Modelos de Processo

Herysson R. Figueiredo
herysson.figueiredo@ufn.edu.br

Sumário

- Modelos de Processo
- Modelo de Processo Genérico
- Modelos de Processo Prescritivo
- Modelos de Processo Especificado
- Processo Unificado

O que é?

Quando se trabalha na elaboração de um produto ou sistema, é importante seguir uma série de passos previsíveis — um roteiro que ajude a criar um resultado de alta qualidade e dentro do prazo estabelecido. O roteiro é denominado “processo de software”.

Quem realiza?

Os engenheiros de software e seus gerentes adaptam o processo às suas necessidades e então o seguem. Os solicitantes do software têm um papel a desempenhar no processo de definição, construção e teste do software.

Por que ele é importante?

Porque propicia estabilidade, controle e organização para uma atividade que pode, sem controle, tornar-se bastante caótica. Entretanto, uma abordagem de engenharia de software moderna deve ser “ágil”. Deve demandar apenas atividades, controles e produtos de trabalho que sejam apropriados para a equipe do projeto e para o produto a ser produzido.

Quais são as etapas envolvidas?

O processo adotado depende do software a ser desenvolvido. Um determinado processo pode ser apropriado para um software do sistema “aviônico” de uma aeronave, enquanto um processo totalmente diferente pode ser indicado para a criação de um site.

Qual é o artefato?

Do ponto de vista de um engenheiro de software, os produtos de trabalho são os programas, os documentos e os dados produzidos em consequência das atividades e tarefas definidas pelo processo.

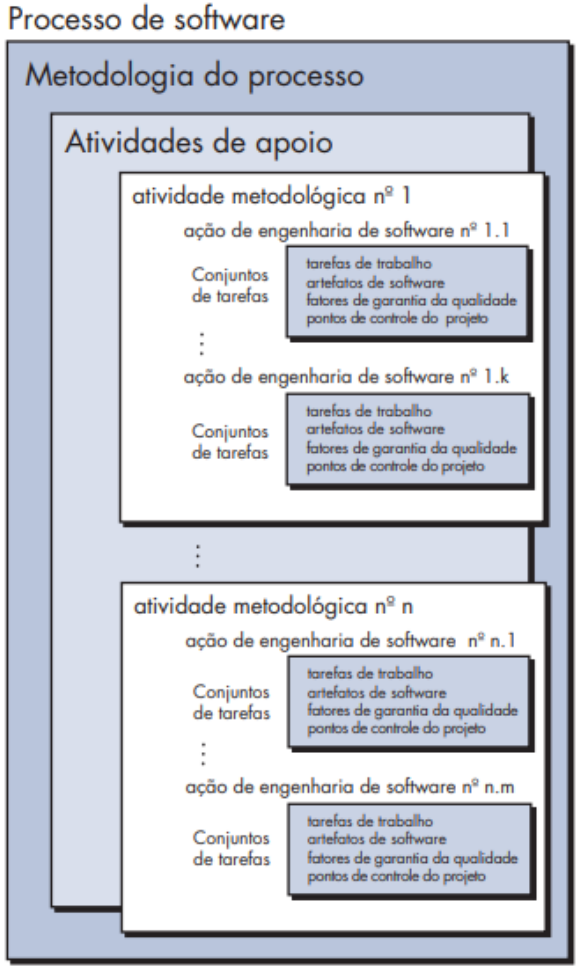
Como garantir que o trabalho foi feito corretamente?

Há muitos mecanismos de avaliação dos processos de software que possibilitam às organizações determinarem o nível de “maturidade” de seu processo de software. Entretanto, a qualidade, o cumprimento de prazos e a viabilidade a longo prazo do produto que se desenvolve são os melhores indicadores da eficácia do processo utilizado.

Modelo de Processo Genérico

Uma metodologia de processo genérica para engenharia de software estabelece cinco atividades metodológicas:

- **comunicação;**
- **planejamento;**
- **modelagem;**
- **construção;**
- **entrega.**



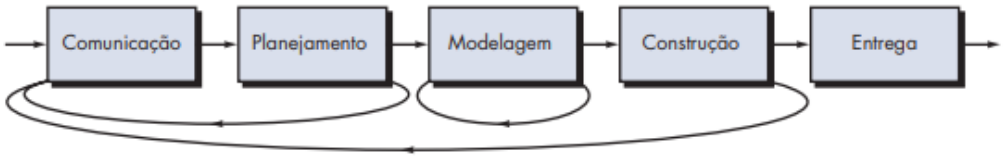
Fluxo de Processo

Um **fluxo de processo linear** executa cada uma das cinco atividades metodológicas em sequência, começando com a de comunicação e culminando com a do emprego.

(a) Fluxo de processo linear

Fluxo de Processo

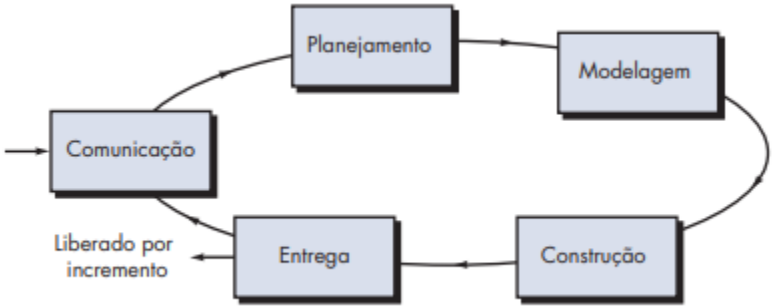
Um **fluxo de processo iterativo** repete uma ou mais das atividades antes de prosseguir para a seguinte.



(b) Fluxo de processo iterativo

Fluxo de Processo

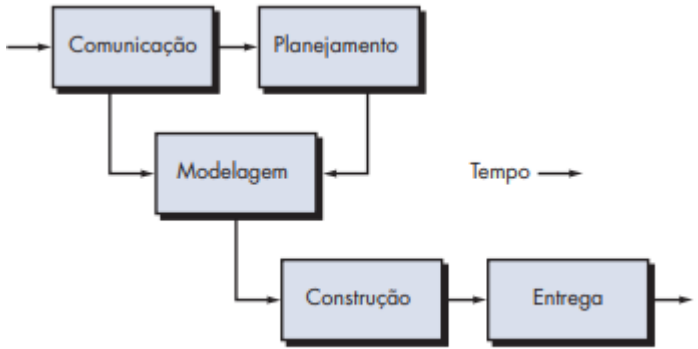
Um f**luxo de processo evolucionário** executa as atividades de uma forma “circular”. Cada volta pelas cinco atividades conduz a uma versão mais completa do software.



(c) Fluxo de processo evolucionário

Fluxo de Processo

Um **fluxo de processo paralelo** executa uma ou mais atividades em paralelo com outras atividades (por exemplo, a modelagem para um aspecto do software poderia ser executada em paralelo com a construção de um outro aspecto do software).



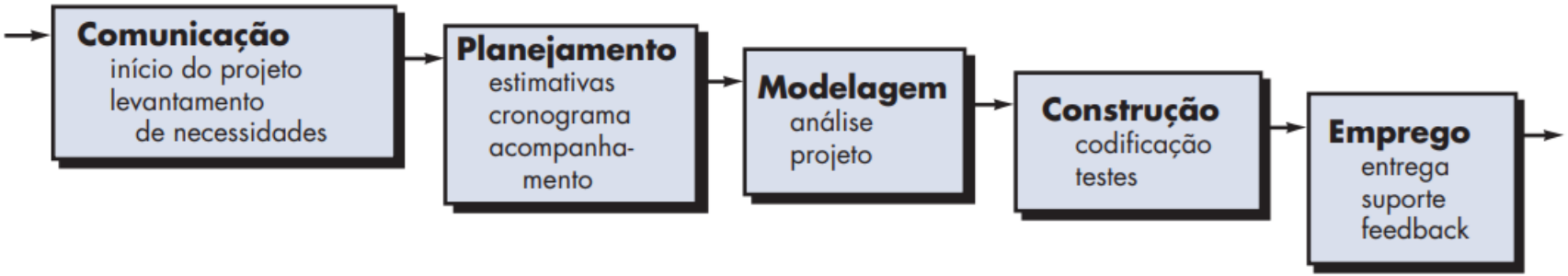
(d) Fluxo de processo paralelo

Modelos de Processos Prescritivos (Tradicionais)

O modelo cascata

O modelo **cascata**, algumas vezes chamado ciclo de vida clássico, sugere uma abordagem sequencial e sistemática para o desenvolvimento de software, começando com o levantamento de necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem, construção, emprego e culminando no suporte contínuo do software concluído

O modelo cascata



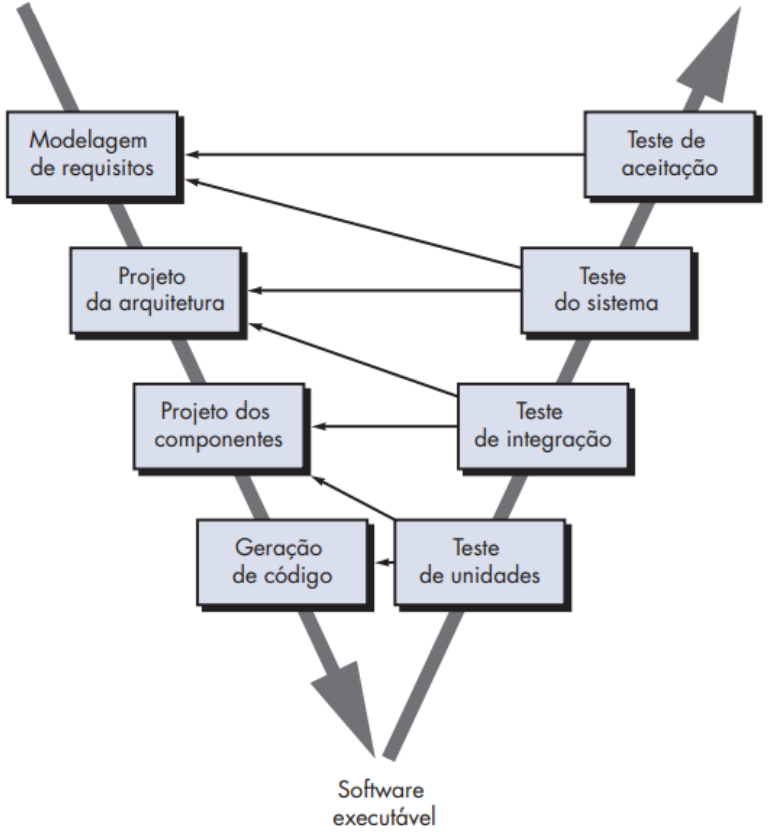
concluído

O modelo V

O **modelo V** descreve a relação entre ações de **garantia da qualidade** e as ações associadas à comunicação, modelagem e atividades de construção iniciais. À medida que a equipe de software desce em direção ao lado esquerdo do V, os requisitos básicos do problema são refinados em representações progressivamente cada vez mais detalhadas e técnicas do problema e de sua solução.

O modelo V

Uma vez que o código tenha sido gerado, a equipe se desloca para cima, no lado direito do V, realizando basicamente uma série de testes (ações de garantia da qualidade) que validem cada um dos modelos criados à medida que a equipe se desloca para baixo, no lado esquerdo do V



Problemas

Problemas às vezes encontrados quando se aplica o modelo cascata, temos:

- Projetos reais raramente seguem o fluxo sequencial que o modelo propõe. Embora o modelo linear possa conter iterações, ele o faz indiretamente. Como consequência, mudanças podem provocar confusão à medida que a equipe de projeto prossegue;

Problemas

Problemas às vezes encontrados quando se aplica o modelo cascata, temos:

- Frequentemente, é difícil para o cliente estabelecer explicitamente todas as necessidades. O modelo cascata requer isso e tem dificuldade para adequar a incerteza natural que existe no início de muitos projetos;

Problemas

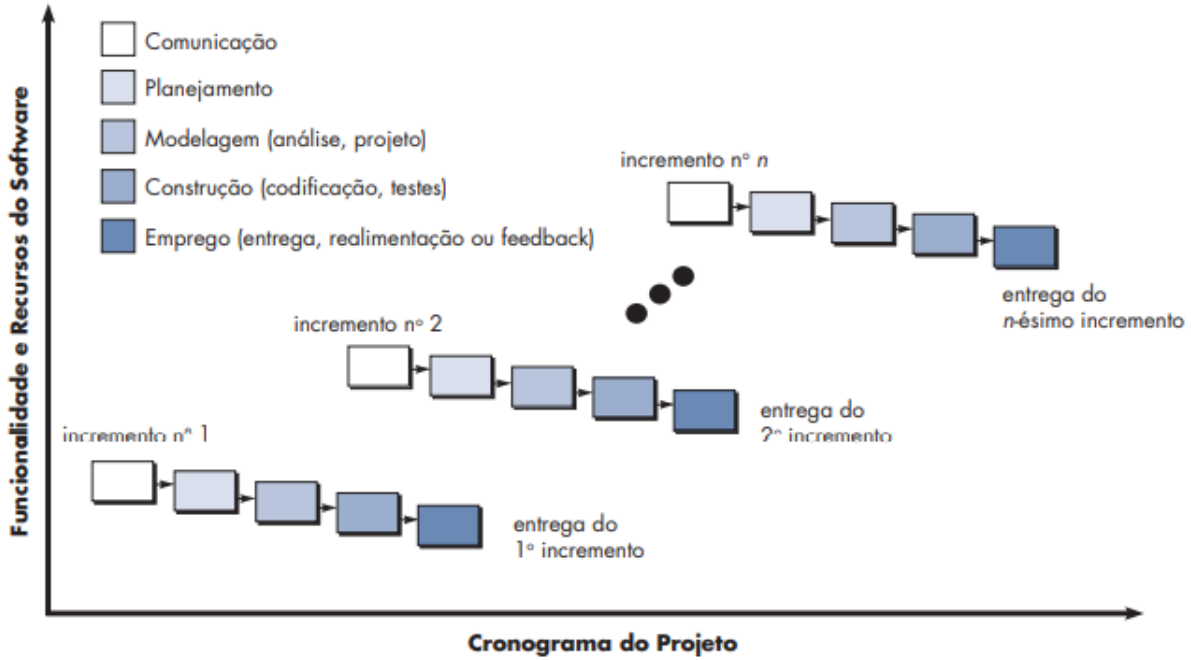
Problemas às vezes encontrados quando se aplica o modelo cascata, temos:

- O cliente deve ter paciência. Uma versão operacional do(s) programa(s) não estará disponível antes de estarmos próximos do final do projeto. Um erro grave, se não detectado até o programa operacional ser revisto, pode ser desastroso.

Modelos de processo incremental

O **modelo incremental** combina elementos dos fluxos de processos **lineares** e **paralelos**, o modelo incremental aplica **sequências lineares**, de forma escalonada, à medida que o tempo vai avançando. Cada sequência linear gera “incrementais” (entregáveis/aprovados/liberados) do software.

Modelos de processo incremental



Modelos de processo evolucionário

Modelos evolucionários são iterativos. Apresentam características que possibilitam desenvolver versões cada vez mais completas do software. Nos parágrafos seguintes, são apresentados dois modelos comuns em processos evolucionários.

Prototipação

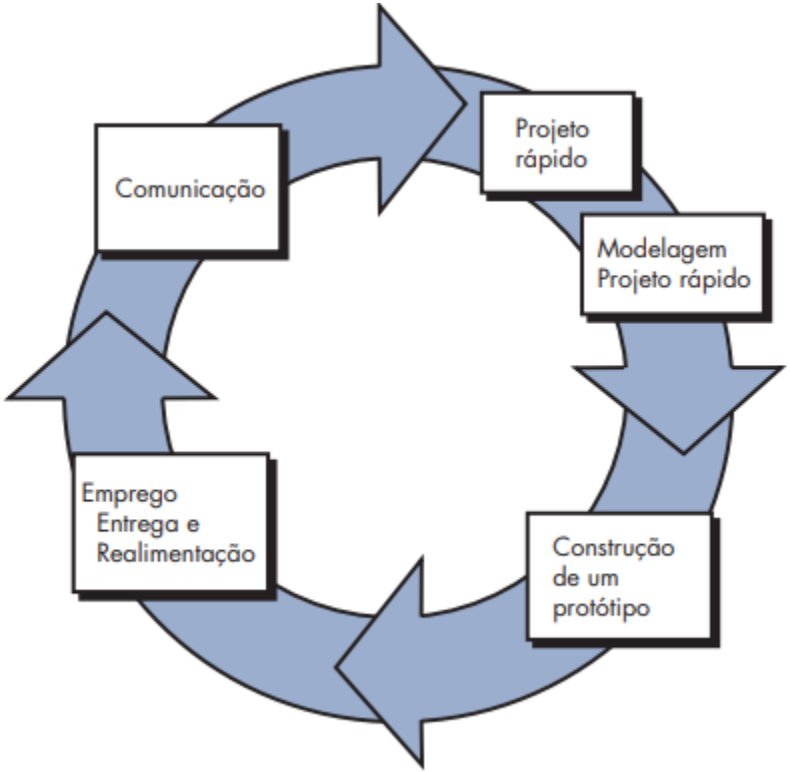
O paradigma da **prototipação** começa com a comunicação. Faz-se uma reunião com os envolvidos para definir os objetivos gerais do software, identificar quais requisitos já são conhecidos e esquematizar quais áreas necessitam, obrigatoriamente, de uma definição mais ampla . Uma iteração de prototipação é planejada rapidamente e ocorre a modelagem (na forma de um "**projeto rápido**")

> Um projeto rápido se concentra em uma representação daqueles aspectos do software que serão visíveis aos usuários finais (por exemplo, o layout da interface com o usuário ou os formatos de exibição na tela).

Prototipação

O projeto rápido leva à construção de um **protótipo**, que é empregado e avaliado pelos envolvidos, que fornecerão um retorno (***feedback***), que servirá para **aprimorar os requisitos**. A iteração ocorre conforme se ajusta o protótipo às necessidades de vários interessados e, ao mesmo tempo, possibilita a melhor compreensão das necessidades que devem ser atendidas.

Prototipação



Problemas

A prototipação pode ser problemática pelas seguintes razões:

- Os interessados enxergam o que parece ser uma versão operacional do software, ignorando que o protótipo é mantido de forma não organizada e que, na pressa de fazer com que ele se torne operacional, não se considera a qualidade global do software, nem sua manutenção a longo prazo.

Problemas

A prototipação pode ser problemática pelas seguintes razões:

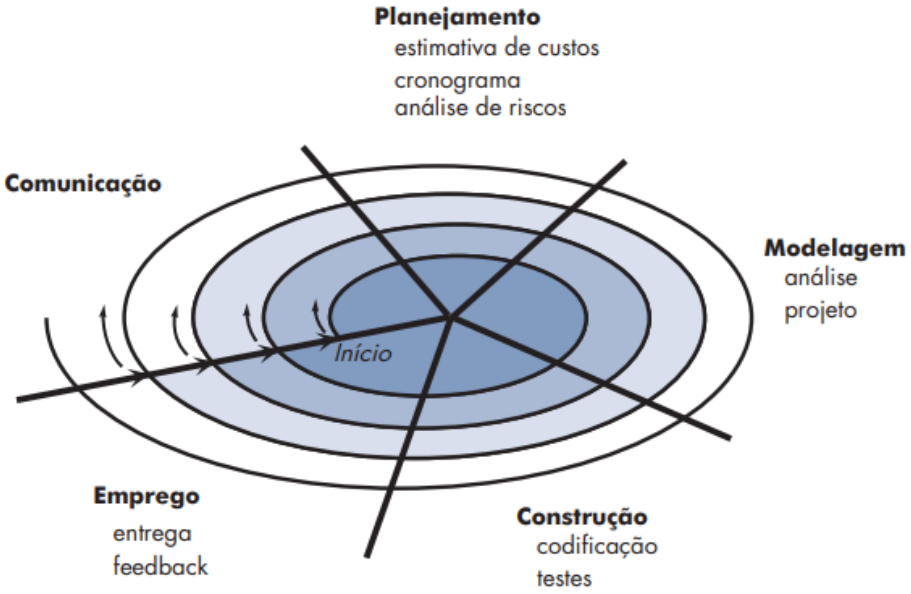
- O engenheiro de software, com frequência, assume compromissos de implementação para conseguir que o protótipo entre em operação rapidamente.

Modelo Espiral

Originalmente proposto por Barry Boehm, o modelo espiral é um modelo de processo de software evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata. Fornece potencial para o rápido desenvolvimento de versões cada vez mais completas do software.

Modelo Espiral

O modelo espiral de desenvolvimento é um gerador de modelos de processos dirigidos a riscos e é utilizado para guiar a engenharia de sistemas intensivos de software, que ocorre de forma concorrente e tem múltiplos envolvidos.



Problemas

O **modelo espiral** pode ser problemática pelas seguintes razões:

- Pode ser difícil convencer os clientes (particularmente em situações contratuais) de que a abordagem evolucionária é controlável;
- Ela exige considerável especialização na avaliação de riscos e depende dessa especialização para seu sucesso;
- Se um risco muito importante não for descoberto e administrado, indubitavelmente ocorrerão problemas.

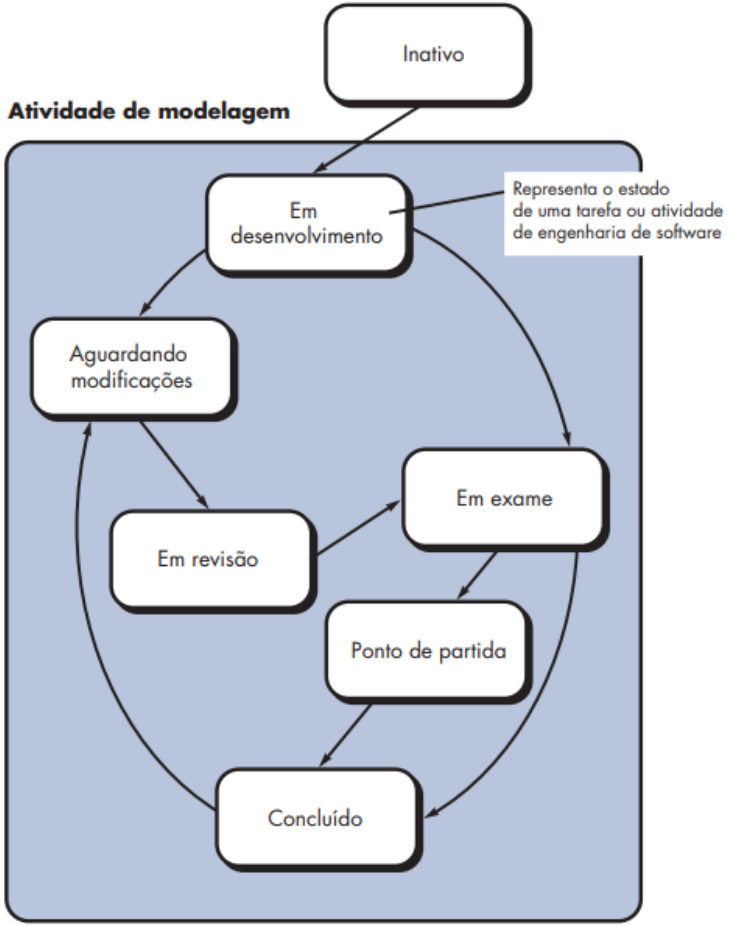
Modelo Concorrente

O **modelo de desenvolvimento concorrente**, algumas vezes denominado engenharia concorrente, possibilita à equipe de software representar elementos concorrentes e iterativos de qualquer um dos modelos de processos descritos neste capítulo. Por exemplo, a atividade de modelagem definida para o modelo espiral é realizada invocando uma ou mais das seguintes ações de engenharia de software: prototipagem, análise e projeto.

Modelo Concorrente

A figura ao lado mostra um esquema de uma atividade da engenharia de software, dentro da atividade de **modelagem**, usando uma abordagem de **modelagem** concorrente.

Similarmente, outras atividades, ações ou tarefas (por exemplo, **comunicação** ou **construção**) podem ser representadas de maneira análoga. Todas as atividades de engenharia de software existem **concorrentemente**, porém estão em diferentes estados.



Modelos de Processo Especializados

Modelos de Processo Especializados

Os modelos de processo especializado levam em conta muitas das características de um ou mais dos modelos tradicionais. Tais modelos tendem a ser aplicados quando se opta por uma abordagem de engenharia de software especializada ou definida de forma restrita.

Desenvolvimento baseado em componentes

O **modelo de desenvolvimento baseado em componentes** incorpora muitas das características do modelo espiral. É evolucionário em sua natureza, demandando uma abordagem iterativa para a criação de software. O modelo de desenvolvimento baseado em componentes desenvolve aplicações a partir de componentes de software pré-empacotados.

O modelo de métodos formais

O **modelo de métodos formais** engloba um conjunto de atividades que conduzem à especificação matemática formal do software. Os métodos formais possibilitam especificar, desenvolver e verificar um sistema baseado em computador através da aplicação de uma notação matemática rigorosa.

Desenvolvimento de *software* orientado a aspectos

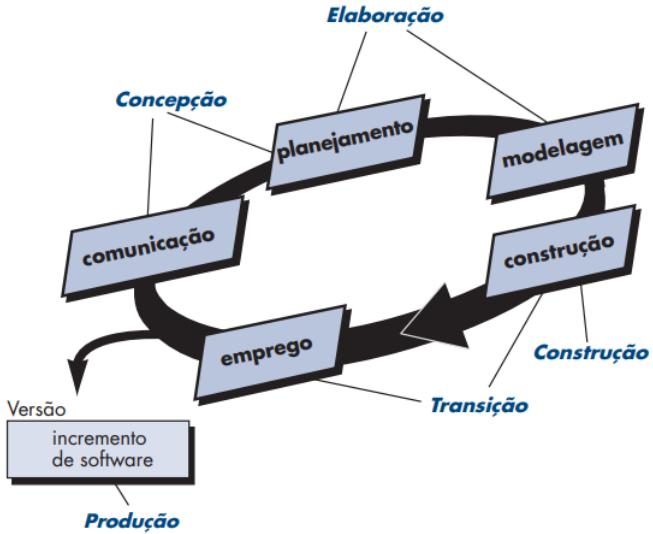
O desenvolvimento de software orientado a aspectos AOSD, *Aspect-Oriented Software Development*), com frequência conhecido como programação orientada a aspectos (AOP, *Aspect-Oriented Programming*), é um paradigma de engenharia de software relativamente novo que oferece uma abordagem metodológica e de processos para definir, especificar, projetar e construir aspectos — "mecanismos além das sub-rotinas e herança para localizar a expressão de uma restrição cruzada"

Quando restrições cruzam múltiplas funções, recursos e informações do sistema, elas são, frequentemente, denominadas **restrições cruzadas**

O Processo Unificado

O Processo unificado

O **Processo Unificado** é uma tentativa de aproveitar os melhores recursos e características dos modelos tradicionais de processo de software, mas caracterizando-os de modo a implementar muitos dos melhores princípios do desenvolvimento ágil de software.



Bibliografia

BEZERRA, Eduardo. Princípios de análise e projeto de sistemas com UML. 2. ed. rev. atual. Rio de Janeiro, RJ: Campus, 2007. 369 p.

LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. 3. ed., reimpr. Porto Alegre: Bookman, 2007. 695 p. ISBN 978-85-60031-52-8

WAZLAWICK, Raul Sidnei. Análise e projeto de sistemas de informação orientados a objetos. 2. ed. rev. e atual. Rio de Janeiro: Elsevier, 2011. 330 p. (Série SBC, Sociedade Brasileira de computação) ISBN 978-85-352-3916-4

PRESSMAN, Roger S. Engenharia de software. 6. ed. Rio de Janeiro, RJ: McGraw Hill, 2006. 720 p