UNIVERSIDADE FEDERAL DO PARANÁ - UFPR
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CI 221
DISCIPLINA: Engenharia de Software **DATA: _____/_____/_____**
AULA NÚMERO: 18 **PROFESSOR: Andrey**

APRESENTAÇÃO

Nesta aula serão apresentados e discutidos os conceitos de Gerenciamento de Projetos e CMM

DESENVOLVIMENTO

CMM (Capability Matury Model)

Nos últimos anos, as organizações de desenvolvimento de software têm aumentado sua percepção em relação aos problemas que tipicamente as tocam. Software com bugs, prazos e orçamentos não cumpridos e insatisfação de clientes e usuários são eventos muito mais freqüentes do que se desejaria.

Já desde a década de 70 existe um consenso na comunidade de Engenharia de Software de que estes problemas estão, em grande parte, relacionados ao fato de que o desenvolvimento de software é muitas vezes realizado de forma "artesanal", isto é, através de métodos improvisados pelos desenvolvedores, os quais, por sua vez, muitas vezes dependem muito mais de seu talento individual (nem sempre abundante) do que de uma sólida formação acompanhada de métodos formais que dirijam suas atividades. Isto levou ao desenvolvimento e introdução das chamadas Metodologias de Desenvolvimento de Sistemas (MDS), através das quais se procurava padronizar boas práticas de engenharia de software, normalmente vinculadas a técnicas específicas, tais como a Análise Estruturada de Sistemas ou a Modelagem de Dados. Sucesso na implantação e uso das MDS foi relativo. Nas (poucas) organizações onde existiam as condições para seu sucesso, resultados significativos foram alcançados. Em muitos lugares, no entanto, tais resultados não foram os esperados.

Dentre os motivos que levaram ao relativo insucesso das MDS, um dos mais importantes é o foco excessivo que estas colocam nas atividades de Engenharia, em detrimento das atividades de Gerenciamento. Boas técnicas de desenvolvimento não adiantam se o projeto como um todo é mal conduzido.

Combinando esta percepção com os conceitos mais genéricos do Gerenciamento da Qualidade Total, a comunidade começou a priorizar o foco das iniciativas de melhoria na definição de melhores processos de gerenciamento de projetos de software. A consolidação destas idéias se deu através do modelo SEI-CMM, da Carnegie-Mellon University.

O CMM procura orientar a organização no sentido de implementar a melhoria contínua do processo de software, e o faz através de um modelo de 5 níveis, priorizado de forma lógica as ações a serem realizadas. Quanto maior o nível, maior a maturidade da organização, o que se traduz em maior qualidade dos produtos final, prazos e custos mais baixos e maior previsibilidade em cronogramas e orçamentos.

No nível 1, chamado de Inicial, o desenvolvimento é caótico. Não existem procedimentos padronizados, estimativas de custos e planos de projeto. Cada qual desenvolve como quer, não existe documentação e não há mecanismos de controle que permitam ao gerente saber o que está acontecendo, identificar problemas e riscos e agir de acordo. Como conseqüência, os desvios não são corrigidos e ocorrem os problemas como prazos não cumpridos, orçamentos estourados, software sem qualidade e usuários insatisfeitos. Na verdade, raramente existe um cronograma ou um orçamento. Infelizmente, estima-se que mais de três quartos das empresas norte-americanas encontram-se neste nível, e não há razões para acreditar que a situação seja melhor no Brasil.

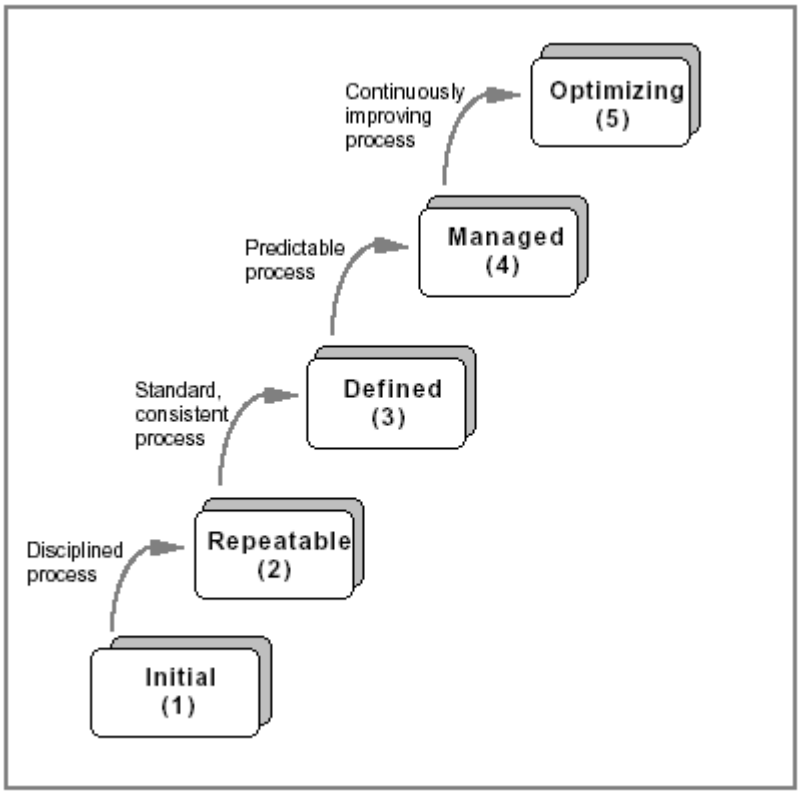


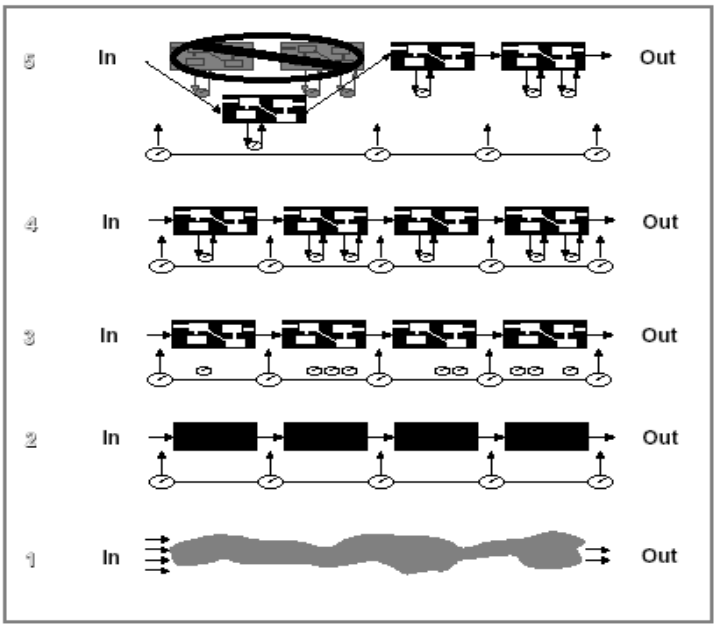
Figure 2.1 The Five Levels of Software Process Maturity

Para passar ao nível 2, a organização deve instituir controles básicos de projeto, incluindo o Gerenciamento de Requisitos e de Projetos (técnicas para planejar e estimar o esforço em projetos, e controlar o progresso), Controle Gerencial (verificação pela Gerência do progresso do projeto em momentos pré-determinados, incluindo a qualidade dos produtos), a instituição de um Grupo de Garantia de Qualidade e de procedimentos básicos de Gerenciamento de Configuração (para garantir que mudanças no projeto e manutenções solicitadas não destruam o que já foi feito, garantindo um mínimo de estabilidade no desenvolvimento; nada é mais deletério para um projeto do que requerimentos que mudam constantemente e sem controle).

Chegando ao nível 2, chamado Repetível, a organização está em condições de ter maior controle sobre seus projetos, e pode-se esperar que as estimativas sejam mais precisas (já que se desenvolve uma base histórica intuitiva) e a qualidade do software produzido seja maior. No entanto, caso a empresa enfrente o desafio de atacar projetos de características distintas das que está acostumada (usando uma nova tecnologia, por exemplo), esta informação será irrelevante, e a empresa poderá regredir ao nível 1.

Para passar ao nível 3, Definido, é necessário introduzir uma Metodologia de Desenvolvimento formal padronizada, com um ciclo de vida definido, acompanhada de métodos, técnicas e ferramentas apropriadas, como inspeções e técnicas abrangentes de teste. É o momento também de estabelecer o SEPG, isto é, o time encarregado exclusivamente da melhoria contínua do processo de software. Ao chegar a este nível, a empresa terá um fundamento claro para desenvolver sistemas e também para melhorar o próprio processo, especialmente quando surgirem crises.

No nível 3, entretanto, os controles ainda são basicamente qualitativos, não havendo meios de quantificar a qualidade dos produtos e a eficiência do processo. Assim, a empresa deve estabelecer métricas de forma a medir características específicas dos produtos. A forma de coletar, armazenar e analisar estes dados é definida e, com base nesta informação, pode-se sugerir melhorias específicas nos produtos. Neste ponto, a empresa estará no nível 4, ou Gerenciado,



Para subir do nível Gerenciado para o último nível, o de Otimização, deve-se estabelecer meios para a coleta automática de métricas e para a utilização da informação coletada de forma a prevenir problemas. A idéia é analisar as causas dos problemas e atacá-las para evitar que volvem a ocorrer. Enquanto os dados coletados no nível 4 podem informar, por exemplo, quantos erros existem em um programa, a preocupação no nível 5 é melhorar o processo para evitar que tais erros aconteçam no próximo projeto.

CMMI (**Capability Maturity Model Integration**)

O **CMMI** (**Capability Maturity Model Integration**) é um modelo de referência que contém práticas (*Genéricas ou Específicas*) necessárias à maturidade em disciplinas específicas. Desenvolvido pelo SEI (*Software Engineering Institute*), o CMMI é uma evolução do CMM e procura estabelecer um modelo único para o processo de melhoria corporativo, integrando diferentes modelos e disciplinas.

O CMMI possui duas representações: "contínua" ou "por estágios". Estas representações permitem a organização utilizar diferentes caminhos para a melhoria de acordo com seu interesse.

Representação Continua

Possibilita à organização utilizar a ordem de melhoria que melhor atende os objetivos de negócio da empresa. É caracterizado por Níveis de Capacidade (*Capability Levels*):

- Nível 0: Incompleto (Ad-hoc)
- Nível 1: Executado (Definido)
- Nível 2: Gerenciado / Gerido
- Nível 3: Definido
- Nível 4: Quantitativamente gerenciado / Gerido quantitativamente
- Nível 5: Em otimização (ou Optimizado)

Representação Por Estágios

Disponibiliza uma seqüência pré-determinada para melhoria baseada em estágios que não deve ser desconsiderada, pois cada estágio serve de base para o próximo. É caracterizado por Níveis de Maturidade (*Maturity Levels*):

- Nível 1: Inicial (Ad-hoc)
- Nível 2: Gerenciado / Gerido
- Nível 3: Definido
- Nível 4: Quantitativamente gerenciado / Gerido quantitativamente
- Nível 5: Em otimização

O modelo CMMI v1.2(CMMI-DEV) contém 22 áreas de processo:

- Análise Causal e Resolução
- Gerência de Configuração
- Análise de Decisão e Resolução
- Gerenciamento Integrado de Projeto
- Medição e Análise
- Inovação Organizacional e Implantação
- Definição de Processo Organizacional
- Foco de Processo Organizacional
- Desempenho de Processo Organizacional
- Treinamento Organizacional
- Monitoração e Controle de Projeto
- Planejamento de Projeto
- Garantia da Qualidade de Processo e Produto
- Integração de Produto
- Gerenciamento Quantitativo de Projeto
- Gerenciamento de Requisitos
- Desenvolvimento de Requisitos
- Gerenciamento de Riscos
- Gerenciamento de Acordo com Fornecedor
- Solução Técnica
- Validação
- Verificação

Entre abril de 2002 e junho de 2006 foram conduzidas 1581 avaliações em 1377 organizações. Segue abaixo o resultado obtido pelas empresas na avaliação (resultados encaminhados para o SEI até 30 de junho de 2006):

- 18,2%: nível 5 (*Optimizing*);
- 4,4%: nível 4 (*Quantitatively Managed*);
- 33,8%: nível 3 (*Defined*);
- 33,3%: nível 2 (*Managed*);
- 1,9%: nível 1 (*Initial*);
- 8,4%: sem qualificação (*Not Given*).

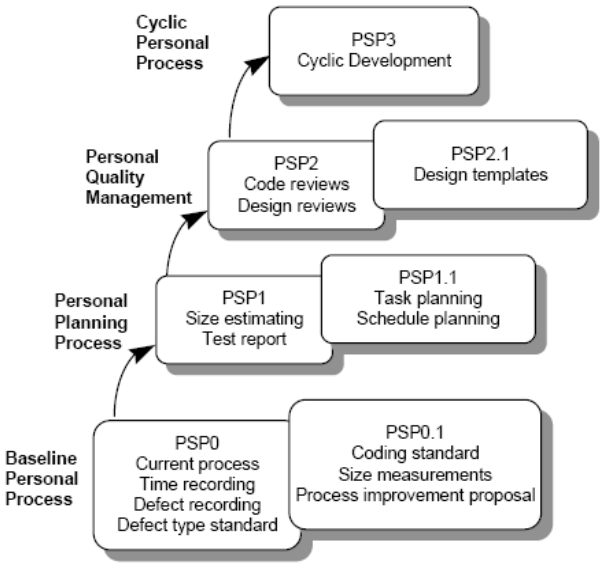
Personal Software Process (PSP)

Personal Software Process (PSP) é um processo de desenvolvimento de software projetado para ser utilizado por engenheiros de software para a elaboração de projetos individuais. Enquanto o CMM é focado na melhoria da capacidade organizacional, o foco do PSP é o engenheiro individual.

Os objetivos principais do PSP são:

- Melhorar a estimativa de prazo e esforço para o desenvolvimento de um módulo de software ou programa;
- Melhorar o planejamento e o acompanhamento de cronogramas;
- Evitar o excesso de compromissos;
- Criar um comprometimento pessoal com a qualidade e com a melhoria contínua do processo;

Os processos do PSP



ATIVIDADES

1. Pesquise o CMMI mais detalhadamente e discuta os prós e contras dos modelos contínuo e por estágios.
2. A partir da especificação do CMMI, selecione uma área de processo e faça uma lista das Metas Específicas e das práticas específicas para esta área.
3. Pesquise quais os benefícios quantitativos do PSP

BIBLIOGRAFIA

PRESSMAN, R. S. Engenharia de Software, 6a. ed.. McGraw-Hill. São paulo, 2006.

PAULK; CURTIS; CHRISSIS; WEBER. *Capability Maturity Model for Software, Version 1.1*. Technical Report. CMU/SEI-93-TR-024 ESC-TR-93-177. February 1993.

WIKIPEDIA. Capability Maturity Model Integration. <http://pt.wikipedia.org/wiki/CMMI> Acessado em 26/08/2009.