INTEGRATING QUALITY ACTIVITIES IN THE PROJECT LIFE CYCLE

TOPICS

- Classic and other software development methodologies.
- Factors affecting intensity of quality assurance activities in the development process.
- Process Model:ETVX
- Verification, validation and qualification
- A model for SQA. defect removal effectiveness and cost.

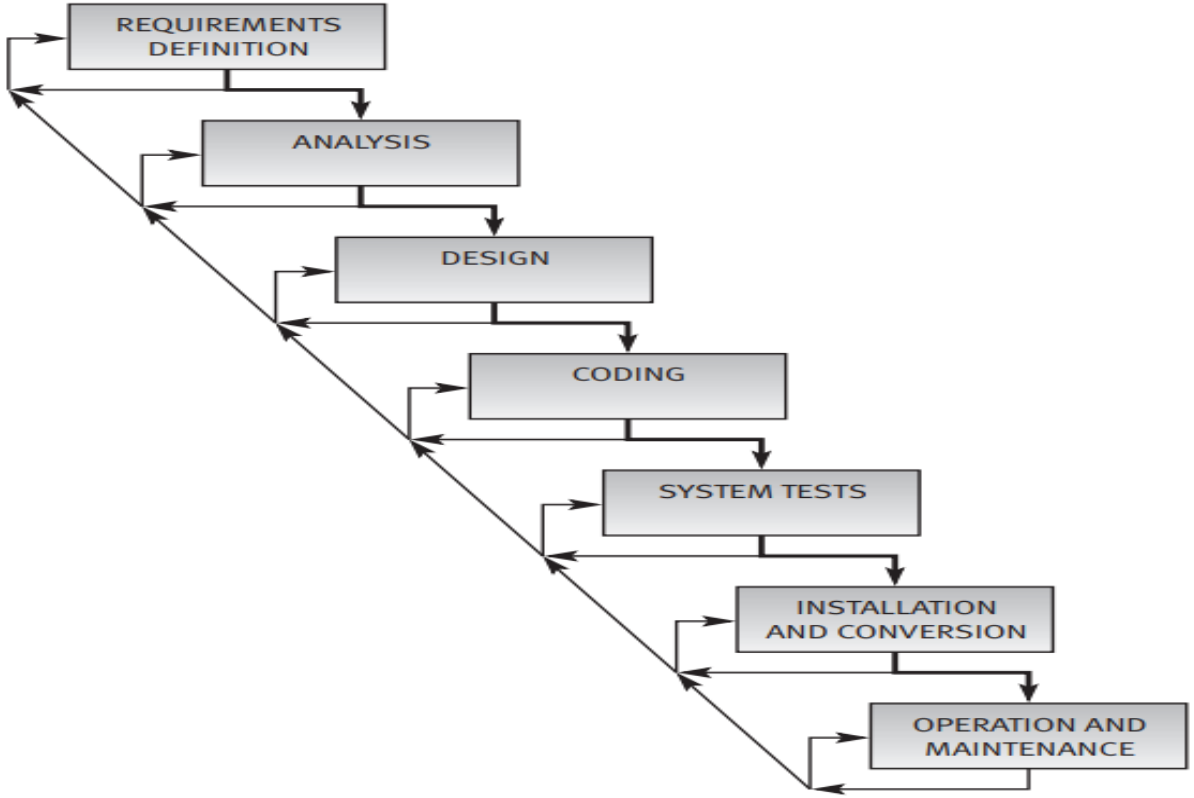
CLASSIC AND OTHER SOFTWARE DEVELOPMENT METHODOLOGIES

- The Software Development Life Cycle (SDLC) model
- The prototyping model
- The spiral model
- The object-oriented model.

THE SOFTWARE DEVELOPMENT LIFE CYCLE MODLE

- Classic model.
- The model displays the major building blocks for the entire development process.
- The most common illustration of the SDLC model is the *waterfall model*

WATERFALL MODEL



THE WATERFALL MODEL SEVEN-PHASE PROCESSES

- Requirements definition.
  - The customers must define their requirements. In many cases the software system is part of a larger system. Information about the other parts of the expanded system helps establish cooperation between the teams and develop component interfaces.

THE WATERFALL MODEL SEVEN-PHASE PROCESSES

- Analysis
  - The main effort here is to analyze the requirements' implications to form the initial software system model.

THE WATERFALL MODEL SEVEN-PHASE PROCESSES

- Design.
  - This stage involves the detailed definition of the outputs, inputs and processing procedures, including data structures and databases, software structure

THE WATERFALL MODEL SEVEN-PHASE PROCESSES

- Coding.
  - The design is translated into a code.
  - Coding involves quality assurance activities such as inspection, unit tests and integration tests.

THE WATERFALL MODEL SEVEN-PHASE PROCESSES

- System tests.
  - The main goal of testing is to *uncover as many software errors* as possible so as to achieve an acceptable level of software quality once corrections have been completed.

THE WATERFALL MODEL SEVEN-PHASE PROCESSES

- Installation and conversion.
  - The system is installed to serve as firmware, that is, as part of the information system that represents a major component of the expanded system.
  - If the new information system is to replace an existing system, a software conversion process has to be initiated *to make sure that the organization's activities continue uninterrupted* during the conversion phase.

THE WATERFALL MODEL SEVEN-PHASE PROCESSES

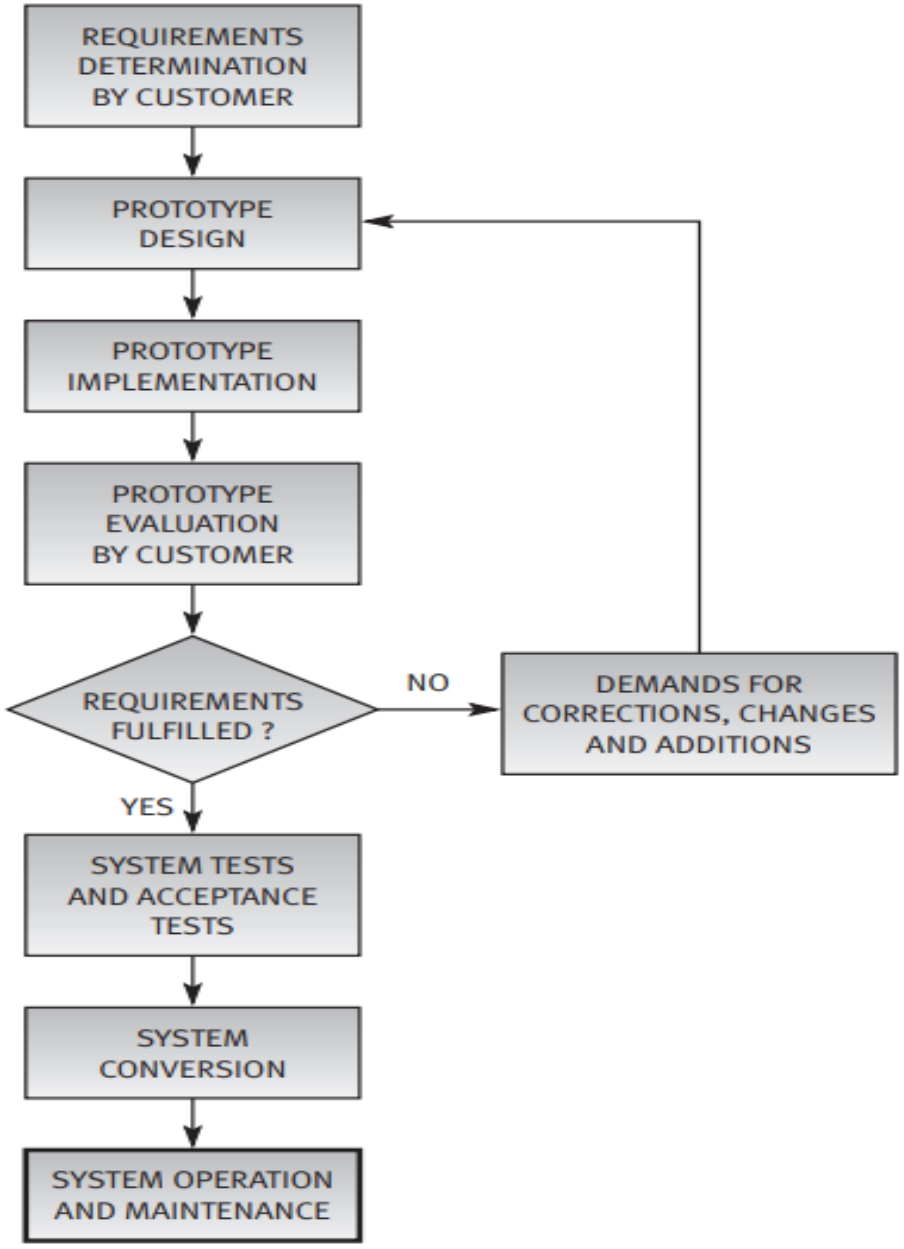
- Regular operation and maintenance.
  - Throughout the regular operation period, which usually lasts for several years or until a new software generation appears on the scene, maintenance is needed.
  - Maintenance incorporates three types of services.
    - Corrective – repairing software faults
    - Adaptive – using the existing software features to fulfill new requirements
    - Perfective – adding new minor features to improve software performance

THE PROTOTYPING MODEL

The prototyping methodology makes use of..

- developments in information technology, namely, advanced application generators that allow for fast and easy development of software prototypes.
- active participation in the development process by customers and users capable of examining and evaluating prototypes.

The Prototyping Model



THE PROTOTYPING MODEL

- Prototyping as a software development methodology has been found to be efficient and effective mainly for small- to medium-sized software development projects.

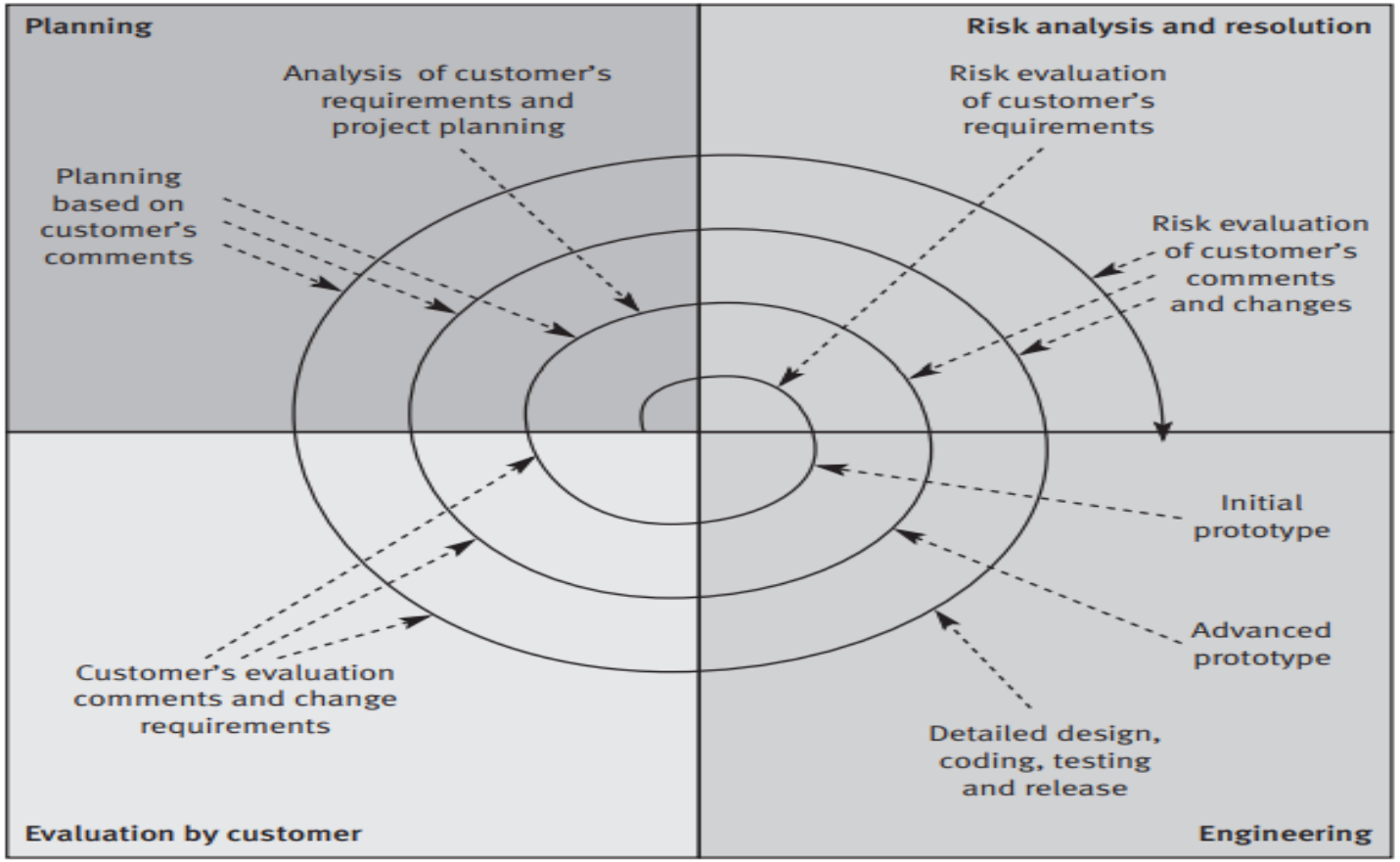
PROTOTYPING VS SDLC METHODOLOGY

- Advantages of prototyping:
  - Shorter development process.
  - Substantial savings of development resources(man-days).
  - Better fit to customer requirements and reduced risk of project failure.
  - Easier and faster user comprehension of the new system.

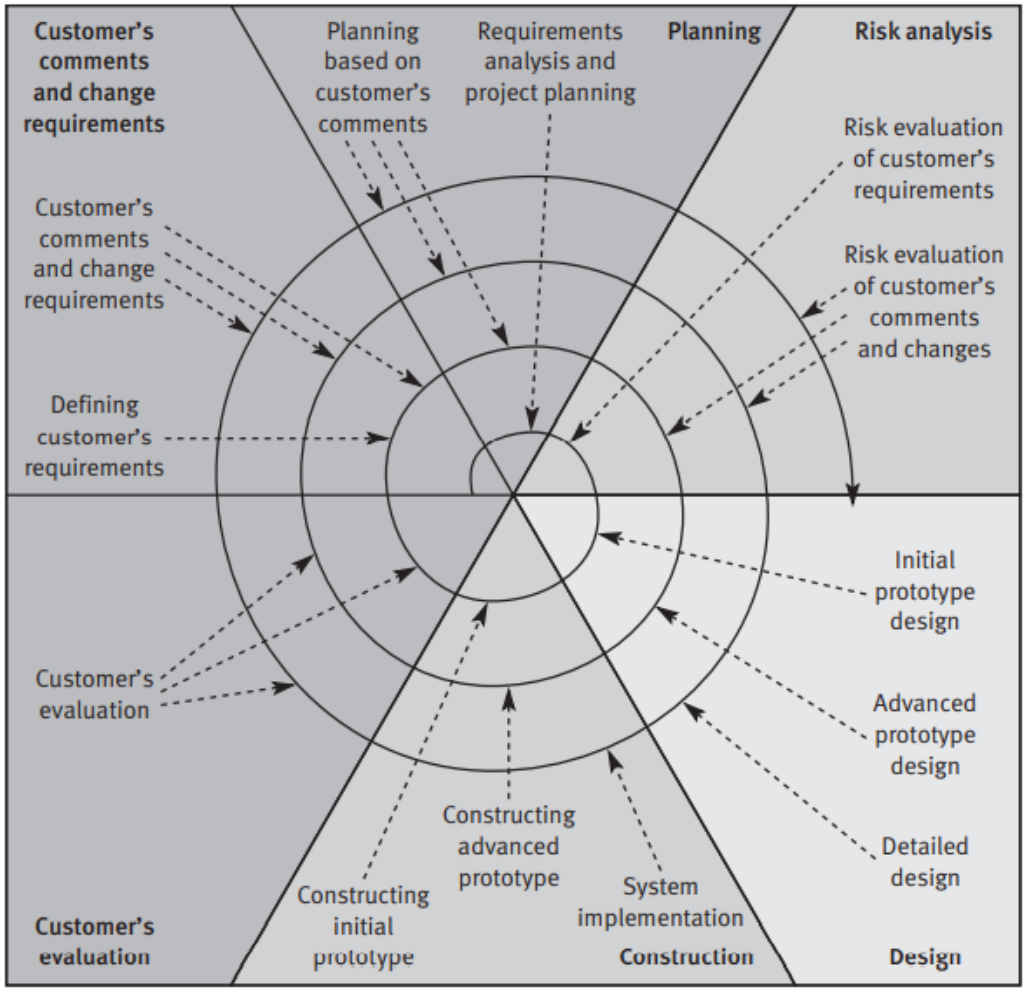
THE SPIRAL MODEL

- The spiral model, as revised by Boehm (1988, 1998), offers an improved methodology for overseeing large and more complex development projects displaying higher prospects for failure, typical of many projects begun in the last two decades.

THE SPIRAL MODEL



THE ADVANCED SPIRAL MODEL

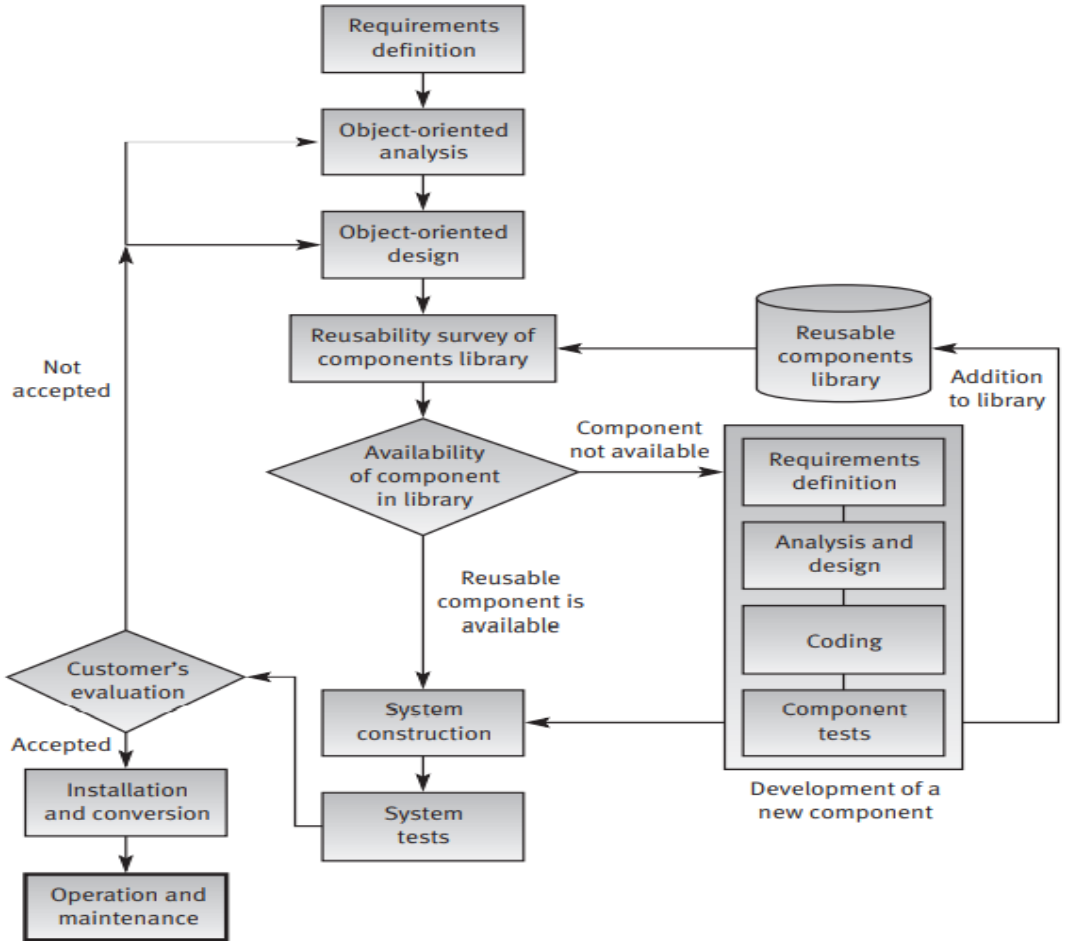


THE OBJECT-ORIENTED MODEL

- The object-oriented model differs from the other models by its intensive reuse of software components.

- This methodology is characterized by its easy integration of existing software modules (called objects or components) into newly developed software systems.

THE OBJECT-ORIENTED MODEL



ADVANTAGE OF OBJECT-ORIENTED MODEL

- Economy – The cost of integrating a reusable software component is much lower than the cost of developing new components.
- Shorter development time – The integration of reusable software components reduces scheduling pressures.
- Improved quality – Used software components are expected to contain considerably fewer defects than newly developed software components due to detection of faults by former users.

FACTORS AFFECTING INTENSITY OF QUALITY ASSURANCE ACTIVITIES IN THE DEVELOPMENT PROCESS

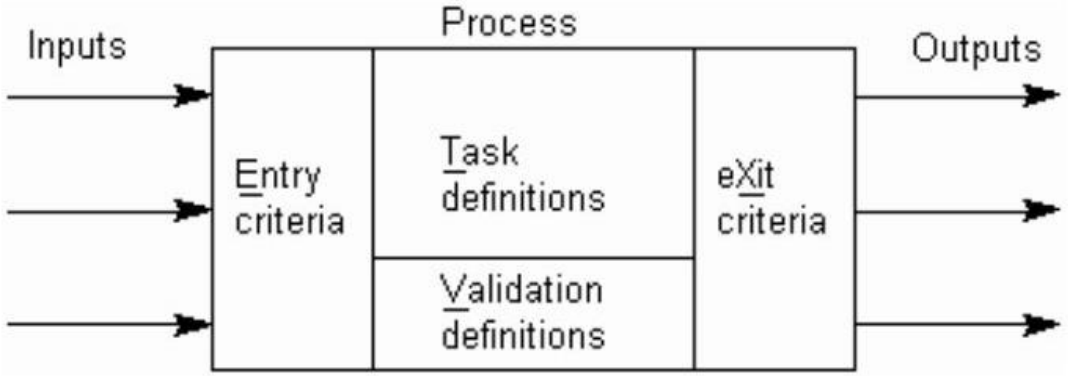
Quality assurance planners for a project are required to determine:

- The list of quality assurance activities needed for a project.

- For each quality assurance activity:
  - Timing
  - Type of quality assurance activity to be applied
  - Who performs the activity and the resources required.
  - Resources required for removal of defects and introduction of changes.

PROCESS MODEL: ETVX

- ETVX: Entry-Task-Validation-Exit



PROCESS MODEL: ETVX

- ***Entry criteria*** define what inputs are required and what quality these must be to achieve the exit criteria. Entry criteria should be communicated to supplier processes, to become their exit criteria. If supplier processes are sufficiently well controlled, then there is no need to check inputs.

- ***Task definitions*** specify the actions within the process.

PROCESS MODEL: ETVX

- ***Validation definitions*** identify test points within the process and define the tests and criteria for checking at these points. This enables problems to be caught close to their cause, reducing rework and scrap costs, and enabling problem causes to be addressed.

- ***Exit criteria*** define what outputs are required and what quality these must be to meet the needs of customer processes. Exit criteria may be derived from the entry criteria of customer processes.

VERIFICATION

- "**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

- "**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

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

A MODEL FOR SQA DEFECT REMOVAL EFFECTIVENESS AND COST

- The model deals with two quantitative aspects of an SQA plan consisting of several defect detection activities:

1. The plan's total effectiveness in removing project defects.
2. The total costs of removal of project defects

THE DATA

- Defect origin distribution
- Defect removal effectiveness
- Cost of defect removal

DEFECT ORIGIN DISTRIBUTION

- Defect origins (the phase in which defects were introduced) are distributed throughout the development process, from the project's initiation to its completion

Table 7.3: A characteristic distribution of software defect origins

No.	Software development phase	Average percentage of defects originating in phase
1	Requirements specification	15%
2	Design	35%
3	Coding (coding 30%, integration 10%)	40%
4	Documentation	10%

DEFECT REMOVAL EFFECTIVENESS

- It is assumed that any quality assurance activity filters (screens) a certain percentage of existing defects. It should be noted that in most cases,
- The percentage of removed defects is somewhat lower than the percentage of detected defects as some corrections are ineffective or inadequate.
- The remaining defects, those undetected and uncorrected, are passed to successive development phases.

DEFECT REMOVAL EFFECTIVENESS

Table 7.4: Average filtering (defect removal) effectiveness by quality assurance activities

No.	Quality assurance activity	Average defect filtering effectiveness rate
1	Requirements specification review	50%
2	Design inspection	60%
3	Design review	50%
4	Code inspection	65%
5	Unit test	50%
6	Unit test after code inspection	30%
7	Integration test	50%
8	System tests / acceptance tests	50%
9	Documentation review	50%

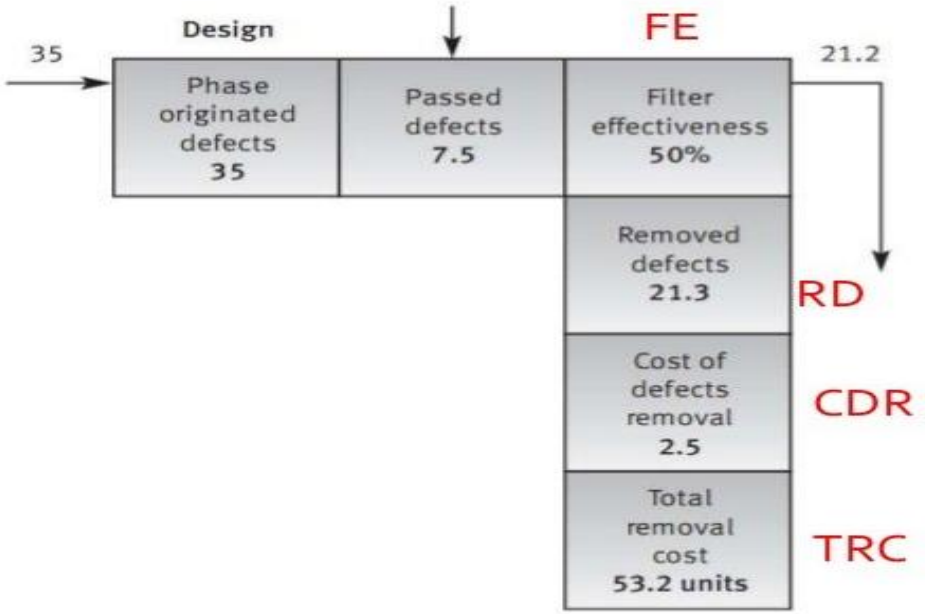
COST OF DEFECT REMOVAL

Table 7.5: Representative average relative defect-removal costs

No.	Software development phase	Average relative defect cost (cost units)
1	Requirements specification	1
2	Design	2.5
3	Unit tests	6.5
4	Integration tests	16
5	System tests / acceptance tests / system documentation review	40
6	Operation by customer (after release)	110

THE MODEL

- POD = Phase Originated Defects (from Table 7.3)
- PD = Passed Defects (from former phase or former quality assurance activity)



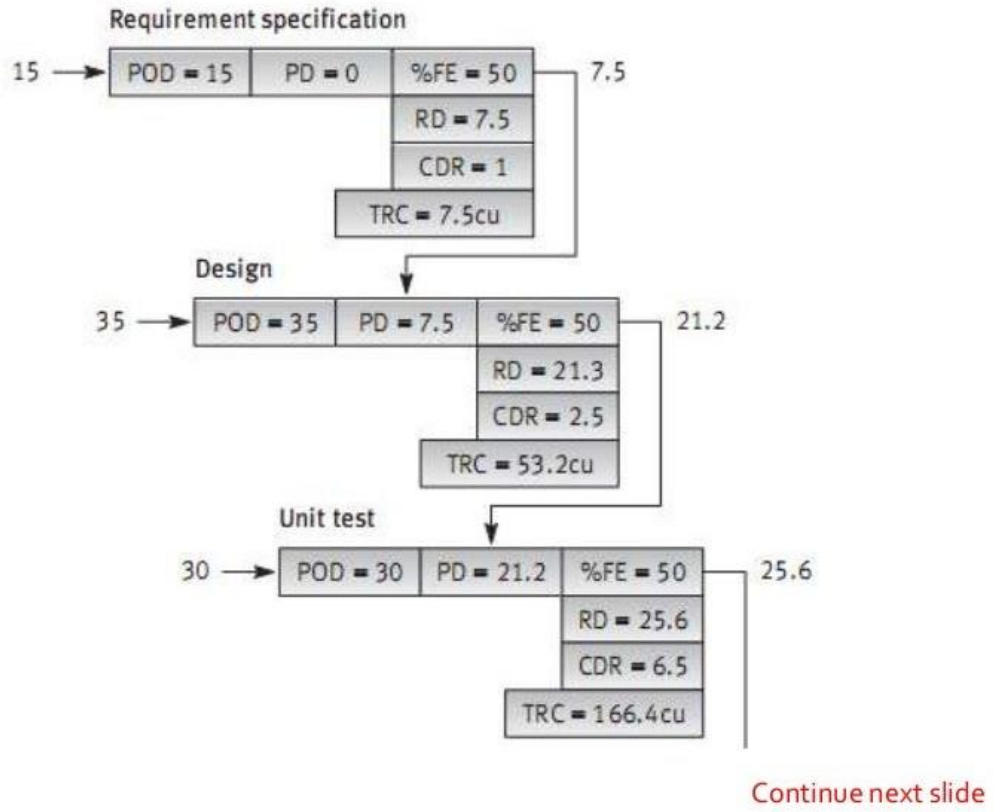
EXAMPLE I STANDERS QA PLAN

- the model applies to a standard quality assurance plan that is composed of six quality assurance activities as shown in Table 7.6.

Table 7.6: Standard quality assurance plan

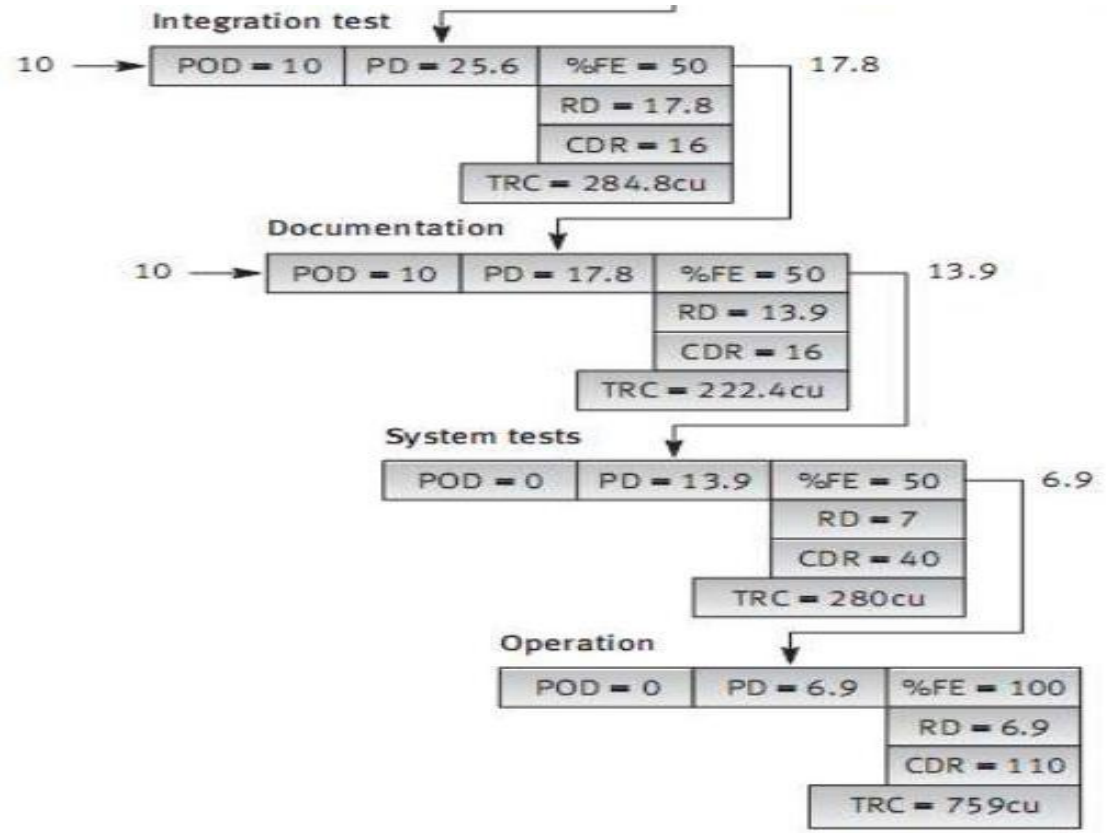
No.	Quality assurance activity	Defect removal effectiveness	Cost of removing a detected defect (cost units)
1	Requirement specification review	50%	1
2	Design review	50%	2.5
3	Unit test – code	50%	6.5
4	Integration test	50%	16
5	Documentation review	50%	16
6	System test	50%	40
7	Operation phase	100%	110

EXAMPLE I STANDARDS QA PLAN



Continue next slide

EXAMPLE I STANDARDS QA PLAN

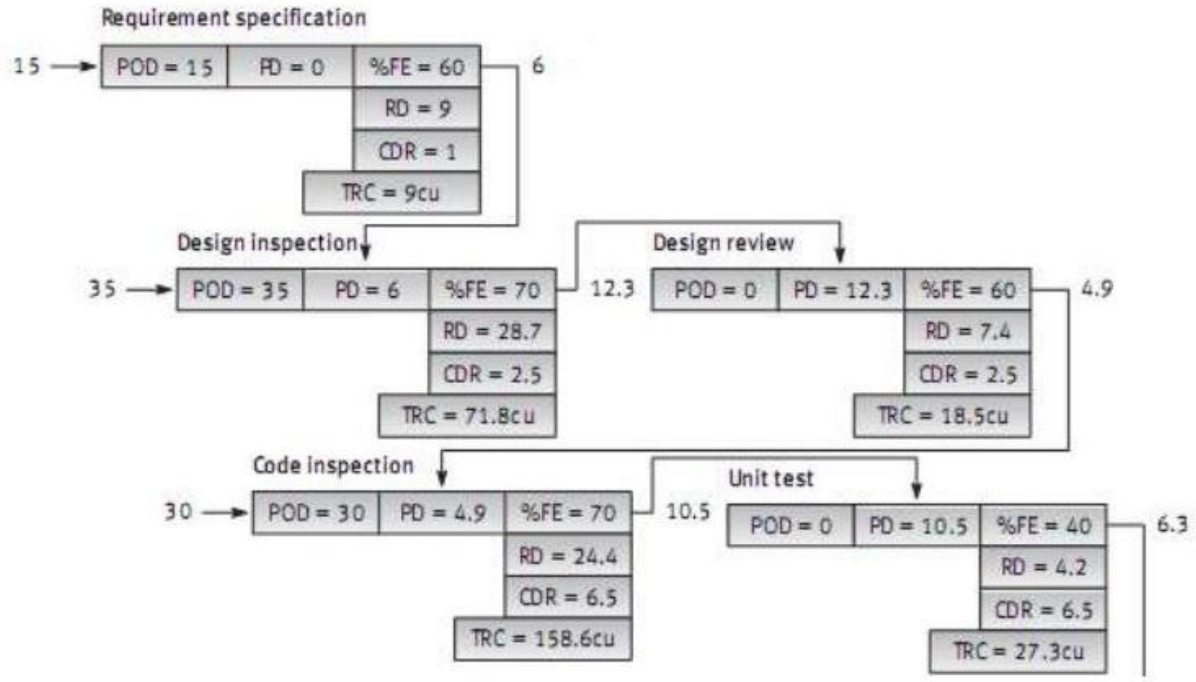


EXAMPLE 2 COMPREHENSIVE QA PLAN

Table 7.7: Comprehensive quality assurance plan

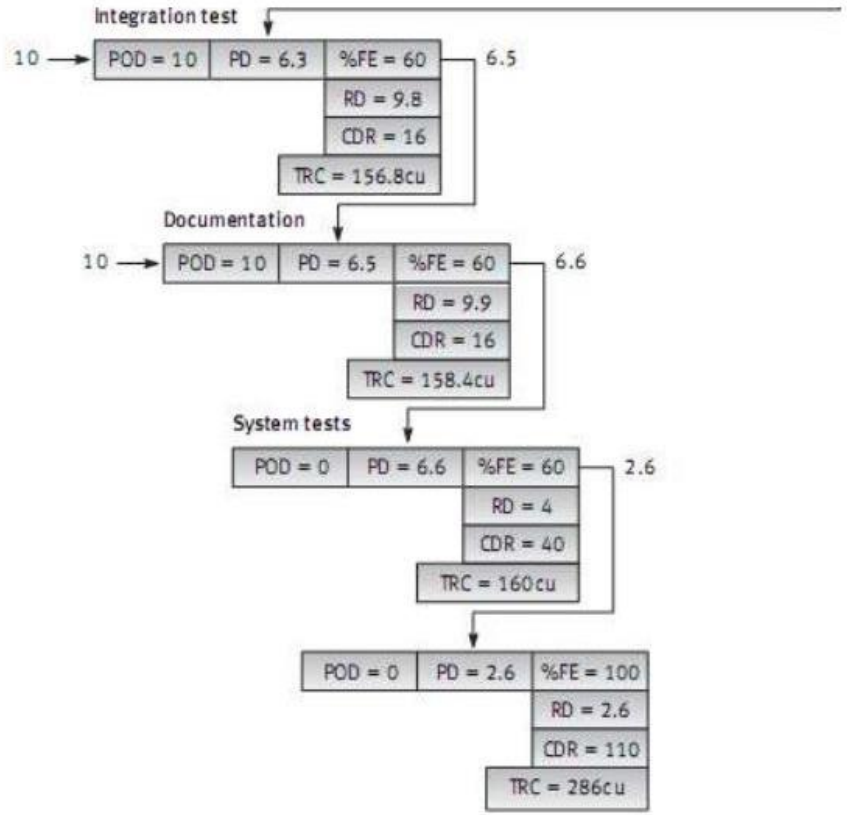
No.	Quality assurance activity	Defect-removal effectiveness	Cost of removing a detected defect (cost units)
1	Requirement specification review	60%	1
2	Design inspection	70%	2.5
3	Design review	60%	2.5
4	Code inspection	70%	6.5
5	Unit test – code	40%	6.5
6	Integration test	60%	16
7	Documentation review	60%	16
8	System test	60%	40
9	Operation phase	100%	110

EXAMPLE 2 COMPREHENSIVE QA PLAN



Continue next slide

EXAMPLE 2 COMPREHENSIVE QA PLAN



COMPARISON OF THE STANDARDS AND COMPREHENSIVE QA PLAN

Table 7.8: Comparison of the standard and comprehensive quality assurance plans

No.	Quality assurance activity	Standard plan		Comprehensive plan	
		Percentage of removed defects	Cost of removing defects (cost units)	Percentage of removed defects	Cost of removing defects (cost units)
1	Requirements specification review	7.5%	7.5	9%	9
2	Design inspection	–	–	28.7%	71.8
3	Design review	21.3%	53.2	7.4%	18.5
4	Code inspection	–	–	24.4%	158.6
5	Unit test – code	25.6%	166.4	4.2%	27.3
6	Integration test	17.8%	284.8	9.8%	156.8
7	Documentation review	13.9%	222.4	9.9%	158.4
8	System test	7.0%	280	4%	160
	Total for internal quality assurance activities	**93.1%**	**1014.3**	**97.4%**	**760.4**
	Defects detected during operation	6.9%	759	2.6%	286
	Total	**100.0%**	**1773.3**	**100.0%**	**1046.4**