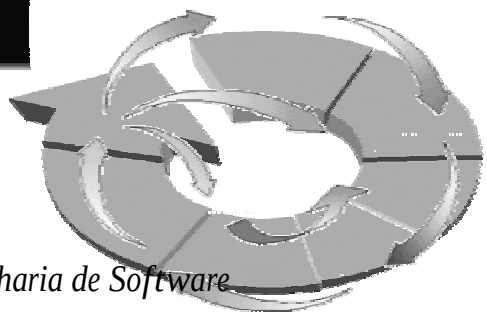
PUC CAMPINAS



Engenharia de Software

Tema da Aula

Modelos de Ciclos de Vida de Software

2 – Outros Modelos

Prof. Cristiano R R Portella
portella@widesoft.com.br

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software

As duas primeiras décadas da ES foram dominadas pelo paradigma do desenvolvimento linear (engenharia progressiva do produto de software).

Em seguida aparece o modelo evolucionário (ou evolutivo) que se baseia num desenvolvimento de um produto inicial que vai sendo submetido a avaliações do usuário e sendo simultaneamente refinado através de sucessivas versões, até que alcance a funcionalidade desejada.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software

Assim, atividades de desenvolvimento e de validação são realizadas paralelamente, com um rápido feedback entre elas.

O modelo evolucionário requer iteratividade (repetir fases do processo) e interatividade (trabalho conjunto entre usuário e desenvolvedor).

Inicialmente o modelo evolucionário previa ciclos lineares (paradigma do desenvolvimento linear no modelo incremental); depois o modelo ganhou a forma de espiral.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Mudança de Paradigma

Paradigma Desenvolvi/o linear
- Modelo cascata (desenv. Linear puro).
- Modelo incremental (linear que cada etapa, porém com iteratividade e desenvolvimento/teste durante todo o processo)

Paradigma desenv. evolutivo
- Modelo Evolutivo (ou exploratório)
- Modelo por Prototipação

Ciclos de Vida de Software

Modelo Incremental

Ciclos de Vida de Software
Modelo Incremental

Criado pela European Space Agency em 1991, é uma adaptação da forma de pensar o desenvolvimento de software como um “desenvolvimento linear”, pois trata-se de um arranjo de vários pequenos ciclos em cascata.

A diferença é que cada versão do produto de software tem uma nova funcionalidade (incremento), definida no início desse ciclo. Depois de desenvolvida cada versão, será entregue ao usuário para testes. Cada versão implementa uma nova funcionalidade (incremento).

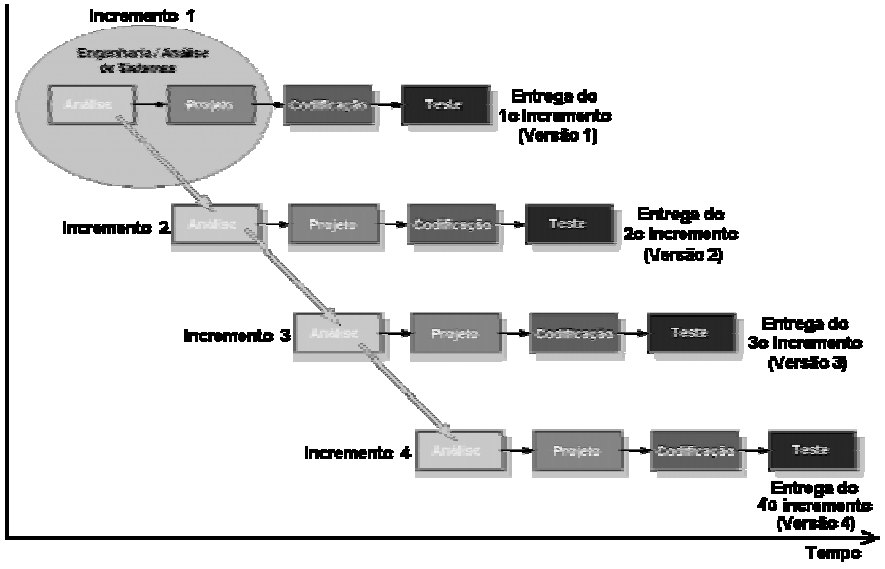
PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Incremental

1. Identificar (grosso modo) requisitos e conceitos do novo sistema.
2. Desenvolver um projeto do produto (será o projeto básico).
3. Construir o produto. A primeira versão é básica e será o núcleo do produto (funcionalidade básica).
4. Essa versão é entregue para o usuário testar.
5. A partir dessa versão, o usuário vai detectar a próxima funcionalidade necessária no sistema. O ciclo volta para a fase '1', porém o projeto agora irá se restringir apenas à nova funcionalidade.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Incremental



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Incremental

Problemas:

1. Parte da falsa premissa de que os requisitos iniciais serão mantidos (primeira versão – núcleo do produto). Pela própria experiência do usuário eles irão mudar (além de mudanças na organização, no processo, no produto, na tecnologia etc).
2. Em razão das mudanças de requisitos (item 1), o escopo total do projeto só é conhecido aos poucos, portanto as estimativas de prazo/custo estarão erradas.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Incremental

Problemas:

3. Após o primeiro ciclo, o usuário deverá “descobrir” sozinho os requisitos do próximo incremento. Na prática ele precisa de suporte do desenvolvedor para formular esses requisitos.

Ciclos de Vida de Software

Modelo Evolutivo

PUC CAMPINAS
Engenharia de Software

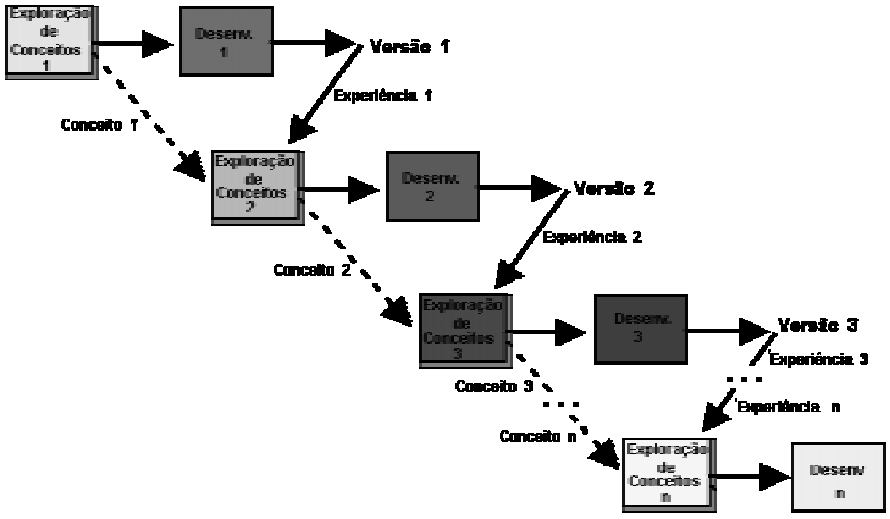
Ciclos de Vida de Software
Modelo Evolutivo

Modelo Evolutivo ou Desenvolvimento Exploratório, tem como objetivo promover um desenvolvimento conjunto (desenvolvedor trabalhando junto com o usuário) a fim de descobrir os requisitos de maneira incremental.

Diferente do desenvolvimento incremental, esse modelo volta sempre à fase de definição de requisitos, num movimento exploratório de conceitos do produto e necessidades do usuário.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Evolutivo (Exploratório)



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software

Modelo de Prototipação

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Prototipação

Modelo de Prototipação, também chamado de Protótipo Descartável, tem como objetivo obter uma melhor definição dos requisitos do sistema.

Todavia, diferente do modelo Incremental e Evolutivo, que fazem essa exploração enquanto constroem o produto final, a prototipação exploratória usa os protótipos apenas para descobrir a funcionalidade desejada e os riscos implícitos. Depois ele é descartado e inicia-se a construção do produto final.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Prototipação

Como o protótipo não será usado como produto final, pode-se (deve-se) omitir aspectos de estética, performance, segurança etc assim como deve-se trabalha com ferramentas adequadas à rápida geração de código.

O protótipo não precisa conter toda a funcionalidade do produto final; apenas os aspectos sobre os quais havia alto nível de incerteza, como por exemplo detalhes das entradas ou saídas do sistema, a exatidão de um algoritmo ou a correta aplicação de uma nova tecnologia.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Prototipação

Vendo o que o protótipo apresenta, o usuário descobre novas necessidades (vendo o que ele tem, descobre o que falta).

O desenvolvimento começa com uma versão preliminar, após rápida interação desenvolvedor-usuário.

Após o desenvolvimento é dado aos usuários a oportunidade de "brincar" com o protótipo. Baseado nessa experiência o usuário opina sobre o que está certo, o que está errado, o que está faltando e o que está sobrando.

PUC CAMPINAS
Engenharia de Software

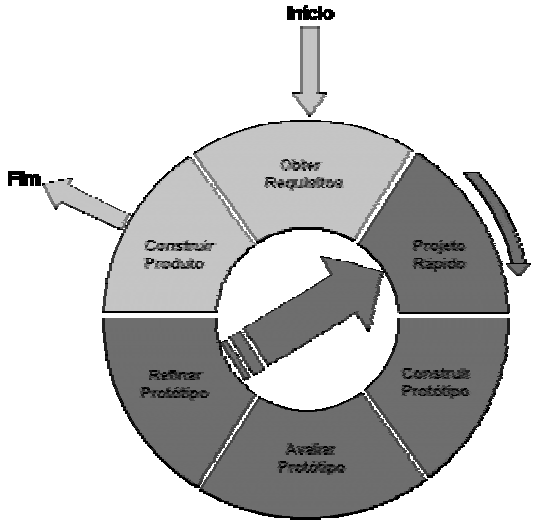
Ciclos de Vida de Software
Prototipação

Após essa interação, o protótipo é modificado e novamente entregue ao usuário.

Esse ciclo se repete até que seja possível definir os requisitos do produto a ser desenvolvido.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Prototipação



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Prototipação

Problemas:

A prototipação (e em geral todos os modelos evolucionários) é mais efetiva quanto a desenvolver produtos que atendem melhor os requisitos do usuário, em relação ao modelo em cascata.

Todavia esses modelos criam um processo não gerenciável do ponto de vista do prazo (quando a prototipação acaba ?, qual a viabilidade do projeto todo ?)

Ciclos de Vida de Software
Prototipação

Problemas:

Também existe um custo extra, pelo período de prototipação, que só se justifica se o nível de incerteza quanto aos requisitos é muito alto.

O usuário geralmente quer usar o protótipo como produto (até que o produto seja desenvolvido). Se aceito, aparecerão problemas como: fazer "pequenas adaptações", questões de segurança, performance etc.

Ciclos de Vida de Software

Modelo Espiral

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

O modelo espiral (Boehm, 86) começa uma nova tendência na modelagem, através de ciclos repetitivos em espiral.

A espiral inicia de um centro onde o movimento ainda é pouco acentuado (o produto está em sua versão inicial; apenas um cerne do que será o protótipo final). Com o passar dos ciclos, o movimento fica maior em cada quadrante e isso simboliza bem o fato de que, à medida que a espiral cresce, o trabalho num determinado quadrante aumenta.

PUC CAMPINAS
Engenharia de Software

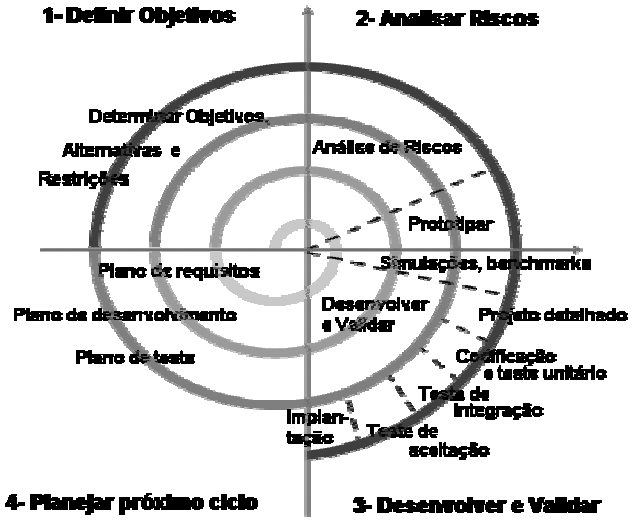
Ciclos de Vida de Software
Modelo Espiral

No modelo espiral tradicional, existem quatro quadrantes, onde se executa prioritariamente as seguintes tarefas:

1o quadrante	Planejar
2o quadrante	Analisar riscos
3o quadrante	Construir
4o quadrante	Avaliar

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

- O modelo mantém as características positivas de documento associado a fases do ciclo (cascata) com fases sobrepostas (incremental) e uso de prototipação (aproveitando o protótipo para produto final).
- Parte da premissa de que a forma de desenvolvimento de software não pode ser completamente determinada de antemão (centro da espiral).
- Assim, através de fases sucessivas e algumas atividades-chave (análise de risco, prototipar, simular, validar etc) chegamos a um projeto detalhado de software e a construção simultânea do produto final.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

Boehm caracteriza seu modelo como um “gerador de modelo de processo”.

Cada ciclo do modelo em espiral possui quatro momentos principais, com as seguintes atividades:

1o) Elaborar os objetivos do projeto, restrições e alternativas para as entidades de software.

2o) Avaliar as alternativas com relação aos objetivos e restrições e identificar as principais fontes de risco. Se necessário, criar protótipo para simular o software real.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

3o) Elaborar a definição das entidades de software através de um projeto.

No último ciclo, quando o projeto detalhado estiver pronto, ele será construído, testado e implantado, numa série de atividades iguais ao modelo em cascata.

4o) Planejar o próximo ciclo da espiral (objetivo, atividades, ferramentas etc).

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

Análise de Riscos:

O modelo espiral trouxe uma ênfase à Análise de Riscos no desenvolvimento de software (gerência segura).

Um risco é um problema potencial em um sistema, ou a probabilidade de ocorrer um evento perigoso ou indesejado em determinado momento ou circunstância.

Através da Análise de Riscos, podemos escolher caminhos com as melhores chances de sucesso, dentro de prazos razoáveis.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

Análise de Riscos:

1o) Probabilidade da ocorrência de um evento.

2o) Conseqüente perda estimada (exposição ao risco)

1o) Probabilidade “p” de que um determinado evento ocorra (1=< p >= 0)

Se existir probabilidade do evento ocorrer (> 0), será um evento aleatório e deve ser estimado.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

Probabilidade de um evento ocorrer pr(E):

$$pr(E) = \frac{no\_de\_eventos\_favoráveis}{no\_de\_eventos\_possíveis}$$

Por exemplo, o robô Sojourner (missão Nasa em Marte) estava exposto ao risco do congelamento de um sensor de impacto (temperatura ambiente de –58ºC) e caso isso ocorresse, o impacto poderia danificar esse sensor. Nesse caso, equipes de Terra teriam que modificar o código para que o programa executasse sem esse sensor.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

$$pr(E) = \frac{\text{No de falhas em instruções de comando, em razão da perda do sensor}}{\text{No total de instruções de comando que necessitam de entrada do sensor}}$$

pr(E) = 6.552 / 18.820 = **0,35**

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

Assumindo que uma falha no módulo de comando resulte uma perda de $ 45.000 (custo de homens/mês necessários para modificar o módulo de comando de forma que ele volte a funcionar sem o sensor quebrado ..

Exposição ao Risco

Er = (0,35 * 45.000)/10 = 1.575

O coeficiente 1.575 indica risco excessivo ou pode ser "bancado" ?

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

Se for risco excessivo, o que precisará ser feito no projeto para eliminar esse risco ?

- Reforçar o sensor
- Colocar sensor reserva
- Proteger o sensor
- Mudar projeto do sensor
- Criar módulo de software flexibilizado para executar sem falhas, com e sem o sinal do sensor.
- etc

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral

Assim, o modelo em espiral preconiza o desenvolvimento de software através da avaliação de riscos obtidos através de sucessivas prototipagens (e/ou simulações e benchmarking).

Desvantagem:

Aumento de prazo ⇨ custos no projeto.

Justificativa para isso:

- Complexidade do projeto
- Incerteza nos requisitos

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral Atualizado

O modelo em espiral tem uma versão atualizada, com as seguintes alterações:

1. Passou de 4 para 6 quadrantes:
   - Solicitação do cliente
   - Planejamento
   - Análise de risco
   - Modelagem
   - Construção, teste, instalação e suporte para o cliente testar
   - Avaliação do cliente

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral Atualizado

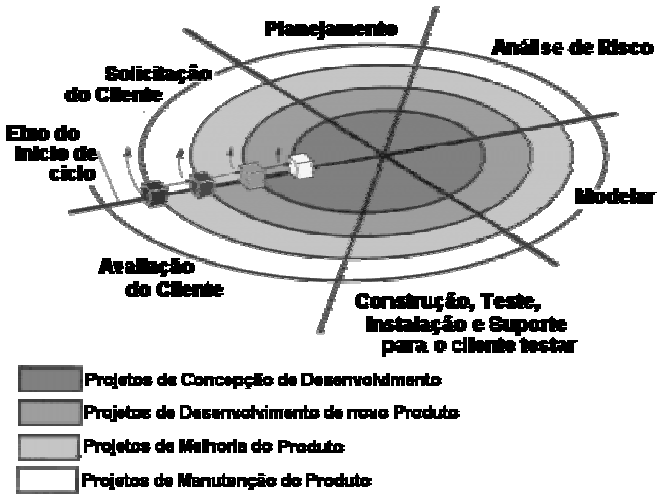
Outra alteração do modelo espiral atualizado em relação ao tradicional é a divisão do curso do desenvolvimento (curva) em 4 tempos (a espiral está dividia em 4 tempos, que definem quatro objetivos diferentes para o trabalho), que são:

- Projeto de concepção de desenvolvimento
- Projeto de desenvolvimento do novo produto
- Projeto de melhoria do produto
- Projeto de manutenção do produto

Como cada fase tem um objetivo diferente, gera um produto diferente no seu término (vide figura a seguir).

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral Atualizado



Modelo Espiral para Desenvolvimento Baseado em Componentes

O desenvolvimento baseado em componentes, é uma conseqüência da adoção da Orientação a Objeto.

- ✓ Sistemas baseados em componentes "COTS" (Commercial Off-The-Shelf – de prateleira) reusáveis em larga escala ou CBS (COTS Based Systems).

  (vide desenvolvimento baseado em COTS e Métricas COTS)

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral e Componentes

Trata-se de uma variação do modelo de desenvolvimento em espiral atualizado, onde as fases 4 e 5 (modelagem e construção-teste-instalação-suporte para teste) tem duas possíveis formas de desenvolvimento:

1. Identificar componentes:
   - Identificar componentes candidatos (produtos de software que tem a funcionalidade requerida)
   - Procurar o componente (bibliotecas, outros projetos, mercado, internet etc)

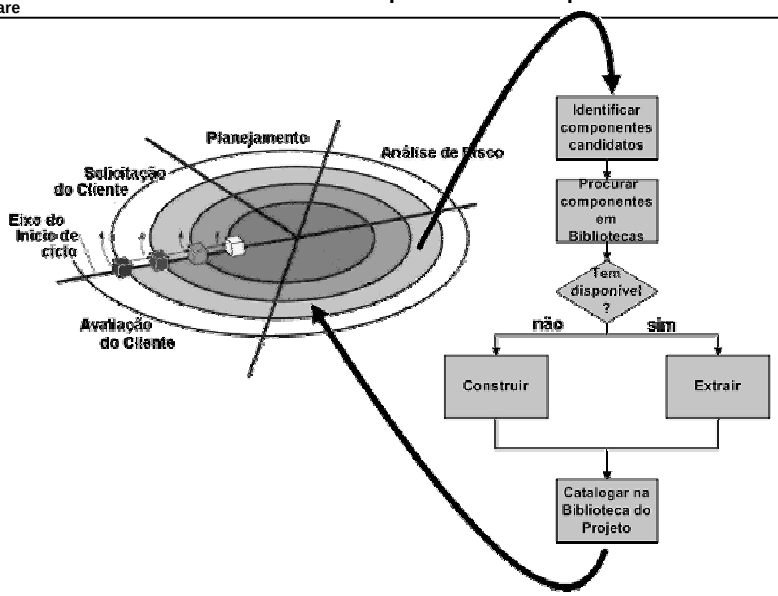
PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral e Componentes

2. Obter e disponibilizar componentes:
   - Caso não existam componentes prontos, a fase será desenvolvida como no modelo espiral adaptado (neste ciclo da espiral), com a construção do componente necessário.
   - Se existirem componentes, eles serão recuperados e catalogados na biblioteca do projeto. Em seguida serão aplicados na construção do produto, encerrando essa fase.
   - O ciclo continua na fase de avaliação do cliente).

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral e Componentes



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software

Modelo Espiral WIN-WIN

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral WIN-WIN

O diálogo usuário-desenvolvedor tem características de negociação, onde o usuário quer a máxima funcionalidade e o desenvolvedor deve lembra-lo sobre prazo/custo/performance decorrentes do tamanho do projeto (existem muitas outras situações de negociação entre interesses do usuário e restrições do desenvolvedor).

A melhor negociação (para a organização) é aquela dirigida para resultados Ganha-Ganha, onde o cliente ganha por obter um produto que satisfaça a maioria de suas necessidades e o desenvolvedor possa trabalhar com orçamentos e prazos realistas.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral WIN-WIN

O modelo Espiral Win-Win (Boehm, 98) define um conjunto de atividades de negociação que se iniciam a cada ciclo da espiral. Essas atividades são assim definidas:

1. Identificar os interessados (*) na próxima rodada de negociações.

   (*) Interessados: pessoas da organização que têm interesse no sucesso do sistema, ou serão criticadas se ele falhar.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral WIN-WIN

2. Determinar as condições favoráveis (de ganho) para esses interessados.

3a. Negociar as solicitações e restrições e, ao final, reconciliar as opções dentro de uma situação ganha-ganha para todos os envolvidos (todos os interessados e toda a equipe de projeto).

3b. Estabelecer os objetivos negociados, as restrições e as alternativas como metas para o próximo ciclo de desenvolvimento do produto.

4. Avaliar riscos e alternativas.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral WIN-WIN

5. Definir qual será o próximo incremento do sistema (próximo nível do produto e do processo).
6. Validar as definições ou apurar incorreções.
7. Revisar as eventuais incorreções e obter comprometimento.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral WIN-WIN

O modelo Win-Win ainda acrescenta 3 milestones de processo chamados "pontos-ancora", cuja finalidade é ajudar a encerrar um ciclo na espiral. Esses 3 pontos representam visões diferentes do progresso como se fosse um corte transversal na espiral.

1. Objetivo do Ciclo de Vida (LCO)- Define um conjunto de objetivos para cada atividade principal do ciclo de vida.

   Por exemplo, cjto. de objetivos para serem alcançados com uma definição de alto nível dos requisitos do produto de software.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral WIN-WIN

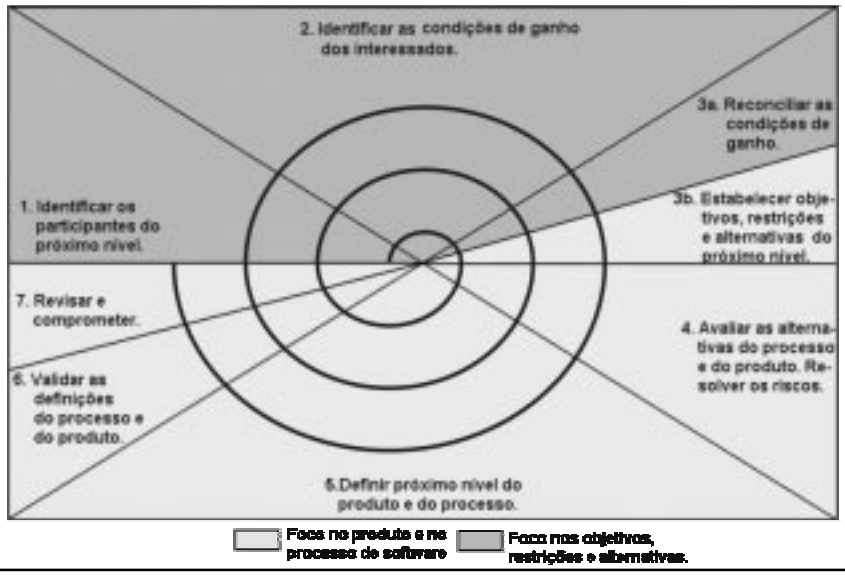
2. Arquitetura do Ciclo de Vida (LCA) – Estabelece objetivos que precisam ser realizados no projeto de arquitetura.

   Por exemplo, estabelecer como objetivo que no projeto de arquitetura foi analisada a possibilidade da aplicação de componentes reutilizáveis e considerados seus impactos sobre a arquitetura do produto.

3. Conjunto de Objetivos relacionados com a preparação do software para a instalação, distribuição e suporte.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo Espiral WIN-WIN



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software

Modelo RAD

Rapid Application Development

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RAD

Segue o padrão incremental, porém enfatiza um desenvolvimento rápido (geralmente 60 a 90 dias).

Trata-se de uma adaptação do modelo linear-seqüencial (cascata) no qual a velocidade é obtida pela aplicação de componentes reutilizáveis (COTS) e múltiplas equipes de desenvolvimento.

Adequado para casos em que os requisitos e escopo estão bem definidos.

Fases:

Inicialmente o projeto é dividido em sub-projetos. Todos os sub-projetos que podem iniciar juntos são designados para equipes diferentes.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RAD

Cada sub-projeto executará as seguintes fases:

1. Modelagem de Negócio:

   Modelar o fluxo da informação fluindo através da organização (negócio), a fim de responder as seguintes questões:

   - Que informações dirigem o processo de negócio ?
   - Que informações são geradas ?
   - Quem (área/processo) gera as informações ?
   - Para onde a informação vai ?

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RAD

2. Modelagem de Dados:

   O fluxo definido no item 1 é agora transformado num conjunto de objetos de dados.

3. Modelagem do Processo:

   Analisa as transformações nos objetos de dados, através de adição, modificação, exclusão ou recuperação de dados, a fim de implementar uma funcionalidade de negócio.

4. Geração da Aplicação:

   O modelo RAD pressupõe o uso de técnicas L4G (por exemplo a geração automática de código)

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RAD

5. Testar e concluir o desenvolvimento.

   Em seguida a equipe é dissolvida para iniciar outro sub-projeto ou mesmo reintegrar-se em equipes que estão com o trabalho em desenvolvimento.

L4G – Técnicas de linguagens de Quarta Geração:

Linguagens não-procedurais (definir “o que” e não “como”).

Linguagens de “query” para manipulação de bases, geração de relatórios e consultas, telas, manipulação de dados e geração de código.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RAD

Desvantagens:

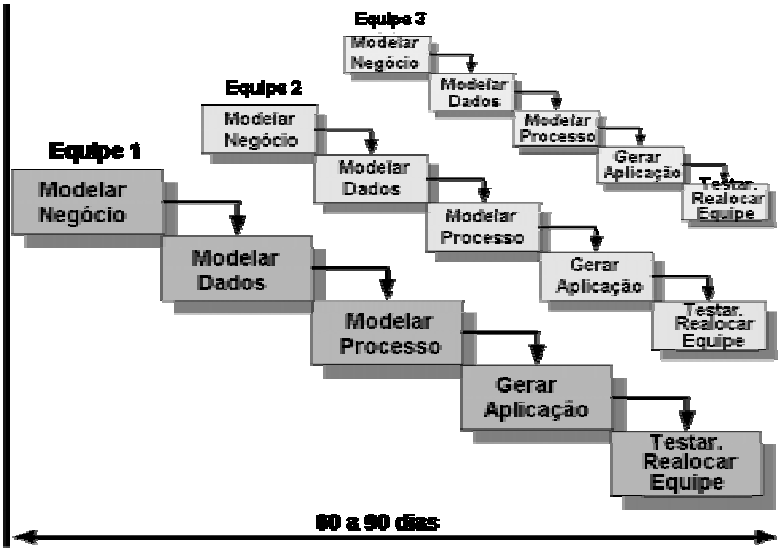
Para grandes projetos, precisa de um número maior de pessoas.

O produto tem que ser apto a ser modularizado (sub-projetos) e permitir a aplicação de componentes reutilizáveis.

Não é adequado a projetos de alto risco ou com grande interação com softwares já existentes.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RAD



Modelo RUP
RATIONAL UNIFIED PROCESSING

Desenvolvimento de software a partir de um framework genérico, que pode ser especializado para diferentes aplicações, organizações, níveis de competência e tamanho de projetos.

Características:

- ✓ Baseado em componentes reutilizáveis
- ✓ Usa UML para modelar o produto
- ✓ Dirigido a casos de uso (Use Case)
- ✓ Centrado na arquitetura
- ✓ Iterativo e Incremental

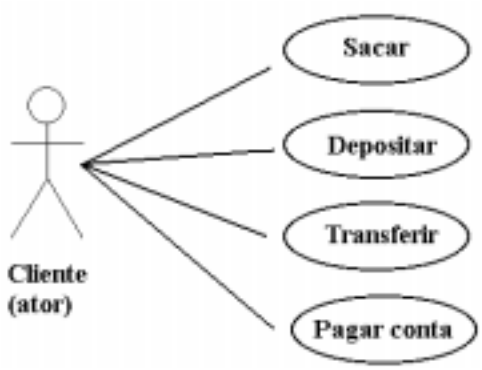
PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RUP

- Recomenda o uso de Use Cases para capturar a funcionalidade do produto. Um modelo de Casos de Uso é formado pelo conjunto de diagramas de Caso de Uso. Vantagem: está dirigido a cada usuário (ator).
- A arquitetura de software deve contemplar seus aspectos estático e dinâmico.

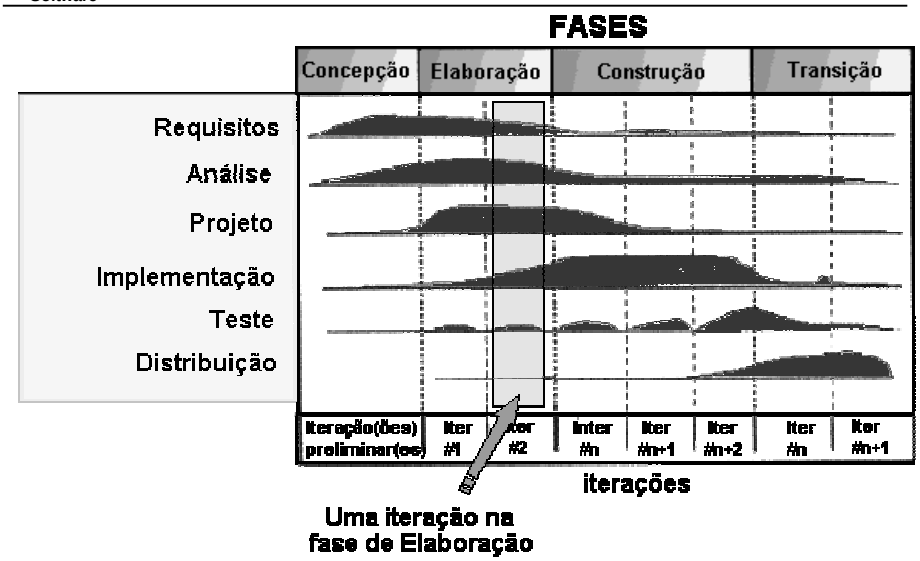
PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RUP



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo RUP



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software

Outros Modelos de Ciclo de Vida

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo de Desenvolvimento Concorrente

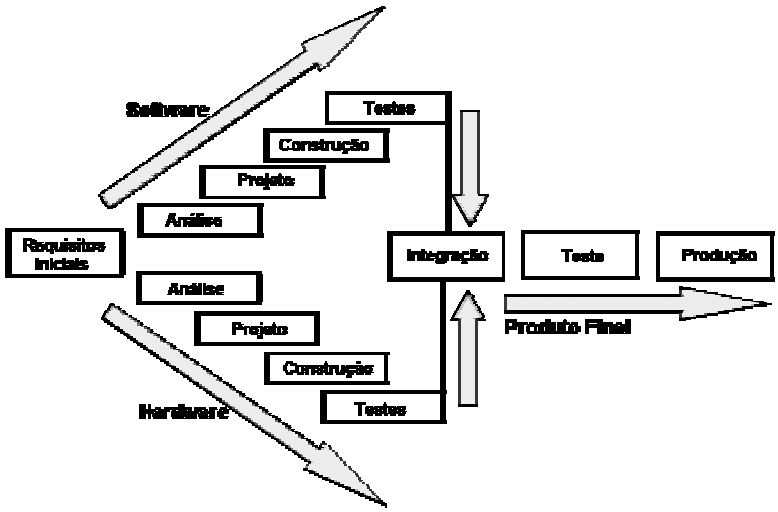
(Davis e Sitaran-94)

Em todas as fases do projeto existem atividades que podem ser desenvolvidas em paralelo (de forma concorrente).

Esse modelo é adequado para projetos com software e hardware ou ainda cliente/servidor.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo para Embedded Software (DoD)



PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo “Sincronizar e Estabilizar”

Modelo “Sincronizar e Estabilizar” -Microsoft

Modelo incremental com prototipação rápida e testes de regressão automática (Cusimano e Selby-97).

À medida em que o projeto progride, os incrementos de software são periodicamente estabilizados (tirar defeitos graves.

Microsoft: cada desenvolvedor tem um “Build” diário (empreitada), indicando o incremento de software a ser desenvolvido.

PUC CAMPINAS
Engenharia de Software

Ciclos de Vida de Software
Modelo “Sincronizar e Estabilizar”

Fases:

1. Planejar:

   Visão, especificação e cronograma do projeto (incremento).

2. Projetar e Construir:

   Com prototipação rápida. Depois desenvolver o produto definitivo.

3. Estabilizar:

   Testes internos (na empresa) e externo (sites Beta).