Análise e Projeto de Sistemas

O Processo de Software

Prof. Laudelino Cordeiro Bastos

Abrangência da Engenharia de Software

- A Engenharia de Software, que teve origem na Engenharia de Sistemas e de Hardware, abrange um conjunto de três elementos fundamentais: Métodos, Ferramentas e Procedimentos (Processos).
- Tal conjunto tem como objetivos:
  - Possibilitar ao gerente o controle do processo de desenvolvimento.
  - Oferecer ao profissional uma base para a construção de software de alta qualidade.

Métodos da Engenharia de Software

- Os métodos proporcionam os detalhes de "como fazer" para construir o software. Eles envolvem um amplo conjunto de tarefas:
  - Planejamento e estimativa de projeto.
  - Análise de requisitos de software.
  - Projeto da estrutura de dados.
  - Arquitetura do programa e algoritmos de processamento.
  - Codificação.
  - Testes.
  - Manutenção.

Ferramentas da Engenharia de Software

- As ferramentas fornecem suporte automatizado ou semi-automatizado aos métodos:
  - Atualmente existem ferramentas para sustentar cada um dos métodos.
  - Quando as ferramentas são integradas é estabelecido um sistema de suporte ao desenvolvimento de software chamado Computer Aided Software Engineering (CASE).

Procedimentos da Engenharia de Software

- Os procedimentos constituem o elo de ligação entre os métodos e ferramentas. Eles definem:
  - A seqüência em que os métodos serão aplicados.
  - Os produtos que se exige que sejam entregues (documentos, relatórios, formulários, diagramas, código).
  - Os controles que ajudam a assegurar a qualidade e coordenar as alterações.
  - Os marcos de referência que possibilitam administrar o progresso do software.

Paradigmas da Engenharia de Software

- Os paradigmas da Engenharia de Software compreendem um conjunto de etapas que envolve métodos, ferramentas e procedimentos.
- Essas etapas são chamadas de Ciclos de Vida ou Modelos de Processo de Software.
- Os ciclos de vida são uma estratégia de desenvolvimento.

Escolha de um Modelo de Processo

- Qual processo de desenvolvimento será adotado? Para responder essa pergunta, é necessário analisar:
  - Verificação da adequação do modelo de processo à aplicação.
  - Métodos e ferramentas a serem utilizados.
  - Controles e produtos que precisam ser entregues.
  - Características dos processos: visibilidade, clareza, apoio de ferramentas, produtividade, qualidade, etc.

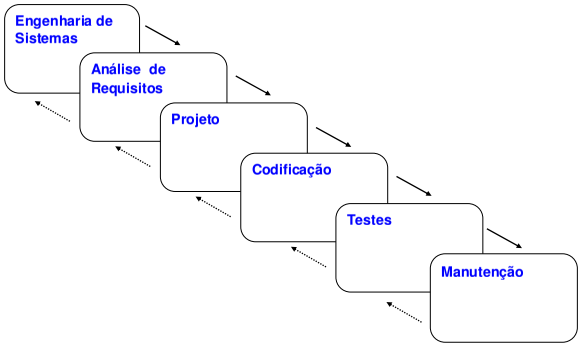
Modelo de Ciclo de Vida Clássico (Cascata)

- Modelo mais antigo da engenharia de software.
- Modelado em função do ciclo da engenharia convencional.
- Requer uma abordagem sistemática e sequencial para o desenvolvimento de software.
- Cada atividade é uma fase em separado. A passagem entre fases é formal.

Modelo de Ciclo de Vida Clássico (Cascata)

- Cada fase envolve a elaboração de um ou mais documentos, que devem ser aprovados antes de se iniciar a fase seguinte. Assim, uma fase só deve ser iniciada após a conclusão daquela que a precede.
- Na prática, como essas fases se sobrepõem de alguma forma, geralmente se permite um retorno à fase anterior para a correção dos erros encontrados.

Modelo de Ciclo de Vida Clássico



Ploblemas com o Ciclo de Vida Básico

- Os projetos reais raramente seguem o fluxo seqüencial que o modelo propõe.
- É difícil estabelecer explicitamente, logo no início, todos os requisitos. No começo dos projetos sempre existe uma incerteza natural.
- O cliente deve ter paciência. Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento.

Vantagens do Ciclo de Vida Clássico

- A visibilidade do processo permite um fácil entendimento do mesmo.
- Embora o Ciclo de Vida Clássico tenha fragilidades, ele é significativamente melhor do que uma abordagem casual no desenvolvimento de software.

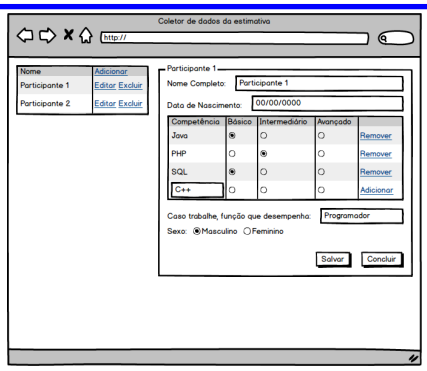
Prototipação (1)

- Processo que possibilita que o desenvolvedor crie um modelo do software que deve ser construído.
- Idealmente, o modelo (protótipo) serve como um mecanismo para identificar os requisitos de software:
  - Protótipo do sistema que retrata a interação com o usuário.
  - Protótipo que implemente algumas funções exigidas.

Prototipação (2)

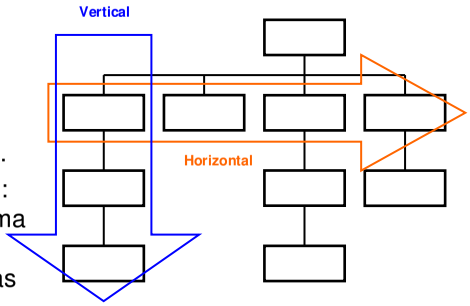
- A Prototipação pode ser classificada em:
  - Prototipação de baixa fidelidade, quando os protótipos são desenhados na forma de esboços e sem muitos detalhes. Os protótipos criados nesse tipo de prototipação são também chamados de mockups ou wireframes.
  - Prototipação de alta fidelidade, quando os protótipos apresentam um nível alto de detalhes do ponto de vista da aparência. São construídos utilizando a própria ferramenta de desenvolvimento de software.

Prototipação (3): Exemplo de Mockup



Prototipação (4): Vertical e Horizontal

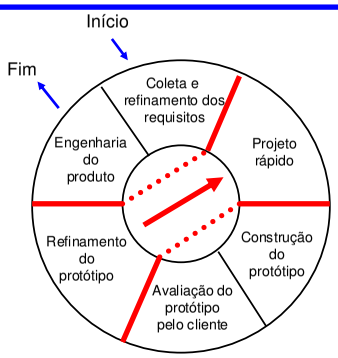
- Vertical: visão profunda, mas específica.
- Horizontal: oferece uma visão ampla, mas rasa.



Prototipação (5)

- A prototipação é apropriada quando:
  - O cliente definiu um conjunto de objetivos gerais para o software, mas não identificou requisitos de entrada, processamento e saída com detalhes.
  - O desenvolvedor não tem certeza da eficiência de um algoritmo ou da forma da interação homem/máquina.

Prototipação (3)



Atividades na Prototipação (1)

- Coleta e Refinamento dos Requisitos: desenvolvedor e cliente definem os objetivos gerais do software, identificam quais requisitos são conhecidos e as áreas que necessitam de definições adicionais.
- Projeto Rápido: representação dos aspectos do software que são visíveis ao usuário, como as abordagens de entrada e os formatos de saída.

Atividades na Prototipação (2)

- Construção do Protótipo: a implementação do projeto rápido serve como o "primeiro sistema", recomendado que se deixe de lado futuramente.
- Avaliação do Protótipo: cliente e desenvolvedor avaliam o protótipo.

Atividades na Prototipação (3)

- Refinamento do Protótipo: cliente e desenvolvedor refinam os requisitos do software a ser desenvolvido. Ocorre nesse ponto um processo de iteração até que as necessidades do cliente sejam satisfeitas e o desenvolvedor compreenda o que precisa ser feito.
- Engenharia do Produto: identificados os requisitos, o protótipo deve ser descartado e a versão de produção deve ser construída considerando os critérios de qualidade.

Problemas com a Prototipação

- O cliente não sabe que o software, que ele vê, não considerou, durante o desenvolvimento, a qualidade global e a manutenibilidade a longo prazo:
  - Ele não aceita bem a idéia que a versão final do software será construída e "força" a utilização do protótipo como produto final.
- O desenvolvedor freqüentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo:
  - Depois de um tempo, ele se familiariza com essas escolhas, e esquece que elas não são apropriadas para o produto final.

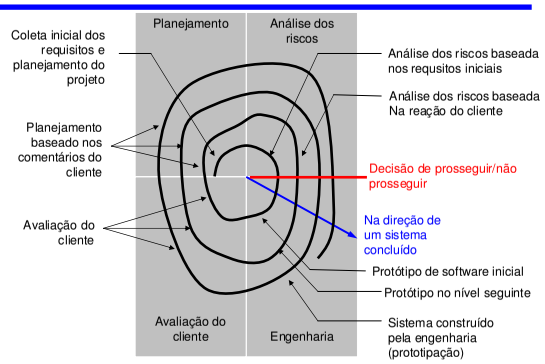
Vantagens da Prototipação

- Ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente:
  - A chave é definir as regras do jogo logo no começo.
  - O cliente e o desenvolvedor devem ambos concordar que o protótipo seja construído para servir como um mecanismo a fim de definir os requisitos.

Ciclo de Vida em Espiral (1)

- Engloba as melhores características do ciclo de vida Clássico como o da Prototipação, adicionando um novo elemento: a Análise dos Riscos.
- Segue a abordagem de passos sistemáticos do Ciclo de Vida Clássico incorporando-os numa Estrutura Iterativa que reflete mais realisticamente o mundo real.
- Usa a Prototipação, em qualquer etapa da evolução do produto, como mecanismo de redução de riscos.

Ciclo de Vida em Espiral (2)



Atividades do Ciclo de Vida em Espiral

- Planejamento: determinação dos objetivos, alternativas e restrições.
- Análise de Risco: análise das alternativas, identificação e resolução dos riscos.
- Construção: desenvolvimento do produto no nível seguinte.
- Avaliação do Cliente: avaliação do produto e planejamento das novas fases.

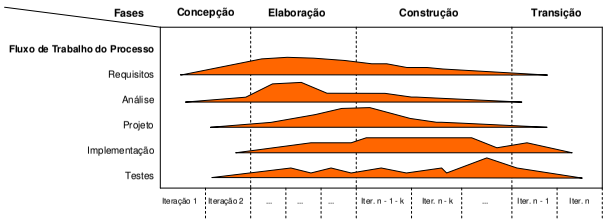
Comentários sobre o Ciclo de Vida Espiral

- É uma abordagem realística para o desenvolvimento de software em grande escala.
- Usa uma abordagem que capacita o desenvolvedor e o cliente a entenderem e a reagirem aos riscos em cada etapa evolutiva.
- Pode ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável.
- Exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso.
- A cada iteração ao redor da espiral, versões progressivamente mais completas do software são construídas.

Processo Unificado

- O Processo Unificado é um modelo de processo de software bastante elaborado, que procura incorporar elementos de vários dos modelos de processo anteriormente apresentados, em uma tentativa de incorporar as melhores práticas de desenvolvimento de software, dentre elas a prototipação, a iteração e a entrega incremental.
- Usa uma abordagem de orientação a objetos em sua concepção e é projetado e documentado utilizando a notação UML para ilustrar os processos em ação.

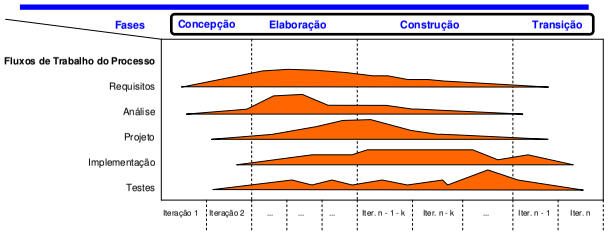
Processo Unificado



Fases do Processo Unificado

- O Processo Unificado considera quatro fases no desenvolvimento de software:
  - Iniciação ou Concepção (Inception): ênfase no escopo do sistema.
  - Elaboração (Elaboration): ênfase na arquitetura.
  - Construção (Construction): ênfase no desenvolvimento.
  - Transição (Transition): ênfase na implantação.
- Cada fase é composta pela iteração dos seguintes Fluxos de Trabalho do Processo (Disciplinas):
  - Requisitos.
  - Análise.
  - Projeto.
  - Implementação.
  - Testes.

Processo Unificado - Fases



Concepção - Fases do Processo Unificado

- Durante a fase de concepção, o foco está na comunicação com o cliente para a identificação de requisitos e nas atividades de planejamento, a fim de delimitar o escopo do projeto.
- Para conseguir isso, devem ser identificadas todas as entidades externas com as quais o sistema irá interagir (atores), definindo a natureza dessa interação em um alto nível, identificando todos os casos de uso e o plano do projeto.
- Protótipos podem ser construídos para apoiar a comunicação com o cliente.

Elaboração - Fases do Processo Unificado

- Os objetivos desta fase são analisar o domínio do problema, estabelecer a arquitetura do mesmo, refinar o plano do projeto e identificar seus maiores riscos.
- Em termos do processo de desenvolvimento, o foco são as atividades de análise e projeto.
- Decisões sobre a arquitetura tem que ser feitas com uma compreensão de todo o sistema: seu escopo, suas principais funcionalidades e seus requisitos não funcionais, tais como os requisitos de desempenho.

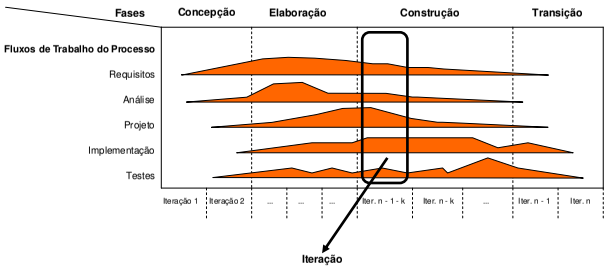
Construção - Fases do Processo Unificado

- Na fase de construção, todos os demais componentes e as características da aplicação são desenvolvidos e integrados ao produto, sendo que todas as características são exaustivamente testadas.
- A fase de construção é, em certo sentido, um processo de fabricação onde a ênfase é colocada sobre o gerenciamento de recursos e controle de operações para otimizar custos, programações e qualidade.
- O gerenciamento do processo executa uma transição a partir do desenvolvimento da propriedade intelectual durante as fases de concepção e elaboração, para o desenvolvimento do software e da documentação a serem entregues durante a construção e a transição.

Transição - Fases do Processo Unificado

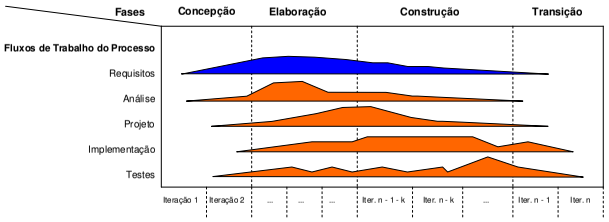
- O propósito desta fase é fazer a transição do sistema do ambiente de desenvolvimento para o ambiente de produção, ou seja para os usuários.
- Uma vez que o produto tenha sido entregue para o usuário final, surgem normalmente questões como o desenvolvimento de novas versões, a correção de alguns problemas ou mesmo desenvolver os características que foram adiadas.

Processo Unificado



- As iterações ocorrem com a execução dos fluxos de trabalho de cima para baixo.

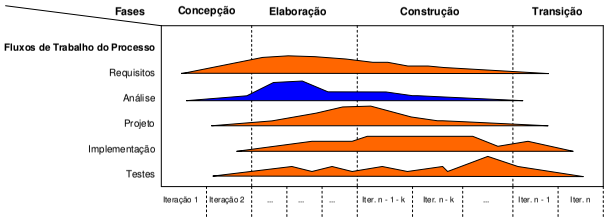
Requisitos (Processo Unificado)



Requisitos

- O objetivo do fluxo de trabalho de Requisitos é descrever o que o sistema deve fazer. Para conseguir isso, deve-se obter, organizar, documentar as funcionalidades e restrições exigidas.
- A modelagem de Casos de Uso representa os Requisitos Funcionais.
- Um Caso de Uso especifica uma seqüência de ações, incluindo caminhos alternativos, que o sistema realiza e que levam a um resultado observável produzido por um determinado ator.

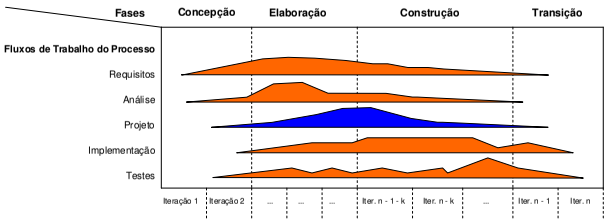
Análise (Processo Unificado)



Análise

- Neste fluxo de trabalho são analisados os requisitos e é montado o modelo de classes e de objetos, com foco nas classes de negócio, mais o dicionário de informações.
- O Modelo de Caso de Uso é a entrada para o Modelo de Análise.
- O Modelo de Análise é uma especificação detalhada dos requisitos, sendo usado pelos desenvolvedores para uma compreensão mais precisa dos Casos de Uso, definindo-os como uma colaboração entre tipos conceituais de objetos.

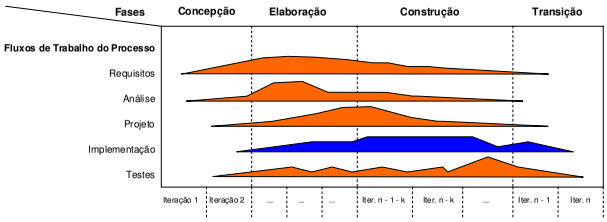
Projeto (Processo Unificado)



Projeto

- O Modelo de Projeto é uma descrição da implementação do sistema, a partir do Modelo de Análise, e precisa se adequar ao ambiente de implementação (tecnologia de distribuição de objetos, ambiente gráfico, bancos de dados, reuso de sistemas legados, bibliotecas).
- São incluídos os Diagramas de Sequencia, de Estado e de Atividades.

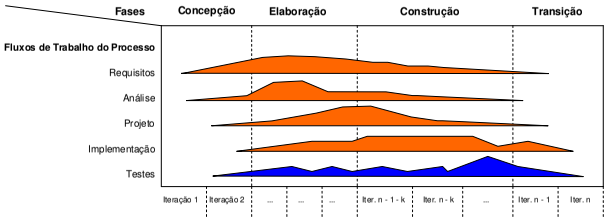
Implementação (Processo Unificado)



Implementação

- Os objetivos da implementação são:
  - Definir a organização do código em termos de subsistemas de implementação, organizados em camadas.
  - Implementar classes e objetos em termos de componentes (arquivos-fonte, códigos binários, executáveis e outros).
  - Testar os componentes desenvolvidos como unidades.
  - Integrar os resultados produzidos por implementadores individuais, ou por equipes, em um sistema executável.

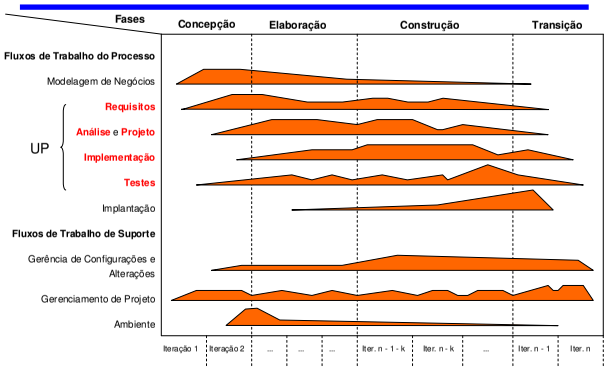
Testes (Processo Unificado)



Testes

- Os objetivos da atividade de Testes são:
  - Verificar a interação entre os objetos.
  - Verificar a integração adequada de todos os componentes do software.
  - Verificar se todos os requisitos foram corretamente implementados.
  - Identificar e garantir que os defeitos sejam encontrados e corrigidos antes da implantação do software.

Processo Unificado Rational (RUP)



Processo Unificado Rational (RUP)

- O nome do Processo Unificado (*Unified Process*), em relação ao Processo Unificado Rational (Rational *Unified Process*) é geralmente usado para descrever o processo genérico, incluindo os elementos que são comuns à maioria dos refinamentos. O nome Processo Unificado é também usado para evitar possíveis problemas de violação de marca registrada sobre o Rational *Unified Process* e RUP, que são marcas registradas da IBM.

Obrigado.