Staffing Configuration Management

Define project management

- **Software project management** is the art and science of planning and leading software projects.[1] It is a sub-discipline ofproject management in which software projects are planned, implemented, monitored and controlled.

11.1 What is Project Management?

- Project management encompasses all the activities needed to plan and execute a project:
  - Deciding what needs to be done
  - Estimating costs
  - Ensuring there are suitable people to undertake the project
  - Defining responsibilities
  - Scheduling
  - Making arrangements for the work
  - *continued ...*

What is Project Management?

- Directing
- Being a technical leader
- Reviewing and approving decisions made by others
- Building morale and supporting staff
- Monitoring and controlling
- Co-ordinating the work with managers of other projects
- Reporting
- Continually striving to improve the process

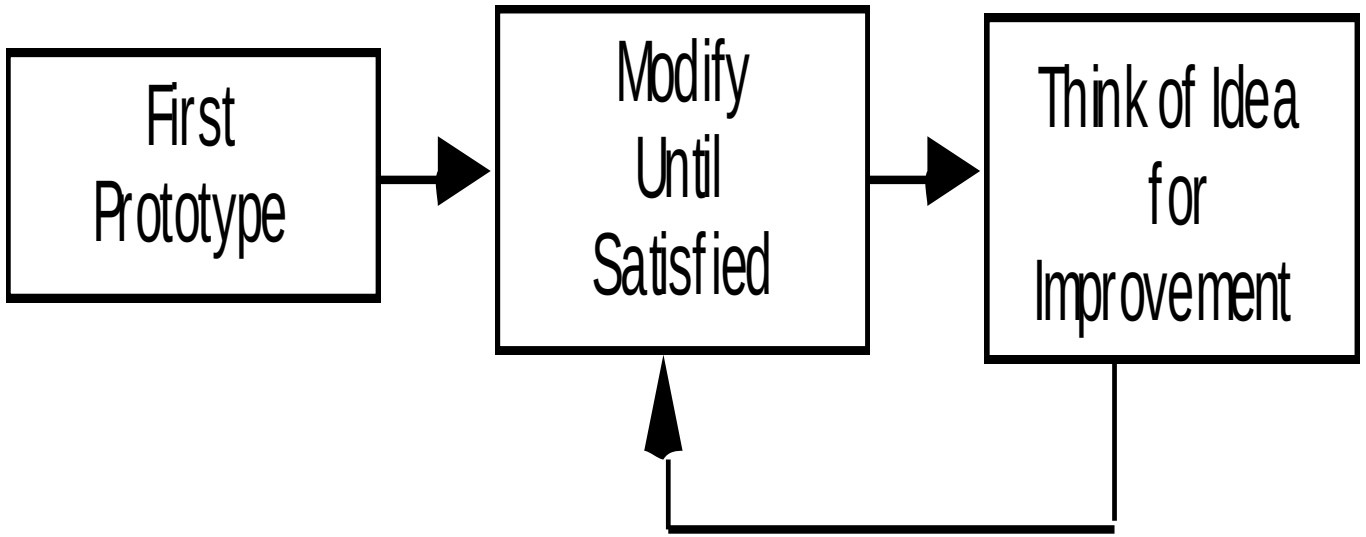
Process & Functionality & Mechanism,

- Software process models are general approaches for organizing a project into activities.
  - Help the project manager and his or her team to decide:
    - What work should be done;
    - In what sequence to perform the work.
  - The models should be seen as *aids to thinking*, not rigid prescriptions of the way to do things.
  - Each project ends up with its own unique plan.

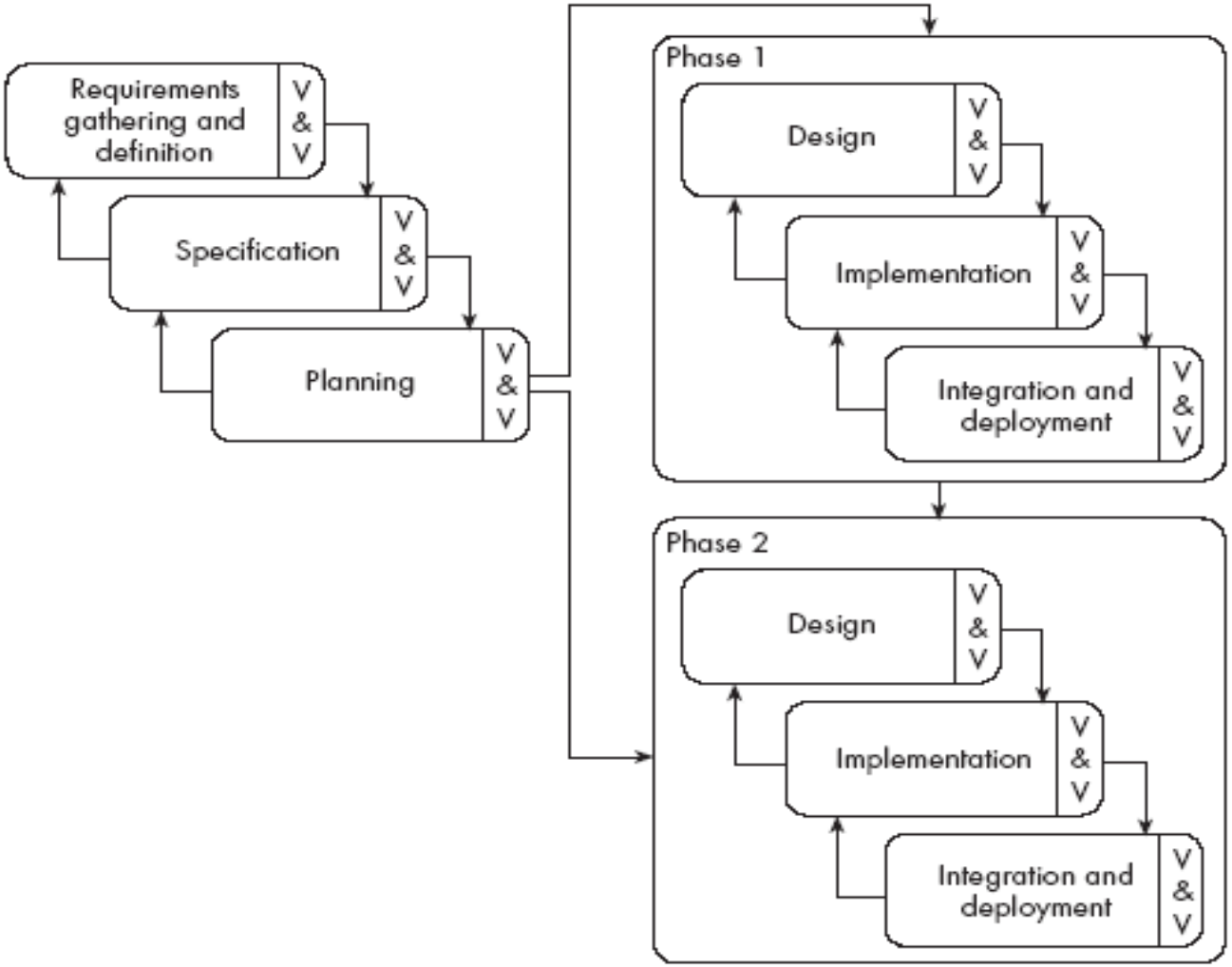

Sub Process Correlations in
Project Management Process
Project
Initiation
Project
Planning
Project Tracking
& Oversight
Risk
Management
Project Closing
Management

Process management

- **Process management** is the ensemble of activities of planning and monitoring the performance of a process. The term usually refers to the management of business processes and manufacturing processes
- Process management is the application of knowledge, skills, tools, techniques and systems to define, visualize, measure, control, report and improve processes with the goal to meet customer requirements profitably. It can be differentiated fromprogram management in that program management is concerned with managing a group of inter-dependent projects. But from another viewpoint, process management includes program management. In project management, process management is the use of a repeatable process to improve the outcome of the projec


First Prototype
Modify Until Satisfied
Think of Idea for Improvement

The phased-release model

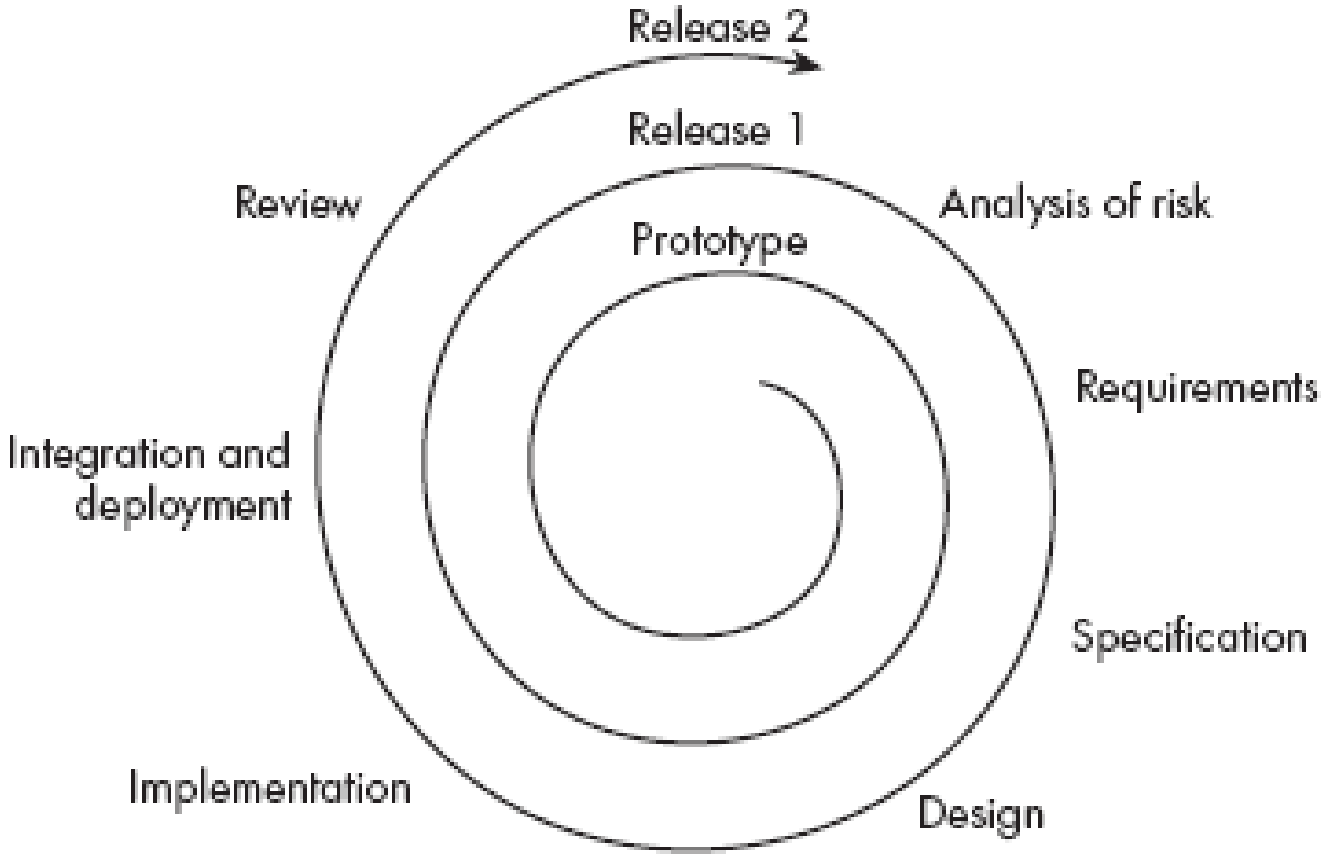


- It introduces the notion of *incremental* development.
  - After requirements gathering and planning, the project should be broken into separate subprojects, or *phases*.
  - Each phase can be released to customers when ready.
  - Parts of the system will be available earlier than when using a strict waterfall approach.
  - However, it continues to suggest that all requirements be finalized at the start of development.

Choosing a process model

- From the waterfall model:
  - Incorporate the notion of stages.
- From the phased-release model:
  - Incorporate the notion of doing some initial high-level analysis, and then dividing the project into releases.
- From the spiral model:
  - Incorporate prototyping and risk analysis.
- From the evolutionary model:
  - Incorporate the notion of varying amounts of time and work, with overlapping releases.
- From concurrent engineering:
  - Incorporate the notion of breaking the system down into components and developing them in parallel.

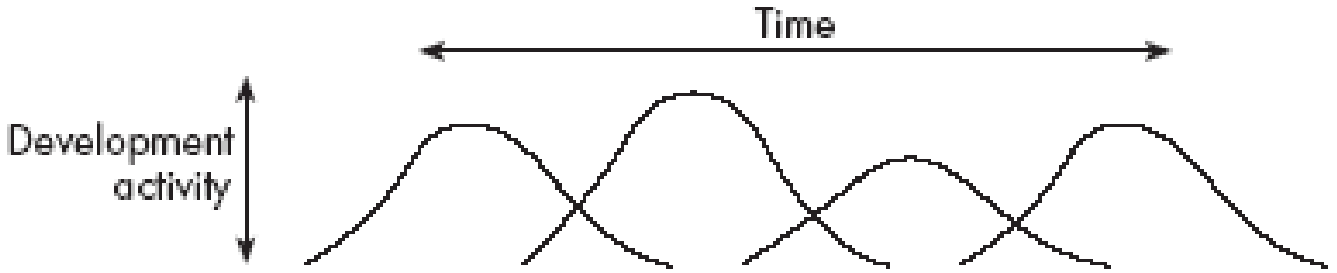
The spiral model



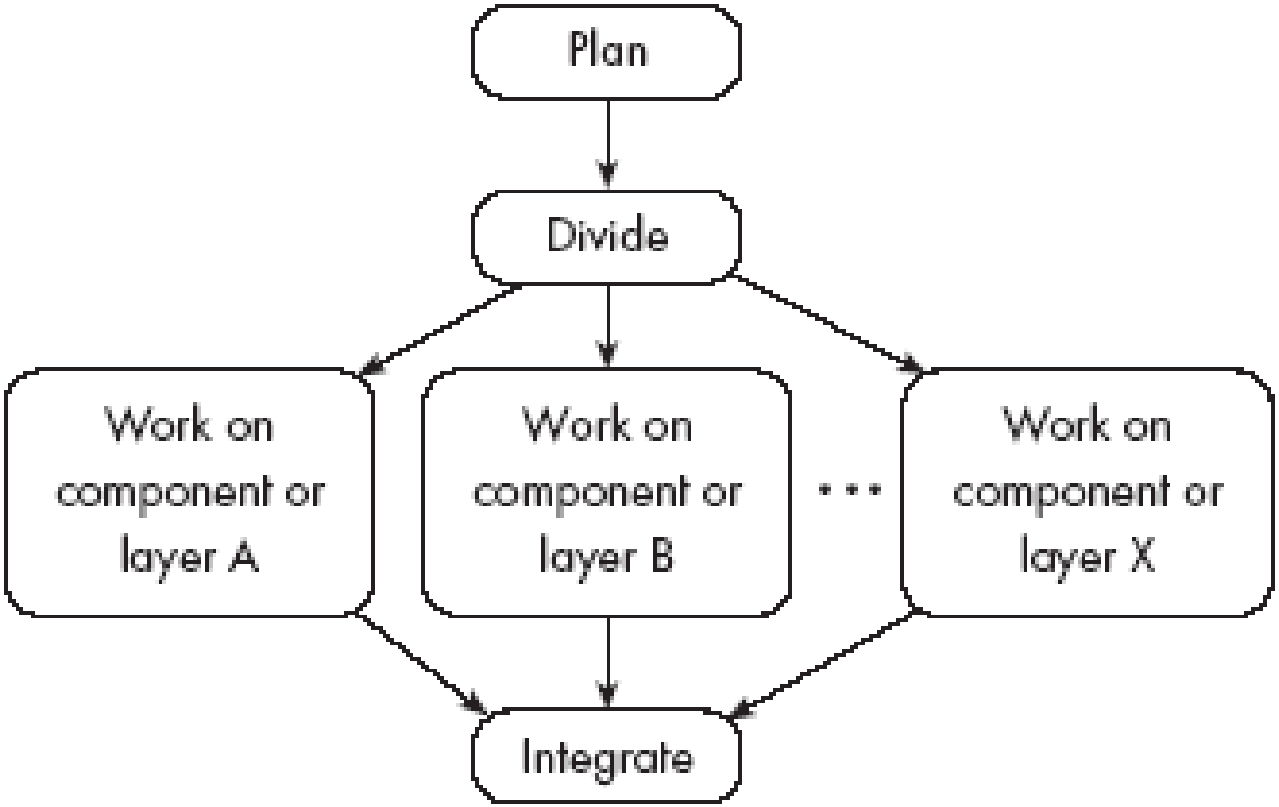
The spiral model

- It explicitly embraces prototyping and an *iterative* approach to software development.
  - Start by developing a small prototype.
  - Followed by a mini-waterfall process, primarily to gather requirements.
  - Then, the first prototype is reviewed.
  - In subsequent loops, the project team performs further requirements, design, implementation and review.
  - The first thing to do before embarking on each new loop is risk analysis.
  - Maintenance is simply a type of on-going development.

The evolutionary model



The concurrent engineering model



Choosing a process model

- From the waterfall model:
  - Incorporate the notion of stages.
- From the phased-release model:
  - Incorporate the notion of doing some initial high-level analysis, and then dividing the project into releases.
- From the spiral model:
  - Incorporate prototyping and risk analysis.
- From the evolutionary model:
  - Incorporate the notion of varying amounts of time and work, with overlapping releases.
- From concurrent engineering:
  - Incorporate the notion of breaking the system down into components and developing them in parallel.

11.3 Cost estimation

- To estimate how much software-engineering time will be required to do some work.
  - *Elapsed time*
    - The difference in time from the start date to the end date of a task or project.
  - *Development effort*
    - The amount of labour used in *person-months* or *person-days.*
    - To convert an estimate of development effort to an amount of money:

      You multiply it by the *weighted average cost* (*burdened* cost) of employing a software engineer for a month (or a day).

Principles of effective cost estimation

- Principle 2: Include all activities when making estimates.
  - The time required for *all* development activities must be taken into account.
  - Including:
    - Prototyping
    - Design
    - Inspecting
    - Testing
    - Debugging
    - Writing user documentation
    - Deployment.

Principles of effective cost estimation

- Principle 4: Be sure to account for *differences* when extrapolating from other projects.
  - Different software developers
  - Different development processes and maturity levels
  - Different types of customers and users
  - Different schedule demands
  - Different technology
  - Different technical complexity of the requirements
  - Different domains
  - Different levels of requirement stability

Principles of effective cost estimation

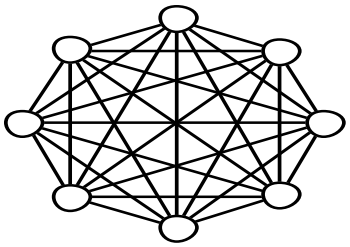
- Principle 4: Be sure to account for *differences* when extrapolating from other projects.
  - Different software developers
  - Different development processes and maturity levels
  - Different types of customers and users
  - Different schedule demands
  - Different technology
  - Different technical complexity of the requirements
  - Different domains
  - Different levels of requirement stability

Staffing Plan

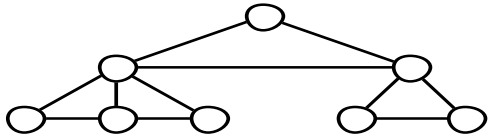
- *Describe*the numbers and types of personnel needed to conduct the project. Describe the required skill levels, start times, duration on the project, method of obtaining the personnel, training required, and phasing out of project personnel.

11.4 Building Software Engineering Teams

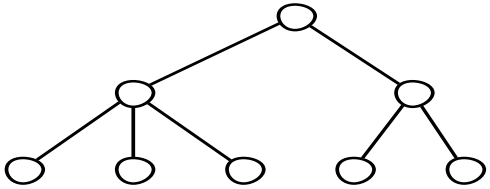
- Software engineering is a human process.
  - Choosing appropriate people for a team, and assigning roles and responsibilities to the team members, is therefore an important project management skill
  - Software engineering teams can be organized in many different ways



a) Egoless



b) Chief programmer



c) Strict hierarchy

Software engineering teams

- Egoless team:
  - In such a team everybody is equal, and the team works together to achieve a common goal.
  - Decisions are made by consensus.
  - Most suited to difficult projects with many technical challenges.

Software engineering teams

- Hierarchical manager-subordinate structure:
  - Each individual reports to a manager and is responsible for performing the tasks delegated by that manager.
  - Suitable for large projects with a strict schedule where everybody is well-trained and has a well-defined role.
  - However, since everybody is only responsible for their own work, problems may go unnoticed.

Software engineering teams

- Chief programmer team:
  - Midway between egoless and hierarchical.
  - The chief programmer leads and guides the project.
  - He or she consults with, and relies on, individual specialists.

Choosing an effective size for a team

- For a given estimated development effort, in person months, there is an optimal team size.
  - Doubling the size of a team will not halve the development time.
- Subsystems and teams should be sized such that the total amount of required knowledge and exchange of information is reduced.
- For a given project or project iteration, the number of people on a team will not be constant.
- You can not generally add people if you get behind schedule, in the hope of catching up.

Technical Process

- This section describes the top-level technical processes used on the project including the technical methods, tools, and techniques; major software documents; and supporting activities such as configuration management and quality assurance.

- **Methods, Tools, and Techniques**Describe the following:
- The computing system environment including hardware and operating system environment;
- Software tools including design tools, source code control, time accounting, compiler or IDE, debugging aids, defect tracking, and so on
- Development methodologies including requirements development practices, design methodologies and notations, programming language, coding standards, documentation standards, system integration procedure, and so on

Software Documentation

- Change Control Plan
- Change Proposals
- Vision statement
- Top 10 Risks List
- Software Development Plan, including project cost and schedule estimates
- User Manual/Requirements Specification
- Quality Assurance Plan
- Software Architecture
- Staged Delivery Plan
- Individual Stage Plans, including miniature milestone schedules
- Coding Standard
- Detailed Design Documents
- Software Construction Plans

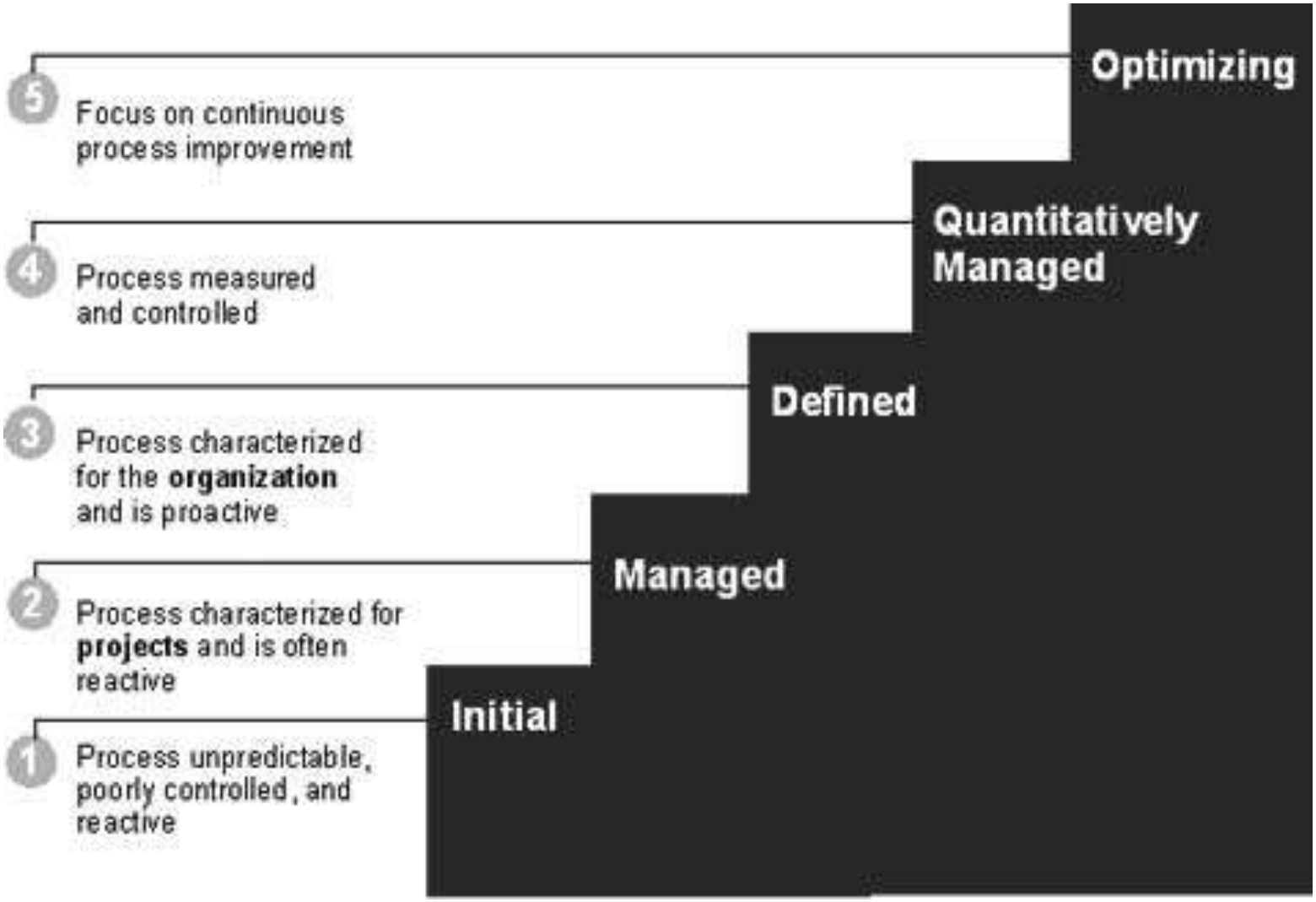
Capability Maturity Model (CMM)[

- he **Capability Maturity Model (CMM)**[1] is a development model created after study of data collected from organizations
- The term "maturity" relates to the degree of formality and optimization of processes to formally defined steps, to managed result metrics, to active optimization of the processes.
- The model's aim is to improve existing software-development processes

- The CMM model's application in software development has sometimes been problematic. Applying multiple models that are not integrated within and across an organization could be costly in training, appraisals, and improvement activities. TheCapability Maturity Model Integration (CMMI) project was formed to sort out the problem of using multiple models for software development processes, thus the CMMI model has superseded the CMM model, though the CMM model continues to be a general theoretical process capability model used in the public domain

maturity model

- A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. Each maturity level provides a layer in the foundation for continuous process improvement.
- In CMMI models with a staged representation, there are five maturity levels designated by the numbers 1 through 5
- Initial
- Managed
- Defined
- Quantitatively Managed
- Optimizing


5
Focus on continuous process improvement
Optimizing
4
Process measured and controlled
Quantitatively Managed
3
Process characterized for the **organization** and is proactive
Defined
2
Process characterized for **projects** and is often reactive
Managed
1
Process unpredictable, poorly controlled, and reactive
Initial

Maturity Level 1 - Initial

- At maturity level 1, processes are usually ad hoc and chaotic. The organization usually does not provide a stable environment. Success in these organizations depends on the competence and heroics of the people in the organization and not on the use of proven processes.
- Maturity level 1 organizations often produce products and services that work; however, they frequently exceed the budget and schedule of their projects.

Maturity Level 2 - Managed

- At maturity level 2, an organization has achieved all the **specific** and **generic goals** of the maturity level 2 process areas. In other words, the projects of the organization have ensured that requirements are managed and that processes are planned, performed, measured, and controlled.

Maturity Level 3 - Defined

- A critical distinction between maturity level 2 and maturity level 3 is the scope of standards, process descriptions, and procedures. At maturity level 2, the standards, process descriptions, and procedures may be quite different in each specific instance of the process (for example, on a particular project). At maturity level 3, the standards, process descriptions, and procedures for a project are tailored from the organization's set of standard processes to suit a particular project or organizational unit. The organization's set of standard processes includes the processes addressed at maturity level 2 and maturity level 3.

Maturity Level 4 - Quantitatively Managed

- At maturity level 4 Subprocesses are selected that significantly contribute to overall process performance. These selected subprocesses are controlled using statistical and other quantitative techniques.
- Quantitative objectives for quality and process performance are established and used as criteria in managing processes. Quantitative objectives are based on the needs of the customer, end users, organization, and process implementers. Quality and process performance are understood in statistical terms and are managed throughout the life of the processes.
- For these processes, detailed measures of process performance are collected and statistically analyzed. Special causes of process variation are identified and, where appropriate, the sources of special causes are corrected to prevent future occurrences.

Maturity Level 5 - Optimizing

- Processes are continually improved based on a quantitative understanding of the common causes of variation inherent in processes.
- Maturity level 5 focuses on continually improving process performance through both incremental and innovative technological improvements.
- Quantitative process-improvement objectives for the organization are established, continually revised to reflect changing business objectives, and used as criteria in managing process improvement.
- The effects of deployed process improvements are measured and evaluated against the quantitative process-improvement objectives. Both the defined processes and the organization's set of standard processes are targets of measurable improvement activities.