Course Title: Applied SDLC & Software Testing

Faculty: Srinivas K

Mithila Vartak

Dr. Prithvi Sekhar Pagala

L&T Technology Services

LTTS GLOBAL ENGINEERING ACADEMY

Agenda

1 V Model SDLC

2 Agile Model

3 CI/CD SDLC Model

4 Software Testing

5 Engineering Practices

6 QA Session

Course Objectives & Goal

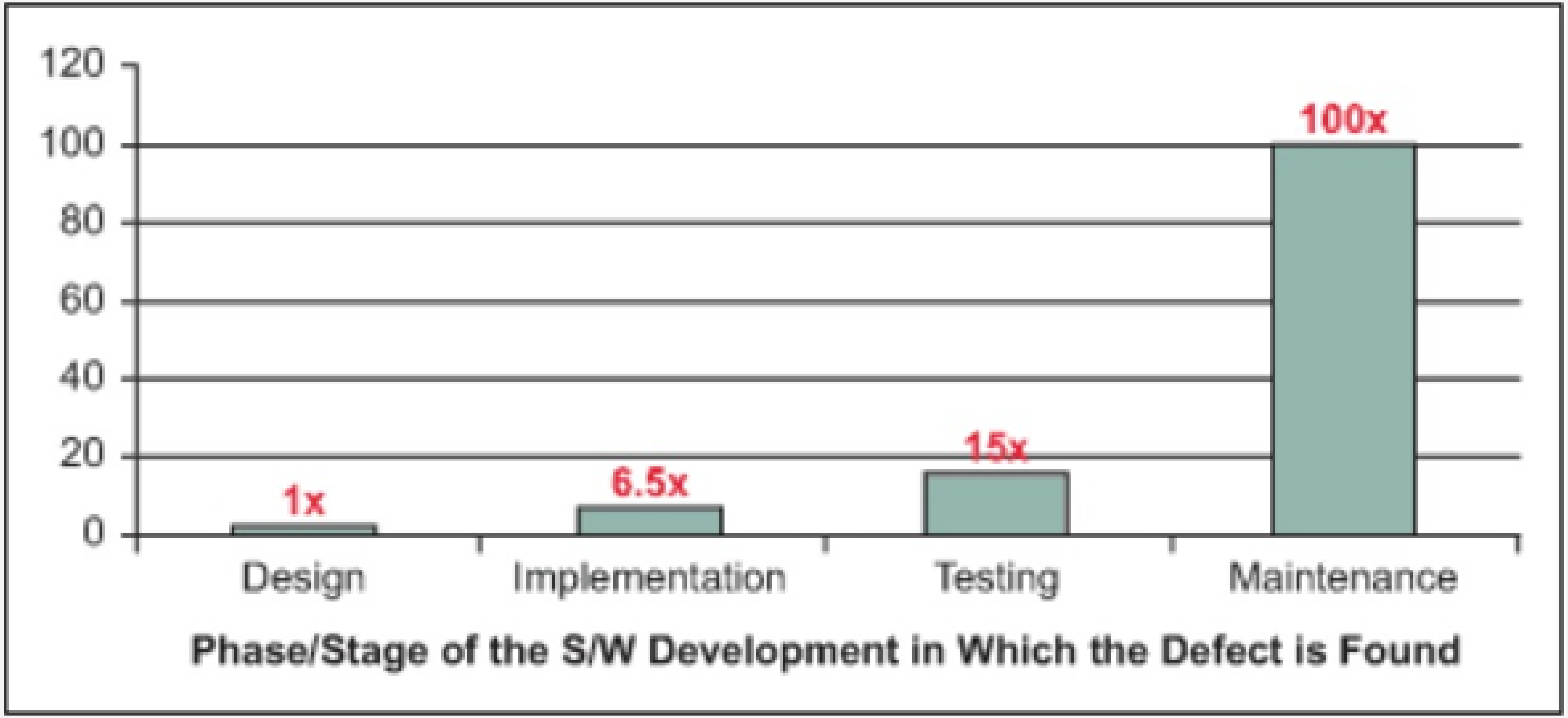
Course Objectives & Goal

- **At the completion of the Course participants will be able to:**
  - Apply core software engineering practices at conceptual level for a given problem.
  - Demonstrate the ability to participate effectively in agile practices/process for software development.
  - Demonstrate the ability to participate effectively in Continuous Integration/Continuous Delivery (CI/CD) for software development
  - Effectively participate in various techniques and processes for building secure and high-quality software.

Verification & Validation Model SDLC

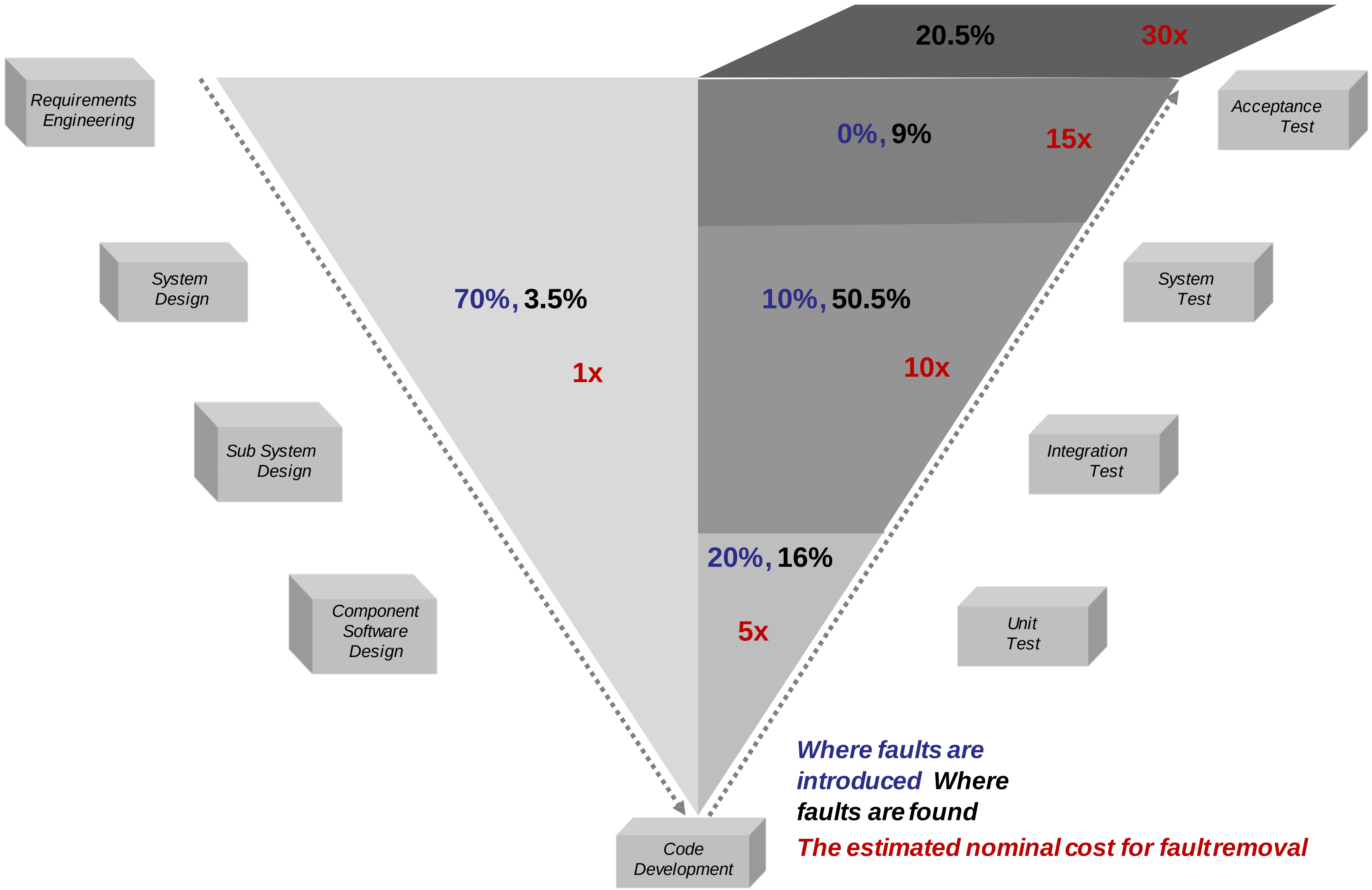
Need for Requirements to Testing - 1

Figure 1: Relative Costs to Fix Software Defects (Source: IBM Systems Sciences Institute)



Source:https://www.bmc.com/blogs/what-is-shift-left-shift-left-testing-explained/ ; https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf

Need for Requirements to Testing - 2

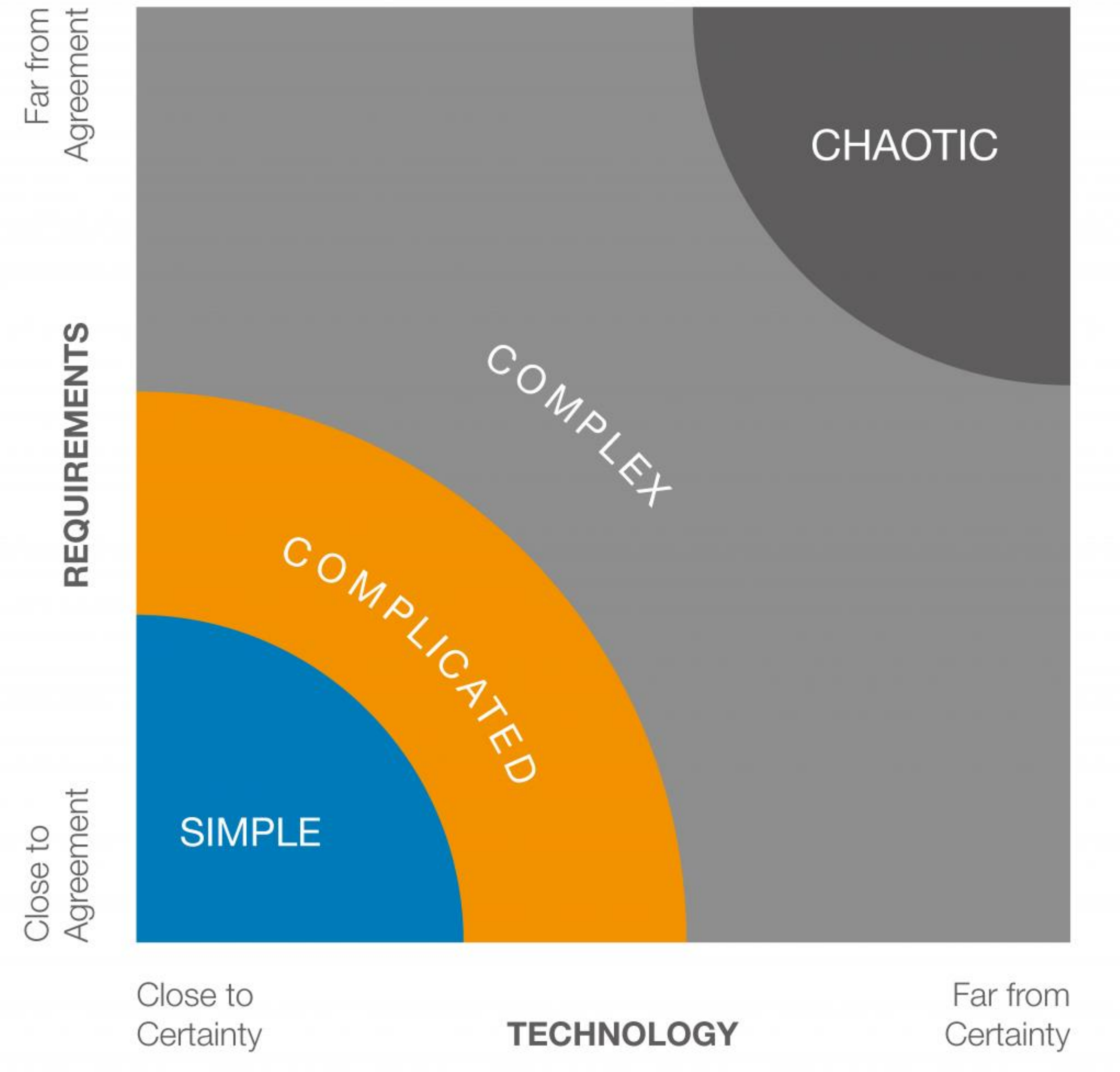


Where faults are introduced ***Where faults are found***

The estimated nominal cost for faultremoval

Source:https://www.bmc.com/blogs/what-is-shift-left-shift-left-testing-explained/ ; https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf

Types of Projects



- Simple – Direct Deliverables
- Complicated – R&D Projects
- Complex – Innovation Projects
- Chaotic – Basic Research

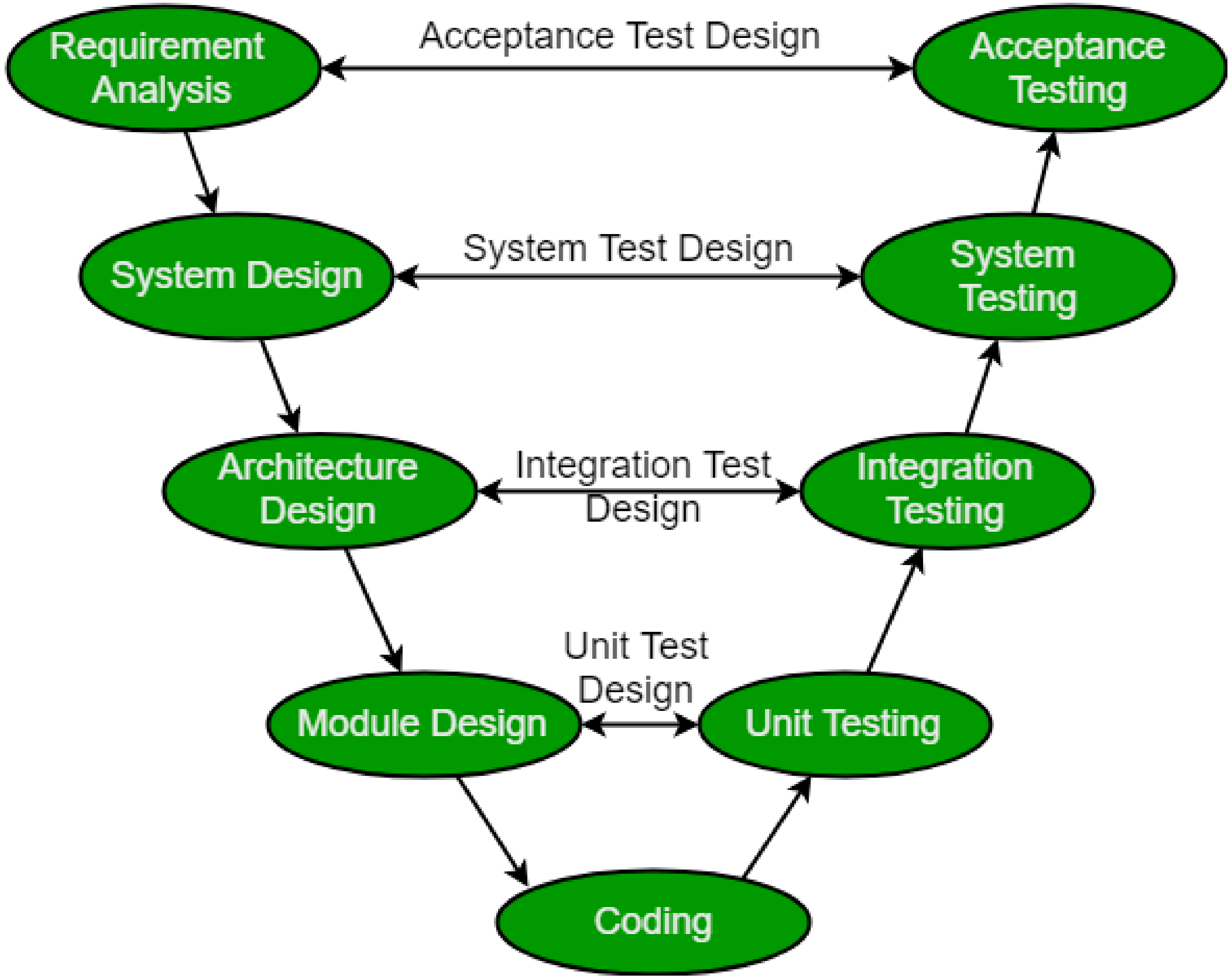
Traditional Approach

- Typical Challenges
  - Assumes correct requirements are known at the start
  - Change is risky and costly, so new needs often can’t be met
  - Hand-offs between groups require work, and invite disconnects
  - Testing is done towards the end, when defects are more expensive
  - No working Model until late in the process, so lots of surprises!

- Why do we have change requests?
  - End user not clear about end result
  - Assumptions differ
  - Lack of clarity within development team
  - Market changes
  - Customer expectation changes

V Model SDLC

- **V Model**



Applications:

- Requirements are well defined, clearly documented and fixed.
- Product definition is stable.
- Technology is not dynamic and is well understood by the project team.
- There are no ambiguous or undefined requirements.
- The project is short.

Source: https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/

Principles of V – Model SDLC

Large to Small

- Testing is done in a hierarchical perspective

Data/Process Integrity

- Successful design of any project requires the incorporation and cohesion of both data and processes
- Process elements must be identified at each and every requirements

Cross Referencing

- Direct correlation between requirements and corresponding testing activity is known as cross-referencing

Tangible Documentation

- Every project needs to create a document
- Documentation is used to maintaining the application once it is available in a production environment.

Advantages & Disadvantages of V – Model SDLC

Advantages

- This is a highly disciplined model and Phases are completed one at a time.
- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.
- This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
- It enables project management to track progress accurately.

Disadvantages

- High risk and uncertainty.
- It is not a good for complex and object-oriented projects.
- It is not suitable for projects where requirements are not clear and contains high risk of changing.
- This model does not support iteration of phases.
- It does not easily handle concurrent events.

Applications of V – Model SDLC

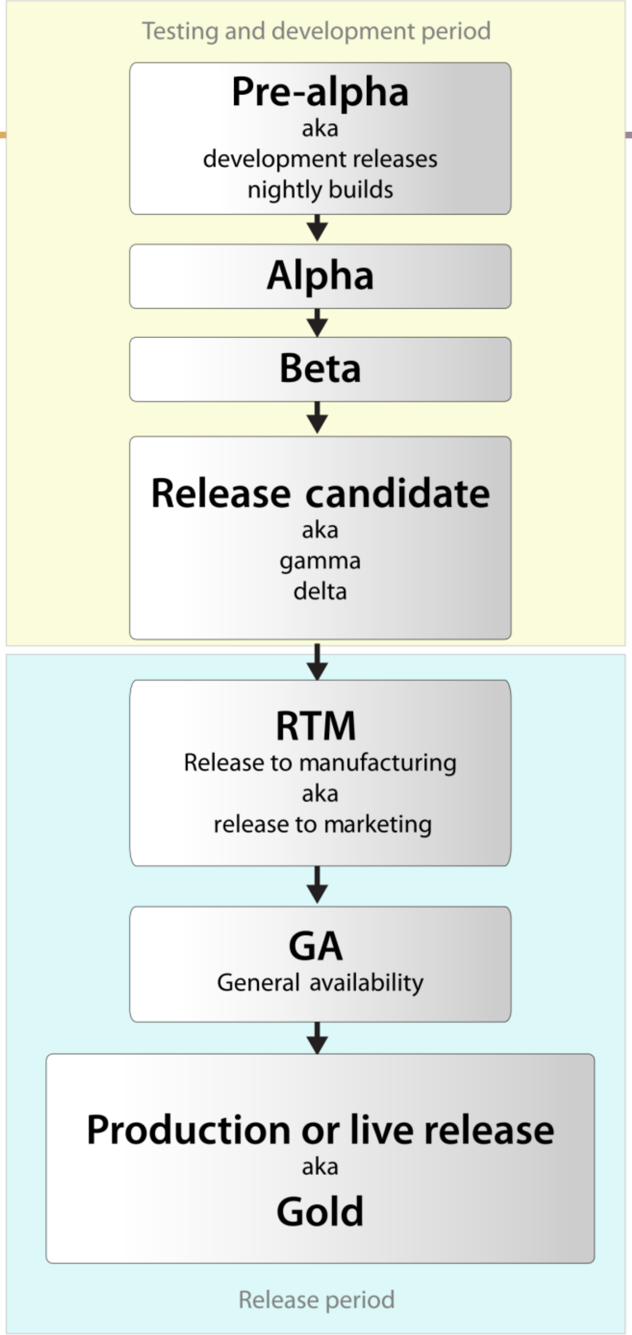
Medical Device Industry

Automotive Industry

Industrial Products Industry

Domain Specific – Standards and Compliance

Release Cycle



Testing and development period

Pre-alpha
aka
development releases
nightly builds

Alpha

Beta

Release candidate
aka
gamma
delta

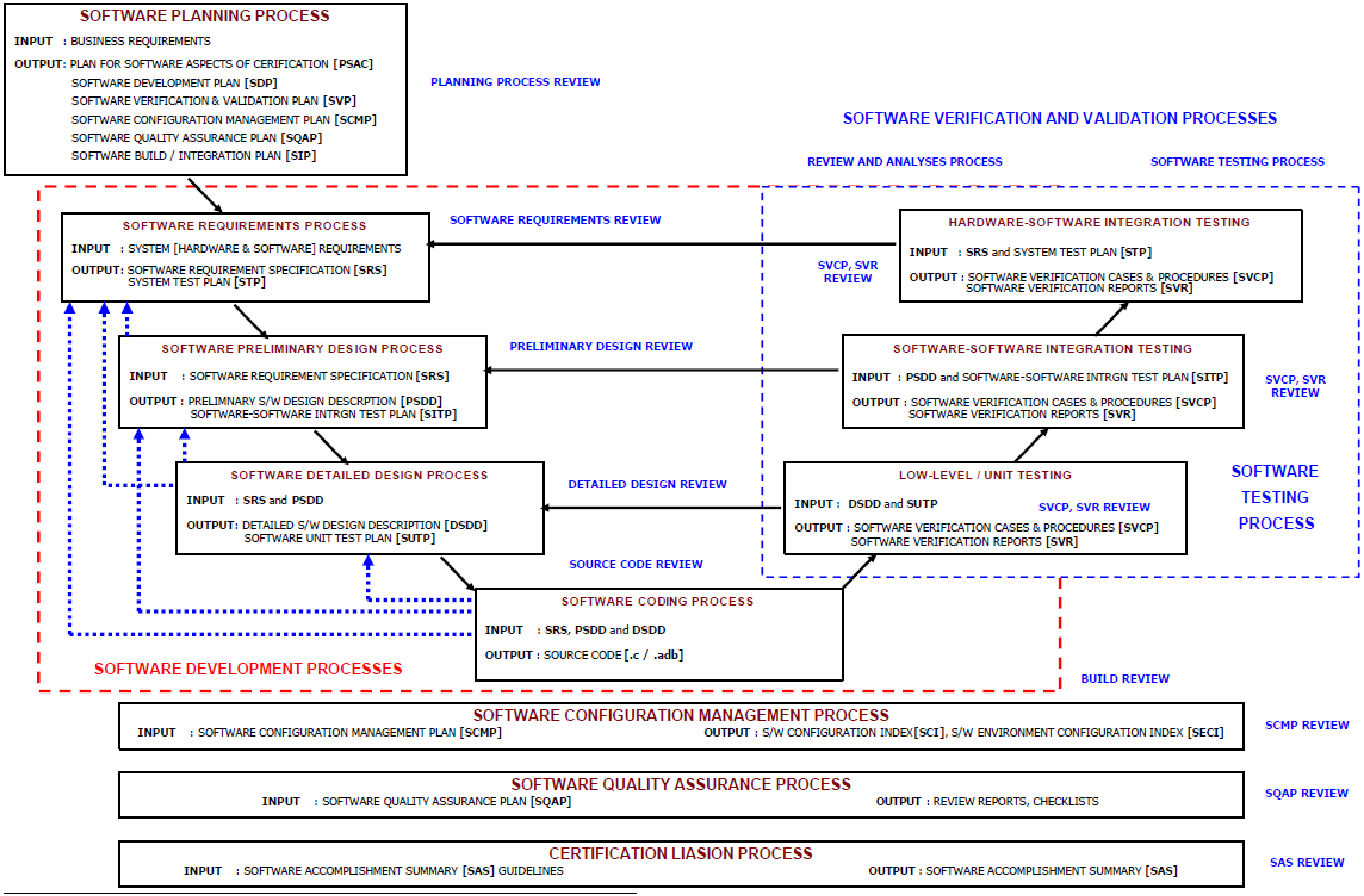
RTM
Release to manufacturing
aka
release to marketing

GA
General availability

Production or live release
aka
Gold

Release period

Aero - DO178B/C – V Model – SDLC & STLC..



SOFTWARE PLANNING PROCESS

INPUT : BUSINESS REQUIREMENTS

OUTPUT: PLAN FOR SOFTWARE ASPECTS OF CERIFICATION [PSAC]
SOFTWARE DEVELOPMENT PLAN [SDP]
SOFTWARE VERIFICATION & VALIDATION PLAN [SVP]
SOFTWARE CONFIGURATION MANAGEMENT PLAN [SCMP]
SOFTWARE QUALITY ASSURANCE PLAN [SQAP]
SOFTWARE BUILD / INTEGRATION PLAN [SIP]

PLANNING PROCESS REVIEW

SOFTWARE VERIFICATION AND VALIDATION PROCESSES

REVIEW AND ANALYSES PROCESS

SOFTWARE TESTING PROCESS

SOFTWARE REQUIREMENTS PROCESS

INPUT : SYSTEM [HARDWARE & SOFTWARE] REQUIREMENTS

OUTPUT: SOFTWARE REQUIREMENT SPECIFICATION [SRS]
SYSTEM TEST PLAN [STP]

SOFTWARE REQUIREMENTS REVIEW

HARDWARE-SOFTWARE INTEGRATION TESTING

INPUT : SRS and SYSTEM TEST PLAN [STP]

OUTPUT : SOFTWARE VERIFICATION CASES & PROCEDURES [SVCP]
SOFTWARE VERIFICATION REPORTS [SVR]

SVCP, SVR REVIEW

SOFTWARE PRELIMINARY DESIGN PROCESS

INPUT : SOFTWARE REQUIREMENT SPECIFICATION [SRS]

OUTPUT : PRELIMNARY S/W DESIGN DESCRPTION [PSDD]
SOFTWARE-SOFTWARE INTRGN TEST PLAN [SITP]

PRELIMINARY DESIGN REVIEW

SOFTWARE-SOFTWARE INTEGRATION TESTING

INPUT : PSDD and SOFTWARE-SOFTWARE INTRGN TEST PLAN [SITP]

OUTPUT : SOFTWARE VERIFICATION CASES & PROCEDURES [SVCP]
SOFTWARE VERIFICATION REPORTS [SVR]

SVCP, SVR REVIEW

SOFTWARE DETAILED DESIGN PROCESS

INPUT : SRS and PSDD

OUTPUT: DETAILED S/W DESIGN DESCRIPTION [DSDD]
SOFTWARE UNIT TEST PLAN [SUTP]

DETAILED DESIGN REVIEW

LOW-LEVEL / UNIT TESTING

INPUT : DSDD and SUTP

SVCP, SVR REVIEW

OUTPUT : SOFTWARE VERIFICATION CASES & PROCEDURES [SVCP]
SOFTWARE VERIFICATION REPORTS [SVR]

SOFTWARE TESTING PROCESS

SOURCE CODE REVIEW

SOFTWARE CODING PROCESS

INPUT : SRS, PSDD and DSDD

OUTPUT : SOURCE CODE [.c / .adb]

SOFTWARE DEVELOPMENT PROCESSES

BUILD REVIEW

SOFTWARE CONFIGURATION MANAGEMENT PROCESS

INPUT : SOFTWARE CONFIGURATION MANAGEMENT PLAN [SCMP]

OUTPUT : S/W CONFIGURATION INDEX[SCI], S/W ENVIRONMENT CONFIGURATION INDEX [SECI]

SCMP REVIEW

SOFTWARE QUALITY ASSURANCE PROCESS

INPUT : SOFTWARE QUALITY ASSURANCE PLAN [SQAP]

OUTPUT : REVIEW REPORTS, CHECKLISTS

SQAP REVIEW

CERTIFICATION LIASION PROCESS

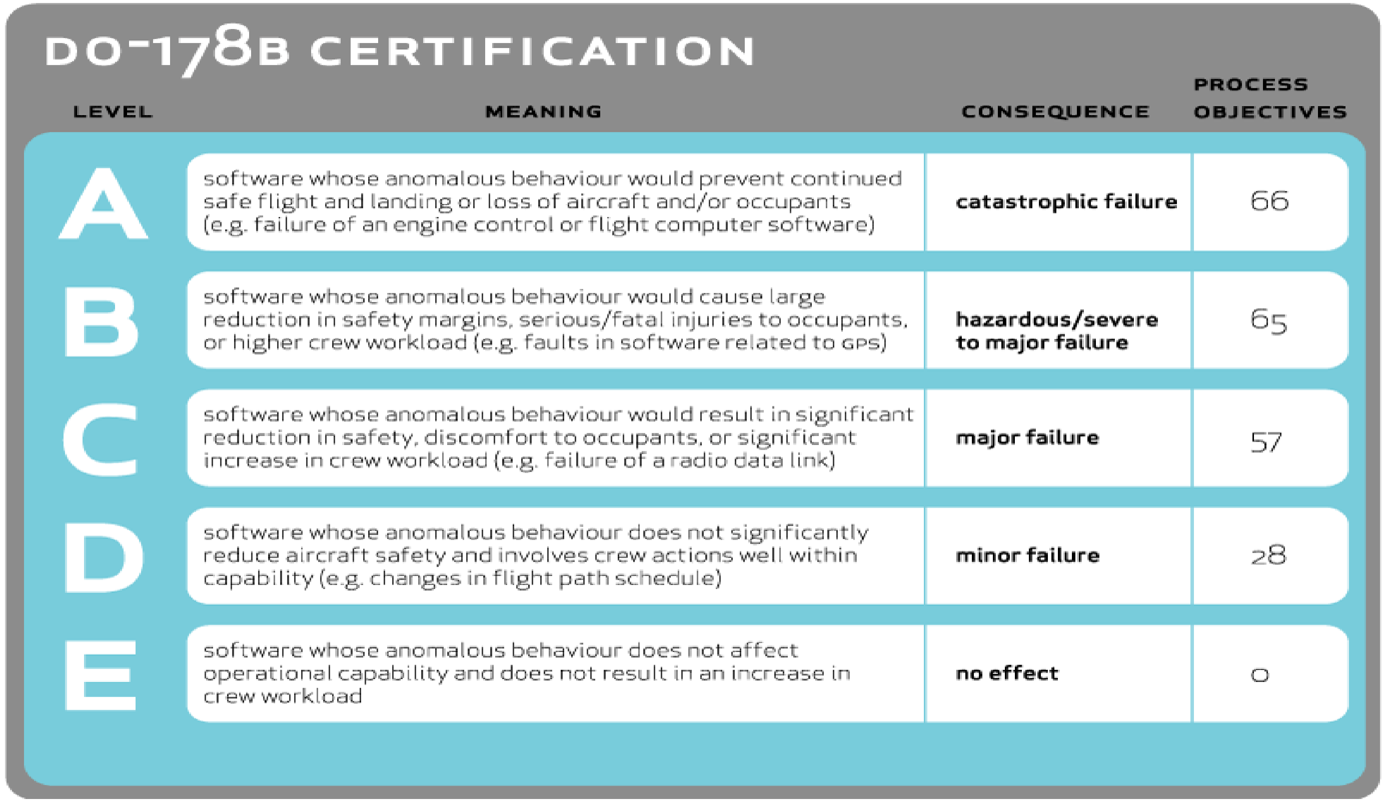
INPUT : SOFTWARE ACCOMPLISHMENT SUMMARY [SAS] GUIDELINES

OUTPUT : SOFTWARE ACCOMPLISHMENT SUMMARY [SAS]

SAS REVIEW

Aero - DO178B Software Levels & RBT Methods – V Model

- DO-178B requires that all system/software requirements be mapped to one of the five software levels.

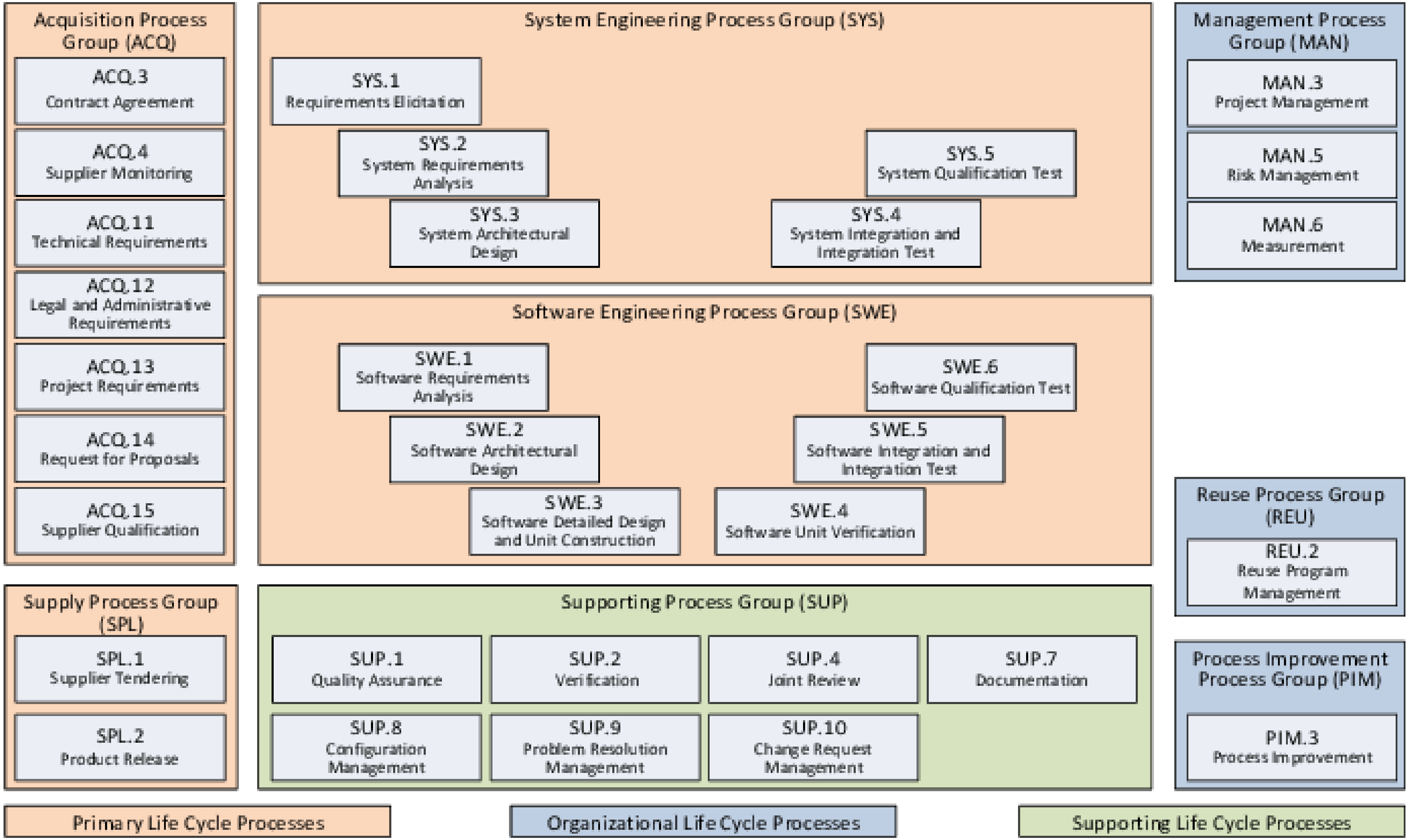


DO-178B CERTIFICATION

LEVEL	MEANING	CONSEQUENCE	PROCESS OBJECTIVES
A	software whose anomalous behaviour would prevent continued safe flight and landing or loss of aircraft and/or occupants (e.g. failure of an engine control or flight computer software)	**catastrophic failure**	66
B	software whose anomalous behaviour would cause large reduction in safety margins, serious/fatal injuries to occupants, or higher crew workload (e.g. faults in software related to GPS)	**hazardous/severe to major failure**	65
C	software whose anomalous behaviour would result in significant reduction in safety, discomfort to occupants, or significant increase in crew workload (e.g. failure of a radio data link)	**major failure**	57
D	software whose anomalous behaviour does not significantly reduce aircraft safety and involves crew actions well within capability (e.g. changes in flight path schedule)	**minor failure**	28
E	software whose anomalous behaviour does not affect operational capability and does not result in an increase in crew workload	**no effect**	0

- Types of Software Testing Requirements based on software levels definitions and failure condition categorization of DO-178B as above.
  - Level E: no specific requirement.
  - Level D: 100% Requirement Coverage.
  - Level C: Level D + 100% Statement Coverage.
  - Level B: Level C + 100% Decision Coverage.
  - Level A: Level B + 100% Modified Condition / Decision Coverage (MC/DC).

Automotive - ASPICE – Model 3.0



Acquisition Process Group (ACQ)

- ACQ.3 Contract Agreement
- ACQ.4 Supplier Monitoring
- ACQ.11 Technical Requirements
- ACQ.12 Legal and Administrative Requirements
- ACQ.13 Project Requirements
- ACQ.14 Request for Proposals
- ACQ.15 Supplier Qualification

System Engineering Process Group (SYS)

- SYS.1 Requirements Elicitation
- SYS.2 System Requirements Analysis
- SYS.3 System Architectural Design
- SYS.4 System Integration and Integration Test
- SYS.5 System Qualification Test

Management Process Group (MAN)

- MAN.3 Project Management
- MAN.5 Risk Management
- MAN.6 Measurement

Software Engineering Process Group (SWE)

- SWE.1 Software Requirements Analysis
- SWE.2 Software Architectural Design
- SWE.3 Software Detailed Design and Unit Construction
- SWE.4 Software Unit Verification
- SWE.5 Software Integration and Integration Test
- SWE.6 Software Qualification Test

Reuse Process Group (REU)

- REU.2 Reuse Program Management

Supply Process Group (SPL)

- SPL.1 Supplier Tendering
- SPL.2 Product Release

Supporting Process Group (SUP)

- SUP.1 Quality Assurance
- SUP.2 Verification
- SUP.4 Joint Review
- SUP.7 Documentation
- SUP.8 Configuration Management
- SUP.9 Problem Resolution Management
- SUP.10 Change Request Management

Process Improvement Process Group (PIM)

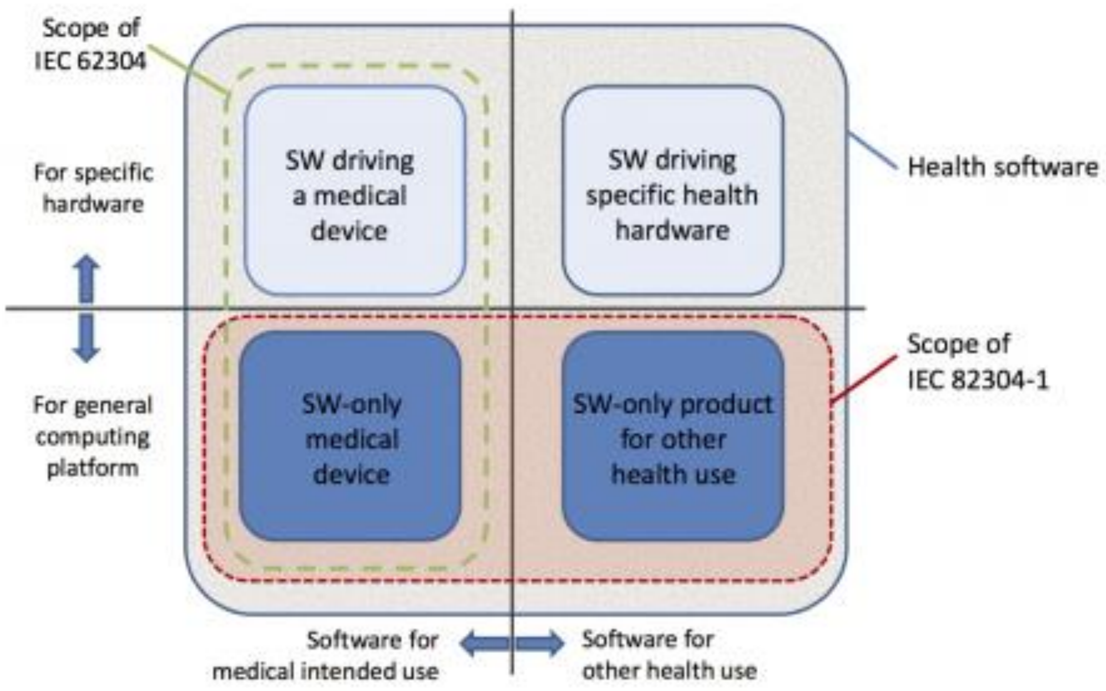
- PIM.3 Process Improvement

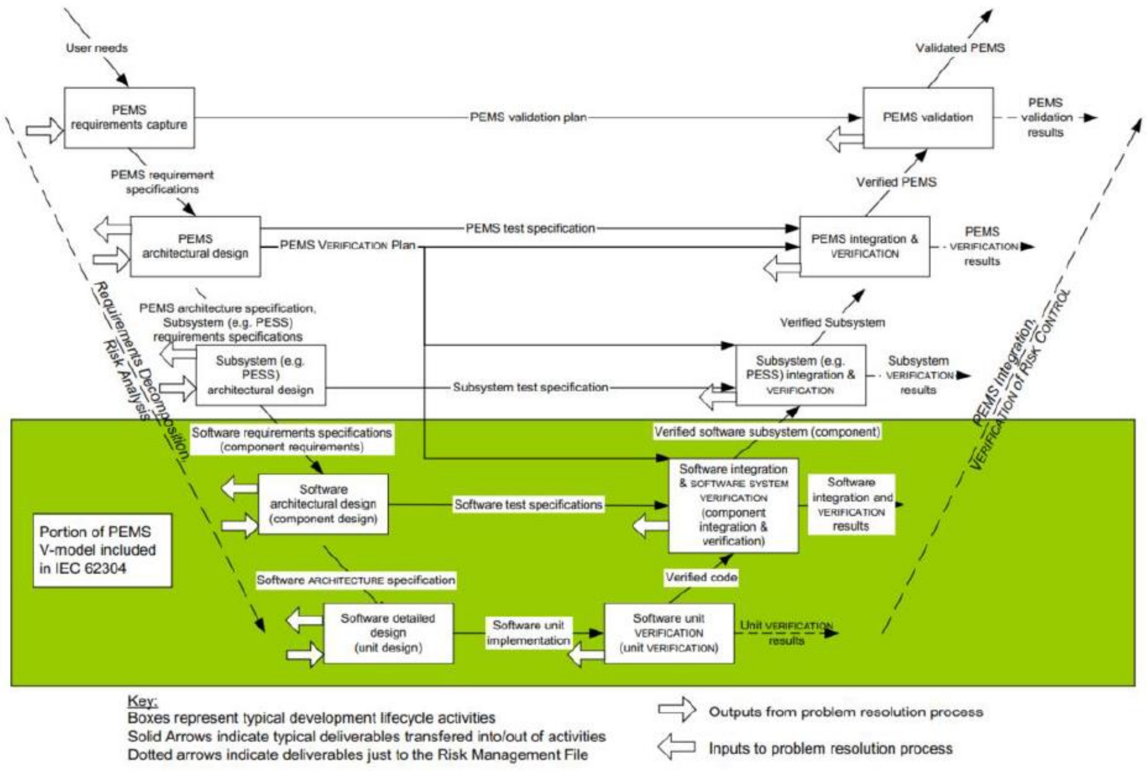
Primary Life Cycle Processes

Organizational Life Cycle Processes

Supporting Life Cycle Processes

Medical – IEC62304 and IEC 82304





Key:
Boxes represent typical development lifecycle activities
Solid Arrows indicate typical deliverables transfered into/out of activities
Dotted arrows indicate deliverables just to the Risk Management File

Source - https://blog.cm-dm.com/post/2016/01/15/IEC-82304-1-latest-news-about-the-standard-on-Health-Software

Agile Software Development

Activity - Folding Airplanes

- How good can you get at making paper airplanes?
- Each airplane must be made from ¼ of an A4 sheet
- Each plane must have a blunt tip..
- Each airplane must tested and shown to fly 3 meters in the testing area. Each plane shall only be tested once.
- Only successfully tested planes count towards your goal
- Planes may only be flown in the testing area.
- Each team member may only do 1 “fold” of the paper at a time.
- You must then pass the airplane to another team member to do the next fold.

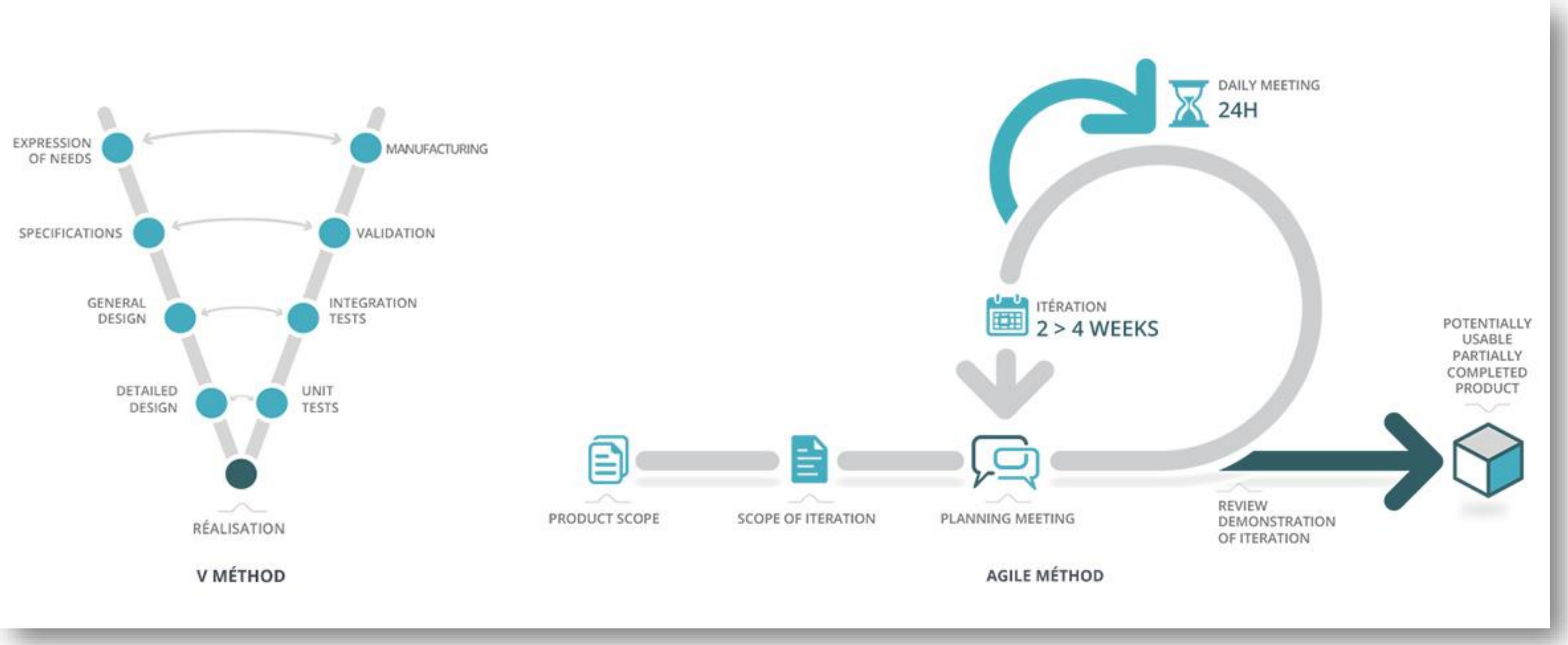
Activity – DIVIDE INTO 3 TEAMS ALL EQUAL SIZE

- Plan activities for a TV Show covering latest events – a Talk Show

- Requirements-
  - 30min show duration / week
  - Based on latest news and updates
  - Requires involvement of industry experts
  - Has a fixed pattern
  - Roles involves – writer, actors, anchor, producer, director, etc….

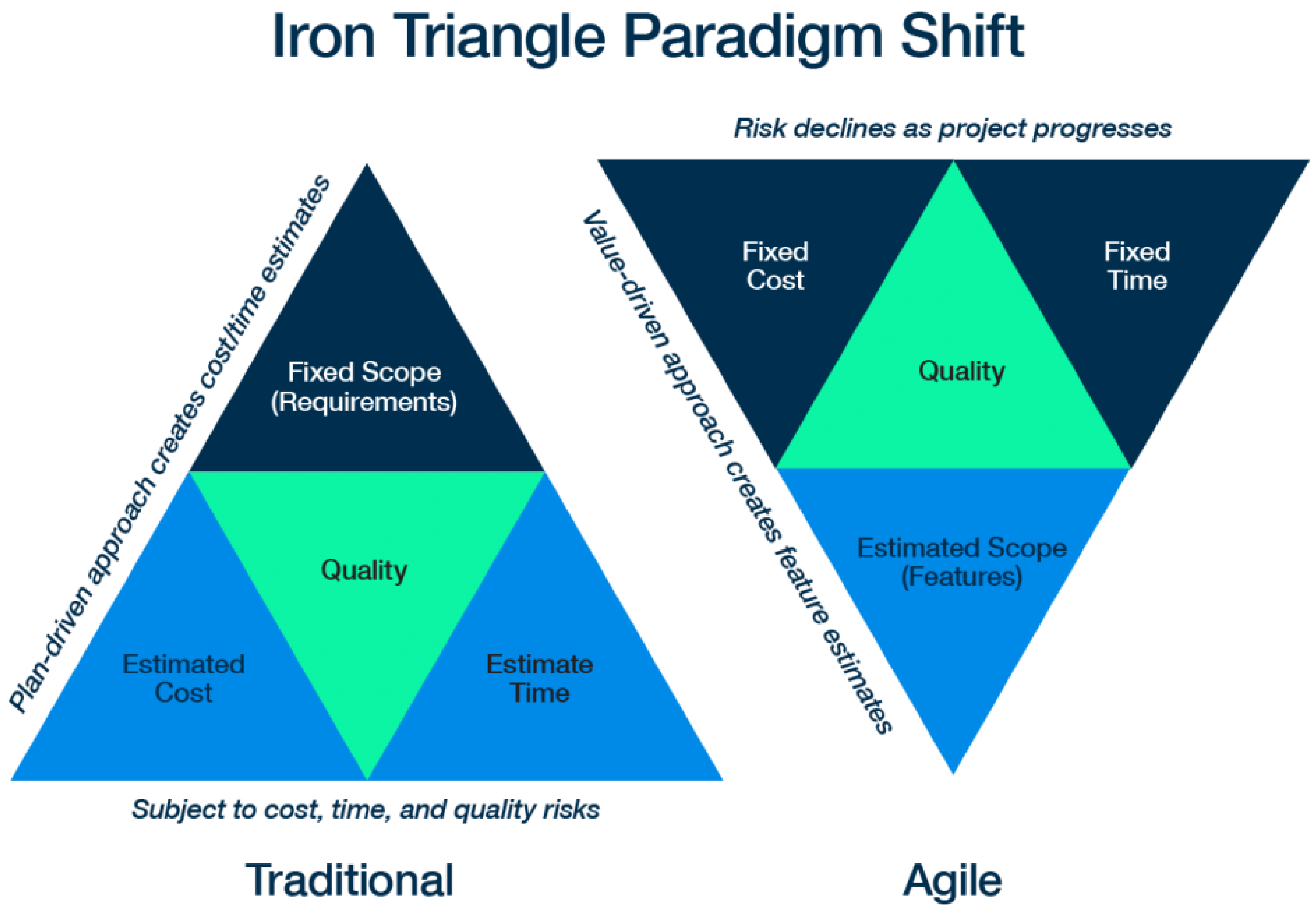
- Objective: PREPARE AN EXECUTION PLAN FOR NEXT 2 MONTHS BY IDENTIFYING ALL KEY ACTIVITIES TO BE PLANNED IN EACH WEEK

V- Model Vs Agile Model



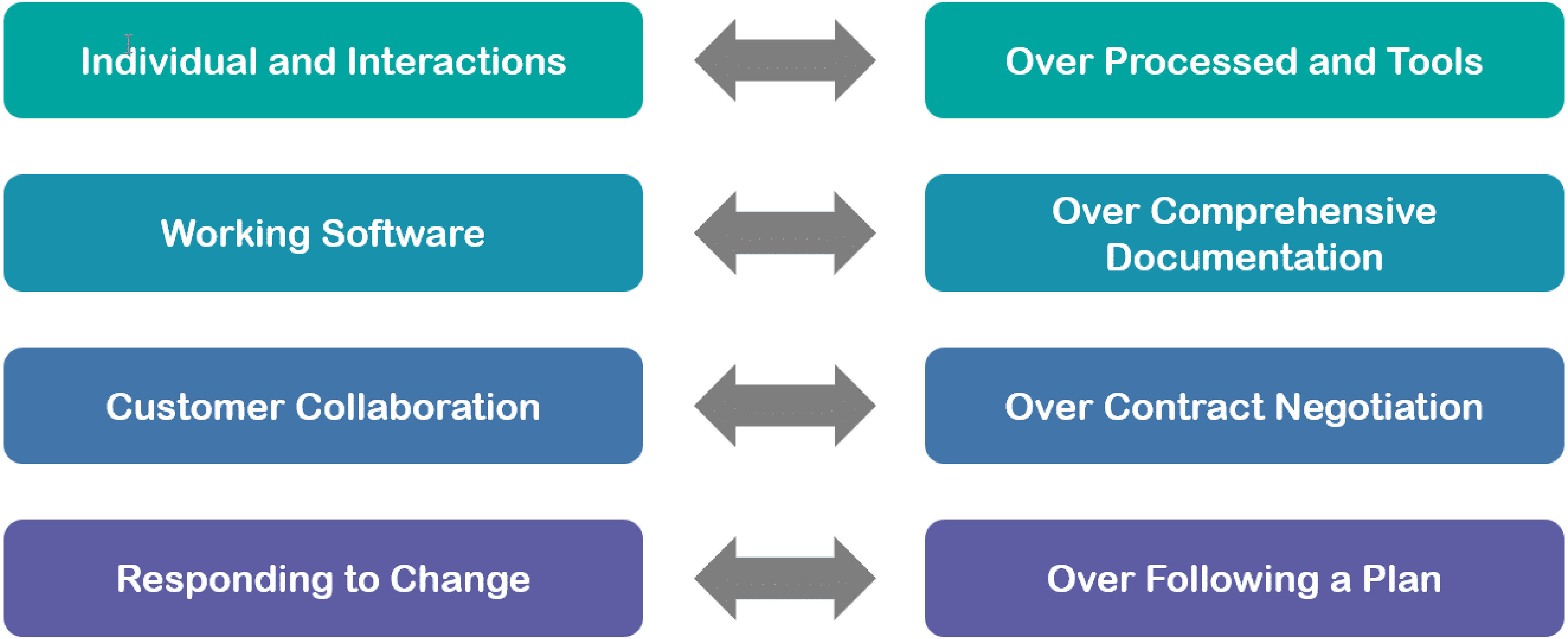
Source: https://www.cegedim-insurance.com/en-EN/solutions-services/our-added-value/Pages/agile-methodology.aspx

Paradigm Shift


Iron Triangle Paradigm Shift
Plan-driven approach creates cost/time estimates
Fixed Scope (Requirements)
Quality
Estimated Cost
Estimate Time
Subject to cost, time, and quality risks
Traditional
Risk declines as project progresses
Value-driven approach creates feature estimates
Fixed Cost
Fixed Time
Quality
Estimated Scope (Features)
Agile

Source : https://www.verticalmotion.ca/2019/05/08/how-agile-keeps-projects-on-track-on-time-on-budget/

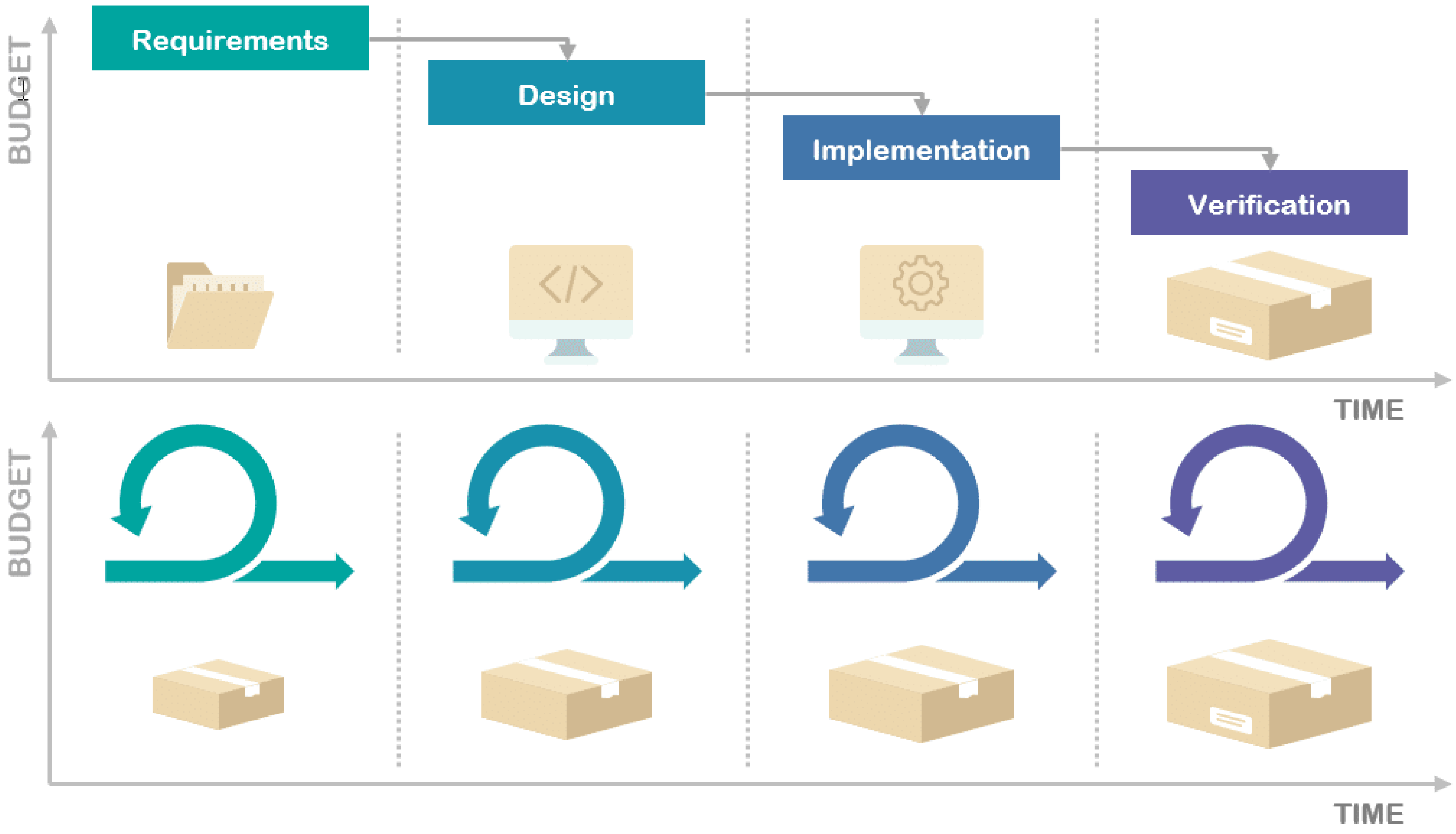
Agile Manifesto – 4 Core Values



Individual and Interactions	Over Processed and Tools
Working Software	Over Comprehensive Documentation
Customer Collaboration	Over Contract Negotiation
Responding to Change	Over Following a Plan

Source and Resource: https://www.toolsqa.com/agile/agile-manifesto/

Incremental and Iterative Approach



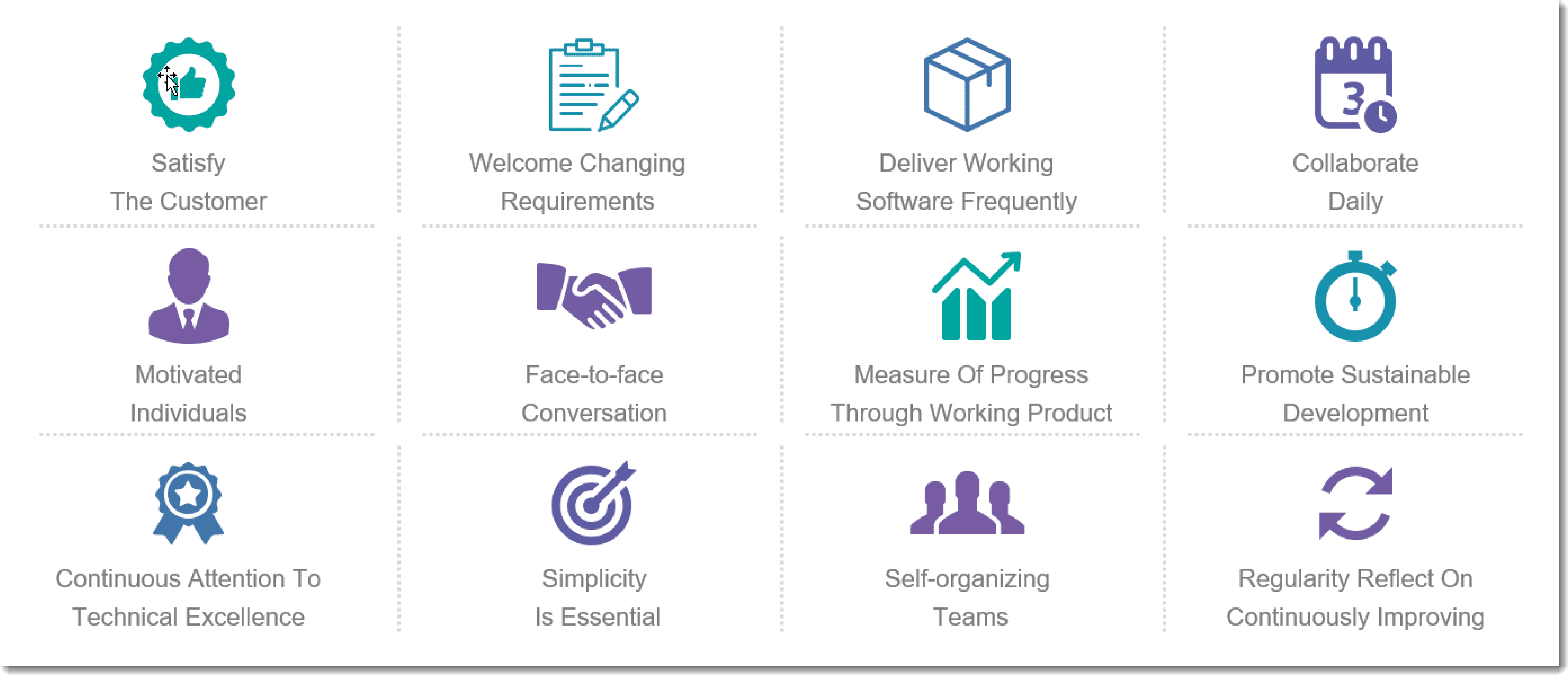
WATERFALL

Development of the software flow sequentially from start point to the end point.

AGILE

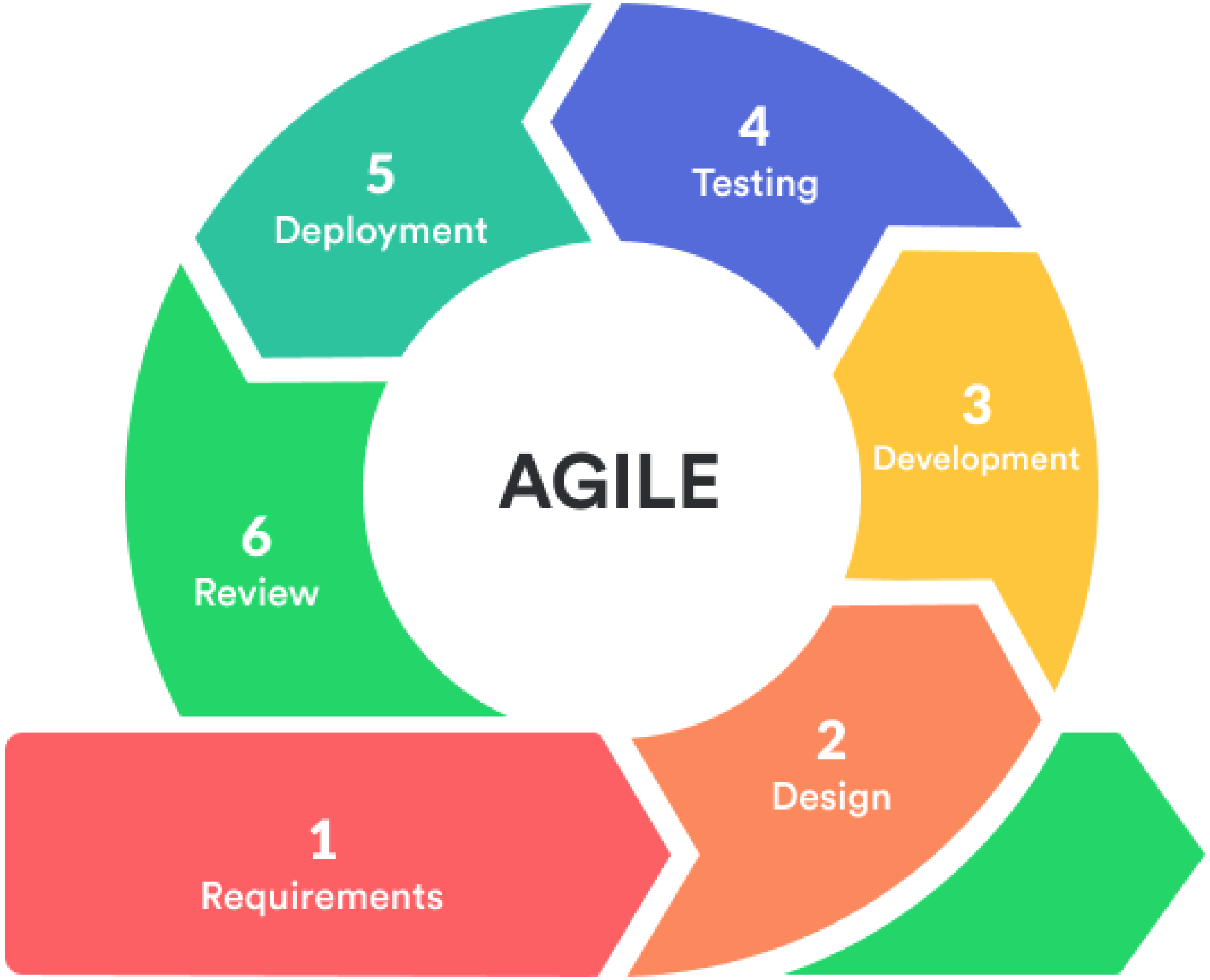
Agile method proposes and increment and iterative approach to software design.

Agile Manifesto – 12 Principles



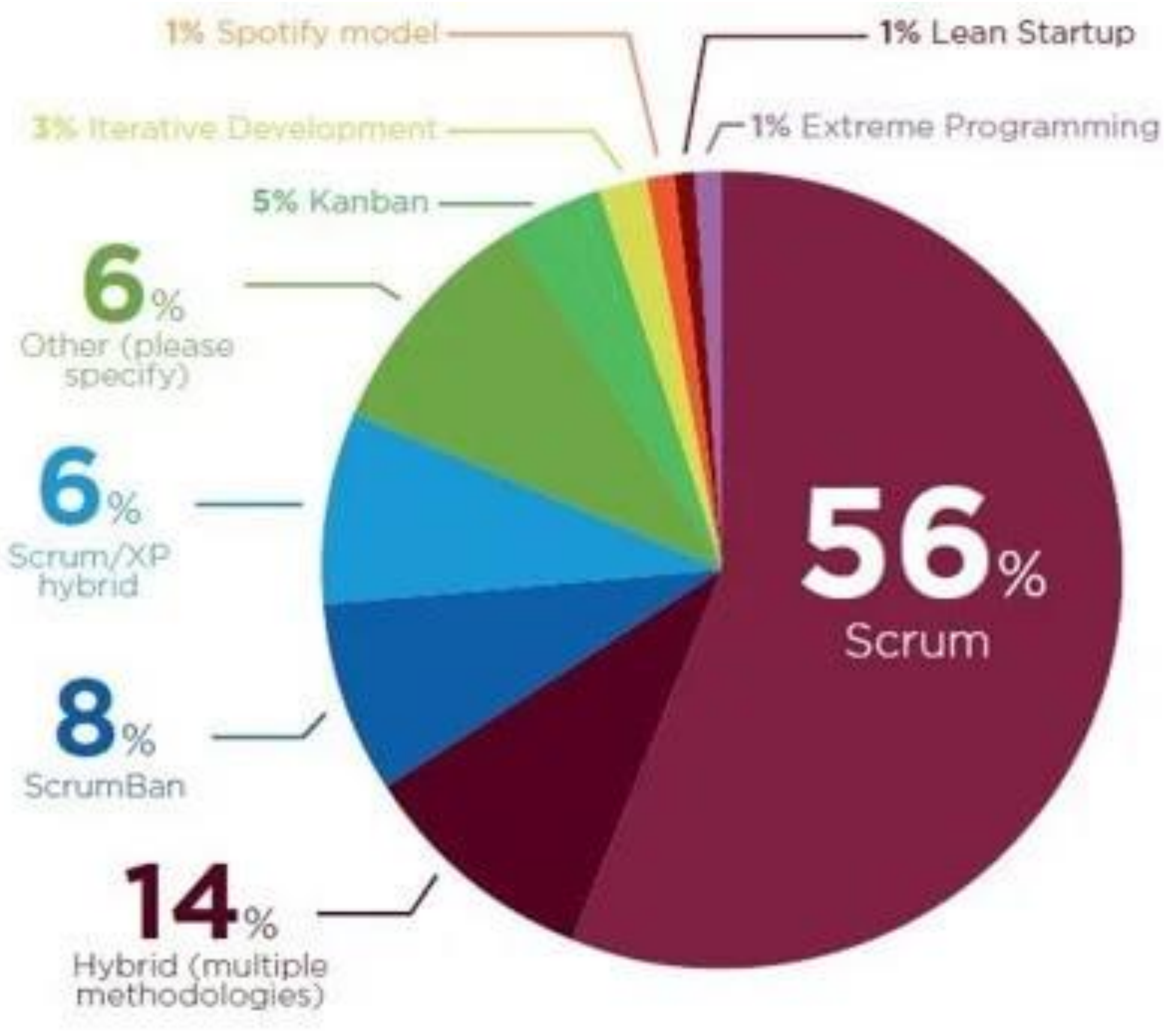
Satisfy The Customer	Welcome Changing Requirements	Deliver Working Software Frequently	Collaborate Daily
Motivated Individuals	Face-to-face Conversation	Measure Of Progress Through Working Product	Promote Sustainable Development
Continuous Attention To Technical Excellence	Simplicity Is Essential	Self-organizing Teams	Regularity Reflect On Continuously Improving

Phases in Agile Model



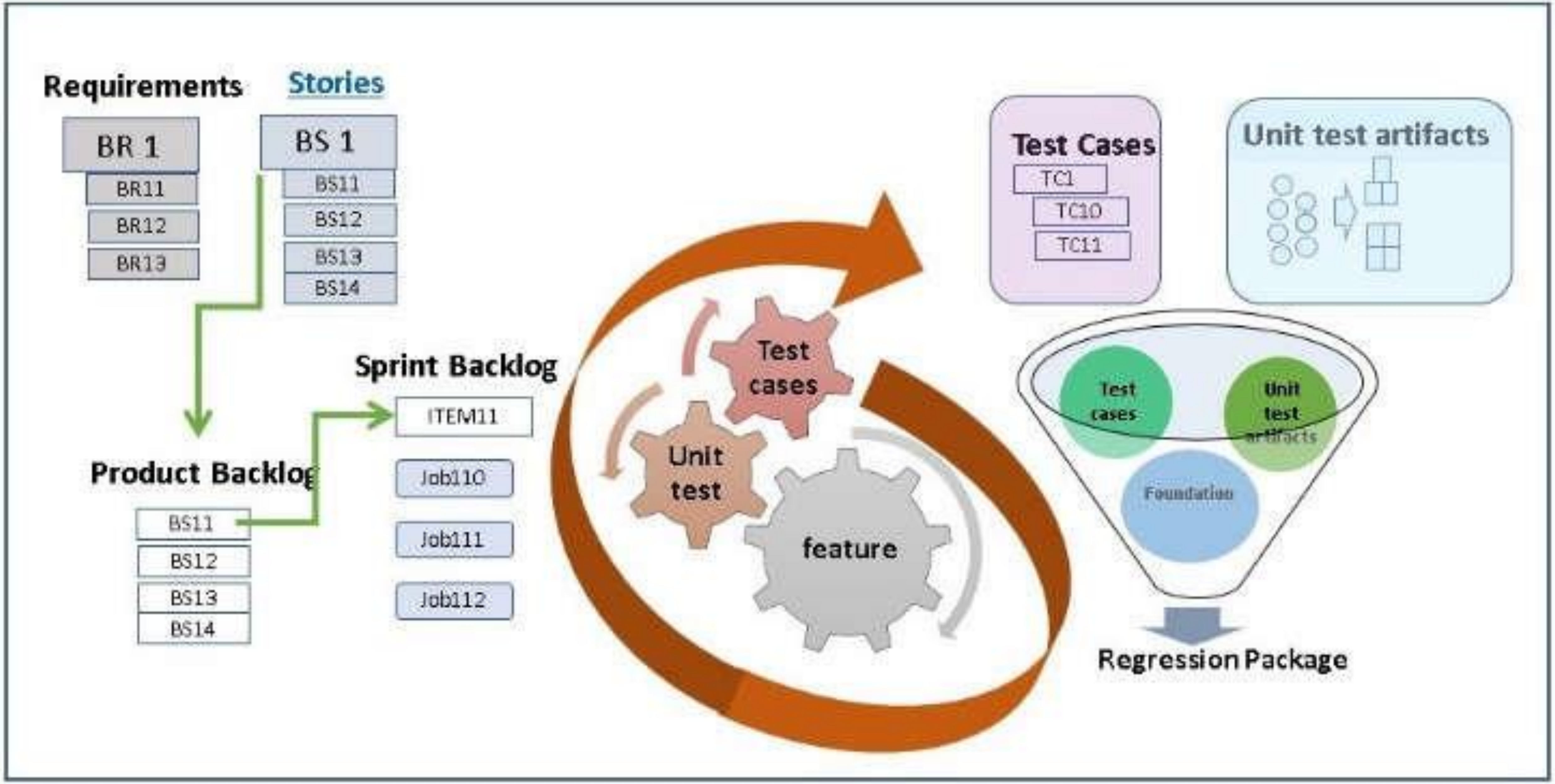
Source: https://mlsdev.com/blog/agile-sdlc

Agile Methodologies

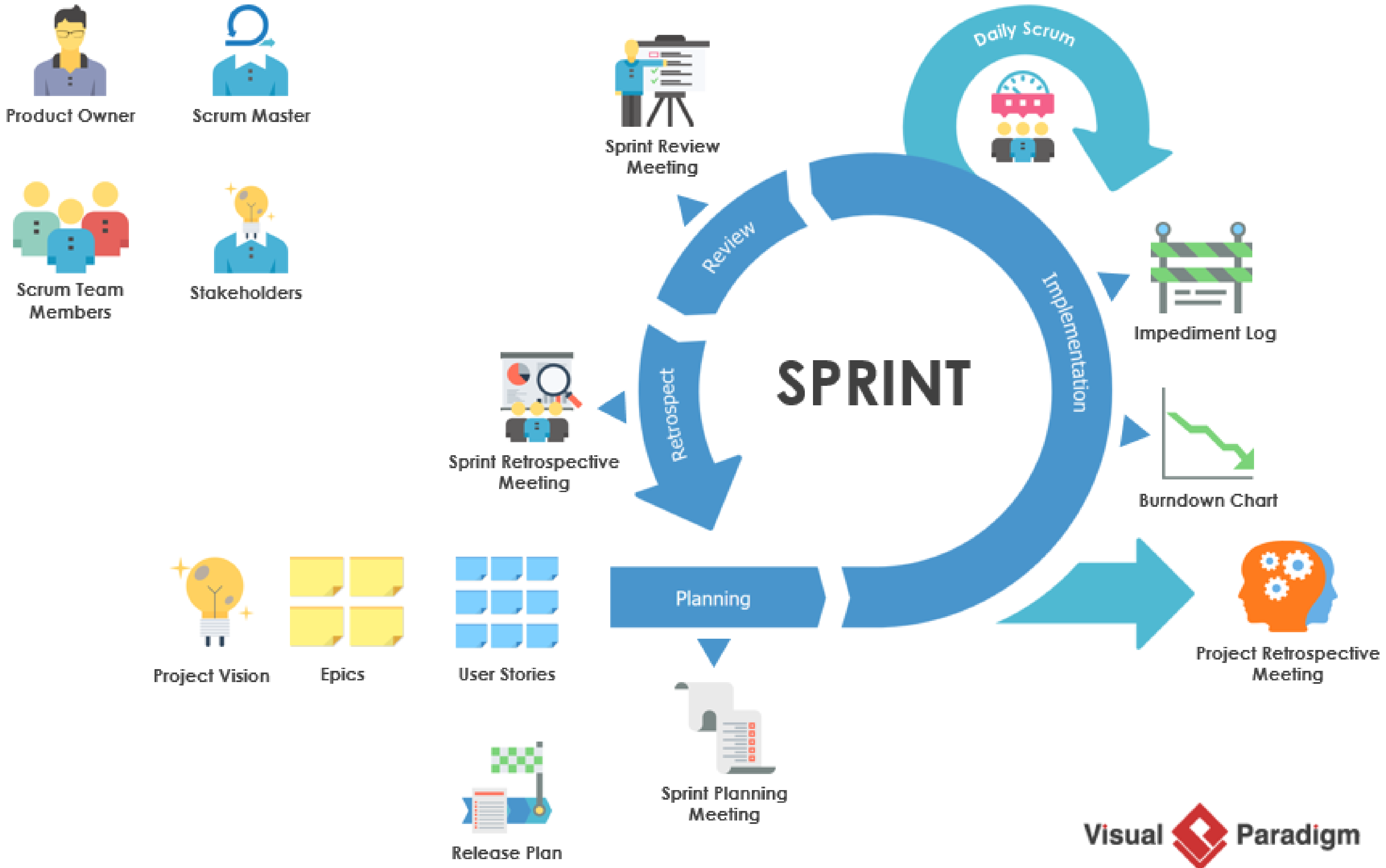


Source: https://adevait.com/blog/remote-work/adopting-agile-the-latest-reports-about-the-popular-mindset

Alternative View



The Agile – Scrum Framework



Source:

Definitions

- **Roles**
  - Product Owner
  - Scrum Owner
  - Scrum Team
- **Artifacts**
  - Product backlog
  - Sprint backlog
  - Burndown charts
- **Ceremonies**
  - Sprint planning
  - Sprint review
  - Sprint retrospective
  - Daily scrum meeting

Roles

Product Owner

- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results

Scrum Master

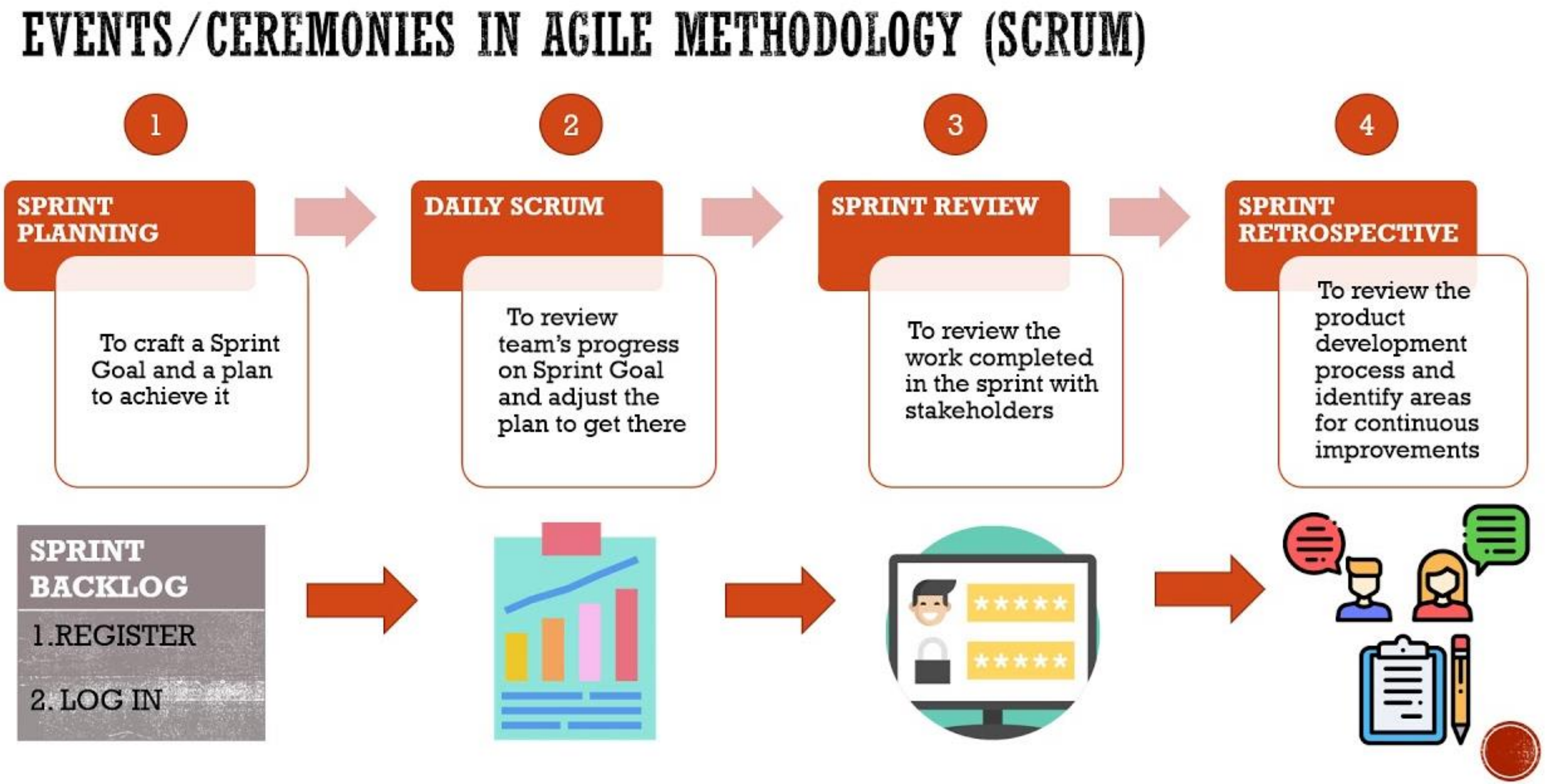
- Servant Leader of the team
- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences

SCRUM Team

- Responsible for implementing the product
- Team size is 6(±3) people.
- Self-organizing – set a realistic target for the Sprint and do the best to hit that target. They are responsible for delivering high-quality work at a sustainable pace.
- Cross-functional – Team has all the diverse skills needed to produce working product/software in a Sprint.
- Each person might have one (or more) of the following skills: architecture, coding, testing, documentation, etc.

Ceremonies

- Time Boxed
- Daily Standup – 15min -



EVENTS/CEREMONIES IN AGILE METHODOLOGY (SCRUM)

1 SPRINT PLANNING

To craft a Sprint Goal and a plan to achieve it

2 DAILY SCRUM

To review team's progress on Sprint Goal and adjust the plan to get there

3 SPRINT REVIEW

To review the work completed in the sprint with stakeholders

4 SPRINT RETROSPECTIVE

To review the product development process and identify areas for continuous improvements

SPRINT BACKLOG

1.REGISTER

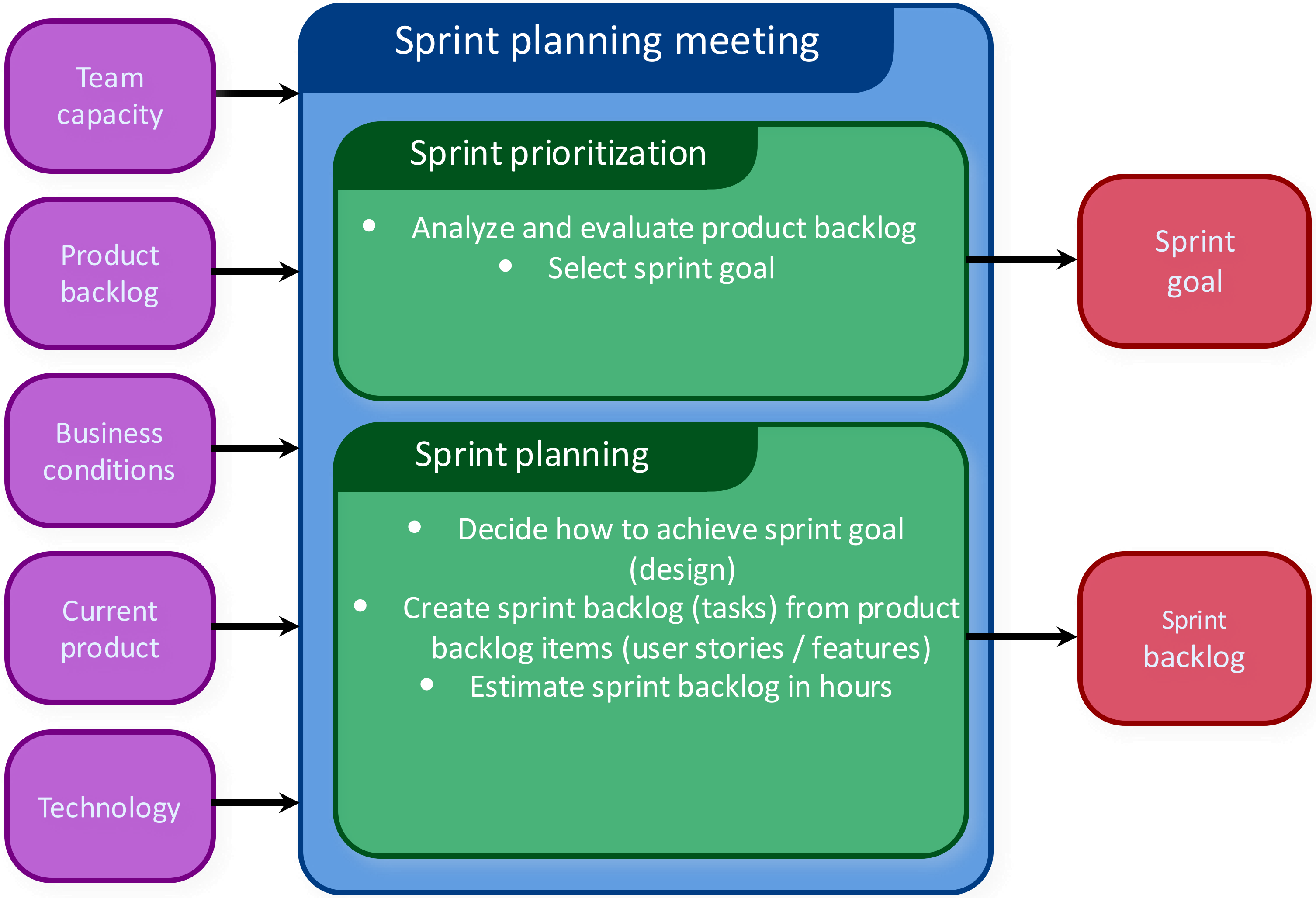
2. LOG IN

Sprint Planning

- Team selects items from the product backlog they can commit to completing
- Sprint backlog is created
- Tasks are identified and each is estimated (1-16 hours)
- Collaboratively, not done alone by the Scrum Master
- High-level design is considered

Outcome

- Creating Product Backlog
- Determining the Sprint Goal.
- Creating Sprint Backlog



Team capacity

Product backlog

Business conditions

Current product

Technology

Sprint planning meeting

Sprint prioritization

- Analyze and evaluate product backlog
- Select sprint goal

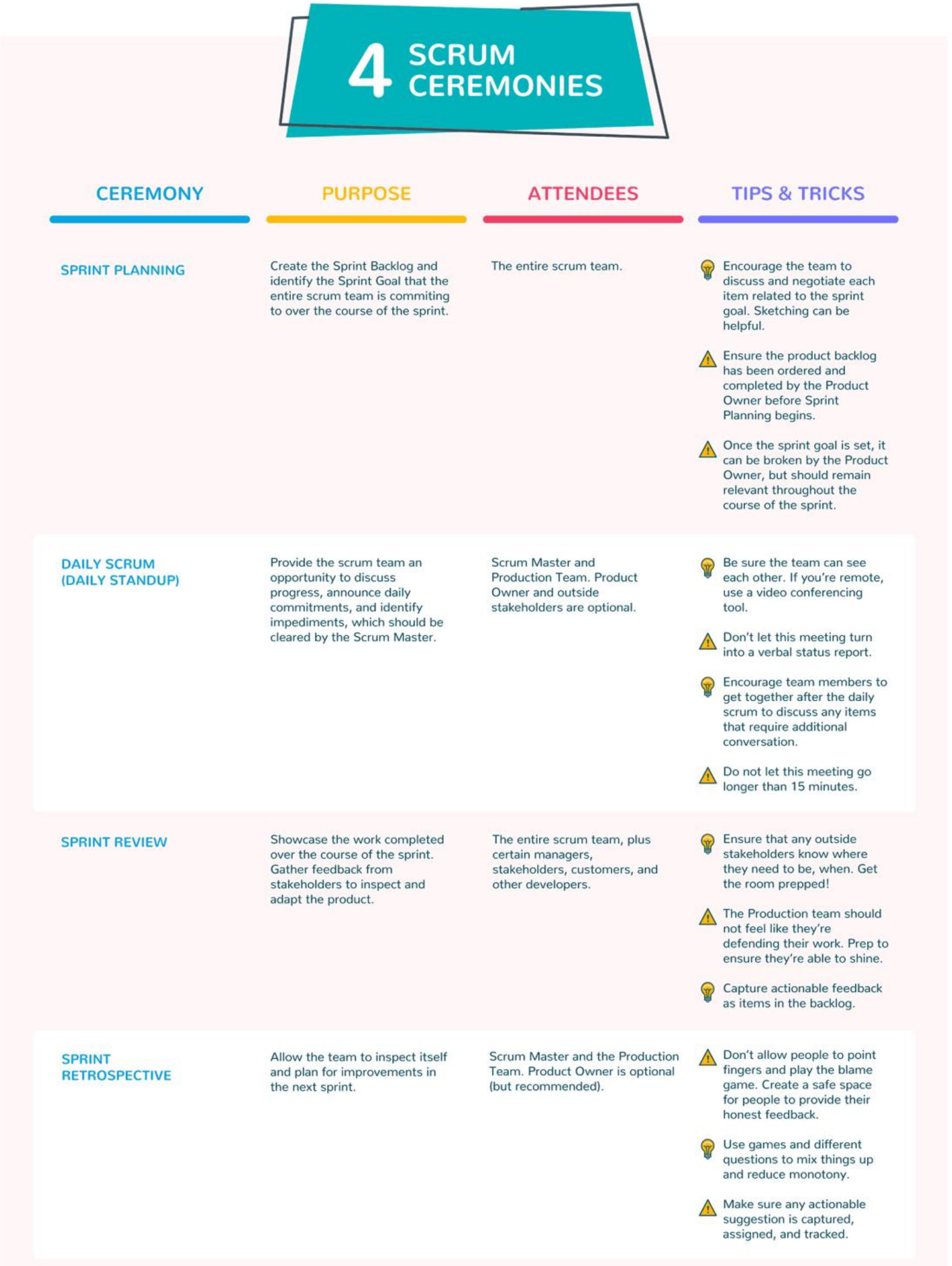
Sprint planning

- Decide how to achieve sprint goal (design)
- Create sprint backlog (tasks) from product backlog items (user stories / features)
- Estimate sprint backlog in hours

Sprint goal

Sprint backlog

Ceremonies



4 SCRUM CEREMONIES

CEREMONY	PURPOSE	ATTENDEES	TIPS & TRICKS
SPRINT PLANNING	Create the Sprint Backlog and identify the Sprint Goal that the entire scrum team is commiting to over the course of the sprint.	The entire scrum team.	💡 Encourage the team to discuss and negotiate each item related to the sprint goal. Sketching can be helpful. ⚠ Ensure the product backlog has been ordered and completed by the Product Owner before Sprint Planning begins. ⚠ Once the sprint goal is set, it can be broken by the Product Owner, but should remain relevant throughout the course of the sprint.
DAILY SCRUM (DAILY STANDUP)	Provide the scrum team an opportunity to discuss progress, announce daily commitments, and identify impediments, which should be cleared by the Scrum Master.	Scrum Master and Production Team. Product Owner and outside stakeholders are optional.	💡 Be sure the team can see each other. If you're remote, use a video conferencing tool. ⚠ Don't let this meeting turn into a verbal status report. 💡 Encourage team members to get together after the daily scrum to discuss any items that require additional conversation. ⚠ Do not let this meeting go longer than 15 minutes.
SPRINT REVIEW	Showcase the work completed over the course of the sprint. Gather feedback from stakeholders to inspect and adapt the product.	The entire scrum team, plus certain managers, stakeholders, customers, and other developers.	💡 Ensure that any outside stakeholders know where they need to be, when. Get the room prepped! ⚠ The Production team should not feel like they're defending their work. Prep to ensure they're able to shine. 💡 Capture actionable feedback as items in the backlog.
SPRINT RETROSPECTIVE	Allow the team to inspect itself and plan for improvements in the next sprint.	Scrum Master and the Production Team. Product Owner is optional (but recommended).	⚠ Don't allow people to point fingers and play the blame game. Create a safe space for people to provide their honest feedback. 💡 Use games and different questions to mix things up and reduce monotony. ⚠ Make sure any actionable suggestion is captured, assigned, and tracked.

Resource : https://thedigitalprojectmanager.com/scrum-ceremonies-made-simple/

Ceremony - Daily Stand Up Example

Purpose : Make our progress, plans, and blocks visible to each other

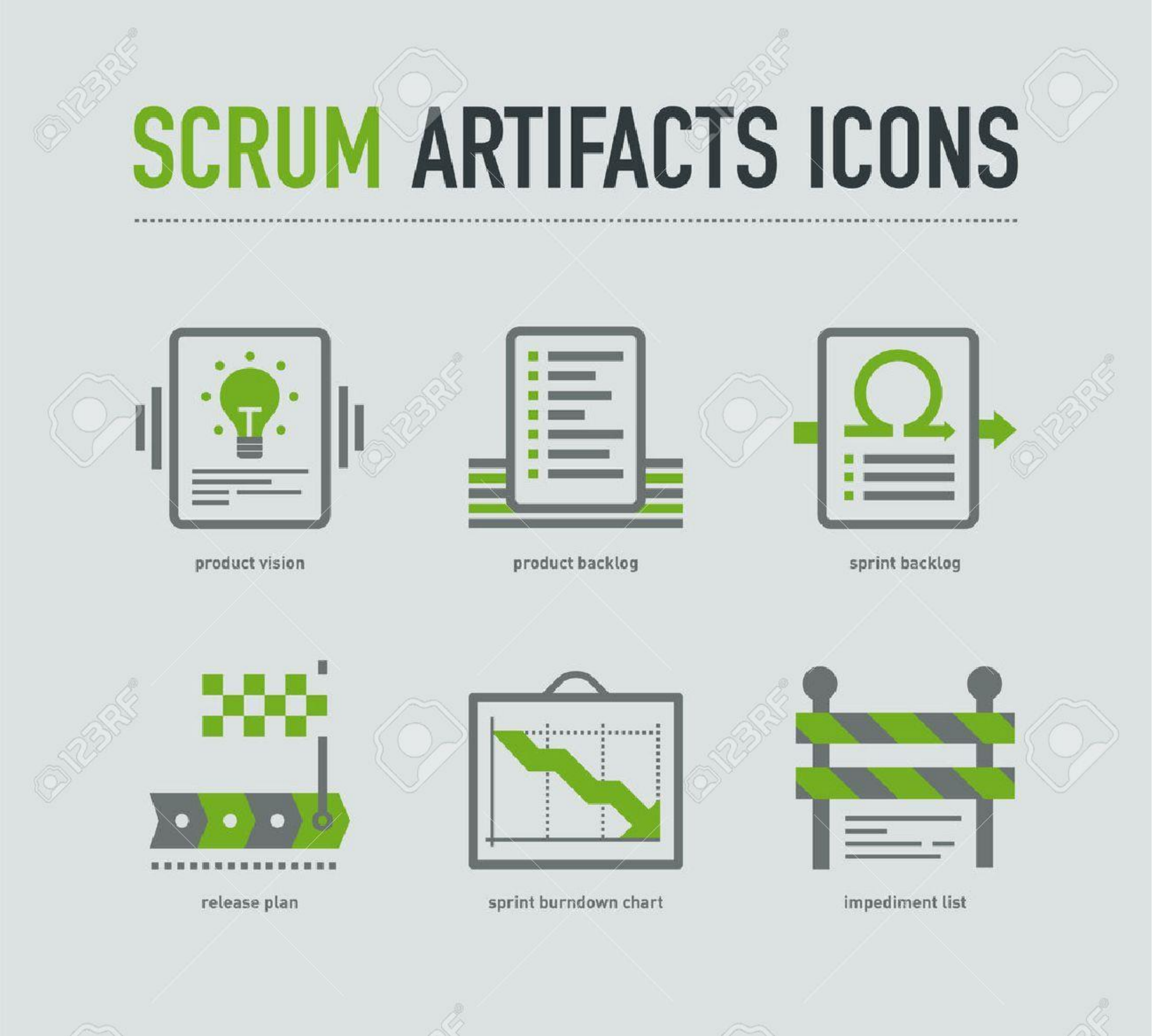
When: Daily, the Dev/Product Team **stands** in a circle and reports:

  - What have I done since the last Daily Scrum
  - What do I plan to do by the next Daily Scrum
  - What are my blocks
- 15-minute timebox (@10:00)
  - During meeting: everyone listens, no discussion
  - After meeting is done: further discussion as needed
- Product Owner can attend, but must not interfere
- Scrum Master makes note of the blocks
- After Daily Scrum, Scrum Master helps remove blocks, and people can meet in smaller groups to discuss issues

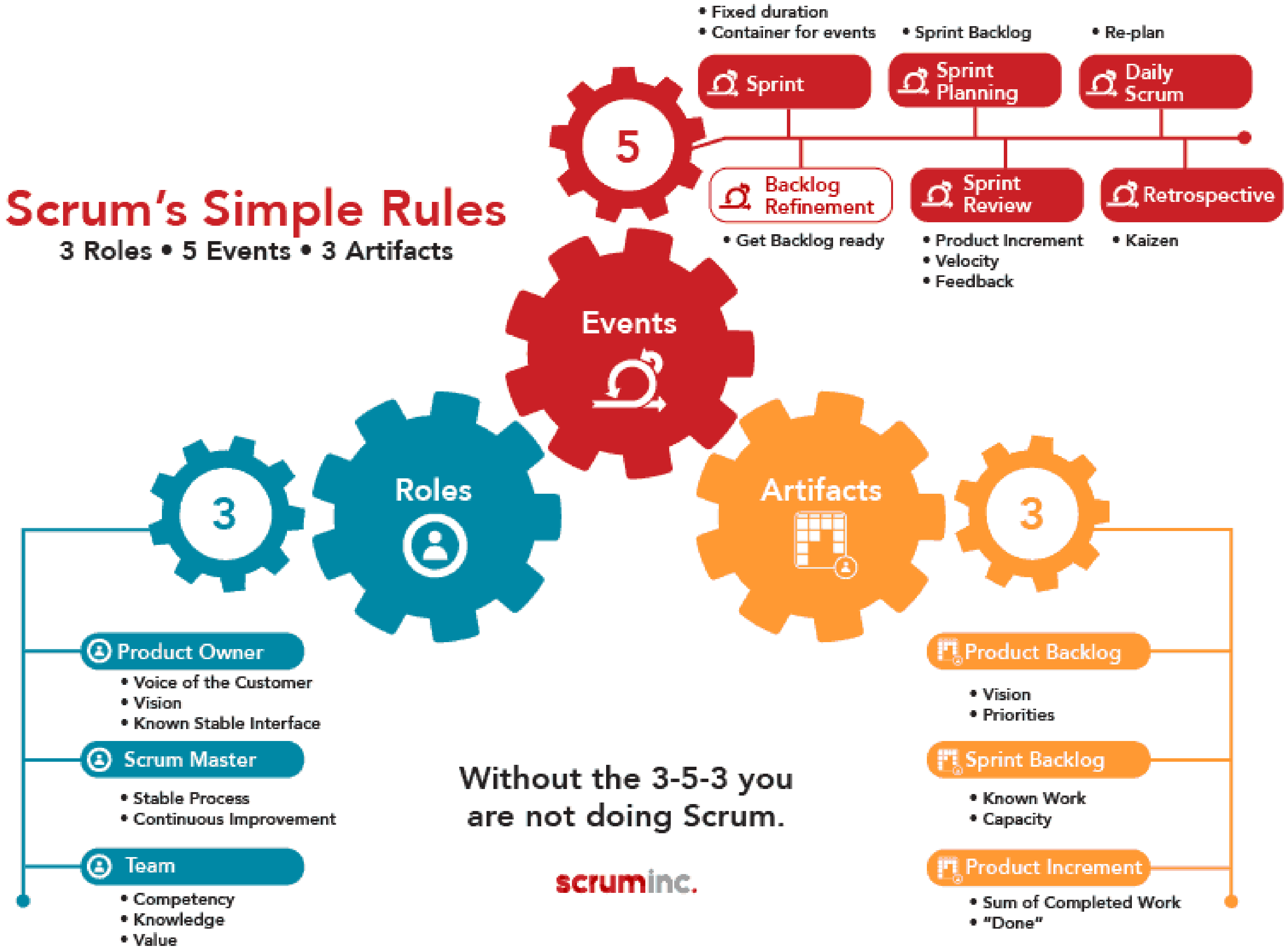
Note: These are *not* status for the ScrumMaster, They are commitments in front of peers

Artifacts

Samples



Recap and Summary



Scrum's Simple Rules

3 Roles • 5 Events • 3 Artifacts

- Sprint
  - Fixed duration
  - Container for events
- Sprint Planning
  - Sprint Backlog
- Daily Scrum
  - Re-plan
- Backlog Refinement
  - Get Backlog ready
- Sprint Review
  - Product Increment
  - Velocity
  - Feedback
- Retrospective
  - Kaizen

- Product Owner
  - Voice of the Customer
  - Vision
  - Known Stable Interface
- Scrum Master
  - Stable Process
  - Continuous Improvement
- Team
  - Competency
  - Knowledge
  - Value

- Product Backlog
  - Vision
  - Priorities
- Sprint Backlog
  - Known Work
  - Capacity
- Product Increment
  - Sum of Completed Work
  - "Done"

Without the 3-5-3 you are not doing Scrum.

scruminc.

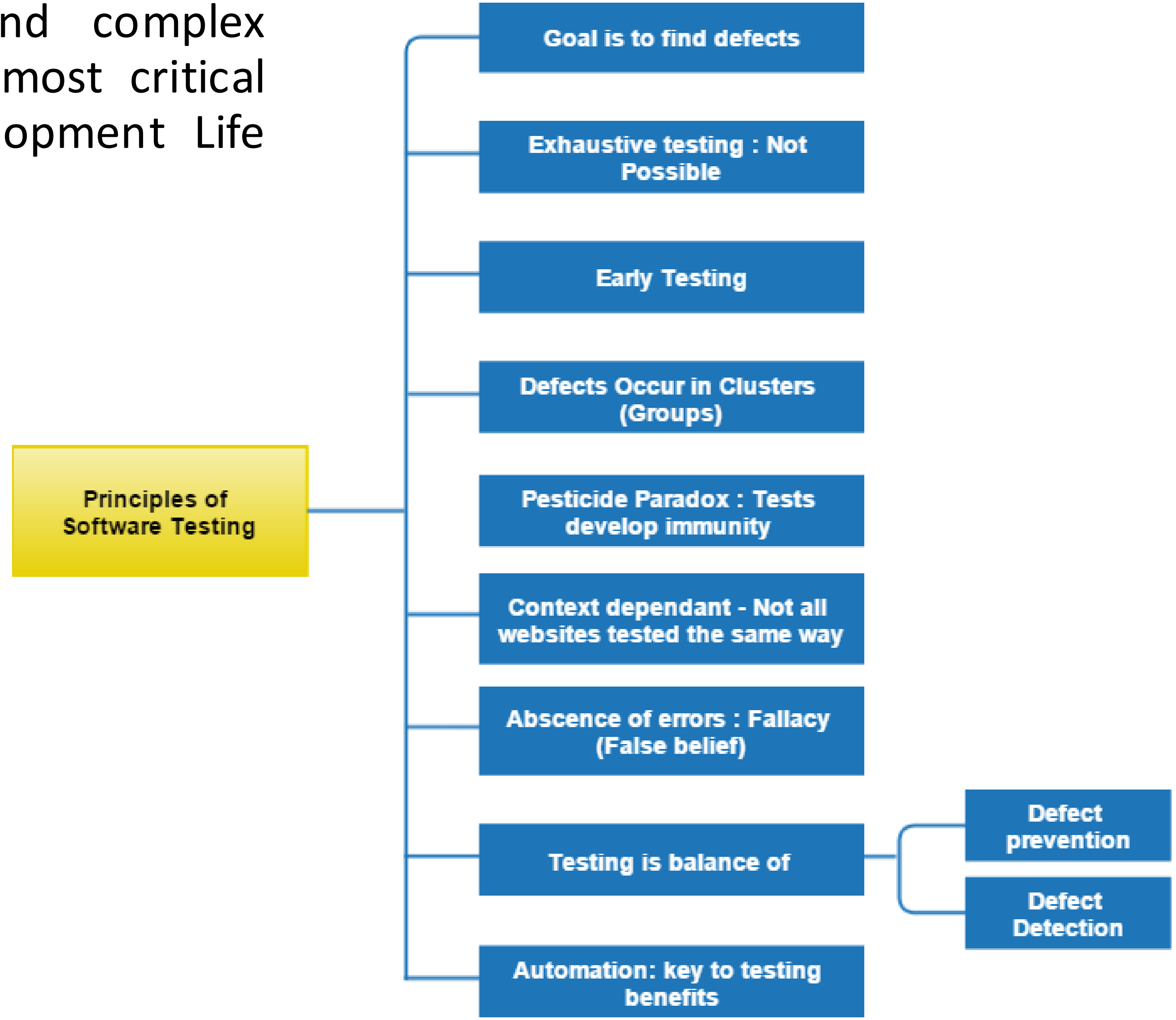
Software Testing

Activity - Search Major – Failures/Bugs

Objectives & Principles of Software Testing

- A combination of various important and complex activities, software testing is one of the most critical process performed during Software Development Life Cycle (SDLC).

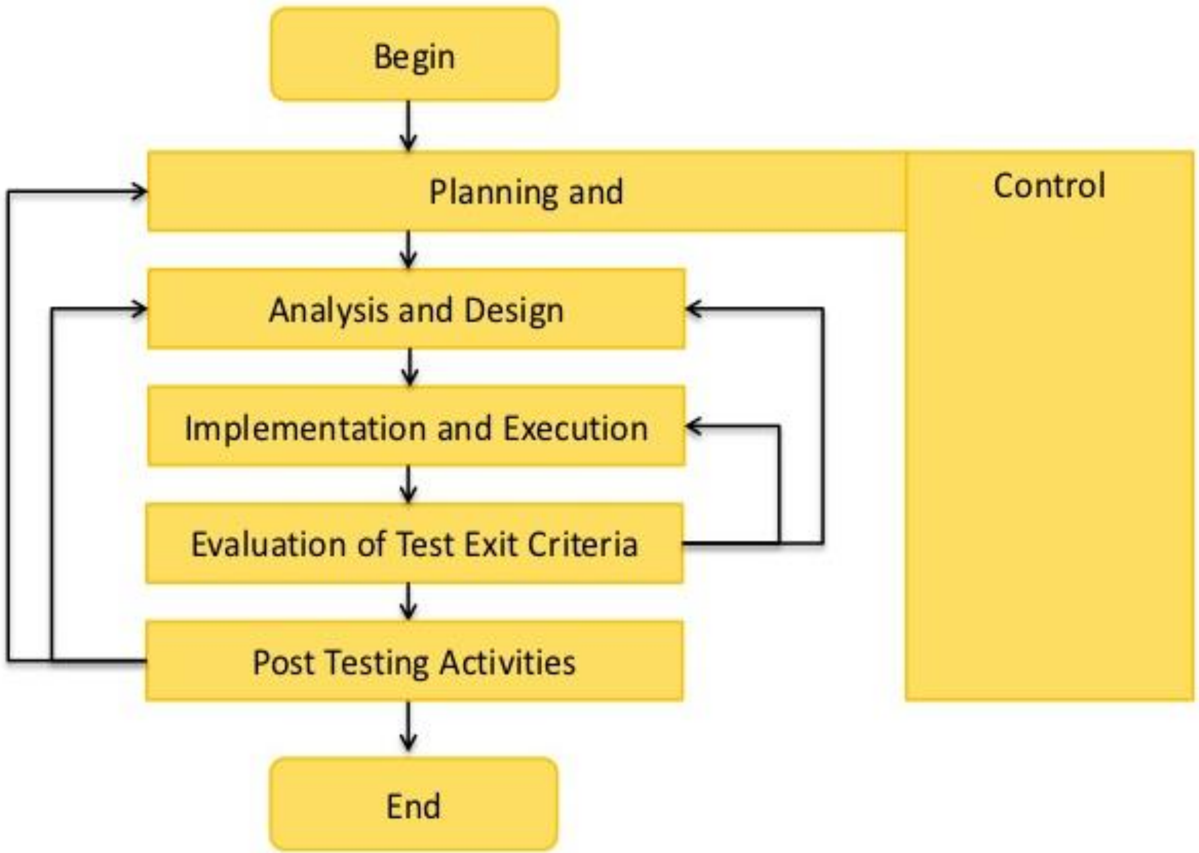
- **Testing Objectives**
  - Verification
  - Validation
  - Find Defects
  - Prevent Defects
  - Providing Information
  - Improve quality of software
  - Verifying performance & functionality
  - For optimum user experience


Principles of Software Testing
Goal is to find defects
Exhaustive testing : Not Possible
Early Testing
Defects Occur in Clusters (Groups)
Pesticide Paradox : Tests develop immunity
Context dependant - Not all websites tested the same way
Abscence of errors : Fallacy (False belief)
Testing is balance of
Defect prevention
Defect Detection
Automation: key to testing benefits

Source: https://www.testnbug.com/2014/12/principles-of-testing/

Fundamental Test Process

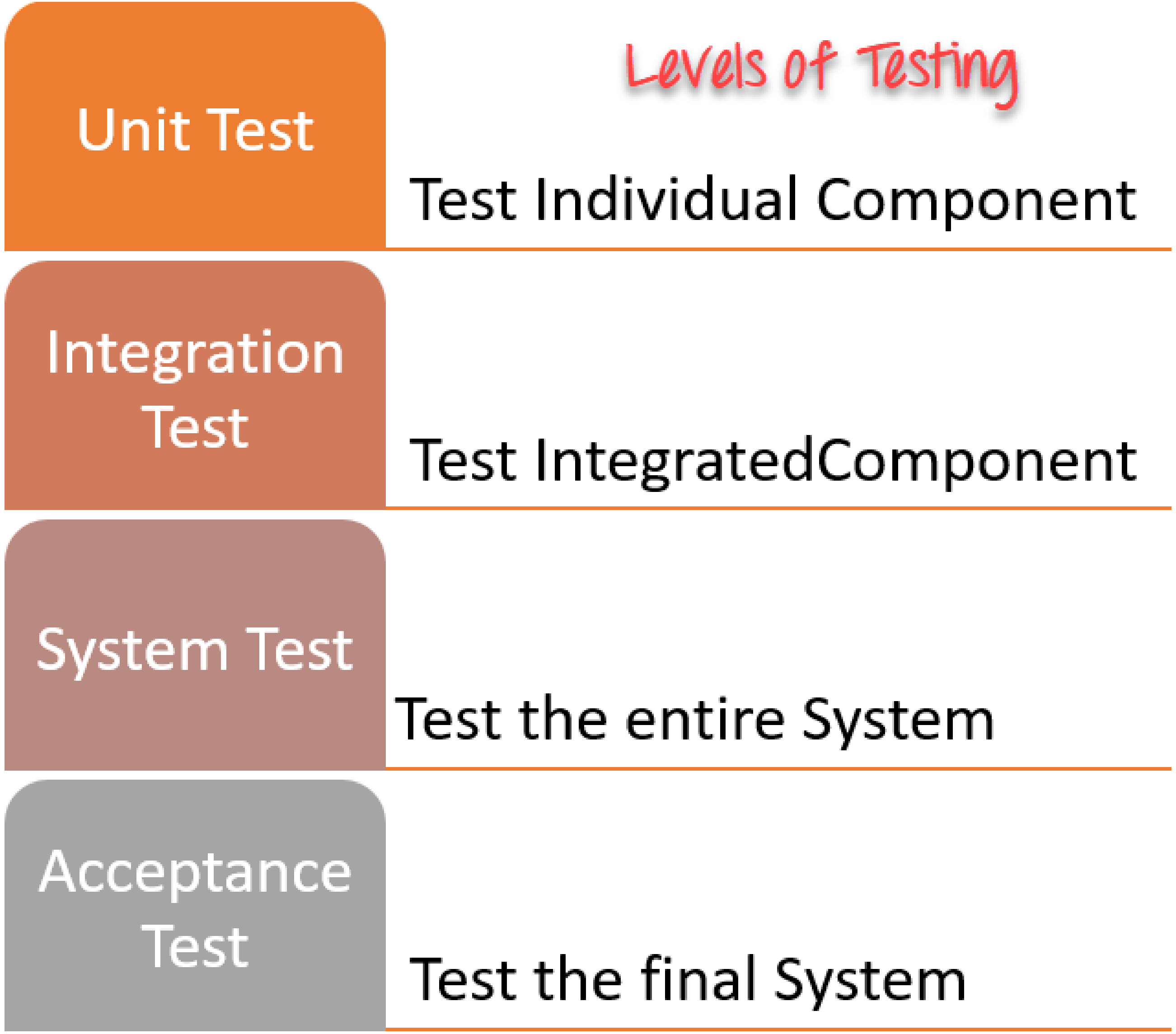
- Testing is a process rather than a single activity.
- Testing must be planned and it requires discipline to act upon it.
- The quality and effectiveness of software testing are primarily determined by the quality of the test processes used.
- The activities of testing can be divided into the following basic steps:
  - Planning and Control
  - Analysis and Design
  - Implementation and Execution
  - Evaluating exit criteria and Reporting
  - Test Closure activities



Source: https://pt.slideshare.net/nknysh/software-testing-foundations-part-1-basics-continued-

Testing Levels

- Tests are grouped together based on where they are added in SDLC or the by the level of detailing they contain.
- There are four levels of testing: unit testing, integration testing, system testing, and acceptance testing.
- The purpose of Levels of testing is to make software testing systematic and easily identify all possible test cases at a particular level.
- There are many different testing levels which help to check behavior and performance for software testing.
- These testing levels are designed to recognize missing areas and reconciliation between the development lifecycle states.



Levels of Testing

Unit Test — Test Individual Component

Integration Test — Test IntegratedComponent

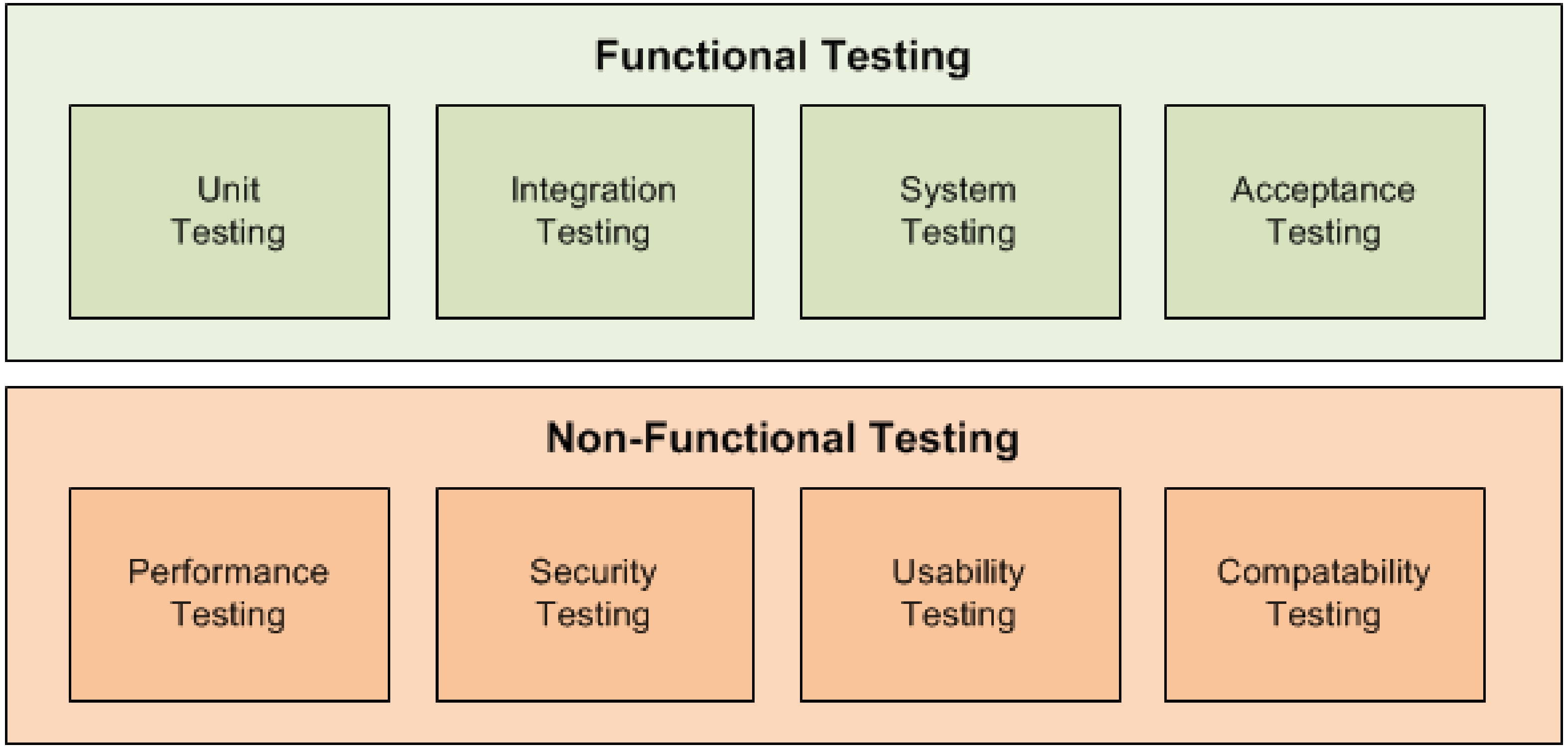
System Test — Test the entire System

Acceptance Test — Test the final System

Source: https://www.guru99.com/levels-of-testing.html

Testing Methodologies

- Testing methodologies are the strategies and approaches used to test a particular product to ensure it is fit for purpose.
- Testing methodologies usually involve testing that the product works in accordance with its specification, has no undesirable side effects when used in ways outside of its design parameters and worst case will fail-safely
- Software testing methodologies encompass everything from unit testing individual modules, integration testing an entire system to specialized forms of testing such as security and performance.



Functional Testing

Unit Testing

Integration Testing

System Testing

Acceptance Testing

Non-Functional Testing

Performance Testing

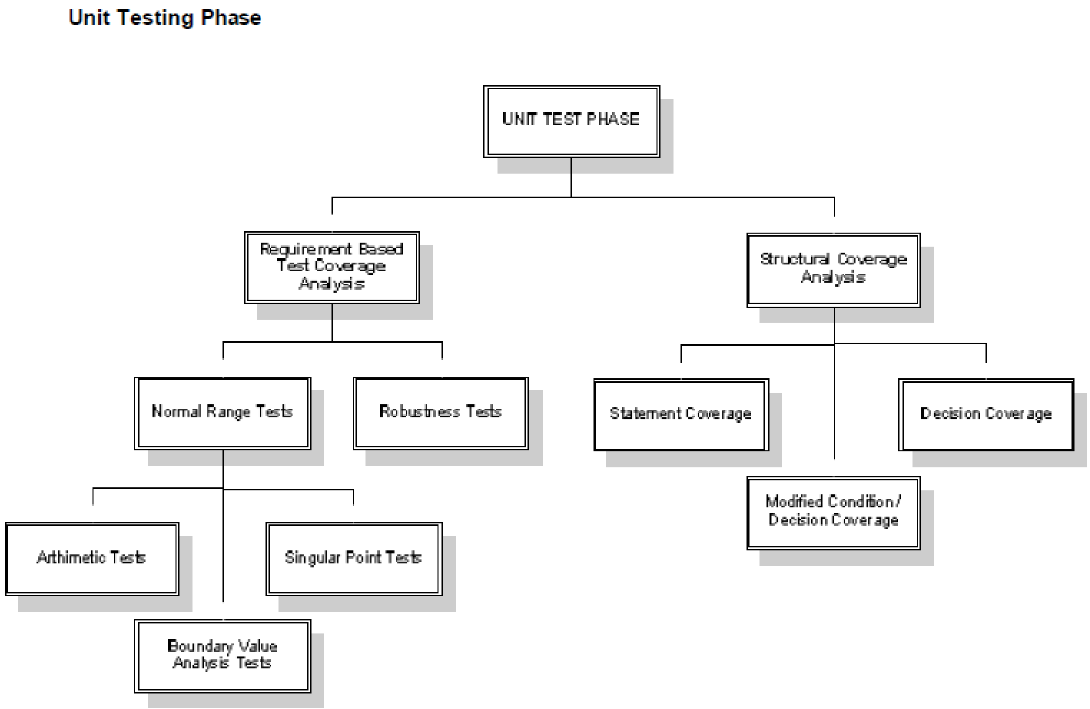
Security Testing

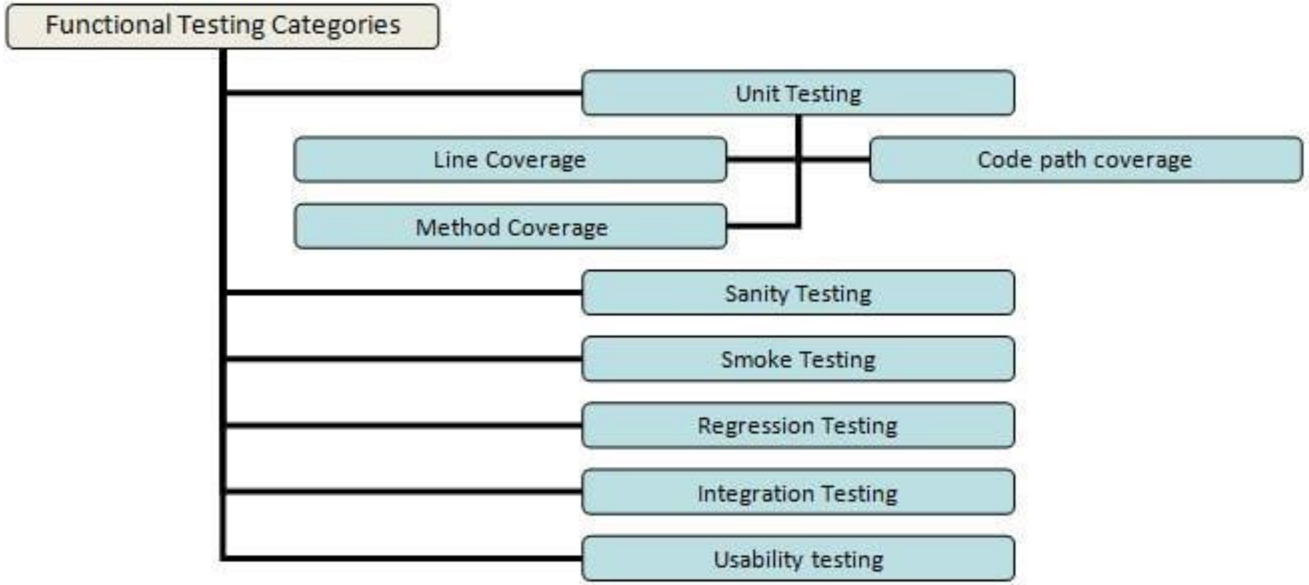
Usability Testing

Compatability Testing

Source: https://www.inflectra.com/ideas/topic/testing-methodologies.aspx

Different Representations





Source :https://www.browserstack.com/guide/functional-testing and

Unit Testing..

Robustness or Abnormal Tests

- The objective of robustness test cases is to demonstrate the ability of the software to respond to abnormal inputs and conditions.
  - Exercise real and integer inputs using equivalence class of invalid boundary values.
  - System initialization exercised with abnormal conditions.
  - Determine the possible failure modes of incoming data especially complex, digital data strings from an external system..
  - Compute out of range loop counts as appropriate.
  - Check for arithmetic overflow for time related functions.
  - Exercise transitions that are not allowed by the software requirements for state transitions.

Unit Testing..

Normal Range Tests

- The objective of normal range test cases is to demonstrate the ability of the software to respond to normal inputs and conditions.
  - Real and integer input variables should be exercised using valid equivalence classes and valid boundary values.
  - For time -related functions, such as filters , integrators and delays , multiple iterations of the code should be performed to check the characteristics of the function in context.
  - For state transitions, test cases should be developed to exercise the transitions possible during normal operation.
  - For software requirements expressed by logic equations, the normal range test cases should verify the variable usage and the Boolean operators.

Unit Testing..

- Floating-point nodes shall be tested to show that the operations are performed correctly.
- Integer or fixed-point nodes shall be stressed to a minimum and a maximum value.
- Addition and Subtraction: A test case must exist where all inputs are not 0.0
- Multiplication: A test must exist where all inputs are not 1.0
- Division: A test case must exist where the numerator and denominator are not 0.0
  - **Arithmetic Tests.**
  - The goal of arithmetic tests is to verify all computations and their precision using random input values.

    If Z = X +/- Y then DELTA of Z = (DELTA of X + DELTA of Y)

    If Z = X * Y then DELTA of Z = (X * DELTA of Y + Y * DELTA of X)

    If Z = X / Y then DELTA of Z = (Y * DELTA of X + X * DELTA of Y) / Square (Y)
  - **Singular Point Tests.**
  - The purpose of this test is to implement conditions such as comparisons that shall be verified by making minor variations (ca lled delta) to the operands involved. Cases can cover :
    - Case where the condition is verified
    - Case where the condition is not verified
    - Floating-point except for equal to (=) and not equal to (!=) : Within 10% above and within 10% below (see note)
    - Floating-point equal to (=) and not equal (!=) to : Within 10% above, equal to, within 10% below (see note)
    - Signed and Unsigned Integer : 1 count above, equal to , 1 count below
    - Discrete Word : Equal to and not equal to
    - Boolean : Equal to and not equal to

    Note: For the comparison “X < Y”, there must be one test case where Y < X < (1.1 * Y) and another test case where (0.9 * Y) < X < Y, where 1.1 and 0.9 are DELTA. X and Y may be reversed. If the value is 0.0, use +1.0 and -1.0 instead of 10% above and below.

Unit Testing..

Boundary Value Tests

- This is done to verify whether the test inputs are tested at the minimum, nominal and median

((minimum + maximum)/2) values

- A test for the maximum value of the type of that variable ,
- A test for the minimum value of the type of that variable and
- A test for the median value of the type of that variable . A test for the maximum value of the type of that variable

Floating-point : not required

Integer : -32768 to 32767 (16-bit) or -2147483648 to 2147483647 (32-bit)

Unsigned Integer : 0 to 65535 (16-bit) or 0 to 4294967295 (32-bit)

Discrete Word : 0000H to FFFFH (16-bit) or 00000000H to FFFFFFFFH (32-bit)

Boolean : FALSE to TRUE

Modified Condition/Decision

- The Modified Condition/Decision Coverage enhances the condition/decision coverage criteria by requiring that each condition be shown to independently affect the outcome of the decision. This kind of testing is performed on mission critical application which might lead to death, injury or monetary loss.
- Designing Modified Condition Coverage or Decision Coverage requires more thoughtful selection of test cases which is carried out on a standalone module or integrated components.
- Characteristics of Modified Condition/Decision Coverage:
  - Every entry and exit point in the program has been invoked at least once.
  - Every decision has been tested for all the possible outcomes of the branch.
  - Every condition in a decision in the program has taken all possible outcomes at least once.
  - Every condition in a decision has been shown to independently affect that decision's outcome.

Integration Testing

Upon completion of unit testing, the units or modules are to be integrated which gives raise to integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

Correct with software requirements,

Correct data flow,

Correct control flow,

Correct timing,

Correct memory usage.

Typical errors revealed by Software Integration Testing:

- Incorrect initialization of variables and constants.
- Parameter passing errors.
- Data corruption, especially global data.
- Inadequate end-to-end numerical resolution.
- Incorrect sequencing of events and operations

Integration Strategies:

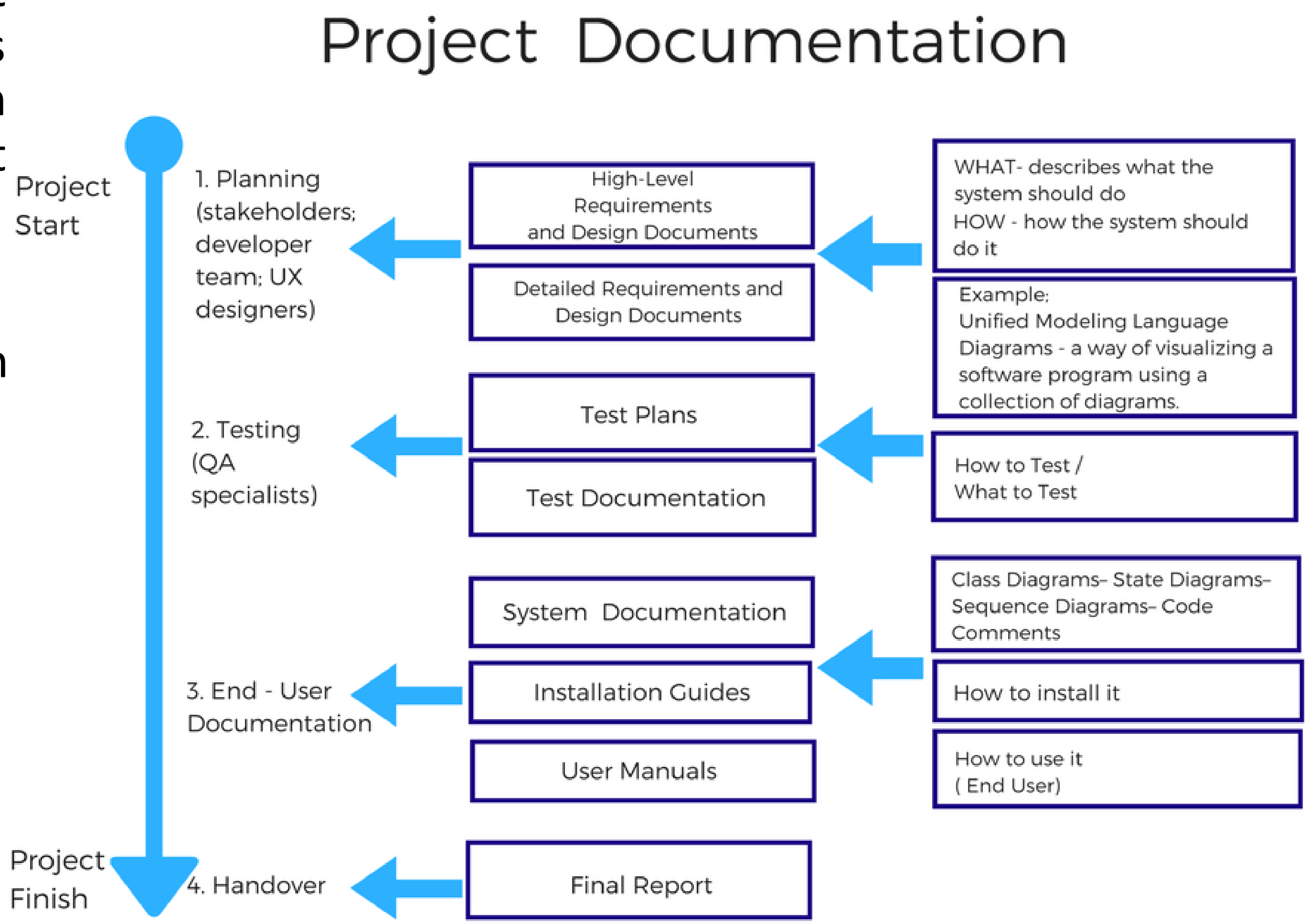
- Big-Bang Integration
- Top Down Integration
- Bottom Up Integration
- Hybrid Integration

Software Documentation

Software Documentation

- Any written text, illustrations or video that describe a software or program to its users is called program or software document. User can be anyone from a programmer, system analyst and administrator to end user.

- Software documentation is a critical process in the overall software development process.



Project Documentation

Project Start

1. Planning (stakeholders; developer team; UX designers)

High-Level Requirements and Design Documents

Detailed Requirements and Design Documents

WHAT- describes what the system should do
HOW - how the system should do it

Example:
Unified Modeling Language Diagrams - a way of visualizing a software program using a collection of diagrams.

2. Testing (QA specialists)

Test Plans

Test Documentation

How to Test /
What to Test

System Documentation

Class Diagrams- State Diagrams- Sequence Diagrams- Code Comments

3. End - User Documentation

Installation Guides

How to install it

User Manuals

How to use it
(End User)

Project Finish

4. Handover

Final Report

Source:https://blog.prototypr.io/software-documentation-types-and-best-practices-

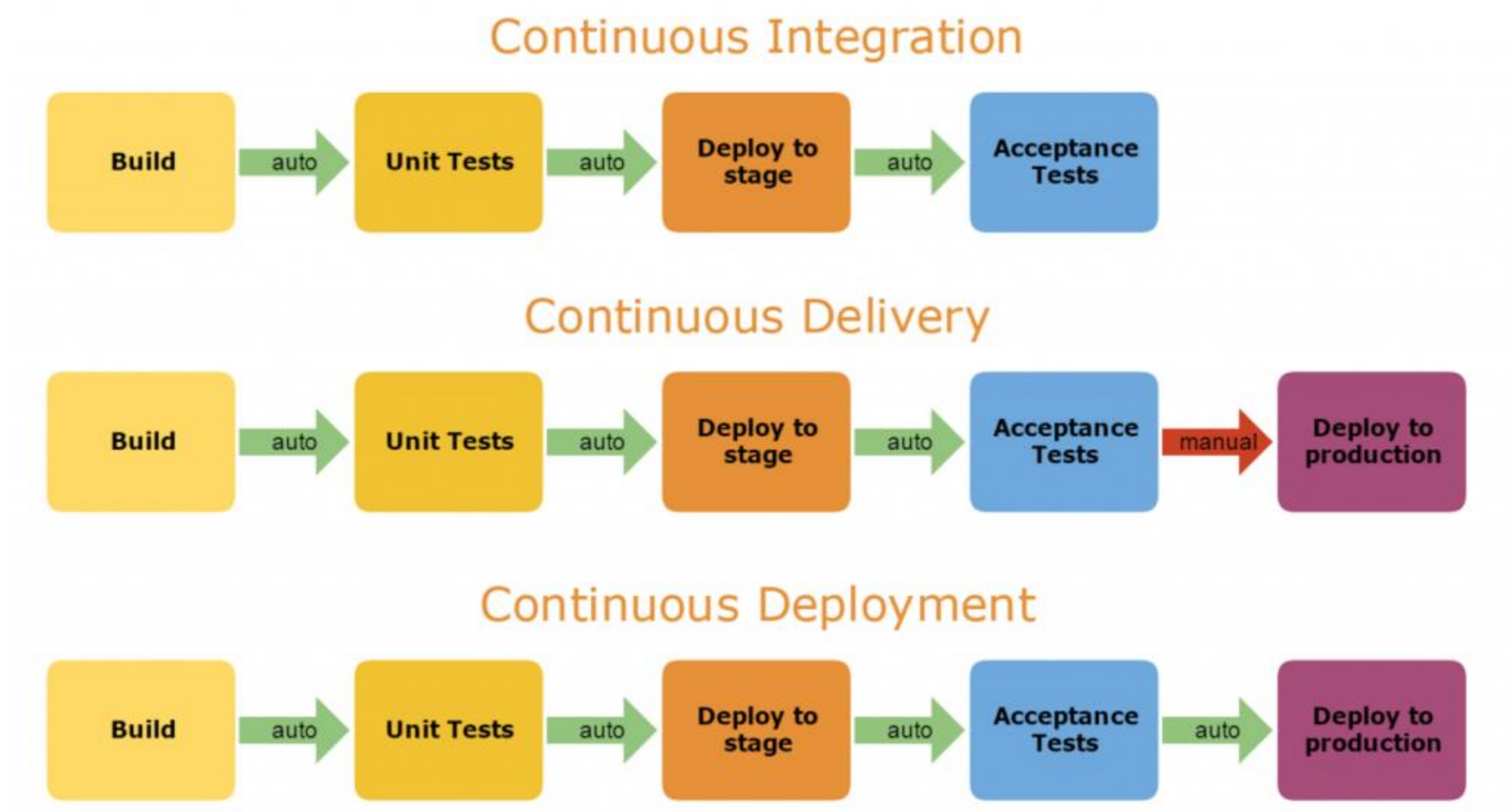
Software Documentation – Templates

- Project Plan
  - Gantt Chart
  - Work Breakdown Structure (WBS)
- Requirement Specifications Document (FRS/SRS)
- Design Document
  - System Level (High Level)
  - Integration & Unit Level (Low Level)
- Test Plan
- Test Documentation
  - Requirement Traceability Matrix – RTM
  - Test Design Document
  - Test Report
  - Defect Report
- Release Document/Note

Source:

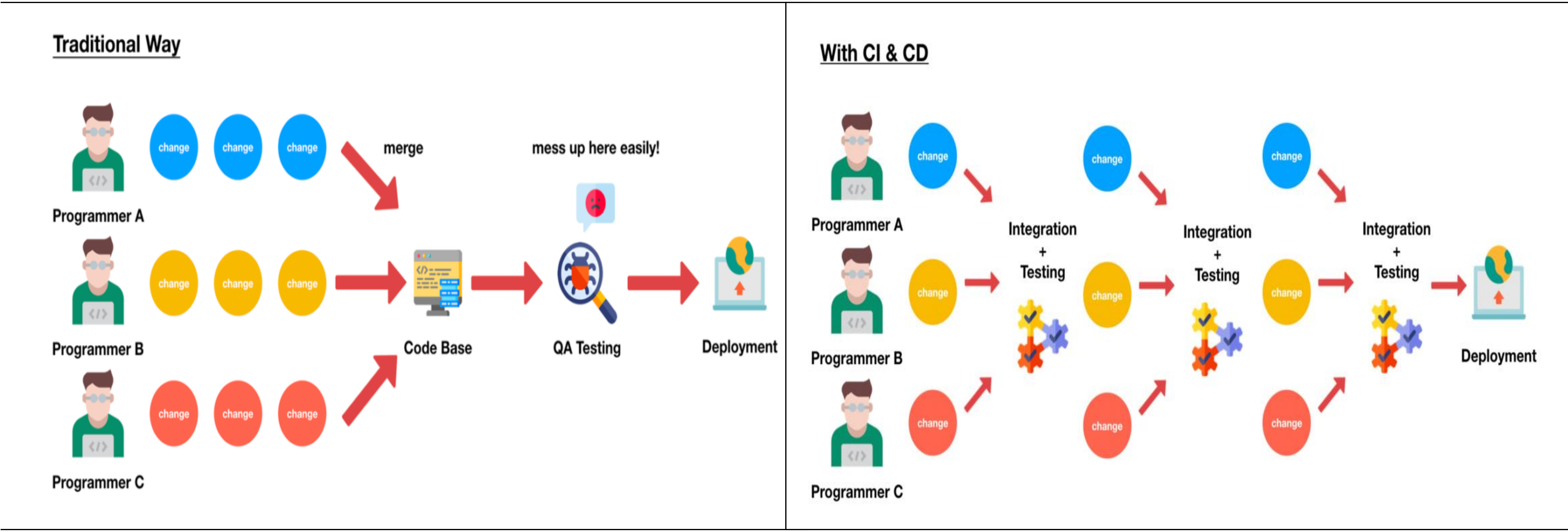
Continuous Integration/Continuous Delivery (CI/CD)

Understanding Continuous Integration, Delivery & Deployment



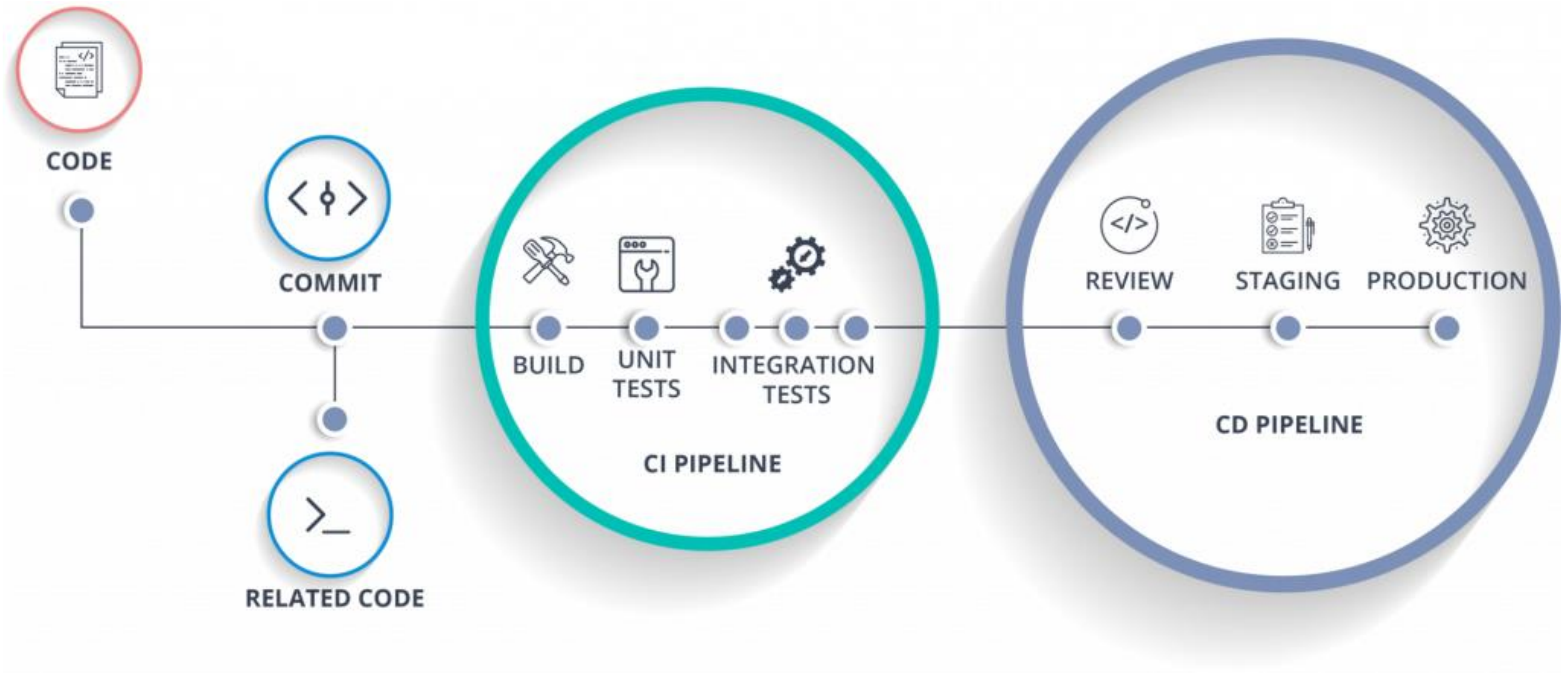
Source: https://www.linuxnix.com/what-is-continuous-integration-delivery-deployment-and-ci-cd-pipeline/

Traditional vs CI/CD



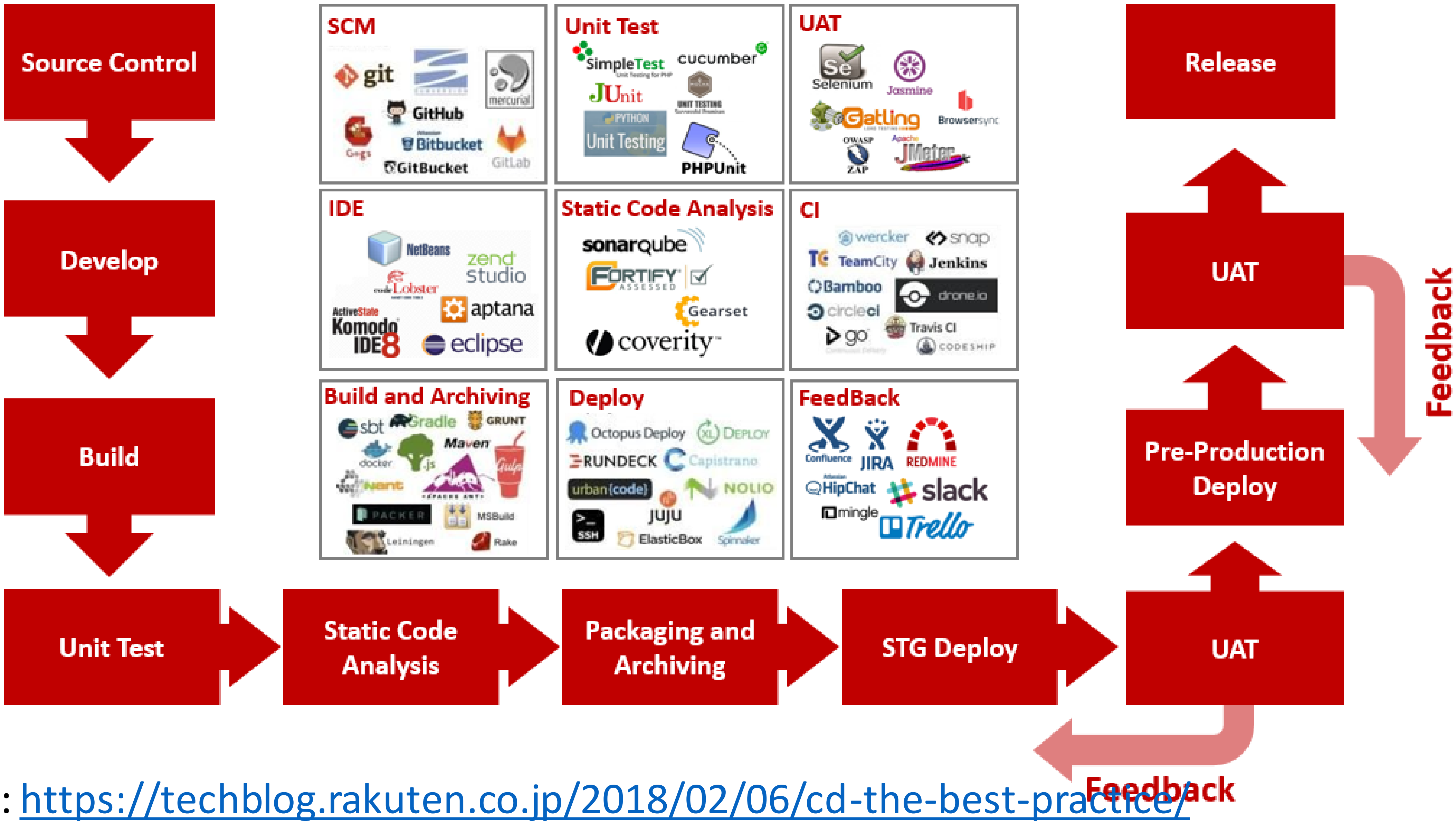
Source: https://blog.oursky.com/2019/08/19/how-to-build-cicd-pipeline/

Understanding CI/CD Pipeline



Source: https://www.linuxnix.com/what-is-continuous-integration-delivery-deployment-and-ci-cd-pipeline/

CI/CD Tools



Source Control

Develop

Build

Unit Test

Static Code Analysis

Packaging and Archiving

STG Deploy

UAT

Pre-Production Deploy

UAT

Release

Feedback

Feedback

SCM

Unit Test

UAT

IDE

Static Code Analysis

CI

Build and Archiving

Deploy

FeedBack

Source: https://techblog.rakuten.co.jp/2018/02/06/cd-the-best-practice/

Code Quality

Code Quality

- **Clean code makes you work fast**
- **Measure the size of functions**
- **Name your functions well**
- **Build a Code Quality Assurance System for your team**
  - Version control tool to ensure code quality and transparency (e.g Git)
  - Style guide for readable and comprehensible code
    - Use linters to automatically test code style (C C++ PC-lint, Python pylint, JavaScript ESLint)
  - Improve code quality with functional tests (unit, integration, system)
  - Track Test Coverage (e.g LDRA)
    1. Statement coverage (%): number of statements executed during a test divided by all statements
    2. Branch coverage (%): number of executed conditions divided by all conditions
    3. Function coverage (%): number of executed functions divided by all functions
    4. Lines coverage (%): number of lines ran during a test divided by all lines
  - Use Continuous Integration Tools (e.g Jenkis, TravisCI)

Source: https://codingsans.com/blog/code-quality#code_quality

Test Frameworks

Test Frameworks - Sample

Robot

PyTest

pytest

unittest

unittest

DocTest

DOCtest

Python Test Frameworks

JUNIT

Mocha

Se

Selenium

TestNG

Java Test Frameworks

Embunit

LDRA

VisualAssert

DOCtest

C++ testing framework

Check

C / C++ Test Frameworks

MSTest

nunit

xUnit.net

C# Test Frameworks

Activities & Hands On

Project Templates

Activity

- Requirements
  - Built your Requirements
    - Identify Feature
    - State of Art
      - Ageing - Time
      - Ageing – Cost / Feature evolution
    - Identify your requirements
    - 4 W and 1 H
    - SWOT analysis
  - Derive High Level Requirements
  - Derive Low Level Requirements
- Design
  - UML – Structural and Behavioral
    - Min 2 each (min 4 diagrams)
- Test Plan
  - High Level /Integrated Test Plan
  - Unit Level Test Plan
- Implement
  - Code
  - Testing
  - Code Quality Check
  - Issue Tracking
  - Peer Review

Questions and Answers

Sources and Acknowledgement

This content is developed from sources on internet and internal resources

Respective external sources are referenced in the slides

Thank You !

L&T Technology Services