Peer Inspections:

1. Transitioning engineering information from one artifact set to another. thereby assessing consistency, feasibility, understandability and technology constraints.
2. Major milestone demonstrations that force the artifacts to be assessed against tangible criteria in the context of relevant use cases.
3. Environment tools (compilers, debuggers ...)
4. Life-cycle testing for detailed insight
5. Change management metrics for objective insight into multiple-perspective change trends and convergence or divergence from quality and progress goals.

The old way and the new:

Principles of Conventional Software Engineering:

After years of software development experience, the software industry has learned many lessons and formulated many principles.

1. Make quality #1
2. High-quality software is possible
3. Give products to customers early.
4. Determine the problem before the writing of requirements.
5. Evaluate design alternatives.

6. Use an appropriate process model.
7. Use different languages for different phases.
8. Minimize intellectual distance.
9. Put techniques before tools.
10. Get it right before you make it faster.
11. Inspect code.
12. Good management is more important then good technology.
13. People are the key to Success.
14. Follow with care.
15. Take responsibility.
16. Understand the customer's priorities.
17. The more they see, the more they need.
18. Plan to throw one away.
19. Design for change
20. Design with out documentation is not design.
21. Use tools but be realistic.
22. Avoid tricks.
23. Encapsulate.
24. Use coupling & cohesion.
25. Use the McCabe complexity measure
26. Dont test Your own Software
27. Analyze causes for errors.
28. Realize that Software's entropy increases.
29. People and time are not interchangeable
30. Expect excellence.

The Principles of Modern Software Management:

1. Base the process on an architecture-first approach.
2. Establish an iterative life-cycle process that confronts risk early.
3. Transition design methods to emphasize component-based development.
4. Establish a change management environment.

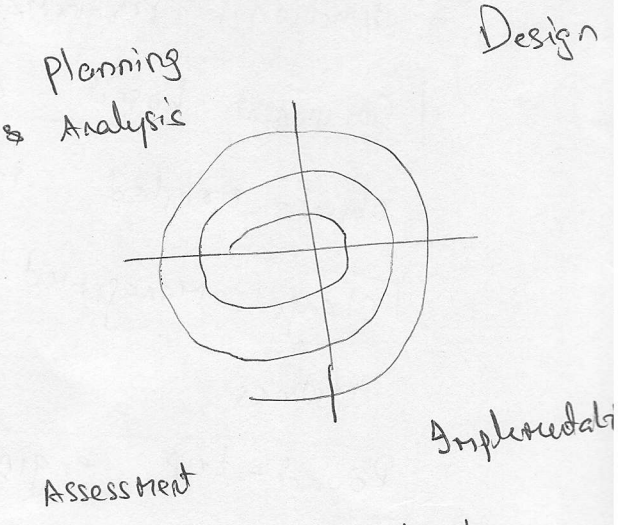
Waterfall Process

Requirements first
Custom development
Change avoidance
Ad hoc tools

Requirements analysis
↓
Design
↓
Code & Unit test
↓
Subsystem Integration
↓
System test

Iterative Process

Architecture first
Component-based development
change Management.
Round-trip engineering



5. Enhance change freedom through tools that support round-trip engineering.
6. Capture design artifacts in rigorous, model-based notation.

7. Instrument the process for objective quality control and progress assessment.
8. Use a demonstration-based approach to assess intermediate artifacts.
9. Plan intermediate releases in groups of usage scenarios with evolving levels of detail.
10. Establish a configurable process that is economically scalable.

Architecture-first approach → The central design element

Design, Integration first then production and test.

Iterative-life-cycle process → The risk-management process

Risk control through ever-increasing function, Performance, quality.

Component-based development → The technology element

object-oriented methods

Change-Management environment → The control element

Metrics.

Round-trip engineering → The automation element

Complementary tools.

figure: The top five principles of a modern process

Transitioning To an Iterative Process:

Modern Software development processes have moved away from the Conventional Waterfall model, in each Stage of the development process is dependent on Completion of previous Stage.

The following map the processes exponent parameters of COCOMO II

1. Application Precedentedness – Domain experience is a critical factor in understanding how to plan and execute a software development project.
2. Process flexibility:
3. Architecture risk resolution
4. Team cohesion
5. Software process maturity. The Software Engineering Institute's Capability Maturity Model (CMM) is a well-accepted benchmark for software process assessment. Just as domain experience is crucial for avoiding the application risk and exploiting the available domain assets and lessons learned, software process maturity is crucial for avoiding software development risks and exploiting the organization's software assets and lessons learned.

UNIT – III

Life-Cycle Phases:

Most Unsuccessful Projects one of following

1. An Overemphasis on research and development
2. An overemphasis on production.

A Modern Software development process must be defined to support:

a) Evolution of plans, requirements and architecture, together with well defined synchronization points.
b) Risk Management and objective measures of Progress and quality.
c) Evolution of system capabilities through demonstrations of increasing functionality.

Engineering and production stages:

1. The Engineering stage, driven by less predictable but smaller teams doing design and synthesis activities.
2. The Production stage, driven by more predictable but larger teams doing construction, test and deployment activities.

Inception phase: The overriding goal of the inception phase is to achieve Concurrence among stake holders on the life-cycle objectives for the Project.

Primary objectives:

1. Establishing the Project's Software Scope and boundary conditions.
2. Discriminating the critical use cases of the system
3. Demonstrating at least one candidate architecture
4. Estimating the cost and Schedule
5. Estimating potential risks.

Essential Activities:

1. Formulating the scope of the Project.
2. Synthesizing the architecture.
3. Planning and preparing a business case

Primary Evaluation Criteria:

→ Do all stake holders concur on the scope definition and cost and schedule estimates?

→ Are requirements understood, as evidenced by the fidelity of the critical use

→ Are the cost and schedule estimates, priorities, risk and development processes credible?

→ Do the depth and breadth of an architecture prototype demonstrate the preceding criteria?

→ Are actual resource expenditure Vs planned expenditures acceptable?

Elaboration phase:

Primary objectives:

1. Baselining the architecture
2. Baselining the Vision
3. Baselining a high-fidelity plan for construction phase
4. Demonstrating that the baseline architecture will support the vision at a reasonable cost. in a reasonable time.

Essential Activities:

> Elaborating the Vision.
> Elaborating the process and infrastructure
> Elaborating the architecture and selecting components.

Primary Evaluation Criteria:

→ Is the vision stable?
→ Is the architecture stable?
→ Does the executable demonstration show that major risk elements have been addressed and credibly resolved?
→ Is the construction phase plan of sufficient fidelity and is it backup with a credible basis of estimate?
→ Do all stake holders agree that the current Vision can be met if the current plan is executed to develop the

complete System in the context of the current architecture.

→ Are actual resource expenditures versus planned expenditures acceptable?

Construction Phase:

Primary Objectives:

1. Minimizing development costs by optimizing resources and avoiding unnecessary scrap and rework
2. Achieving adequate quality as rapidly as practical.
3. Achieving useful versions

Essential Activities:

> Resource Management, Control and process optimization
> Complete Component development and testing against evaluation criteria.
> Assessment of

Primary Evaluation criteria:

→ Is the product baseline mature enough to be deployed in the user community?
→ Is this product baseline stable enough to be deployed in the user community?
→ Are the stake holders ready for transition to the user community?
→ Are actual resource expenditures versus planned expenditures acceptable?