CITS5502 – Capability maturity

Unit coordinator: Arran Stewart

Sources

- Pressman 9th ed, Ch 26

Why try to improve processes?

- In competitive markets, there is pressure to deliver software faster and cheaper, which meets customer needs
- Organisations may look to process improvement to improve software quality, reduce costs, or speed up their processes
- It's clear that the processes used to develop software do have a bearing on the quality of the software produced; therefore people reason that improving the processes can improve the software.

Major approaches to Software Process Improvement (SPI)

- **Process maturity models**
  - Focuses on project management, introducing good software engineering practice
  - Defines *levels* of process maturity
  - These reflect the extent to which good practices have been adopted into processes
  - Primary goals: improved product quality and process predictability.
- **Agile**
  - Focuses on iterative development and *reduction of overheads*
  - Goals include rapid delivery of functionality and responsiveness to changing customer requirements.

CMM

- We'll look at one maturity model–based approach, the *CMM* (Capability Maturity Model), and its successor, *CMMI* (Capability Maturity Model Integration)

CMM background – statistical quality control

- US engineer W.E. Deming worked with Japanese manufacturing industries after WWII to help improve quality.
- The idea of *statistical quality control* is due to Deming and others:
    - "reduce product defects by analyzing and modifying the process so that the chances of introducting defects are reduced and defect detection is improved"
    - Once defects have been reduced, standardise the process and start again

Watts Humphrey and SEI

- Watts Humphrey – Pioneer of the work on CMM
- Humphrey stated that Deming's concepts of statistical quality control "are just as applicable to software as they are to automobiles, cameras, wristwatches and steel".[1]

[1] Humphrey, W. (1989). Managing the Software Process. Reading, Mass.: Addison-Wesley.

[2] CMM: Described in Humphrey (1989) and later (mid 90s) papers led by Mark Paulk

Watts Humphrey and SEI

- Watts Humphrey – Pioneer of the work on CMM
- Humphrey stated that Deming's concepts of statistical quality control "are just as applicable to software as they are to automobiles, cameras, wristwatches and steel".[1]
- 1980s at Software Engineering Institute (SEI) at Carnegie Mellon University: Humphreys founded the Software Process Program, aimed at understanding and managing the Software Process.
- Software Process Program gave rise to CMM[2] .

[1] Humphrey, W. (1989). Managing the Software Process. Reading, Mass.: Addison-Wesley.

[2] CMM: Described in Humphrey (1989) and later (mid 90s) papers led by Mark Paulk

CMM

- CMM: a development model created from a study of data collected from organizations that contracted with the U.S. Department of Defence (who funded the research)
- The intent of the capability maturity model is to provide an overall indication of the “process maturity” exhibited by a software organization.
- This is accomplished using some type of ordinal scale.

CMM is described as

> *“. . . sets of recommended practices in a number of key process areas that have been shown to enhance software process capability.”*[3]

[3] Paulk et al (2002–), “The Capability Maturity Model for Software” in Wiley Encyclopedia of Software Engineering. https://doi.org/10.1002/0471028959.sof589

Typical assumptions in applying statistical quality control

- The development and maintenance process can be defined
- Significant aspects of that process can be measured
- When measured, the variation in the data will not be unreasonably large
- Given the signal-to-noise ratio is acceptable, then the conditions hold from project to project is predictable

Given a repeatable process, the outcomes may be optimized with respect to some organizational objective function

Concepts in CMM

Immature vs mature organizations[4]

Immature:

- Software processes are generally improvised during the course of a project.
- Managers are reactionary, solving immediate crises.
- Functionality and quality are compromised when hard deadlines approach.
- No objective basis for judging product quality; thus quality is difficult to predict.

[4]Paulk et al (1993), *Capability Maturity Model for Software, Version 1.1*

Concepts in CMM

Immature vs mature organizations[5]

Mature:

- The software process is accurately communicated to existing and new staff.
- Work activities are actually carried out according to the planned process.
- The mandated processes are usable and consistent with the way work actually gets done.
- Roles and responsibilities are clear, schedules and budgets are based on historical performance.

[5] Paulk et al (1993), *Capability Maturity Model for Software, Version 1.1*

Concepts in CMM, cont'd

Software process	methods, activities, practices used to develop and maintain software
Software process capability	the range of expected results achieved by following a software process.
Software process performance	the actual results achieved by following a software process.

Concepts in CMM, cont'd

Software process maturity: the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective.

As an organization gains in process maturity, it institutionalizes its process by policies, standards and organizational structures.

CMM – levels of maturity

- Identify an evolutionary path from immature to mature.
- Identify a number of well-defined plateaus along the way.
    - Each plateau (level) is a set of process goals which, when satisfied, stabilize some component of the software process.
- Process goals are prioritized, so organizations understand what to do next.

CMM – levels of maturity

CMM suggests 5 levels of maturity:

- Initial
- Repeatable
- Defined
- Managed
- Optimizing

CMM – levels of maturity

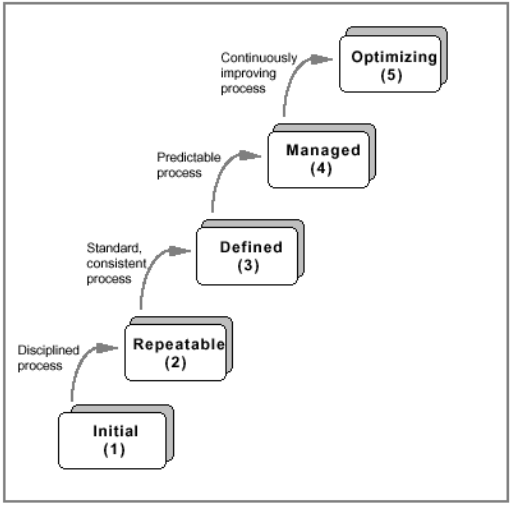


Figure 2.1 The Five Levels of Software Process Maturity

Initial

Level 1, Initial

Few processes are defined; success depends more on individual heroic efforts than on following a process and using a synergistic team effort.

- Unstable environment
- Commitments made that can't be met
- Plans are scrapped during crises, jumping to coding and testing.
- Can success happen? Yes, but only through heroic efforts.

Initial, cont'd

- Such efforts will not likely be repeatable
- Capability is a characteristic of individuals, not the organization.
- Would you work for or with such an organization? Have you?

Repeatable

Level 2, Repeatable

Basic project management processes are established to track cost, schedule, and functionality. Planning and managing new products is based on experience with similar projects.

- Management policies are in place.
- Planning and managing are based on experience.
- Policies are enhanced on a project by project basis.
- Commitments are realistic.

Repeatable, cont'd

- Managers track software costs, schedules, functionality.
- Project standards are defined and followed.
- Planning and tracking are stable.
- Earlier successes can be repeated.

Defined

Level 3, Defined

Processes for management and engineering are documented, standardized, and integrated into a standard software process for the organization.

- A standard process for software engineering and management across the organization is documented.
- There is a group responsible for this standard software process.
- There are organization-wide training programs.

Defined, cont'd

- Projects tailor the standard process to account for their unique features.
- Such a tailored process contains readiness criteria, inputs, verification mechanisms, outputs and completion criteria.
- This process capability is based on a common, organization-wide understanding of activities, roles and responsibilities.

Managed

Level 4, Managed

Detailed software process and product quality metrics establish the quantitative evaluation foundation. Meaningful variations in process performance can be distinguished from random noise, and trends in process and product qualities can be predicted.

- The organization sets quantitative quality goals for software products and processes
- A process database is used to collect and analyze data from projects' processes.
- Variation in process performance is narrowed to acceptable levels.
- Software processes are instrumented.

Managed, cont'd

- Risks in new product development are well understood.
- The process capability is quantifiable and predictable.
- Software products are of predictably high quality.

Optimizing

Level 5, Optimizing

The organization has quantitative feedback system in place to identify process weaknesses and strengthen them pro-actively. Project teams analyze defects to determine their causes; software processes are evaluated and updated to prevent known types of defects from recurring.

- The entire organization is focused on continuous process improvement.
- Weaknesses can be identified and remedies found.
- Data on software process allows cost-benefit analyses on new technologies.

Visibility

- Typically in an organization, software engineers have detailed insight into the state of a project they are working on – though in large projects, typically only in their area of responsibility
- Senior management, on the other hand, do not have such detailed insight – they rely on periodic reviews in order to monitor progress.
- In CMM, a consequence of achieving the maturity levels is that managers are better informed, and have greater abbility to control the process

Visibility – level 1, initial

- The project is a black box
- Requirements flow into the project haphazardly
- The staging of activities is hidden
- Managers can't establish project status

Visibility – level 2, repeatable

- Requirements are controlled
- Visibility at defined occasions
- The project is a sequence of black boxes
- Management reacts to problems as they occur

Visibility – level 3, defined

- Tasks inside each black box are known
- Management and engineers understand their “roles and responsibilities” within each box
- Management is proactive in dealing with problems, because of rapid status updates

Visibility – level 4, managed

- Processes are “instrumented” (details of estimates and outcomes are recorded)
- Decisions are based on hard facts
- Outcomes can be predicted more accurately
- Variability in outcomes gets smaller

Visibility – level 5, optimizing

- Improved methods of software development are regularly tried
- Defect-prone activities are replaced
- Managers can predict the impact of process changes

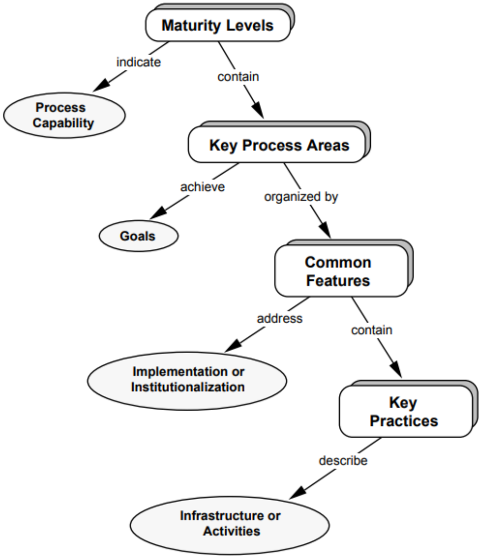
Specific recommendations

So what actual, concrete recommendations does the CMM make?

> *"Each maturity level has been decomposed into constituent parts. . . . [T]he decomposition of each maturity level ranges from abstract summaries of each level down to their operational definition in the key practices. . . . Each maturity level is composed of several key process areas. Each key process area is organized into five sections called common features. The common features specify the key practices that, when collectively addressed, accomplish the goals of the key process area."*[6]

[6] Paulk et al (1993), *Capability Maturity Model for Software, Version 1.1*

Specific recommendations

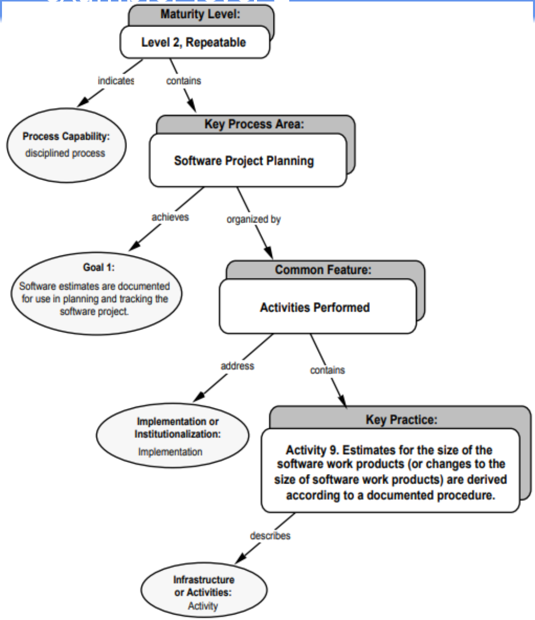


Key process areas – example, level 2

Key Process Areas: Level 2

- Requirements management
- Software project planning
- Software project tracking and oversight
- Software subcontract management
- Software quality assurance
- Software configuration management

Key practice – example, level 2


Maturity Level:
Level 2, Repeatable
indicates
contains
Process Capability:
disciplined process
Key Process Area:
Software Project Planning
achieves
organized by
Goal 1:
Software estimates are documented for use in planning and tracking the software project.
Common Feature:
Activities Performed
address
contains
Implementation or Institutionalization:
Implementation
Key Practice:
Activity 9. Estimates for the size of the software work products (or changes to the size of software work products) are derived according to a documented procedure.
describes
Infrastructure or Activities:
Activity

Prediction of Performance

Three improvements are expected as the software process matures:

- The difference between targeted results and actual results decreases.
- The variability of actual results around targeted results decreases.
- Targeted results improve as maturity increases.

Implications of CMM

- Can't skip levels (cultural changes are required)
- Can take a long time to go from one level to the next

Comments on CMM

- Doesn't account for small, innovative firms
- Small organizations probably cannot afford the overhead required by CMM.

Alternative maturity models

Other organizations have also developed comparable process maturity models:

- The SPICE approach to capability assessment and process improvement
- The Bootstrap project, which uses the SEI's maturity levels, includes guidelines for developing a company-wide quality system to support process improvement.
- PSP and TSP – individual and team-specific SPI framework that focus on process in-the-small.
- TickIT – an auditing method that assesses an organization's compliance to ISO Standard 9001:2000.

Successor – CMMI

- The original CMM was developed and upgraded by the SEI throughout the 1990s as a complete SPI framework.
- Today it has evolved into the Capability Maturity Model Integration (CMMI), a comprehensive process metamodel
- The CMMI model is very complex, with more than 1,000 pages of description.

CMMI cont'd

Two sub-models:

- “Staged CMMI”
- “Continuous CMMI”

The Staged CMMI model is analogous to the CMM, except that there are 6 rather than 5 levels.

Empirical evidence

- Certainly there are success stories from CMM. But to what extent do these show that adopting CMM *results in* improvements?

Empirical evidence

- Certainly there are success stories from CMM. But to what extent do these show that adopting CMM *results in* improvements?
- Selection bias. Adopting CMM is unlikely to be done at random. What sort of organisations would adopt CMM? Can we reliably generalize to other sorts of organisation?

Empirical evidence

- Certainly there are success stories from CMM. But to what extent do these show that adopting CMM *results in* improvements?
- Selection bias. Adopting CMM is unlikely to be done at random. What sort of organisations would adopt CMM? Can we reliably generalize to other sorts of organisation?
- Reporting bias. Suppose an organisation tries to adopt CMM and it goes badly. How likely are they to report this to outsiders?

Critique by James Bach

James Bach, founder of Satisfice, formerly at Apple and Borland International:

> *"[given that] the CMM is a broad, and increasingly deep, set of assertions as to what constitutes good software development practice, it's reasonable to ask where those assertions come from, and whether they are in fact complete and correct."*[7]

[7] J Bach (1994), "The Immaturity of CMM", originally published in *American Programmer*. Available at https://www.satisfice.com/blog/archives/6208

Bach, cont'd

"My thesis. . . is that the CMM is a particular mythology about software process evolution that cannot legitimately claim to be a natural or essential representation of software processes."

Bach, cont'd

The CMM is

"at best a consensus among a particular group of software theorists and practitioners"

"at worst a whitewash that obscures the true dynamics of software engineering [and] suppresses alternative models"

"If an organization follows it for its own sake, it may lead to the collapse of that company's competitive potential."

Bach – key criticisms

- CMM has no formal theoretical basis – based on the experience of "very knowledgeable people". But any other model based on experiences of other knowledgeable people has equal veracity.
- CMM has only vague empirical support – evidence forr it could equally well support other models.
  - Based mainly on experience of large government contractors, and Watts Humphrey's experience in the mainframe world.

Bach – key criticisms, cont'd

- CMM "reveres process, but ignores people". Compare the work Gerald Weinberg (author of "The Psychology of Computer Programming") – for Weinberg, the problems of human interaction *define* engineering.
    - Humphrey and CMM mention assume that defined processes can render individual excellence less important – process makes up for mediocrity.
- CMM "reveres institutionalization of process for its own sake".
    - But even if processes are not institutionalized formally, they may very well be in place, informally, by virtue of the skill of the team members.
- CMM "contains very little information on process dynamics". Many seemingly arbitrary choices – why isn't "training" on level 1? Why is defect prevention a level 5 practice?

Bach – key criticisms, cont'd

- CMM "encourages displacement of goals from the true mission of improving process to the artificial mission of achieving a higher maturity level". Can blind an organization to the most effective use of its resources.

Other criticisms

Short blog post by Bertrand Meyer, developer of the Eiffel language.[8] Commenting on CMMI, but applies largely to both CMM and CMMI:

> *A much better name would have been "Catalog of Assessable Process Practices", which is even pronounceable as an acronym, and defines the key elements: the approach is based on recognized best practices; these practices apply to processes (of an organization); they must be subject to assessment (the most visible part of CMMI — the famous five levels — although not necessarily the most important one); and they are collected into a catalog. If "catalog" is felt too lowly, "collection" would also do.*

[8] Meyer (2013), "What is wrong with CMMI?" https://bertrandmeyer.com/2013/05/12/what-is-wrong-with-cmmi/