QUALITY MANAGEMENT

Software Quality Management

Software Quality

- Quality software refers to software that is reasonably bug or defect-free, is delivered on time and within the specified budget, meets the requirements and/or expectations, and is maintainable.
- In the software engineering context, software quality reflects both functional qualities as well as a structural qualities.

User satisfaction = compliant product + good quality + delivery within budget and schedule

"Robert Glass [Gla98]"

Software Quality Management

Quality Dimensions

David Garvin [Gar87]:

- Performance Quality. Does the software deliver all content, functions, and features that are specified as part of the requirements model in a way that provides value to the end user?
- Feature quality. Does the software provide features that surprise and delight first-time end-users?
- Reliability. Does the software deliver all features and capabilities without failure? Is it available when it is needed? Does it deliver functionality that is error-free?
- Conformance. Does the software conform to local and external software standards that are relevant to the application? Does it conform to the de facto design and coding conventions? For example, does the user interface conform to accepted design rules for menu selection or data input?

Software Quality Management

Quality Dimensions Cont'd

David Garvin [Gar87]:

- Durability. Can the software be maintained (changed) or corrected (debugged) without the inadvertent generation of unintended side effects? Will changes cause the error rate or reliability to degrade with time?
- Serviceability. Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period? Can support staff acquire all information they need to make changes or correct defects?
- Aesthetics. Most of us would agree that an aesthetic entity has a certain elegance, a unique flow, and an obvious "presence" that are hard to quantify but evident nonetheless.
- Perception. In some situations, you have a set of prejudices that will influence your perception of quality.

Software Quality Factors

Software Quality Factors

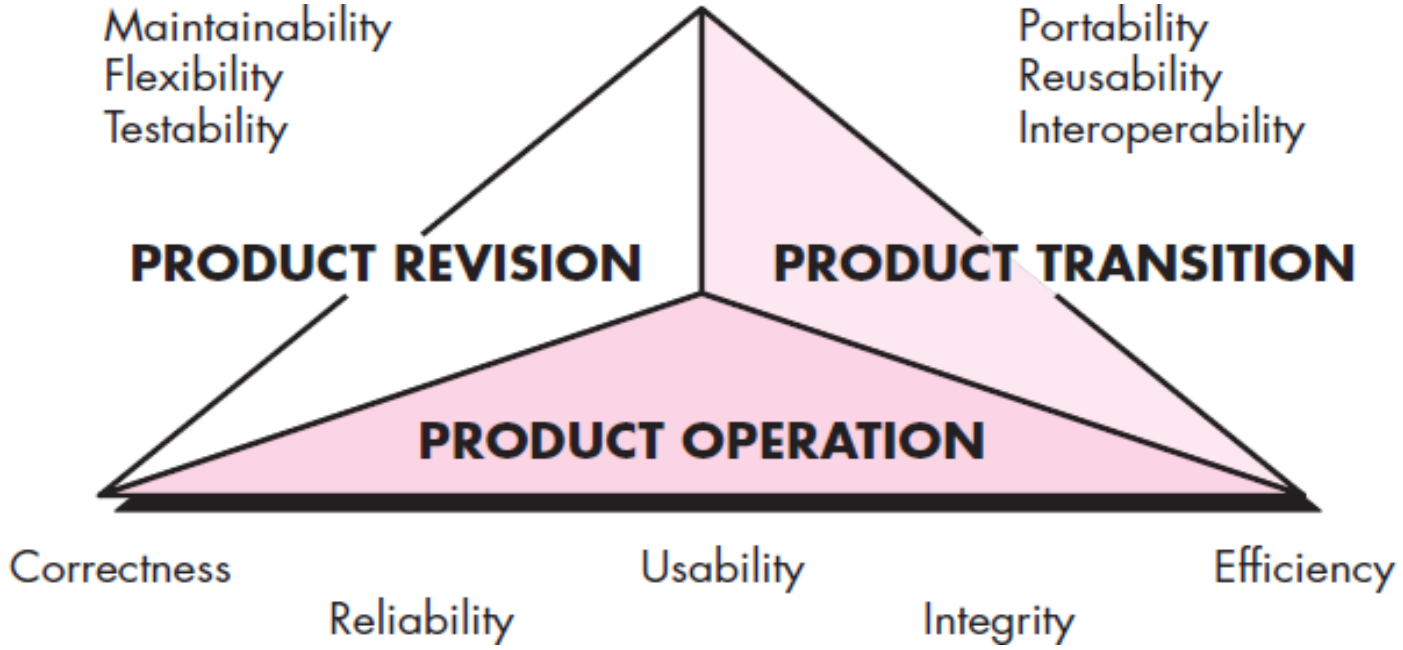
- The various factors, which influence the software, are termed software factors.
- They can be broadly divided into two categories.
- The first category factors of are those that can be measured directly such as the number of logical errors.
- The second category clubs those factors which can be measured only indirectly.

- McCall's Factor Model
- ISO 9126 Quality Factors

Software Quality Factors

McCall's Factor Model

- This model classifies all software requirements into 11 software quality factors.
- The 11 factors are grouped into three categories – product operation, product revision, and product transition factors



Software Quality Factors

McCall's Factor Model Cont'd

1. Product Operation Software Quality Factors

According to McCall's model, the product operation category includes five software quality factors, which deal with the requirements that directly affect the daily operation of the software.

Correctness

- These requirements deal with the correctness of the output of the software system.
- Output mission - The required accuracy of output can be negatively affected by inaccurate data or inaccurate calculations.
- The completeness of the output information can be affected by incomplete data.
- The up-to-dateness of the information is defined as the time between the event and the response by the software system.
- The availability of the information.

Software Quality Factors

McCall's Factor Model Cont'd

1. Product Operation Software Quality Factors Cont'd

- **Reliability**

  Reliability requirements deal with a service failure. They determine the maximum allowed failure rate of the software system, and can refer to the entire system or to one or more of its separate functions.

- **Efficiency**

  It deals with the hardware resources needed to perform the different functions of the software system. It includes processing capabilities (given in MHz), its storage capacity (given in MB or GB) and data communication capability (given in MBPS or GBPS).

- **Integrity**

  This factor deals with the software system security, that is, to prevent access to unauthorized persons, and also to distinguish between the group of people to be given read as well as write permits.

- **Usability**

  Usability requirements deal with the staff resources needed to train a new employee and to operate the software system.

Software Quality Factors

McCall's Factor Model Cont'd

2. Product Revision Quality Factors

- Maintainability

This factor considers the efforts that will be needed by users and maintenance personnel to identify the reasons for software failures, correct the failures, and verify the success of the corrections.

- Flexibility

This factor deals with the capabilities and efforts required to support adaptive maintenance activities of the software. These include adapting the current software to additional circumstances and customers without changing the software.

- Testability

Testability requirements deal with the testing of the software system as well as with its operation. It includes predefined intermediate results, log files, and also the automatic diagnostics performed by the software system prior to starting the system, to find out whether all components of the system are in working order and to obtain a report about the detected faults.

Software Quality Factors

McCall's Factor Model Cont'd

3. Product Transition Software Quality Factor

- Portability

Portability requirements tend to the adaptation of a software system to other environments consisting of different hardware, different operating systems, and so forth. The software should be possible to continue using the same basic software in diverse situations.

- Reusability

This factor deals with the use of software modules originally designed for one project in a new software project currently being developed. They may also enable future projects to make use of a given module or a group of modules of the currently developed software. The reuse of software is expected to save development resources, shorten then development period, and prove higher-quality modules.

- Interoperability

Interoperability requirements focus on creating interfaces with other software systems or with other equipment firmware.

Software Quality Factors

ISO 9126 Quality Factors

The ISO 9126 standard was developed in an attempt to identify the key quality attributes of computer software. The standard identifies six key quality attributes.

- Functionality.

The degree to which the software satisfies stated needs as indicated by the following sub-attributes: suitability, accuracy, interoperability, compliance, and security.

- Reliability.

The amount of time that the software is available for use as indicated by the following sub-attributes: maturity, fault tolerance, and recoverability.

- Usability.

The degree to which the software is easy to use is indicated by the following sub-attributes: understandability, learnability, and operability.

Software Quality Factors

ISO 9126 Quality Factors Con't

Efficiency.

The degree to which the software makes optimal use of system resources is indicated by the following sub attributes: time behavior, and resource behavior.

Maintainability.

The ease with which repair may be made to the software is indicated by the following sub-attributes: analyzability, changeability, stability, and testability.

Portability.

The ease with which the software can be transposed from one environment to another is indicated by the following sub attributes: adaptability, installability, conformance, and replaceability.

Software Quality Assurance (SQA)

Software Quality Assurance (SQA)

- Software Quality Assurance (SQA) is a set of activities for ensuring quality in software engineering processes.
- It ensures that developed software meets and complies with the defined or standardized quality specifications.
- SQA is an ongoing process within the Software Development Life Cycle (SDLC) that routinely checks the developed software to ensure it meets the desired quality measures

It includes the following activities:

- Process definition and implementation
- Auditing
- Training

Software Quality Assurance (SQA)

Software Quality Assurance (SQA)

Processes could be:

- Software Development Methodology
- Project Management
- Configuration Management
- Requirements Development/Management
- Estimation
- Software Design
- Testing, etc.

Once the processes have been defined and implemented, Quality Assurance has the following responsibilities:

- Identify the weaknesses in the processes.
- Correct those weaknesses to continually improve the process.

Software Quality Assurance (SQA)

Components of SQA System

- Pre-project components
- Components of project life cycle activities assessment
- Components of infrastructure error prevention and improvement
- Components of software quality management
- Components of standardization, certification, & SQA system assessment
- Organizing for SQA – the human components

Capability Maturity Model (CMM Model)

Maturity Models

- A maturity model is a structured collection of elements that describe characteristics of effective processes.

A maturity model provides

  - a place to start
  - the benefit of a community's prior experiences
  - a common language and a shared vision
  - a framework for prioritizing actions
  - a way to define what improvement means for your organization

- A maturity model can be used as a benchmark for assessing different organizations for equivalent comparison.

- A Capability Maturity Model (CMM) is a reference model of mature practices in a specified discipline, used to improve and appraise a group's capability to perform that discipline.

Capability Maturity Model (CMM Model)

CMM

- CMM: Capability Maturity Model
- Developed by the Software Engineering Institute of the Carnegie Mellon University
- The framework describes the key elements of an effective software process.

The maturity level of an organization in terms of its capability to build high-quality software within a short turn-around time with optimal cost.

Capability Maturity Model (CMM Model)

Motivation:

- Organizations should understand its current standing
- Develop a plan to improve the process
- Incrementally introduce changes
- Give time to internalize and truly follow new methods
- Order in which the improvement steps should be taken.

- Describes an evolutionary improvement path for software organizations from an ad hoc, immature process to a mature, disciplined one.
- Provides guidance on how to gain control of processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence

Capability Maturity Model (CMM Model)

CMM Levels

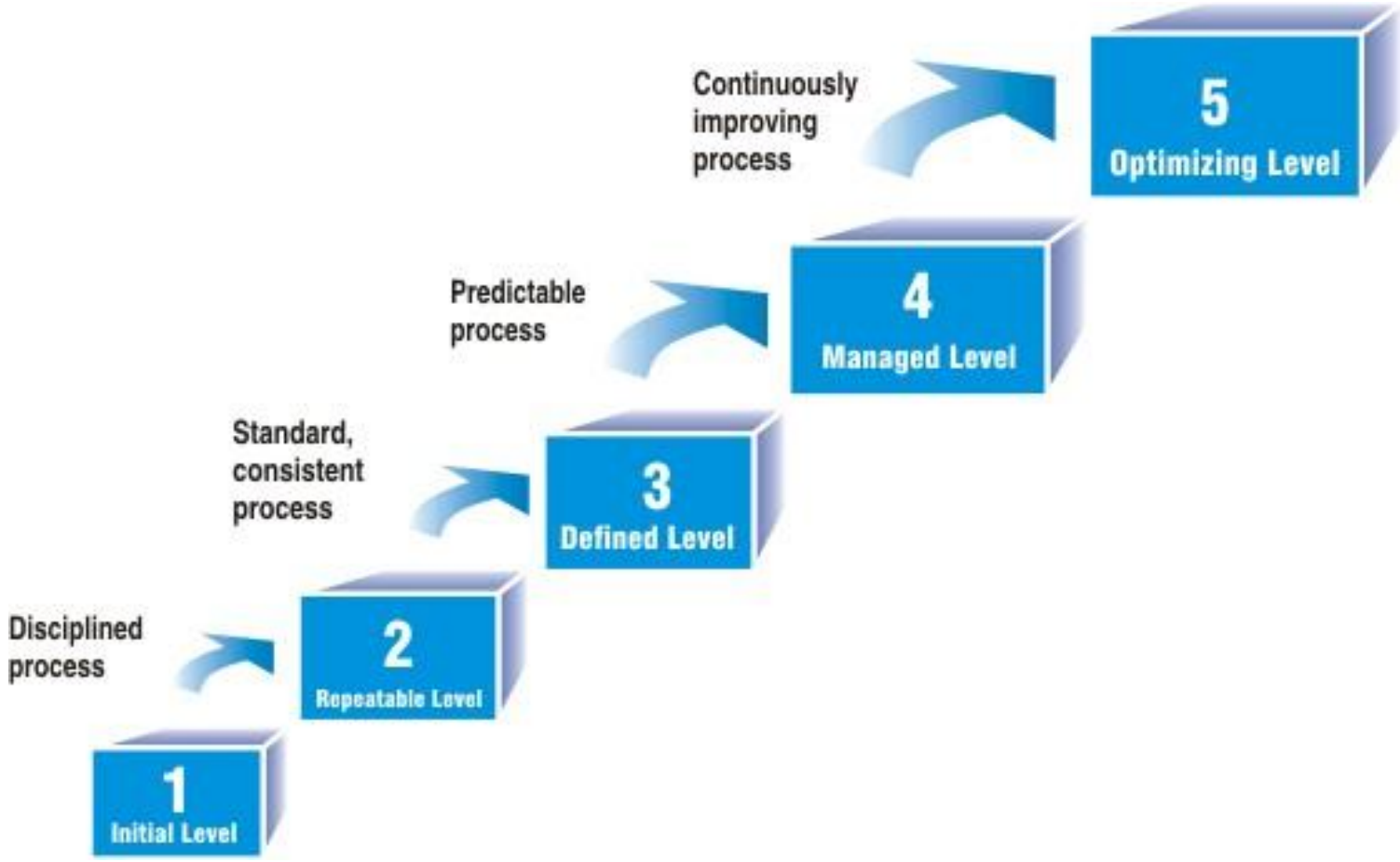
- Initial
- Repeatable

  disciplined
- Defined

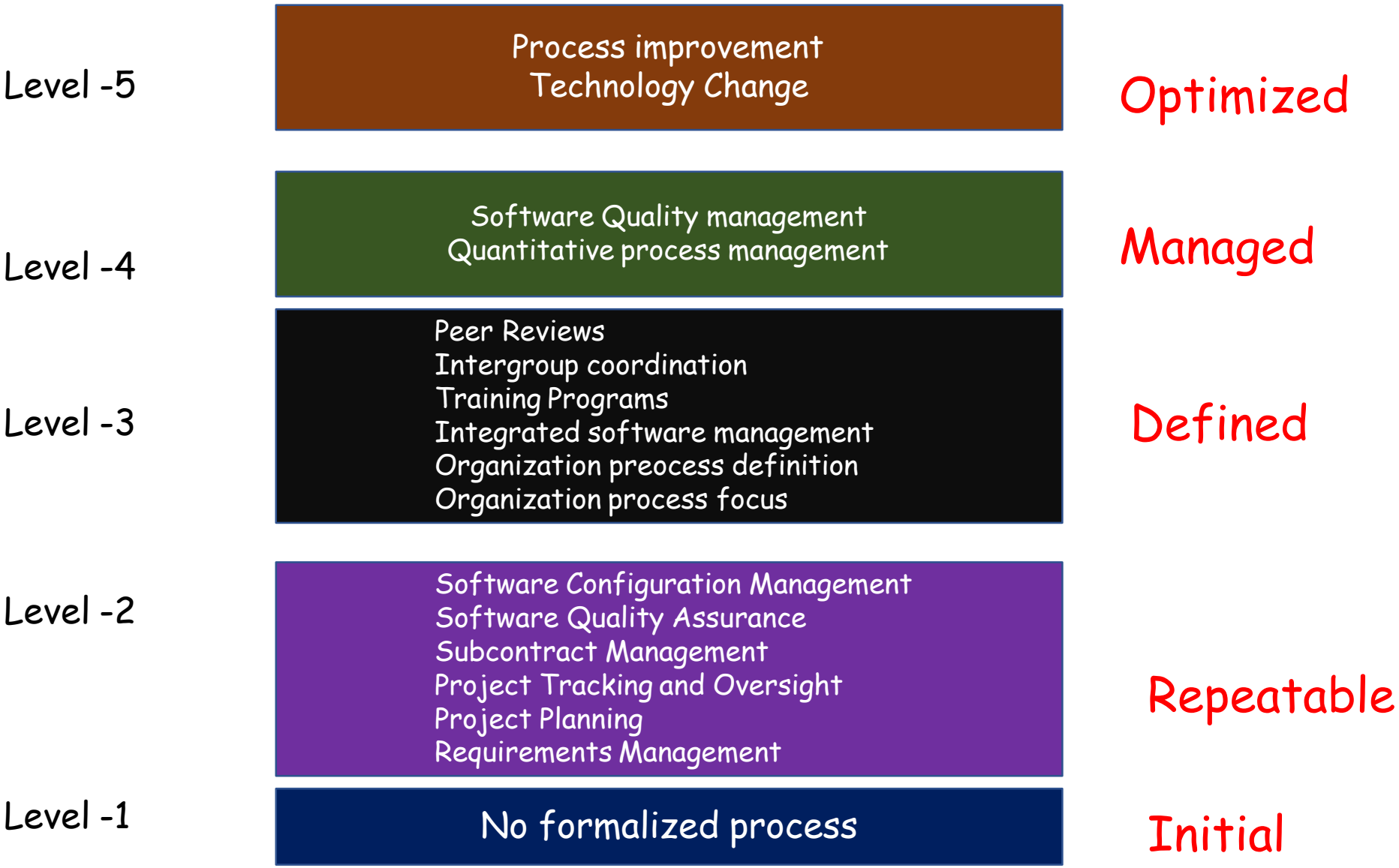
  Standard, consistent
- Managed

  predictable
- Optimized

  Continuously improving



Capability Maturity Model (CMM Model)



Capability Maturity Model (CMM Model)

Level 1: Initial

The software process is characterized as ad hoc. Few processes are defined, and success depends on individual effort.

- At this level, frequently have difficulty making commitments that they staff can meet with an orderly process
- Products developed are often over budget and schedule.
- Wide variations in cost, schedule, functionality and quality targets
- Capability is a characteristic of the individuals, not of the organization

Capability Maturity Model (CMM Model)

Level 2: Repeatable

- Basic process management processes are established to track cost, schedule, and functionality. The necessary process

discipline is in place to repeat earlier successes on projects with similar applications.

- Realistic project commitments based on results observed on previous projects
- Software project standards are defined and faithfully followed
- Processes may differ between projects
- Process is disciplined
- Earlier successes can be repeated

Level 3: Defined

- The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization.

Capability Maturity Model (CMM Model)

Level 4: Managed

Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

- Narrowing the variation in process performance to fall within acceptable quantitative bounds
- When known limits are exceeded, corrective action can be taken
- Quantifiable and predictable

  - predict trends in process and product quality

Level 5: Optimizing

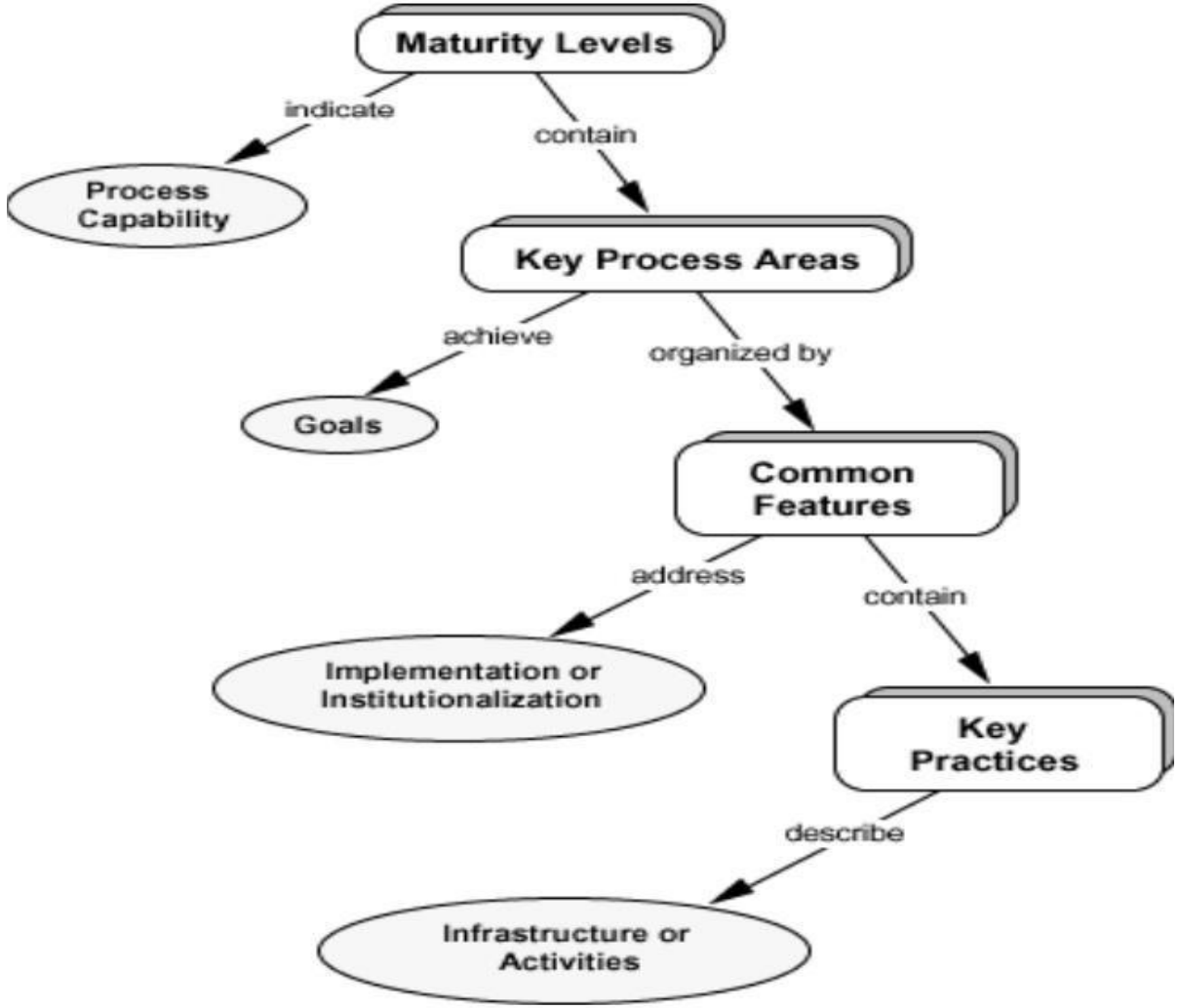
Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

- Detect and remove the cause of defects early
- Identify and deploy new tools and process improvements to meet needs and business objectives.
- Fosters Organizational Innovation and Deployment
- Gives impetus to Causal Analysis and Resolution

Capability Maturity Model (CMM Model)

Internal Structure to Maturity Levels

- Except for level 1, each level is decomposed into key process areas (KPA)
- Each KPA identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing software capability.
  - Commitment
  - Ability
  - Activity
  - Measurement
  - Verification



CMM Integration (CMMI Model)

CMM Integration

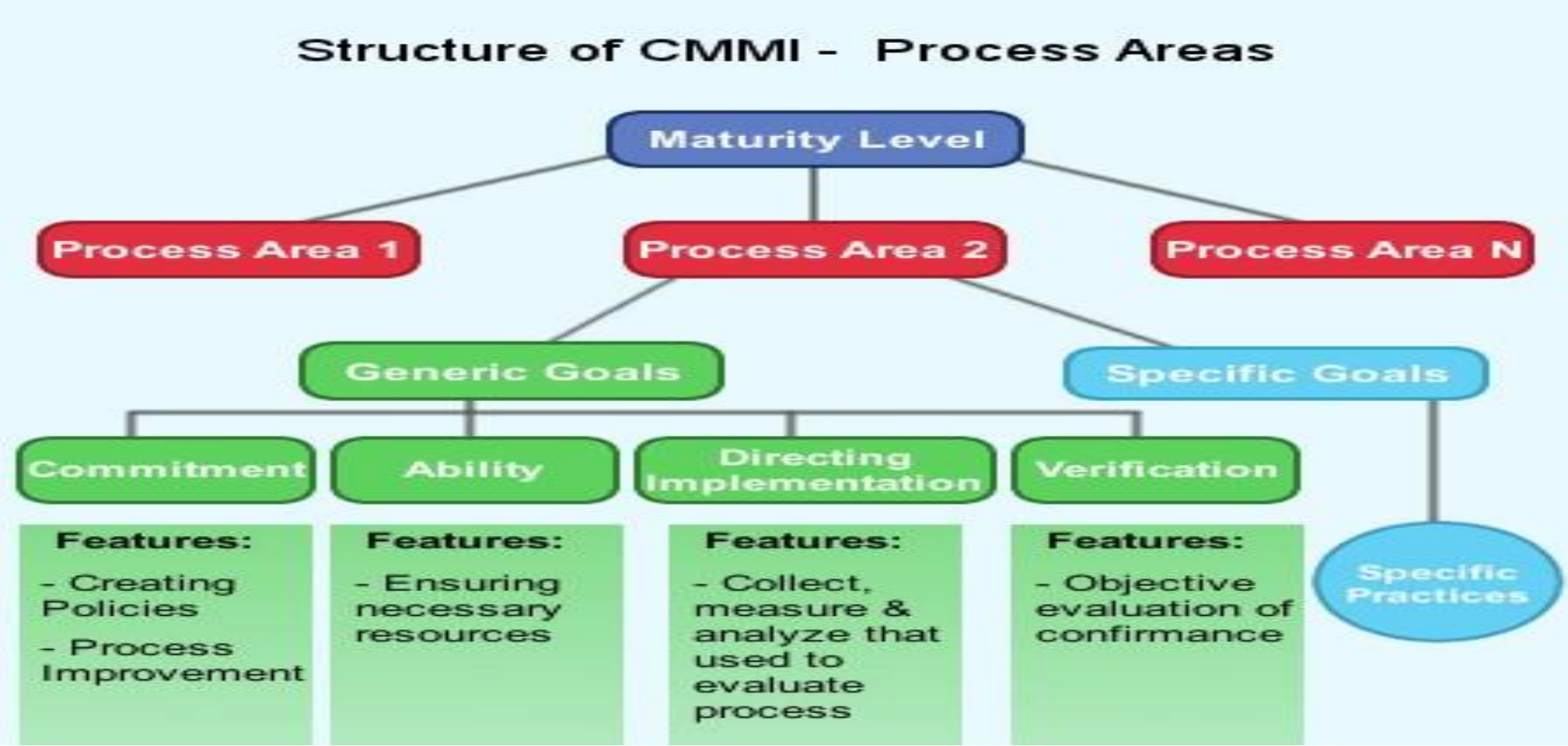
- The CMMI framework is the current stage of work on process assessment and improvement that started at the software engineering institute in the 1980s.
- A capability level is a well-defined evolutionary table describing the organization's capability relative to a process area.
- A capability level consists of related specific and generic practices for a process area that can improve the organization's processes associated with that process area. Each level is a layer in the foundation for continuous process improvement.

The CMMI model is divided into five maturity levels:

1. Initial
2. Managed
3. Defined
4. Quantitatively Managed
5. Optimizing

The main difference between CMM (Capability Maturity Model) and CMMI (Capability Maturity Model Integration) is that the former focuses on evaluating whether an organization completes specific tasks related to the process or not, while the latter focuses on building an architecture for the whole development process.

CMM Integration (CMMI Model)



CMM Integration (CMMI Model)

The CMMI Process Areas (PAs) can be grouped into the following four categories.

- Process Management
- Project Management
- Engineering
- Support

Category	Process area
Process Management	• Organizational process definition • Organizational process focus • Organizational training • Organizational process performance • Organizational innovation and development
Project management	• Project planning • Project monitoring and control supplier agreement management • Integrated project management • Risk management • Integrated testing • Quantitative project management

Difference between CMM and CMMI

Parameters of Comparison	CMM	CMMI
Definition	CMM stands for Capability Maturity Model.	CMMI stands for Capability Maturity Model Integration.
Meaning	It is a behavior model developed to measure an organization's software development process.	It is a successor of the CMM model and is more effective and task-oriented.
Purpose	To evaluate the process maturity levels in software.	To combine many software models into one and to overcome the drawbacks of CMM.
Stages	This model has five stages: • Initial • Repeat • Defined • Managed • Optimized.	1. Initial 2. Managed 3. Defined 4. Quantitatively Managed 5. Optimizing
Efficiency	Less effective one	More effective one