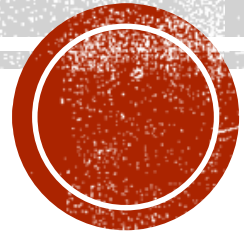
MODELOS DE PROCESSO PRESCRITIVO



MODELOS DE PROCESSO

- Processos são construídos de acordo com modelos ou estilos, que quando aplicados ao desenvolvimento de software são conhecidos como ciclos de vida.
- Pode-se afirmar que existem duas grandes famílias de modelos: os prescritivos e os ágeis.

NÃO EXISTE UM MODELO MELHOR QUE TODOS OS OUTROS

- Projetos diferentes tem diferentes necessidades.
- O engenheiro de software deve escolher aquele mais adequado à sua equipe e ao projeto que vai desenvolver.
- Se ele escolher bem, terá um processo eficiente, baseado em padrões e lições aprendidas e com possibilidade de capitalizar experiências.
- O controle será eficiente e os riscos, erros e retrabalho serão minimizados.
- Mas se escolher mal poderá estar gerando trabalho repetitivo e frustrante para a equipe, o que aliás, pode acontecer também quando não se escolhe ciclo de vida algum.

SEGUNDO ELLIS (2010), PARA ESCOLHER UM MODELO DE CICLO DE VIDA O ENGENHEIRO DE SOFTWARE DEVE RESPONDER ÀS SEGUINTES PERGUNTAS:

- Quão bem os analistas e o cliente podem conhecer os requisitos do sistema?
- O entendimento sobre o sistema poderá a mudar a medida que o desenvolvimento
  avançar?
- Quão bem é compreendida a arquitetura do sistema?
- É provável que sejam feitas grandes mudanças de rumo ao longo do projeto?

Segundo Ellis (2010), para escolher um modelo de ciclo de vida o engenheiro de software deve responder às seguintes perguntas:

- Qual o grau de confiabilidade necessário em relação ao cronograma?
- Quanto planejamento é efetivamente necessário?
- Qual o grau de risco que este projeto apresenta?
- Existe alguma restrição de cronograma?
- Será necessário entregar partes do sistema funcionando antes de terminar o projeto todo?
- Qual o grau de treinamento e adaptação necessários para a equipe poder utilizar o ciclo de vida que parece mais adequado ao projeto?
- Em função das respostas um dos ciclos seguintes poderá ser escolhido.

CODIFICAR E CONSERTAR (CODE AND FIX)

- Passos:
  1. Construa junto com o cliente um entendimento preliminar sobre o sistema que deve ser desenvolvido.
  2. Implemente uma primeira versão deste sistema.
  3. Interaja com o cliente de forma a corrigir a versão preliminar até que esta satisfaça ao cliente.
  4. Faça testes e corrija os inevitáveis erros.
  5. Entregue o produto.

SEM PREVISIBILIDADE

- Pode-se dizer que é uma forma bastante ingênua de modelo de processo.
- Pode-se dizer também que nem sequer parece ser um processo, pois não há previsibilidade em relação às atividades e resultados obtidos.
- O modelo é usado porque é simples, não porque funciona bem.
- Muitos projetos reais, mesmo dirigidos por outros modelos de ciclo de vida, por vezes caem na prática de codificar e testar em função de pressão de cronograma.

ONDE É USADO

- Empresas que não usam modelo de processo algum possivelmente usam Codificar e Consertar por default.
- O modelo Codificar e Consertar é, possivelmente, bastante usado em empresas de pequeno porte, mas deve ser entendido mais como uma maneira de pressionar programadores (code rush) do que como um processo organizado (McConnell, 1996).
- Se o sistema não satisfaz o cliente, cobra-se do programador uma versão funcional adequada no menor tempo possível.

VANTAGENS

- Não se perde tempo com documentação, planejamento ou projeto: vai-se direto à codificação.
- O progresso é facilmente visível à medida que o programa vai ficando pronto.
- Não há necessidade de conhecimentos ou treinamento especiais. Qualquer pessoa que programe pode desenvolver software com este modelo de ciclo de vida.

DESVANTAGENS

- Um dos problemas com esta técnica é que o código que sofreu várias correções fica cada vez mais difícil de modificar.
- Além disso, com bastante freqüência o que o cliente menos precisa é de código ruim produzido rapidamente, ele precisa ter certas necessidades atendidas.
- Essas necessidades são levantadas na fase de requisitos, que se for feita às pressas, deixará de atingir estes objetivos.

DESVANTAGENS

- É difícil avaliar o progresso até que o programa esteja funcionando.
- É muito difícil avaliar a qualidade e os riscos do projeto.
- Se no meio do projeto a equipe descobrir que decisões arquiteturais estavam erradas, não há solução a não ser começar tudo de novo.
- Codificar e consertar sem um plano de teste sistemático tende a produzir sistemas instáveis e com grande probabilidade de conterem erros.

CASCATA ENTRELAÇADO (SASHIMI)

- O modelo conhecido como Sashimi (DeGrace, 1990), ou Cascata Entrelaçado (Overlapped Waterfall), é uma tentativa de atenuar a característica BDUF do modelo Cascata.
- Ao invés de cada fase produzir documentação completa para a fase seguinte, o modelo Sashimi propõe que cada fase procure iniciar o tratamento de questões da fase seguinte e continue tratando as questões da fase anterior.

SASHIMI


Concepção do Software
Análise de Requisitos
Projeto da Arquitetura
Projeto Detalhado
Codificação e Depuração
Teste de Sistema
SASHIMI

SCRUM

- A idéia do modelo Sashimi de que cada fase se entrelaça apenas com a anterior e a posterior, entretanto, vai contra a observação de Royce.

- Em função disso, uma das evoluções de Sashimi mais importantes é o modelo SCRUM, que consiste em levar a idéia de fases entrelaçadas ao extremo pela redução do processo a uma única fase na qual todas as fases do modelo Cascata são realizadas paralelamente por profissionais especializados trabalhando em equipe.

VANTAGENS

- O modelo Sashimi tem o mérito de indicar que, por exemplo, a fase de análise de requisitos só estará completa depois que o projeto da arquitetura tiver terminado, e assim por diante.
- Este estilo de processo é adequado se o engenheiro de software avaliar que poderá obter ganhos de conhecimento sobre o sistema ao passar de uma fase para outra.
- O modelo também provê uma substancial redução na quantidade de documentação, pois as equipes das diferentes fases trabalharão juntas boa parte do tempo.

DESVANTAGENS

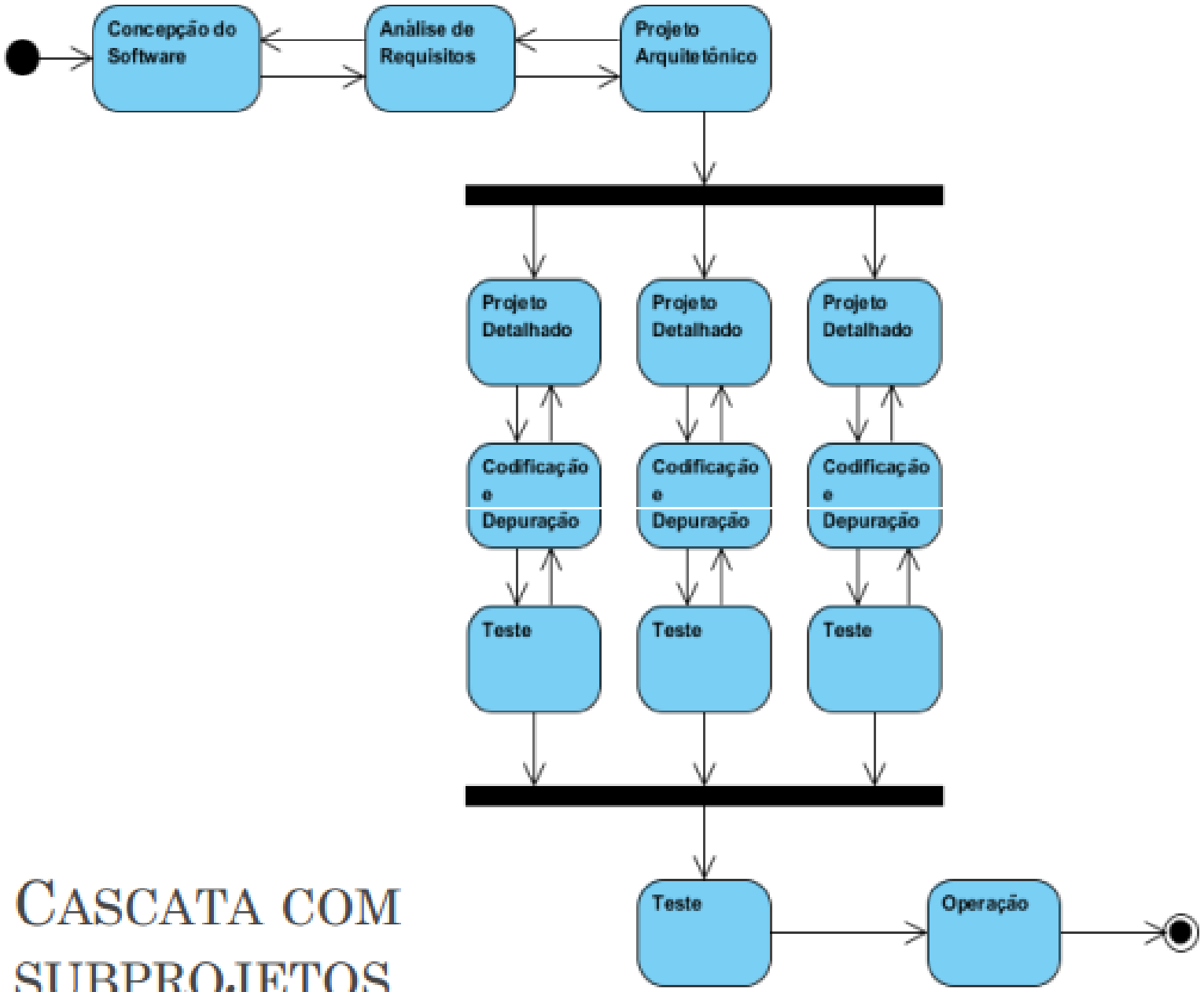
- Entre os problemas do modelo está o fato de que fica mais difícil definir marcos (millestones), pois não fica muito claro quando exatamente uma fase termina.

- Além disso, a realização de atividades paralelas com este modelo pode levar a falhas de comunicação, aceitação de hipóteses erradas e ineficiência no trabalho.

CASCATA COM SUBPROJETOS (WATERFALL WITH SUBPROJECTS)

- Permite que algumas fases do modelo Cascata sejam executadas em paralelo.
- Após a fase de projeto da arquitetura, o projeto pode ser subdividido de forma que vários subsistemas sejam desenvolvidos em paralelo por equipes diferentes, ou pela mesma equipe em momentos diferentes.

CASCATA COM SUBPROJETOS



VANTAGENS

- Este modelo é bem mais razoável de utilizar do que o modelo Cascata puro, visto que o fato de quebrar o sistema em subsistemas menores permite que subprojetos mais rápidos e fáceis de gerenciar sejam realizados.

- Esta técnica explora melhor as potencialidades de modularidade do projeto.

- Com ela, o progresso é mais facilmente visível, porque pode-se produzir várias entregas de partes funcionais do sistema a medida que ficam prontas.

DESVANTAGENS

- A maior dificuldade com este modelo está na possibilidade de surgirem interdependências imprevistas entre os subsistemas.
- O projeto arquitetônico deve ser bem feito de forma a minimizar tais problemas.
- Além disso, este modelo exige maior capacidade de gerência para impedir que sejam criadas inconsistências entre os subsistemas.

CASCATA COM REDUÇÃO DE RISCO (WATERFALL WITH RISK REDUCTION)

- Procura resolver um dos principais problemas do BDUF que é a dificuldade em se ter uma boa definição dos requisitos do projeto nas fases iniciais.
- Este modelo basicamente acrescenta uma fase de redução de riscos antes do início do processo em cascata.
- O objetivo do modelo é a redução do risco com os requisitos.
- A tônica é a utilização de técnicas que garantam que os requisitos serão os mais estáveis possíveis.

TÉCNICAS USADAS DURANTE A FASE ESPIRAL

- Desenvolver protótipos de interface com o usuário.
- Desenvolver storyboards com o usuário.
- Conduzir vários ciclos de entrevistas com o usuário e cliente.
- Filmar os usuários utilizando os sistemas antigos, sejam eles informatizados ou não.
- Utilizar extensamente quaisquer outras práticas de eliciação de requisitos.

CASCATA COM REDUÇÃO DE RISCO

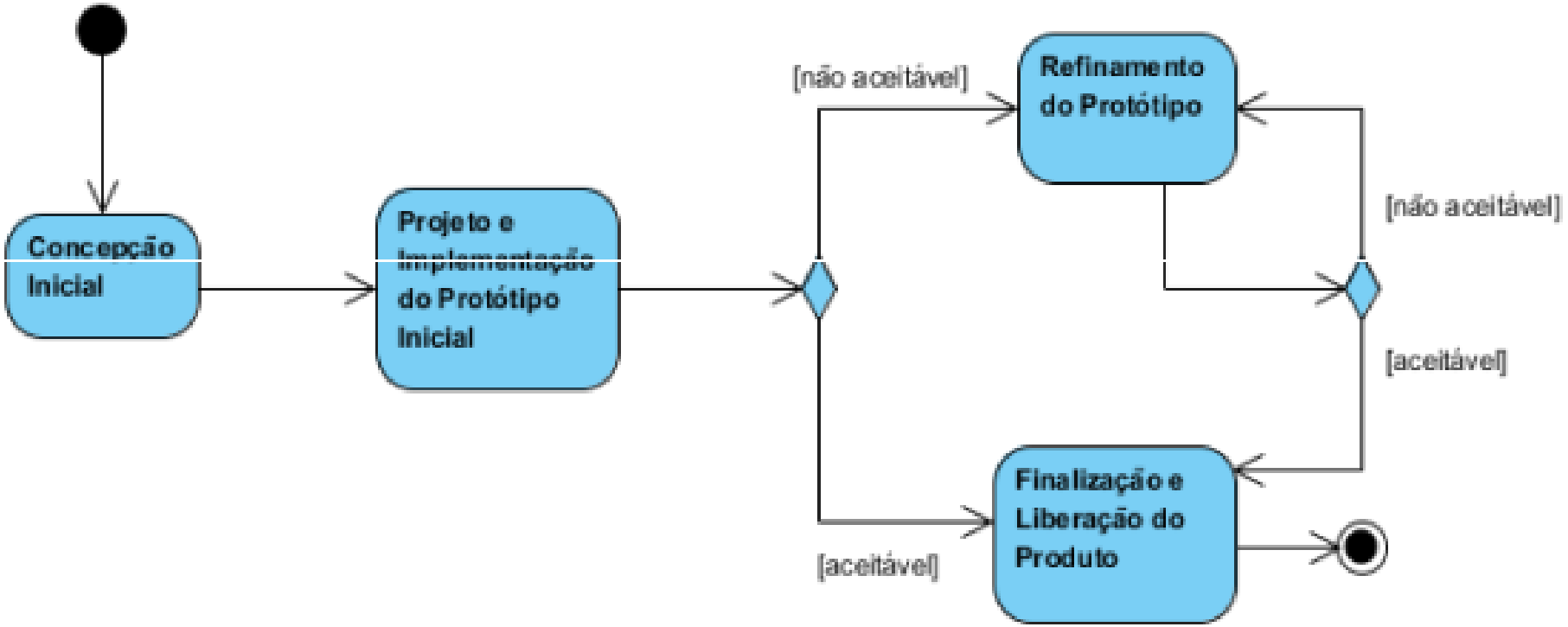
DISCUSSÃO

- A espiral de redução de riscos não precisa ficar restrita aos requisitos do projeto, ela pode ser aplicada a quaisquer outros riscos identificados.
- Este tipo e modelo é adequado a projetos com um grande número de riscos importantes.
- A espiral de redução de riscos é uma forma de a equipe garantir alguma estabilidade ao projeto antes de comprometer um grande número de recursos na sua execução.

PROTOTIPAÇÃO EVOLUCIONÁRIA (EVOLUTIONARY PROTOTYPING)

- O modelo de Prototipação Evolucionária (Brooks, 1975) sugere que a equipe de desenvolvimento trabalhe junto ao cliente os aspectos mais visíveis do sistema, na forma de protótipos (usualmente de interface), até que o produto seja aceitável.

Prototipação evolucionária



VANTAGENS

- Este modelo pode ser particularmente interessante se for difícil fazer o cliente entender os requisitos.
  - Neste caso, um protótipo do sistema será uma ferramenta mais fácil para o analista se comunicar com o cliente e chegar a um acordo sobre o que deve ser desenvolvido.

- Este modelo também pode ser interessante quando tanto a equipe quanto o cliente não conhecem bem os requisitos do sistema.
  - Pode ser difícil elaborar requisitos quando não se sabe exatamente o que é necessário, sem ver o software funcionando e testando o mesmo.

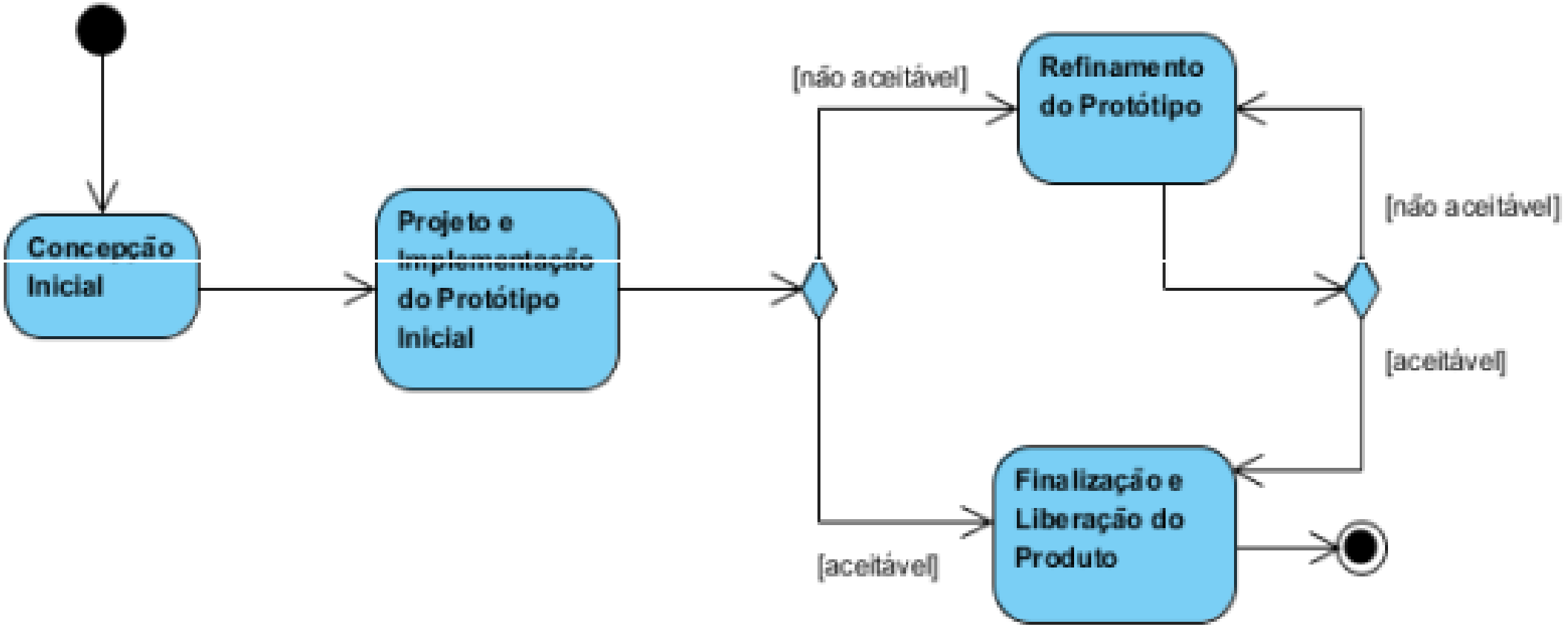
DESVANTAGENS

- O modelo não é muito bom em relação à previsão de tempo para desenvolvimento e também em relação à gerência do processo, já que é difícil avaliar quando cada fase foi efetivamente realizada.
- Pode até acabar acontecendo que o modelo se torne mais uma desculpa para usar Codificar e Consertar do que efetivamente seguir um ciclo de vida bem definido.
- Para evitar isso deve-se garantir que o processo efetivamente obtenha uma concepção real do sistema, com requisitos definidos da melhor forma possível, e com um projeto realista antes de iniciar a codificação propriamente
dita.

ENTREGAS EM ESTÁGIOS

- É uma variação mais bem estruturada do modelo Prototipação Evolucionária embora também seja considerado uma variação do modelo Cascata.
- Ao contrário da prototipação Evolucionária, o modelo Entregas em Estágios prevê que a cada ciclo a equipe planeje e saiba exatamente o que vai entregar ao cliente.
- A abordagem é interessante porque haverá vários pontos de entrega e o cliente poderá acompanhar mais diretamente a evolução do sistema.
- Não existe, portanto, o problema do modelo Cascata, onde o sistema só é entregue quando está totalmente acabado.

Prototipação evolucionária



VANTAGENS

- Uma das principais vantagens deste modelo (Ellis, 2010) é o fato de colocar funcionalidades úteis nas mãos do cliente antes de completar todo o projeto.
- Se os estágios forem planejados cuidadosamente, funcionalidades importantes estarão disponíveis muito mais cedo do que com outros ciclos de vida.
- Além disso, este modelo provê entregas mais cedo e de forma contínua, o que pode aliviar um pouco a pressão de cronograma colocada na equipe.

É NECESSÁRIO PLANEJAR EM DOIS NÍVEIS

- Técnico: as dependências técnicas entre os diferentes módulos entregáveis devem ser cuidadosamente verificadas. Se um módulo tem dependências com outro, o segundo deve ser entregue antes deste.

- Gerencial: deve-se procurar garantir que os módulos sejam efetivamente significativos para o cliente. Será menos útil entregar funcionalidades parciais que não possam produzir nenhum trabalho consistente.

ENTREGA EVOLUCIONÁRIA (EVOLUTIONARY DELIVERY)

- É um meio termo entre a Prototipação Evolucionária e a Entrega em Estágios.
- Neste modelo a equipe também desenvolve uma versão do produto, mostra ao cliente, e desenvolve novas versões baseadas no feedback do cliente.
- Depende do grau em que se pretende acomodar os requisitos solicitados:
  - se a ideia é acomodar todos ou a grande maioria dos novos requisitos, então a abordagem tende mais para Prototipação Evolucionária.
  - Se as entregas continuarão sendo planejadas de acordo com o previsto e as novas funcionalidades acomodadas aos poucos nas entregas, então a abordagem se parece mais com Entrega em Estágios.

ENTREGA EVOLUCIONÁRIA

- Este modelo permite então ajustes que lhe dão um pouco da flexibilidade do modelo de Prototipação Evolucionária ao mesmo tempo em que se tem o planejamento da Entrega em Estágios.
- As diferenças entre este modelo e os anteriores então está mais na ênfase do que nas atividades relacionadas.
- No modelo de Prototipação Evolucionária a ênfase está nos aspectos visíveis do sistema.
- Na Entrega Evolucionária, porém, a ênfase está nas funcionalidades mais críticas do sistema.

EXERCICIOS

- Criar resumos dos modelos apresentados