SE345

Atılım University
Dept of Software Engineering

Asst. Prof. Dr. Aylin AKCA-OKAN

Tentative Course Schedule

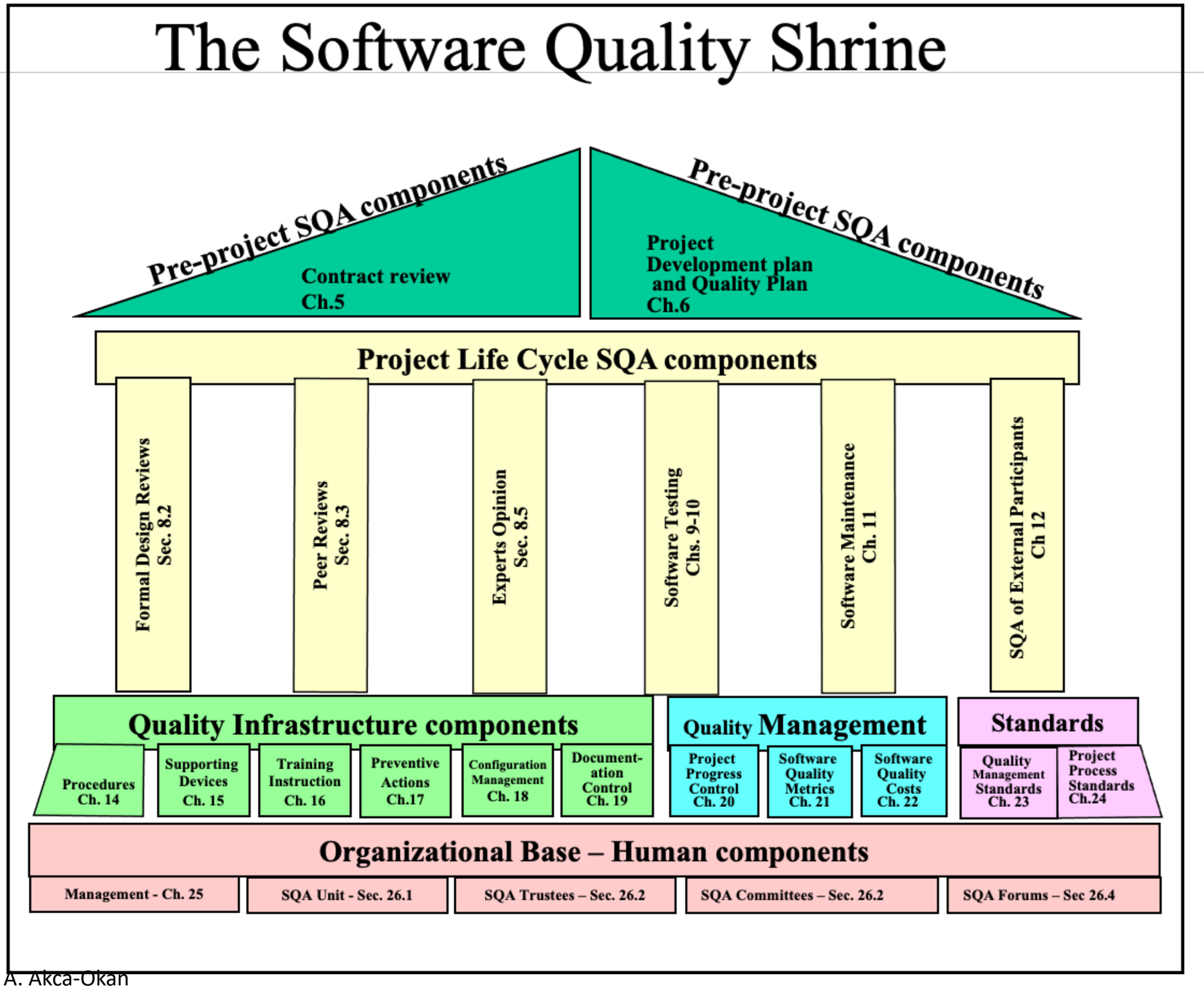
Wk	Subjects	Chapter
1	Introduction to Software Quality and Assurance	Chapter 1
2	Introduction to Software Quality and Assurance	Chapter 1
3	Software Quality Factors	Chapter 3
4	Overview of Components of the SQA System	Chapter 4
5	Overview of Components of the SQA System	Chapter 4
6	Integrating Quality Activities in Project Life Cycle	Chapter 7
7	Software Quality Metrics	Chapter 21
8	Midterm	
9	Presentations	
10	Presentations	
11	Reviews, Inspection and Audits	Chapter 8
12	Procedures and Work Instructions	Chapter 14
13	Software Change Process	Chapter 18
14	SOA Process Standards	Chapter 23

Why we’re discussing models?

- The models represent the framework of a disciplined approach to development.
- SQA must take place in conjunction with the completing of activities in the models or looking at the work products produced from these activities.
- Need to understand models before we can produce plans that are integrated into these models.

Let’s remember the famous Shrine!

The Software Quality Shrine



In summary ...

The SQA function professionals need to be familiar with the various software engineering models to be able to fulfil tasks such as:

- preparing a quality plan that is properly integrated into the project plan,
- providing development teams with professional support to perform quality assurance activities, and
- following up on the performance of these activities.

Software Development Models

Software Development Life Cycle is a structured procedure for building high-quality software.

- It breaks development into clear phases like planning, building, testing, and maintaining.
- This framework helps developers and stakeholders stay on track and have a clear path to follow, so the development process is more efficient and smoother.

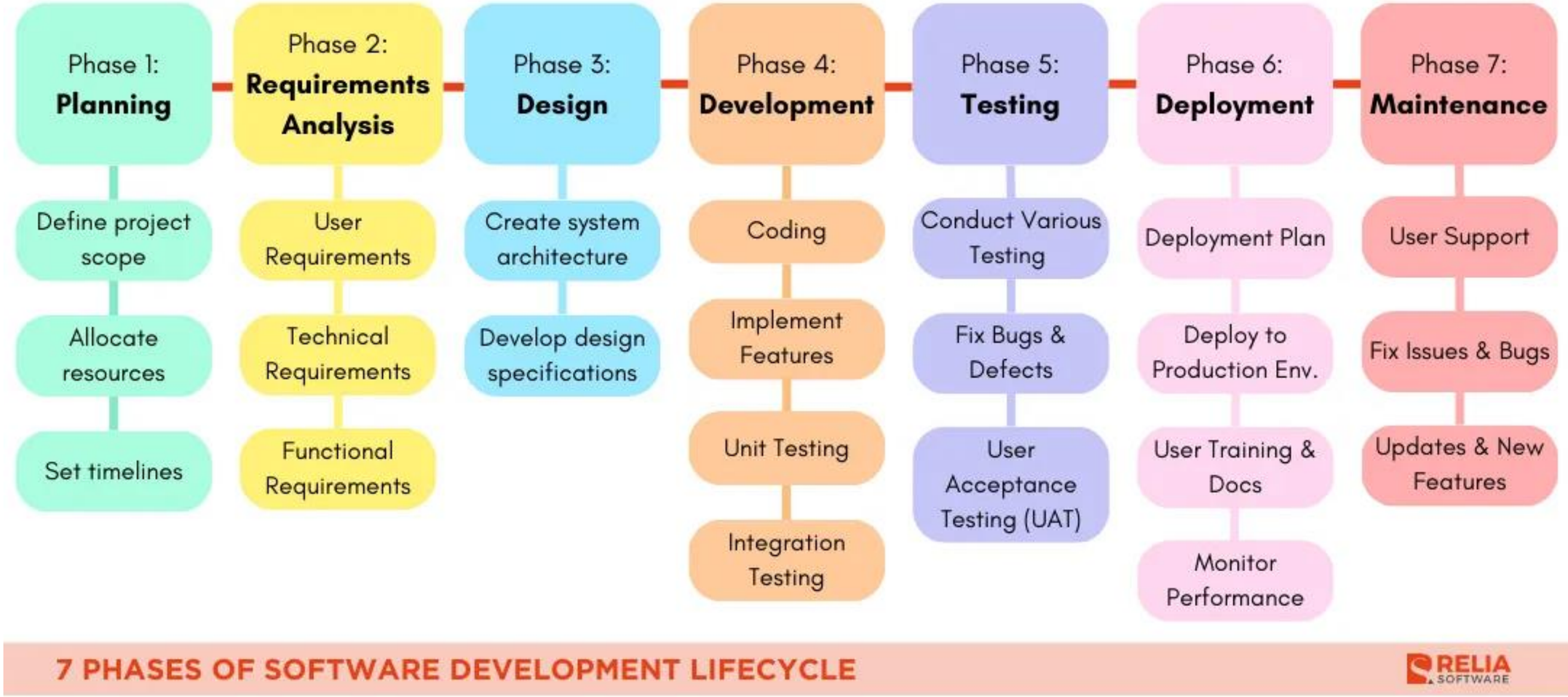
Traditional

- Waterfall
- Incremental
- Iterative
- Prototyping
- Spiral
- ...

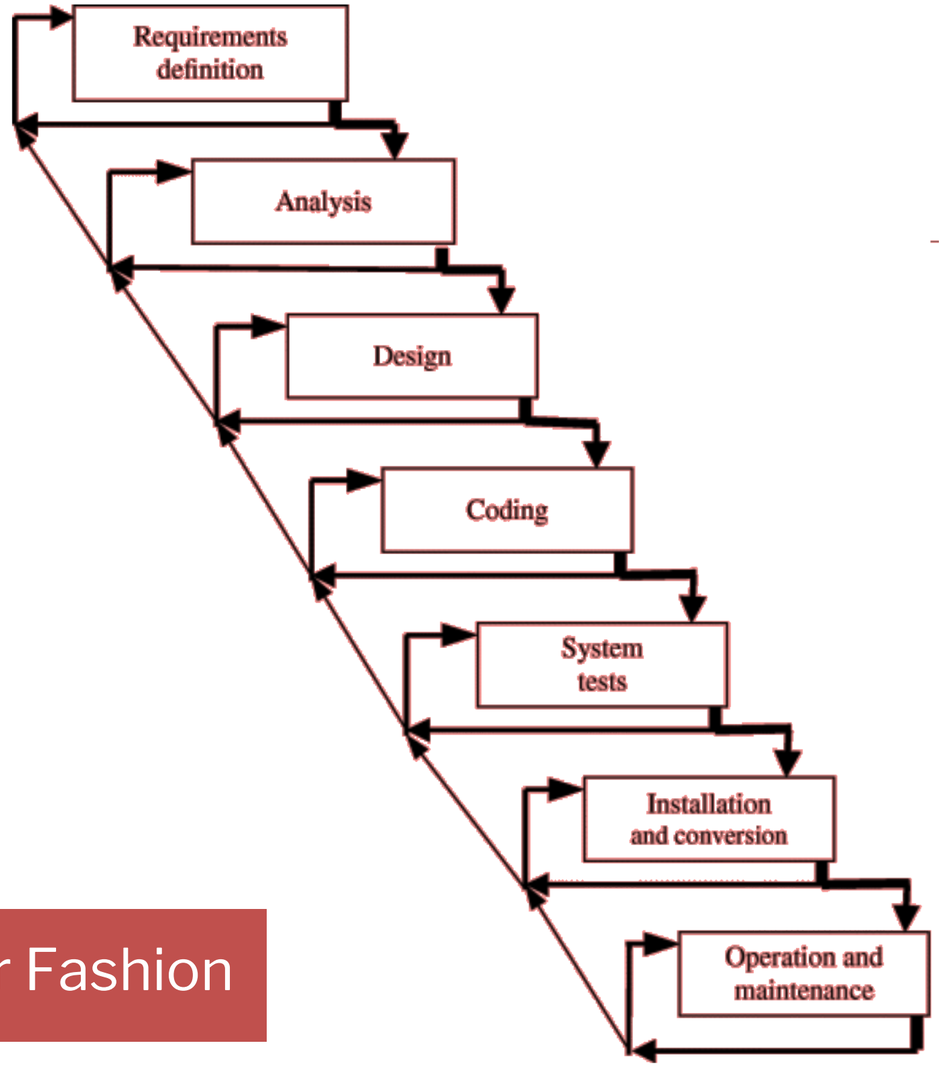
Agile

- Extreme Programming - XP
- Scrum
- Feature driven
- Kanban
- ...

Software Development Models



Waterfall Model



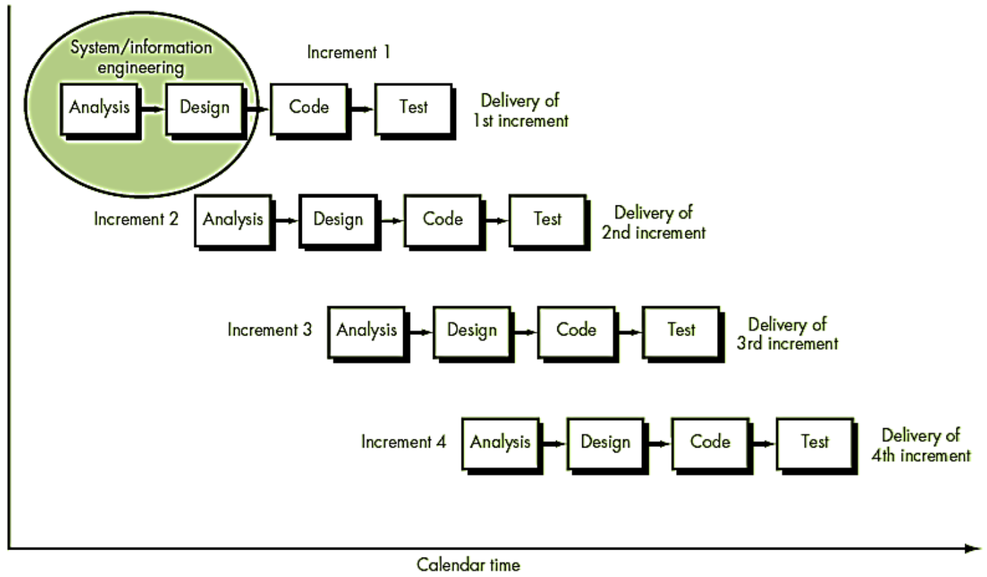
Linear Fashion

- Classical Model known as one-shot or once-through model.
- Flow is downwards with the possibility to go backward. (exception rather than a rule)
- If there is uncertainty about how a system is to be implemented, use a more flexible, iterative approach.

Is suitable for projects with:

- ➔ Clear, unambiguous, and stable user
- ➔ requirements
- ➔ Familiar, proven technology
- ➔ Low complexity
- ➔ Adequate time
- ➔ Stable schedule

Incremental Development

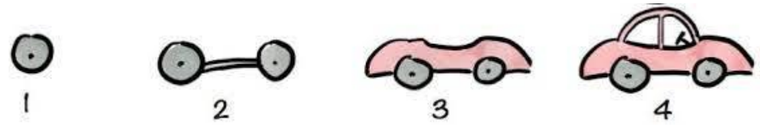


- Need to provide a limited set of functionalities to users quickly
- Then refine and expand on that functionality in later software releases

Deliverable increments

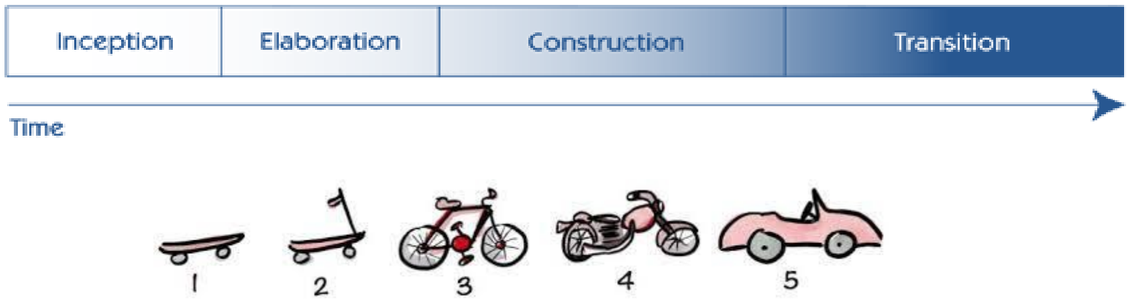
- 1st increment = core product
- nth increment = core product + more features
- Up to a complete product

Linear & Parallel Fashion



The Iterative Approach

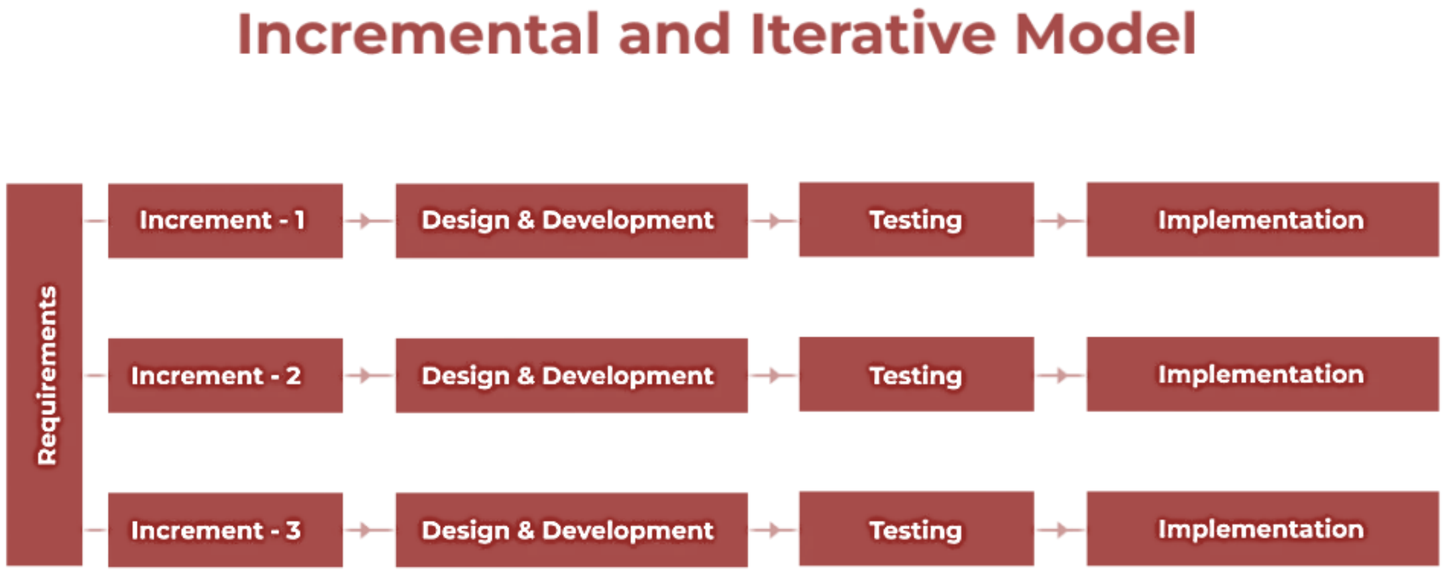
In the iterative approach, the lifecycle phases are decoupled from the logical software activities that occur in each phase, allowing us to revisit various activities, such as requirements, design, and implementation, during various iterations of the project.



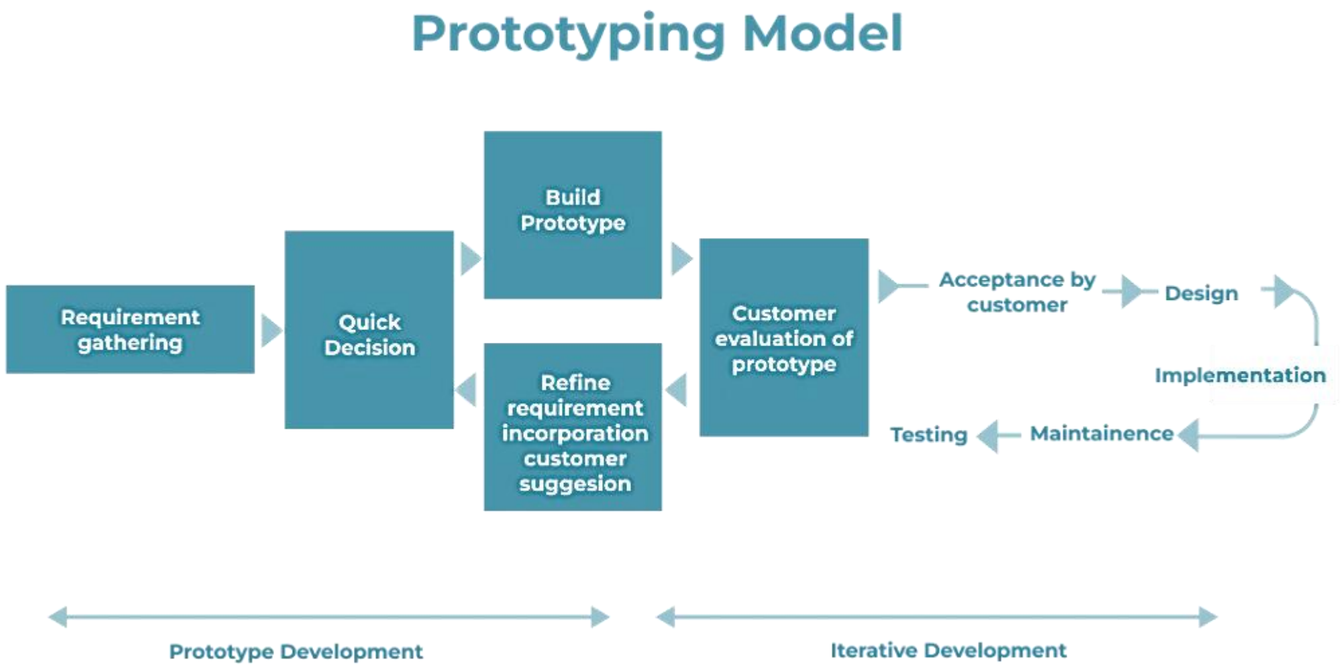
Incremental and Iterative Model

The Incremental and Iterative software development model is mainly used for custom software development. The work in this process is done in iterations, which act as stages. There is a defined framework for the project, which is divided into iterations.

- In each iteration, you can add new modules and functions without changing existing ones.
- The overall design framework and design remain constant in the project, but the software keeps evolving with every change in each iteration, making it more solid and complete in time.



Prototyping



Iterative & Evolutionary Fashion

In each iteration a prototype software is developed.

In the first iteration, only part of the specified requirements is implemented, and in each of the next iterations the new prototype implements an additional part of the requirements.

Each prototype is examined and evaluated by the customer and the user's team. Their demands for corrections, changes, and additions related to the current prototype are to be considered by the developer in the next integration prototype.

These iterations will continue till all the requirements are fulfilled.

Prototyping - Other concerns

Instead of throwing away, customer may request to fix problems to make the prototype a final product. But since prototype is built in a rush

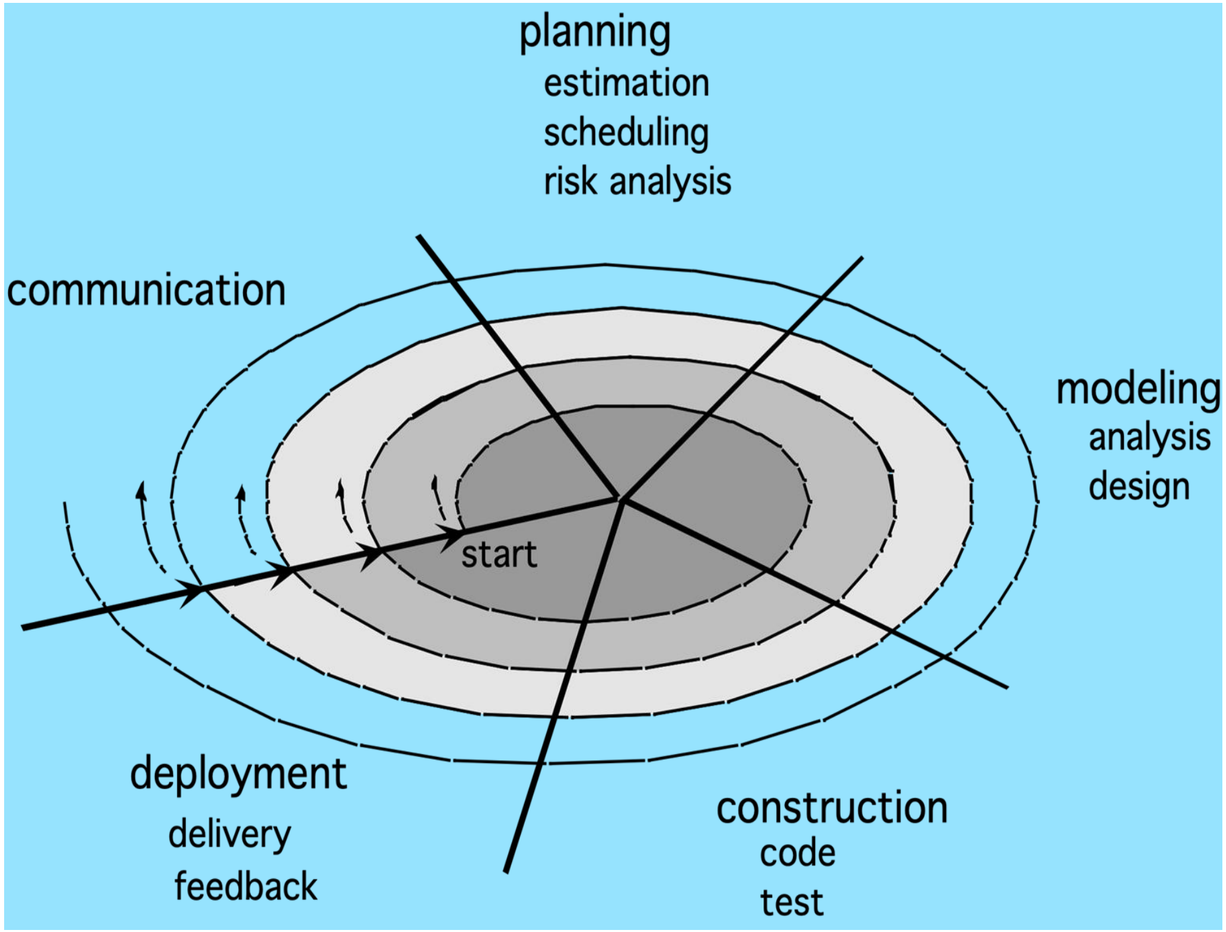
- It has a low quality design and implementation and
- Maintaining it would be very difficult for developer.

Since objective of prototype is just to build a demo, programming language and algorithms to build the system may be inappropriate for real system.

Developer may become comfortable with these choices, and forget all reasons why they were inappropriate.

Less than an ideal choice may become a part of real system.

Spiral Model



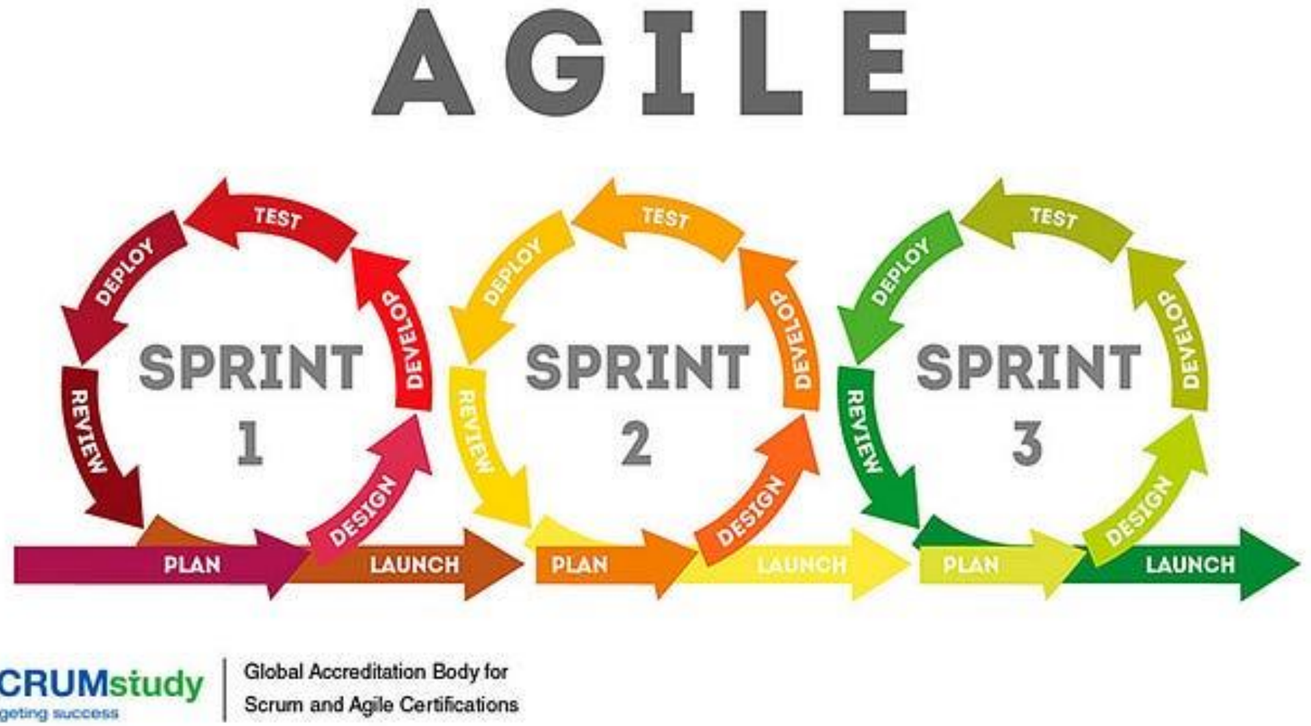
Iterative & Evolutionary Fashion

- Combined iterative fashion (6 m-2y) with systemics aspects of waterfall model
- Intended for large, expensive and complicated projects
- Increasingly more complete versions of software/product
- Early releases can be models or prototypes, later releases include increasingly more complete version of the system
- Combines iterative development with risk assessment, represented as a spiral. Risk analysis is conducted for each round:
  - emphasis on risk (evaluation and resolution)

Agile: Scrum

Scrum is the most popular model in Agile methodology.

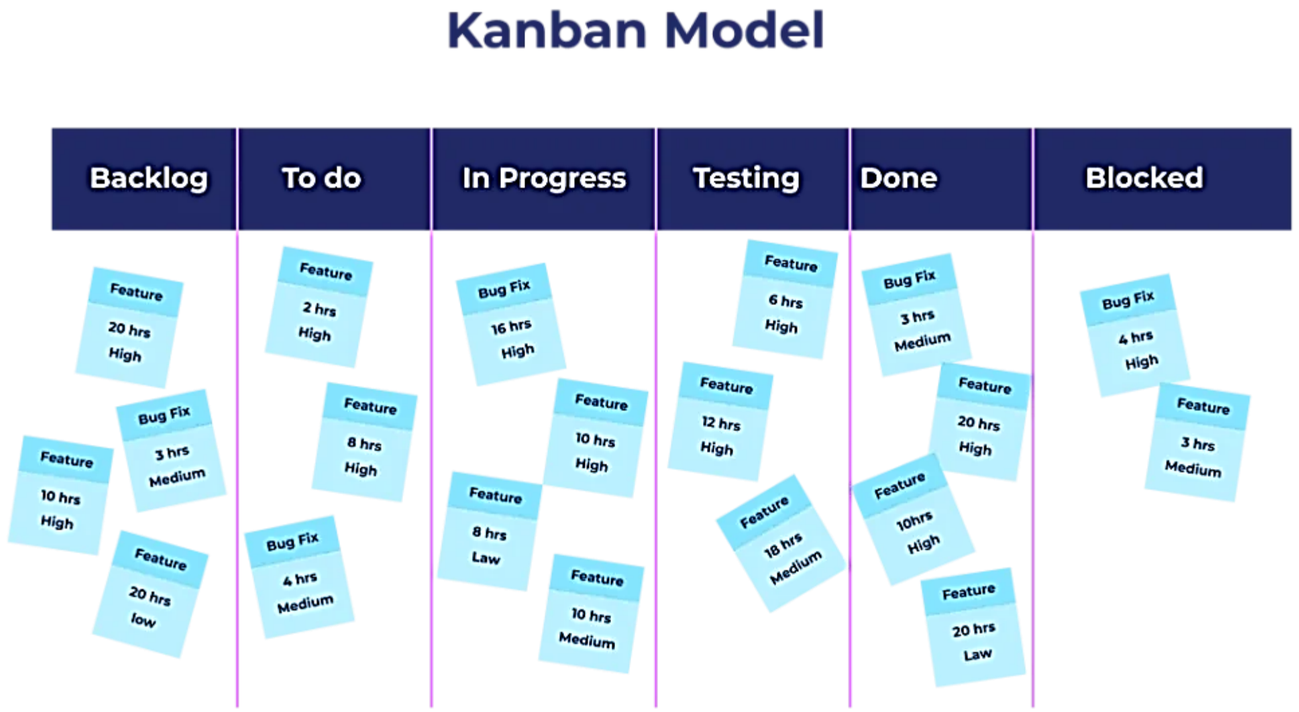
- The iterations in this software development model are known as sprints.
- Each sprint work is processed for a duration of 2 to 4 weeks.
- After each sprint, there is testing, adding new features and other functionalities, and then the next sprint is planned.
- In the Scrum model, the client feedback is constantly implemented into the sprints, helping the software to be business-focused and provide proper solutions.



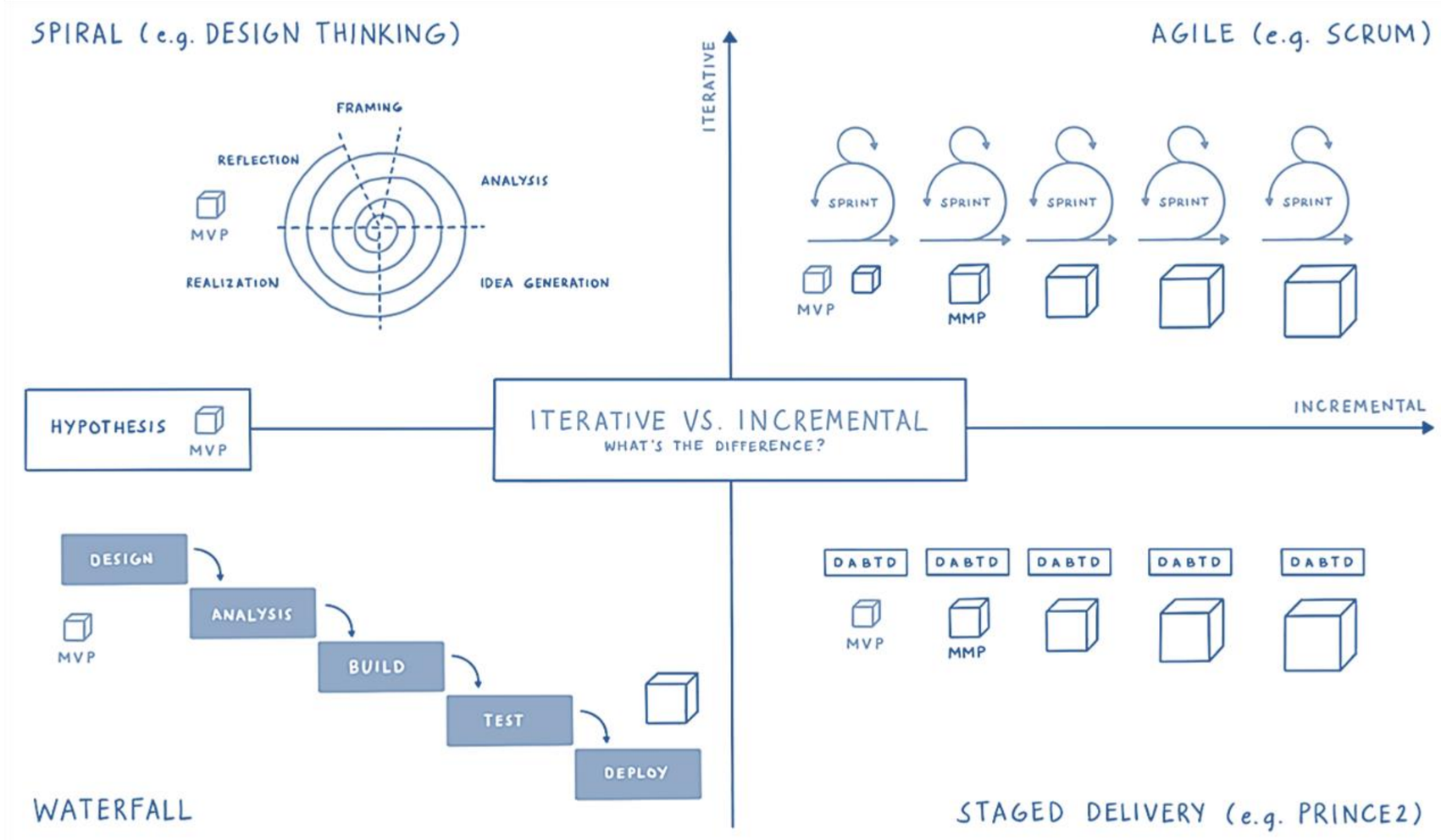
Agile: Kanban

Kanban is one of the rare SDLC models, as it works without any iterations.

- This software development model involves making a visual board for different tasks at various stages of development as part of the SDLC paradigm.
- The method is to have a sticky note board outlining the project and giving its details, such as progress status, people handling each phase, and more.
- The Kanban method makes it easy to see each working phase's work progress, update, and timeline.
- It ensures project transparency, organisation, and evaluation of priority tasks.



How do we decide?!

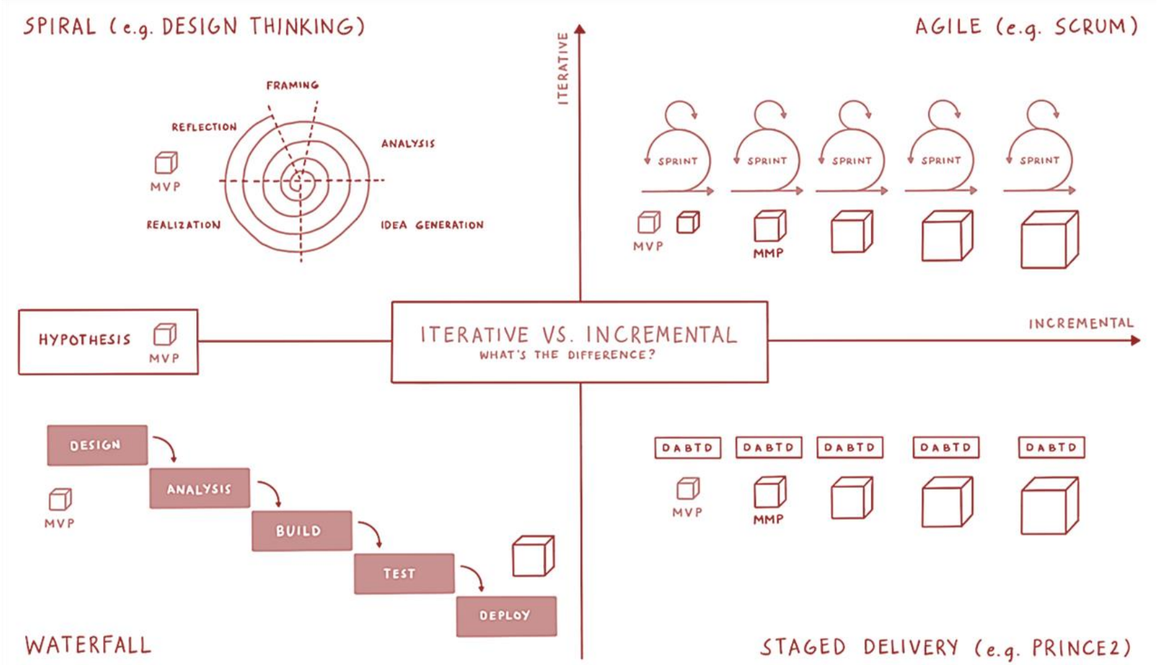


Factors Affecting Intensity of QA Activities in Development Process

Quality assurance planners for a project are required to determine the list of quality assurance activities needed.

For each quality assurance activity:

- Timing
- Type of quality assurance activity to be applied
- Who performs the activity and the resources required



Factors Affecting Intensity of QA Activities in Development Process

Project Factors

- Magnitude of the project
- Technical complexity and difficulty
- Extent of reusable software components
- Severity of failure outcomes if the project fails

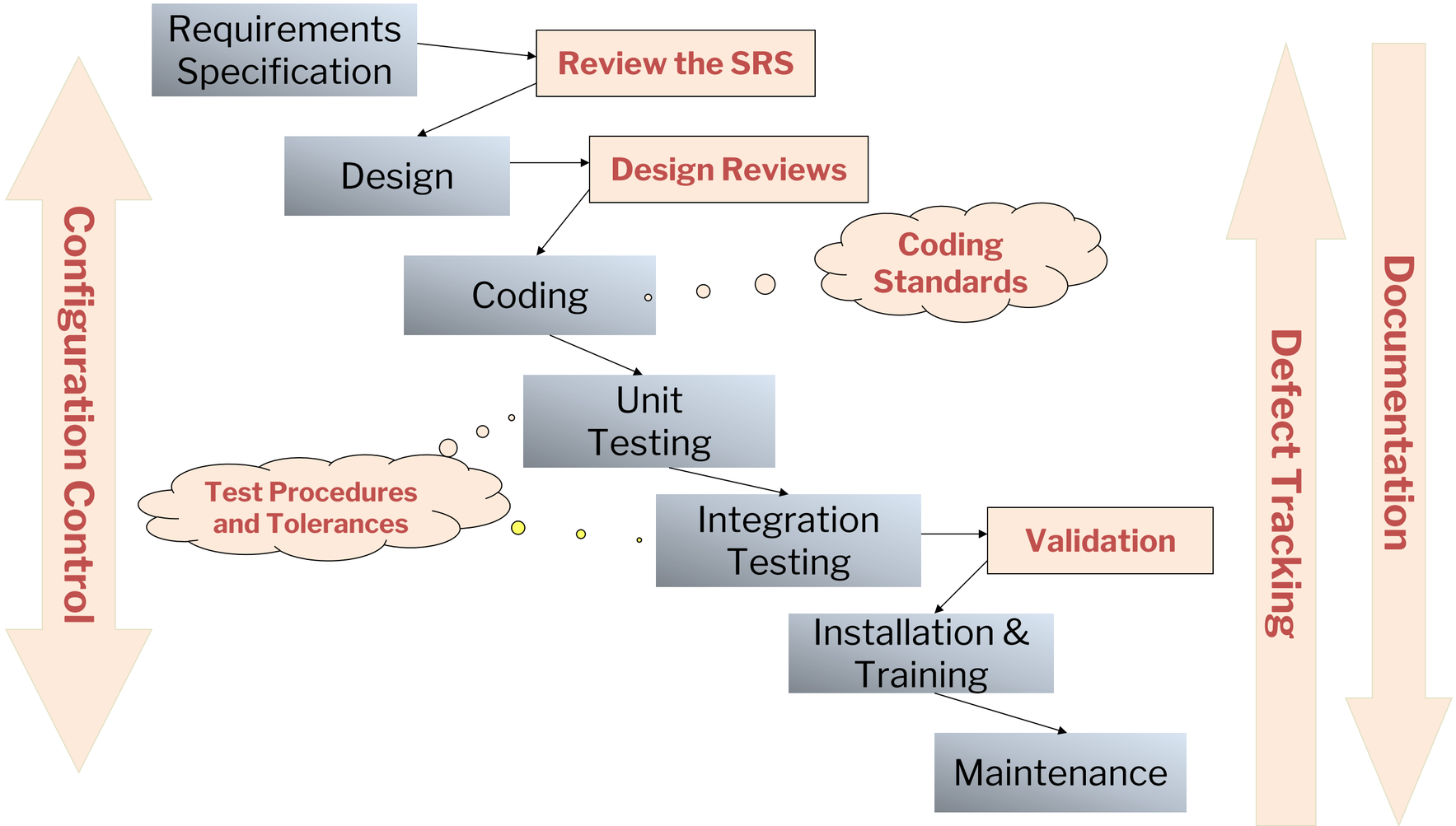
Team Factors

- Professional qualification of the team members
- Team acquaintance with the project and its experience in the area
- Availability of staff members who can professionally support the team

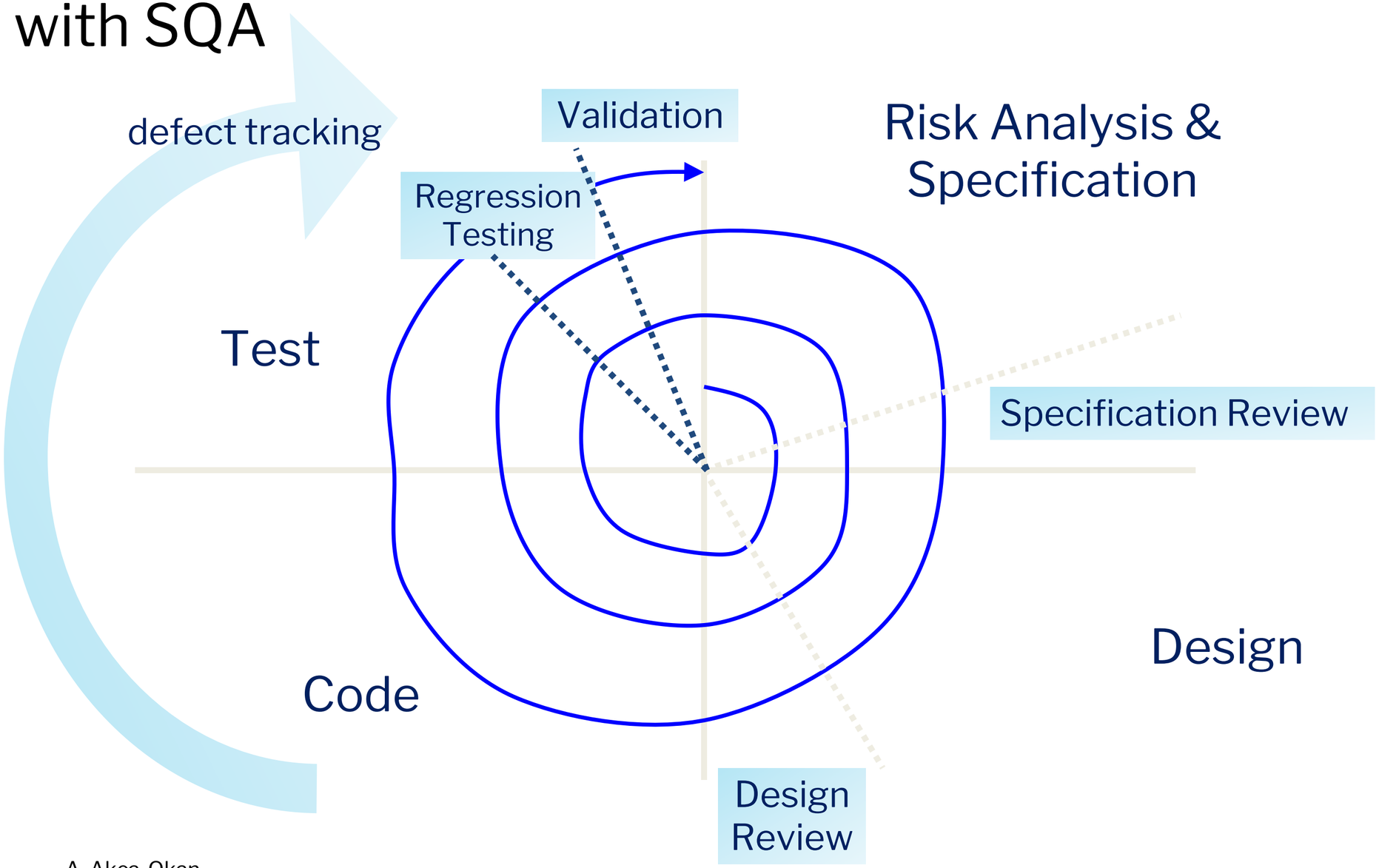
Primary Point

SQA Activities must be built into the project plan!

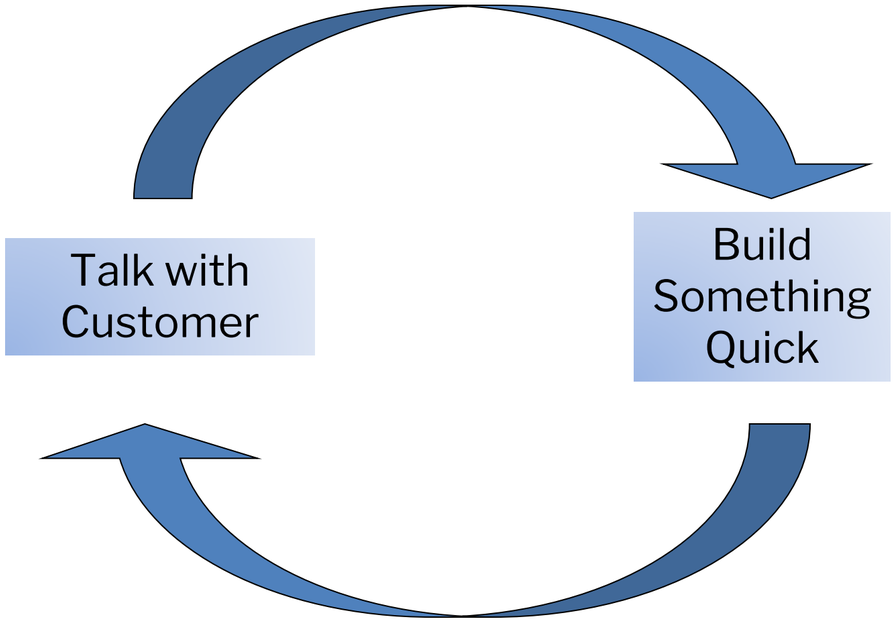
Waterfall with SQA Activities

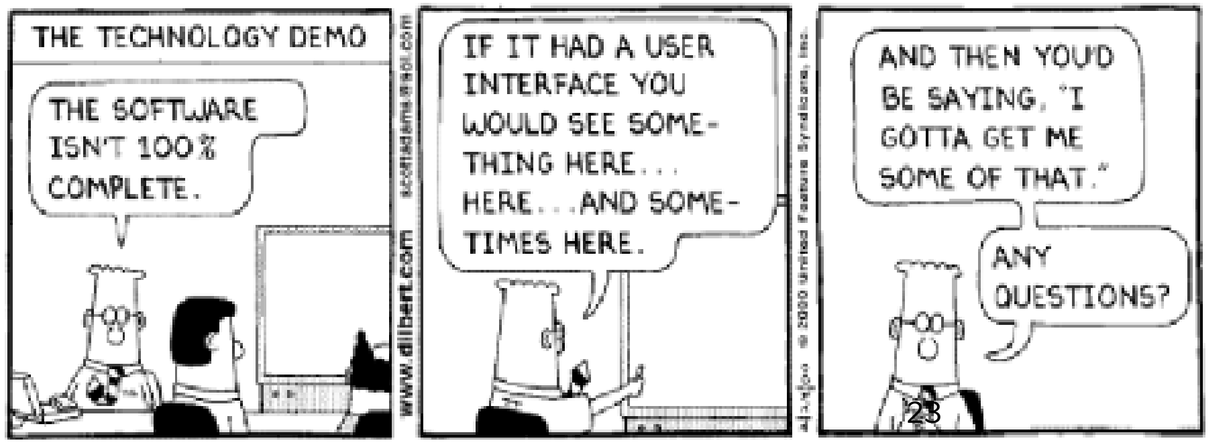


Spiral with SQA



Prototyping and SQA





SQA – Life Cycle

SQA specific activities- conducted throughout the SDLC referred as - SQA Life Cycle.

Initialisation Phase:

- Writing and reviewing management plan by QA dept.

Requirement Phase: Assuring that

- SR are complete and testable
- SR are properly expressed as functional, performance and interface requirements

Design Phase: Assuring that

- Approved design standards followed
- All SR are collected to software components, etc.

Implementation phase:

- Auditing the results of coding and design activities including schedule in Software development phase, status of all deliverable items.

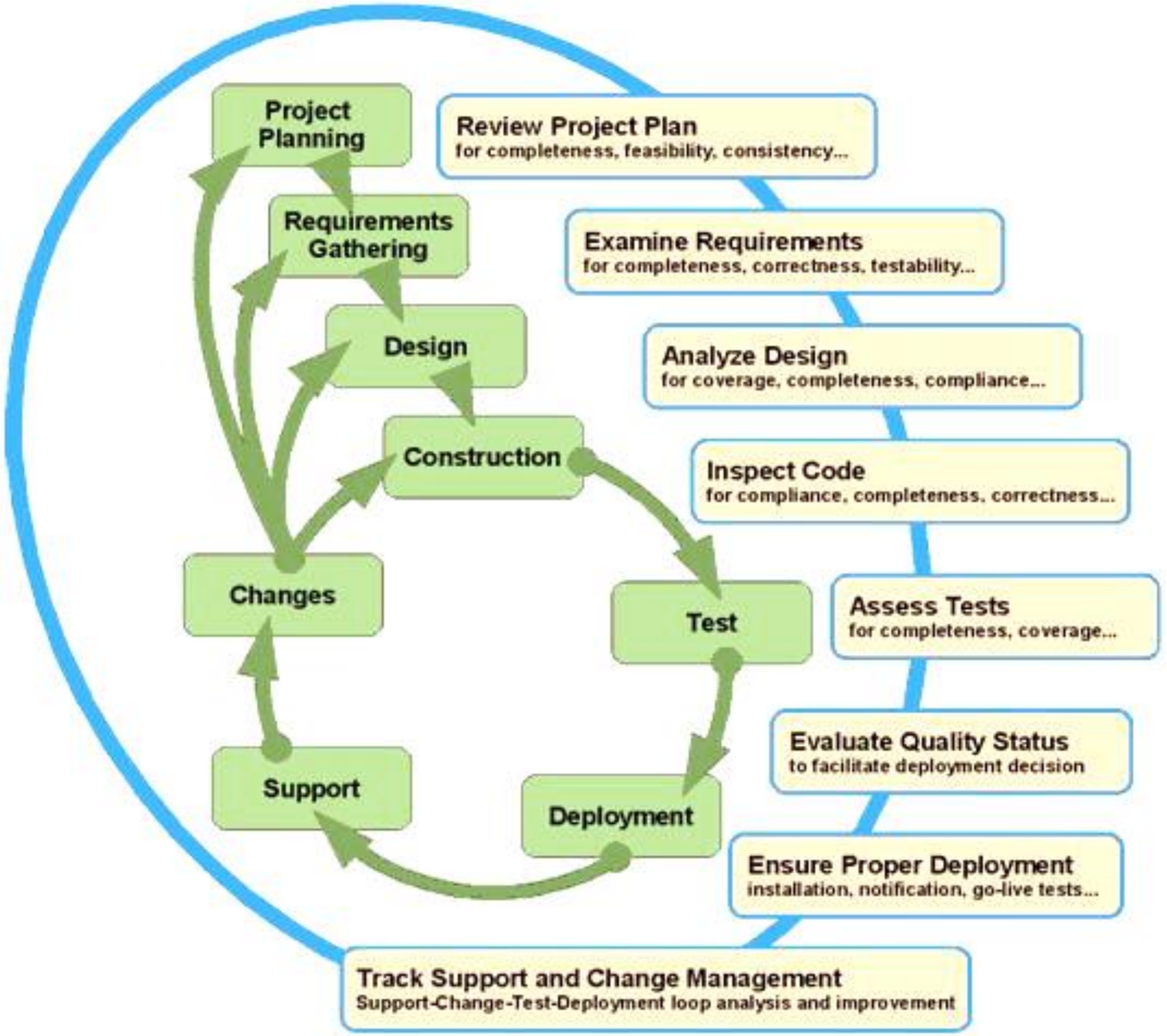
Integration and testing Phase: Assuring

- Readiness of all deliverables items and after testing ,test reports are complete and correct.
- All test run according to test plans and procedures

Acceptance and Delivery Phase: Assuring

- The readiness of all deliverable items

SQA – Software Life Cycle



SQA Activities

In order to maintain a high level of quality , the following SQA activities maybe identified:

1. Revision

- A draft version of an artifact /document submitted by the group will be evaluated by the members of SQA team.

Review is composed of

- Walkthrough
- Inspection
- Formal Review

2. Process Evaluation

SQA team is expected to

- Define the process standards such as how reviews should be conducted, and when reviews should be held
- Monitor the development process to ensure that the standards are being followed
- Report the software project management and to customer

3. Software Standards

- Develop and maintain the product and process standards

Documentation Standards

- Specify contents for planning, control , and product documentation

Design Standards

- Standards for evaluation of design

Code standards

- Specify the language in which the code is to be written and define any restrictions on use of language features

Example - 1

A software development team has planned the quality assurance activities for its new consumer club project.

- The current project contract, signed with a leading furniture store, is the team's 11th consumer club project dealing in the last three years.
- The team estimates that about seven man-months need to be invested by the two team members assigned to the project, whose duration is estimated at four months.
- It is estimated that a reusable components library can supply 90% of the project software.

The main considerations affecting quality assurance plan

➔ Degree of team acquaintance with the subject
➔ High percentage of software reuse
➔ Size of the project (in this case, medium)
➔ Severity of failure results if the project fails

Example - 1 Cont.

Three quality assurance activities were planned by the project leader. The quality assurance activities with their duration

No.	Quality assurance activity	Duration of quality assurance activity (days)	Duration of corrections and changes (days)
1	Design review of requirements definition	0.5	1
2	Inspection of the design	1	1
3	System test of completed software package	4	2

Example - 2

The real-time software development unit of a hospital's information systems department has been assigned to develop an advanced patient monitoring system.

- The new monitoring unit is to combine of patient's room unit with a control unit.
- The patient's room unit is meant to interface with several types of medical equipment, supplied by different manufacturers, which measure various indicators of the patient's condition.
- A sophisticated control unit will be placed at the nurses' station, with data to be communicated to cellular units carried by doctors.

- The project leader estimates that 14 months will be required to complete the system; a team of five will be needed, with an investment of a total of 40 man-months.
- She estimates that only 15% of the components can be obtained from the reusable component library.
- The SDLC methodology was chosen to integrate application of two prototypes of the patient's room unit and two prototypes of the control unit for the purpose of improving communication with the users and enhancing feedback of comments at the analysis and design phases.

Example - 2 Cont.

The main considerations affecting quality assurance plan

- ➔ High complexity and difficulty of the system
- ➔ Low percentage of reusable software available
- ➔ Large size of the project
- ➔ High severity of failure outcomes if the project fails

The quality assurance activities with their duration

No.	Quality assurance activity	Duration of quality assurance activity (days)	Duration of corrections and changes (days)
1	Design review of requirements definition	2	1
2	Design review of analysis of patient's room unit	2	2
3	Design review of analysis of control unit	1	2
4	Design review of preliminary design	1	1
5	Inspection of design of patient's room unit	1	2
6	Inspection of design of control unit	1	3
7	Design review of prototype of patient's room unit	1	1
8	Design review of prototype of control unit	1	1
9	Inspection of detailed design for each software interface component	3	3
10	Design review of test plans for patient's room unit and control unit	3	1
11	Unit tests of software code for each interface module of patient's room unit	4	2
12	Integration test of software code of patient's room unit	3	3
13	Integration test of software code of control unit	2	3
14	System test of completed software system	10	5
15	Design review of user's manual	3	2

Discussion Question

A software development team is working on a new mobile banking application for a mid-sized bank aiming to expand its digital services.

- This is the third mobile application the team has developed, but the first project with a focus on financial services, which comes with additional security and regulatory requirements.
- The project is estimated to require eight man-months of effort, with three team members assigned over a period of five months.
- It is anticipated that only 30% of the code can be sourced from existing reusable components due to the unique nature of the financial services and integrations with banking APIs.
- The technical complexity is high, as the app must include secure authentication, data encryption, and real-time transaction monitoring.
- A failure in the app could result in significant financial losses and damage to the bank's reputation.

Discussion Question - Cont.

The main considerations affecting quality assurance plan

Project Factors

- ❏ Magnitude of the project
- ❏ Technical complexity and difficulty
- ❏ Extent of reusable software components
- ❏ Severity of failure outcomes if the project fails

Team Factors

- ❏ Professional qualification of the team members
- ❏ Team acquaintance with the project and its experience in the area
- ❏ Availability of staff members who can professionally support the team

- ➔ The team's moderate experience with mobile applications but limited exposure to the financial domain.
- ➔ A low percentage of reusable components, indicating a need for extensive custom development and testing.
- ➔ The medium-to-large size of the project.
- ➔ High severity of failure results due to potential financial and reputational consequences.

Discussion Question - Cont.

Some of quality assurance activities with their estimated durations

Name of Activity	Activities Applied to the Mobile Banking Project	Estimated Duration (Days)	Duration of Corrections (Days)
Contract Reviews	Review customer requirements for security and compliance, project schedule, resources, and potential risks for financial services.	10	3
Reviews	Conduct formal reviews of security features, data encryption methods, and compliance documents before progressing through development stages.	12	4
Expert Opinions	Engage external security experts to review encryption and authentication methods to ensure regulatory standards are met.	7	3
Staff Training and Certification	Arrange specialized training on secure coding practices and financial service regulations to strengthen team expertise.	14	5
Compliance with Standards	Ensure that the project adheres to relevant financial software standards and regulatory requirements, such as ISO/IEC 12207.	9	3

Verification, Validation and Qualification

Verification – The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Validation - The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements .

Qualification - The process used to determine whether a system or component is suitable for operational use.

IEEE Std 610.12-1990 (IEEE 1990)

Verification, Validation and Qualification

Verification – Examines consistency of the product being developed with products developed in previous phases

Validation - Represents customer interests by examining the extent of compliance to his/her original requirements

Qualification – Qualification on operational aspects where maintenance is the main issue

IEEE Std 610.12-1990 (IEEE 1990)

Model for SQA defect removal effectiveness and cost

The model's quantitative results

The model deals with 2 quantitative aspects of SQA plan:

a. The SQA plan's total effectiveness in removing project defects

b. The total costs of removal of project defects

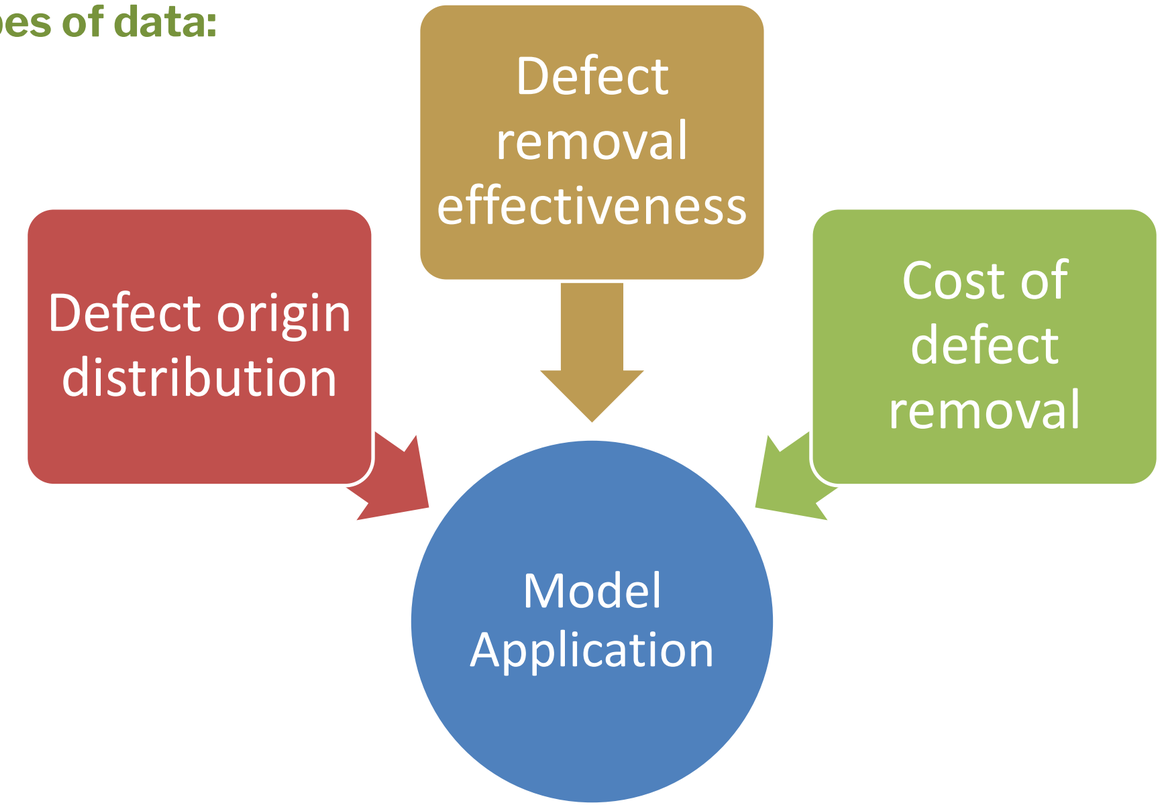
The model allows calculating estimates of the cost of decisions regarding the quality assurance plan, e.g.:

- Addition or elimination of a quality assurance activity from a given plan.
- Application of current quality assurance procedures activity versus application of a more efficient yet more costly procedure.

Utilisation of the model thus enables comparison of SQA policies/strategies and activity plans.

Model for SQA defect removal effectiveness and cost

The application of the model is based on 3 types of data:



- Defect origin distribution in which phase did the defects occur
- Defect removal effectiveness, and how effective are we at removal of defects?
- Cost of defect removal. how much does it cost per defect per phase

Model for SQA defect removal effectiveness and cost

The model's quantitative results

The model deals with 2 quantitative aspects of SQA plan:

a. The SQA plan's total effectiveness in removing project defects

b. The total costs of removal of project defects

Defects originating and defect removal costs

Software development phase	Average % of defects originating in phase	Average relative defect removal cost
Requirement specification	15%	1
Design	35%	2.5
Unit coding	30%	6.5
Integration coding	10%	16
Documentation	10%	40
System testing	-----	40
Operation	-----	110

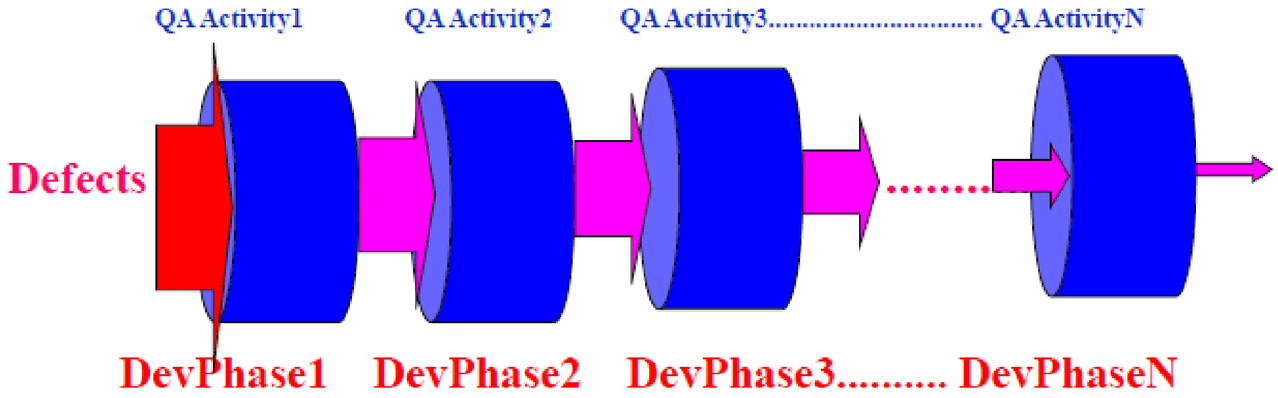
The phase of defect introduction

Distributed throughout the development process

Defect removal effectiveness

Generally, the percentage of removed defects is lower than the percentage of detected defects, because some corrections are ineffective or inadequate.

- **We simply miss some!!**
- Others are undetected and uncorrected and passed on to successive development phases.
- **Lingering defects coupled with introduced defects in current development phase add up!!!**
- For discussion purposes, we will assume the filtering effectiveness of accumulated defects of each quality assurance activity is not less than 40%, that is, each activity removes at least 40% of the incoming defects.



An activity removes at least 40% of the incoming defects.

Defects removal effectiveness for quality assurance plans

Quality assurance activity	Defects removal effectiveness for standard SQA plan	Defects removal effectiveness for comprehensive SQA plan
Specification requirement review	50%	60%
Design inspection	-----	70%
Design review	50%	60%
Code inspection	-----	70%
Unit test	50%	40%
Integration tests	50%	60%
Documentation review	50%	60%
System test	50%	60%
Opertion phase detection	100%	100%

Cost Removal

- Removal of defects differs very significantly by development phase.
- **Costs are MUCH greater in later development phases.**

Note: In general, defect removal data is not commonly available.

- Most agree with the data based on key studies.

Cost Progression: Based on typical data, the **cost of removing a detected defect (CDR) increases exponentially** across the life cycle:

- Requirement Specification Review (CDR: 1 cost unit)
- Design Review (CDR: 2.5 cost units)
- Unit Test (CDR: 6.5 cost units)
- System Test (CDR: 40 cost units)
- Operation Phase (CDR: 110 cost units)

Defects removal effectiveness for quality assurance plans

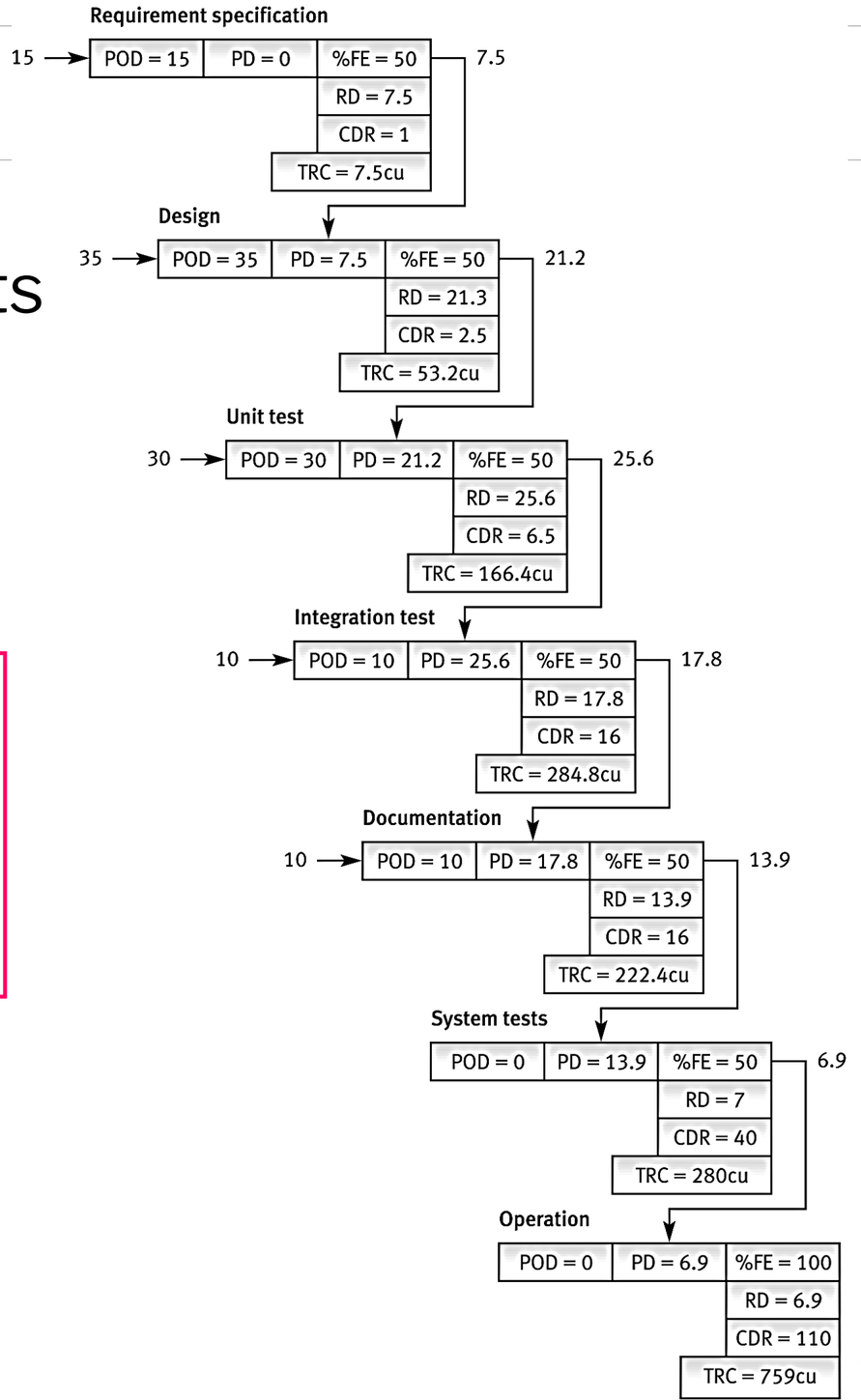
Defect removal phase	Defect removal effectiveness	Average relative defect removal cost {cost unit}				
		Defect origination phase				
		Req	Des	Uni	Int	Doc
Requirement specification (Req)	50%	1	---	---	---	---
Design (Des)	50%	2.5	1	---	---	---
Unit coding (Uni)	50%	6.5	2.6	1	---	---
Integration (Int) System documentation (Doc)	50% 50%	16 16	6.4 6.4	2.5 2.5	1	1
System testing / Acceptance testing (Sys)	50%	40	16	6.2	2.5	1
Operation by customer (after release)	100%	110	44	17	6.9	2.5

Model Assumptions

- Development process (DP) is linear and sequential
- DP is waterfall
- New defects are introduced in each development phase
- Review and SQA activities are filters: Review and test software quality assurance activities serve as filters, removing a percentage of defects and letting the rest pass to next development phase
- Filtering efficiency is the same for each defect
- In the model, each of the quality assurance activities is represented by a filter unit
- At each phase, incoming defects are the sum of those not removed plus new defects in the current phase
- Incoming defects = New defects + Defects not removed
- Average defect removal is the same for all phases (1 unit)
- Cost of defect removal is calculated for each SQA activity by multiplying the number of defects removed by the relative cost of removing a defect.
- Defect Removal Cost = # of removed defects x relative cost of removing one defect
- Remaining defects after final stage will be passed to customer

But we can do better using a comprehensive quality assurance plan with more activities, and hence better filtering!!

The standard quality assurance plan
The process of removing 100 defects



- **POD =** Phase Originated Defects
- **PD =** Passed Defects (from former phase or former quality assurance activity)
- **%FE =** % of Filtering Effectiveness (also termed % screening effectiveness)
- **RD =** Removed Defects
- **CDR =** Cost of Defect Removal
- **TRC =** Total Removal Cost. TRC = RD x CDR.

A filter unit for defect-removal effectiveness and costs

Design phase (POD=35)

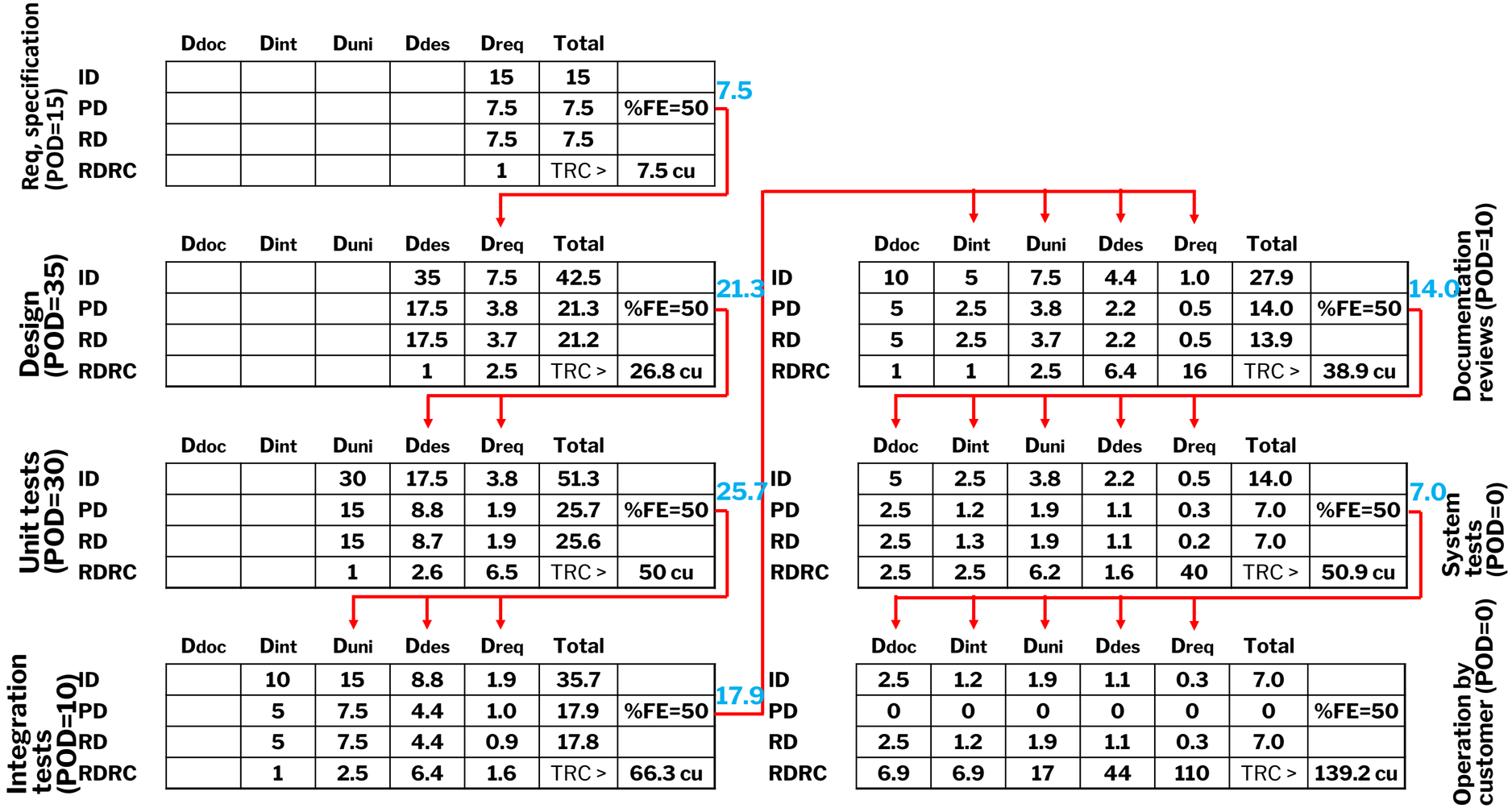
7.5

Design phase (POD=35)	Ddoc	Dint	Duni	Ddes	Dreq	Total	
ID				**35**	**7.5**	**42.5**	
PD				**17.5**	**3.8**	**21.3**	**%FE=50**
RD				**17.5**	**3.7**	**21.2**	
RDRC				**1**	**2.5**	TRC >	**26.8cu**

21.3

ID : Incoming Defects
PD : Passed Defects
RD : Removed Defects
RDRC : Removed Defect Relative Cost
TRC = RD x RDRC

Defect correction effectiveness and cost - standard plan model of the process of correction 100 defects



ID : Incoming Defects
PD : Passed Defects
RD : Removed Defects
RDRC : Removed Defect Relative Cost
TRC = RD x RDRC