

The Process

Software Engineering

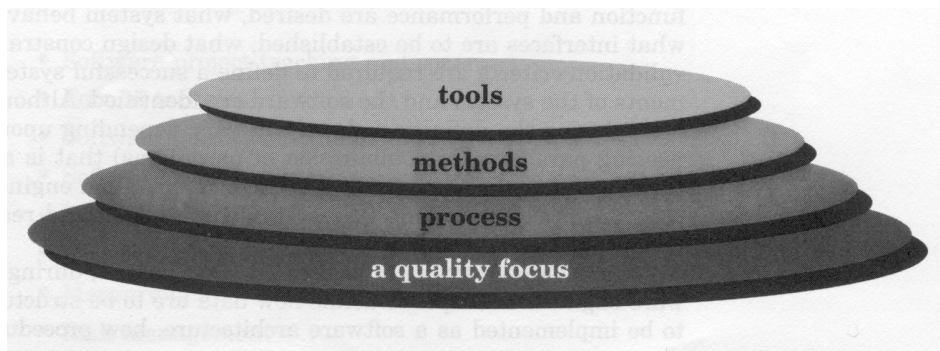
“The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, that is, the application of engineering to software.”

IEEE Standard 610.12

Two aspects of software engineering:¹

- technology
- process

Software process defines the approach that is taken as software is developed.



- ❑ layered technology
- ❑ process layer defines framework for project planning and management and for applying technologies
- ❑ methods – provide technical *how to's*
- ❑ tools – automated or semi-automated support for process and methods

¹ See Pressman chapter 2 and Sommerville chapter 3

Software Process

“..Software Process can be defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g. project plans, design documents, code, test cases, and user manuals.)”

CMM Version 1.1, SEI-93-TR-24

- different projects require different processes
- work products produced as consequence of process activities
- software processes can be adapted for particular project
- best indicators of a process are: quality, timeliness and long term-viability of resulting product

Generic View of Process - 3 phases

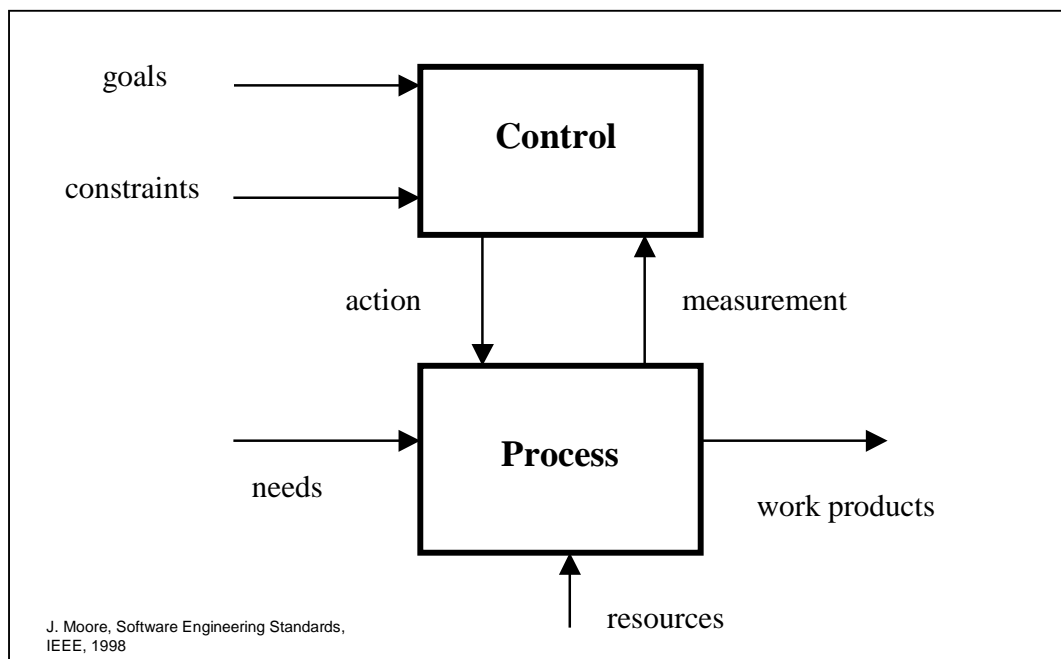
- definition phase - focus on what
 - information to be processed
 - functions and performance desired
 - system behaviour expected
 - interfaces
 - design constraints
 - validation criteria
 - involves system engineering, project planning and requirements

- development phase - focus on how
 - data is to be structured
 - function is to be implemented within a software architecture
 - procedural details are to be implemented
 - design is to be translated into a programming language
 - testing will be done
 - involves design, coding and testing

- support or maintenance phase - focus on change
 - **corrective maintenance** – to fix defects
 - **adaptive maintenance** – accommodates changes in external environment, e.g. OS, business rules
 - **perfective maintenance** or enhancement – addition of other useful functions, extends software beyond original functional requirements
 - **preventative maintenance** – *software reengineering* needed because of deterioration due to change. Makes software more easily corrected, adapted and enhanced

Umbrella Activities

- complement or overlay the process phases
- applied throughout software development process
- includes
 - software project tracking and control
 - formal technical reviews
 - software quality assurance
 - software configuration management
 - document preparation and production
 - reusability management
 - measurement
 - risk management



The Capability Maturity Model for Software (CMM)

“The Capability Maturity Model for Software (CMM) is a framework that describes the elements of an effective software process. The CMM describes an evolutionary path from an ad hoc, chaotic process, to a mature disciplined process.”

CMM Version 1.1, SEI-93-TR-24

- SEI produced a 5-point grading scheme to determine compliance with the CMM
- measures global effectiveness of a software engineering company's practices, its process maturity

Level 1 **Initial**

- add hoc, relies on individual effort
- few processes defined

Level 2 **Repeatable**

- basic project management established to track cost, schedule and functionality
- can repeat earlier successes on projects with similar applications

Level 3 Defined

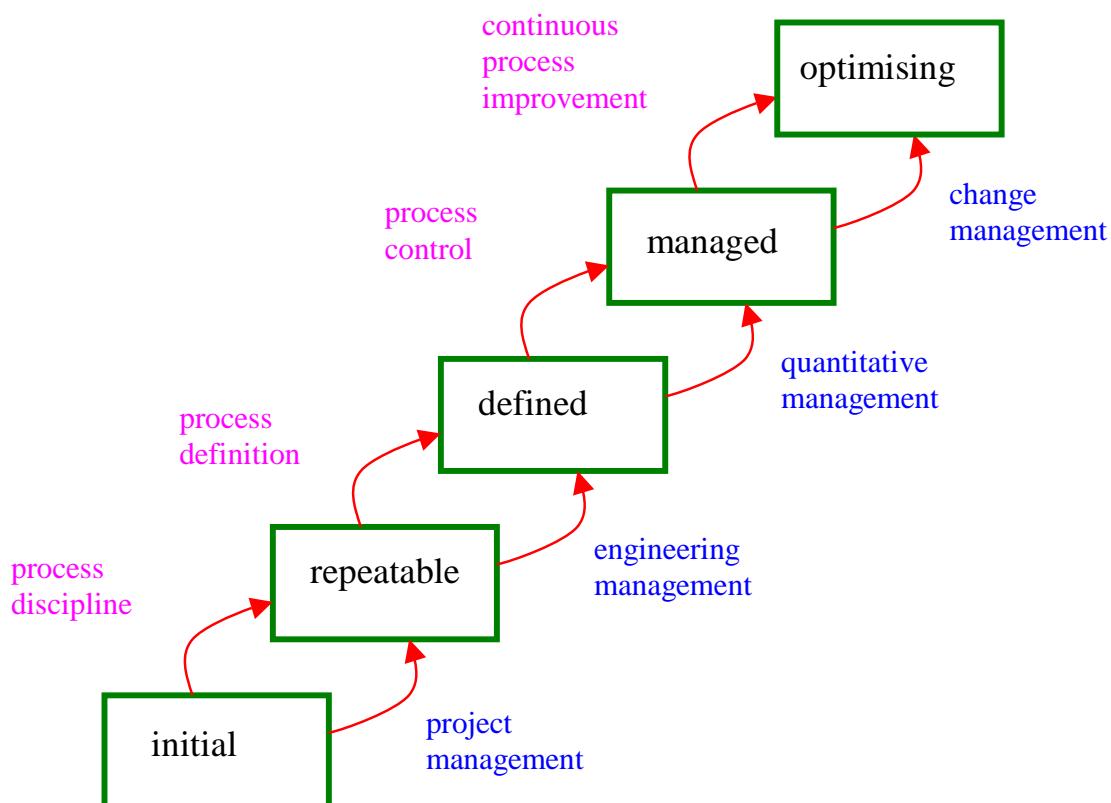
- software process for management and engineering is documented, standardised and integrated into organisation-wide software process
- all projects use documented and approved versions of organisation's processes
- level most aimed for by software developers via standards such as ISO 9001

Level 4 Managed

- measurement and use of metrics
- e.g. cyclomatic complexity of code – indicates amount of testing effort required
- measures of software process and product quality collected
- quantitative understanding of both products and process used for better monitoring and control
- requires well-defined notations for activities (such as requirements specification, design model) so that metrics can be easily extracted mechanically

Level 5 **Optimising**

- reached by very few developers
- continuous process improvement enabled by quantitative feedback from the process and testing innovative ideas and technology
- can predict things like number of residual bugs in a product based on measurements during project
- requires every process rigorously defined and followed to the letter, otherwise variation creeps in and statistics are no good for prediction



CMM Architecture

- SEI has associated Key Process Area with each level

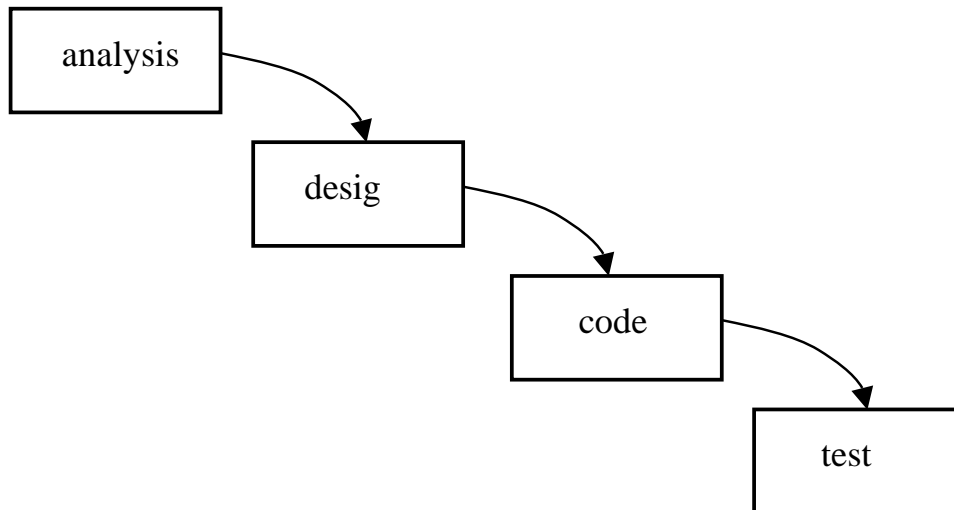
- a KPA describes software engineering functions which must be present for a level

- Each KPA described by
 - goals
 - commitments
 - abilities
 - methods for monitoring implementation

- 18 KPAs defined across maturity model

- for example, for level 2
 - software configuration management
 - software quality assurance
 - software subcontract management
 - software project tracking and oversight
 - software project planning
 - requirements management

Classic Life Cycle or Waterfall Model



- ❑ oldest & most widely used paradigm
- ❑ provides a basic template for software engineering process
- ❑ assumes a complete system will be delivered

Drawbacks

- ❑ real projects rarely follow it
- ❑ difficult to establish all requirements explicitly, no room for uncertainty
- ❑ customer must have patience, not fast enough for delivery of modern internet based software
- ❑ major mistake can be disastrous
- ❑ unnecessary delays, “blocking states”
- ❑ difficult to trace requirements from analysis model to code
- ❑ inflexible partitioning of the project into distinct stages
- ❑ this makes it difficult to respond to changing customer requirements
- ❑ therefore, this model is only appropriate when the requirements are well-understood