Conventional Software Management Theory and Practices.

* The best thing about software is its flexibility.
* It can be programmed to do almost anything.
* The worst thing is about software is also its flexibility.
* The "almost anything" has made it difficult to plan, monitor and control s/w development.

① * The success rate for software projects is very low. Only about 10% of s/w projects are delivered successfully within initial budget & schedule estimates.

② * "Mgmt discipline" is more of a discriminator in succes or failure than are technology advances.

③ * The level of s/w scrap & rework is indicative of an immature process.

* This analyses provides current norms for conventional software management performance.

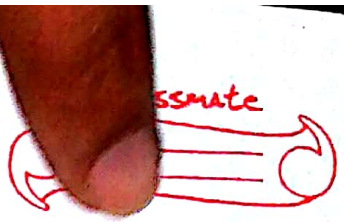
The Waterfall Model

→ source of conventional software process.

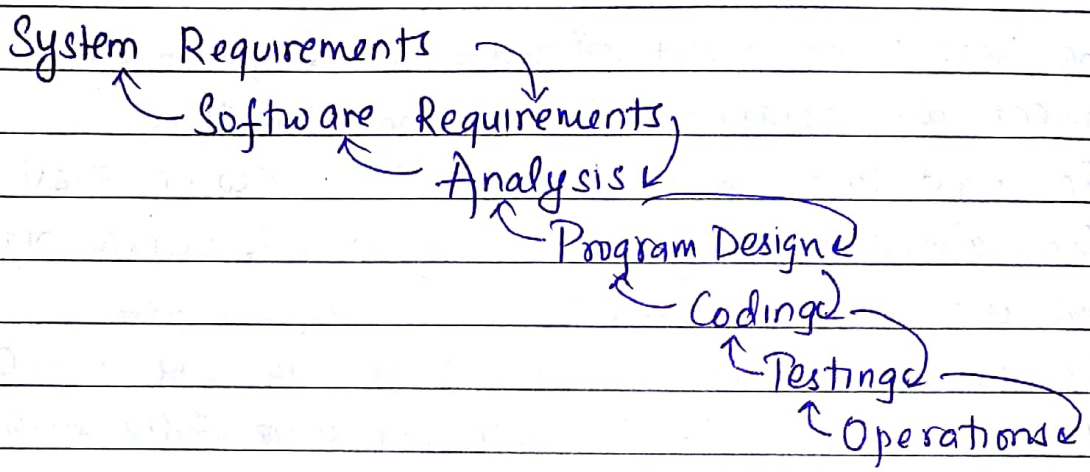
→ The two basic steps to building a program (WM part 1)

Analysis and Coding

They both involve creative work that directly contributes to the usefulness of the end product.



WM (Part 2)



WM Part 3:

Five necessary improvements for this approach to work.

1. Complete program design before analysis & coding design.
2. Maintain current & complete documentation.
3. Do job twice if possible.
4. Plan, control and monitor testing.
5. Involve the customer.

Improvement presented.

(1) Program design comes first.
(2) Document the design.
(3) Do it twice
(4) Plan, control & monitor testing
(5) Involve the customer.

In practice (Conventional).
If s/w is destined for trouble then it exhibits following symptoms.
① Protractive integration & late design breakage
② Late Risk Resolution
③ Requirement driven functional decomposition
④ Adversarial Stakeholder Rel^nships
⑤ Focus on document & review meeting.

classmate
Date
Page

In Practice :-

→ Despite the advice of many software experts and theory behind the waterfall model, some software projects still practice the conventional software management approach.

→ The projects destined for trouble frequently, if conventional s/w management was applied, exhibits the following symptoms:

① Protracted integration & late design breakage.
② Late risk resolution.
③ Requirements - driven functional decomposition
④ Adversarial stakeholder relationships
⑤ Focus on documents reviews & meeting
~~⑥~~

① Protracted integration and Late Design Breakage.

→ In the conventional ~~waterf~~ project development on waterfall model the following sequence was common,

ⓐ early succes via paper design and thorough briefings.
ⓑ Commitment to code late in the life cycle.
ⓒ integration nightmares due to unforeseen implementation issue and interface ambiguities
ⓓ heavy budget and schedule pressures to get the system working.
ⓔ a very fragile, unmaintainable product delivered late.

→ Conventional techniques that imposed a waterfall model on the design procen inevitably resulted in late integration and performance showstoppers.

→ In conventional model, the entire system was designed on paper, then implemented all at once, then integrated.

⇒ Only at the end of procen was it possible to perform system testing to verify fundamental architecture was sound.

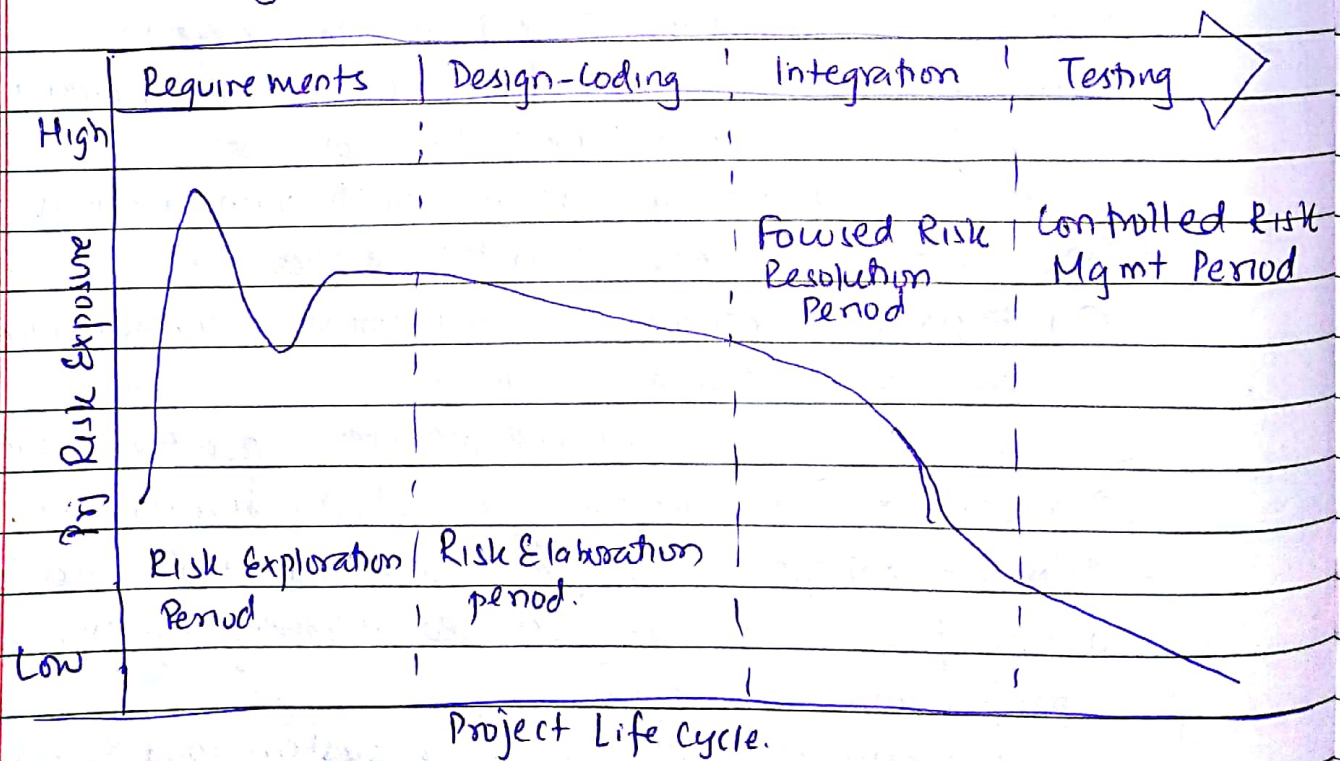
→ Testing activity consumes 40% or more of life cycle resources

② Late Risk Resolution.

→ Serious issue with waterfall life cycle was lack of early risk resolution.

→ This was not so much a result of the waterfall life cycle as it was of focus on early paper artifacts, in which real design, implementation, and integration risks were still relatively intangible.

→ Requirement → Risk Exposure very high.
Design & Coding → Risk Exposure less in comparison to requirement.
Integration → Focused Risk Resolution Period and less in comparison.
Testing → Controlled Risk Mgmt period & less risk in comparison



③ Requirement-Driven Functional Decomposition.

→ Traditionally the s/w development process has been requirements-driven: A process/attempt is made to provide a precise requirements & implement those requirement.

→ It totally depends upon specifying requirements completely & unambiguously b4 other activities begin.

→ It treats all reqmt equally.

→ However, in real world these condition rarely occur.

→ Also equal treatment of reqmt drains substantial no. of engineering hours from driving reqmt & wastes those effort on paperwork, which is inevitably discarded later.

→ Another property, ~~is that~~ of the conventional approach is that the requirements were typically specified in a functional manner.

→ Built into the classic waterfall process was fundamel assumption that s/w itself was decomposed into funcn & reqmt were allocated to resulting components.

→ The functional decomposition ~~was~~ also became anchored in contracts, subcontracts, WBS often precluding an architecture driven approach.

④ Adversarial Stakeholder Relationships.

→ Conventional process tended to result in adversarial stakeholder relationships in large part. coz of (difficulties of requirement specification) & (exchange of info solely through paper documents).

→ The following sequence of events was typical for most contractual software efforts:

① Contractor prepared a draft contract - deliverable document that captured an intermediate artifacts & delivered to customer for review.

(ii) Customer was expected to provide comments
(iii) Contractor incorporated these comments & submitted a final version for approval.

→ The overhead of such paper exchange review process was intolerable.

→ This approach also resulted in customer-contractor relationship deteriorating into mutual distrust making it difficult to achieve a balance among requirements, schedule & cost.

(5) Focus on Documents & Review Meetings.

→ Conventional process focused on ~~finding~~ producing various documents that attempted to describe the software product, with insufficient focus on producing tangible increment of product themselves.

→ Major milestones were usually implemented as meetings defined solely in term of specific documents.

→ Contractors were driven to produce literally tons of paper to meet milestone & demonstrate progress to stakeholders rather than spending energy to reduce risk & produce quality s/w.

→ Most design therefore resulted in low engineering value & high cost in terms of effort & schedule involved in their preparation & conduct.

1.2 Software Economics and Cost Estimation.

→ Most s/w cost models can be abstracted into a function of 5 basic parameters.

ⓐ Size ⓑ Process ⓒ Personnel ⓓ Environment ⓔ Reqd Quality.

ⓐ Size:-

- Size of end product, which is typically quantified in terms of no. of source instructions or no. of func^{n} points reqd to develop required functionality

ⓑ Process

- Process used to produce the end product, in particular the ability of procen to avoid non-value adding activities (rework, bureaucratic delays, communication overhead).

ⓒ Personnel.

- Capabilities of s/w engineering personnel and particularly, the experience with the computer science issues and the applications domain issues of prj

ⓓ Environment.

- It is made up of tools & techniques to support efficient s/w development and to automate the procen.

ⓔ Quality:

- Reqd quality of product including its features, performance, reliability and adaptability

Relationship among these 5 parameters.

Effort = (Personnel)(Environment)(Quality)(Size Process)

→ One important aspect of software economics (as represented with today's s/w cost models) is that the relationship bet^n effort and size exhibits a diseconomy of scale.

→ This diseconomy of scale of software development is a result of the process exponent being greater than 1.0.

There are 3 generations of software development

(1) Conventional:
- 1960's and 1970's
- Organizations used custom tools, custom process and virtually all custom components built in primitive languages.
- Project performance was highly predictable in that cost, schedule, quality objectives.

(2) ~~Tradit~~ Transition.
- 1980s and 1990s
- Organization used more repeatable process, off the shelf tools & mostly (>70%) custom components built in higher-level langs.
- Some of the components (<30%) were available as commercial product including the OS, dBMS, n/w ing etc

(3) Modern Practices:
- 2000
- Organization used managed & measured process, integrated

automation environments, and mostly (70%.) off the shelf components and 30%. of components are custom built.

Conventional	Transition	Modern
Target objective: Improved ROI (Return of Improvement)		
1960s-1970s	1980s-1990s	2000s and soon
Waterfall Model	Process Improvement	Iterative Development
Functional Design	Encapsulation-based	Component Based
Diseconomy of scale	Diseconomy of scale	Return of Investment
Corresponding environment, size and process technologies		
Environment/tools:		
Custom	Off-the shelf, separate	Off-the-shelf, integrated.
Size:		
100% custom	30% component-based. 70% custom	70% component based 30% custom
Process		
Ad-hoc	Repeatable	Managed/Measured
Typical project Performance:		
Predictably bad	Unpredictable	Predictable
Always:	Infrequently:	Usually
Over budget	On budget	On budget
Over schedule	On schedule	On schedule

- In all era, the complementary growth is the main key in all technologies.
- The transition to modern procen and the promise of improved s/w economics are by no means guranteed.
- Organizations are achieving better economies of scale in successive technology eras - with very large projects, long-lived products & lines of busines comprising multiple similar projects.

Software Cost Estimation.

→ One critical problem in s/w cost estimation is a lack of well-documented case studies of projects that used an interative development approach.

→ Furthermore, because ~~of~~ the s/w industry has inconsistenly defined metrics or atomic units of measure, the data from actual prj are highly suspect in terms of consistency and comparability.

→ ~~The fromo~~ It is hard enough to collect a homogeneous set of prj data within one organization; it is extremely difficult to homogenize data across different organization:

→ For.eg:- the fundamental unit of size (a source LOC and funcn pt) can be and is counted differently across the ~~country~~ industry.

→ There have been ~~too~~ many long-standing debates among developers and vendors of s/w cost estimation models & tools.
  - which cost estimation model to use.
  - whether to measure s/w size in source lines of code or funcnpt.
  - what ~~contin~~ constitutes a good estimate.

(I) There are several popular cost estimation models like COCOMO, CHECKPOINT, ESTIMACS, Knowledge Plan etc as well as numerous organization-specific models.
Among which COCOMO is and its successors is basis of many of my s/w economics arguments and perspectives.
COCOMO is also one of the most open & well documented cost estimation models.
The evolution of COCOMO into its current version, COCOMO II provides an interesting historical perspective over the years of its evolution.

(II) The measurement of s/w size has been the subject of much rhetoric.
There are basically two objective pt of view: (a) source ~~code~~ lines of code and (b) function points. Both perspectives have proven to be more valuable than a third which is subjective or ad-hoc pt of view.

→ (i) Many s/w experts have argued with SLOC is a lousy measure of size. However, when a code segment is described as 1000 source line program, most ppl feel ~~fee~~ comfortable.

(ii) If description were 20 func^n pt, 6 classes, 5 use case, 4 obj pt. then most ppl would ask further question to gain an understanding of the subject code. So SLOC is one measure that still has some value

(iii) Today, language advance and use of components, automatic source code generation and object orientation have made SLOC an ambiguous measure.

→ (a) The use of func^n point has a large following however. The primary advantage of using func^n pt is that this method is independent of technology & therefore a much better primitive unit for comparison among prj and organizations.

S/W Cost Estimation

long standing debate b/w developer & vendors on

① What cost model to use?
COCOMO. ESTIMACS. CHECKPOINT.

② Whether to measure size of s/w in LOC or func^n pt?
→ If small LOC, otherwise func^n pt
Func^n pt is independent of tech so a better option for comparison among organization for cross proj - use func^n pt.

③ What constitutes a good estimate?
→ cost model are bottom up approach. defines target cost then easy proj manager/cost manager to iterate that target. proj manager/cost manager to iterate through several estimates sensitivity analysis

classmate
Date
Page

(ii) The main disadvantage is that primitive definition are abstract and measurement are not easily derived directly from the evolving artifacts.

→ Although both measures have their drawbacks. An organization can make either one work.
Anyone doing a cross proj work should be using func^n pt as a measure of size. The Func^n pt are also accurate estimator in early phases of proj life cycle.
In later phases however SLOC becomes a more useful metric.

(iii)

- The general accuracy of conventional cost models (like COCOMO) has been described as "within 20% of actuals, 70% of the time".
- This level of unpredictability in the conventional s/w devpment process should be truly frightening to every investor, especially in light of fact that few proj miss their estimate by doing better than expected.
- Most cost models is bottom-up rather than top-down. The s/w project manager defines the target cost of the software, then manipulates the parameter and sizing & until the target cost can be justified.
- In The process, it is absolutely necessary to analyze the cost risks & understand the sensitivities & trade-offs objectively which forces s/w expert to examine risks associated & discuss with other stakeholders. This provides a good vehicle for basis of estimates & an overall cost analysis.
- A practical lesson learned from the field

cost estimates done by ppl who are independent of development team are usually inaccurate.
The only way to produce a credible estimate is that project manager & the s/w architecture, test manager to iterate through several estimates & sensitivity analysis. The team must take the ownership of that cost estimate for prj to succeed.

1.3 Improving Software Economics.

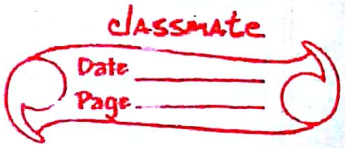
→ Improvement in the economics of software development have been not only difficult to achieve but also difficult to measure and substantiate.

→ The presentation of important dimension around the 5 basic parameters of s/w cost model.

1. Reducing the size or complexity of what needs to be developed.
2. Improving development process.
3. Using more-skilled personnel and better teams.
4. Using better environments.
5. Trading off or backing off on quality thresholds

→ The important trends in improving software economics:-

Cost Model Parameter	Trends
Size.	
Abstractions component based development technologies	→ High order languages (C++, Java etc) → Object-oriented (analysis, design) → Reuse → Commercial components.



Process

Methods & Techniques → Iterative Development
→ Process Maturity Models
→ Architecture-first development
→ Acquisition reform.

Personnel

People factors → Training & personnel skill development
→ Team work
→ Win-win cultures

Environment

Automation technologies & tools → Integrated tools (compiler, editor etc)
→ Open systems
→ Hardware platform performance
→ Automation of coding, documents, testing

Quality

Performance, Reliability, Adaptability → Hardware platform performance
→ Demonstration-based assesment
→ Statistical quality control.

(1) Reducing Software Product size.

→ Component based development is introduced as general term for "reducing the source" language size necessary to achieve a software solution.

→ It is also a primary motivation behind improvements in higher order languages (C++, Java, Ada 95 etc), object oriented technologies (Visual Modeling tools, architecture frameworks etc)

(i) Languages.

- Universal Func" pt are useful estimators for language-independent early life cycle estimates.
- The basic units of func" pt are external user inputs, external output, internal logic data groups etc.
- SLOC metrics are useful estimators for s/w after a candidate solution is formulated & an implementation language is known.
- The modern languages such as C++, Ada 95, Java and Visual Basic: The level of expressibility is very attractive, but care must be taken coz it has its misses.
- However, each lnguage has its own use. like Visual Basic for building simple interactive applⁿ, Ada 95 for catastrophic, cost-of-failure system etc.
- Universal Func" pt can be used to indicate the relative program sizes reqd to implement a given functionality.

(ii) ReUse.

- ~~It has the following~~ Most reusable components of value are transitioned to commercial products with following ~~organizations~~ characteristics:-
  - they have an economic motivation for continued support.

(b) They take ownership of improving product quality, adding new features & transitioning to new technologies

(c) They have a sufficiently broad customer base to be profitable.

→ Reusing existing components & building reusable components have been natural s/w engineering activities.

→ Reuse achieves undeserved importance within the s/w engineering community only coz we don't do it as well as we should.

→ One of the biggest disadvantage to reuse has been fragmentation of languages, OS, notations etc.

(iii) Commercial Components.

→ A common approach today is to maximize integration of commercial components and off-the-shelf products.

→ While ~~of~~ the use of commercial components is certainly desirable ∴ it has not proven to be straightforward.

→ There are certain advantages of using s/w's commercial components:-

(i) Predictable license costs
(ii) Available now.
(iii) Rich in functionality
(iv) Dedicated support organization
(v) Broadly used, mature technology.

→ At the same time, there are disadvantages also.

(i) Frequent upgrades
(ii) Up-front license fees
(iii) Recurring maintenance fees.
(iv) Dependency on vendor.
(v) Functionality constraints
(vi) Run-time efficiency sacrifices.

(2) Improving Software Process :-

→ Process is an overloaded term.

→ For ~~object~~ s/w oriented organizations, there are many processes and subprocesses.

→ There are 3 distinct process perspectives.

(i) Metaprocess :- an organization's policies, procedures and practices for pursuing a software intensive line of business. The focus of this process is on organizational & economics, long-term strategies, and a slw ROI.

(ii) Macroprocess: a ~~process~~ project's policies, procedures and practices for producing a complete slw product within certain cost, schedule & quality constraints. The focus of this process is on creating an adequate instance of metaprocess for a specific set of constraints.

(iii) Microprocess: a project's team's policies, procedures & practices for achieving an artifact of the slw process. The focus of this process is on achieving an intermediate product baseline with adequate quality and adequate functionality as economically ~~and rapidly~~.

→ Among these, the macroprocess is the project-level process that effects the cost-model.

→ All prj processes consist of productive activities and overhead activities.

→ for slw efforts, these activities include modeling, coding, debugging etc.

→ The objective of process improvement is to maximize the

④ Reducing product size.
3 pts in consideration
① Language.
- Universal Func^n pt estimators for language-independent early life cycle estimate
② Reuse
③ Commercial Components
⑤ Improving S/w process
→ Metaprocess, Macroprocess, Microprocess.
⑥ Improving Team Effectiveness
① Principle of job matching, top talent, team balance, careers progression, phaseout

allocation of resources to productive activities and minimize the impact of overhead activities on resources such as personnel, computers & schedule.
- The quality of software process strongly affects reqd effort and therefore the schedule for producing s/w product.

(3) Improving Team Effectiveness.
→ It has been long seen that differences in personnel account for the greatest swings in productivity.
→ Balance & coverage are two of the most important aspects of excellent teams.
→ Whenever a team is out of balance, it is vulnerable. Teamwork is is very important than sum of individuals.
→ Some maxims of team mgmt include:
(i) A well managed project can succeed with nominal engineering team.
(ii) A mismanaged prj will almost never succeed, even with an expert team of engineers.
(iii) A well-architected system can be built by a nominal team of s/w builders
(iv) A poorly architected system will flounder even with an expert team of builders.
→ In examining how to staff a s/w project, Boehm offered the following staffing principles
(i) The principle of top talent: Use better & fewer ppl.
(ii) The principle of job matching: Fit tasks to the skills and motivation of the ppl available.
(iii) The principle of career progression: An organization does best in

long run by helping ppl to self actualize

iv) The principle of team balance: Select ppl who will complement and harmonize with one another.

v) The principle of phase out: Keeping a misfit on the team doesn't benefit anyone.

→ The skills that a project manager must possess are: decision-making skill, team-building skill, selling-skill, customer-interface skill and hiring skill.

(4) Improving Automation through S/w environment

→ Tools & Environment used in s/w process generally have a linear effects on productivity of process.

→ Configuration mgmt envt provide foundation for executing & instrumenting the process.

→ Tools & envt must be viewed as the primary delivery vehicle for process automation & improvement. So impact can be higher.

→ One of the primary contribution of the envt is: to automate manual tasks that are inefficient or error-prone.

→ The transition to a mature s/w process introduces new challenges and opportunities for mgmt control of concurrent activities & for tangible process & quality assessment.

→ An envt that provides semantic integration and process automation can improve productivity, improve s/w quality, and accelerate the adoption of modern techniques.

→ A robust, integrated development envt must support the automation of the development process.

→ Round trip engineering is used to describe the key capability of envts that support iterative development.

Round trip engineering describes the envt support needed to change an artifact freely & have other artifacts automatically changed so that consistency is maintained among the entire set of reqmt, design, implementation etc.

Forward Engineering is automation of one engineering artifact from another, more abstract representation.

Reverse engineering is the generation or modification of a more abstract representation from an existing artifact.

(5) Achieving Required Quality

→ Many of what are accepted today as s/w best practices are derived from the development procen and technologies.

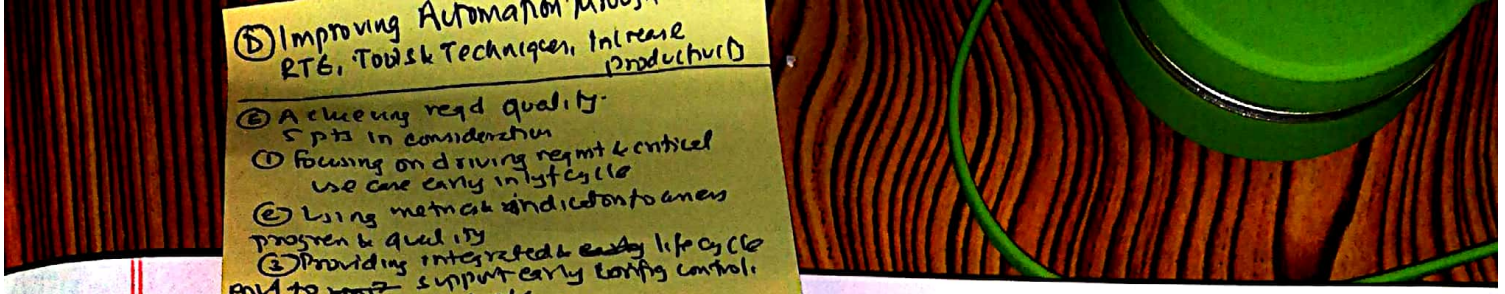
→ Key practices that improve overall s/w quality include the following:

@ Focusing on driving requirements and critical use cases early in the life cycle, focusing on reqmt completeness and traceability late in the life cycle & focusing throughout the life cycle on a balance between reqmt evolution, design evolution & plan evolution.

(b) Using metrics and indicators to measure the progress and quality of an architecture as it evolves from high-level prototype into a fully compliant product.

(c) Providing integrated life-cycle envt that support early and continuous configuration control, change mgmt, rigorous design methods, document automation & regression test automation.

(d) Use visual modeling and high-level languages that support



classmate
Date
Page

architectural control, abstraction, reliable programming, reuse & self-documentation,

(e) Early & continuous insight into performance issues through demonstration based evaluation.

1.6 Principles of Conventional Software Engineering.

→ In the s/w industry, David's top 30 principles are stated below:- (10 are stated below)

(1) Make Quality.

→ Quality must be quantified and mechanisms put into place to motivate its achievement.

(2) High-Quality Software is possible.

→ Techniques that have been demonstrated to increase quality include involving the customer, prototyping, simplifying design, conducting inspections, and hiring the best people.

(3) Give products to customers early.

→ No matter how hard you try to learn users needs during requirements phase, the most effective way to determine real needs is to give users a product & let them play with it.

(4) Determine the problem before writing the requirements

When faced with what they believe is a problem, most engineers rush to offer a solution. Before you try to solve a problem, be sure to explore all the alternatives & don't be blinded by the obvious soln.

(5) Evaluate design alternatives:

→ After the requirements are agreed upon, you must examine a variety of architectures and algorithms. You certainly do not want to use an "architecture" simply bcoz it was used in the requirements specification.

(6) Use an appropriate proven model:

→ Each project must select a proven that makes sense for that prj on the basis of corporate culture; willingness to take risks, application area, volatility of requirements & the extent to which requirements are well understood.

(7) Minimize intellectual distance

→ To minimize intellectual distance, the software's structure should be as close as possible to the real-world structure

(8) Get it right b4 you make it faster.

→ It is far easier to make a working program run faster than that it is to make a fast program work. Don't worry about optimizing during initial coding.

(9) Inspect code:

Inspecting detailed design and code is a much better way to find errors and testing.

(10) Good management is more important than good technology.

The best technology will not compensate for poor mgmt, and a good manager can produce great results even with

meager resources. Good management motivates people to do their best. but there are no universal "right" styles of management.

(11) People are the key to success.
Highly skilled people with appropriate experience, talent and training are key. The right ppl with insufficient tools, languages and procen will succeed. The wrong people with appropriate tools, languages & procen will probably fail.

(12) Understand the customer's priorities.
It is possible the customer would tolerate 90% of functionality delivered late if they could have 10% of it on time.

1.7 Principles of Modern Software Management.

→ The 10 principles of modern slw mgmt are as follows.

(1) Base the process on an architecture-first approach.
→ This requires that a demonstrable balance be achieved among the driving requirements, the architecturally significant design decisions etc. b4 the resources are committed for full scale development.

(2) Establish an iterative life-cycle procen that confronts risk early.
→ An iterative procen that refines the problem understanding, an effective soln & an effective plan over several iterations encourages a balanced treatment of all stakeholder objectives.

(3) Transition design methods to emphasize component-based development
→ Moving from line of code mentality to a component based mentality is necessary to reduce the amount of human-generated source code & custom development.

(4) Establish a change mgmt environment
→ The dynamics of iterative development, including concurrent workflows by different teams working on shared artifacts, necessaries objectively controlled baselines

(5) Enhance change freedom through tools that support round-trip engineering
→ RTE is the envt support necessary to automate & synchronize engineering info in diff formats. Without substantial automation of the s bookkeeping, documentation, testing, etc, it is difficult to reduce iteration cycles to manageable time frames in which change is encouraged rather than avoided.

(6) Capture design artifacts in rigorous, model-based notation
→ A model based approach (UML) supports the evolution of semantically rich graphical & textual design notation. Visual modeling with rigorous notation & a formal m/c procenable dng provides for far more objective measures than traditional approach.

(7) Instrument the procen for objective quality control & procen anenment.
→ Life cycle anessment of progren and quality of all intermediate products must be integrated into the procen. The best anessment mechanisms are well-defined measures derived directly from evolving engineering artifacts & integrated into all actnines & teams.

(8) Use a demonstration-based approach to assess intermediate artifacts.

→ Transitioning the current state of the product artifacts into an executable demonstration of relevant scenarios stimulates earlier convergence on integration, a more tangible understanding of design trade offs & earlier elimination of architectural defects.

(9) Plan intermediate releases in groups of usage scenarios with evolving levels of detail.

→ It is essential that the s/w mgmt process drive toward early and continuous demonstration within the operational context of the system, namely its use cases. The evolution of prj increments and generations must be commensurate with the current level of understanding of the requirements & architecture.

(10) Establish a configurable process that is economically scalable.

→ No single process is suitable for all s/w developments. A pragmatic process framework must be configurable to a broad spectrum of applications. The process must ensure that there is economy of scale and return of investment by exploiting a common process spirit, extensive process automation, and common architecture, pattern and components.

1.8 Iterative Process

→ Modern approaches generally require that an initial version of the system be rapidly constructed early in the development process, with an emphasis on addressing the high-risk areas, stabilizing the basic architecture & refining the driving reqmts.

→ An iterative process emphasizes the whole system rather than the individual parts.

→ Risk is reduced early in the life-cycle through continuous integration and refinement of reqmt, architecture & plans.

→ The economic benefits inherent in transitioning from the conventional waterfall model to an iterative development process are significant but difficult to quantify.

→ The following points map the process to principles of a modern process:

(1) Application precedentedness.

→ one of the primary reasons that s/w industry has moved to iterative life-cycle process.

- Early iterations in the life cycle establish precedents from which the product, the process, and the plans can be elaborated in evolving levels of detail.

(2) Process flexibility:

→ Project artifacts must be supported by efficient change mgmt commensurate with prj needs.

→ A configurable process that allows a common framework to be adapted across a range of prj is necessary to achieve a s/w return of investment.

(3) Architecture risk resolution.

→ Architecture-first development is a crucial theme underlying a successful iterative development process.

→ A prj team develops and stabilizes an architecture by developing all the components that make up entire suite of appln components.

→ · An architecture first & component based approach forces the infrastructure common & control mechanisms to be elaborated early in life cycle & drives all component make decisions in architecture procen.

(4) Team cohesion

→ Successful teams are cohesive, & cohesive teams are succenful.

→ Cohesive teams avoid sources of prj turbulences and entropy that may result from difficulties in synchronizing prj stakeholder expectations

→ One of the reason for turbulence is miscommunication.

→ These model-based formats have also enabled the round-trip engineering support needed to establish change freedom sufficient for evolving design representations.

(5) Software procen maturity:

→ The CMM is a well-accepted benchmark for s/w procen assessment.

→ · A software procen maturity is crucial for avoiding s/w development risks & exploiting the organizations s/w assets & lessons learned.

→ A truly mature procen are enabled through an integrated environment that provides the appropriate level of automation to instrument the procen for objective quality control.

Software Process Primitives and Procen Mgmt framework

(2.1) Software Process Life-Cycle Phases

→ Succenful modern projects and even succenful projects developed under the conventional process tend to have a very well-defined project milestone

→ Earlier phases focus on achieving functionality.

→ Later phases focus on achieving product that can be shipped to a customer.

→ A modern software development procen must be defined to support the following:

(a) Evolution of plan, requirement & architecture together with well defined synchronization points

(b) Evolution of system capabilities through demonstration of increasing functionality

(2.2) Engineering and Production Stages:-

There are two stages in the life cycle of software process:

(i) Engineering stage: Driven by less predictable but smaller teams doing design & synthesis activities

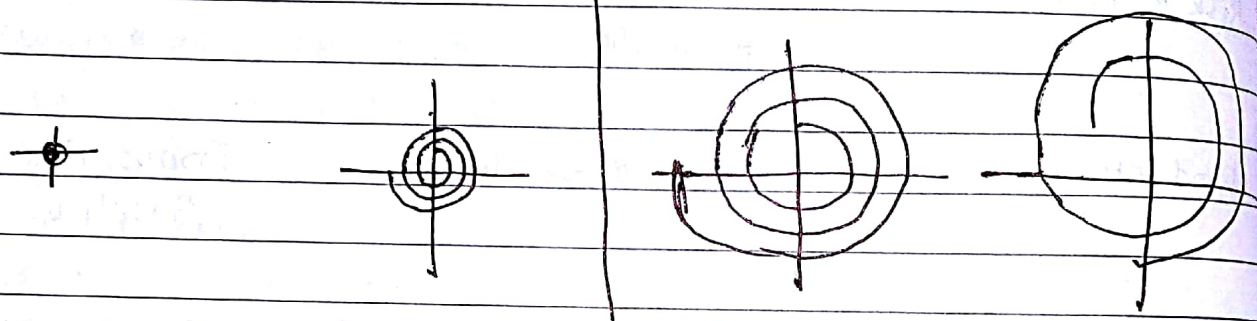
(ii) Production stage:- Driven by more predictable but larger teams doing construction, test & deployment activities.

The two stages of life cycle: engineering & production

Life Cycle Aspect	Engineering Stage Emphasis	Production Stage Emphasis
Risk Reduction	Schedule, technical feasibility	Cost
Products	Architecture Baseline	Product Release Baseline
Activities	Analysis, Design, Planning	Implementation, Testing
Assessment	Demonstration, inspection, analysis	Testing.
Economics	Resolving diseconomies of scale	Exploiting economies of scale
Management	Planning	Operations

The engineering stages is decomposed into two distinct phases. inception and elaboration and production stage.
These four phases of the life-cycle process are loosely mapped to conceptual framework of spiral model.

Engineering Stage		Production Stage	
Inception	Elaboration	Construction	Transition



Idea	Architecture	Beta Releases	Products

1) Inception Phase:-

Primary Objectives:-

→ Establishing project's scope and boundary conditions, including operational concept, acceptance criteria and clear indication of what is & not intended in product.

→ Discriminating critical use cases of system and primary scenarios of operation that will drive major trade offs.

→ Demonstrating at least one candidate architecture

→ Estimating cost and schedule of the project

→ Estimating potential risks

Essential Activities

→ Formulating scope of the project. This activity involves capturing the reqmt and operational concept in an info repository that describes the user's view of the reqmt. This info repository should be sufficient to define problem space & derive acceptance criteria for end product.

2) Synthesizing the architecture: Design trade-offs, problem space ambiguities and available solution-space assets are evaluated. An information repository is created that is sufficient to demonstrate the feasibility of at least one candidate architecture and initial baseline of make/buy decisions so that the cost, schedule & resource estimates can be derived.

3) Planning & preparing a business case. Alternatives of risk mgmt, staffing, iteration plans and cost/schedule/profitability trade-offs are evaluated. The infrastructure sufficient to support the life-cycle development task is determined.

Primary Evaluating Criteria

→ Do all stakeholders concur on the scope definition and cost ~~definition and co~~ and schedule estimates?

→ Are reqmt understood, as evidenced by fidelity of critical use cases?

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

→ Are actual resource expenditures versus planned expenditures acceptable?

2) Elaboration Phase (most critical phase)

~~Primary~~ - At the end of this phase, engineering is considered complete and project faces its reckoning.

Primary Objectives:

- Baselining the architecture, as rapidly as practical.
- Baselining the vision
- Baselining the high-fidelity plan for the construction phase

- Demonstrating that the baseline architecture will support the vision at a resonable cost in reasonable time.

Essential Activities

→ Elaborating the vision. This activity involves establishing a high-fidelity understanding of the critical use cases that drive architectural or planning decisions.

- Elaborating the process & infrastructure. The construction process, the tools and process automation support & intermediate milestone and their respective evaluation criteria are established.

- Elaborating the architecture and selecting ~~architecture~~ components. Potential components are evaluated & make/buy decisions are sufficiently understood so that construction phase cost and schedule can be determined with confidence.

Primary Evaluation Criteria.

→ Is the vision stable?
→ Is the architecture stable?
→ Does the executable demonstration show that the major risk elements have been addressed & credibly resolved?
→ Is the construction phase plan of sufficient fidelity & it is backed up with a credible basis of estimate?
→ Are actual resource expenditures versus planned expenditures ~~are~~ acceptable?

Construction Phase:-

- All remaining components and application features are integrated into the application and all features are thoroughly tested.
- Newly developed software is integrated where required.

Primary Objectives:-

→ Minimizing development costs by optimizing resources and avoiding unnecessary scrap and network
→ Achieving adequate quality as rapidly as practical.
→ Achieving useful versions as rapidly as practical.

Essential Activities

→ Resource mgmt, control and proces optimization
→ Complete component development & testing against evaluation criteria
→ Assessment of product releases against acceptance criteria of the vision.

Primary Evaluation Criteria

→ Is this 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 stakeholders ready for transition ~~rea~~ to user community?
→ Are actual resource expenditures versus planned expenditures acceptable?

Transition phase

→ This phase is entered when a baseline is mature enough to be deployed in the end-user domain.

→ This phase concludes when the deployment baseline has achieved the complete vision.

Primary Objectives

1) Achieving user self-supportability
2) Achieving stakeholder concurrence that deployment baselines are complete and consistent with the evaluate criteria of the vision
3) Achieving final product baselines as rapidly and cost-effectively as practical

Essential Activities:-

- Beta testing to validate the new system against user expectations
- Beta testing and parallel operation relative to a legacy system it is replacing.
- Conversion of operational db
- Training of users & maintainers.

Evaluation Criteria

- Is this user satisfied?
- Are actual resource expenditures versus planned expenditures acceptable?

Artifact of the Process.

→ To make the development of a complete s/w system manageable, distinct collections of information are organized into artifact sets.

→ While a set represents a complete aspect of system, an artifact represents cohesive info that typically is developed and reviewed as a single entity.

→ Life cycle software artifacts are organized into 5 distinct sets that are roughly partitioned by underlying language of set:

- management
- requirement
- design
- implementation
- deployment.

Reqmt set	Design set	Implementation set	Deployment set
1) Vision Document	1) Design model	1) Source code baseline	1) 'Associated run-time files
2) Reqmt Model	2) Test model	2) Associated compile time files	2) User Manual
	3) S/w Architecture description	3) Component + Executables	

Mgmt set

Planning Artifacts	Operational Artifacts
1) WBS (Work breakdown Structure)	5) Release Description
2) Busines case	6) Status Assessment
3) Release Specification	7) S/w change order db
4) S/w development plan	8) Deployment docms
	9) Envt:

The Management Set.

→ This captures the artifacts associated with process planning and execution.

→ Special artifacts included in this set are the

(1) WBS (activity breakdown and financial tracking mechanism)
(2) Business case (Cost, schedule, profit expectations)
(3) Release specification (scope, plan, objective for release baselines)
(4) Release descriptions (result of release baselines)
(5) Status assessment (periodic snapshot of progress)
(6) S/w development plan (project process instance)
(7) environment (h/w & s/w tools, process automation, documentation)
(8) deployment documents (cutover plan, training course)
(9) s/w change orders (description of discrete baseline changes)

→ This set includes relevant stakeholder reviews, analysis of changes bet^n current & previous version of artifact etc.

The Engineering set.

→ It consist of reqmt set, design set, implementation set, & deployment set.

→ Reqmt set.

(a) Analysis of consistency with release specification of mgmt set
(b) Anaanalysis of consistency bet^n vision & reqmt models
(c) Analysis of changes bet^n current & previous version of reqmt artifact
(d) Subjective review of other dimension of quality.

Sets are human-readable format that are evaluated, assessed & measured through combination of following:

Design set

→ UML notation is used to engineer design models.
→ Analysis of consistency with reqmt models
→ Analysis of internal consistency & quality of design model
→ Analysis of changes between current & previous version of design model
→ Subjective review of other dimension of quality.

Implementation set.

→ This includes source code that represents implementation of components & any executables necessary for stand alone testing of components
→ Analysis of consistency with design model
→ Translation of deployment set notations to evaluate consistency and completeness.
→ Assessment of component source or executable files against relevant evaluation criteria.
→ Analysis of changes between current & previous version of implementation model.
→ Subjective review of other dimension of quality.

Deployment Set.

→ This includes user deliverables & m/c language notations, executable s/w, executable target specific data necessary to use product in target envt.
→ Testing against usage scenarios & quality attributes defined in reqmt set to evaluate consistency & completeness.
→ Testing partitioning, replication & allocation strategies in mapping components of implementation set to physical resources of deployment system

→ Analysis of changes between the current and previous versions of deployment
→ Subjective review of other dimension of quality.

Software Procen Various Elements.

① Management Artifacts.
→ This artifact includes the following artifacts:-

a) Businen Case
→ It provides all the info necesary to determine whether the prj is worth investing in.
→ It details expected revenue, expected cost, technical & mgmt plans etc.
→ In large scale endeavor, the businen case may be implemented in a full-scale proposal with multiple volumes of info.
→ In small scale endeavor, it may be implemented in a brief plan with an attached spreadsheet.
→ Main purpose is to transform the vision into economic terms so that organization can make an accurate ROI anenment.

b) S/W Development Plan
→ It elaborates the procen framework into fully detailed plan.
→ It is the defining document for project's procen
→ It must comply with (contract & organization standards), evolve along with design & reqmt & be used consistently across all subordinate organizations doing s/w development

→ Two indications of a useful SDP are periodic updating & understanding and acceptance by managers & practitioners alike.

c) Work Breakdown Structure
→ It is the vehicle for budgeting & collecting costs
→ To monitor & control a project's financial performance. The s/w project manager must have insight into project costs & how they are expended.
→ The str of cost accountability is a serious prj planning constraint
→ A prj manager should not layout lower levels of WBS until level of stability in product str is achieved.

d) S/w Change Order Db:-
→ Managing change is one of the fundamental primitives of an iterative development proces
→ With greater change of freedom, prj can iterate more productively
→ The flexibility increases content, quality & number of iteration that prj can achieve within a given schedule.
→ Change freedom has been achieved in practice through automation, and today's iterative development envt carry the burden of change mgmt.
→ Once s/w is placed in controlled baseline, all changes must be formally tracked & managed.

e) Release Specification
→ The scope, plan & obj evaluation criteria for each baseline release are derived from vision stmt as well as other sources.
→ These artifacts are intended to evolve along with proces achieving

greater fidelity as life cycle process and reqmt understanding matures.

→ There are 2 important forms of requirement.

① ~~Relase~~ Vision statement which captures the contract betn the development group and the buyer.

② Evaluation criteria are transient snapshots of objectives for a given intermediate life cycle milestone.

E.C are defined as mgmt artifacts rather than part of reqmt set. are derived from vision stmt & other sources

→ The system reqmt are captured in the vision stmt.

~~Release Description~~ → Lower level of reqmt are driven by the process in the form of evaluation criteria.

→ So, lower level reqmt can evolve as summarized in following conceptual example for a relatively large prj

ⓐ Inception Iteration:- (Typically 10-20 evaluation criteria)

ⓑ Elaboration " :- (Typically 50 evaluation criteria)

ⓒ Construction Iteration - (" 100 " ")

ⓓ Transition " :- (" 1000 " ")

⑤ Release Description (Docm)

→ It describe the results of each release including performance against each of evaluation criteria in corresponding release specification

→ Release baselines should be accompanied by release description docm that describes the evaluation criteria for that config baseline and provides substantiation.

→ This docm should also include metrics summary that quantifies its quality in absolute & relative terms

(g) Status Assessments.

→ It provide periodic snapshot of prj health & status including s/w prj managers risk anessment, quality indicators and mgmt indicators.

→ Although the period may vary the forcing function needs to persist.

→ The paramount objective of a good mgmt procen is to ensure that the expectation of all stakeholders are synchronized & consistent.

→ Typical status anenments should include a review of resources, personnel staffing, financial data etc.

(h) Environment.

→ An important emphasis is to define development & maintenance envt as a first-clan artifact of the procen.

→ A robust, integrated development envt must support automation of the development procen.

→ This envt should include reqmt mgmt, visual modeling, document automation, feature & defect tracking.

→ A common theme from succesful s/w prj is that they hire good ppl & provide them with good tools to accomplish their jobs.

(i) Deployment.

→ It could take many form depending on prj & could include several docm subset for transitioning product into operational status.

→ For commercial s/w product it may include marketing plan, training courses etc.

Engineering Artifact

1) Vision Document(VD)

→ It provides a complete vision for the s/w system under development & supports the contract between the funding authority and the development organization.

→ A prj vision is meant to be changeable as understanding evolves of the reqmt, architecture, plans & technology.

→ A good vision docm should change slowly.

→ V.D is a written form of users perspective, focusing on enential features of system & acceptable levels of quality..

→ It should contain 2 appendixes, first describe operational concept & second the change risks inherent in vision stmt.

→ It must also include a description of what will be included as well as those features considered but not included.

→ The V.D also ~~contain~~ provides the contractual basis of reqmt visible to stakeholders.

2) Architecture Description

→ It provides an organized view of the s/w architecture under development.

→ It is extracted largely from design model and includes view of design, implementation & deployment sets sufficient to understand how operational concept will be achieved.

→ The architecture ~~de~~ can be described using a subset of design model or as an abstraction of design model with supplementary material, or a combination of both.

E.g:- A subset form could be satisfied by table of contents

3) Software User Manual

→ This provides user with reference documentation necenary to support the delivered s/w.

→ Although content is highly variable across appln domains, user manual should include installation procedures, usage procedures & guidance etc.

→ For s/w products with user interface, this manual should be developed early in the life cycle coz it is a necenary mechanism for communicating & stabilizing an imp set of reqmt

→ It should be written by members of test team, who are likely to understand the users perspective than development team.

Pragmatic Artifacts.

→ Conventional docm-driven approaches squandered incredible amt of engineering tym on developing, formatting, updating docm.

→ In some domain, document driven approach have degenerated over the past 30 years into major obstacles to procen improvement

→ The quality of docm become more important than quality of engineering info.

→ Lengthy & detailed docm which were generally perceived to demonstrate more progren resulted in premature engineering details.

→ A more effective approach is to redirect this documentation effort to improving the rigor and understandability of info source and allowing on-line review of info source.

This philosophy raises the following cultural issues

→ Ppl want to review info but don't understand the language.
→ Ppl want to review the info but don't have access to tools.
→ Useful documentation is self-defining.

* Technical & Mgmt perspective of S/W Architecture.

S/W Architecture: A Management Perspective.

- The most critical technical product of a s/w prj is its architecture.
- From a management perspective, there are three different aspects of architecture

(i) An architecture:- is the design of s/w system. as opposed to design of component.
include all necessary to specify a complete bill of materials

(ii) An architecture baseline:- slice of information across the engineering artifact sets sufficient to satisfy all stakeholders that the vision can be achieved within parameters of busines case

(iii) Architecture description: is an organized subset of info extracted from design set models. It communicates how intangible concept is realized in tangible artifacts.

The importance of s/w architecture & it's close linkage with modern s/w development process can be summarized as follows:

→ Achieving a stable s/w architecture represent a ~~transition from~~ ~~ing~~ significant prj milestone at which critical make/buy decisions should have been resolved.

→ Architecture representations provides a basis for balancing the trade-offs betn the problem space & soln space

→ Poor architecture & immature process are often given as reasons for prj failures

→ A mature process, an understanding of primary reqmts and a ~~not~~ demonstrable architecture are imp prerequisites for predictable planning.

S/W Architecture: A technical perspective

- S/W Architecture encompasses the str of s/w system, ~~or~~ their behavior and patterns that guide these elements, their collaborations and their composition.

→ The contex of s/w arch str, behavior & patterns must include functionality, performance etc.

→ An arch ~~model~~ framework is defined in terms that are abstraction of UML model in the design set.

→ Most real world systems require four ~~process~~ views: design, process, component & deployment.

→ · Design - describes architecturally significant str & func^n of design model
Procen = describes concurrency & control thread relnship among design, component & deployment views.
Component :- describes str of ~~implementation~~ set
Deployment :- describes str of deployment set

→ Design views is probably necessary in every system; the other 3 views can be added to deal with complexity of system at hand.

- The usecase view describes how the system's critical use cases are realized by elements of design model.

- The design view describes architecturally significant elements of the design model. This view addrenes basic str & functionality of soln

- The procen view addrenes run-time collaboration issues involved in executing the architecture on a distributed deployment model including the logical s/w n/w topology, IPC & state mgmt.

- The component view describes the architecturally significant elements of implementation set. This view addrenes the s/w source code realization of system from the perspective of projects integrators & developers.

- The deployment view addrenes the executable realization of the system including allocation of logical procen in distribution view to physical resources of deployment n/w

→ Although the technical details of architecture description are not central s/w mgmt the underlying spirit of architecture first development is crucial to succen.

Software Process Workflows

Workflow → means a thread of cohesive & mostly sequential activities. They are mapped to product artifacts.
There are 7 top-level workflows:

① Mgmt workflow:- controlling the process & ensuring win conditions for all stakeholders

② Envt workflow:- automating the process and evolving the maintenance environment.

③ Reqmt workflow:- analyzing the problem space and evolving the reqmt artifacts

④ Design workflow: modeling the soln & evolving the architecture & design artifacts

⑤ Implementation workflow: programming the components & evolving the implementation & deployment artifacts

⑥ Assessment workflow:- assessing the trends in process and product quality.

⑦ Deployment workflow:- transitioning the end products to the user.

Different principles:-

(1) Architecture-first Approach.
- Extensive reqmt, analysis, design, implementation & assessment activities are performed by construction phase.
- This early life-cycle focus on implementing & testing the architecture must precede full-scale development & testing of all component & must precede the downstream focus on complement & quality.

(2) Iterative life-cycle process:-
→ Each phase portrays at least two iterations of each workflow.
→ This default is intended to be descriptive
→ Some prj may require only one iteration in a phase; other may require several iterations

(3) Round-trip Engineering
→ Raising the envt activities to a first-class workflow is critical.
- Envt is tangible embodiment of prj's process, methods & notations for producing artifacts

(4) Demonstration-based approach.
- Implementations & assessment activities are initiated early in life cycle
- It reflects the emphasis on constructing executable subsets of evolving architecture

Iteration Workflow:-

→ An iteration consists of loosely sequential set of activities in various proportions depending on where iteration is located in development cycle.

→ An individual's iteration's workflow generally includes the following sequence:-

a) Management:- Iteration planning to determine content of release
Develop detailed plan for iteration

b) Environment:- Evolving the s/w change order db to reflect
- new baselines.
Changes to existing baselines for all product, test & envt components.

~~c) Requirement:-~~

c) Requirement:- Analyzing the baseline plan, the baseline architecture and the baseline reqmt set artifacts to fully elaborate the use case to be demonstrated at end of this iteration & their evaluation criteria.

d) Design:- evolving the baseline architecture & baseline design set artifacts to elaborate fully the design model & test model components necessary to demonstrate against evaluation criteria allocated to this iteration.

e) Implementation:- ~~Evolv~~ developing or acquiring any new components, & enhancing or modifying any existing

components, to demonstrate the evaluation criteria allocated to this iteration.

- Assessment: evaluating results of iteration, including compliance with allocated evaluation criteria & quality of current baselines.

- Deployment: transitioning the release either to an external organization or to internal closure so that lessons learned can be captured & reflection in the next iteration.

→ Iterations in inception & elaboration phases focus on mgmt, requirements and design activities.
→ Iterations in construction phase focus on design, implementation and assessment
→ Iterations in transition phase focus on assessment & deployment.

Status Monitoring.

→ Managing risk requires continuous attention to all interacting activities of s/w development effort
→ Periodic status assessment are mgmt reviews conducted at regular intervals to address progress and quality indicators & maintain open communication among all stakeholders.
→ The paramount objective of these assessment is to ensure that expectation of all stakeholders are synchronized & consistent.

→ Status Monitoring provide the following.
- A mechanism for openly addressing, communicating & resolving mgmt issues, technical issues, proj risks
- Objective data derived directly from on-going activities & evolving product configuration
- A mechanism for disseminating lessons, progress, quality trends, and experience info to and from all stakeholders in an open forum.

→ Recurring themes from unsuccessful proj include status assessments that are:
(1) high-overhead activities
(2) frequently canceled

→ They also force the s/w proj manager to collect & review data periodically & encourage dissemination of best practices to & from other stakeholders.

* Milestones:-
→ It must have well defined expectations and provide tangible results.
→ Three type of joint mgmt reviews are conducted throughout the process:-
@ Major milestone:- These systemwide events are held at end of development phase. They provide visibility to systemwide issues, synchronize the ~~magazines~~ mgmt and engineering perspective.

Minor milestone:

These iteration focused-events are conducted to review the content of an iteration in detail & to authorize continued work.

Status assessment

These periodic events provide management with frequent and regular insight into the progress being made.

MAJOR MILESTONE:-

The milestone described in the section may be conducted as one continuous meeting of all concerned parties or incrementally through mostly on-line review of various artifacts:

→ There are considerable differences in levels of ceremony for these events.

① Life Cycle Objective Milestone

→ occurs at the end of inception phase.

→ Goal is to present to all stakeholders a recommendation on how
- to proceed with development including
  - a plan,
  - estimated cost & schedule
  - expected benefits & savings.

→ The vision stmt & critical issues relative to requirements & operational concept are addressed.

→ A draft architecture document and a prototype architecture demonstration provide evidence of completeness of vision &

s/w development plan.

② Life Cycle Architecture Milestone.

→ occurs at the end of elaboration phase.

→ The plans of this milestone include:

① demonstrate executable architecture to all stakeholders (main goal)

② high-fidelity construction phase plan.

③ low-fidelity transition phase plan

→ The ~~require~~ critical issues like.

① stable vision & use case model.

② Evaluation criteria for construction releases

③ Initial operational capability

④ Draft user manual

~~for~~ which are related to reqmt and operational concept are addressed.

→ This will also produce baseline (architecture, vision, s/w development plan) & for evaluation criteria for initial operational capability milestone.

→ The status of ~~sofi~~ software product at the end of this milestone is.

① stable design set

② Make/buy/reuse decisions

③ Critical component prototypes.

→ A succesfully completed life-cycle arch milestone will result in authorization from the stakeholders to proceed with construction phase.

→ A software development plan ready for transition exhibits following characteristics:-

→ The critical use cases have been defined, agreed upon by stakeholders

→ A stable architecture has been baselined in source language format.

→ The risk profile is well understood.

→ The development plan for construction & transition phase with enough fidelity that construction iteration can proceed with predictable results.

(3) Initial Operational Capability Milestone

→ It occurs late in the construction phase.

→ The goal are to assess the readiness of the s/w to begin the transition into customer/user sites ~~& to~~ and to authorize the start of acceptance testing.

→ high fidelity transition phase plan.

→ The issues ~~addressing~~ are addressed concerning,
- installation instruction
- s/w version description
- user & operator manual.

→ ~~are addressed~~ readiness of test envt

→ test s/w for acceptance testing is assessed.

→ The acceptance criteria for product release and releasable user manual is status of requirements.

→ The initiation of transition phase is not necessarily the completion of construction phase.

(4) Product Release Milestone

→ occurs at the end of the transition phase.
→ goal is to assess the completion of s/w & its transition to support organization. if any.
→ The result of acceptance testing are reviewed & all open issues are addressed
→ These issues could increase installation instruction, s/w version description, user & operator manual etc.
→ s/w quality metrics are reviewed to determine whether quality is sufficient for transition to support organization.

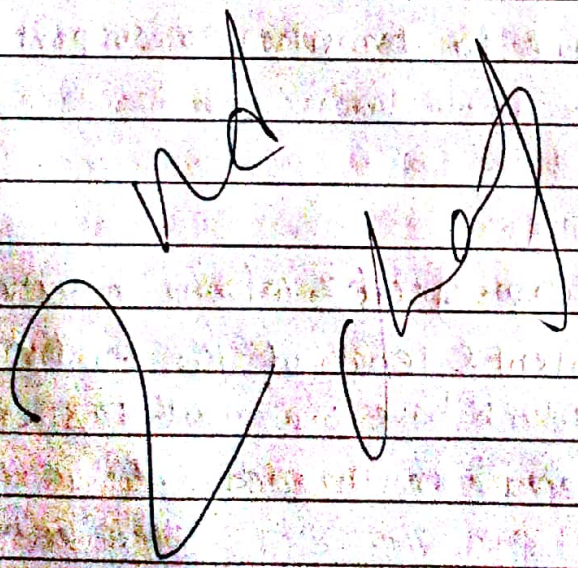
* Minor Milestone.

→ The number of iteration-specific, informal milestones needed depends on the content & length of the iteration.
→ For most iterations, which have one-month to six-month duration, only two minor milestone are needed: the iteration readiness review, the iteration assessment review
→ All iteration are not created equally, coz it can take very different priorities. depending on where prj is in the life cycle.
→ Early iteration focus on analysis & design with (experimentation & risk assessment).
→ Later iteration focus more on completeness, consistency, usability & change mgmt.

(a) Iteration readiness → This informal milestone is conducted at the start of each iteration to review detailed plan and evaluation criteria have been allocated to this iteration.

Iteration Assessment Review.

This informal milestone is conducted at the end of each Iteration to assess the degree to which Iteration achieved the objectives & satisfied its evaluation criteria, to review Iteration results etc.



Techniques of Planning, Controlling & Automatic Software Process.

8.1) Iterative Process Planning.

→ Like s/w development, project planning requires an iterative process.
→ Like s/w, plan is an intangible piece of intellectual property to which all the same concepts must be applied.
→ Plans have an engineering stage (where plan is developed) & production stage (where plan is executed)

* Work Breakdown Structures:- (WBS)

→ A good WBS & its synchronization with the process framework are critical factor in s/w prj success
→ WBS is simply a hierarchy of elements that decomposes the prj plan into discrete work tasks.
→ A WBS provides the following info str:
  @ A delineation of all significant work.
  (b) A clear task decomposition for assignment of responsibilities
  (c) A framework for scheduling, budgeting & expenditure tracking.

* Conventional WBS Issues

Conventional WBS frequently suffer from three fundamental flaws:-
1. They are prematurely structured around the product design
2. They are prematurely decomposed, planned, & budgeted in either too much or too little detail.
3. They are prj-specific & cross-prj comparisons are usually difficult or impossible

(1) A WBS is architecture for financial plan.

- Just as s/w architectures need to capsulate components that are likely to change, so must planning the architectures
- To couple the plan tightly to the product str may make sense if both are reasonably mature.
- However, a looser coupling is desirable if either the plan or architecture is subject to change.

(2)

(2) Large prj tend to be overplanned & small prj tend to be underplanned.

- The mgmt team plans & conducts the prj with coarse tasking & cost and schedule accountability.
- Both approaches are out of balance.
- In general WBS elaborated to at least 2 or 3 levels may make sense. It for large scale other level are added as per necessity in later phases.

(3) Most organization allow individual prj to define their own prj-specific

- str tailored to prj manager's style, customer's demand or other styles
- → With no standard WBS, it is extremely difficult to compare plans, financial data, schedule data, cost trends, quality trends etc across multiple projects
- Each prj organizes work differently & uses different units of measure.

Evolutionary WBS:-

→ It should organize the planning elements around the process framework rather than product ~~or~~ framework

→ This ~~best~~ approach better accomodates the expected changes

in the evolving plan and allows the level of planning fidelity to evolve in a straightforward way.

→ The basic recommendation for WBS is to organize the hierarchy as follows:-

(a) First-level WBS ~~are~~ elements are workflows (mgmt, envt, reqmt----)
These elements are usually allocated to single team.
They constitute the anatomy of a project for the purposes of planning & comparison with other prj.

(b) Second-level elements are defined for each phase of life cycle (inception.---).
These elements allow fidelity of plan to evolve in ~~the~~ natural way. with the level of understanding of reqmt, arch & risk.

(c) Third-level elements are defined for the focus of activities that produce the artifacts of each phase
These lowest level elements collects cost of discrete artifact or may be decomposed further into several lower level activities. to produce a single artifact.

→ The WBS decomposes character of a prj and maps it to the life cycle, the budget & personnel.

→ While s/w development plan & business case provide a context for review, the WBS & relative budget allocated among elements provide most meaningful indicators of mgmt approach, priority & concern.

* Planning Guidelines:-

→ S/W proj span broad range of applⁿ domains.

→ It is risky but valuable to make specific planning recommendations independant of proj context.

→ It is valuable coz most ppl in mgmt position are looking for starting points, they can flesh out specific details with.

→ Proj independent planning advice is also very risky.

→ There is a risk that guidelines may be adopted blindly without being adapted to specific project circumstances. Along with that there is a risk of misinterpretation also.

→ Two simple planning guidelines must be considered when proj plan is initiated & assessed.

→ First guideline prescribe a default allocation of cost among first level WBS elements.

WBS budgeting defaults.

First-level WBS Element	Default Budget.
Mgmt	10%.
Envt	10%.
Reqmt	10%.
Design	15%.
Implementation	25%.
Assessment	25%
Deployment	5%.
Total.	100%.

→ The source of data in the first table & second table is not bank authorized or case studies experience, it is came from author experience.

→ First table provides default allocation for budgeted costs of each first level WBS elements.
→ This allocation provide good benchmark for assessing the plan by understanding the rationale for deviation from these guidelines.
→ Important pt is cost allocation not effort allocation.

→ Second table prescribes the allocation of effort & schedule across the life-cycle phases.
→ These values can vary widely, depending on specific constraints of an application.
→ They provide an average expectation across a spectrum of application domains.

Default distribution of effort & schedule by phase.

Domain	Inception	Elaboration	Construction	Transition
Effort	5%	20%	65%	10%
Schedule	10%	30%	50%	10%

* Cost and schedule Estimation Process.
→ Proj plan need to be derived from 2 perspectives.
→ First is a forward-looking, top-down approach.
→ Second is a backward-looking, bottom up approach

@ First Approach.

→ It starts with an understanding of general reqmt & constraint; derives a macro-level budget & schedule; then decomposes these elements into lower-level budgets and intermediate milestones.

→ From this perspective, following planning sequence would occur:-

(1) The s/w prj manager develops characterization of overall size, procen, envt, ppl & quality required for prj.

(2) A macro-level estimate of total effort & schedule is developed using s/w cost estimation model.

(3) The s/w prj manager partition estimate for effort into top-level WBS along with schedule partitioning into major milestone and ~~sta~~ effort partitioning into staffing profile. Now there is a prj-level plan

(4) Now, subprj managers are given responsibility for decomposing each of WBS elements into lower-levels using their top-level allocation, staffing profile & major milestone dates as constraints.

(b) Second Approach

→ It starts with end in mind, analyze macro-level budget & schedule; then sum all these elements into higher-level budgets and intermediate milestones.

→ From this perspective the following planning sequence would occur:-

(1) The lowest level WBS elements are elaborated into detailed tasks for which budgets & schedules are estimated by responsible WBS element manager.

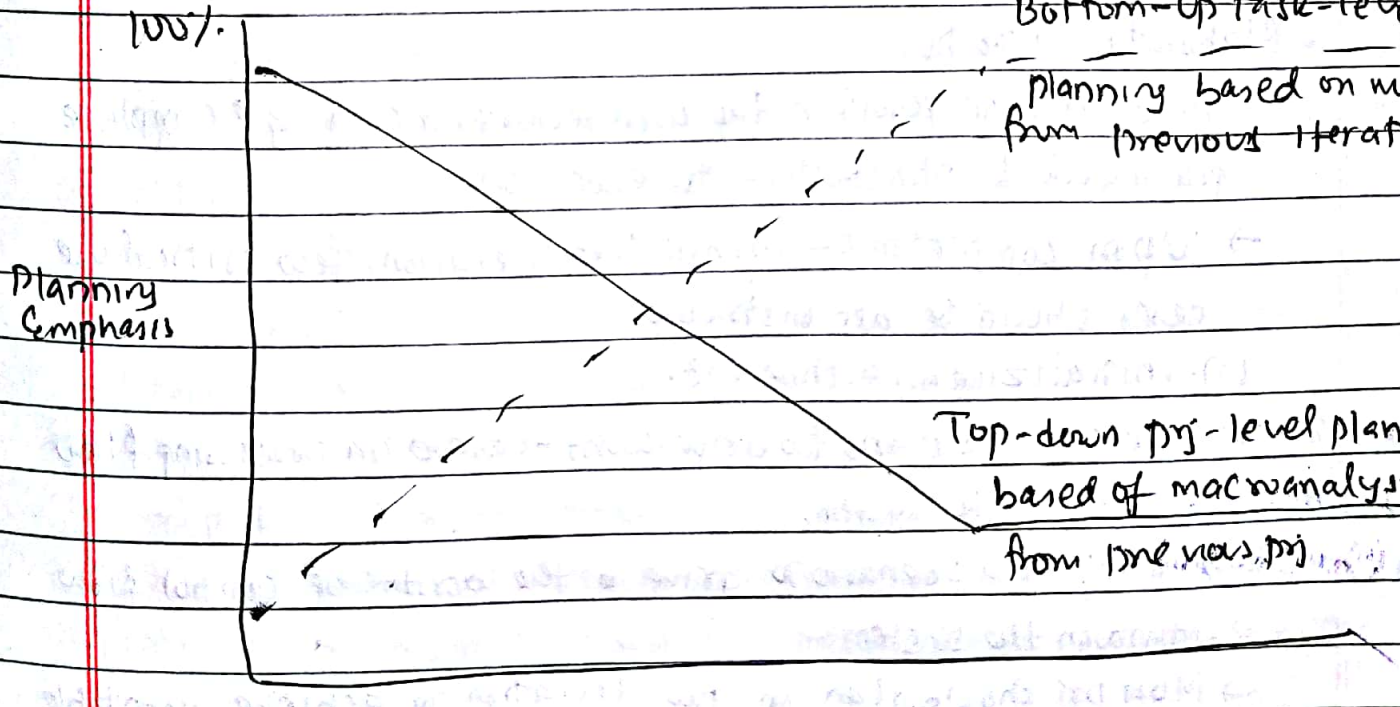
(2) Estimates are combined & integrated into higher level budgets and milestones. The biases of individual & estimators need to be homogenized so that there is a consistent base of negotiation.

(3) Comparisons are made with top-down budgets & schedule milestones: Gross diff are assessed and adjustment are made in order to converge on agreement betn top-down & bottom-up estimates.

These two approaches must be used together in ~~planning~~ balance, throughout life cycle of project

During engineering stage, top-down perspective will dominate bcoz there is usually not enough depth of understanding nor stability in detailed task sequences to perform credible bottom up planning.

During production stage, bottom up perspective will dominate bcoz there is enough precedent experience and planning fidelity.



* Iteration Planning Procen

→ Iteration is used to mean a complete synchronization across a project with well-orchestrated global anenment of entire project plan.

→ Other micro-iterations, such as monthly, weekly or daily builds are performed en route to these prj-level synchronization pt.

- Inception iteration:- The early prototyping activities integrate the fandation components of candidate architecture.
  - → They provide an executable framework for elaborating the critical use case of system.
  - → This framework includes existing components, commercial components and custom prototypes.
  - → This above above components are sufficient to demonstrate a candidate architecture and regmt understanding to establish busines case, vision & slw development plan.

- Elaboration iteration:-
  - → These iteration result in the architecture, including complete framework & infrastructure for execution
  - → Upon completion of architecture iteration, few critical use cases should be demonstrable:
  - (1) initializing an architecture.:
  - (2) injecting a scenario to drive worst-case data processing flow through the system.
  - (3) to inject a scenario to drive the worst-case control flow through the system
  - → Most prj should plan on two iteration to achieve acceptable behavior

- Construction iteration:-

→ Most project require at least 2 major construction iterations: an alpha release and beta release

→ An alpha release include executable capability for all critical use cases. It usually represents only 70% of total product breadth & perform at quality levels below expected in final product

→ A beta release provides 95% of total product capability breadth & achieves some of important quality attributes.

→ Typically a few more features need to be completed & improvement in robustness & performance are necessary for final product release to be acceptable.

- Transition Iteration

- Most project use a single iteration to transition a beta release into a final product.

- However, coz of overhead associated with a full-scale transition to the user community, must prj learn to live with single iteration betn beta release & final product release.

* Project Organization and Responsibilities.

This shows a default project organization and maps project-level roles and responsibilities.

The main feature of default organization are as follows:-

→ Prj Mgmt team is an active participant ; responsible for producing as well as managing.

→ Architecture team is responsible for real artifacts and for the

integration of components, not just staff function.

→ The development team owns the component construction & maintenance activities. The assessment team is separate from development.

→ Quality is everyone's job, integrated into all activities and checkpoints. Each team takes responsibility for a different quality perspective.

Software Mgmt Team

Software Mgmt
- → System Engineering
- → Financial administration
- → Quality Assurance

Artifacts
- Business case
- Vision.
- WBS
- status assessment
- Reqmt set.

Responsibilities
- Resource commitment
- Personnel assignment
- Plans, priorities
- Scope definition
- Risk mgmt
- Project control

Life Cycle Focus.

Inception	Elaboration	Construction	Transition
Elaboration phase planning	Construction phase planning	① Transition phase planning	- Customer satisfaction
Team formulation	② Full staff recruitment	② Construction plan optimization	- Contract closure
Contract baselining	③ Risk resolution	③ Risk mgmt	- Sales support
Architecture costs.	④ Product acceptance criteria		- Next gen planning.
	⑤ Construction cost		

Software Architecture Team

Software Architecture Team

- Demonstration
- Use case modelers
- Design modelers
- Performance analysts

Artifact

- Architecture description
- Requirement set
- Design set
- Release specification

Responsibilities

- Reqmt trade offs
- Design trade-offs
- Component selection
- Initial Integration
- Technical risk resolution

Life Cycle Focus

Inception	Elaboration	Construction	Transition
• Architecture prototyping	• Architecture baselining	- Architecture maintenance	- Architecture maintenance
• Make/buy trade-offs	• Primary scenario demonstration	- Multiple component issue resolution	- Performance tuning.
• Primary scenario definition	• Make/buy trade-off baselining.	- Quality improvement	
• Criteria definition of architecture evaluation			

Software Development Team:-

Software Development → Component teams

Artifacts
- Design set
- Implementation set
- Deployment set

Responsibilities
- Component design
- " implementation
- " stand-alone test
- " maintenance
- " documentation

Life Cycle Focus

Inception	Elaboration	Construction	Transition
→ Prototyping support → Make/buy trade-offs	• Critical component design • Critical component implementation & test • Critical component baseline	• Component design • Component implementation • Component stand-alone test • Component maintenance	• Component maintenance • Component documentation

Software Assessment Team

Software Assessment

- Release testing
- Change management
- Deployment
- Environment support

Artifacts

- Deployment set
- SCO database
- User manual
- Environment
- Release specification
- Release description
- Deployment docms

Responsibilities

- Project infrastructures
- Independent testing
- Reqmt verification
- Metric Analysis
- Change Management

Life Cycle Focus

Inception	Elaboration	Construction	Transition
- Infrastructure planning	- Infrastructure baseline	- Infrastructure upgrades	- Infrastructure maintenance
- Primary scenario prototyping	- Architecture release testing	- Release testing	- Release baselining
	- Change mgmt	- Change Management	- Deployment to user
	- Initial user manual	- User manual baseline	- Change mgmt

* Evolution of Organizations.

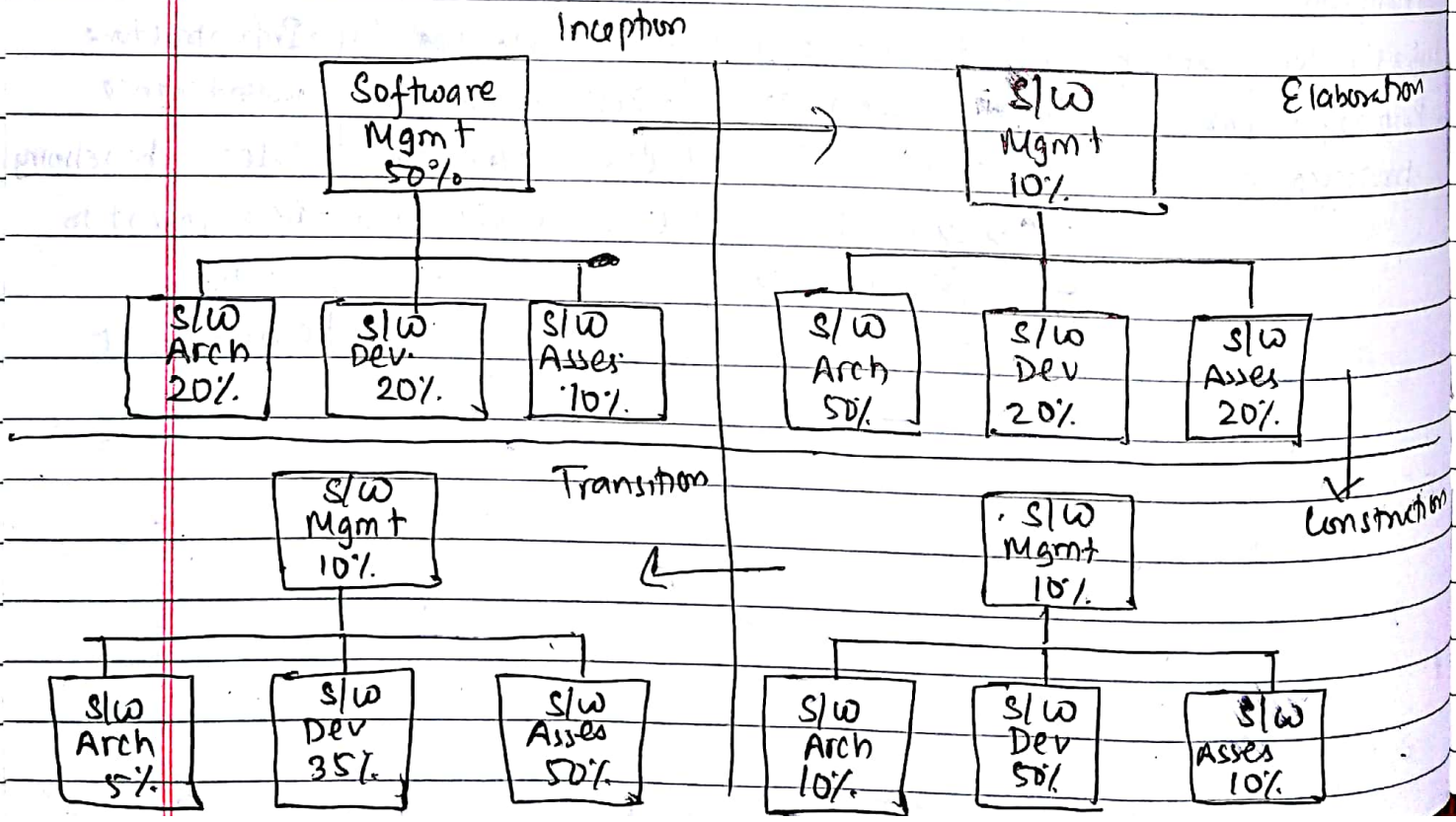
A different set of Activities is emphasized in each phase, as follows.

① Inception: organization focused on planning, with enough support from other teams to ensure that the plans represent a consensus of all perspectives.

② Elaboration: architecture focused organization in which driving forces of the prj reside in the s/w architecture team & are supported by s/w (development & anenment teams)

③ Construction: a fairly balanced organization in which most of the activity resides in s/w (development & anenment) teams

④ Transition team:- a customer focused organization in which usage feedback dorives the deployment activities.



Process Automation - Tools & Environment

→ Automating the process & and establishing an infrastructure for supporting various proj workflows are important activities of engineering stage of life cycle

→ The envt that provides procen automation is tangible artifact that is critical to life-cycle cost of system being developed.

→ Each level of procen requires certain degree of automation for it to be carried out efficiently.

① Metaprocen:- An organization's policies, procedure and practices for managing a s/w-intensive line of busines.
- Automation support for this level is called an infrastructure.

② Macroprocen:- a project's policies, procedure & practices for producing a complete software product within certain cost, schedule, quality constraints.
- Automation support for a project's procen is called an environment.

③ Microprocen:- a project's policies, procedure & practices for ~~producing a compte~~ achieving an artifact of the s/w procen.
→ Automation support for generating an artifact is generally called a tool.

Tools: Automation Building Blocks.

→ There are many tools to automate the s/w development process.
→ Each of the procen workflows has a distinct need for automation support.

→ Some of the critical concerns anciated with each workflow are written below.

① Management

→ There are many opportunities for automating project planning & control activities of mgmt workflow.

→ S/w cost estimation tools & WBS tools are useful for generating the planning artifacts

→ This automation support can considerably improve the insight of metrics collection & reporting concepts.

② Environment

→ Configuration mgmt and version control are enential in a modern iterative development proces

→ There are change mgmt automation that must be supported by the environment

③ Requirement

Conventional approaches decomposed system reqmt
into
subsystem reqmt
into
component reqmt
into
unit reqmt.

→ In modern process, system are captured in vision stmt. Lower level of reqmt are driven by the proces organized by iteration rather than lower level component - in the form of evaluation criteria.

→ Iterative models allows the customer and developer to work with tangible evolving versions of the system. Reqmt can be evolved along with an architecture and evolving set of applⁿ increments.

④ Design

→ The tools that support the reqmt, design, implementation & anenment workflow are usually used together.

→ The primary support reqd for design workflow is visual modeling which is used for capturing design models, transitioning them into source code.

⑤ Implementation

→ The implementation workflow relies primarily on a programming envt but also must include substantial integration with change mgmt tools, visual modeling tools, test automation tools to support productive iteration.

⑥ Assessment & Deployment

→ Assessment workflow requires all tools as well as additional capabilities to support test (automation & mgmt)

→ To increase change freedom, testing & document production must be mostly automated.

→ Defect tracking is important tool that supports anenment : provides

The Project Environment

→ It evolves around 3 discrete states: the prototyping envt, the development envt & the maintenance envt.

1) The prototyping environment include an architecture testbed for prototyping project architectures to evaluate trade-offs during the inception and elaboration phase of the life cycle. The informal configuration of tools should be capable of supporting the following activities..
   - Performance trade-offs & technical risk analysis
   - Make/buy trade-offs.
   - Fault tolerance.

2) The developing environment should including a full suite of development tools needed to support the various process workflows & to support RTE to max^m extent possible.

3) The maintenance environment should typically coincide with a mature version of development environment. In some cases, the maintenance envt may be a subset of of dev envt delivered as one of the project's end practices

Toward the end, there are four important envt disciplines that are critical to mgmt context & success of modern iterative development process.

Round-Trip Engineering:-

→ As the s/w industry moves into maintaining different information sets for engineering artifacts, more automation support is needed to ensure efficient and error-free transition of data from one artifact to another.

→ RTE is the envt support necessary to maintain consistency among the engineering artifacts.

→ The automated translation of design model to source code is fairly well established.

→ Compilers & linkers have long provided automation of source code into executable code.

→ Today's envt don't support automation to the greatest extent possible

→ The primary reason for RTE is to allow freedom in changing s/w engineering data sources.

→ The configuration control of all technical artifacts is crucial to maintaining a consistent & error-free representation of evolving product

Change Mgmt:-

→ Tracking changes in the technical artifacts is crucial to understanding the true technical progress trends & quality trends toward delivering an acceptable end product.

→ In a modern process in which reqmt, design, and implementation set artifacts are captured early in life cycle & evolved through multiple generations.

@S/w Change Orders

→ The atomic unit of S/w work that is authorized to create, modify or obsolesce components within a configuration baseline is called S/w change order (SCO)

→ They are a key mechanism for partitioning, allocating & scheduling S/w work against an established S/w baseline & for assessing progress and quality.

→ The basic field of SCO are title, description, metrics, resolution, assessment & disposition

Title:- suggested by originator & is finalized upon acceptance by CCB.

Description: Problem description includes the name of originator, date of origination, CCB-assigned SCO identifier and relevant version of related support S/w. describes the change needed

Metrics: The metrics collected for each SCO are imp for planning, scheduling & assessing quality improvement. Change categories are type 0 (critical bug), type 1 (bug), type 2 (enhancement) and so on.

Resolution: It includes the name of the person responsible for implementing the change, components changed, the actual metrics and description of change.

Assessment: It describes the assessment technique as either inspection, analysis, demonstration or test.

Disposition:- The SCO is assigned one of the following states by the CCB.

proposed, accepted, rejected, archived, inprogress, in assessment and closed

(b) Configuration Baseline: (CB)

→ A configuration baseline is named a collection of s/w components and supporting documentation that is subject to change mgmt and is upgraded, maintained, tested, statused as a unit

→ There are generally two clanes of baselines: external product releases and internal testing releases.

→ A CB is named collection of components that is treated as a unit.

→ It is controlled formally for it is a packaged exchanged betⁿ groups.

→ A prj may release a ~~config~~ CB to user community for beta testing.

→ There are 3 levels of baseline releases are required for most systems: major, minor and interim.

→ Each level corresponds to a numbered identifier such as

M is major release number, M is the minor release number,

X is interim release identifier.

→ Major & Minor releases are intended to be external product releases that are persistent and supported for a period of time.

→ An interim release corresponds to developmental configuration.

→ Once s/w is placed in controlled baseline, all changes are tracked.

Change categories are as follows:

(a) Type 0: critical failures

(b) Type 1: bug or defect

(c) Type 2: a change that is enhancement rather than response to defect

(d) Type 3: change that is necessitated by update to reqmt.

(e) Type 4: changes that are not accomodated by the other categories

(c) Configuration Control Board

→ A CCB is a team of people that functions as the decision-authority on the content of configuration baselines

→ A CCB usually include s/w manager, s/w arch manager, s/w development manager, s/w assessment manager & other stakeholders

→ While CCB typically takes action through board meetings, on-line distribution, concurrence & approval of CCB actions may make sense under many proj circumstances

→ The operational concept of an iterative development process must include comprehensive change mgmt of evolving s/w baselines.

→ The fundamental process for controlling the s/w development & maintenance activities is described through sequence of states traversed by an SCO.

(1) [Proposed] - A proposed change is drafted & submitted to the CCB.

(2) [Accepted, archived, rejected] - The CCB assigns a unique identifier & accepts, archives & rejects each proposed change. Acceptance includes change for resolution in next release; Archiving accepts change but postpones change for resolution in future release; Rejection judges the change to be without merit

(3) [In progress] - The responsible analyzes, implements & tests a solution to satisfy the SCO. This task includes updating documentation, release notes etc.

(4) [In assessment]: The independent test team assesses whether SCO is completely resolved. When independent test team deems the change to be satisfactorily resolved, the SCO is submitted to CCB for closure

(5) [Closed]: When the development organization, independent test organization & CCB concur that the SCO is resolved & transitioned to closed.

INFRASTRUCTURES:

→ The organization's ~~per~~ infrastructure provides the organizational capital assets, including 2 key artifacts:
  → a policy that captures the standards for prj s/w development process
  → environment that captures an inventory of tools.

@ Organization Policy:-

→ usually packaged as a handbook that defines the life cycle & the process primitives (major milestones, metrics, roles, artifacts etc).
→ The handbook provides a general framework for answering the following questions:
  • What gets done? (activities & artifacts)
  • Who does it? (team roles & responsibilities)
  etc.
→ The need for balance is imp consideration in defining organizational policy.
→ Effective organizational policies have several recurring themes.
  • They confine policies to real shalls & enforce them
  • They are concise & avoid policy that fill 6-inch documents.
  • Waivers are exception not rule.
  • Appropriate policy is written at the appropriate level.
→ Standardization of s/w development techniques & tools across lines of business have proven to be difficult ~~as~~ as tools, techniques, methods cultures can be different.
→ The organization policy is defining the document for organizations s/w policies.
→ In any process assessment, this is tangible artifact that says what you do

Organization Environment.

→ O.E for automating the default procen will provide info on how things get done as well as tools & techniques to automate the procen.

→ Some of the typical component of an organization's automation building blocks are as follows:

- Standardized tool selections, which promote common workflows & a higher ROI on training.
- Standard notations for artifacts such as UML for all design model or Ada 95 for all custom daeloped
- Activity templates

→ Other indirectly useful components of organizations infrastructure.

- A library prj artifact examples such as s/w development plans, arch descriptions.
- Existing case studies including objective benchmark of performance for succenful prj that followed the organizational procen.

STAKEHOLDER ENVIRONMENTS.

→ The transition to a modern iterative development procen with supporting ~~her~~ automation should not be restricted to the development team.

→ Many large scale contractual projects include ppl in external organizations that represents other stakeholders participating in the development procen

→ These stakeholder representatives also need accen to development envt resources they can contribute value to overall effort.

→ An on-line envt accensible by external stakeholders allows them to participate in procen as follows.

- Accept & use executable increments for hands-on evaluation
- Use the same on-line tools, data & reports that the s/w development organization uses to manage & monitor the prj

→ There are several important reasons for extending development envt resources into certain stakeholder domains.

- Technical artifacts are not just paper. Electronic artifacts such as visual models & source code are viewed more efficiently by using tools with smart browsers.
- Even paper docms should be delivered electronically to reduce production costs & turnaroundTime.

→ Once ~~elect~~ envt resources are electronically accensible by stakeholders, continous & expedient feedback is much more efficient, tangible & useful.

→ It is imp for stakeholders to avoid abusing accen to participate by adding value, and to avoid interrupting development.

Project Control

→ The quality of software products and the progress made towards project goals must be measurable throughout s/w development cycle.

→ The goal of s/w metrics are to provide the development team and the management team with the following.

- An accurate assessment of progress to date
- Insight into the quality of evolving s/w product
- A basis for estimating the cost and schedule for completing the product with increasing accuracy overtime

* The Seven-Core Metrics.

→ These are used in all s/w projects.

→ They can be used in numerous ways to help manage projects and organizations

→ In an iterative prj the historical values of previous iterations & prj provide precedent data for planning subsequent iterations & projects.

→ The 7 core metrics are based on common sense & field-experience with both succesful & unsuccesful metrics programs

→ Their attributes include the following:-

ⓐ They are simple, objective, easy to collect, easy to interpret and hard to misinterpret

ⓑ Collection can be automated & non intrusive

ⓒ Their fidelity improves the life cycle.

ⓓ They are useful to both mgmt & engineering personnel.

→ Three are mgmt indicators & 4 are quality indicators.

Ⓐ Management Indicators.

→ There are 3 fundamental sets of management metrics: technical progress, financial status & staffing progress.

→ Financial status is well understood; it always has been

→ The problem is to assess how much technical progress has been made. Before, conventional prj technical progress could be viewed on paper documents.

① Work and Progress (work performed over time)

Purpose:
- Iteration planning
- Plan Vs actuals
- Management indicator.

Perspectives
→ SLOC → Scenarios
→ Function point → Test cases
→ Object points → SCOs.

7 Core Metrics:
Ⓐ Mgmt Indicators
- Work & Progress
- Staffing & Team Dynamics
- Budgeted cost & Expenditure

② Budgeted cost & expenditures (cost incurred over time)

Purpose:-
- Financial insight
- Plan Vs Actuals
- Management Indicators

Perspectives
- Cost per month - % of budget expanded.
- Full time staff per month

③ Staffing and Team Dynamics (Personnel changes over time)

Purpose:

- Resource plan vs actuals
- hiring rate
- attrition rate

Perspective:-

- Ppl per month added.
- Ppl per month leaving.

(B) Quality Indicators

→ The 4 quality indicators are based primarily on measurement of s/w change across baselines of engineering data.

① Change Traffic & Stability

Purpose:- - Iteration planning
- Mgmt Indicator of schedule convergence

Perspective:-

SCOs opened vs SCOs closed
by Type (0, 1, 2, 3, 4),
by release/component/subsystem

② Breakage and Modularity

Purpose:-

- Convergence
- Software scrap
- Quality indicator

Perspective:
- Reworked SLOC per change by:
  Type (0,1,2,3,4)
  release/component/subsystem

③ Rework & Adaptability

Purpose:
- Convergence
- Software rework
- Quality indicator

Perspective
- Avg hrs per change, by
  Type (0,1,2,3,4)
  release/component/subsystem

④ MTBF and Maturity

Purpose:
- Test coverage/adequacy
- Robustness for use
- Quality indicator

Perspective
- Failure count
- Test hours until failure
- By release/component/subsystem

Quality Indicators
① Change Traffic & Stability
② Breakage & Modularity
③ Rework & Adaptability
④ MTBF & Maturity

* Metrics Automation

⇒ There are many opportunities to automate the project control activities of a software project

→ For managing against a plan, a s/w proj control panel (SPCP) that maintain an on-line version of status of evolving artifacts provides a key advantage.

→ To implement a complete SPCP, it is necessary to define and develop the following:

① Metrics primitives: indicators, trends, comparisons

② Graphical user interface (GUI): support for SP manager role & flexibility to support other roles

③ Metrics collection agent: Data extraction from environment tools that maintain engineering notations

④ Metrics data mgmt server:- Data mgmt support for populating the metric displays of the GUI.

⑤ Metric definition: Actual metric presentation for reqmt progress, design progress, implementation progress, assessment progress etc

⑥ Actor: Typically monitor & administrator

(SPCP)

* Metrics Automation

⇒ There are many opportunities to automate the project control activities of a software project

→ For managing against a plan, a s/w proj control panel (SPCP) that maintain an on-line version of status of evolving artifacts provides

→ To imp necessary to define and develo

① Metric

② Gra for SP manager role & roles

③ M ction from environment tool- ng notations

(SPCP)

SPCP (S/W Proj Control Panel)
Define & Develop following
- Actor
- Metric definition
- Metrics primitive - quality trend
- Metric data mgmt server
- Metric collection agent
- GUI

④ Metrics data mgmt server:- Data mgmt support for populating the metric displays of the GUI.

⑤ Metric definition: Actual metric presentation for reqmt progress, design progress, implementation progress, assessment progress etc

⑥ Actor: Typically monitor & administrator

→ The s/w prj manager role has defined a top level display with 4 graphical obj :-

1. Project activity status.
   → The graphical obj ~~in upper left~~ provides an overview of the stats of top-level WBS elements.
   → The 7 elements could be coded red, yellow & green to reflect the current earned value status

2. Technical artifact stats.
   → The graphical obj provides an overview of stats of evolving technical artifacts.
   → ~~The require~~.

3. Milestone progress
   → It provides a progress assessment of achievement of milestone against plan & provides indicators of current values

4. Action item progress.
   → It provides a different perspective of progress showing current number of open & closed issues

→ The top-level use case, which describes the basic operational concept for an SPCP, corresponds to monitor interacting with control ~~panel~~ panel :
① Start the SPCP. It shows current info that was saved when user last used it.
② Select the panel preference. User selects from list of previously defined default ~~defined~~ panel preferences. SPCP displays preference selected.

- Select value or graph metric: User selects whether metric should be displayed for a given point in time or in graph.

- Select to superimpose controls: User points to graphical obj and request that control values for that metric & point in time be displayed

- Drill down to trend: User points to graphical obj displaying a pt in time & drills down to view trend for metric.

- Drill down to pt in time: User pts to graphical obj displaying a trend & drills down to view values by metric.

- Drill down to lower level of info: User pts to graphical obj displaying a pt in time & drills down to view next level of info.

Ch 4

Modern Approach to s/w Project & Economics.

4.1) Elements of Modern s/w Projects and Management Principles

Elements:-

(a) Continuous Integration

→ Iterative development produces the architecture first, allowing integration to occur as verification activity of design phase and enabling design flaws to be detected & resolved earlier in life cycle.

→ This integration inherent in an iterative development process also enable better insight into quality trade-offs.

→ System ~~architecture~~ characteristics that are largely inherent in the architecture are tangible earlier in the process.

→ The primary discriminator of a successful modern process is inherent in overall life-cycle expenditures for assessment & testing.

→ Conventional prj mixed in insufficient integration and late discovery of design ~~flaw~~ issues expand 40% or more in integration & test of total resources

Modern prj with mature & iterative process deliver a product with only about 25% of total budget consumed by these activities

(b) Early Risk Resolution

→ focus on early confrontation of risks in the life cycle by big resource commitment in production stage.

→ A modern process attacks 20% of reqmt, use case, risk & components.

→ The effect of the overall life-cycle philosophy on 80/20 lessons learned over the past 30 years of s/w mgmt experience provides useful risk mgmt perspective

- 80% of engineering is consumed by 20% of components
- " " slw cost " " " 20% " "
- " " errors are caused " " " " "
- " " slw scrap & rework caused " " " changes
- " " resource consumption is consumed by 20% of components
- " " progress is made by " " people

(c) Evolutionary Requirements:-

→ Conventional approaches decomposed system reqmt into subsystem reqmt, subsystem into component reqmt & component reqmt into unit reqmt

→ With an early life cycle emphasis on reqmt, 1st design 2nd, complete traceability bet^n reqmt & design components.

→ Most modern architectures that use commercial components, legacy components, distributed resources & object oriented method are not traced to reqmt they satisfy.

→ Top-level system reqmt are retained as the vision, but lower level reqmt are captured in evaluation criteria attached to each intermediate release.

(d) Teamwork Among Stakeholders

- Many aspect of classic development process cause stakeholder relationships to degenerate into mutual distrust, making it difficult to balance reqmt, product features & plan.
- A more iterative process, with more-effective working relnships bet^n stakeholders, allows trade-offs to be based on more objective understanding by everyone.
- This process requires a development organization that is

focused on achieving customer satisfaction & high product quality in profitable manner.

- A modern iterative process that focuses on demonstrable results ~~req~~ requires all stakeholders to be educated in the important distinction betn apparently negative results & evidence of real progress.

(c) Top 10 s/w Mgmt Principles

* Base the process on architecture-first approach.
  - Architecture is solid foundation for 20% of stuff that drives the overall success of the project.

* Establish the iterative-life cycle process that confronts the risk early.

* Transition design methods to emphasize component-based development
  - Complexity of s/w effort is mostly a func^n of number of human-generated ~~effo~~ artifacts.
  - Making ~~s~~ sol^n smaller reduces mgmt completely

* Establish a change mgmt envt.
  - Dynamics of iterative development, including concurrent workflows - by diff team working on shared artifacts

* Enhance change freedom, thru tools that support roundtripengineering
  Automation enables to spend more time on engineering and less on overhead tasks

* Capture design artifacts in model based notation.
  - An engineering notation for design enables complexity control, objective assessment.

* Instrument the process for objective quality control & progress assessment.
  - Progress & quality indicators are derived directly from evolving artifacts.

* Use a demonstration-based approach to assess intermediate artifacts.
  - Integration occurs early & continuously throughout life cycle. Intermediate results are objective & tangible

* Plan intermediate releases in groups usage scenarios with evolving levels of detail.
  - Reqmt, design, plan evolve in balance. Useful s/w releases are available early in life cycle.

* Establish a configurable process that is economically scalable
  - Methods, tools & experience can be applied straightforwardly to a broad domain, providing improved return on investment

Next-Generation Software Economics and Cost Models

* Software Economics :-

Following points ~~improve~~ summarize important themes on how s/w mgmt framework should perform.

(1) Finding & fixing a s/w problem after delivery costs 100 times more than finding & fixing the problem in early design phases.

(2) You can compress s/w development schedules 25% of nominal but not more

(3) For every $1 you spend on development, you will spend $2 in maintenance

(4) S/w development & maintenance costs are primarily a function of the number of source lines of code.

(5) Variations among people account for biggest differences in s/w productivity

(6) Only about 15% of s/w development is devoted to programming

(7) The overall ratio of s/w ~~engineering~~ to h/w cost is still growing. In 1955 ⇒ 15:85, In 1985, 85:15.

(8) Walkthrough catch 60% of error

(9) 80% of contribution comes from 20% of contributions.

(10) S/W system & product typically cost 3 times as much per SLOC as individual S/W programs. S/W system products cost 9 times as much.

Cost Models:-

→ A next-gen cost model should explicitly separate an architectural engineering from appln production. just as an architecture-first process does.

→ The cost of designing, producing, testing & maintaining the architecture baseline is a funcn of ~~staff~~ scale, quality, team skill.

→ There should still be diseconomy of scale in architecture cost models as it is inherently driven by research & development oriented concerns.

→ When an organization achieves stable architecture, production cost should be an exponential funcn of size, quality & complexity.

→ The production stage should ~~still~~ reflect a diseconomy of scale similar to conventional economic models.

→ Next gen S/W cost models should estimate large-scale architecting with economy of scale.

→ Next gen envt & infrastructures are moving to automate & standardize many of these mgmt activities thereby requiring a lower percentage of effort for overhead activities.

→ The architectures and applications have different units of mass and are representations of the soln space

Modern Procen Transition - Paradigm Shifts.

- Several culture shift must be overcome to transition succenfully to a modern s/w mgmt procen..
- There are general indication of a succenful transition to a modern culture.
- This discusses several of rough indicators to look for in order to differentiate py that have made a genuine cultural transition from projects.

• ① Lowel level and mid-level managers are performers.

② Reqmt & designs are fluid & tangible

③ Ambitous demonstration are encouraged

④ Good & bad project performance is much more obvious earlier in the life cycle.

⑤ Early increments will be immature

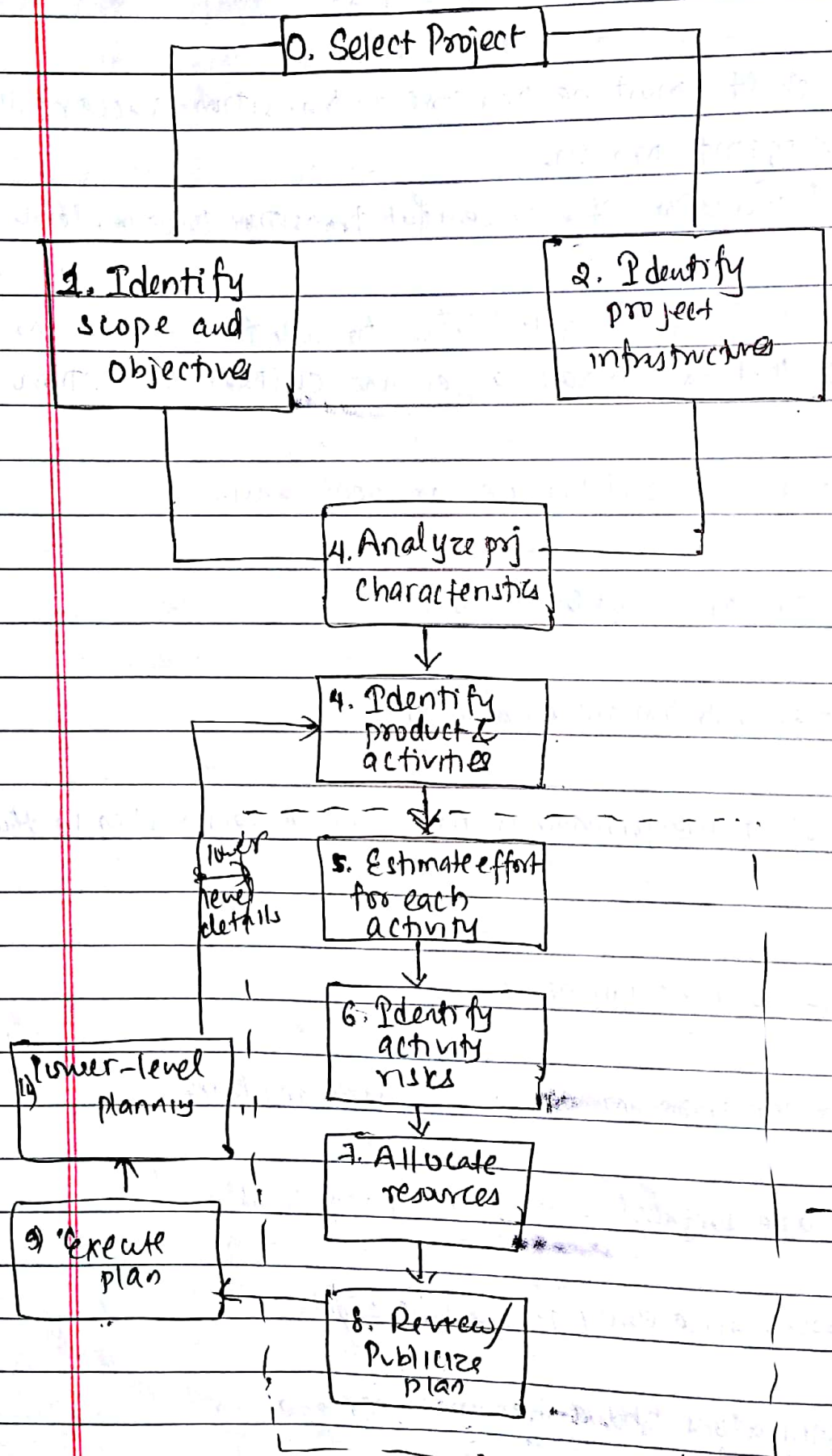
⑥ Artifacts are len important early, more important later.

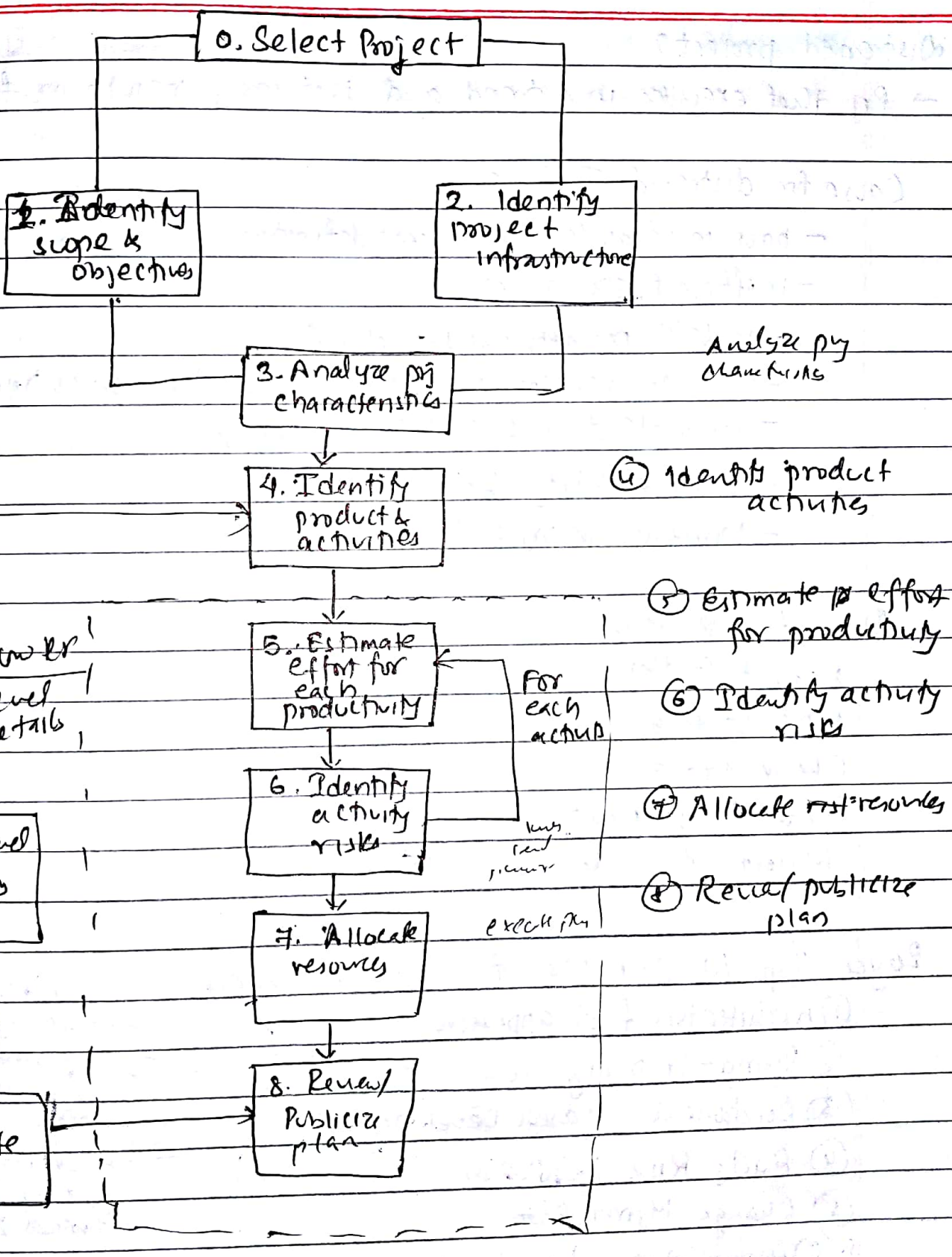
⑦ Real issues are surfaced & resolved systematically

⑧ Performance issues arise early in the life cycle

⑨ Good s/w organizations should be more profitable

Stepwise Project Planning




0. Select Project
1. Identify scope & objectives
2. Identify project infrastructure
3. Analyze proj characteristics
4. Identify product & activities
5. Estimate effort for each productivity
6. Identify activity risks
7. Allocate resources
8. Revise/ Publicize plan
Execute plan
Low-level plannig
lower level details
for each activity
Analyze proj characteristics
(4) Identify product activities
(5) Estimate effort for productivity
(6) Identify activity risks
(7) Allocate resources
(8) Revise/ publicize plan

Distressed project?

→ Prj that executes in a trend and that may result in failure.

Cause for distressed Strategies

- poor, inadequate or no reqmt definitions
- inefficient sponsorship
- lag betn prj approval & kick off
- No plan revisin after significant cuts in resources & time
- Estimate done with little planning.
- No credibility baseline plan
- Unmanageable prj scope.

Prevention strategies

Reqmt Gathering
WBS Strategies
Change Mgmt
Dynamic Risk Asmt
Milestone tracking

Royce Top 10 principal for Modern Proces.

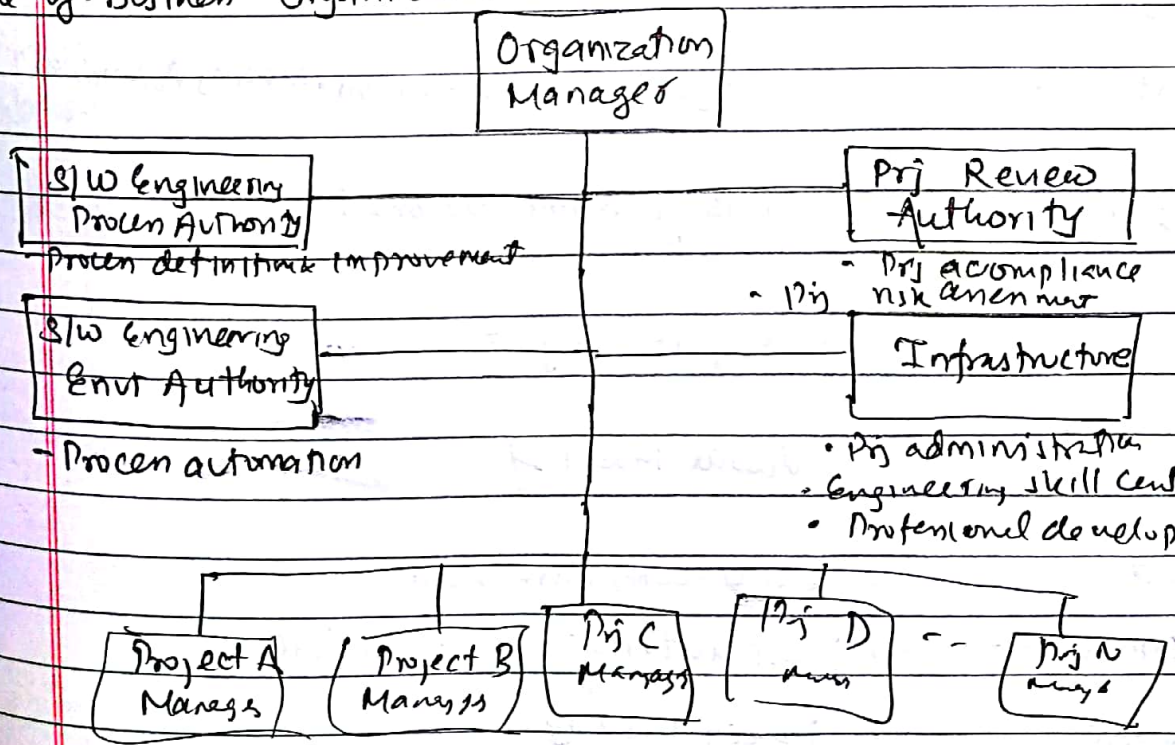
(1) Architecture first approach
(2) Round-Trip Engineering
(3) Component Based Development
(4) Early Risk Resolution
(5) Change Mgmt Envt
(6) Demonstration based approach
(7) Model-Based Notation for artifacts
(8) Instrument procen for obj quality control

→ Model Based notation for artifacts
→ Instrument process for Objective quality control
→ Use configurable process that is scalable for improved Reu.

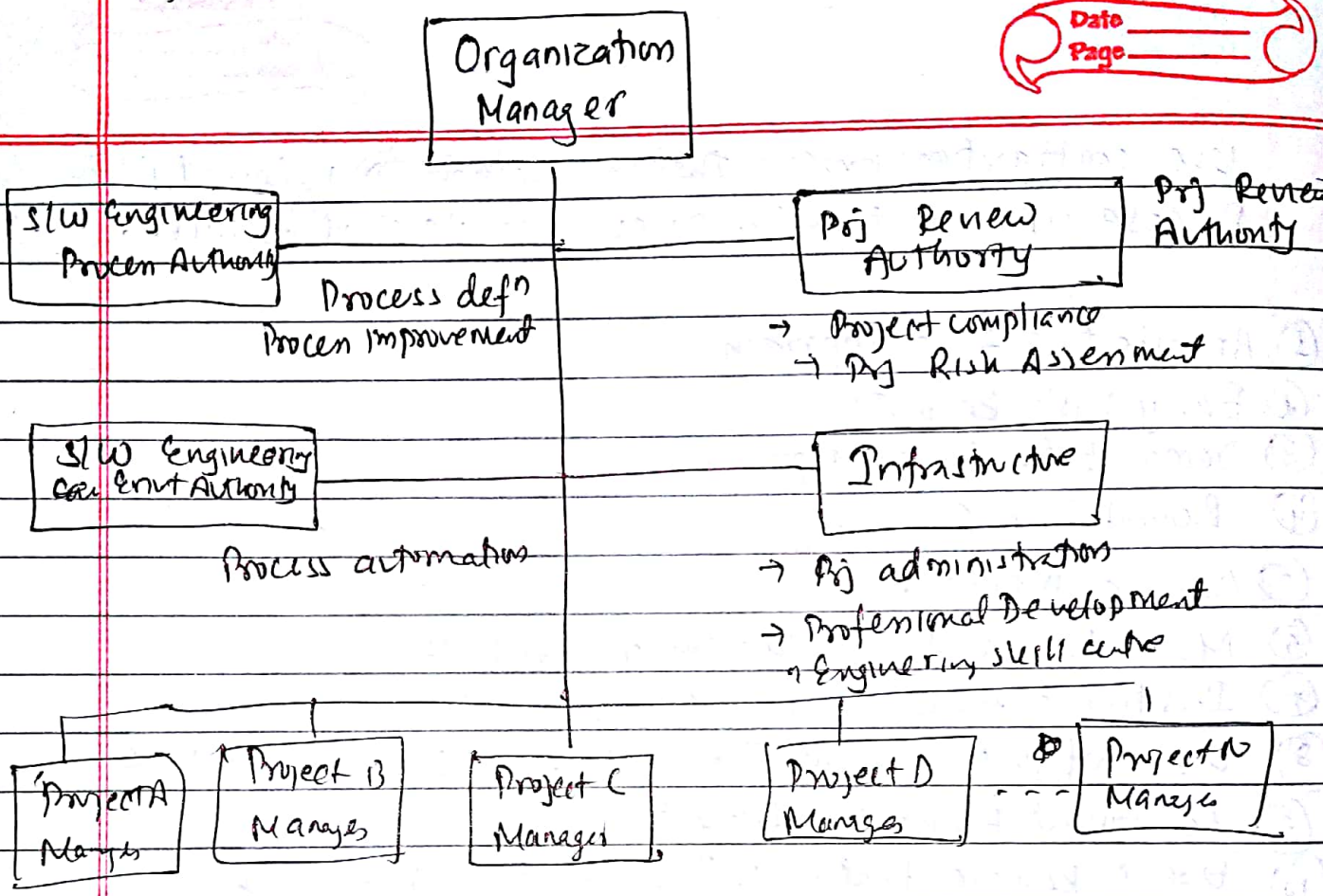
Use configration procen that is scalable for improved ROI
Release iteration ~~throug~~ with evolving levels of details.

1. Architecture first approach
2. Early Risk Resolution
3. Demonstration based approach
4. Round-Trip Engineering
5. Change Mgmnt Envt
6. Model Based ~~Iterati~~ Notation for artifacts
7. Instrmnt procen for objective quality control
8. Use configration control that is scalable for improved ROI
9. Component Based Development
10. ~~Use~~ Release iteration with evolving level of details,

Line-of-Busines Organization



Line of Business Organization



Workflows

Mgmt ——→ Workflow Automation, Metrics Automation

Reqmt. Envt ——→ Change Mgmt, Document Automation

Reqmt ——→ Reqmt Mgmt

Design ——→ Visual Modeling

Implement ——→ Editor-compiler-linker

Assessment ——→ Test automation, Defect tracking

Deployment ——→ Defect tracking

Workflows tools:

Mgmt ⟶ metric & workflow automation

Envt ⟶ ~~test act~~ change mgmt, docu automation.

Reqmt ⟶ reqmt Mgmt

Design ⟶ visual modeling

Implementation ⟶ editor-compiler-linker

Assessment ⟶ defect tracking, test automation

Deployment ⟶ defect tracking

Envt.

RTE
Change mgmt - SCO, CB, CCB.
Infrastructure - organizational (policy & envt)
Stakeholder envt

Iteration Workflows:- Up-to-date risk assessment
Controlled baseline artifacts
Demonstrable results.
- reqmt understanding.
- design artifacts
- plan credibility

The COCOMO Cost Estimation Model.

(Construction Cost Model)

→ ~~com~~ popular, open & well documented cost model.

→ developed by Barry Boehm

→ The COCOMO Estimates equations follow simple form :-

$$\text{Effort} = C_1 \, EAF \, (\text{size})^{P1}$$
$$\text{Time} = C2(\text{Effort})^{P2}$$

/

$$\text{Effort} = C1 \, EAF \, (\text{size})^{P1}$$
$$\text{Time} = C2(\text{Effort})^{P2}$$

Effort = no. of staff months

C1 = constant scaling effcoefficient for Effort

EAF = an effort adjustment factor that characterize domain, touls, personnel, envt used to produce artifact of procen

Size = size of end product, measured by no. of delivered source instructions

C2 = constant scalingcoefficient for Schedule.

Time = total no of months.

P1 = an exponent that characterizes the economies of scale inherent in process used to produce and product, in particular the ability to avoid non-value adding activities

P2 = An exponent that characterizes inertia inherent and parallelism in managing sw dev effort.

COCOMO.

→ original COCOMO Model in 1981.

→ At that time, it proved to be a breakthrough coz of it provided a defined framework for communication of trade-offs & priorities associated with s/w cost & schedule management.

→ Another huge benefit was ability to make estimate, ~~credible reference basis~~ reference credible basis ; reason abt its strengths & weaknesses and negotiate with stakeholders.

→ It was originally made for db of 56 prj.

→ 3 modes reflected to the differences in procen across the s/w domains.

→ Modes were organic, semi-detached and embedded

Organic : Effort = $3.2\ EAF\ (size)^{1.05}$ / $3.2\ EAF\ (size)^{1.08}$
Time ~~Size~~ = $2.5\ (Effort)^{0.38}$ / $2.5\ (Effort)^{0.38}$

Semidetached : Effort = $3.0\ EAF\ (size)^{1.12}$ / $3.0\ EAF\ (size)^{1.12}$
Time ~~Size~~ = $2.5\ (Effort)^{0.35}$ / $2.5\ (Effort)^{0.35}$

Embedded : Effort = $2.8\ EAF\ (size)^{1.2}$ / $2.8\ EAF\ (size)^{1.2}$
Time ~~Size~~ = $2.5\ (Effort)^{0.32}$ / $2.5\ (Effort)^{0.32}$

COCOMO-II

→ It is an effort performed by USC Center for S/W Engineering with financial & technical help from numerous tech industry affiliates.

→ Objectives of this prj are:

① To develop a S/W cost & schedule estimation model for life-cycle practices of 1990s & 2000s.

② To develop S/W database & tool for support for improvement of cost models.

③ To provide quantitative analysis framework for evaluating S/W technologies & their economic impact.

→ It's strategy is to preserve openness of original COCOMO, tailor it to marketplace, key the i/p & o/p to level of info available & enable model to be tailored to various projects

→ To support this strategy, it defines 3 different model for cost estimation

Prototyping	Early Design	Post Architecture
Inception	Elaboration	Construction & Transition
Low-fidelity estimate	Moderate fidelity estimate	High-fidelity estimates
Rough reqmt	~~Stable reqs~~ well-understood reqmt	Stable reqmt baseline
Arch concept	well-understood arch	Stable arch baseline

Overall cost estimation equation

$$Effort = 2.45 \; E_{ARCH} (Size)^{P} \quad \text{P - process exponent}$$

- E_{ARCH}: product of 7 early design adjustment
- Size: no. of func^{n} pt

classmate
Date
Page

- CMM (Capability Maturity Model)
- provides a well known benchmark for s/w procen maturity
- → defined 5 levels of s/w procen maturity ~~levels~~ with Key pt Areas (KPAs)
- → ① Initial — org with immature procen.
  ② Repeatable - reqmt mgmt, s/w proj planning, s/w proj tracking, quality assurance
  ③ Defined - organizational procen (focus & definition), training program
  ④ Managed. - quality mgmt defect prevention, process measurement & analysis
  ⑤ Optimizing - technology innovation

- goal is to achieve level 3 procen
- A software capability evaluation (SCE) is commonly used to asses an organization's maturity.
- SCE's also determines whether the organization "says what it does or does what it says" by evaluating its s/w procen.
- Key drawbacks - KPA's primary focus on docm artifact of conventional procen; such as design, reqmt, contract etc.
  Only a few KPA actually address the evolving engineering artifact, the level of automation or s/w arch procen.
  The principle of modern procen ~~mgmt~~ are ~~as~~ not addressed.
- Another drawback → inherent depiction of configuration mgmt & quality assurance.

Level 1 - Initial - Unpredictable high risk
Level 2 - Repeatable - Treading water, surviving
level 3 - Defined - Stable, predictable progress
Level 4 - Managed - Very predictable, trustworthy
Level 5 - Optimizing - Continuously improving

Culture shifts:-

(1) Lower-level & mid-level managers are performers:-
"Pure manager" - not required in organization for 25 or fewer ppl.
The performers who have managed the plan should handle it.
If another takes over, then the proj will be as good as crap.

(2) Requirement & design are fluid & tangible.

(3) Ambitious ~~design~~ demonstration are encouraged
The purpose of early life cycle demonstration is to expose design flaws. Stakeholders mustn't react to those flaws, digression design. Evaluation criteria in early stage is goal not regret. If obstacles not emphasized, new organization will set up future iteration to be less ambitious. Stakeholders must keep pressures to solve the issues in design flaws, o

(4) Good & Bad proj performance is obvious earlier in the life cycle
In iterative dev → success breeds success, failure in early stage are very risky & difficult to turn around
Early phase make or break proj. So best team performs planning & architecture phases. If this done right then ote it progresses nicely

(5) Early increments will be immature
External stakeholders can't expect perfect, unflawed, complete, reliable proj in the specification. Development proj must be held responsible for demonstrate tangible improvements in successive increment. Early flaws is difficult to accept so they must be made improved by later increments. Mgmt & Dev heroics

of problem

(6) Artifacts are less important early, more important later.
It is waste of time to worry abt artifacts until the baseline is achieved that is stable enough to warrant time-consuming analysis of these quality factors

(7) Real issues are surfaced & resolved systematically
Successful proj recognize design & reqmt evolve together, with continuous integration, trade-off etc. On a healthy proj, making progress. it should be easy to differentiate real issue from apparent issues.

(8) Quality assurance is everyone's job not a separate discipline.

(9) Performance issues arises early in the lifecycle.
→ Early performance issues are sign of immature design but a mature design process. Stakeholders are concerned over them.

POKHARA UNIVERSITY

Level: Bachelor | Semester: Fall | Year : 2017
Programme: BE | Full Marks: 100
Course: Network Programming | Pass Marks: 45
Time : 3hrs.

Candidates are required to give their answers in their own words as far as practicable.

The figures in the margin indicate full marks.

Attempt all the questions.

1.	a)	Which transport level protocols will you use to exchange control information (play, pause, rewind, forward, etc.) and real-time audio-video data between client and server in the movie streaming application. Justify your answers.	7
	b)	Explain TCP state transition with a supporting diagram.	8
2.	a)	What are the different socket structures used in Unix system to make system calls such as connect and bind independent of IP versions.	8
	b)	Discuss the use of fork function to develop a concurrent server with the help of pseudo-code?	7
3.	a)	What is I/O multiplexing? Explain the use of select function in the context of I/O multiplexing in detail.	7
	b)	Write a program to create a TCP echo server.	8
4.	a)	What are the socket options? Which functions are used to set and get a value of socket options? Explain them in detail.	8
	b)	Explain the mechanism of passing file descriptor in the Unix System.	7
5.	a)	Explain the Winsock architecture. What are the different Winsock asynchronous database functions?	8
	b)	Differentiate between recv and WSARecv based on their uses, input/output arguments and return values.	7
6.	a)	How do you implement stream communication in Winsock? Describe each step with the help of relevant APIs.	8
	b)	Explain with the help of pseudo-code the use of accept with select such that the accept function doesn't block.	7
7.		Write short notes on: (**Any two**)	2×5
	a)	netstat	
	b)	ifconfig/ipconfig	
	c)	TFTP	

POKHARA UNIVERSITY

Level: Bachelor | Semester:Spring | Year : 2017
Programme:BE | Full Marks: 100
Course: Software Project Management | Pass Marks: 45
Time : 3hrs.

Candidates are required to give their answers in their own words as far as practicable.

The figures in the margin indicate full marks.

Attempt all the questions.

1.	a)	Recent flood in Nepal had very devastating effect. If you have been asked to develop a software project. What would be your plan, methodology and process in developing it? What are its constraints? How would you deal it?	10
	b)	Define conventional software management. How is it differ from modern software management? Explain it with example.	5
2.	a)	List down the basic parameters required for software cost model. Explain any two basic parameters briefly.	3+5
	b)	Compare & contrast software process workflow & iteration workflow.	7
3.	a)	Explain the different stages of Life-Cycle Phases. Also explain the evaluation criteria for the Life-Cycle Phases.	5+3
	b)	Explain different Perspective of Model-Based Software Architectures.	7
4.	a)	Who are the important stakeholders in software projects? Identify.	8
	b)	What is WBS? Explain the various approaches for building a WBS with appropriate example.	2+5
5.	a)	Define CASE. How is it helpful in process automation? Justify.	2+6
	b)	List the points for software management based practices.	7
6.	a)	Explain the reasons behind the following assertion: "Adding more manpower to a late project makes it later".	8
	b)	Explain the Next-Generation Cost Models with their advantages and disadvantages.	4+3
7.	Write short notes on: (**Any two**)		2×5
	a)	Modern Process Transition	
	b)	Iterative process	
	c)	Status Monitoring	

GANDAKI COLLEGE OF ENGINEERING AND SCIENCE

Level: Bachelor | **Year: 2017**

Program: BE | **Full Marks: 100**

Course: Network Programming | **Time: 3 hrs**

Pass Mark: 45

Candidates are required to give their answers in their own words as far as practicable. The figures in the margin indicate full marks.

Attempt all the questions.

1. a) What are the significances of protocol in communication? Compare TCP/IP with OSI reference model. 8

   b) Explain the various TCP state with suitable diagram. 7

2. a) Describe about UNIX domain communication protocols. How can we use socket pair function for client server application? 8

   b) Describe the asynchronous and synchronous I/O models with suitable diagrams. 7

3. a) Explain about the different options which will affect the common socket. 8

   b) Explain the fcntl () and ioctl () functions. 7

4. a) Explain the procedure steps for crashing, rebooting and shutdown of server hosts. 8

   b) What do you mean by Winsock architecture? Briefly explain the role of main dll files available in Winsock. 7

5. a) Explain the purpose and usage of Windows UDP sockets and their different function.

   b) Explain windows socket extension functions and their significance.

   a) Describe the methods for handling blocked windows socket system calls.

   b) Briefly explain about the Remote login and Netstat.

Write short notes on *(Any Two)*

a) Asychronous database functions.

b) Daemon process

c) SO_LINGER socket options.