ESTGF POLITÉCNICO DO PORTO
Felgueiras
Escola Superior De Tecnologia e Gestão

Modelo para a Qualidade e Melhoria do Processo

2º ano
1º Semestre

Capability Maturity Model Integration (CMMI)

Por:
Cristóvão Sousa e Carla Pereira

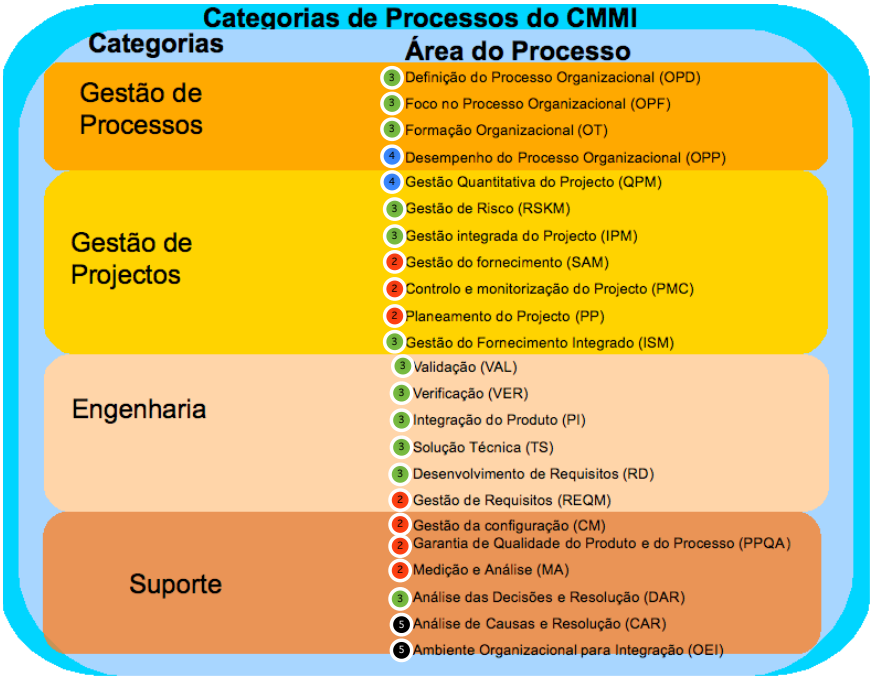
CMMI

- Produzido pelo SEI (Software Engineering Institute) da Universidade Carnegie Mellon (CMU)
- É uma evolução do CMM (Capability Maturity Model)
- É um modelo de gestão da qualidade, aplicável aos processos de desenvolvimento de software
- Constitui uma estrutura de trabalho que descreve os elementos chave para um processo de software eficaz e um caminho evolutivo até um processo maduro e disciplinado
- Objectivo: Melhoria contínua no desenvolvimento de software
- CMMI - modelo de referência que fornece orientação para o desenvolvimento de processos de software
- Reconhecido mundialmente por testar a maturidade dos processos de desenvolvimento da organização
- Avalia e "posiciona" a empresa numa escala progressiva de 5 níveis de maturidade, especificando o que é exigido em cada um dos níveis

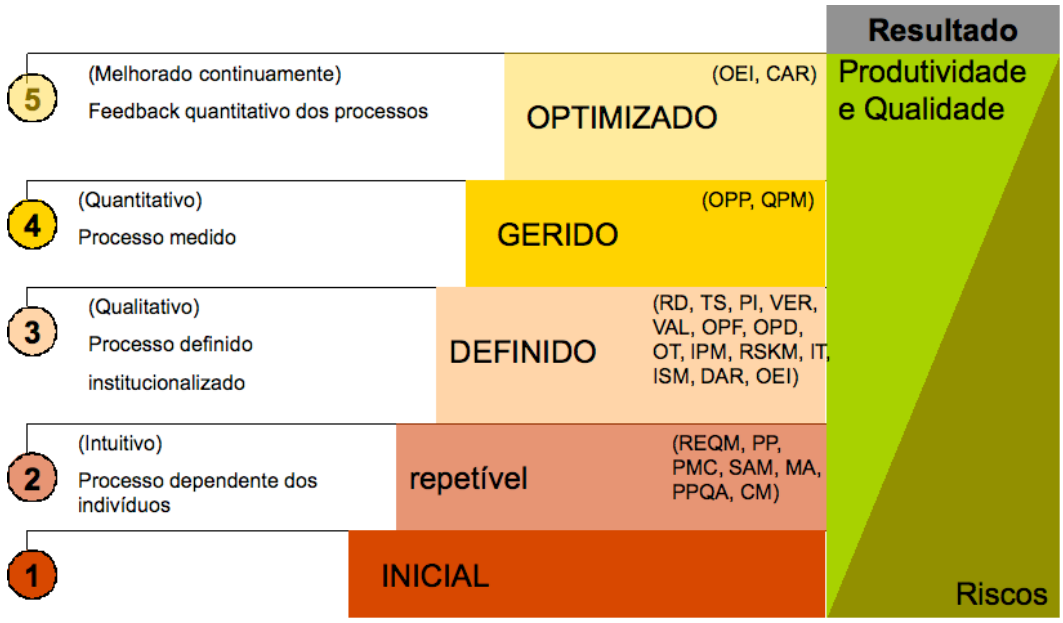
CMMI

- Estruturado em 5 níveis de maturidade:
  - **Inicial** – processo não estruturado (ad hoc)
  - **Controlado** (ou "Repetível") – processo de gestão básico estabelecido
  - **Definido** – documentação e normalização das actividades
  - **Controlado quantitativamente (Gerido)** – métricas para o processo e qualidade do produto
  - **Otimizado** – melhoria do processo com base em feedback quantitativo
- **Abrange 25 áreas de processo divididas em 4 categorias: 1) Gestão de processos; 2) Gestão de projectos; 3) Engenharia; 4) Suporte.**
- Cada nível (excepto o primeiro) é caracterizado pela implementação de determinadas áreas chave do processo.

CMMI



CMMI: níveis de maturidade



CMMI: níveis de maturidade (1)

- **Nível 1 (Inicial)**: Processos são normalmente ad-hoc e caóticos. A organização não proporciona uma ambiente estável para acolher o processo.
  - Características:
    - Produtividade e qualidade fortemente dependentes da dedicação e competência do pessoal
    - Geralmente o plano é abandonado e entra-se na abordagem codificação – testes
    - Difícil prever prazos, recursos e qualidade final
    - Os maiores problemas encontram-se a nível da organização e gestão
    - Filosofia de "apagar fogos"

CMMI: níveis de maturidade (II)

- **Nível 2 (Repetível)**: Os projectos da organização têm assegurado que os requisitos e os processos são controlados, executados e medidos.
  - Características:
    - Prioridade a uma gestão eficaz dos projectos
    - Processos de gestão são documentados e acompanhados
    - Os objectivos de negócio conduzem à definição dos processos de gestão
    - Anteriores sucessos de gestão podem ser repetidos sistematicamente

CMMI: níveis de maturidade(3)

- **Nível 3 (Definido)**: orientado aos processos. Além dos fluxos de atividades, gerem os aspectos organizacionais, técnicos e de integração de equipas e fornecedores em função da definição do processo.

CMMI: níveis de maturidade (4)

- **Nível 4 (Gestão Quantitativa)**: gestão do processo com métricas quantitativas através do tempo. Permite avaliar o processo de desempenho dos vários ciclos de desenvolvimento e comparar os indicadores. Deste é possível efectuar previsões.

CMMI: níveis de maturidade (5)

- **(5) Optimizado** - Uma organização continua a melhorar os seus processos baseados num entendimento quantitativo das causas comuns das variações inerentes aos processos.
  - Características:
    - Os processos são continuamente e sistematicamente melhorados através da análise quantitativa
    - As fontes "comuns" de problemas são compreendidas e eliminadas

CMMI: evolução dos níveis de maturidade

- Cada nível disponibiliza os mecanismos fundamentais onde assenta o nível seguinte
- Não é possível saltar degraus na escada da evolução, embora se possam iniciar algumas actividades presentes nos níveis mais elevados
- O CMMI é um referencial que representa o caminho de melhoria recomendado, para as empresas de software que pretendem aumentar a capacidade do seu processo de software

CMMI e Metodologias ágeis

Área	Métodos com maior contribuição	Percentual de satisfação (nível de capacidade ou NC + percentual)
Gerenciamento de Requisitos	XP	NC-2 em 75%, NC-3 em 50%
Medição e Análise	XP, ASD, SCRUM	NC-2 em 75%, NC-3 em 50%
Planejamento de Projeto	XP, SCRUM	NC-3 em 100%
Monitoramento e Controle	XP, SCRUM	NC-3 em 100%
Gerenciamento de Subcontratado	Não se aplica	-
Garantia de Qualidade de Produto e Processo	FDD, Crystal, EVO	NC-2 em 75%, NC-3 em 50%
Gerenciamento da Configuração	Nenhum	NC-1 em 10%
Percentual de satisfação para nível de maturidade 2		**72%**
Percentual de satisfação para nível de maturidade 3		**60%**

fonte: (alegria e bastarrica, 2006)

CMMI e Metodologias ágeis

Maturity Level	Process Area	SCRUM	XP
Level 2: Managed	Requirements Management (REQM)	LA	LA
	Project Planning (PP)	LA	LA
	Project Monitoring and Control (PMC)	PA	PA
	Supplier Agreement Management (SAM)	NA	NA
	Measurement and Analysis (MA)	LA	LA
	Process and Product Quality Assurance (PPQA)	NA	PA
	Configuration Management (CM)	NA	PA
Level 3: Defined	Requirements Development (RD)	LA	LA
	Technical Solution (TS)	NA	NA
	Product Integration (PI)	NA	LA
	Verification (VER)	NA	LA
	Validation (VAL)	LA	LA
	Organizational Process Focus (OPF)	PA	PA
	Organizational Process Definition (OPD)	PA	PA
	Organizational Training (OT)	NA	NA
	Integrated Project Management (IPM)	NA	NA
	Risk Management (RSKM)	NA	NA
	Decision Analysis and Resolution (DAR)	NA	NA
Level 4: Quantitatively managed	Organizational Process Performance (OPP)	NA	NA
	Quantitative Project Management (QPM)	NA	NA
Level 5: Optimizing	Organizational Innovation and Deployment (OID)	NA	NA
	Causal Analysis and Resolution (CAR)	NA	NA

LA = Largely Addressed / PA = Partially Addressed / NA = Not Addressed

fonte: www.blogcmmi.com.br

ESTGF POLITÉCNICO DO PORTO
Qualidade de Software

CMMI e Metodologias ágeis

Objetivo do CMMI	Práticas Ágeis
Gerenciamento de Requisitos	Stories, Product Backlog; Planning Games; Information Radiator; On-line Customer; Self-organizing Teams
Estabelecimento de Estimativas	Planning Games; Tasks and Effort Estimations for one to two weeks iterations on information radiator
Desenvolvimento do Plano de Projeto	Planning Games; Tasks on information radiator; Product Backlog
Aprovação do Projeto	Planning Games; On-line Customer; Self-organizing Teams; Reflection Workshops

fonte: Pikkarainen e Mantyniemi, 2006

Engenharia de Software II - Engenharia Informática
* Cristóvão Sousa e Carla Pereira*

Engenharia de Software II

2º ano
1º Semestre

Métricas de Software

Por:
Cristóvão Sousa

Porquê medir?

- Genéricamente permite:
  - **Determinar** a qualidade de um produto ou processo
  - **Prever** a qualidade de um produto ou processo
  - **Melhorar** a qualidade …
- Especificamente:
  - Estimar os custos e a calendarização de projetos de software
  - Avaliar a produtividade
  - Prever necessidade de recursos humanos
  - Antecipar necessidades de manutenção

Algumas definições

- Medida
  - indicação quantitativo de uma dimensão, capacidade, tamanho, (…), the um atributo de um produto ou processo
- Métrica
  - medida quantitativa do grau em que o sistema, componente ou processo possui um determinado atributo.
    - Exemplo: Número de erros por linha de código por homem/ hora

Exemplos

- de **Métricas**
  - *taxa de defeitos ou*
  - *taxa de erros, calculadas (medidas) por:*
    - *individuo*
    - *módulo*
    - *durante o SDLC*
- ***é importante categorizar os erros quanto à:***
  - *origem, tipo, custo, …*

Classificação típica

- Processo
  - resultados, tangíveis, decorrentes da execução das atividades do processo de desenvolvimento de software
  - *deliverables, documentation por produto, …*
- Produto
  - atividades relacionadas com a produção do próprio software
- Recursos
  - *inputs* relacionados com as atividades de desenvolvimento de software
  - conhecimento, hardware, pessoas, etc.

Produto vs. Processo

- Natureza das métricas relacionadas com o processo
  - modelo seguido para a gestão do SDLC
  - *work products, milestones, …*
- Natureza das métricas relacionadas com o produto
  - perceber e avaliar o estado do projeto
  - perceber avaliar os riscos
  - áreas do problemas não abrangidas
  - *skills* das equipas e da sua capacidade de controlar a qualidade do que desenvolvem

Tipos de **medidas**

- Diretas (*internas*)
  - Custo, esforço, LOC*, rapidez, memória
- Indiretas (*externas*)
  - Funcionalidade, qualidade, complexidade, eficiência, fiabilidade, capacidade de assegurar a manutenção

LOC= Lines Of Code; KLOC = 1000 LOC; SLOC = Statement lines of code

Métricas

- Exemplos de acordo com as medidas anteriores
  - Exemplos medidas baseadas em LOC:
    - Erros/LOC ou Custo/LOC ou Páginas de documentação/ KLOC
  - **Medidas baseadas em LOC são fáceis de usar, de calcular mas dependentes da linguagem e programador.**

Métricas relacionadas com a complexidade

- dado um problema
  - n_1 = the number of distinct operators
  - n_2 = the number of distinct operands
  - N_1 = the total number of operators
  - N_2 = the total number of operands

- Operadores: If, Else, { }, ( ), +, - , * etc…
- Operandos: 2, 5, variável *x*, …

Fonte: Halstead's Software Science (entropy measures)
Nota: *Entropia*: Medida da desordem de um sistema.

Métricas relacionadas com a complexidade

- n1 = 10
- n2 = 4
- N1 =13
- N2 = 7

```
if (k < 2)
{
  if (k > 3)
    x = x*k;
}
```

Métricas baseadas em entropia

Program vocabulary: $\eta = \eta_1 + \eta_2$

Program length: $N = N_1 + N_2$

Calculated program length: $\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$

Volume: $V = N \times \log_2 \eta$

Difficulty : $D = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2}$

Effort: $E = D \times V$

- **effort** - medida interessante para avaliar a capacidade de interpretação de uma programa

Fonte: Wikipedia (http://en.wikipedia.org/wiki/Halstead_complexity_measures)

Métricas baseadas em entropia

- The unique operators are:
  - main, (), {}, int, scanf, &, =, +, /, printf
- The unique operands are:
  - a, b, c, avg, "%d %d %d", 3, "avg = %d"

```
main()
{
    int a, b, c, avg;
    scanf("%d %d %d", &a, &b, &c);
    avg = (a + b + c) / 3;
    printf("avg = %d", avg);
}
```

$\eta_1 = 10, \eta_2 = 7, \eta = 17$

$N_1 = 16, N_2 = 15, N = 31$

Calculated Program Length: $\hat{N} = 10 \times log_2 10 + 7 \times log_2 7 = 52.9$

Volume: $V = 31 \times log_2 17 = 126.7$

Difficulty: $D = \frac{10}{2} \times \frac{15}{7} = 10.7$

Effort: $E = 10.7 \times 126.7 = 1,355.7$

Time required to program: $T = \frac{1,355.7}{18} = 75.3$ seconds

Number of delivered bugs: $B = \frac{1,355.7^{\frac{2}{3}}}{3000} = 0.04$

Fonte: Wikipedia (http://en.wikipedia.org/wiki/Halstead_complexity_measures)

McCabe's Complexity Measures

em parte já abordado nas aulas práticas

Métricas de Qualidade de Software

- Funcionalidade
  - características do sistema
- Usabilidade
  - documentação, sensação, experiência
- Fiabilidade
  - segurança, frequências das falhas
- Performance
  - rapidez
- Capacidade de fornecer suporte

Medidas da qualidade de software

- *grau em que o programa opera de acordo com as especificações* (**Correctness**)
  - Defeitos/KLOC; Fallhas/hora
- *grau em que o programa admite alterações* (**Susceptível de manutenção**)
  - Tempo médio por alteração; Alterações/versão; Custo/correção
- *grau em que o programa resiste a ata ques* (**Integridade**)
  - tolerância a falhas
- Facilidade de uso (**Usabilidade**)
  - Tempo/formação; *skills* mínimos necessários; Aumento da produtividade; …

Defeito - não conformidade com um requisito

Outras Métricas

- Design Metrics
  - Data complexity e System Complexity
- Coupling metrics (input / output metrics)
- Component level metrics
  - Cohesion, Coupling, …
- OO oriented
  - Weight methods per class
  - Depth of Inheritance Tree (e.g., number of children, …)
  - Coupling between classes (Collaborations)
  - Class size
  - Number of operations overriden
  - …

Ferramentas

- NetBeans
  - https://code.google.com/p/source-code-metrics/
- Eclipse
  - http://eclipse-metrics.sourceforge.net
- Source Monitor
  - http://www.campwoodsw.com/sourcemonitor.html