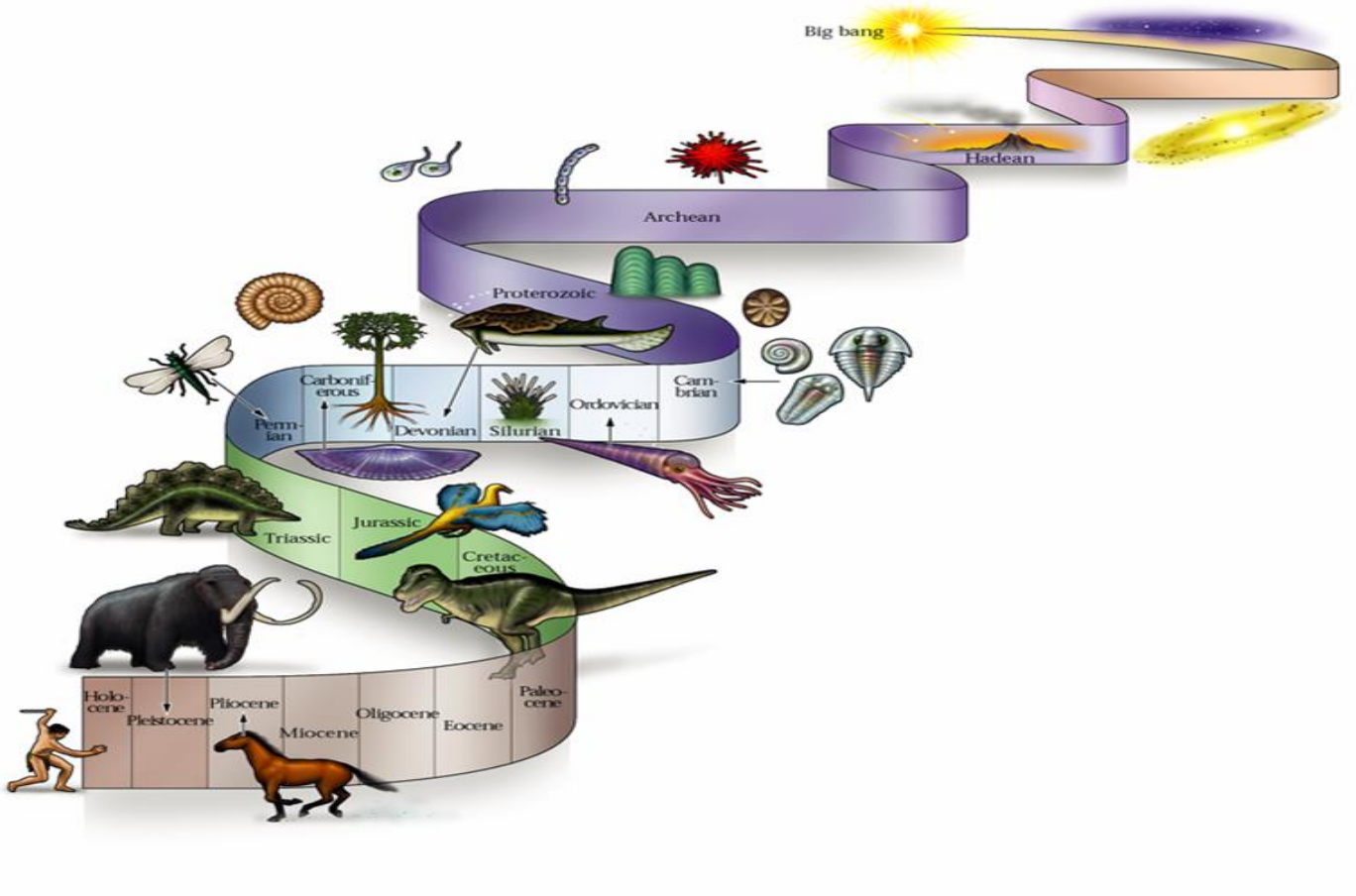
JYOTHISHMATHI INSTITUTE OF TECHNOLOGY & SCIENCE
Nustulapur, Karimnagar

Prepared by:

P.Balakishan

Associate Professor,Dept. of CSE

Part 1
Evolution of Software Economics



Part 1

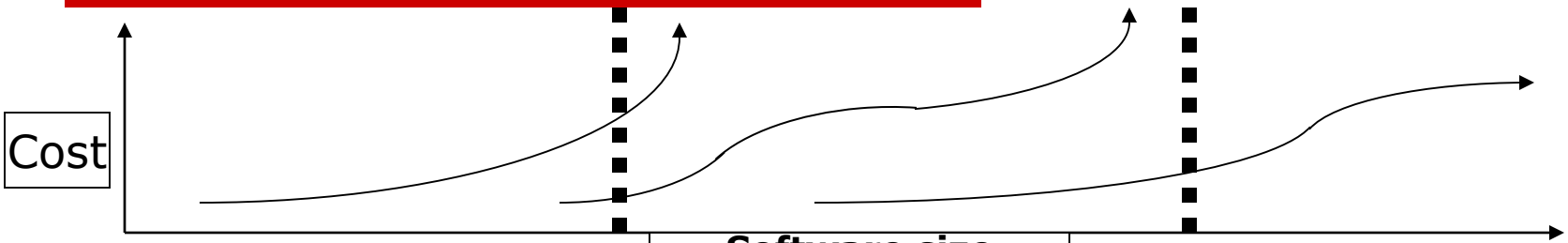
Evolution of Software Economics

- Most software cost models can be abstracted into a function of five basic parameters:
  - Size (typically, number of source instructions)
  - Process (the ability of the process to avoid non-value-adding activities)
  - Personnel (their experience with the computer science issues and the applications domain issues of the project)
  - Environment (tools and techniques available to support efficient software development and to automate process)
  - Quality (performance, reliability, adaptability…)

Part 1

Evolution of Software Economics

Three generations of software economics

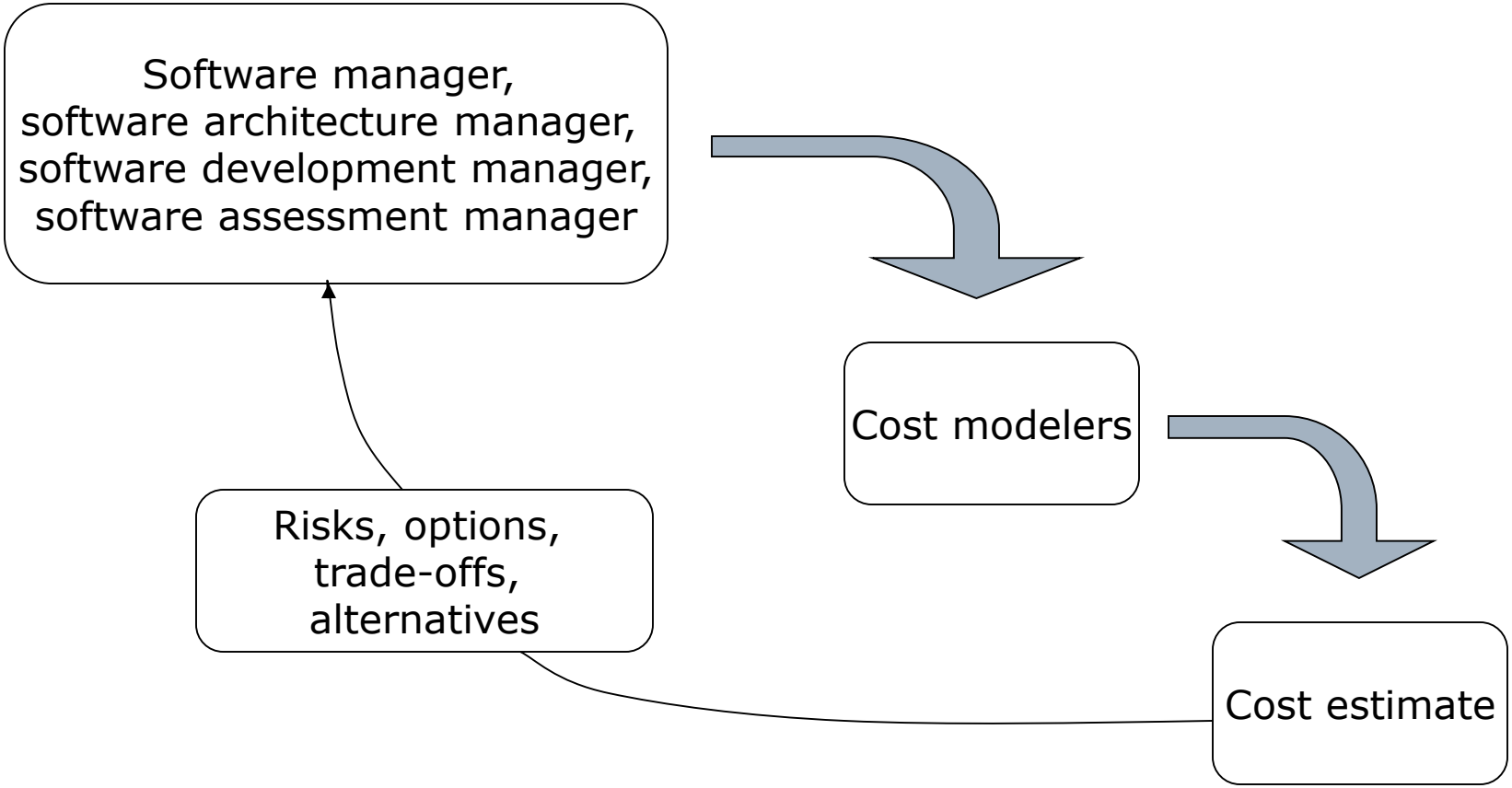


1960s-1970s	1980s-1990s	2000 and on
Waterfall model Functional design Diseconomy of scale	Process improvement Encapsulation-based Diseconomy of scale	Iterative development Component- based Return to investment
Environments/tools: Custom	**Environments/tools:** Off-the-shelf, separate	**Environments/tools:** Off-the-shelf, integrated
Size: 100% custom	**Size:** 30%component-based, 70% custom	**Size:** 70%component-based, 30% custom
Process: Ad hoc	**Process:** Repeatable	**Process:** Managed/measured
Typical project performance		
Predictably bad Always: -Over budget -Over schedule	**Unpredictable** Infrequently: -On budget -On schedule	**Predictable** Usually: -On budget -On schedule

Part 1

Evolution of Software Economics

The predominant cost estimation process



Part 1

Evolution of Software Economics

Pragmatic software cost estimation

- A good estimate has the following attributes:
  - It is conceived and supported by the project manager, architecture team, development team, and test team accountable for performing the work.
  - It is accepted by all stakeholders as ambitious but realizable.
  - It is based on a well defined software cost model with a credible basis.
  - It is based on a database of relevant project experience that includes similar processes, technologies, environments, quality requirements, and people.
  - It is defined in enough detail so that its key risk areas are understood and the probability of success is objectively assessed.

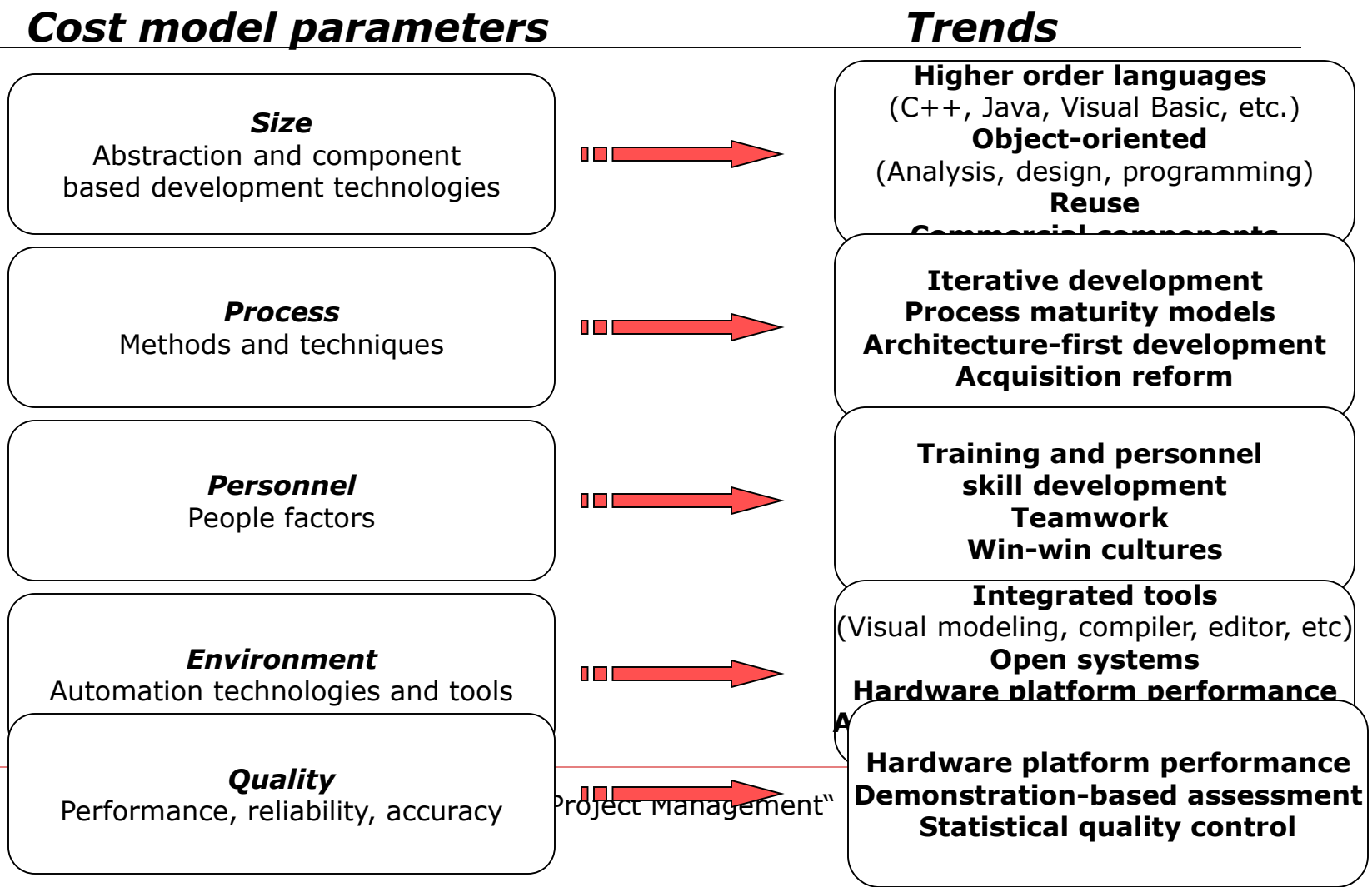
Part 1
Improving Software Economics

- Five basic parameters of the software cost model:
  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

Part 1

Improving Software Economics

Important trends in improving software economics



Part 1

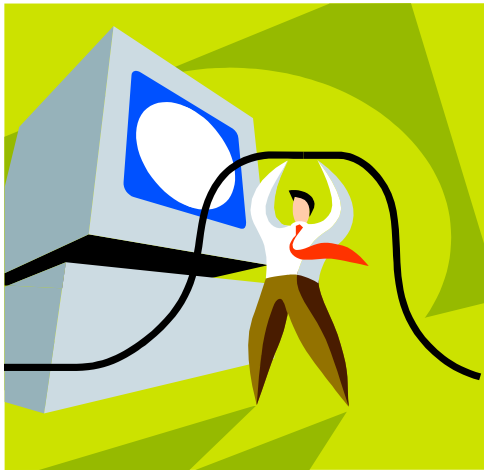
Improving Software Economics

Reducing Software Product Size

"The most significant way

to improve affordability and return on investment is usually to produce a product that achieves the design goals with the minimum amount of human-generated source material."

Reuse, object-oriented technology, automatic code production, and higher order programming languages are all focused on achieving a given system with fewer lines of human-specified source directives.



Part 1

Improving Software Economics

Reducing Software Product Size - Languages

UFP -Universal Function Points
The basic units of the function points are external user inputs, external outputs, internal logic data groups, external data interfaces, and external inquiries.

Language	SLOC per UFP
Assembly	320
C	128
Fortran 77	105
Cobol 85	91
Ada 83	71
C++	56
Ada 95	55
Java	55
Visual Basic	35

SLOC metrics
are useful estimators for software after a candidate solution is formulated and an implementation language is known.

Part 1

Improving Software Economics

Reducing Software Product Size – Object-Oriented Methods

- *"An object-oriented model of the problem and its solution encourages a common vocabulary between the end users of a system and its developers, thus creating a shared understanding of the problem being solved."*

  Here is an example of how object-oriented technology permits corresponding improvements in teamwork and interpersonal communications.

- *"The use of continuous integration creates opportunities to recognize risk early and make incremental corrections without destabilizing the entire development effort."*

  This aspect of object-oriented technology enables an architecture-first process, in which integration is an early and continuous life-cycle activity.

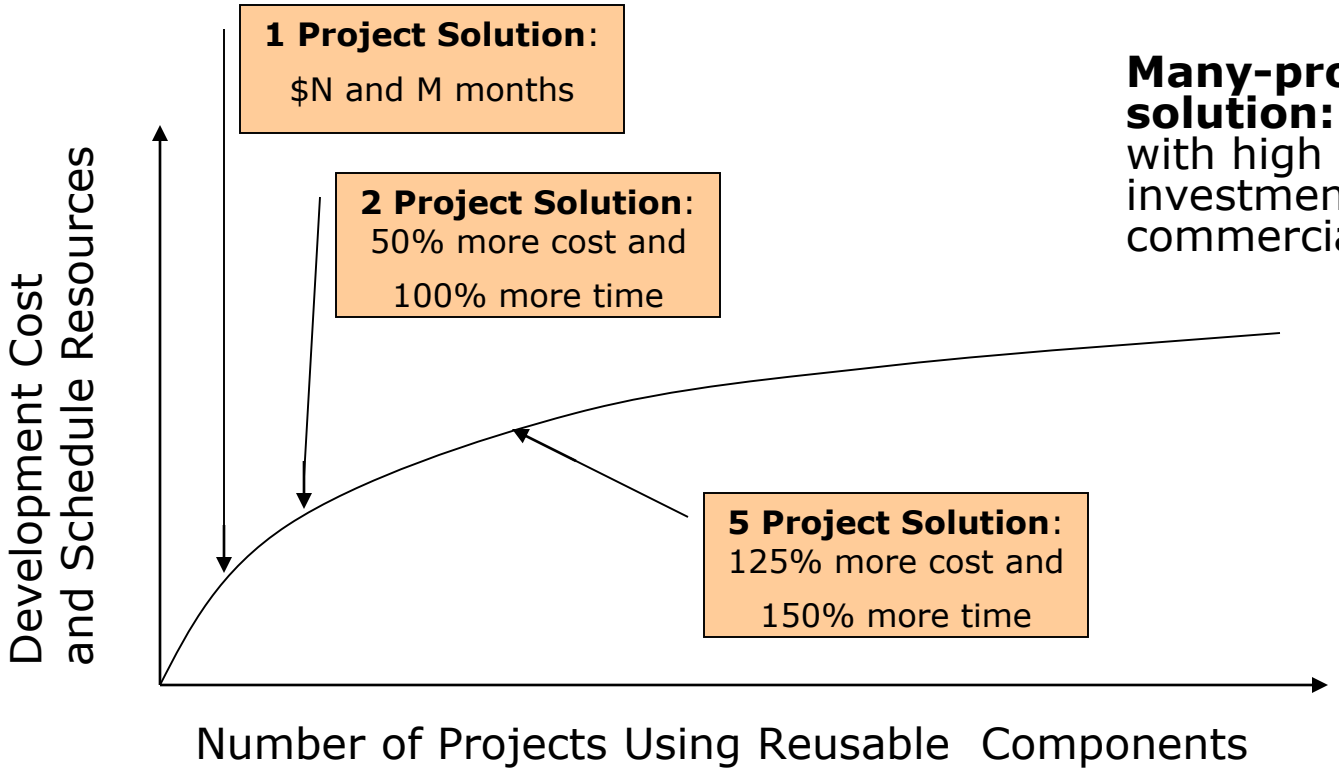
- *An object-oriented architecture provides a clear separation of concerns among disparate elements of a system, creating firewalls that prevent a change in one part of the system from rending the fabric of the entire architecture."*

  This feature of object-oriented technology is crucial to the supporting languages and environments available to implement object-oriented architectures.

Part 1

Improving Software Economics

Reducing Software Product Size – Reuse



Many-project solution: Operating with high value per unit investment, typical of commercial products

Part 1

Improving Software Economics

Reducing Software Product Size – Commercial Components

APPROACH	ADVANTAGES	DISADVANTAGES
Commercial components	Predictable license costs Broadly used, mature technology Available now Dedicated support organization Hardware/software independence Rich in functionality	Frequent upgrades Up-front license fees Recurring maintenance fees Dependency on vendor Run-time efficiency sacrifices Functionality constraints Integration not always trivial No control over upgrades and maintenance Unnecessary features that consume extra resources Often inadequate reliability and stability Multiple-vendor incompatibility
Custom development	Complete change freedom Smaller, often simpler implementations Often better performance Control of development and enhancement	Expensive, unpredictable development Unpredictable availability date Undefined maintenance model Often immature and fragile Single-platform dependency Drain on expert resources

Part 1

Improving Software Economics

Improving Software Processes

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

Three levels of processes and their attributes