Curso de Sistemas de Informações

ENGENHARIA DE SOFWARE

***I* Processo de desenvolvimento de software**
e
***II* Modelos de processo de desenvolvimento de software**

Prof. André Luiz de Castro Leal
Doutorando em Informática pela PUC-RIO
Mestrado em Ciência da Computação pela UFV
Especialista em Gestão de Ti
Especialista em Ciência da Computação
andrecastro@ufrrj.br

Engenharia de Software

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Macro Estrutura do Conteúdo

1) Definição
2) Elementos
3) Modelos de Ciclo de Vida ou Modelos de Processo
   3.1 Modelos Sequenciais
   3.2 Modelos Incrementais
   3.3 Modelos Evolutivos
   3.4 Prototipação
4) Automatização de Processos

O que é um processo de software

Um processo de software pode ser visto como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software. Um processo eficaz deve, claramente, considerar as relações entre as atividades, os artefatos produzidos no desenvolvimento, as ferramentas e os procedimentos necessários e a habilidade, o treinamento e a motivação do pessoal envolvido.

Elementos que compõem um processo de software

Os processos de software são, geralmente, decompostos em diversos processos:
- Processo de desenvolvimento
- Processo de garantia da qualidade
- Processo de gerência, entre outros.

Esses processos, por sua vez, são compostos de atividades, que também podem ser decompostas. Para cada atividade de um processo é importante saber quais as suas sub-atividades, as atividades que devem precedê-las (pré-atividades), os artefatos de entrada (insumos) e de saída (produtos) da atividade, os recursos necessários (humanos, hardware, software etc) e os procedimentos (métodos, técnicas, modelos de documento etc) a serem utilizados na sua realização.

UFRRJ
Engenharia de Software

Elementos que compõem um processo de software

- Processo de Software
  - Processos
    - Atividades
      - Pré-atividades
      - Sub-atividades
      - Artefatos
        - Insumos
        - Produtos
      - Recursos
        - Recursos Humanos
        - Ferramentas de Software
        - Hardware
      - Procedimentos
        - Métodos
        - Técnicas
        - Roteiros

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Definição de Processos

O objetivo de se definir um processo de software é favorecer a produção de sistemas de alta qualidade, atingindo as necessidades dos usuários finais, dentro de um cronograma e um orçamento previsíveis.

Um processo de software não pode ser definido de forma universal. Para ser eficaz e conduzir à construção de produtos de boa qualidade, um processo deve ser adequado às especificidades do projeto em questão. Deste modo, processos devem ser definidos caso a caso, considerando-se as características da aplicação (domínio do problema, tamanho, complexidade etc), a tecnologia a ser adotada na sua construção (paradigma de desenvolvimento, linguagem de programação, mecanismo de persistência etc), a organização onde o produto será desenvolvido e a equipe de desenvolvimento.

UFRRJ
Engenharia de Software

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Definição de Processos e Ciclo de Vida

A escolha de um modelo de ciclo de vida (ou modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida, geralmente, organiza as macro-atividades básicas do processo, estabelecendo precedência e dependência entre as mesmas. Para a definição completa do processo, cada atividade descrita no modelo de ciclo de vida deve ser decomposta e para suas sub-atividades, devem ser associados métodos, técnicas, ferramentas e critérios de qualidade, entre outros, formando uma base sólida para o desenvolvimento. Adicionalmente, outras atividades tipicamente de cunho gerencial, devem ser definidas, entre elas atividade de gerência de projetos e de controle e garantia da qualidade.

Os seguintes fatores influenciam a definição de um processo e, por conseguinte, a escolha do modelo de processo a ser usado como base: tipo de software (p.ex., sistema de informação, sistema de tempo real etc), paradigma de desenvolvimento (estruturado, orientado a objetos etc), domínio da aplicação, tamanho e complexidade do sistema, estabilidade dos requisitos, características da equipe etc.

Definição de Processos e Ciclo de Vida

Um modelo de ciclo de vida ou modelo de processo pode ser visto como uma representação abstrata de um esqueleto de processo, incluindo tipicamente algumas atividades principais, a ordem de precedência entre elas e, opcionalmente, artefatos requeridos e produzidos. De maneira geral, um modelo de processo descreve uma filosofia de organização de atividades, estruturando as atividades do processo em fases e definindo como essas fases estão relacionadas. Entretanto, ele não descreve um curso de ações preciso, recursos, procedimentos e restrições. Ou seja, ele é um importante ponto de partida para definir como o projeto deve ser conduzido, mas a sua adoção não é o suficiente para guiar e controlar um projeto de software na prática.

Ainda que os processos tenham de ser definidos caso a caso, de maneira geral, o ciclo de vida de um software envolve, pelo menos, as seguintes fases:

UFRRJ
Engenharia de Software

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Definição de Processos e Ciclo de Vida: Fases

a) Planejamento: O objetivo do planejamento de projeto é fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custos e prazos. Uma vez estabelecido o escopo de software, com os requisitos esboçados, uma proposta de desenvolvimento deve ser elaborada, isto é, um plano de projeto deve ser elaborado configurando o processo a ser utilizado no desenvolvimento de software. À medida que o projeto progride, o planejamento deve ser detalhado e atualizado regularmente. Pelo menos ao final de cada uma das fases do desenvolvimento (análise e especificação de requisitos, projeto, implementação e testes), o planejamento como um todo deve ser revisto e o planejamento da etapa seguinte deve ser detalhado. O planejamento e o acompanhamento do progresso fazem parte do processo de gerência de projeto.

b) *Análise e Especificação de Requisitos*: Nesta fase, o processo de levantamento de requisitos é intensificado. O escopo deve ser refinado e os requisitos mais bem definidos. Para entender a natureza do software a ser construído, o engenheiro de software tem de compreender o domínio do problema, bem como a funcionalidade e o comportamento esperados. Uma vez capturados os requisitos do sistema a ser desenvolvido, estes devem ser modelados, avaliados e documentados. Uma parte vital desta fase é a construção de um modelo descrevendo *o que* o software tem de fazer (e não *como* fazê-lo).

Definição de Processos e Ciclo de Vida: Fases

c) Projeto: Esta fase é responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema, modelados na fase anterior e, portanto, requer que a plataforma de implementação seja conhecida. Basicamente, envolve duas grandes etapas: projeto da arquitetura do sistema e projeto detalhado. O objetivo da primeira etapa é definir a arquitetura geral do software, tendo por base o modelo construído na fase de análise de requisitos. Essa arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes. O propósito do projeto detalhado é detalhar o projeto do software para cada componente identificado na etapa anterior. Os componentes de software devem ser sucessivamente refinados em níveis maiores de detalhamento, até que possam ser codificados e testados.

d) *Implementação*: O projeto deve ser traduzido para uma forma passível de execução pela máquina. A fase de implementação realiza esta tarefa, isto é, cada unidade de software do projeto detalhado é implementada.

e) *Testes*: inclui diversos níveis de testes, a saber, teste de unidade, teste de integração e teste de sistema. Inicialmente, cada unidade de software implementada deve ser testada e os resultados documentados. A seguir, os diversos componentes devem ser integrados sucessivamente até se obter o sistema. Finalmente, o sistema como um todo deve ser testado.

UFRRJ
Engenharia de Software

Definição de Processos e Ciclo de Vida: Fases

f) Entrega e Implantação: uma vez testado, o software deve ser colocado em produção. Para tal, contudo, é necessário treinar os usuários, configurar o ambiente de produção e, muitas vezes, converter bases de dados. O propósito desta fase é estabelecer que o software satisfaz os requisitos dos usuários. Isto é feito instalando o software e conduzindo testes de aceitação. Quando o software tiver demonstrado prover as capacidades requeridas, ele pode ser aceito e a operação iniciada.

g) *Operação*: nesta fase, o software é utilizado pelos usuários no ambiente de produção.

h) *Manutenção*: software sofre mudanças após ter sido entregue para o usuário. Alterações ocorrerão porque erros foram encontrados, porque o software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho. Muitas vezes, dependendo do tipo e porte da manutenção necessária, essa fase pode requerer a definição de um novo processo, onde cada uma das fases precedentes é re-aplicada no contexto de um software existente ao invés de um novo.

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Sequênciais

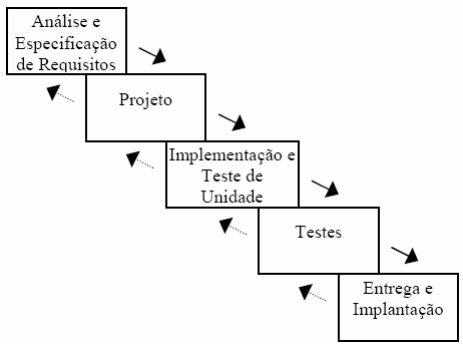
Como o nome indica, os modelos seqüenciais organizam o processo em uma seqüência linear de fases. O principal modelo desta categoria é o modelo em cascata, a partir do qual diversos outros modelos foram propostos, inclusive a maioria dos modelos incrementais e evolutivos.

O Modelo em Cascata

Também chamado de “modelo de ciclo de vida clássico”, o modelo em cascata organiza as atividades do processo de desenvolvimento de forma seqüencial, 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. Uma vez que, na prática, essas fases se sobrepõem de alguma forma, geralmente, permite-se um retorno à fase anterior para a correção de erros encontrados. A entrega do sistema completo ocorre em um único marco, ao final da fase de Entrega e Implantação. O uso de revisões ao fim de cada fase permite o envolvimento do usuário. Além disso, cada fase serve como uma base aprovada e documentada para o passo seguinte, facilitando bastante a gerência de configuração.

UFRRJ
Engenharia de Software

Modelos de Ciclo de Vida de Software: Modelos Sequênciais



O Modelo em Cascata

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Sequênciais

O modelo em cascata é o modelo de ciclo de vida mais antigo e mais amplamente usado. Entretanto, críticas têm levado ao questionamento de sua eficiência. Dentre os problemas algumas vezes encontrados na sua aplicação, destacam-se:

• Projetos reais muitas vezes não seguem o fluxo seqüencial que o modelo propõe.
• Os requisitos devem ser estabelecidos de maneira completa, correta e clara logo no início de um projeto. A aplicação deve, portanto, ser entendida pelo desenvolvedor desde o início do projeto. Entretanto, freqüentemente, é difícil para o usuário colocar todos os requisitos explicitamente. O modelo em cascata requer isto e tem dificuldade de acomodar a incerteza natural que existe no início de muitos projetos.
• O usuário precisa ser paciente. Uma versão operacional do software não estará disponível até o final do projeto.
• A introdução de certos membros da equipe, tais como projetistas e programadores, é freqüentemente adiada desnecessariamente. A natureza linear do ciclo de vida clássico leva a "estados de bloqueio" nos quais alguns membros da equipe do projeto precisam esperar que outros membros da equipe completem tarefas dependentes.

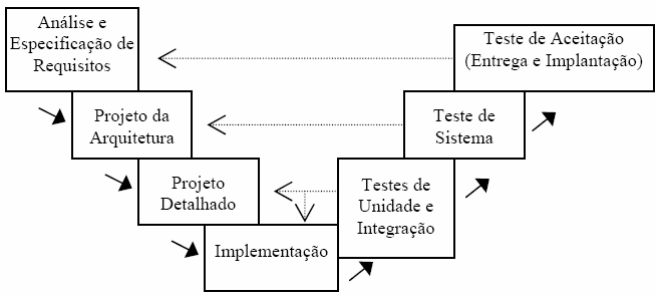
Cada um desses problemas é real. Entretanto, o modelo de ciclo de vida clássico tem um lugar definitivo e importante na engenharia de software. Muitos outros modelos mais complexos são, na realidade, variações do modelo cascata, incorporando laços de feedback. Embora tenha fraquezas, ele é significativamente melhor do que uma abordagem casual para o desenvolvimento de software. De fato, para problemas bastante pequenos e bem definidos, onde os desenvolvedores conhecem bem o domínio do problema e os requisitos podem ser claramente estabelecidos, esse modelo é indicado, uma vez que é fácil de ser gerenciado.

UFRRJ
Engenharia de Software

Modelos de Ciclo de Vida de Software: Modelos Sequênciais

O Modelo em V

O modelo em V é uma variação do modelo em cascata que procura enfatizar a estreita relação entre as atividades de teste (teste de unidade, teste de integração, teste de sistema e teste de aceitação) e as demais fases do processo.



O Modelo em V

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Sequênciais

O modelo em V sugere que os testes de unidade são utilizados basicamente para verificar a implementação e o projeto detalhado. Uma vez que os testes de integração estão focados na integração das unidades que compõem o software, eles também são usados para avaliar o projeto detalhado. Assim, testes de unidade e integração devem garantir que todos os aspectos do projeto do sistema foram implementados corretamente no código.

Quando os testes de integração atingem o nível do sistema como um todo (teste de sistema), o projeto da arquitetura passa a ser o foco. Neste momento, busca-se verificar se o sistema atende aos requisitos definidos na especificação. Finalmente, os testes de aceitação, conduzidos tipicamente pelos usuários e clientes, valida os requisitos, confirmando que os requisitos corretos foram implementados no sistema (teste de validação).

A conexão entre os lados direito e esquerdo do modelo em V implica que, caso sejam encontrados problemas em uma atividade de teste, a correspondente fase do lado esquerdo e suas fases subseqüentes podem ter de ser executadas novamente para corrigir ou melhorar esses problemas.

UFRRJ
Engenharia de Software

Modelos de Ciclo de Vida de Software: Modelos Sequênciais

Os modelos seqüenciais pressupõem que o sistema é entregue completo, após a realização de todas as atividades do desenvolvimento. Entretanto, nos dias de hoje, os clientes não estão mais dispostos a esperar o tempo necessário para tal, sobretudo, quando se trata de grandes sistemas. Dependendo do porte do sistema, podem se passar anos até que o sistema fique pronto, sendo inviável esperar. Assim, outros modelos foram propostos visando a, dentre outros, reduzir o tempo de desenvolvimento.

A entrega por partes, possibilitando ao usuário dispor de algumas funcionalidades do sistema enquanto outras estão sendo ainda desenvolvidas, é um dos principais mecanismos utilizados por esses modelos, como veremos a seguir.

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Incrementais

Há muitas situações em que os requisitos são razoavelmente bem definidos, mas o tamanho do sistema a ser desenvolvido impossibilita a adoção de um modelo seqüencial, sobretudo pela necessidade de disponibilizar rapidamente uma versão para o usuário. Nesses casos, um modelo incremental é indicado.

No desenvolvimento incremental, o sistema é dividido em subsistemas ou módulos, tomando por base a funcionalidade. Os incrementos (ou versões) são definidos, começando com um pequeno subsistema funcional que, a cada ciclo, é acrescido de novas funcionalidades. Além de acrescentar novas funcionalidades, nos novos ciclos, as funcionalidades providas anteriormente podem ser modificadas para melhor satisfazer às necessidades dos clientes / usuários. Vale destacar que a definição das versões (e a correspondente segmentação e atribuição dos requisitos a essas versões) é realizada antes do desenvolvimento da primeira versão.

UFRRJ
Engenharia de Software

Modelos de Ciclo de Vida de Software: Modelos Incrementais

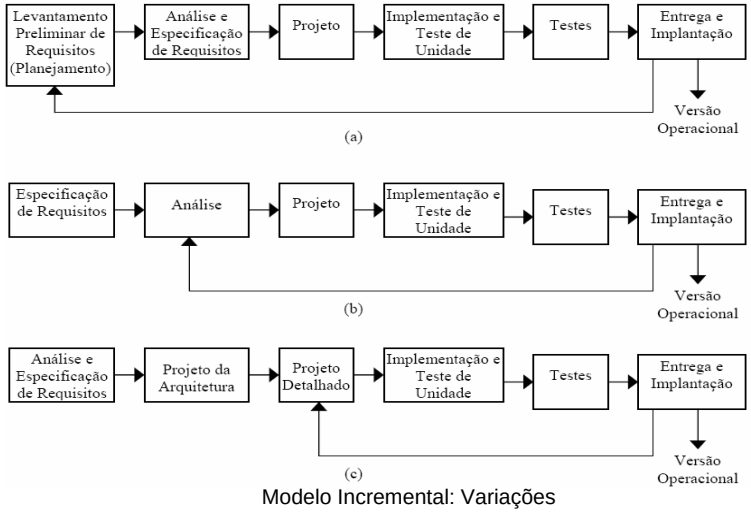
O Modelo Incremental

O modelo incremental pode ser visto como uma filosofia básica que comporta diversas variações. O princípio fundamental é que, a cada ciclo ou iteração, uma versão operacional do sistema será produzida e entregue para uso ou avaliação detalhada do cliente. Para tal, requisitos têm de ser minimamente levantados e há de se constatar que o sistema é modular, de modo que se possa planejar o desenvolvimento em incrementos. O primeiro incremento tipicamente contém funcionalidades centrais, tratando dos requisitos básicos. Outras características são tratadas em ciclos subseqüentes.

Dependendo do tempo estabelecido para a liberação dos incrementos, algumas atividades podem ser feitas para o sistema como um todo ou não.

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Incrementais



Modelo Incremental: Variações

UFRRJ
Engenharia de Software

Modelos de Ciclo de Vida de Software: Modelos Incrementais

Modelo Incremental

Na figura (letra a), inicialmente, durante a fase de planejamento, um levantamento preliminar dos requisitos é realizado. Com esse esboço dos requisitos, planejam-se os incrementos e se desenvolve a primeira versão do sistema. Concluído o primeiro ciclo, todas as atividades são novamente realizadas para o segundo ciclo, inclusive o planejamento, quando a atribuição de requisitos aos incrementos pode ser revista. Este procedimento é repetido sucessivamente, até que se chegue ao produto final.

Outras duas variações bastante utilizadas do modelo incremental são apresentadas na figura (letras b e c). Na figura (letra b), os requisitos são especificados para o sistema como um todo e as iterações ocorrem a partir da fase de análise. Na figura (letra c), o sistema tem seus requisitos especificados e analisados, a arquitetura do sistema é definida e apenas o projeto detalhado, a implementação e os testes são realizados em vários ciclos. Uma vez que nessas duas variações os requisitos são especificados para o sistema como um todo, sua adoção requer que os requisitos sejam estáveis e bem-definidos.

O modelo incremental é particularmente útil quando não há pessoal suficiente para realizar o desenvolvimento dentro dos prazos estabelecidos ou para lidar com riscos técnicos, tal como a adoção de uma nova tecnologia.

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Incrementais

Modelo Incremental

Dentre as vantagens do modelo incremental, podem ser citadas:

• Menor custo e menos tempo são necessários para se entregar a primeira versão;
• Os riscos associados ao desenvolvimento de um incremento são menores, devido ao seu tamanho reduzido;
• O número de mudanças nos requisitos pode diminuir devido ao curto tempo de desenvolvimento de um incremento.

Como desvantagens, podemos citar:

• Se os requisitos não são tão estáveis ou completos quanto se esperava, alguns incrementos podem ter de ser bastante alterados;
• A gerência do projeto é mais complexa, sobretudo quando a divisão em subsistemas inicialmente feita não se mostrar boa.

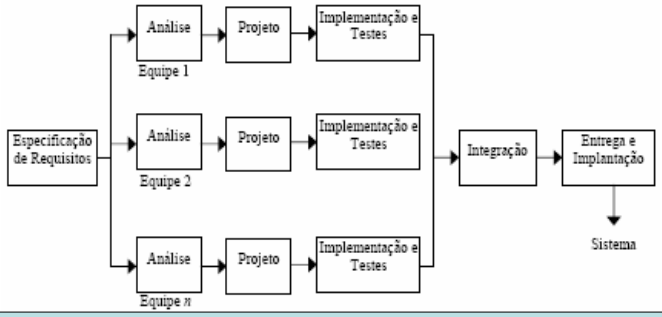
UFRRJ
Engenharia de Software

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Incrementais

Modelo RAD (Rapid Application Development)

O RAD ou modelo de desenvolvimento rápido de aplicações, é um tipo de modelo incremental que prima por um ciclo de desenvolvimento curto (tipicamente de até 90 dias). Assim, como no modelo incremental, o sistema é subdividido em subsistemas e incrementos são realizados. A diferença marcante é que os incrementos são desenvolvidos em paralelo por equipes distintas e apenas uma única entrega é feita.



Modelos de Ciclo de Vida de Software: Modelos Incrementais

Modelo RAD

Assim, como o modelo incremental, o modelo RAD permite variações. Em todos os casos, no entanto, os requisitos têm de ser bem-definidos, o escopo do projeto tem de ser restrito e o sistema modular. Se o projeto for grande, por exemplo, o número de equipes crescerá demais e a atividade de integração tornar-se-á por demais complexa.

Obviamente, para adotar esse modelo, uma organização tem de ter recursos humanos suficientes para acomodar as várias equipes.

UFRRJ
Engenharia de Software

Modelos de Ciclo de Vida de Software: Modelos Evolutivos ou Evolucionários

Sistemas de software, como quaisquer sistemas complexos, evoluem ao longo do tempo. Seus requisitos, muitas vezes, são difíceis de serem estabelecidos ou mudam com freqüência ao longo do desenvolvimento. Assim, é importante ter como opção modelos de ciclo de vida que lidem com incertezas e acomodem melhor as contínuas mudanças. Alguns modelos incrementais, dado que preconizam um desenvolvimento iterativo, podem ser aplicados a esses casos, mas a grande maioria deles toma por pressuposto que os requisitos são bem-definidos. Modelos evolucionários ou evolutivos buscam preencher essa lacuna.

Enquanto modelos incrementais têm por base a entrega de versões operacionais desde o primeiro ciclo, os modelos evolutivos não têm essa preocupação. Muito pelo contrário: na maioria das vezes, os primeiros ciclos produzem protótipos ou até mesmo apenas modelos. À medida que o desenvolvimento avança e os requisitos vão ficando mais claros e estáveis, protótipos vão dando lugar a versões operacionais, até que o sistema completo seja construído. Assim, quando o problema não é bem definido e ele não pode ser totalmente especificado no início do desenvolvimento, deve-se optar por um modelo evolutivo. A avaliação ou o uso do protótipo / sistema pode aumentar o conhecimento sobre o produto e melhorar o entendimento que se tem acerca dos requisitos. Entretanto, é necessária uma forte gerência do projeto e de configuração.

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Evolutivos ou Evolucionários

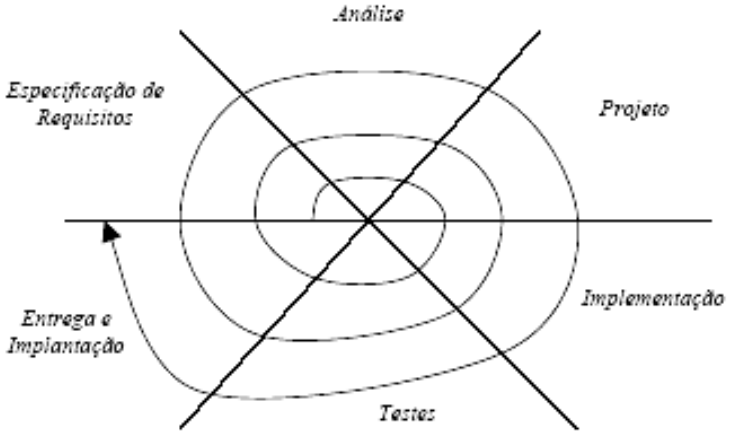
Modelo Espiral

O modelo espiral é um dos modelos evolutivos mais difundidos. nem sempre todas as atividades são realizadas. Por exemplo, o produto resultante do primeiro ciclo pode ser uma especificação do produto ou um estudo de viabilidade. As passadas subseqüentes ao longo da espiral podem ser usadas para desenvolver protótipos, chegando progressivamente a versões operacionais do software, até se obter o produto completo. Até mesmo ciclos de manutenção podem ser acomodados nesta filosofia, fazendo com que o modelo espiral contemple todo o ciclo de vida do software.

É importante ressaltar que, a cada ciclo, o planejamento deve ser revisto com base no feedback do cliente, ajustando, inclusive, o número de iterações planejadas. De fato, este é o maior problema do ciclo de vida espiral, ou de maneira geral, dos modelos evolucionários: a gerência de projetos. Pode ser difícil convencer clientes, especialmente em situações envolvendo contrato, que a abordagem evolutiva é gerenciável.

UFRRJ
Engenharia de Software

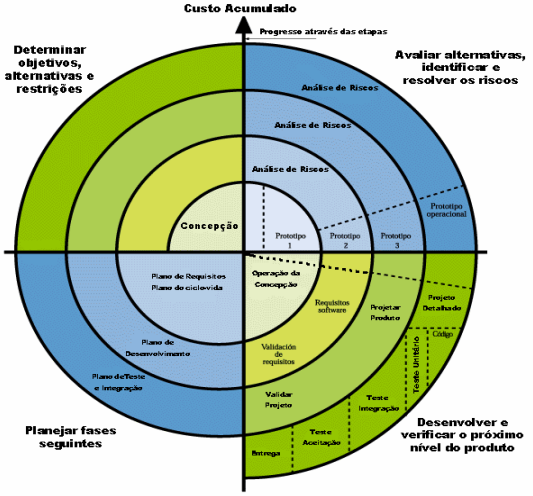
Modelos de Ciclo de Vida de Software: Modelos Evolutivos ou Evolucionários



O Modelo Espiral

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Evolutivos ou Evolucionários



O Modelo Espiral

UFRRJ
Engenharia de Software

Modelos de Ciclo de Vida de Software: Modelos Evolutivos ou Evolucionários

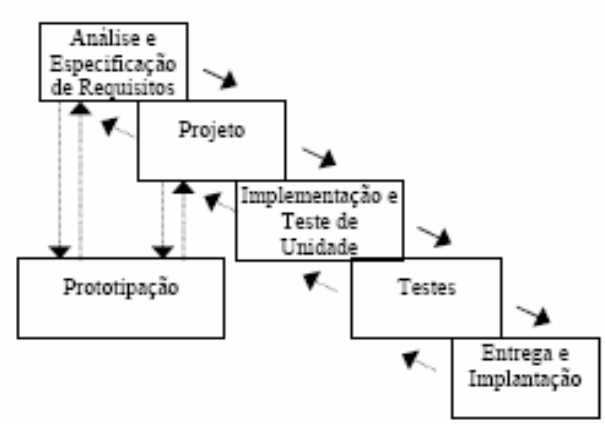
Prototipação

Muitas vezes, clientes têm em mente um conjunto geral de objetivos para um sistema de software, mas não são capazes de identificar claramente as funcionalidades ou informações (requisitos) que o sistema terá de prover ou tratar. Modelos podem ser úteis para ajudar a levantar e validar requisitos, mas pode ocorrer dos clientes e usuários só terem uma verdadeira dimensão do que está sendo construído se forem colocados diante do sistema.

Nestes casos, o uso da prototipação é fundamental. A prototipação é uma técnica para ajudar engenheiros de software e clientes a entender o que está sendo construído quando os requisitos não estão claros. Ainda que tenha sido citada anteriormente no contexto do modelo espiral, ela pode ser aplicada no contexto de qualquer modelo de processo.

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal

Modelos de Ciclo de Vida de Software: Modelos Evolutivos ou Evolucionários



O Modelo em Cascata com Prototipação

Automatização de Processos

Com o aumento da complexidade dos processos de software, passou a ser imprescindível o uso de ferramentas e ambientes de apoio à realização de suas atividades, visando, sobretudo, a atingir níveis mais altos de qualidade e produtividade. Ferramentas CASE (*Computer Aided Software Engineering*) passaram, então, a ser utilizadas para apoiar a realização de atividades específicas, tais como planejamento e análise e especificação de requisitos.

Apesar dos benefícios do uso de ferramentas CASE individuais, atualmente, o número e a variedade de ferramentas têm crescido a tal ponto que levou os engenheiros de software a pensarem não apenas em automatizar os seus processos, mas sim em trabalhar com diversas ferramentas que interajam entre si e forneçam suporte a todo ciclo de vida do desenvolvimento, dando origem ao Ambientes de Desenvolvimento de Software (ADSs). ADSs buscam combinar técnicas, métodos e ferramentas para apoiar o engenheiro de software na construção de produtos de software, abrangendo todas as atividades inerentes ao processo: gerência, desenvolvimento e controle da qualidade.

Material de apoio:

Bibliografia Básica

PRESSMAN, R. Engenharia de software. Rio de Janeiro: MacGraw-Hill, 2006.
SOMMERVILLE, I. Engenharia de software. 8. ed. São Paulo: Addison Wesley, 2007.

Bibliografia Complementar

PFLEEGER, S.L., et al, “Software Engineering”, Prentice Hall, 2005, 3rd edition.
IEEE Computer Society Real-World Software problems: A Self-Study Guide for Today’s Software Professional, Wiley-IEEE Computer Society Press, 2006.
Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2004. Disponível em http://swebok.org.

UFRRJ
Engenharia de Software

Material de apoio:

Referências Bibliográficas

FALBO, Ricardo de Almeida. Engenharia de Software – Notas de Aula. Universidade Federal do Espírito Santo (UFES). 2005.

SWEEBOK. *Software Engineering Body of Knowledge. 2004.* http://www.computer.org/portal/web/swebok

Curso de Sistemas de Informações
Prof. André Luiz de Castro Leal