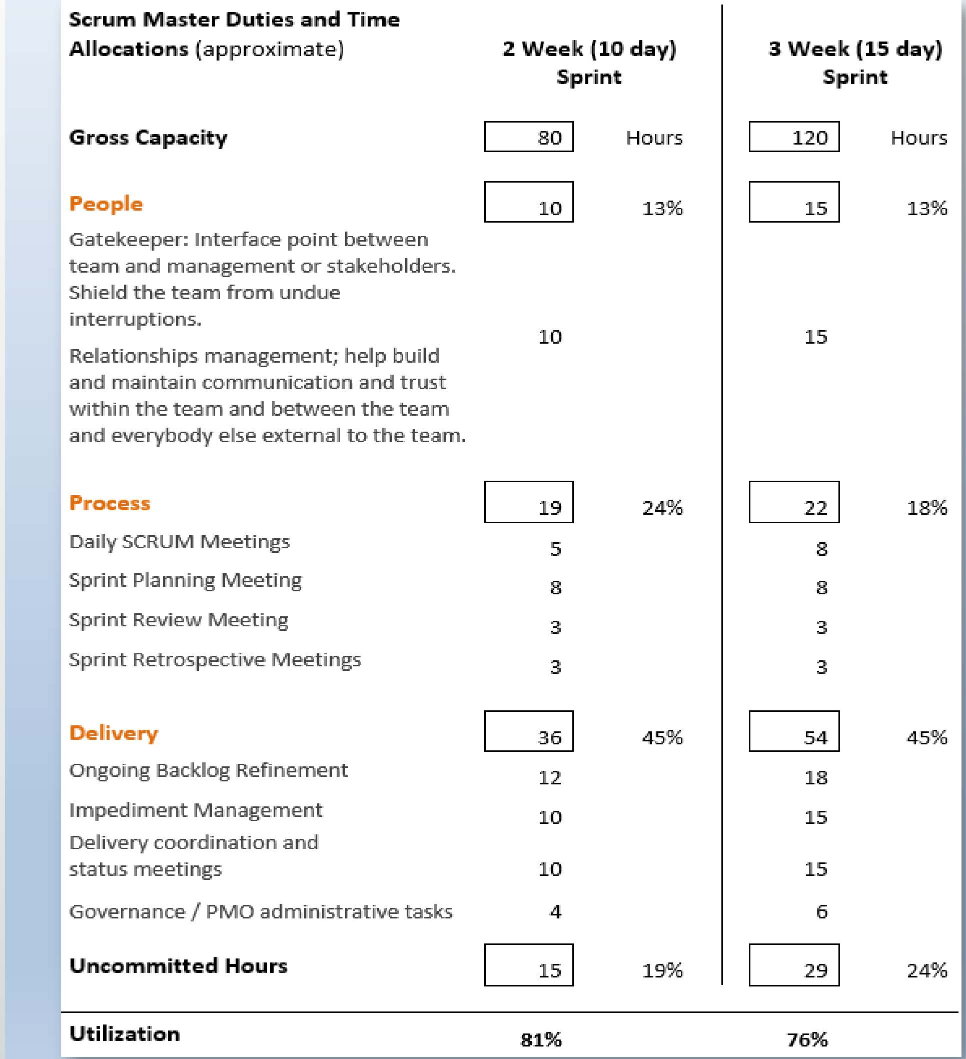
- Scrum process expert
- Ensures consistent team practices
- Coaches team and individuals; to maximize efficiency and quality
- Partners with the Product Owner to maximize alignment
- Assists with logistics, admin, or impediment removal to ensure team can run full throttle.

IT Team

- Typically 7 +/- 2 members
- Armed with skills to deliver increments of working software
- The team is empowered to organize/execute work and to solve problems within their control
- Cross-functional; members learn a bit of how other work is done so they can assist as needed.

Scrum Master

- Duties and Allocations
  - People: Gate keeper; shield the team from undue interruptions and distractions, build and maintain communication between the team and everybody else external to the team.
  - Process: Scrum process activities and meetings.
  - Delivery: Ongoing backlog refinement sessions, impediment management, delivery coordination and status meetings, governance / PMO administrative tasks.



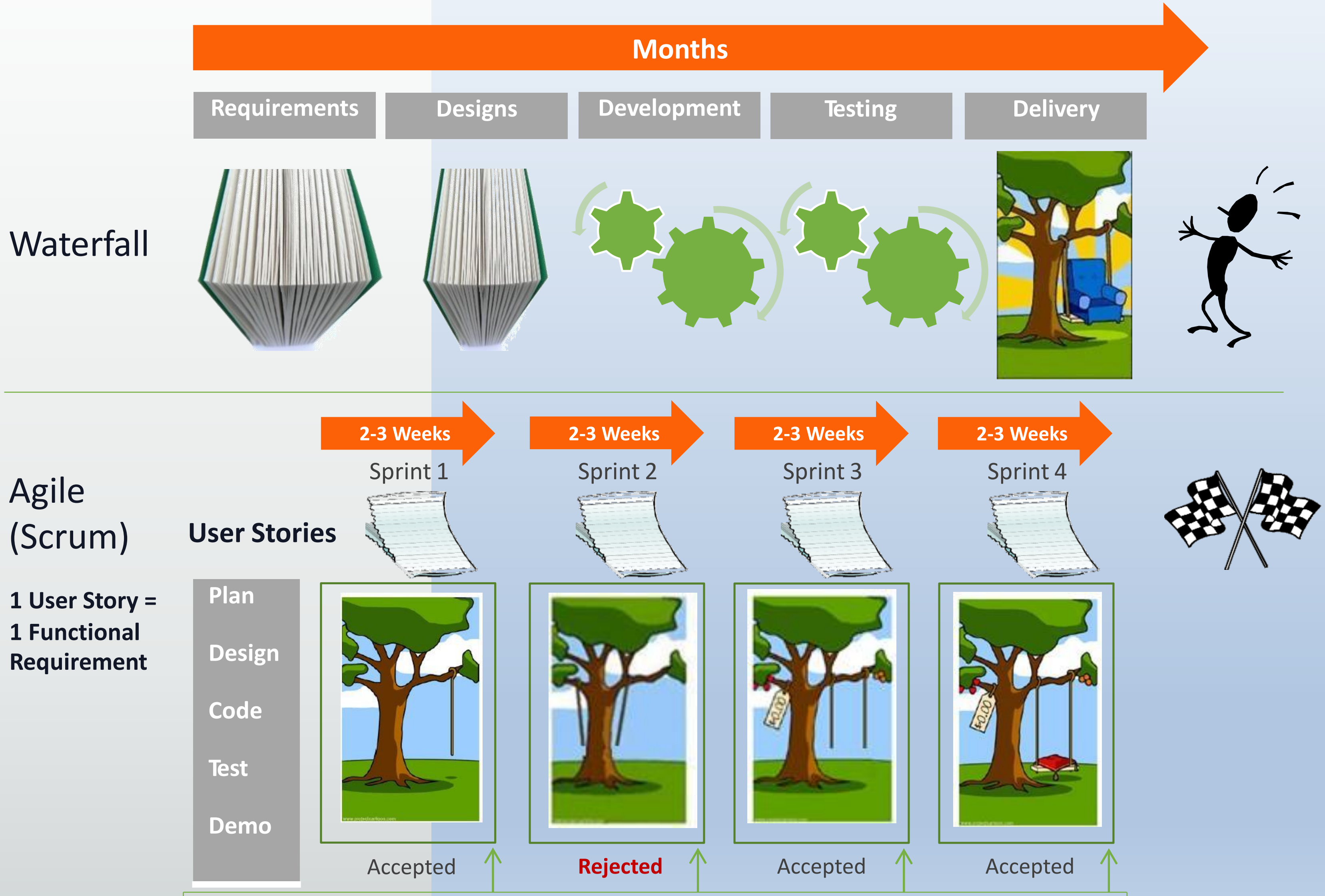
Scrum Master Duties and Time Allocations (approximate)	**2 Week (10 day) Sprint**		**3 Week (15 day) Sprint**	
Gross Capacity	80	Hours	120	Hours
People	10	13%	15	13%
Gatekeeper: Interface point between team and management or stakeholders. Shield the team from undue interruptions.	10		15	
Relationships management; help build and maintain communication and trust within the team and between the team and everybody else external to the team.				
Process	19	24%	22	18%
Daily SCRUM Meetings	5		8	
Sprint Planning Meeting	8		8	
Sprint Review Meeting	3		3	
Sprint Retrospective Meetings	3		3	
Delivery	36	45%	54	45%
Ongoing Backlog Refinement	12		18	
Impediment Management	10		15	
Delivery coordination and status meetings	10		15	
Governance / PMO administrative tasks	4		6	
Uncommitted Hours	15	19%	29	24%
Utilization	**81%**		**76%**	

Business Analyst

- Assists the Product Owner and the Team
    - The Product Owner has a full time job
    - The Product Owner defines the high level functional deliverables (Epics) and priority
    - The BA digs out the detail of each high level functional deliverable into users stories
    - The BA helps create minimum needed designs
    - Pre-Validates the Story as "Done"
    - Helps prepare and execute test plans

Putting It All Together

The Customer Needs This



Months

Requirements | Designs | Development | Testing | Delivery

Waterfall

Agile (Scrum)

User Stories

1 User Story = 1 Functional Requirement

Plan

Design

Code

Test

Demo

2-3 Weeks — Sprint 1 — Accepted

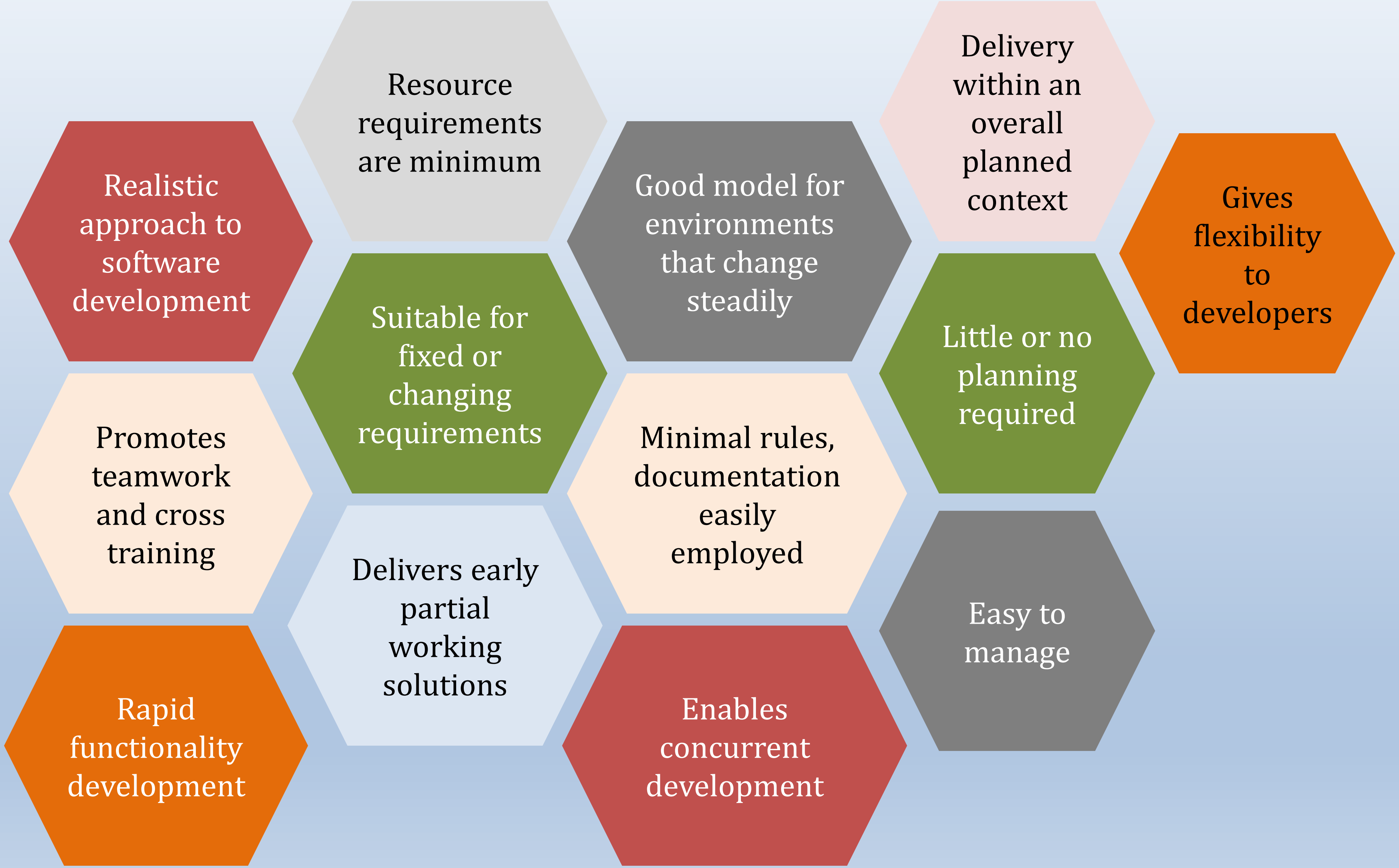
2-3 Weeks — Sprint 2 — **Rejected**

2-3 Weeks — Sprint 3 — Accepted

2-3 Weeks — Sprint 4 — Accepted

- Defects discovered and corrected within <u>each Sprint</u>
- Product Owner / Customer sees functionality <u>each Sprint</u>, accepts or rejects

Agile Model - Advantages



Realistic approach to software development

Promotes teamwork and cross training

Rapid functionality development

Resource requirements are minimum

Suitable for fixed or changing requirements

Delivers early partial working solutions

Good model for environments that change steadily

Minimal rules, documentation easily employed

Enables concurrent development

Delivery within an overall planned context

Little or no planning required

Easy to manage

Gives flexibility to developers

Agile Model - Disadvantages

1 Not suitable for handling complex dependencies.

2 More risk of sustainability, maintainability and extensibility.

3 Overall plan is a must.

4 Strict delivery management to meet deadlines.

5 Depends heavily on customer interaction.

6 Very high individual dependency.

7 Technology transfer is challenging.

8 Lack of documentation.

Thank You!