CMM (for Software) - An Overview

by Cliff Kettemborough*, Section 312
Systems Software Architect

presented to Jim Larson's Programmer's Lunch Series, February 8, 1997

* Available on the Web @ http://epic/nav/doc/sat/

For questions/comments e-mail to crk@jpl.nasa.gov

Claimer/Disclaimer: About half of the material presented herein is from the official CMU/SEI-93-TR-24 Capability Maturity Model document (v1.0). The other half is the product of my knowledge and experiences.

Introduction (1 of 9)

- What it stands for?
  - » CMM = Capability Maturity Model
  - » Keywords: Capability, Maturity, Model
- History
  - » CMM (basic concepts) have been developed by the Software Engineering Institute (SEI) at Carnegie Mellon University, the late ‘80s, work started in 1986

Introduction (2 of 9)

- Basic ideas published in Watts Humphrey's book *Managing the Software Process* (1987)
- The CMM term coined and first release of the framework was published in 1991 (release v1.0)
- Right away a number of large DoD contractor firms embraced it (per DoD recommendations); among them Rockwell Int'l (my former employer at that time,

Introduction (3 of 9)

provided training to key staff and pursued its implementation)

» For (a lot) more information, look up on http://www.sei.cmu.edu

- What is it?
  - » A Systems/Software Engineering Framework (Model) that sets Guidelines and associated Metrics to Measure the Maturity of a Software Development Environment/Organization

Introduction (4 of 9)

(e.g., team, group, department, company, corporation).

- Evolution
  - » It's been finely tuned ever since its inception, even though the main concepts were not changed
  - » This framework (model) is rapidly becoming (if not already is) a standard measure of an organization capability to develop High Quality software

Introduction (5 of 9)

- » U.S. Government (and its departments, e.g., DoD) uses it aggressively, for at least past half decade to decide on awarding contracts to various bidders
- » Lately, it's being embraced by commercial organization as well, due to an emerging TQM and ISO 9000 type of requirements

- Quality Initiatives
  - » **Total Quality Management** (TQM) a quantitative management approach to zero

Introduction (6 of 9)

defect manufacturing by Dr. W. Edward Deming:

- intended primarily to manufacturing
- various forms stemmed from it in non-manufacturing environments

» **United States Malcom Baldridge National Quality Award** named for the former secretary of commerce (1981-1987), managed by National Institute for Standards and Technology and the

Introduction (7 of 9)

Department of Commerce, administered by the American Society for Quality Control:

- intended primarily to manufacturing and service industries

» **International Organization for Standardization 9000 series** (ISO-9000) in the United States, the American National Standards Institute (ANSI)/American Society for Quality Control (ASQC) document Q9000-1 through Q9000-4.

Introduction (8 of 9)

The Institute of Electrical and Electronic Engineers (IEEE) contributes to the improvement of ISO-9000:

- intended primarily to manufacturing and software engineering
- made a certification requirement by the European Economic Community member countries as 1/1/92 for doing business with them

» **Capability Maturity Model** (CMM) developed by the Software Engineering

Introduction (9 of 9)

Institute (SEI), and sponsored by the Department of Defense (DoD):

- intended primarily to software engineering

» At JPL, the development of **D-5000** (for systems engineering) and **D-4000** (for software engineering) standards in the late ‘80s early ‘90s:

- Now, just a guideline with no waivers required
- Development of an equivalent **D-6000** (for hardware engineering), did not materialized

Software Industry Overview* (1 of 5)

- $46 Billion considered waste
- An estimated (according to some sources, some others the figure would be just for the United States) 100 billion lines of (source) code around the World
  - » 70% COBOL and PL/I
  - » 25% FORTRAN, Pascal, BASIC
  - » 5% Other (C, C++, Ada, etc.)

* Source IBM, SEI, etc.

Software Industry Overview (2 of 5)

- Project larger than 4 years are 50% maintenance type
- Lifespan of a program, is about 7 years; after that needs re-written; most they don’t, just being continuously patched
- All projects require 30-45% rework
- 70% of them are over schedule and cost

Software Industry Overview (3 of 5)

- It cost 20x to fix a bug after the release; and 10x after the integration
- On average, experimented programmers make a mistake about every 10 LOC (Lines of (source) Code); they fix about have of them when translated to machine language, and more during (good) testing.

Software Industry Overview (4 of 5)

- 1,000,000 LOC is the equivalent of 4 phone books; MS Windows 95 would take 46 phone books!
- 1,000,000 LOC starts with 100,000 defects; engineers fix about 95% of them during (good, solid) testing; but, this still leaves 5,000 defects in a finished product!

Software Industry Overview (5 of 5)

- Quality vs. Schedule: both or none - “Quality on Time”

Background (1 of 10)

- 30 years or so of Systems and Software Engineering Methodologies developments/evolution
  - Software Engineering Process: “A total set of software engineering activities, procedures and practices needed to transform a users requirements into a software (product).”

Background (2 of 10)

- Emphasis on a structured (Waterfall) paradigm:
  - » Planning , Definitions (mission, goals, objectives, activities, phases, tasks and resources), Feasibility Study
  - » Requirements Engineering (Gathering, Analysis, Specification, Categorization and Prioritization)
  - » Systems Analysis and Modeling

Background (3 of 10)

- System Design (Architectural or High Level and Detail or Low Level)
- Implementation (of the Design) or Coding
- Testing (at various levels: Unit, Integration, System, Functionality, Performance, Regression, etc.)
- System Delivery/Installation/Deployment
- Post Implementation Review, Performance Evaluation

Background (4 of 10)

» Maintenance and Enhancements, along with parallel activities such as:

» Project and Risk Management

» (Independent) Verification and Validation

» Quality/Product Assurance

» Configuration and Change Management

» Metrics (McCabe's and Halstead's) Definition, Collection, Analysis and Implementation

Background (5 of 10)

- Training
- Documentation
- Formal (Fagan's) Inspections
- Formal Reviews at Project/Program Milestones
- Traceability
- Repository-based Development
- Extensive use of tools (CASE technology)

Background (6 of 10)

- » Policies and Procedures Definition and Standardization/Guidelines (DoD 2167, 2168, JPL D4000/5000, IEEE Standards)
- and, more modern (Spiral) paradigm:
  - » all of the above plus a great emphasis on techniques such as:
  - » Continuous Feasibility Assessment
  - » Continuous Risk Assessment
  - » (Rapid) Prototyping Techniques

Background (7 of 10)

- » Rapid Application Development (Design) - RAD
- » Joint Application Development (Design) - JAD, i.e., great emphasis on User and Management Involvement

- Or, an updated WinWin Spiral paradigm:
  - » Capture the WIN conditions of key stakeholders: Users, Customers, Owners, Developers, Maintainers

Background (8 of 10)

- » Groupware Decision Making approach
- » Service paradigm
- Microsoft's (new) RSD (Rapid Software Development) paradigm
- Key Elements: Processes, Infrastructure Development, Training, various levels of Control (for “checks and balances”) and their

Background (9 of 10)

Standardization across the organization as a CULTURE

- Other Key Elements (borrowed from Dr. Deming and/or TQM-like): great emphasis on Teamwork, Improved Communications, Continuous Process Improvement, Quality-driven Awareness

Background (10 of 10)

- Or, concepts such as: Reverse and Forward Engineering, (continuous) Re-engineering capability, (James Martin's) Information Engineering, (James Martin's) Business Process Re-engineering, (James Martin's) Cybercorp, (Peter Drucker's) Knowledge Engineering

Software Process (1 of 3)

- Why?
  - » Creates Visibility within and outside the development organization
  - » Improvability: standard activities and work products/services can be measured and optimized over time
  - » Predictability: software cost, schedule, and quality are more predictable and repeatable
    - – "What can be measured it can be done."

Software Process (2 of 3)

» Manageability: Principles of Process Management (according to SEI with personal modifications):

- *Process Stability* - an orderly and sustained improvement of an ad hoc process is impossible
- *Process Definition* - in a complex process, the organization must identify “key” steps and focus on their improvement
- *Global vs. Local Optimization* - problem solving in isolation tends to degrade the process

Software Process (3 of 3)

- *Process and Product (Service) Certification* - both are (equally) important
- *Process Support* - the process won't improve by itself.

SEI's Fundamental Concepts (1 of 4)

- **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). As an organization matures, the software process becomes better defined and more consistently

SEI's Fundamental Concepts (2 of 4)

implemented throughout the organization.

- **Software process capability** describes the range of expected results that can be achieved by following a software process. The software process capability of an organization provides one means of predicting the most likely outcomes to be expected from the next

SEI's Fundamental Concepts (3 of 4)

software project the organization undertakes.

- **Software process performance** represents the actual results achieved by following a software process. Thus, software process performance focuses on results expected.
- **Software process maturity** is the extent to which a specific process is

SEI's Fundamental Concepts (4 of 4)

explicitly defined, managed, measured, controlled, and effective. Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization.

SEI's CMM - the 5 Stages/Levels (1 of 8)

- As defined by SEI there are Five Levels of Software Maturity:
  - **(1) Initial**
  - **(2) Repeatable**
  - **(3) Defined**
  - **(4) Managed**
  - **(5) Optimizing**

SEI's CMM - the 5 Stages/Levels (2 of 8)

- To move from level 1 to 2 a *Disciplined Process* must be in place
- To move from level 2 to 3 a *Standard, Consistent Process* must be in place
- To move from level 3 to 4 a *Predictable Process* must be in place
- To move from level 4 to 5 a *Continuous Improving Process* must be in place

SEI's CMM - the 5 Stages/Levels (3 of 8)

- Detail Definitions
  - » **(1) Initial** - The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.
  - » **(2) Repeatable** - Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with

SEI's CMM - the 5 Stages/Levels (4 of 8)

similar applications.

» **(3) Defined** - The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

SEI's CMM - the 5 Stages/Levels (5 of 8)

- **(4) Managed** - detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
- **(5) Optimizing** - Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

SEI's CMM - the 5 Stages/Levels (6 of 8)

- Behavioral Characterization of the Maturity Levels
  - » Maturity Levels 2 through 5 can be characterized through the activities performed by the organization to establish or improve the software process, by activities performed on each project, and by the resulting process capability across projects.

SEI's CMM - the 5 Stages/Levels (7 of 8)

A behavioral characterization of Level 1 is included to establish a base of comparison for process improvement at higher maturity levels.

» A **maturity level** is a well-defined evolutionary plateau toward achieving a mature software process. each maturity level comprises a set of process goals that, when satisfied, stabilize an important component of the software process.

SEI's CMM - the 5 Stages/Levels (8 of 8)

Achieving each level of the maturity framework establishes a different component in the software process, resulting in an increase in the process capability of the organization.

Assessment - as defined by SEI (1 of 5)

- What is an Assessment?
  - » An appraisal, buy a trained team of experienced software professionals, of an organization's current software process, based on:
    - Review of 4 to 6 projects
    - Responses to assessment questionnaire
    - In-depth discussions with project managers and practitioners

Assessment - as defined by SEI (2 of 5)

- Collective assessment team knowledge and experience

- Assessment Objectives
  - » Understand the organization's current software engineering practices
  - » Identify key areas for process improvement
  - » Facilitate the initiation of process improvement actions:
    - Provide framework for action
    - Help obtain sponsorship and support for action

Assessment - as defined by SEI (3 of 5)

- Observe Strict Confidentiality
  - » Only composite results are given to management team:
    - – No name attribution to individuals
    - – No project identification
  - » Assessment team and participants agree to keep all information confidential
  - » Organization free to disclose results as desired

Assessment - as defined by SEI (4 of 5)

- » Project specific assessment data is sent to the SEI to be used for statistical reporting purposes only
- Involve Senior Management
  - » The software process is a management responsibility
  - » Sponsorship of assessment and improvement action
  - » Participation in periodic reviews of process

Assessment - as defined by SEI (5 of 5)

- Assessment Summary
  - » An assessment is:
    - Based on the capability maturity model
    - A tool to assist an organization in determining where its software practice is today and where it should be tomorrow
    - A significant step toward improving an organization's software engineering capabilities
    - A beginning to a long term commitment to process improvement
    - A catalyst for change!

Industry Experiences (1 of 3)

- Most large aerospace (Hughes, Rockwell, McDonnel Douglas, TRW, Litton, Logicon, Northrop, Lockheed, Raytheon, Loral, etc.) corporations (but not only: IBM, Xerox, GTE, Beckman Instruments, etc.) have substantial resources allocated to this effort. As a result they are among the best rated in the country on the CMM scale 1 - 5.

Industry Experiences (2 of 3)

- The implementation of this model requires a commitment at many levels of an organization to a culture change. A strategic approach, allocation of resources, and a multi-year effort is a sort list of prerequisites. Requires *training* and the building of an *infrastructure*.

Industry Experiences (3 of 3)

- The pay-off, based on industry experiences: *for every $1 invested in the process a $5 return on investment*.

Survey Findings (1 of 4)

- Latest statistics figures from SEI (1996):
  - » Level 1 (Initial): 68.8% of surveyed organizations
  - » Level 2 (Repeatable): 18.0%
  - » Level 3 (Defined): 11.3%
  - » Level 4 (Managed): 1.5%
  - » Level 5 (Optimizing): 0.4%.

Survey Findings (2 of 4)

- This represents a great improvement over past couple of years. When first surveys were run in very early ‘90s, it was extremely rare to find an organization at level 3; over 90% were at level 1! At levels 4 or 5 there were virtually none!
- The above given distribution broken down by organization type:

Survey Findings (3 of 4)

- **Level 1**:
  - DoD/Federal Contractor: 54%
  - Commercial/In-house: 73%
  - Military/Federal: 80%
  - Other: 92%
- **Level 2**:
  - DoD/Federal Contractor: 19%
  - Commercial/In-house: 19%
  - Military/Federal: 15%
  - Other: 8%

Survey Findings (4 of 4)

- **Level 3**:
  - DoD/Federal Contractor: 23%
  - Commercial/In-house: 6.5%
  - Military/Federal: 5%
- **Level 4**:
  - DoD/Federal Contractor: 3%
  - Commercial/In-house: 1%
- **Level 5**:
  - DoD/Federal Contractor: 1%
  - Commercial/In-house: 0.5%

Section 312 (1 of 3)

- What all of the above mean to us?
- Where do we stand?
- Should we do something about?
- If yes, what and how?
- Section 312 System Software Re-engineering and Architecture Master Plan released at the end of November ‘96:

Section 312 (2 of 3)

- Defines and set the direction to move us toward the next level on the CMM
- Resources and priority allocation to develop in more detail an assessment plan and derived strategies
- Emphasis on Training, Education and Commitment to Culture Changes

Section 312 (3 of 3)

- Long-Term Goal: Develop Section 312 into a (software) Center of Excellence within the division, then within the Laboratory and NASA.