Unit 1: Software Management Practice and Software Economics

B.E. Software 8th Semester

Gandaki College of Engineering & Science

Lamachaur, Pokhara

Project

- An individual or collaborative enterprise that is carefully planned to achieve a particular aim.
- A project is a series of tasks that need to be completed in order to reach a specific outcome.
- A project can also be defined as a set of inputs and outputs required to achieve a particular goal.
- Projects can range from simple to complex and can be managed by one person or a hundred.

Project Definition:

- The dictionary definitions put a clear emphasis on the project being a *planned* activity.
- Planning is in essence thinking carefully about something before you do it.
- Planning defines how we are going to carry out the given activities.
- Set of interrelated tasks to be executed over a fixed period and within certain costs and other limitations.

What is a Project?

- A project is a temporary endeavor undertaken to create a unique product or service.
- One time (non-routine)
- Limited funds/time (May be single phase or multiple phases)
- Specific resources utilized (May involve several specialism)
- Performed by people- Single or multi-person & multi location team(s)
- Planned, controlled (planning, monitoring and controlling is required)
- Specific Deliverables (specific objectives are to be met or a specified product is to be created)

Attributes of Projects

- Unique purpose
- Temporary
- Require resources, often from various areas
- Should have a primary sponsor and/or customer
- Involve uncertainty

Characteristics of Projects

- Non-routine tasks are involved
- Planning is required
- Specific objectives are to be met or a specified product is to be created
- The project has a pre-determined time span
- Work is carried out for someone other than yourself
- Work is carried out in several specialism
- People are formed into a temporary work group to carry out the task
- Work is carried out in several phases
- The resources that are available for use on the project are constrained
- The project may be large or complex.

Examples of Project

- Planning a large reception or an event, that is a project.
- This is because, it was a specific reception for a specific reason and it was held on a specific date and time.
- That means reception was unique, temporary, and it had a defined beginning and end, and reception created a specific product or service to invited persons.

Jobs versus projects

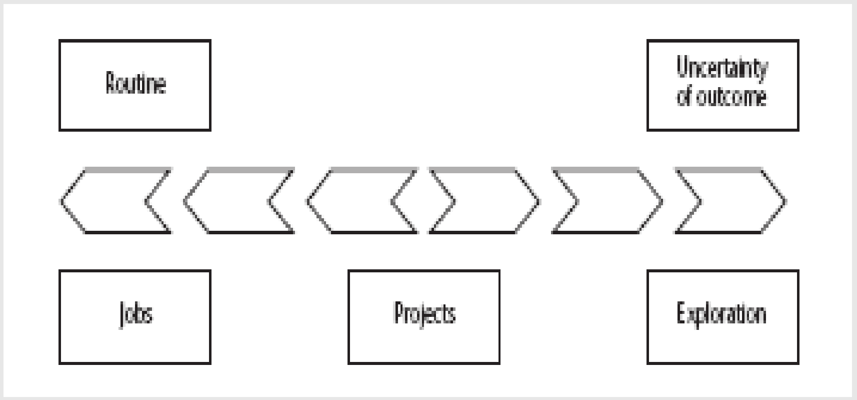


Figure 1.1 Activities most likely to benefit from project management

‘Jobs’ – repetition of very well-defined and well understood tasks with very little uncertainty

‘Exploration’ – e.g. finding a cure for cancer: the outcome is very uncertain

‘Projects’ – in the middle!

Classification of Projects

- There are many ways to classify a project such as:
- By size (cost, duration, team, business value, number of departments affected, and so on)
- By type (new, maintenance, upgrade, strategic, tactical, operational)

Software Project

- Software project is collection and integration of various components of the software.
- It begins with early investigation and ends with implementation.

What is Management?

It includes activities such as:

- *Planning* - deciding what is to be done
- *Organizing* - making arrangements
- *Staffing* - selecting right people for the job etc.
- *Directing* – giving instructions
- *Monitoring*- checking on progress
- *Controlling* – taking action to remedy hold-ups
- *Innovating* – coming up with new solutions
- *Representing* – liaising with clients, users, developers, suppliers and other stakeholders

Management Functions

- **Planning**
  - Predetermining a course of action for accomplishing organizational Objectives
- **Organizing**
  - Arranging the relationships among work units for accomplishment of objectives and the granting of responsibility and authority to obtain those objectives
- **Staffing**
  - Selecting and training people for positions in the organization
- **Directing**
  - Creating an atmosphere that will assist and motivate people to achieve desired end results
- **Controlling**
  - Establishing, measuring, and evaluating performance of activities toward planned objectives

The Management Spectrum

- Effective project management focuses on four P’s (in the order):
- **The People:** Stakeholders, the team leaders, and the software team
- Deals with the cultivation of motivated, highly skilled people and teams
- Includes recruiting, selection, performance management, training, compensation, career development, organization and work design, and team culture development
- **The Problem/Product:** before a project can be planned
- Its objectives and scope should be established
- Alternative solutions should be considered, and
- Technical and management constraints should be identified.
- **The Process:** A software process provides the framework from which a comprehensive plan for software development can be established.
- **The Project:** Planning and controlling a software project is done for one primary reason…. It is the only known way to manage complexity.

Project Management Skills

- Leadership
- Communications
- Problem Solving
- Negotiating
- Influencing the Organization
- Mentoring
- Process and technical expertise

Software Project Management

- Software project management is aimed to ensure that the software is delivered on time, within budget and schedule constraints, and satisfies the requirements of the client.
- Management of software projects is different from other types of management because:
  - Software is not tangible
  - Software processes are relatively new and still “under trial”
  - Larger software projects are usually “one-off” projects
  - Computer technology evolves very rapidly

The Triple Constraint

- Every project is constrained in different ways by its
  - Scope goals: What is the project trying to accomplish?
  - Time goals: How long should it take to complete?
  - Cost goals: What should it cost?
- It is the project manager’s duty to balance these three often competing goals

Software projects versus other types of project

- Many of the techniques of general project management are applicable to software project management,
- Certain characteristics make them different. The characteristics are:
  - Invisibility
  - Complexity
  - Conformity
  - Flexibility

Software projects versus other types of project

- **Invisibility**
  - With physical artifact such as a bridge or road is being constructed the progress being made can actually be seen.
  - However with software, progress is not immediately visible.
- **Complexity**
  - Software products are generally, more complex than other engineering artifact of same value.
- **Conformity**
  - Software developers have to conform to the requirements of human clients.
  - However, the requirements of the human clients keeps on fluctuating..
- **Flexibility**
  - It is easier to change/modify software systems to meet changing organizational/product requirement as compared to other engineering artifacts; it may not be possible to modify a physical artifact at all.
  - This means the software systems are likely to be subject to a high degree of change.

Activities covered by software project management

- A software project is not only concerned with the writing or developing of the software.
- It includes the following activities:
  - The feasibility study
  - Planning
  - Project execution

The feasibility Study

- This is an investigation into whether a prospective (probable) project is worth starting.
- Information is gathered about the requirements of the proposed application.
- The probable developmental and operational costs, along with the value of the benefits of the new system, are estimated.

Planning

- If the feasibility study produces results which indicate that the prospective project appears viable, planning of the project can take place.
- However, for a large project, we would not do all our detailed planning right at the beginning.
- We would formulate an outline plan for the whole project and a detailed one for the first stage.
- More detailed planning of the later stages would be done as they approached.
- This continues till the end of the project.

Project Execution

- The project can now be executed.
- The execution of a project often contains *design* and *implementation* sub-phases.

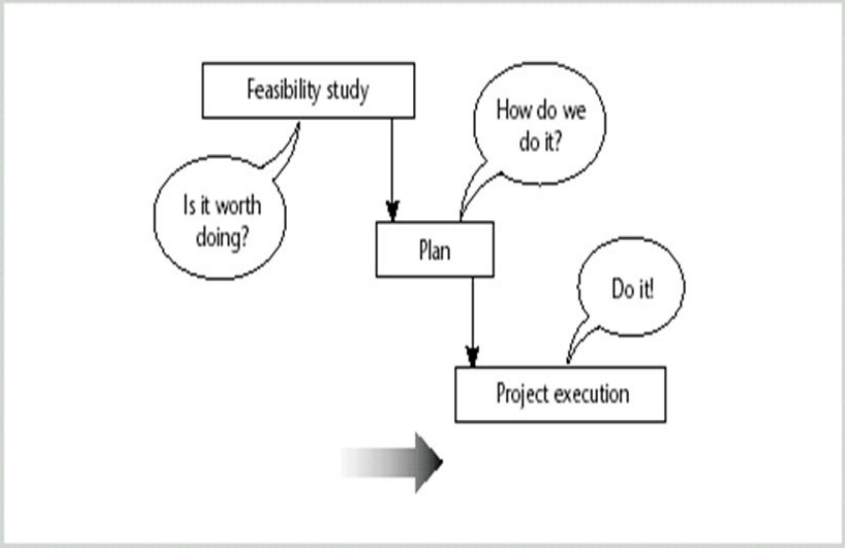


Figure 1.2 The feasibility study/plan/execution cycle

A Classic Project Life Cycle

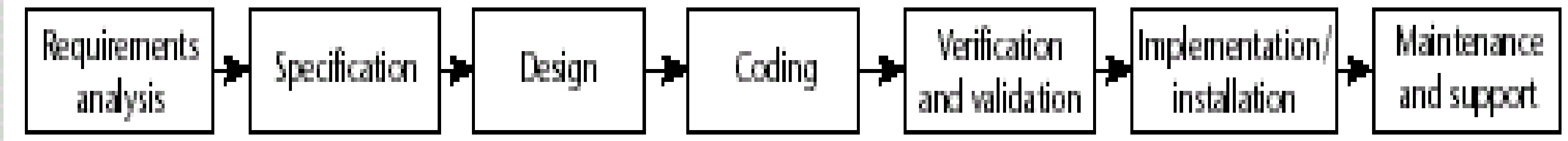


Figure 1.3 A typical project life cycle

Requirements Analysis

- This is finding out in detail what the users require of the system that the project is to implement.
- Original information obtained needs to be updated and supplemented.
- Several different approaches to the users' requirements may be explored.

Specification & Design

Specification

- Detailed documentation of what the proposed system is to do.

Design

- A design has to be drawn up which meets the specification.
- Design will be in two stages.
    i. One will be the external or user design concerned with the external appearance of the application eg: menus, forms, reports etc.
    ii. The other produces the physical design which tackles the way that the data and software procedures are to be structured internally.

Coding & Verification and Validation

Coding

- This may refer to writing code in some programming language.
- Even where software is not being built from scratch, some modification to the base package could be required to meet the needs of the new application.

Verification and validation

- Whether software is developed specially for the current application or not, careful testing will be needed to check that the proposed system meets its requirements.

Implementation/installation

- Some system development practitioners refer to the whole of the project after design as ‘implementation’ (that is, the implementation of the design)
- while others insist that the term refers to the installation of the system after the software has been developed.
- In this latter case it includes setting up operational data files and system parameters, writing user manuals and training users of the new system.

Maintenance and Support

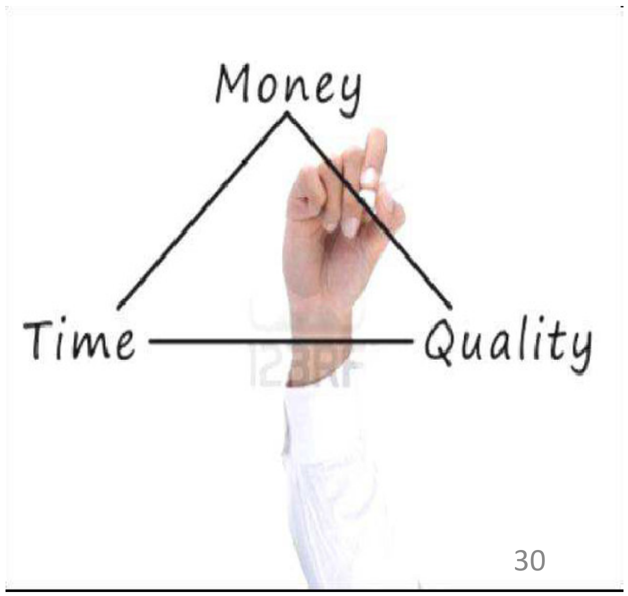
- Once the system has been implemented there is a continuing need for the correction of any errors that may be into the system and for extensions and improvements to the system.
- Maintenance and support activities may be seen as a series of minor software projects.
- In many environments, most software development after implementation is in fact maintenance.

Problems with Software Projects

- **Project Manager:**
  - poor roles definition,
  - lack of estimating & planning skills and
  - lack of decision making skills.
  - Schedule, budget and quality constraints.
- **Developers :**
  - lack of knowledge in the application area,
  - lack of knowledge about developing standards,
  - lack of up to date documentations
  - deadline pressure
  - changes of application requirements.
- **Customers:**
  - monetary constraints
  - receiving products past the due date
  - surprises

Introduction to Project Plan

- The key to a successful project is in the planning.
- Planning is the first thing that is done when one undertakes a project.
- Often project planning is ignored to rush in for the work.
- The value of project planning is
- saving money.
- saving time.
- quality output.
- Note: If you fail to plan, you plan to



Essential Elements For Project Planning

- **Aim of project**
  - What do we want to produce?
- **Outputs**
  - What do we actually need to get there?
- **Quality criteria**
  - What is the quality of the output?
  - We need the completed output to be of certain quality and we need to define what that quality is (we define it using the SMART principle: Specific, Measurable, Achievable, Realistic, Timely).

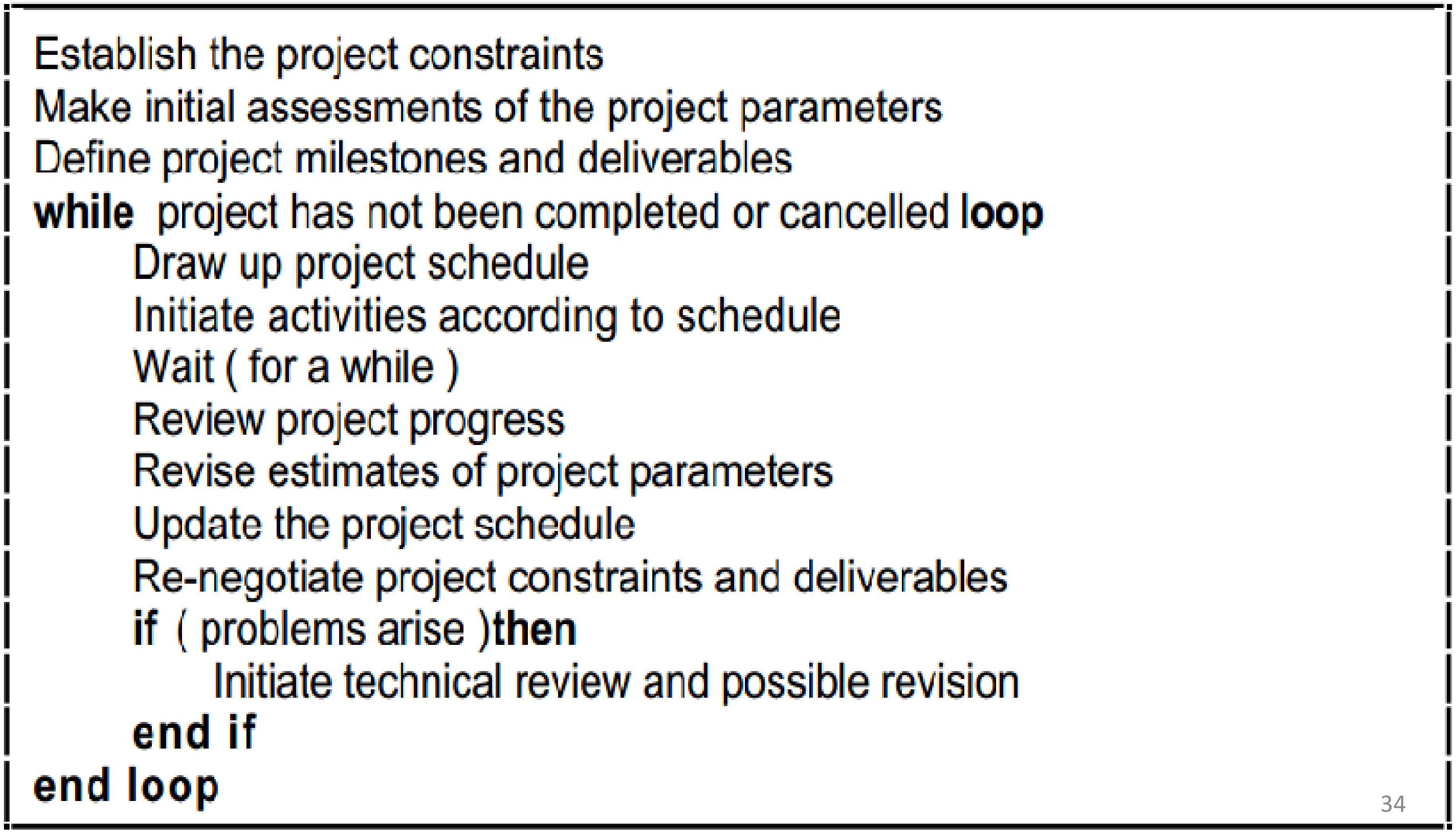
Essential Elements For Project Planning

- **Resources**
  - Includes staff time, particular knowledge or skill sets, money, time.
  - Management structure
  - How are we going to manage the work?
- **Milestones**
  - A defined milestone will help to identify when each section is completed.
- **Tolerances**
  - How far can we let the project stray from the defined targets before sounding the alarm?

Essential Elements For Project Planning

- **Dependencies**
  - Understanding dependencies will help understand the impact of changes in any part of the project.
- **Risks**
  - What could happen that may affect our ability to deliver the project on time?
  - What can we do to avoid them?
- **Scheduling**
  - There is no perfect schedule.
  - Schedule is not engraved on stone. One should expect changes .

Project Planning Process



```
Establish the project constraints
Make initial assessments of the project parameters
Define project milestones and deliverables
while project has not been completed or cancelled loop
        Draw up project schedule
        Initiate activities according to schedule
        Wait ( for a while )
        Review project progress
        Revise estimates of project parameters
        Update the project schedule
        Re-negotiate project constraints and deliverables
        if ( problems arise )then
              Initiate technical review and possible revision
        end if
end loop
```

Steps in Planning

- Specification
- Project Goals
- Global Structure
- Project Breakdown
- Task Delegation
- Time Estimation
- Supporting Plans
- Setting Controls

Specification

- A written definition of requirements and deadlines.
- Should be clear, complete and could eliminate misunderstandings, contradictions of technical difficulties.

Project goals

- Identify the stakeholders of the project.
- Stakeholder is any body impacted directly or indirectly by the project.
- Establish their needs by interviewing or having consolidated meetings.
- Prioritize stakeholders needs.
- Create a set of goals that can be easily measured (you may use the SMART technique for this)

S – specific

M – measurable

A – achievable

R – realistic

T – timely

Global Structure

- Includes tasks that must be accomplished.
- Who will do what?
- When will it be done?

Project Breakdown

- Break project down into a series of task.
- Break each task down into subtasks.
- Continue until all items are doable and understandable.

Task Delegation

- Assign tasks to specific people (or teams).
- Order tasks so that they occur in a logical sequence.
- Match tasks to abilities of the team.

Time Estimation

- Times are based on previous experience.
- They are always wrong so one must plan accordingly.
- Example

How long should it take you to climb the statue of Liberty?

- Estimate the number of steps.
- Estimate the time per steps.

Supporting plans

- **Human resource plan**

- Identify by name the individuals with a leading role in the project and describe roles and responsibilities.
- Describe the number and type of people needed to carryout the project.
- Include SME’s (subject matter experts) and specific trades of the market.
- The above will help establishing the project budget

Supporting plans

Communications plan

- Who needs to be informed about the project?
- How will they receive the information ?

- Weekly/monthly progress reports to include performance, status, milestones achieved, work planned for next periods etc.

Supporting plans

Risk Management Plan

- Identify as many risks as possible.
- Be prepared if something bad happens.

Common project risks

- Time and costs estimates too optimistic.
- Unexpected budget cuts.
- Scope changes.
- Atmospheric events.

Types Of Project Plan

Quality plan

- Describes the quality procedures and standards that will be used in a project.

Validation plan

- Describes the approach, resources and schedule used for system validation.

Configuration management plan

- Describes the configuration management procedures and structures to be used.

Types Of Project Plan

Maintenance plan

- Predicts the maintenance requirements of the system, maintenance costs and effort required.

Staff development plan.

- Describes how the skills and experience of the project team members will be developed.

Software Project Management Framework

- Project management framework contains sets of processes, tools, techniques and templates designed to be used together to manage a project through its life cycle.
- It ensure that all major development processes will be under control.
- To achieve all development process must be continuously monitored.

Software Project Management Framework

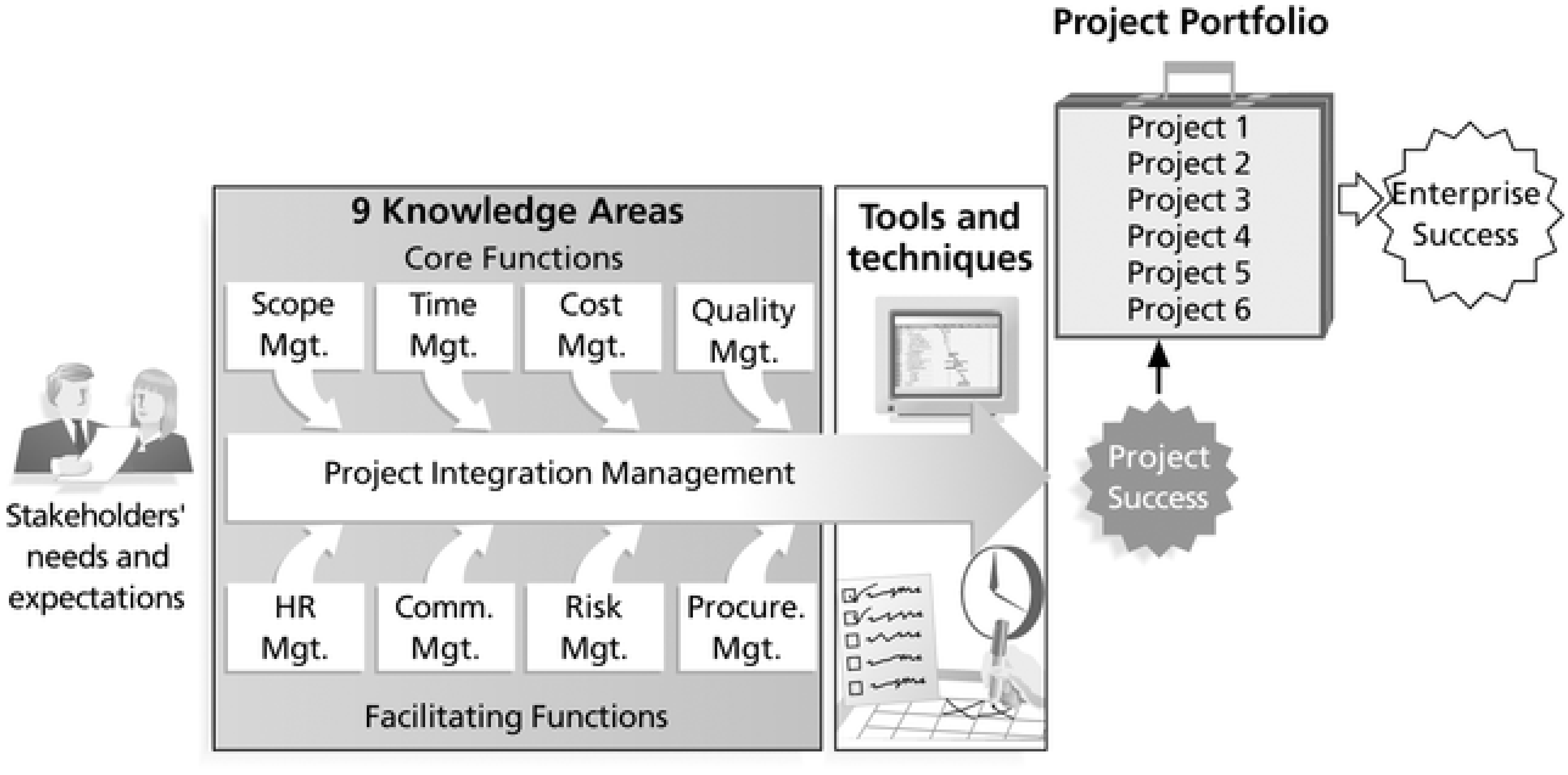


Figure 1-2. Project Management Framework

Project Stakeholders

- Stakeholders are the people involved in or affected by project activities
- Stakeholders include
  - the project sponsor and project team
  - support staff
  - customers
  - users
  - suppliers
  - opponents to the project

Potential for Conflict

- Resource conflict
  - Interdependences
    - Activities, Projects, Projects and Operations
  - Limited Resources
    - People, Equipment, Time, Money, Facilities
- People conflict
  - As a result of resource conflict
  - Resistance to Change
- Project Manager
  - Must be a Conflict Manager

Potential for Conflict (cont..)

- Client
  - Max Flexibility, Max Quality at Min Price in Min Time
- Organization
  - Max Profit, Min disruption to Operations
- Public
  - Obeisance of all relevant Government Regulations
  - Min Environmental Impact

Negotiation & Conflict resolution

Two different types of negotiations

- win-lose
  - your savings are other party's losses

- win-win
  - both parties try to understand the other party needs
  - The win-win approach is a set of principles and practices which enable a set of Interdependent stakeholders to work out a mutually satisfactory (win-win) set of shared commitments.

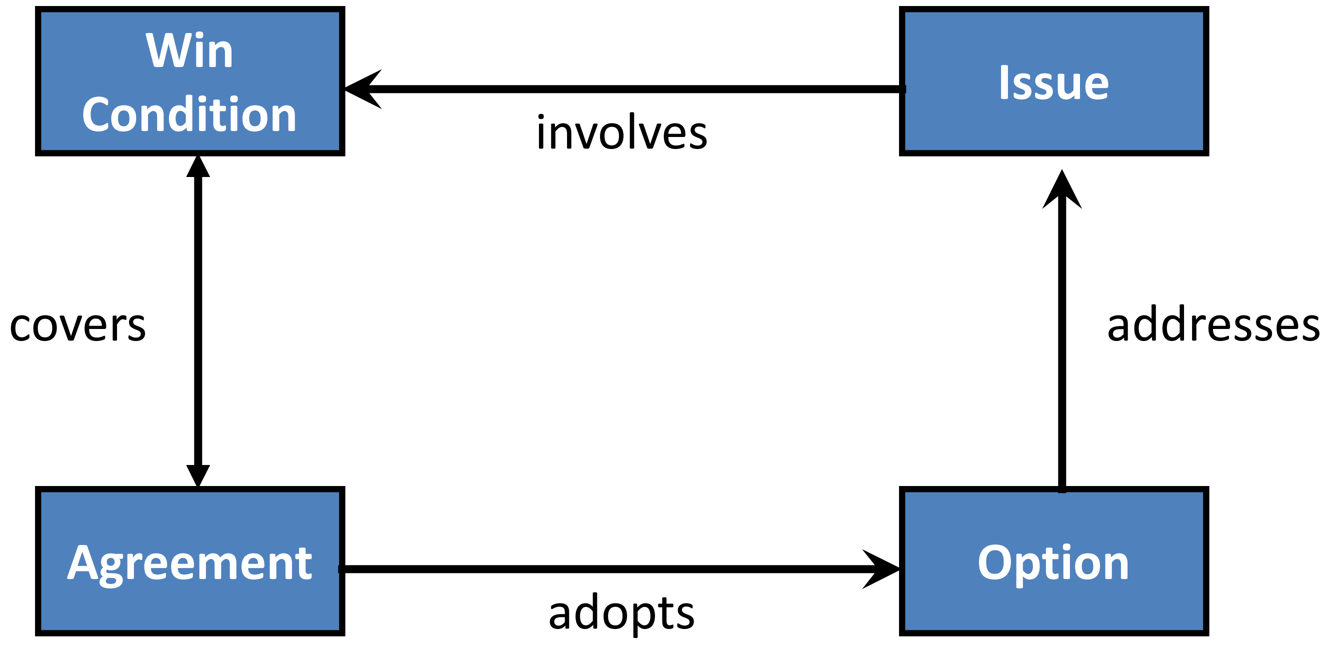
Negotiation & Conflict resolution (cont..)

- **Win-lose Generally Becomes Lose-lose**

Proposed Solution	**"Winner"**	**"Loser"**
Quick, cheap & Sloppy product	Developer & Customer	User
Lots of "Bells & whistles "	Developer & User	Customer
Driving too hard a bargain	Customer & User	Developer

- **Actually, nobody wins in these situations**

WinWin Negotiation Model



- **WinWin Equilibrium State**
  - **All Win Conditions covered by Agreements**
  - **No outstanding Issues**

Nine Project Management Knowledge Areas

- Knowledge areas describe the key competencies that project managers must develop.
  - Four core knowledge areas lead to specific project objectives (scope, time, cost, and quality).
  - Four facilitating knowledge areas are the means through which the project objectives are achieved (human resources, communication, risk, and procurement management).
  - One knowledge area (project integration management) affects and is affected by all of the other knowledge areas.
  - All knowledge areas are important!

9 PM Knowledge Areas (cont..)

- Project Integration Management
  - Describes the processes required to ensure that various elements of the project are properly coordinated. It consist of project plan development, project execution and integrated change control.
- Project Scope Management
  - Describes the processes required to ensure that the project includes all the work required and integrated change control.
- Project Time Management
  - Describes the processes required to ensure the timely completion of the project. It consist of initiation, scope planning, scope definition, scope verification and scope change control.

9 PM Knowledge Areas (cont..)

- Project Cost Management
  - Describes the processes required to ensure that the project is completed within the approved budget. It consist of resource planning, cost estimating, schedule development and schedule control.
- Project Quality Management
  - Describes the processes required to ensure that the project will satisfy the needs for which it was undertaken. It consists of quality planning, quality assurance and quality control.
- Project Human Resource Management
  - Describes the processes required to make the most effective use of the people involved with the project. It consist of organizational planning, staff acquisition and team development.

9 PM Knowledge Areas (cont..)

- Project Communication Management
  - Describes the processes required to ensure timely and appropriate generation, collection, dissemination, storage and ultimate disposition of project information. It consist of communications planning, information distribution, performance reporting and administrative closure,
- Project Risk Management
  - Describes the processes concerned with identifying, analyzing and responding to project risk. It consist of risk management planning, risk identification, qualitative risk analysis, quantitative risk analysis, risk response planning and risk monitoring and control.
- Project Procurement Management
  - Describes the processes required to acquire goods and services from the outside the performing organization. It consist of procurement planning, solicitation planning, solicitation, source selection, contract administration and contract closeout

A Hierarchy of Activities

WBS (Work Breakdown Structure)

The backbone of any project is WBS. It describes the steps necessary to carry out the project and their relationship to each other, Not easy and straight forward.

System:

(IEEE) collection of components organized to accomplish a specific function or set of function.

Program > Project > Work Package > Task > Work Unit

- Program
  - A Group of Related Projects that is managed together, Programs usually include an Element of Ongoing activity
  - An exceptionally large, long range objective that can be broken into projects
  - E.g. Govt. of Pakistan Poverty reduction program

A Hierarchy of Activities (cont..)

Program > Project > Work Package > Task > Work Unit

- Projects
  - A temporary endeavor undertaken to create a unique product or service
  - A specific finite task
- Work Package
  - Project major set of activities / Modules
  - Each WP has clear set of Objectives, Task and deliverables
  - Each WP must have WP leader
  - One continuous set of work units with a clearly defined and observable beginning and end

A Hierarchy of Activities (cont..)

Program > Project > Work Package > Task > Work Unit

- Activity
  - An element of work performed during the course of a project. An activity has an expected duration, cost and resource requirement
  - Any Task, Job or Operation that must be completed to finish a project
  - Synonym for Task
  - Must Result in a Tangible deliverable
- Task
  - A subdivision of an activity
  - Synonym for activity
  - Task must have one or more responsible person
- Work Unit
  - Subdivision of a work package
  - Not recognized as a term

Project Management Tools and Techniques

- Project management tools and techniques assist project managers and their teams in various aspects of project management
- Some specific ones include
  - Project Agreement, scope statement, and WBS (work breakdown structure) (scope)
  - Gantt charts, network diagrams, critical path analysis, critical chain scheduling (time)
  - Cost estimates and earned value management (cost)

Project Portfolio Management

- Many organizations support an emerging business strategy of **project portfolio management**:

- Organizations group and manage projects as a portfolio of investments that contribute to the entire enterprise’s success.

SUCCESS/FAILURE PROFILES

- The most important aspect of the research is discovering why projects fail.
- To do this, The Standish Group surveyed IT executive managers for their opinions about why projects succeed.
- The three major reasons that a project will succeed are user involvement, executive management support, and a clear statement of requirements.
- There are other success criteria, but with these three elements in place, the chances of success are much greater. Without them, chance of failure increases dramatically.

Why the Improvements?

"The reasons for the increase in successful projects vary. First, the **average cost** of a project has been more than cut in half. Better **tools** have been created to monitor and control progress and **better skilled project managers with better management processes** are being used. The fact that there are **processes** is significant in itself."*

What the Winners Do

- Companies that excel in project delivery capability:
  - Use an **integrated project management toolbox** that includes standard and advanced tools and lots of templates.
  - Grow project **leaders**, emphasizing **business and soft skills.**
  - Develop a **streamlined project delivery process.**
  - Measure project health using **metrics**, including customer satisfaction and return on investment.

Categorizing Software Projects:

Compulsory Vs Voluntary systems (projects):

- **Compulsory systems** are the systems which the staff of an organization have to use if they want to do a task.
- Voluntary systems are the systems which are voluntarily used by the users eg. computer gaming etc.

Information Vs Embedded systems (projects):

- Information systems are used by staff to carry out office processes and tasks eg. stock control system.
- Embedded systems are used to control machines eg. a system controlling equipment in a building.

Advantages of Using Formal Project Management

- Better control of financial, physical, and human resources
- Improved customer relations
- Shorter development times
- Lower costs
- Higher quality and increased reliability
- Higher profit margins
- Improved productivity
- Better internal coordination
- Higher worker morale

Why Projects Fail?

- Not enough resources
- Not enough time
- Unclear specifications
- Changes in scope
- Disagreement among stakeholders
- Bad plan
- Lack of project management

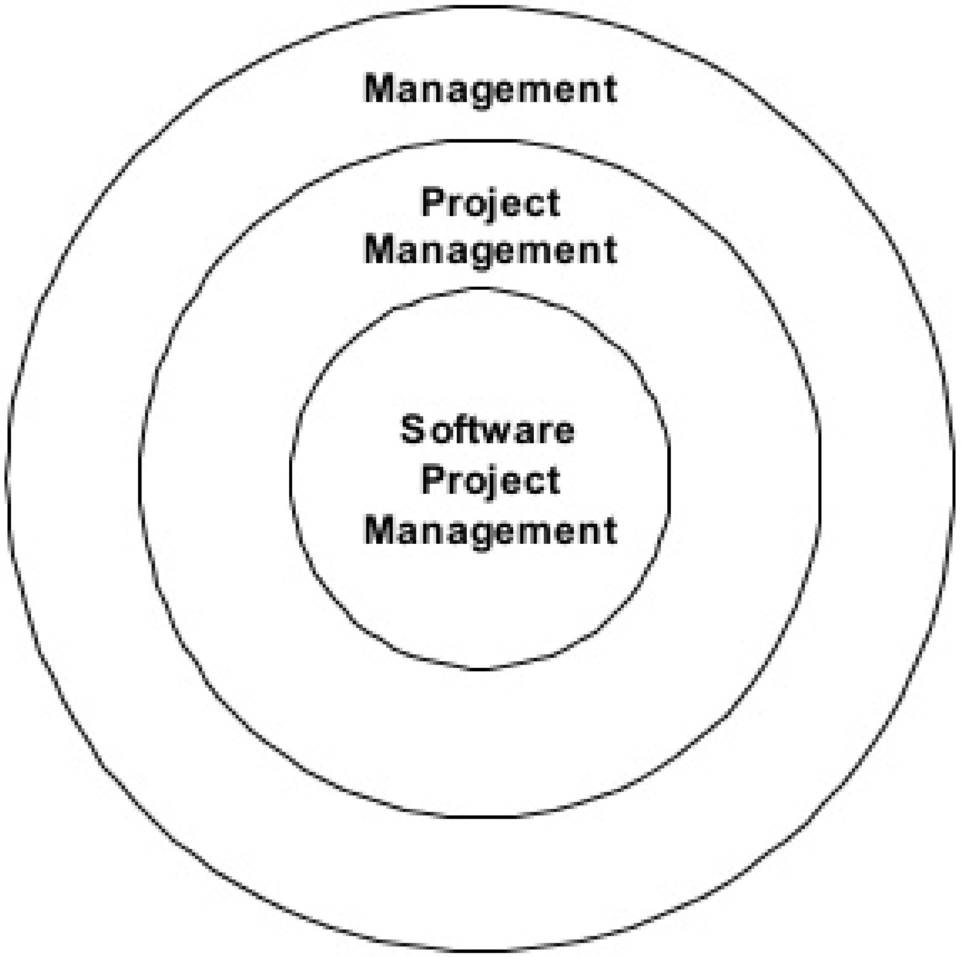
Software Project

- A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.

Has Four P'S

- **People: -**
  - Organized and effective, highly motivated and coordinated team to do quality work & to achieve effective communication.
- **Product: -**
  - Product requirements from customer, portioned and position for work.
- **Process: -**
  - Roadmap adopted to the people & problem.
- **Project: -**
  - Organized to get succeed.

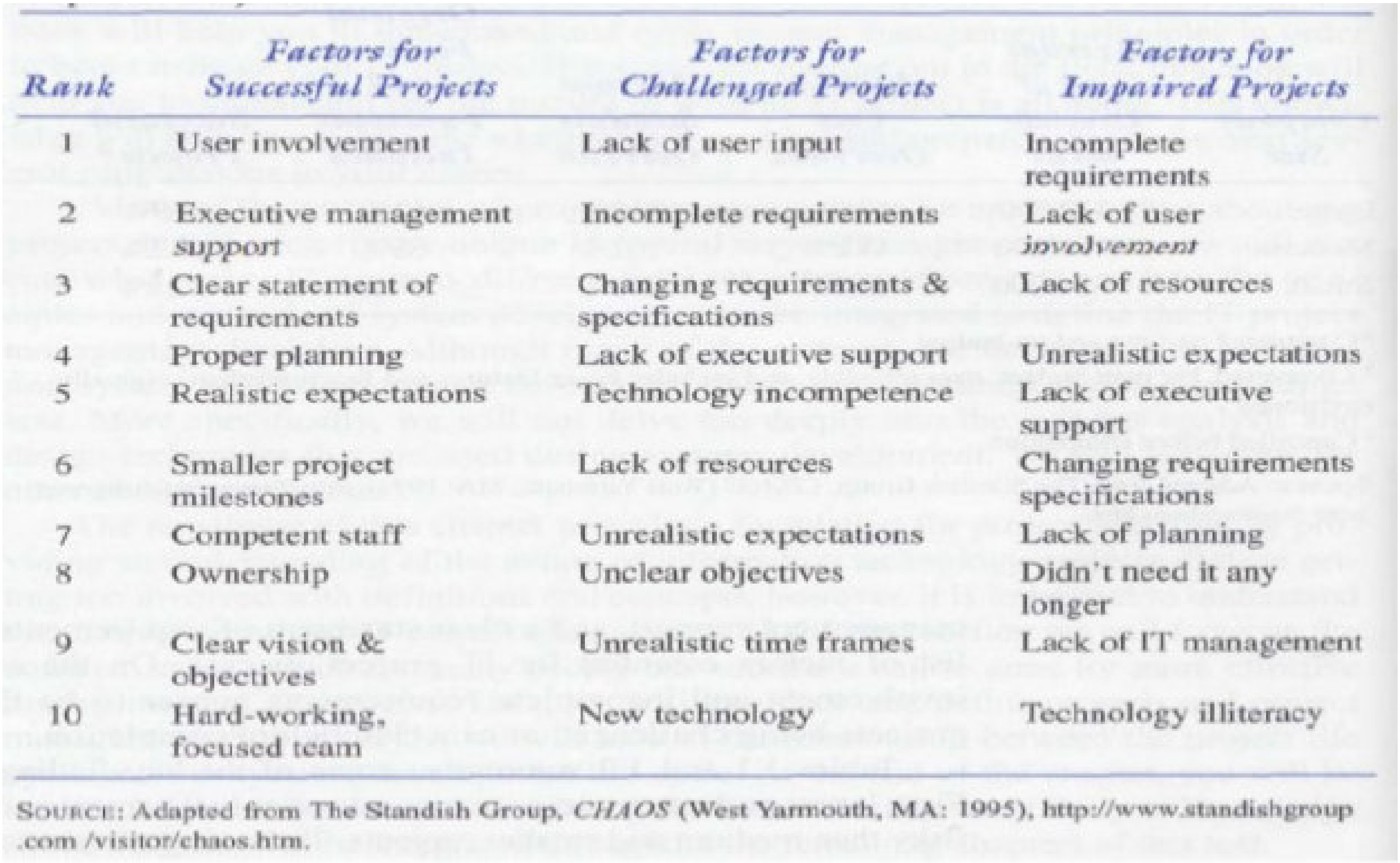
Software Project Management



Project Success Factors

1. Executive support
2. User involvement
3. **Experienced project manager**
4. Clear business objectives
5. Minimized scope
6. Standard software infrastructure
7. Firm basic requirements
8. Formal methodology
9. Reliable estimates
10. Other criteria, such as small milestones, proper planning, competent staff, and ownership

Factors Affecting Software Projects



Rank	Factors for Successful Projects	Factors for Challenged Projects	Factors for Impaired Projects
1	User involvement	Lack of user input	Incomplete requirements
2	Executive management support	Incomplete requirements	Lack of user involvement
3	Clear statement of requirements	Changing requirements & specifications	Lack of resources
4	Proper planning	Lack of executive support	Unrealistic expectations
5	Realistic expectations	Technology incompetence	Lack of executive support
6	Smaller project milestones	Lack of resources	Changing requirements specifications
7	Competent staff	Unrealistic expectations	Lack of planning
8	Ownership	Unclear objectives	Didn't need it any longer
9	Clear vision & objectives	Unrealistic time frames	Lack of IT management
10	Hard-working, focused team	New technology	Technology illiteracy

SOURCE: Adapted from The Standish Group, *CHAOS* (West Yarmouth, MA: 1995), http://www.standishgroup.com /visitor/chaos.htm.

Software Project Management

- It is a process of managing, allocating and timing resources to develop computer software that meets requirements.
- In Software Project Management, the end users and developers need to know the length, duration and cost of the project.
- Software Project Management is a sub-discipline of project management in which software project are planned, implemented, monitored and controlled.

Software Project Management

- It consists of eight tasks:
  - Problem Identification
  - Problem Defining
  - Project Planning
  - Project Organizing
  - Resource Allocation
  - Project Scheduling
  - Tracing, Reporting and controlling
  - Project Transmission
- Advantages of SPM: Easy manage a company's projects; Accessibility and Cost

- So, Software Project Management is the art and science of planning and leading Software Project.
- In Problem Identification and Definition, the decisions are made while approving, declining or prioritizing projects.
- In Problem Identification, project is identified, defined and justified.
- In Problem Definition, the purpose of the project is clarified. The main product is project proposal.
- In Project Planning, it describes a series of actions or steps that are needed to for the development of work product.

- In Project Organization, the functions of the personnel are integrated. It is done in parallel with project planning.
- So that, Software Project Management is Software that has used for Project Planning, Scheduling, Resource Allocation and Change Management.
- The important of SPM – is a kinds of tools allow companies to become competitive in their environments, optimizing time and effort and keeping project on track.
- The main objective of SPM is to deliver an information system that is acceptable to users and is developed on time and within budget.

- So, Software Project Managements is an umbrella activity within Software Engineering
- Project Management involves the Planning, Monitoring and Control of People, Process and Events that occur as Software evolves from preliminary concept to an operational implementation.
- Project Management begins before any technical activity and continues throughout the Definition, Development and Support of Computer Software.

Conventional Software Management Theory and Practice

- The best thing about software is its flexibility:
  - It can be programmed to do almost anything.
- The worst thing about software is its flexibility:
  - The "almost anything" characteristic has made it difficult to plan, monitor, and control software development.
- In the mid-1990s, three important analyses were performed on the software engineering industry.

- All three analyses given the same general conclusion:-
  - "The success rate for software projects is very low".

Conventional Software Management Theory and Practice

- Software development is still highly unpredictable. Only about 10% of software projects are delivered successfully within initial budget and schedule estimates.
- Management discipline is more of a discriminator in success or failure than are technology advances.
- The level of software scrap and rework is indicative of an immature process.

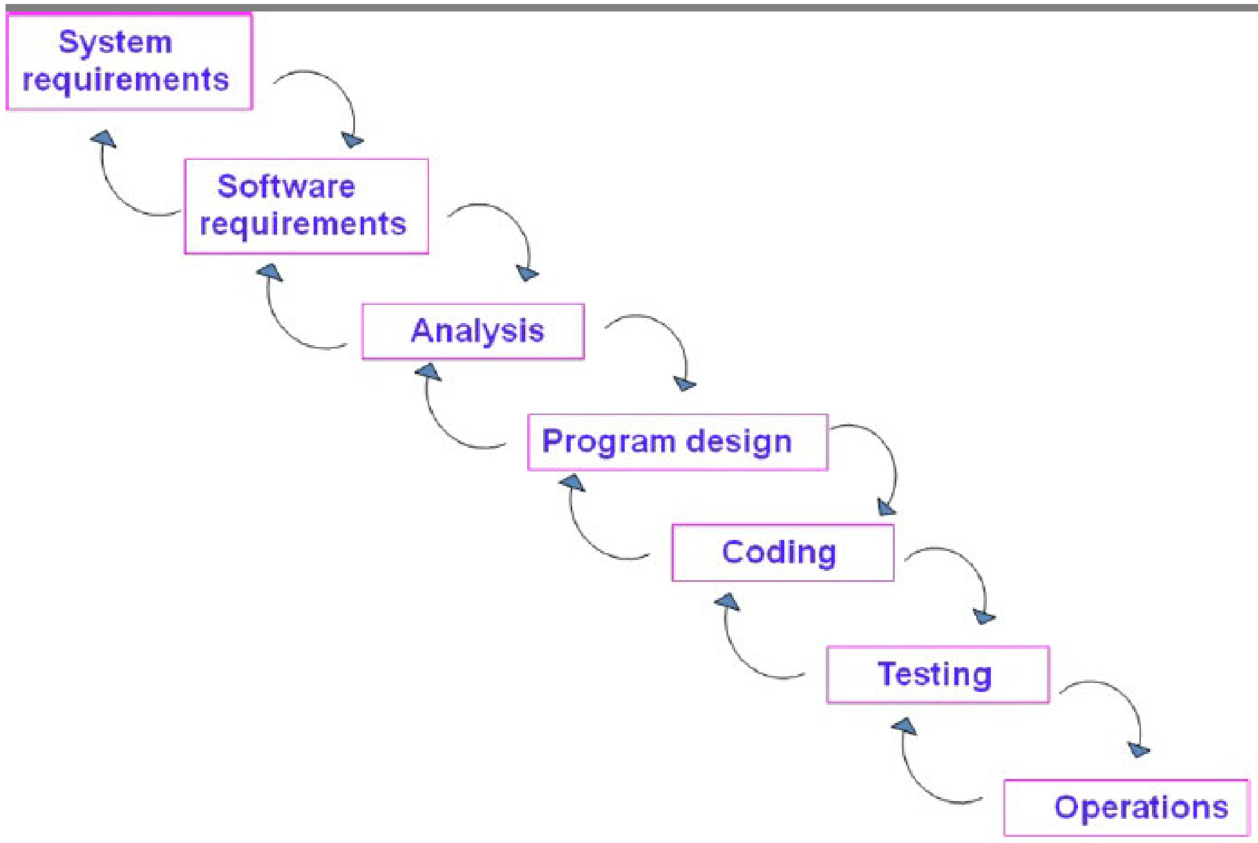
Software Management Process Framework

- WATERFALL MODEL
- It is the baseline process for most conventional software projects have used.
- We can examine this model in two ways:
  - In Theory
  - In Practice

IN THEORY

- In 1970, Winstron Royce presented a paper called "Managing the Development of Large Scale Software Systems" at IEEE WESCON.
- Where he made three primary points:
- 1. There are two essential steps common to the development of computer programs:
  - Analysis
  - Coding
- Analysis and Coding both involve creative work that directly contributes to the usefulness of the end product.
- Waterfall Model Part I: The two basic steps to build a program.

- 2. In order to manage and control all of the intellectual freedom associated with software development one should follow the following steps:
  - System requirements definition
  - Software requirements definition
  - Program design and testing
- These steps addition to the analysis and coding steps.



Waterfall Model Part 2: The large – scale system approach

- 3. Since the testing phase is at the end of the development cycle in the waterfall model, it may be risky and invites failure.
- So we need to do either the requirements must be modified or a substantial design changes is warranted by breaking the software in to different pieces.
- There are five improvements to the basic waterfall model that would eliminate most of the development risks are as follows:

- a) Complete program design before analysis and coding begin (program design comes first):-
- By this technique, the program designer give surety that the software will not fail because of storage, timing, and data fluctuations.
- Begin the design process with program designer, not the analyst or programmers.
- Write an overview document that is understandable, informative, and current so that every worker on the project can gain an elemental understanding of the system.

- **b) Maintain current and complete documentation (Document the design):-**
  - It is necessary to provide a lot of documentation on most software programs.
  - Due to this, helps to support later modifications by a separate test team, a separate maintenance team, and operations personnel who are not software literate.

- **c) Do the job twice, if possible (Do it twice):**
  - If a computer program is developed for the first time, arrange matters so that the version finally delivered to the customer for operational deployment is actually the second version in so far as critical design/operations are concerned.
  - "Do it N times" approach is the principle of modern-day iterative development.

- **d) Plan, control, and monitor testing:**
  - The biggest user of project resources is the test phase. This is the phase of greatest risk in terms of cost and schedule.
  - In order to carryout proper testing the following things to be done:
    - i) Employ a team of test specialists who were not responsible for the original design.
    - ii) Employ visual inspections to spot the obvious errors like dropped minus signs, missing factors of two, jumps to wrong addresses.
    - iii) Test every logic phase.
    - iv) Employ the final checkout on the target computer.

- **e) Involve the customer:**
  - It is important to involve the customer in a formal way so that he has committed himself at earlier points before final delivery by conducting some reviews such as,
    - i) Preliminary software review during preliminary program design step.
    - ii) Critical software review during program design.
    - iii) Final software acceptance review following testing.

IN PRACTICE

- Whatever the advices that are given by the software developers and the theory behind the waterfall model, some software projects still practice the conventional software management approach.
- Projects intended for trouble frequently exhibit the following symptoms:
  - Protracted (delayed) integration and late design breakage
  - Late risk resolution
  - Requirements-driven functional decomposition
  - Adversarial stakeholder relationships
  - Focus on documents and review meetings
- In the conventional model, the entire system was designed on paper, then implemented all at once, then integrated.
- Only at the end of this process was it possible to perform system testing to verify that the fundamental architecture was sound.

Protracted Integration and Late Design Breakage

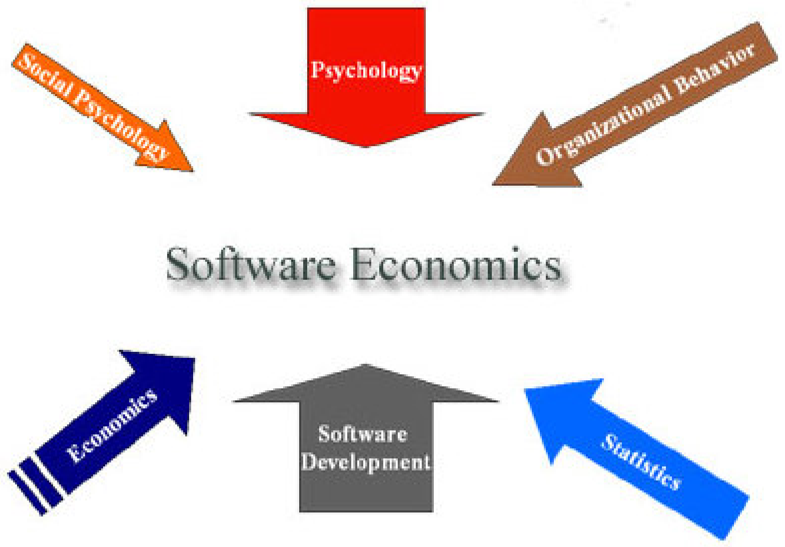
- For a typical development project that used a waterfall model management process, Figure 1-2 illustrates development progress versus time. Progress is defined as percent coded, that is, demonstrable in its target form. (The software was compilable and executable; it was not necessarily complete, compliant, nor up to specifications.) The following sequence was common:
  - Early success via paper designs and thorough (often *too thorough) briefings*
  - Commitment to code late in the life cycle
  - Integration nightmares due to unforeseen implementation issues and interface ambiguities
  - Heavy budget and schedule pressure to get the system working
  - Late shoe-horning of nonoptimal fixes, with no time for redesign
  - A very fragile, unmaintainable product delivered late

Expenditures by Activity for a Conventional Software Project

ACTIVITY	COST
Management	5%
Requirements	5%
Design	10%
Code and Unit Testing	30%
Integration and Test	40%
Deployment	5%
Environment	5%
Total	100%

Software Economics and Cost Estimation

- Software economics is the study of how scarce project resources are allocated for software projects.
- Software economics helps software managers allocate those resources in the most efficient manner.
- Software Economics includes many different disciplines which are shown below:


Social Psychology
Psychology
Organizational Behavior
Software Economics
Economics
Software Development
Statistics

- Software Economics examines the entire idea of software development and it includes many different disciplines such as:
- **Psychology - focuses on the study of behavior** and the reward/punishment model. "What get's rewarded gets done.“
- **Social Psychology - focuses on how people** behave in an organization, quality of work life, and peer pressures.

- **Organizational Behavior - is the process of** analyzing the structure of an organization to understand those structural issues impacting organizational productivity and quality.
- **Economics - the study of prices (components),** costs, and scarcity.
- **Statistics - deals with quantitative and** qualitative techniques for the collection of data, how data is analyzed, and how results are presented.

- **Increasing Marginal Cost**
- As the size of a software project the unit cost (or average cost) rises.
- In other words software has increasing marginal costs and there are few economies of scale when developing a software application.
- Any large engineering or construction project follows this same economic model.
- Hours per function point (average costs) go up as project and organizational size increase.

- **Diseconomies of Scale**
- In all software projects there are some basic principles which cause diseconomies of scale. That is:
  - There are low fixed costs relative to variable costs
  - Communication becomes difficult as project becomes larger
  - Multiple logical paths grow in a nonlinear manner as size increases
  - Interrelationships of functions grow geometrically as project becomes large.

Software Economics Basic Factors:

- All kinds of software cost model can be abstracted into a function of five basic factors:
- 1. Size
- 2. process
- 3. personnel
- 4. environment and
- 5. required quality.

- **1. Size:**
  - of end product, quantified in terms of the number of source instructions. No. of function points.
- **2. Process:**
  - used to produce the end product (rework, bureaucratic delays, communications overheads)
- **3. Personnel:**
  - capability of software engineering human resources.

- **4. Environment:**
  - tools & techniques availability to support of efficient software development and the process.
- **5. Required Quality:**
  - product quality including features, performance reliability etc.
- The relationships among these parameters and the estimated cost written as:
  - Effort=(personnel)(environment)(quality) (sizeProcess)
- There are three generations of S/W development are:
  - Conventional, Transition and Modern practices.

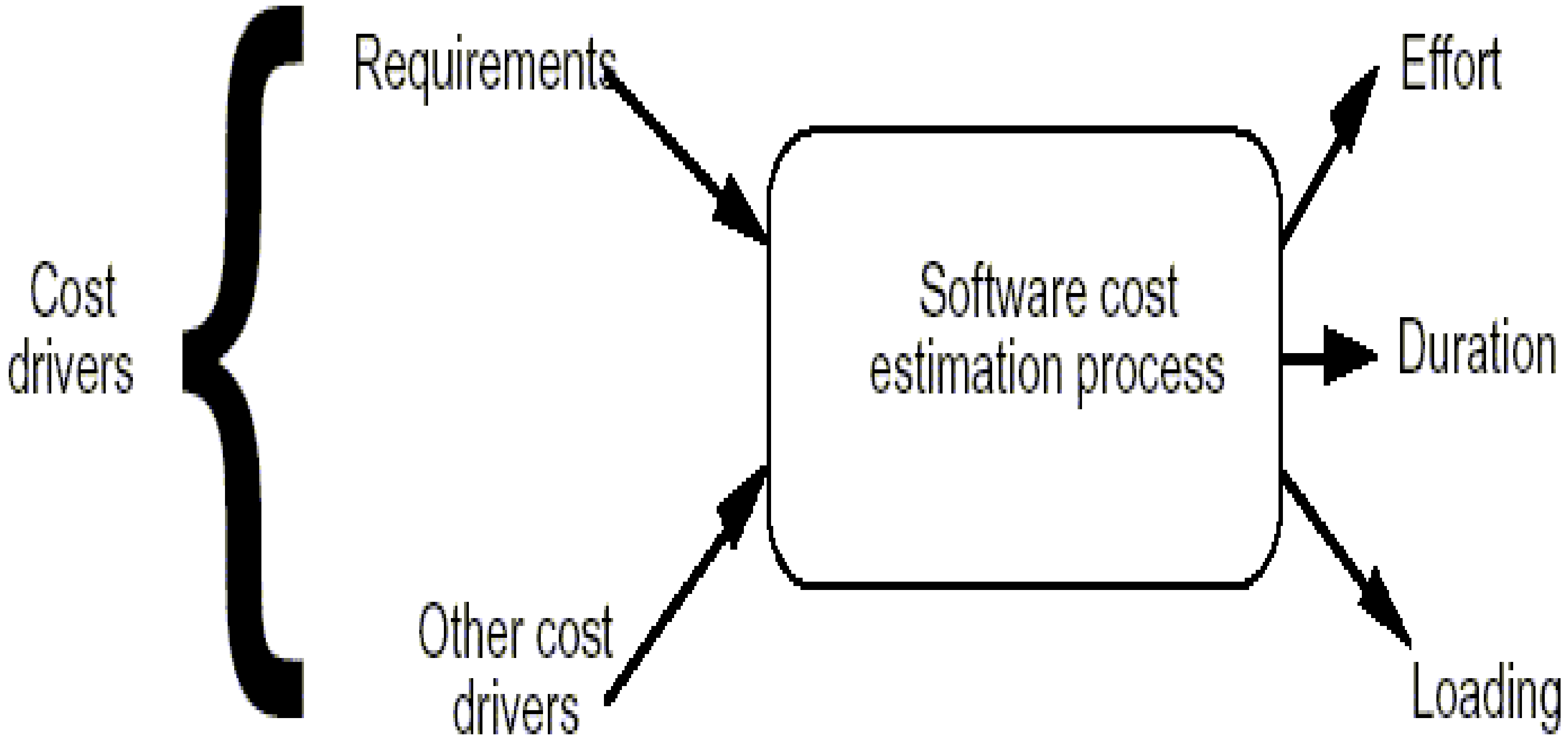
- It can overview return on investment (ROI) by three way:
  - Achieving ROI across line of business (successive system) : cost per unit 1st, 2nd ……… nth system.
  - Achieving ROI across a project with multiple iterations (successive iterations): 1st, 2nd, … nth iteration.
  - Achieving ROI across a life cycle of product releases (successive Releases): 1st, 2nd, … nth release.

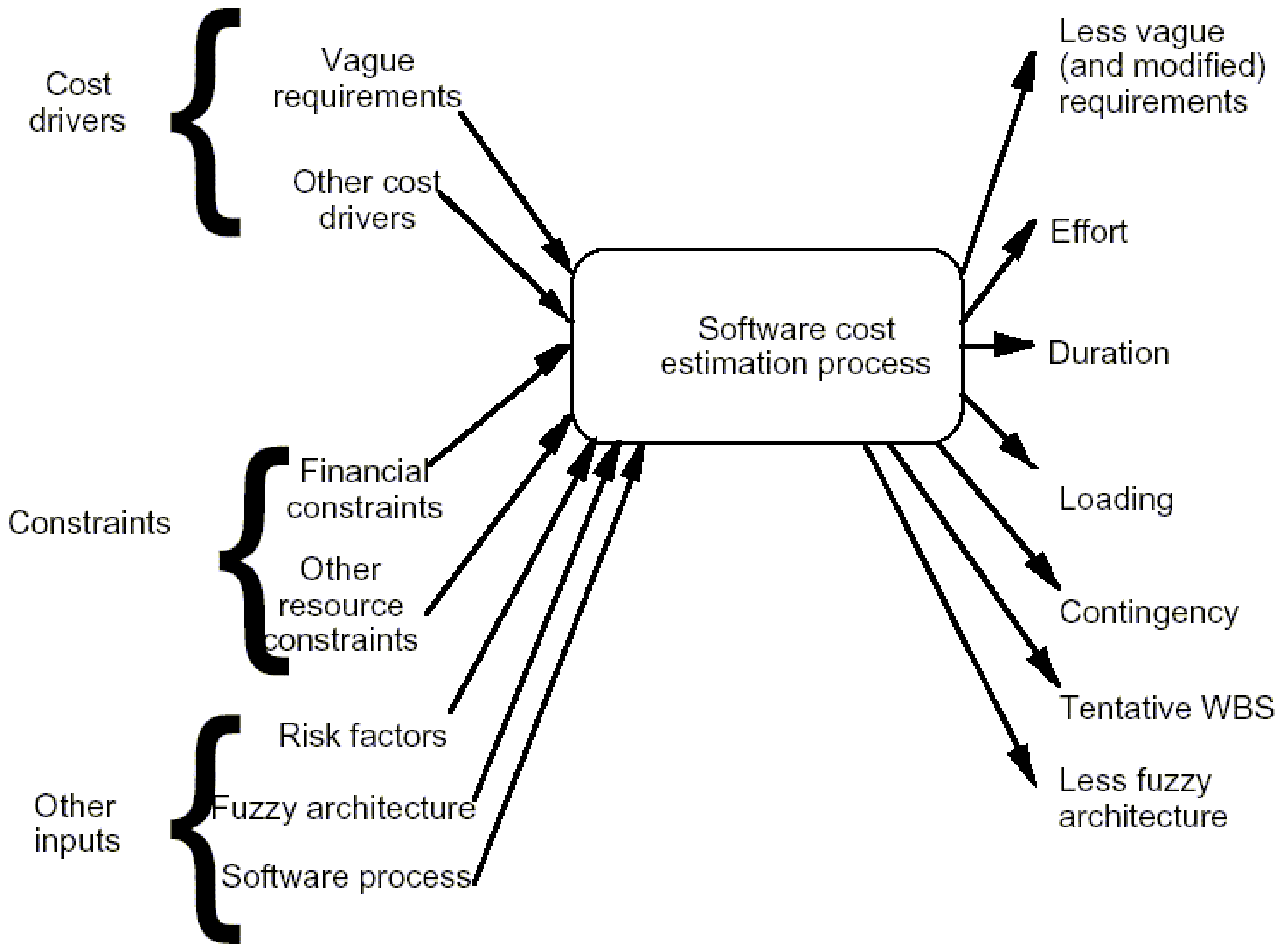
Cost Estimation

- The **costs** of development are primarily the **costs** of the effort involved, so the effort computation is used in both the cost and the schedule **estimate.**
- The initial **cost estimates** may be used to establish a budget for the project and to set a price for the **software** for a customer.

Software Cost Estimation Process

- Definition:
  - A set of techniques and procedures that is used to derive the software cost estimate.
  - Set of inputs to the process and then the process will use these inputs to generate the output.

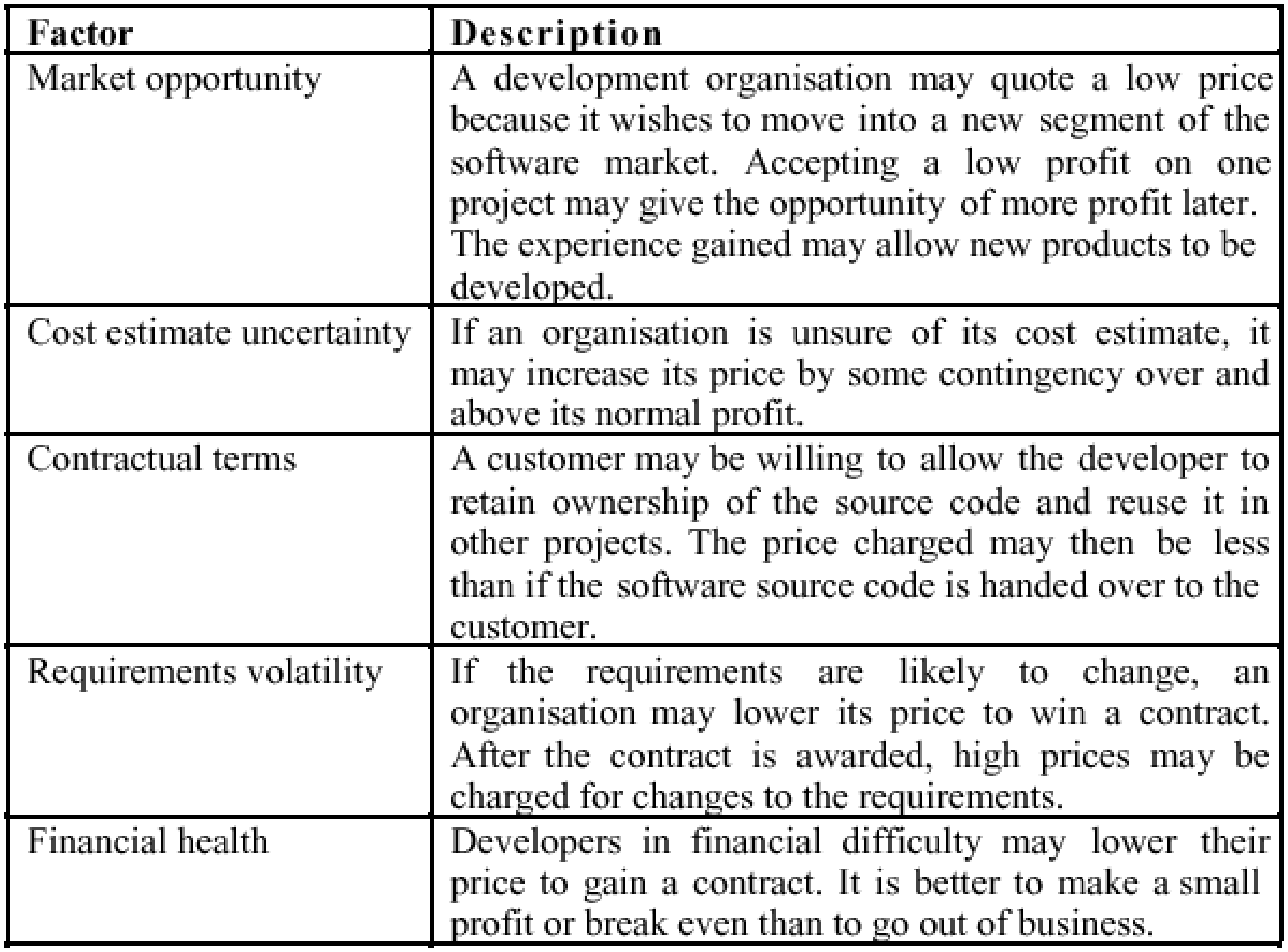

Cost drivers
Requirements
Other cost drivers
Software cost estimation process
Effort
Duration
Loading


Cost drivers
Vague requirements
Other cost drivers
Constraints
Financial constraints
Other resource constraints
Other inputs
Risk factors
Fuzzy architecture
Software process
Software cost estimation process
Less vague (and modified) requirements
Effort
Duration
Loading
Contingency
Tentative WBS
Less fuzzy architecture

Costing and pricing

- Estimates are made to discover the cost, to the developer, of producing a software system
- There is not a simple relationship between the development cost and the price charged to the customer
- Broader organizational, economic, political and business considerations influence the price charged

Software pricing factors



Factor	Description
Market opportunity	A development organisation may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the opportunity of more profit later. The experience gained may allow new products to be developed.
Cost estimate uncertainty	If an organisation is unsure of its cost estimate, it may increase its price by some contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organisation may lower its price to win a contract. After the contract is awarded, high prices may be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a small profit or break even than to go out of business.

Productivity measures

- Size related measures based on some output from the software process. This may be lines of delivered source code, object code instructions, etc.
- Function-related measures based on an estimate of the functionality of the delivered software. Function-points are the best known of this type of measure.

Improving Software Economics

- Reducing Software product size,
- Improving software processes,
- Improving team effectiveness,
- Improving automation,
- Achieving required quality, and
- Peer inspections.

- Most software cost models can be abstracted into a function of five basic parameters:
  - size, process, personnel, environment, and required quality.

1. Reducing the **size or complexity of what** needs to be developed
2. Improving the development **process**
3. Using more-skilled **personnel and better** teams (not necessarily the same thing)
4. Using better **environments (tools to** automate the process)
5. Trading off or backing off on **quality** thresholds

- These parameters are given in priority order for most software domains.
- Below Table 3-1 lists some of the technology developments, process improvement efforts, and management approaches targeted at improving the economics of software development and integration.

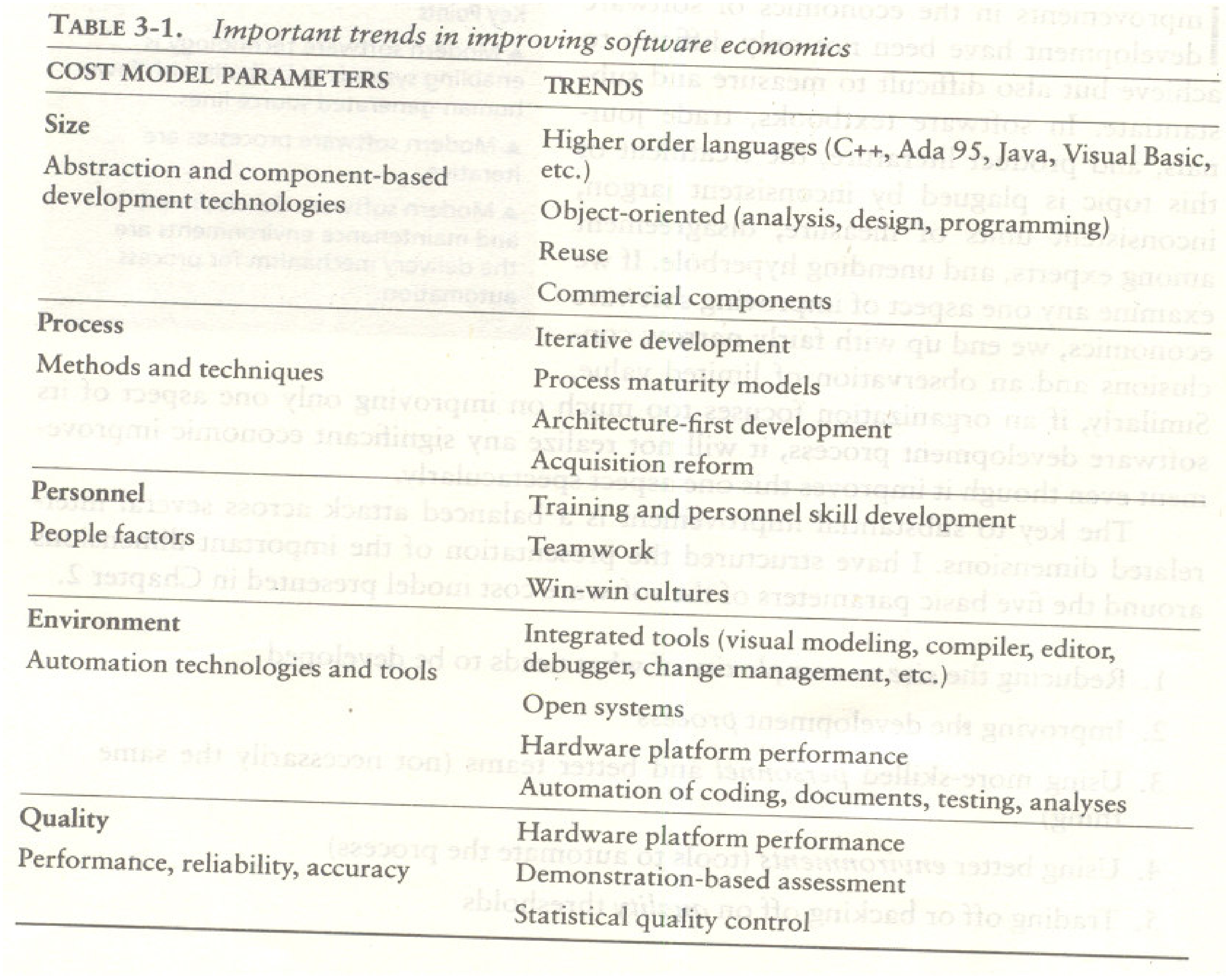


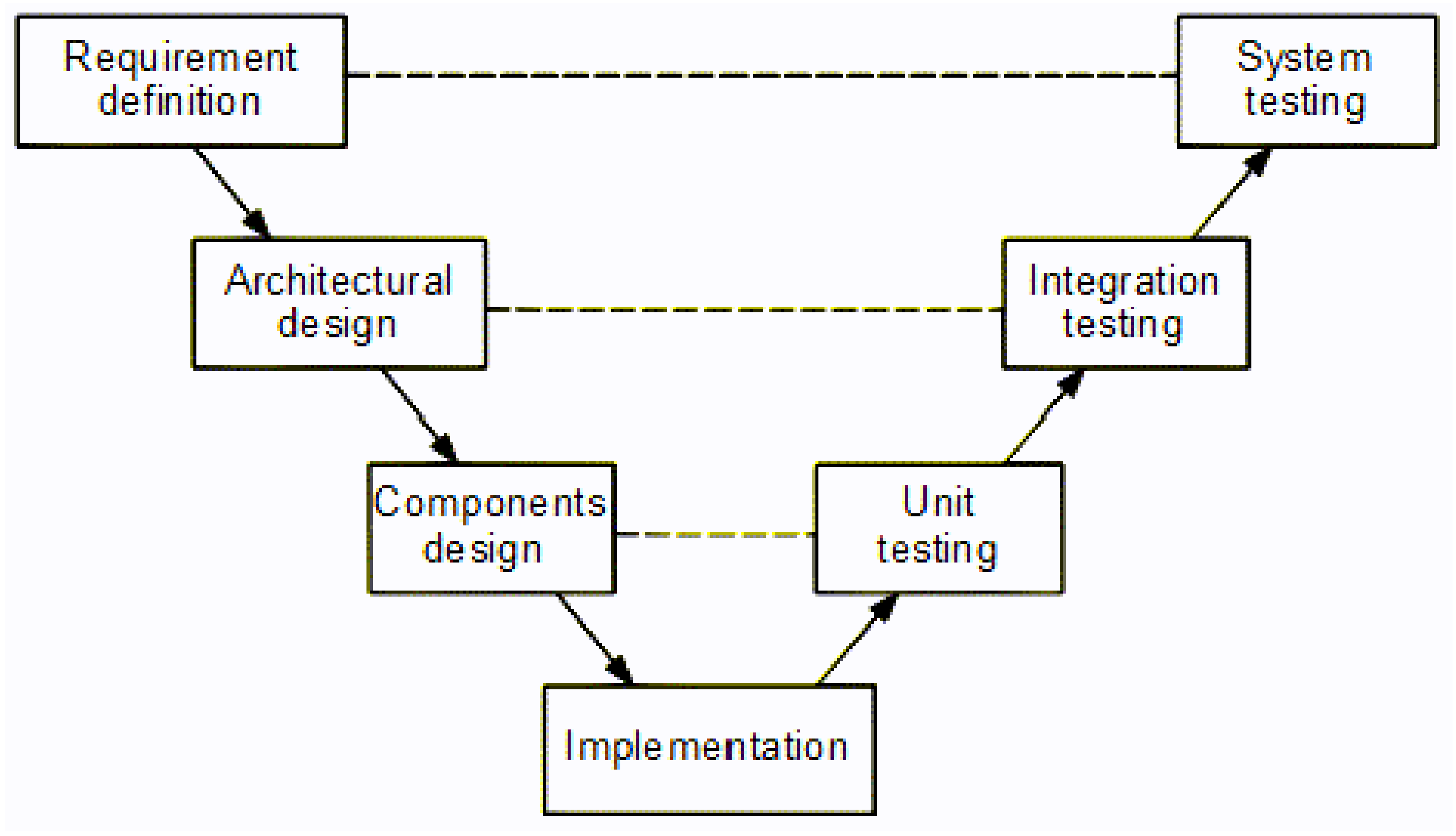
TABLE 3-1. *Important trends in improving software economics*

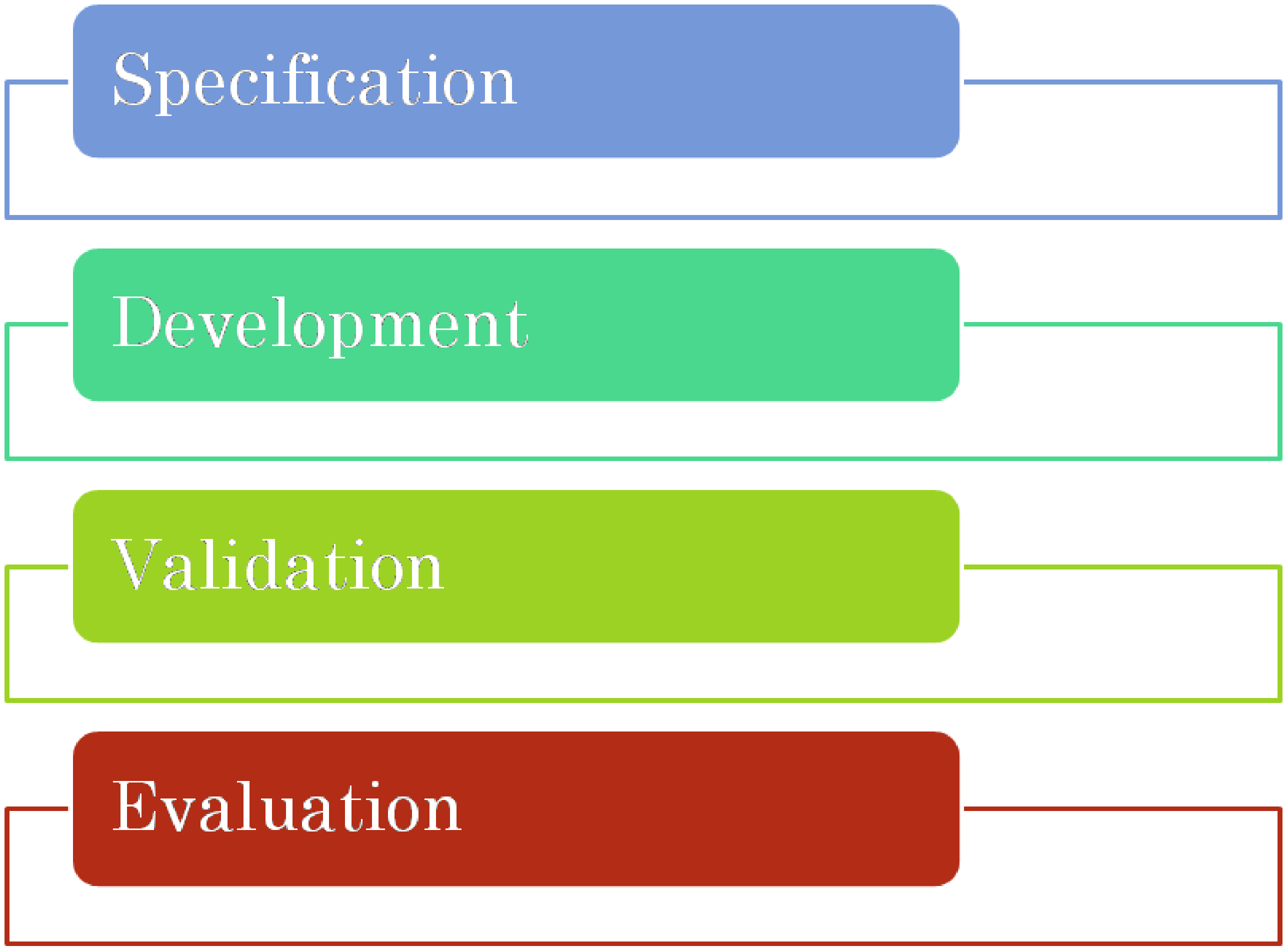
COST MODEL PARAMETERS	TRENDS
Size Abstraction and component-based development technologies	Higher order languages (C++, Ada 95, Java, Visual Basic, etc.) Object-oriented (analysis, design, programming) Reuse Commercial components
Process Methods and techniques	Iterative development Process maturity models Architecture-first development Acquisition reform
Personnel People factors	Training and personnel skill development Teamwork Win-win cultures
Environment Automation technologies and tools	Integrated tools (visual modeling, compiler, editor, debugger, change management, etc.) Open systems Hardware platform performance Automation of coding, documents, testing, analyses
Quality Performance, reliability, accuracy	Hardware platform performance Demonstration-based assessment Statistical quality control

4. Software Process

- A software process (also knows as software methodology) is a set of related activities that leads to the production of the software.
- These activities may involve the development of the software from the scratch, or, modifying an existing system.
- A **software development process, also known** as a **software development lifecycle, is a** structure imposed on the **development of a software product.**

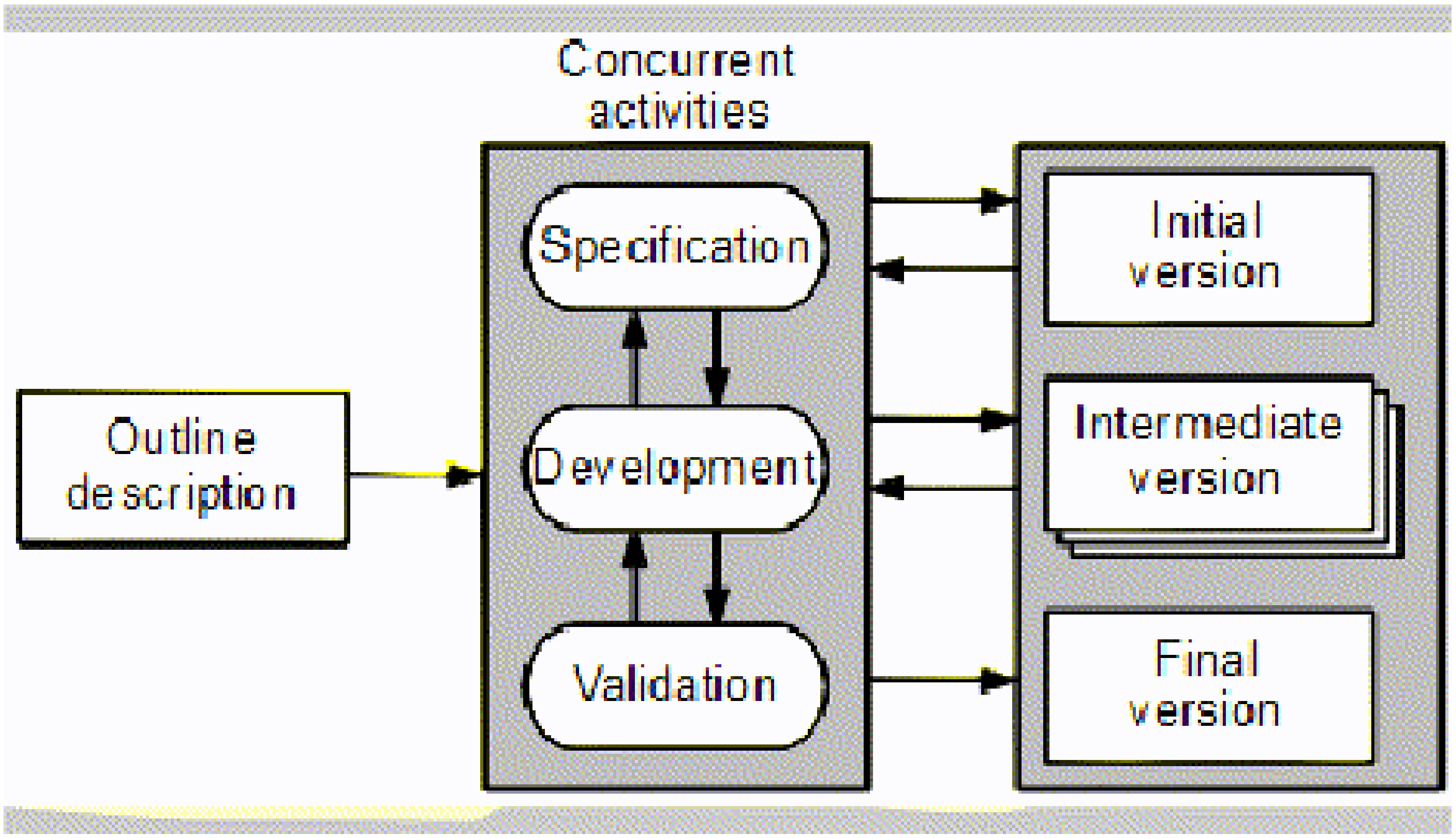
- A **software process is represented as a set of** work phases that is applied to design and build a **software product.**
- The **software that meets the specification is** produced.


Requirement definition
System testing
Architectural design
Integration testing
Components design
Unit testing
Implementation

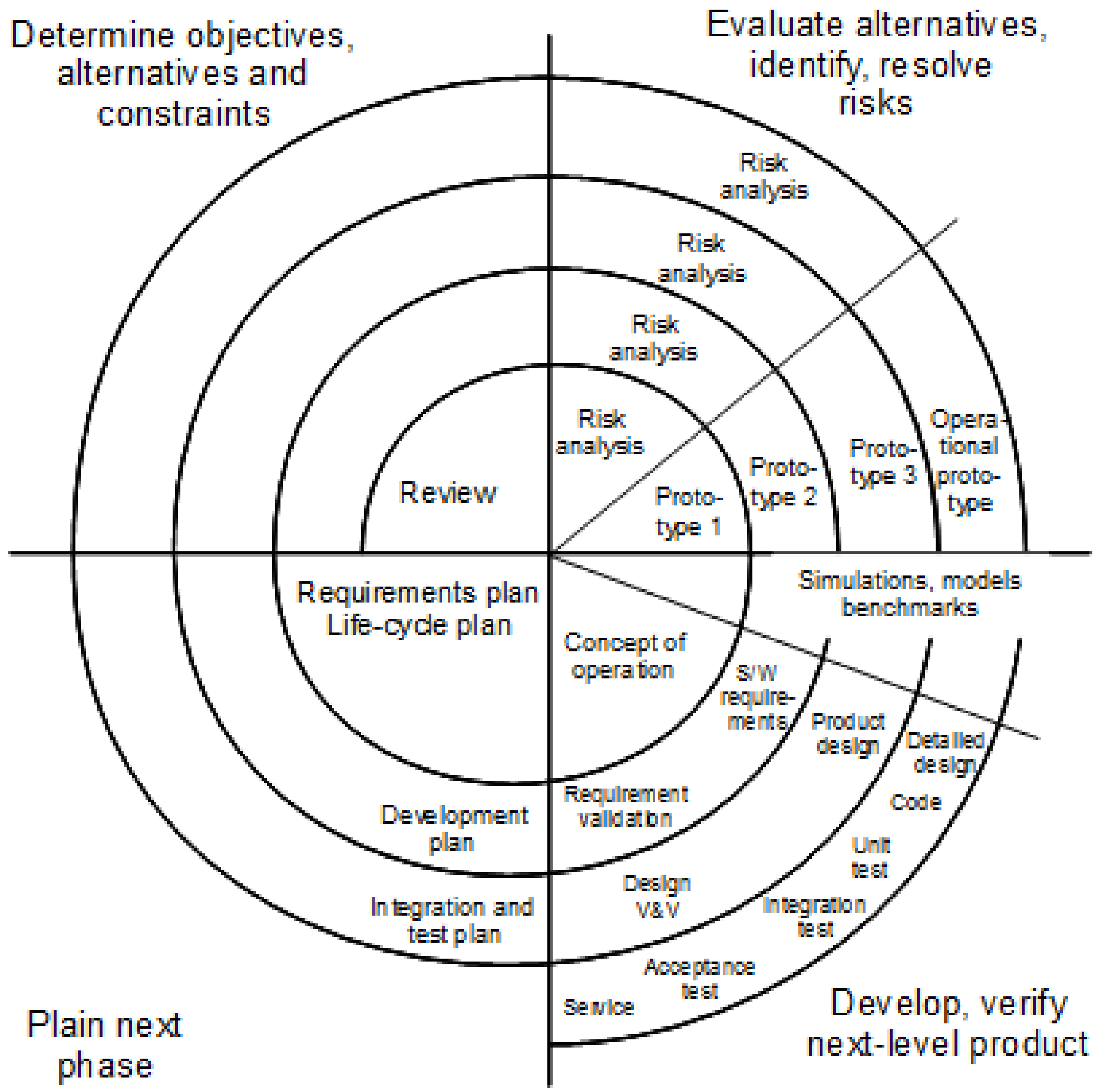


Four Activities of Software Process Framework

Evolutionary Development



Spiral Development (Model)



There are four components includes in Software Process

- **Software Process Model:**
  - The waterfall model, V-model of software process, Evolutionary development, Component based software engineering.
- **Process Iteration:**
  - Incremental delivery, Spiral development,
- **Process activities:**
  - Software Specification, software design and development, software validation and software evolution.
- The Rational Unified Process

Three level of Software Process:

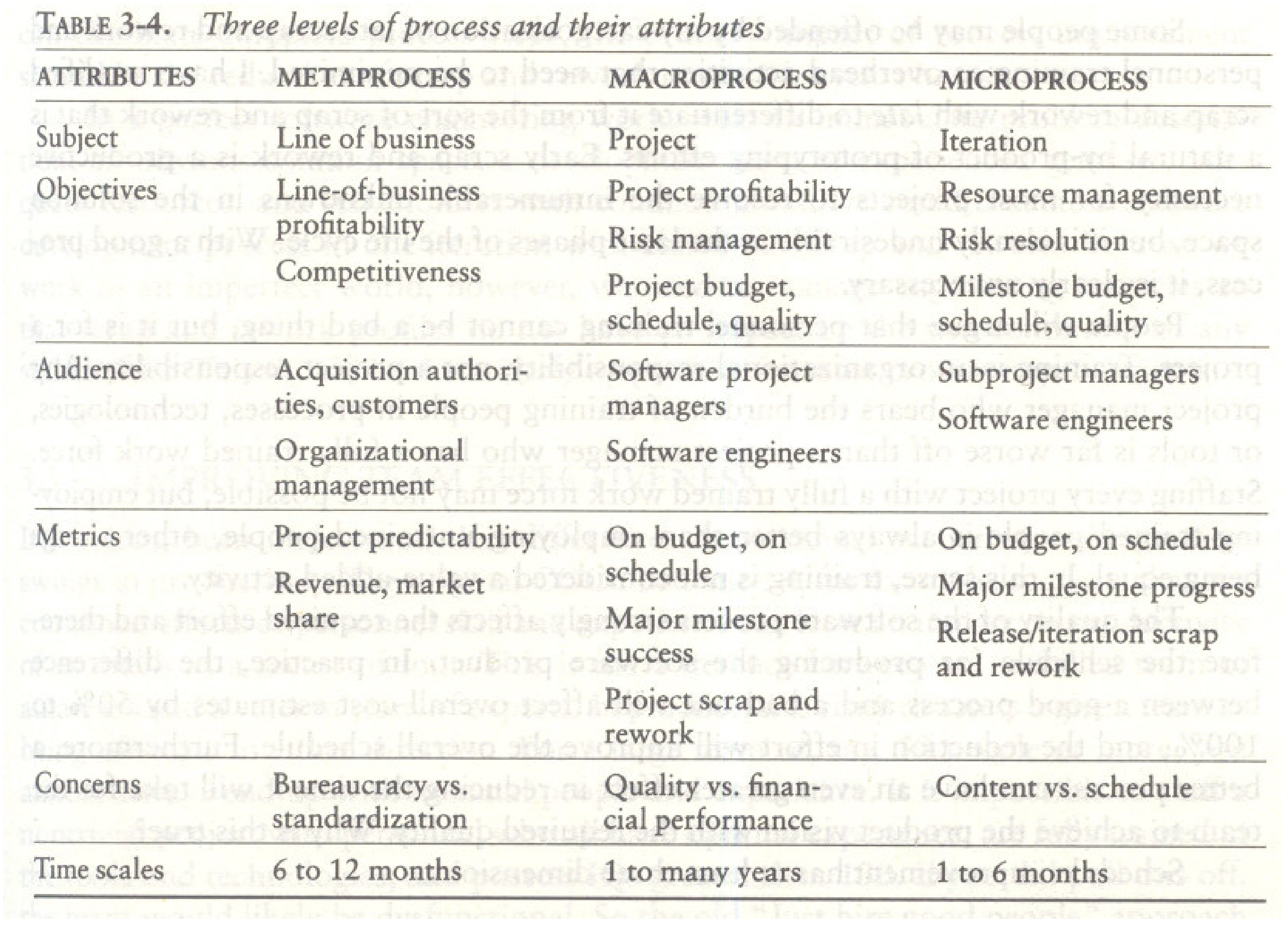


TABLE 3-4. *Three levels of process and their attributes*

ATTRIBUTES	METAPROCESS	MACROPROCESS	MICROPROCESS
Subject	Line of business	Project	Iteration
Objectives	Line-of-business profitability Competitiveness	Project profitability Risk management Project budget, schedule, quality	Resource management Risk resolution Milestone budget, schedule, quality
Audience	Acquisition authorities, customers Organizational management	Software project managers Software engineers	Subproject managers Software engineers
Metrics	Project predictability Revenue, market share	On budget, on schedule Major milestone success Project scrap and rework	On budget, on schedule Major milestone progress Release/iteration scrap and rework
Concerns	Bureaucracy vs. standardization	Quality vs. financial performance	Content vs. schedule
Time scales	6 to 12 months	1 to many years	1 to 6 months

5. Team Effectiveness & Software

- **Environment & Quality Target**
  - Teamwork is much more important than the sum of the individuals. With software teams, a project manager needs to configure a balance of solid talent with highly skilled people in the leverage positions. Some maxims of team management include the following:
    - A well-managed project can succeed with a nominal engineering team.
    - A mismanaged project will almost never succeed, even with an expert team of engineers.
    - A well-architected system can be built by a nominal team of software builders.
    - A poorly architected system will flounder even with an expert team of builders.

Boehm five staffing principles

- The principle of top talent: Use better and fewer people
- The principle of job matching: Fit the tasks to the skills and motivation of the people available.
- The principle of career progression: An organization does best in the long run by helping its people to self-actualize.
- The principle of team balance: Select people who will complement and harmonize with one another
- The principle of phase-out: Keeping a misfit on the team doesn't benefit anyone

- Software project managers need many leadership qualities in order to enhance team effectiveness.
- The following are some crucial attributes of successful software project managers that deserve much more attention:
- **1. Hiring skills. Few decisions are as important as** hiring decisions. Placing the right person in the right job seems obvious but is surprisingly hard to achieve.
- **2. Customer-interface skill. Avoiding** adversarial relationships among stakeholders is a prerequisite for success.

- **3. Decision-making skill. The jillion books** written about management have failed to provide a clear definition of this attribute. We all know a good leader when we run into one, and decision-making skill seems obvious despite its intangible definition.
- **4. Team-building skill. Teamwork requires that** a manager establish trust, motivate progress, exploit eccentric prima donnas, transition average people into top performers, eliminate misfits, and consolidate diverse opinions into a team direction.

- **5. Selling skill.** Successful project managers must sell all stakeholders (including themselves) on decisions and priorities, sell candidates on job positions, sell changes to the status quo in the face of resistance, and sell achievements against objectives. In practice, selling requires continuous negotiation, compromise, and empathy
- **6. Software environment is the term commonly** used to refer to support an application.
- 7. A **software environment for a particular** application could include the operating system, the database system, specific development tools or compiler.

Quality Target

- Software best practices are derived from the development process and technologies.
- Key practices that improve overall software quality include the following:
  - Focusing on driving requirements and critical use cases early in the life cycle, focusing on requirements completeness and traceability late in the life cycle, and focusing throughout the life cycle on a balance between requirements evolution, design evolution, and plan evolution

- Using metrics and indicators to measure the progress and quality of an architecture as it evolves from a high-level prototype into a fully compliant product
- Providing integrated life-cycle environments that support early and continuous configuration control, change management, rigorous design methods, document automation, and regression test automation
- Using visual modeling and higher level languages that support architectural control, abstraction, reliable programming, reuse, and self-documentation
- Early and continuous insight into performance issues through demonstration- based evaluations

- Conventional development processes stressed early sizing and timing estimates of computer program resource utilization. However, the typical chronology of events in performance assessment was as follows
- **Project inception. The proposed design was** asserted to be low risk with adequate performance margin.
- **Initial design review. Optimistic assessments of** adequate design margin were based mostly on paper analysis or rough simulation of the critical threads. In most cases, the actual application algorithms and database sizes were fairly well understood.

- **Mid-life-cycle design review. The assessments** started whittling away at the margin, as early benchmarks and initial tests began exposing the optimism inherent in earlier estimates.
- **Integration and test. Serious performance** problems were uncovered, necessitating fundamental changes in the architecture. The underlying infrastructure was usually the scapegoat, but the real culprit was immature use of the infrastructure, immature architectural solutions, or poorly understood early design trade-offs.

6. Principles of Conventional Software Engineering

- 1. Make quality
- 2. High-quality software is possible.
- 3. Give products to customers early.
- 4. Determine the problem before writing the requirements.
- 5. Evaluate design alternatives.
- 6. Use an appropriate process model.
- 7. Use different languages for different phases.
- 8. Minimize intellectual distance.
- 9. Put techniques before tools.
- 10. Get it right before you make it faster.
- 11. Inspect code.

6. Principles of Conventional Software Engineering

- 12. Good management is more important than good technology.
- 13. People are the key to success.
- 14. Follow with care.
- 15. Take responsibility.
- 16. Understand the customer's priorities.
- 17. The more they see, the more they need.
- 18. Plan to throw one away.
- 19. Design for change.
- 20. Design without documentation is not design.
- 21. Use tools, but be realistic.
- 22. Avoid tricks.
- 23. Encapsulate.
- 24. Use coupling and cohesion.
- 25. Use the McCabe complexity measure.
- 26. Don't test your own software.
- 27. Analyze causes for errors.
- 28. Realize that software's entropy increases.
- 29. People and time are not interchangeable.
- 30. Expect excellence.

7. Principles of Modern Software Management

- Top 10 principles of modern software management are. (The first five, which are the main themes of my definition of an iterative process, are summarized here.
- 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, and the life-cycle plans before the resources are committed for full-scale development.

- **2. Establish an iterative life-cycle process that confronts risk early.**
  - With today's sophisticated software systems, it is not possible to define the entire problem, design the entire solution, build the software, and then test the end product in sequence.
  - Instead, an iterative process that refines the problem understanding, an effective solution, and an effective plan over several iterations encourages a balanced treatment of all stakeholder objectives.
  - Major risks must be addressed early to increase predictability and avoid expensive downstream scrap and rework.

- **3. Transition design methods to emphasize component- based development. Moving from a** line-of-code mentality to a component-based mentality is necessary to reduce the amount of human- generated source code and custom development.
- **4. Establish a change management environment. The** dynamics of iterative development, including concurrent workflows by different teams working on shared artifacts, necessitates objectively controlled baselines

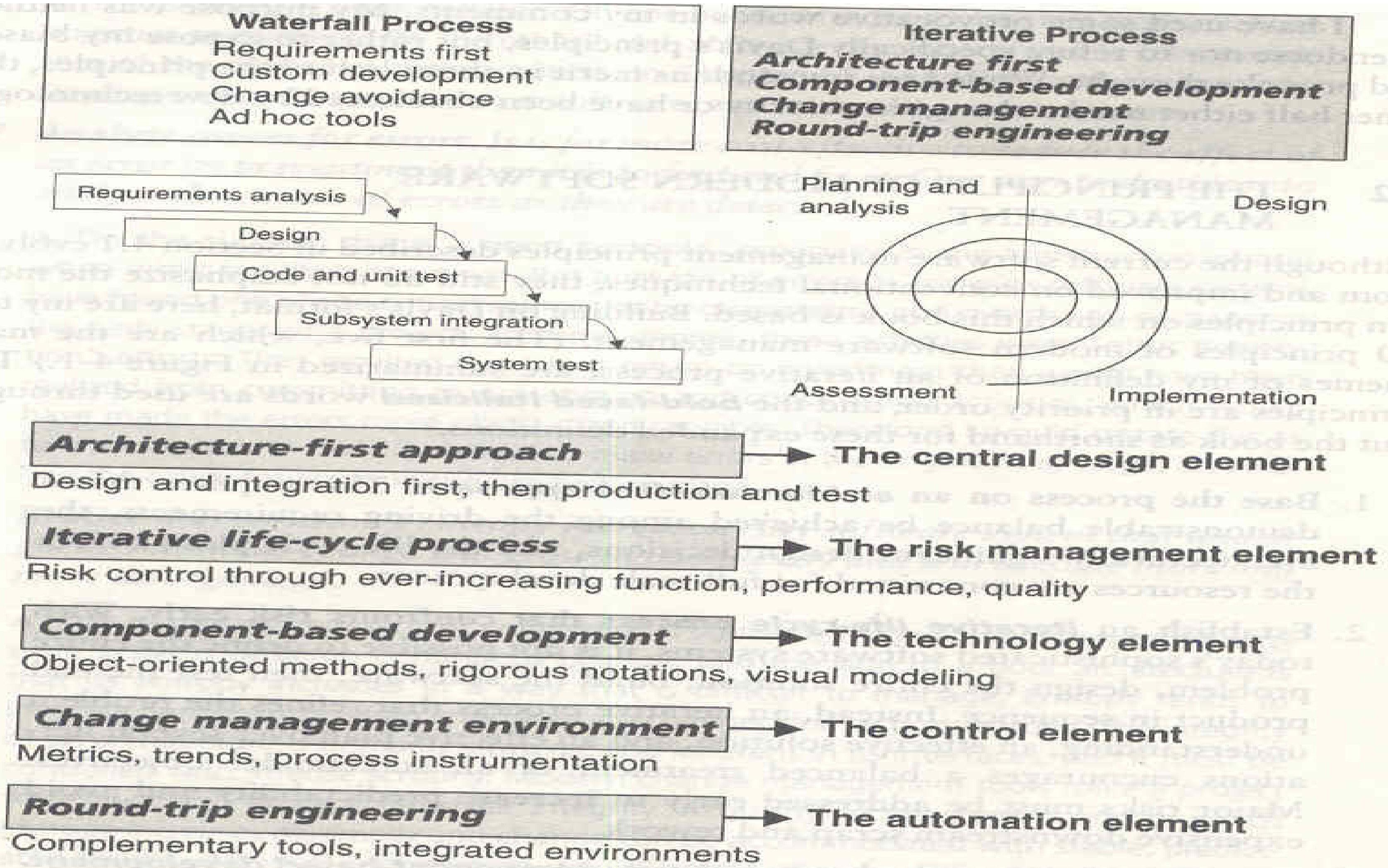

Waterfall Process
Requirements first
Custom development
Change avoidance
Ad hoc tools
Iterative Process
Architecture first
Component-based development
Change management
Round-trip engineering
Requirements analysis
Design
Code and unit test
Subsystem integration
System test
Planning and analysis
Design
Assessment
Implementation
Architecture-first approach
The central design element
Design and integration first, then production and test
Iterative life-cycle process
The risk management element
Risk control through ever-increasing function, performance, quality
Component-based development
The technology element
Object-oriented methods, rigorous notations, visual modeling
Change management environment
The control element
Metrics, trends, process instrumentation
Round-trip engineering
The automation element
Complementary tools, integrated environments

FIGURE 4-1. *The top five principles of a modern process*

- 5. **Enhance change freedom through tools that support *round-trip engineering.*** Round-trip engineering is the environment support necessary to automate and synchronize engineering information in different formats(such as requirements specifications, design models, source code, executable code, test cases).
- 6. **Capture design artifacts in rigorous, *model-based notation. A model based approach (such as UML)*** supports the evolution of semantically rich graphical and textual design notations.
- 7. **Instrument the process for *objective quality control and progress assessment.*** Life-cycle assessment of the progress and the quality of all intermediate products must be integrated into the process.

- **8. Use a demonstration-based approach to assess intermediate artifacts.**
- **9. Plan intermediate releases in groups of usage scenarios with evolving levels of detail. It is essential that the software** management process drive toward early and continuous demonstrations within the operational context of the system, namely its use cases.
- **10. Establish a configurable process that is economically scalable. No single process is** suitable for all software developments.
- Below table maps top 10 risks of the conventional process to the key attributes and principles of a modern process

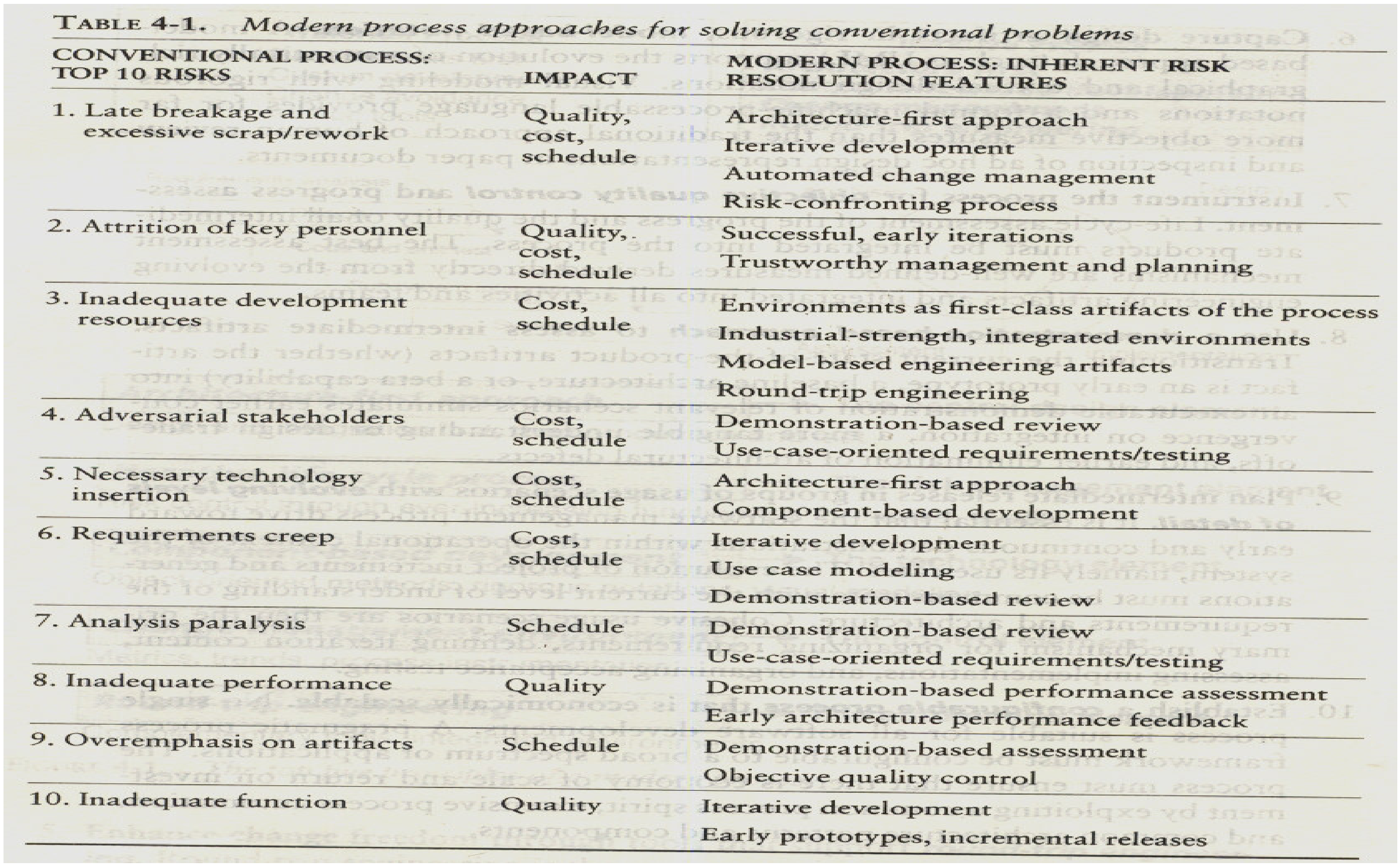


TABLE 4-1. *Modern process approaches for solving conventional problems*

CONVENTIONAL PROCESS: TOP 10 RISKS	IMPACT	MODERN PROCESS: INHERENT RISK RESOLUTION FEATURES
1. Late breakage and excessive scrap/rework	Quality, cost, schedule	Architecture-first approach Iterative development Automated change management Risk-confronting process
2. Attrition of key personnel	Quality, cost, schedule	Successful, early iterations Trustworthy management and planning
3. Inadequate development resources	Cost, schedule	Environments as first-class artifacts of the process Industrial-strength, integrated environments Model-based engineering artifacts Round-trip engineering
4. Adversarial stakeholders	Cost, schedule	Demonstration-based review Use-case-oriented requirements/testing
5. Necessary technology insertion	Cost, schedule	Architecture-first approach Component-based development
6. Requirements creep	Cost, schedule	Iterative development Use case modeling Demonstration-based review
7. Analysis paralysis	Schedule	Demonstration-based review Use-case-oriented requirements/testing
8. Inadequate performance	Quality	Demonstration-based performance assessment Early architecture performance feedback
9. Overemphasis on artifacts	Schedule	Demonstration-based assessment Objective quality control
10. Inadequate function	Quality	Iterative development Early prototypes, incremental releases

8. Iterative Process

- Modern software development processes have moved away from the conventional waterfall model, in which each stage of the development process is dependent on completion of the previous stage.
- The economic benefits inherent in transitioning from the conventional waterfall model to an iterative development process are significant but difficult to quantify.

- As one benchmark of the expected economic impact of process improvement, consider the process exponent parameters of the COCOMO II model. (Appendix B provides more detail on the COCOMO model) This exponent can range from 1.01 (virtually no diseconomy of scale) to 1.26 (significant diseconomy of scale).
- The parameters that govern the value of the process exponent are application precedentedness, process flexibility, architecture risk resolution, team cohesion, and software process maturity.

- The following paragraphs map the process exponent parameters of CO COMO II to top 10 principles of a modern process.
- **Application precedentedness.**
  - Domain experience is a critical factor in understanding how to plan and execute a software development project.
  - For unprecedented systems, one of the key goals is to confront risks and establish early precedents, even if they are incomplete or experimental.
  - This is one of the primary reasons that the software industry has moved to an ***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.***

- **Process flexibility.**
  - Development of modern software is characterized by such a broad solution space and so many interrelated concerns that there is a paramount need for continuous incorporation of changes.
  - These changes may be inherent in the problem understanding, the solution space, or the plans. Project artifacts must be supported by efficient ***change management commensurate with project*** needs.
  - A ***configurable process that allows a common*** framework to be adapted across a range of projects is necessary to achieve a software return on investment.

- **Architecture risk resolution.**
  - ***Architecture-first development is a crucial theme*** underlying a successful iterative development process.
  - A project team develops and stabilizes architecture before developing all the components that make up the entire suite of applications components.
  - An ***architecture-first and component-based development approach forces the infrastructure,*** common mechanisms, and control mechanisms to be elaborated early in the life cycle and drives all component make/buy decisions into the architecture process.

Team Cohesion

- Successful teams are unified, and integrated teams are successful. Successful teams and unified or integrated teams share common objectives and priorities.
- Advances in technology (such as programming languages, UML, and visual modeling) have enabled more rigorous and understandable notations for communicating software engineering information, particularly in the requirements and design artifacts that previously were ad hoc and based completely on paper exchange.
- The ***model-based formats have also enabled the round-trip engineering support needed to establish*** change freedom sufficient for evolving design representations.

Software Process Maturity

- The Software Engineering Institute's Capability Maturity Model (CMM) is a wellaccepted benchmark for software process assessment.
- One of key themes is that truly mature processes are enabled through an integrated environment that provides the appropriate level of automation to instrument the process for ***objective quality control.***

Project Management - Advantages

- Responsiveness to Clients and the Environment
- Ability to make Timely Trade-off Decisions
- Central Locus of Decisions to insure overall Project Optimality
- Better control, better customer relations, Shorter development time, lower costs, Higher quality and reliability, higher profit margins, better co-ordination, higher morale

Project Management -Advantages

(cont..)

- Bosses, customers, and other stakeholders do not like surprises. Good project management (PM) provides assurance and reduces risk
- PM provides the tools and environment to plan, monitor, track, and manage schedules, resources, costs, and quality
- PM provides a history or metrics base for future planning as well as good documentation
- Project members learn and grow by working in a cross functional team environment

Project Management - Disadvantages

- Greater Organizational Complexity
- Lower Personnel Utilization
- More Managerial Conflicts

Unit – Important Questions

- 1. Explain briefly Waterfall model. Also explain Conventional s/w management performance?
- 2. Define Software Economics. Also explain Pragmatic s/w cost estimation?
- 3. Explain Important trends in improving Software economics?
- 4. Explain five staffing principal offered by Boehm. Also explain Peer Inspections?
- 5. Explain principles of conventional software engineering?
- 6. Explain briefly principles of modern software management

Any Queries?