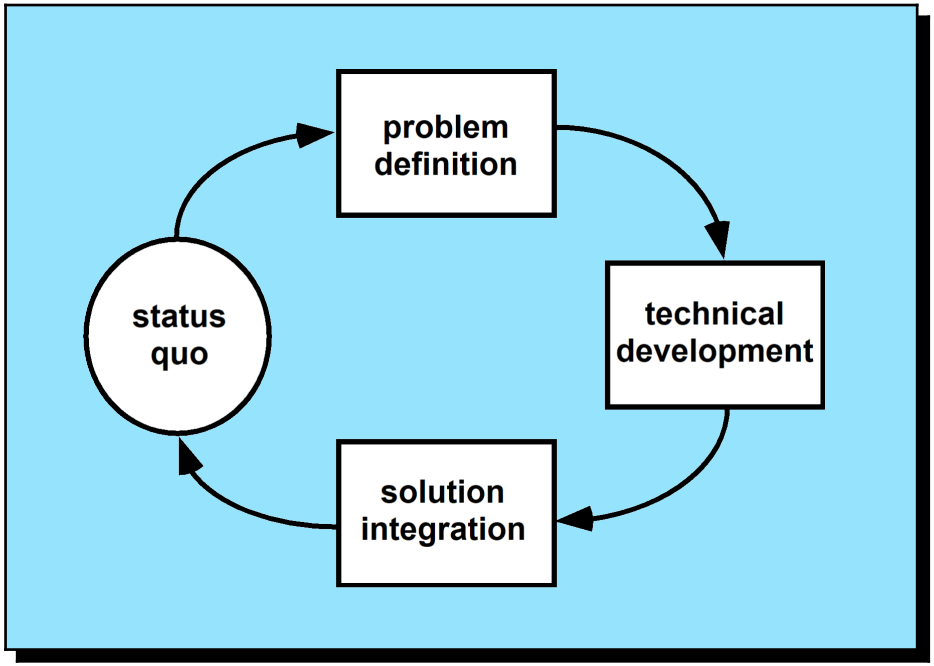
ENGENHARIA DE SOFTWARE 1

Modelos de Processo de SW

- **Modelo de processo de software** é uma representação abstrata de um processo de software
- Estratégia de desenvolvimento que abrange as camadas de processo, métodos e ferramentas.
- É escolhido com base na natureza do projeto e da aplicação, nos métodos e ferramentas a serem usados, e nos controles e nos produtos intermediários e finais que são requeridos.
- Pode ser encarado como um ciclo de solução de problema
- Razões p/ se modelar um processo
  - Formar um entendimento comum/ encontrar inconsistências, redundâncias e omissões/ encontrar e avaliar propostas mais adequadas aos objetivos



Fonte: PRESSMAN (2002)

MODELOS DE PROCESSO DE SOFTWARE

MODELOS TRADICIONAIS

Modelos de processo de SW – Sequencial Linear (cascata)

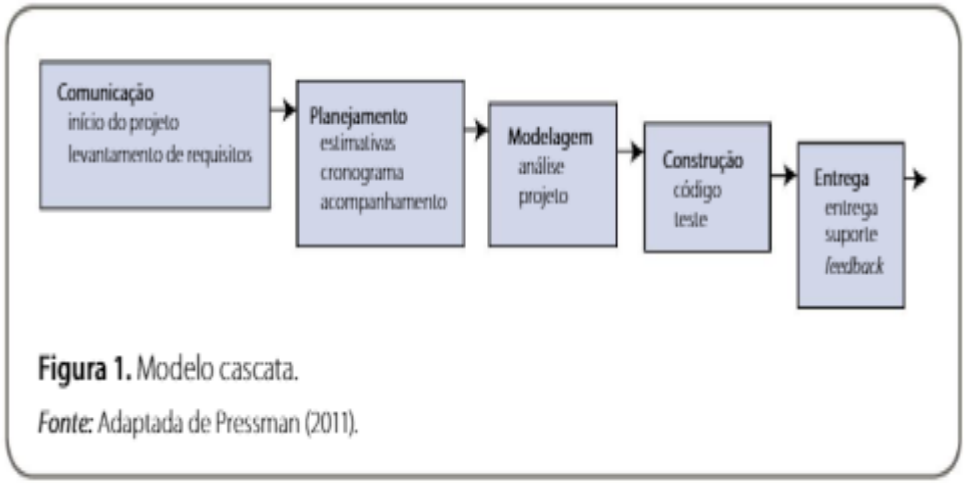


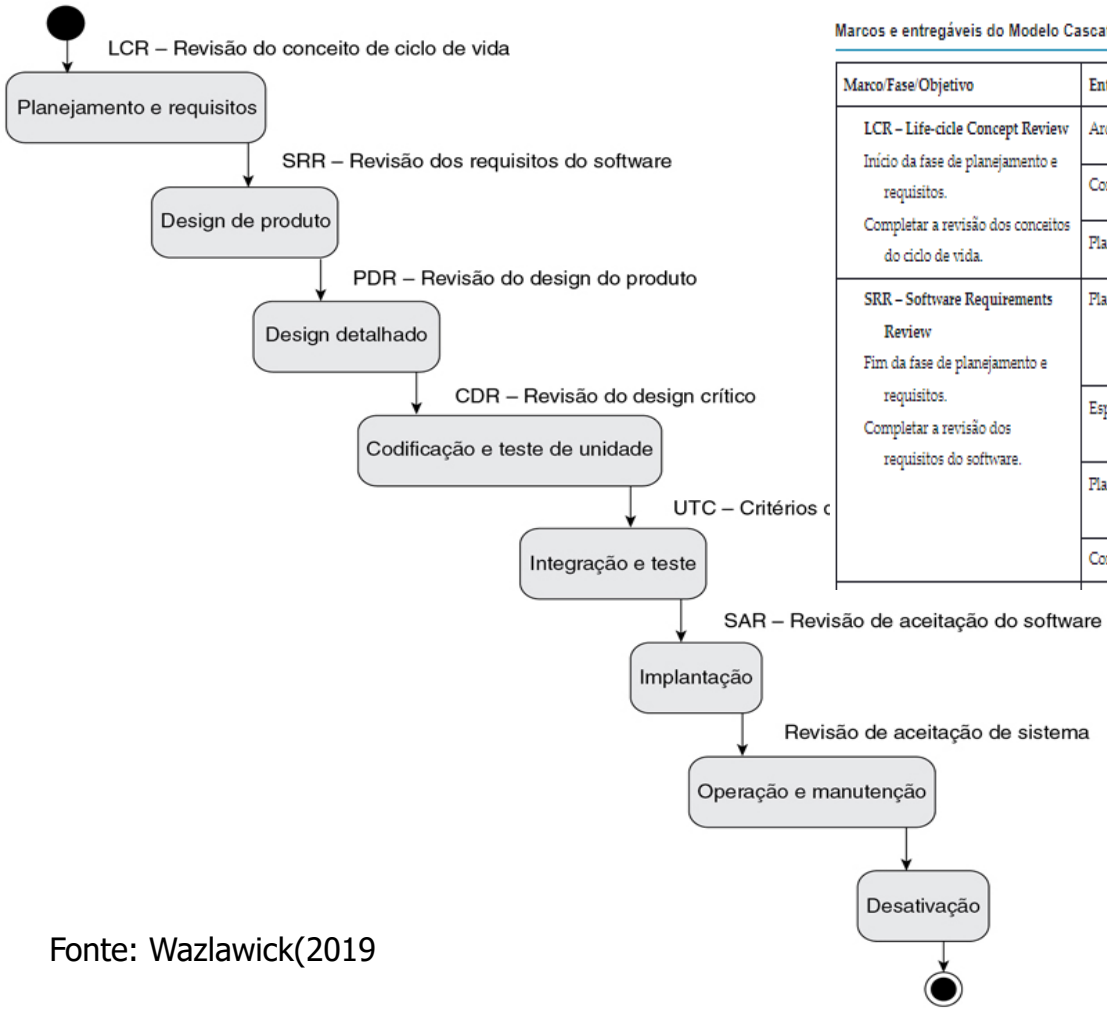
Figura 1. Modelo cascata.

Fonte: Adaptada de Pressman (2011).

- Baseia-se na filosofia **BDUF (Big Design Up Front – design completo antes de tudo)**:
  - ela propõe que, antes de produzir linhas de código, deve-se fazer um trabalho detalhado de análise e design, de forma que, quando o código for efetivamente produzido, esteja o mais próximo possível dos requisitos do cliente.

- **(+) Fácil de entender e planejar**
- **(+) Funciona bem em projetos pequenos com requisitos bem definidos**
- **(-) Projetos reais raramente seguem o fluxo sequencial proposto pelo modelo.**
- **(-) Com frequência, é difícil para o cliente estabelecer explicitamente todas as necessidades no início da maioria dos projetos.**
- **(-) O cliente deve ter paciência. Uma versão operacional do(s) programa(s) não estará disponível antes de estarmos próximos ao final do projeto.**
- **(-) Erros graves podem não ser detectados até o programa operacional ser revisto.**

Modelos de processo de SW – Sequencial Linear (cascata)



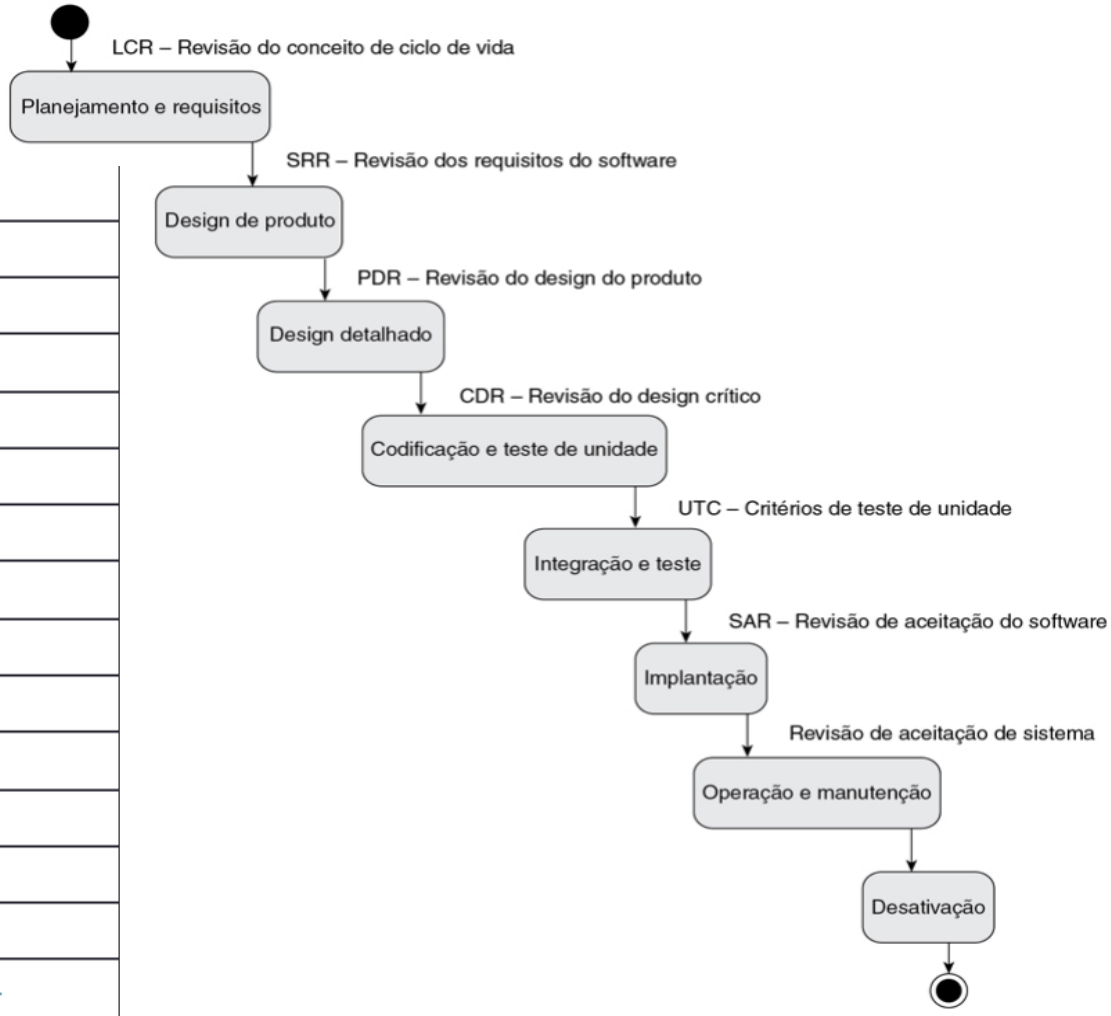
Marcos e entregáveis do Modelo Cascata

Marco/Fase/Objetivo	Entregáveis
LCR – Life-cicle Concept Review Início da fase de planejamento e requisitos. Completar a revisão dos conceitos do ciclo de vida.	Arquitetura de sistema aprovada e validada, incluindo questões básicas de hardware e software.
	Conceito de operação aprovado e validado, incluindo questões básicas de interação humano-computador.
	Plano de ciclo de vida de alto nível, incluindo marcos, recursos, responsabilidades, cronogramas e principais atividades.
SRR – Software Requirements Review Fim da fase de planejamento e requisitos. Completar a revisão dos requisitos do software.	Plano de desenvolvimento detalhado: detalhamento de critérios de desenvolvimento de marcos, orçamento e alocação de recursos, organização da equipe, responsabilidades, cronograma, atividades, técnicas e produtos a serem usados. Plano de uso detalhado: contraparte para os itens do plano de desenvolvimento como treinamento, conversão, instalação, operações e suporte.
	Especificações de requisitos de software aprovadas e validadas: requisitos funcionais, de performance e especificações de interfaces validadas em relação a completude, consistência, testabilidade e exequibilidade.
	Plano de controle de produto detalhado: plano de gerenciamento de configuração, plano de garantia de qualidade, plano geral de V&V (verificação e validação), excluindo detalhes dos planos de testes.
	Contrato de desenvolvimento (formal ou informal) aprovado com base nos itens anteriores.

Fonte: Wazlawick(2019

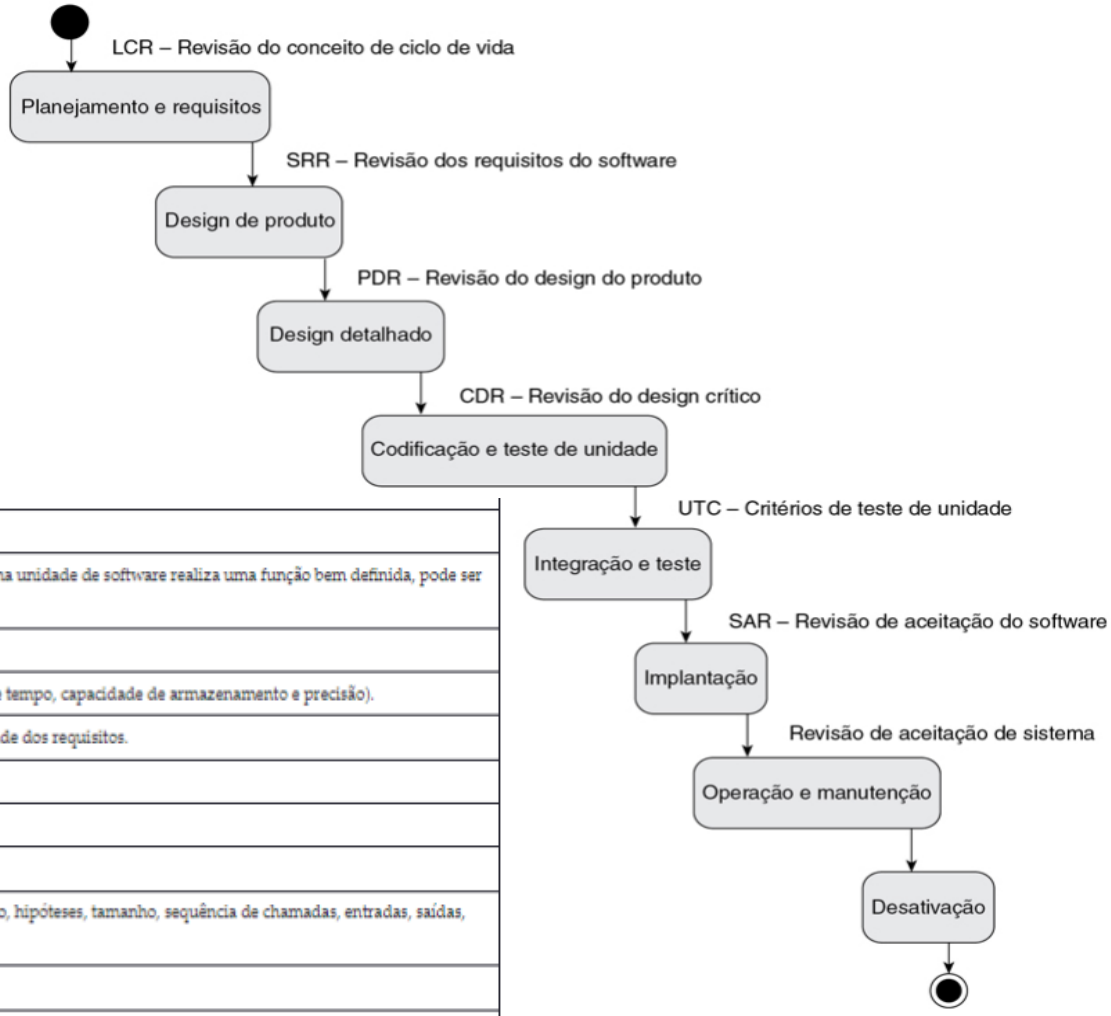
Modelos de processo de SW – Sequencial Linear (cascata)

UTC – Unit Test Criteria Fim da fase de codificação e teste de unidade. Satisfação dos critérios de teste de unidade.	Verificação de todas as unidades de computação usando-se não apenas valores nominais, mas também valores singulares e extremos.
	Verificação de todas as entradas e saídas unitárias, incluindo mensagens de erro.
	Exercício de todos os procedimentos executáveis e todas as condições de teste.
	Verificação de conformação a padrões de programação.
	Documentação em nível de unidade completada.
SAR – Software Acceptance Review Fim da fase de integração e teste. Completar a revisão da aceitação do software.	Testes de aceitação do software satisfeitos.
	Verificação da satisfação dos requisitos do software.
	Demonstração de performance aceitável acima do nominal, conforme especificado.
	Aceitação de todos os produtos do software: relatórios, manuais, especificações e bases de dados.
System Acceptance Review Fim da fase de implantação. Completar a revisão da aceitação do sistema.	Satisfação do teste de aceitação do sistema.
	Verificação da satisfação dos requisitos do sistema.
	Verificação da prontidão operacional de software, hardware, instalações e pessoal.
	Aceitação de todas as entregas relacionadas com o sistema: hardware, software, documentação, treinamento e instalações.
	Todas as conversões especificadas e atividades de instalação foram completadas.
Fim da fase de operação e manutenção. Corresponde à desativação do sistema.	Foram completadas todas as atividades do plano de desativação: conversão, documentação, arquivamento e transição para um novo sistema.



Fonte: Wazlawick(2019

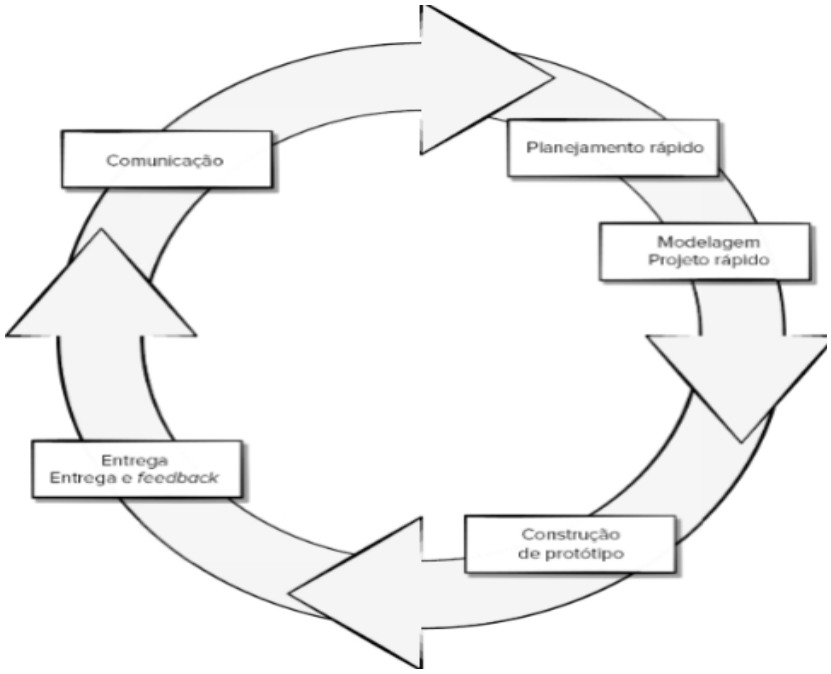
Modelos de processo de SW – Sequencial Linear (cascata)



PDR – Product Design Review Fim da fase de design de produto. Completar a revisão do design do produto.	Especificação do design do produto de software verificada.
	Hierarquia de componentes do programa, interfaces de controle e dados entre as unidades (uma unidade de software realiza uma função bem definida, pode ser desenvolvida por uma pessoa e costuma ter de 100 a 300 linhas de código).
	Estruturas de dados lógicas e físicas detalhadas em nível de seus campos.
	Orçamento para recursos de processamento de dados (incluindo especificações de eficiência de tempo, capacidade de armazenamento e precisão).
	Verificação do design com referência a completude, consistência, exequibilidade e rastreabilidade dos requisitos.
	Identificação e resolução de todos os riscos de alta importância.
	Plano de teste e integração preliminar, plano de teste de aceitação e manual do usuário.
CDR – Critical Design Review Fim da fase de design detalhado. Completar o design e revisar aspectos críticos das unidades.	Especificação de design detalhado revisada para cada unidade.
	Para cada rotina (menos de 100 instruções) dentro de uma unidade, especificar nome, propósito, hipóteses, tamanho, sequência de chamadas, entradas, saídas, exceções, algoritmos e fluxo de processamento.
	Descrição detalhada da base de dados.
	Especificações e orçamentos de design verificados em relação a completude, consistência e rastreabilidade dos requisitos.
	Plano de teste de aceitação aprovado.
	Manual do usuário e rascunho do plano de teste e integração completados.

zlawick(2019

Modelos de processo de SW – prototipagem



Fonte: PRESSMAN (2016)

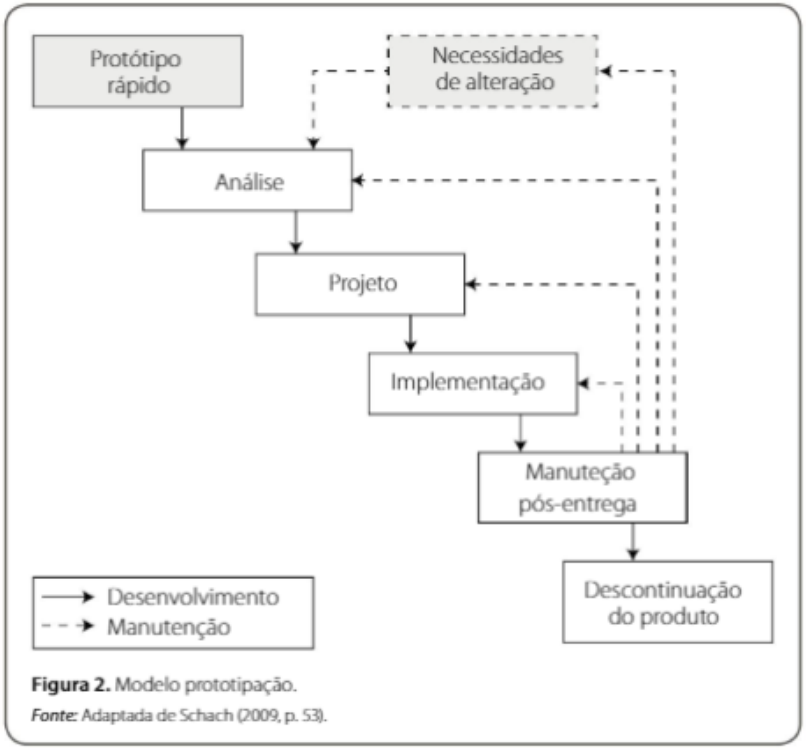
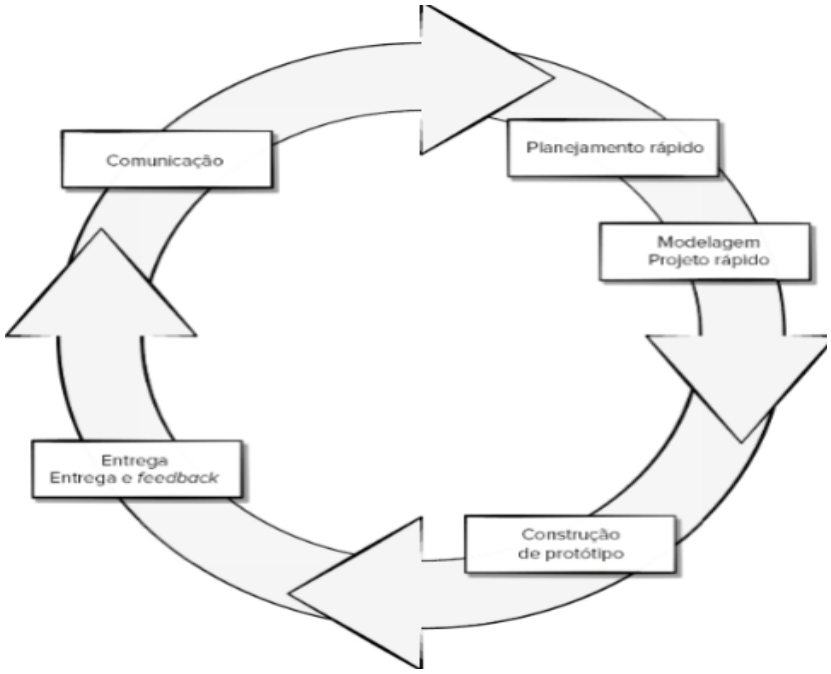


Figura 2. Modelo prototipação.
Fonte: Adaptada de Schach (2009, p. 53).

- Aspectos do modelo de prototipagem
  - (+) identificar mais detalhadamente requisitos de entrada, processamento e saída
  - (+) os requisitos se tornam mais visuais e intuitivos, uma vez que o profissional redigirá o documento de especificação de requisitos tendo o protótipo como base
  - (+) Impacto reduzido por mudanças em requisitos
  - (+) Ciientes estão envolvidos no processo mais cedo e de forma mais frequente
  - (+) Probabilidade reduzida de rejeição do produto final
  - OBS: resista a pressão para aperfeiçoar um protótipo mal feito dentro de uma linha de produção. O resultado quase sempre é de baixa qualidade.
  - Idealmente, o protótipo serve como um mecanismo para a identificação dos requisitos de software

Modelos de processo de SW – prototipagem



Fonte: PRESSMAN (2016)

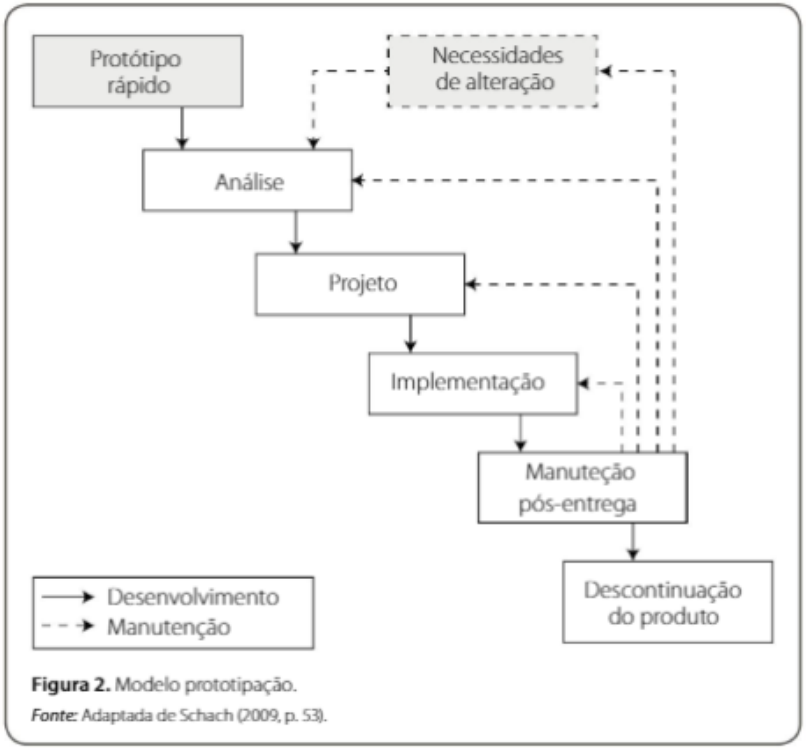
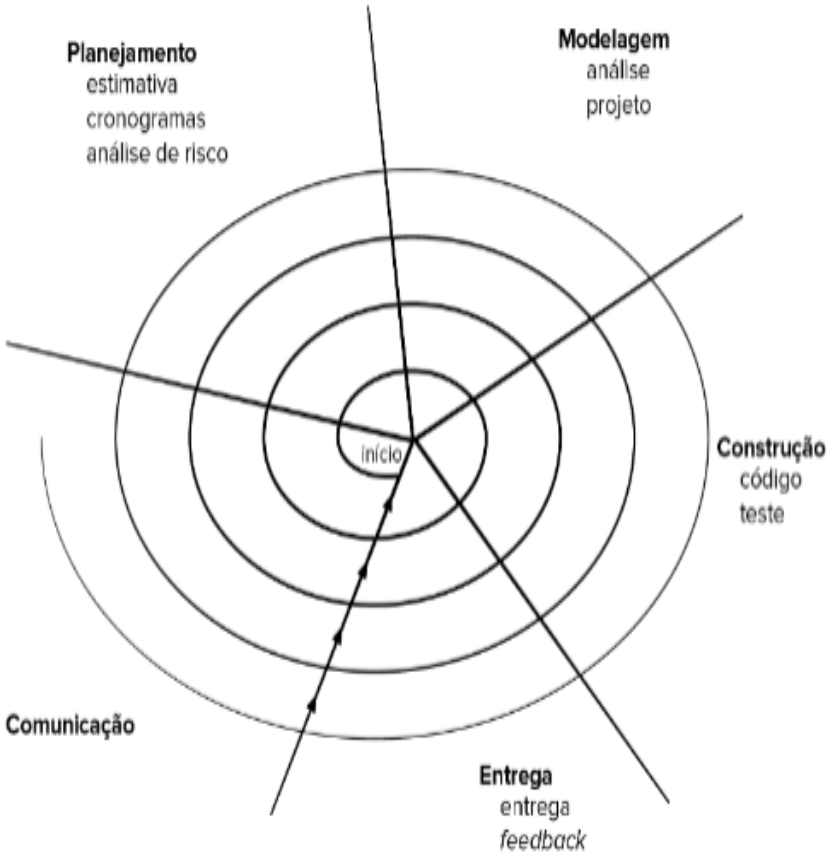


Figura 2. Modelo prototipação.
Fonte: Adaptada de Schach (2009, p. 53).

- Aspectos do modelo de prototipagem
  - (-) Envolvimento dos clientes pode causar atrasos
  - (-) O cliente vê o que parece ser uma versão executável do sw e ignora outros aspectos
  - (-) às vezes, o prótótipo pode ser considerado uma fonte de retrabalho
  - (-) O desenvolvedor frequentemente faz concessões na implementação a fim de conseguir rapidamente um protótipo executável (quais os impactos?)

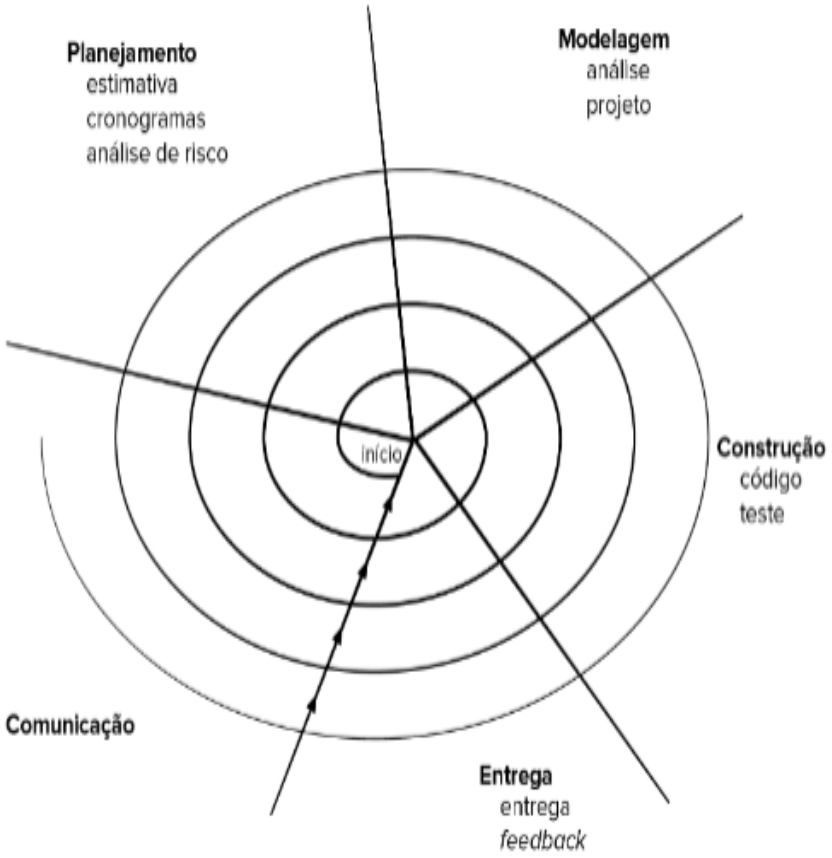
Modelo espiral



Fonte: PRESSMAN (2016)

- Aspectos do modelo espiral
- Combina a natureza iterativa da prototipagem com os aspectos controlados e sistemáticos do linear(cascata)
  - (+) potencial para o desenvolvimento rápido de versões incrementais do sw
  - (+) adaptado p/ aplicação ao longo da vida do sw
  - (+)Envolvimento contínuo do cliente.
  - (+) Os riscos de desenvolvimento são gerenciados.
  - (+) Indicado para projetos grandes e complexos.
  - (+) Funciona bem para produtos extensíveis

Modelo espiral



Fonte: PRESSMAN (2016)

- Aspectos do modelo espiral
- Combina a natureza iterativa da prototipagem com os aspectos controlados e sistemáticos do linear(cascata)
  - (-) Falhas na análise de risco podem condenar o projeto.
  - (-) exige competência considerável na avaliação de riscos e depende daquela p/ obter sucesso
  - (-) O projeto pode ser difícil de gerenciar. Requer uma equipe de desenvolvimento especializada.
  - (-) pode ser difícil convencer os clientes (particularmente em situações de contrato) que a abordagem evolucionária é controlável

Modelos de processo de SW evolucionário
Modelo incremental

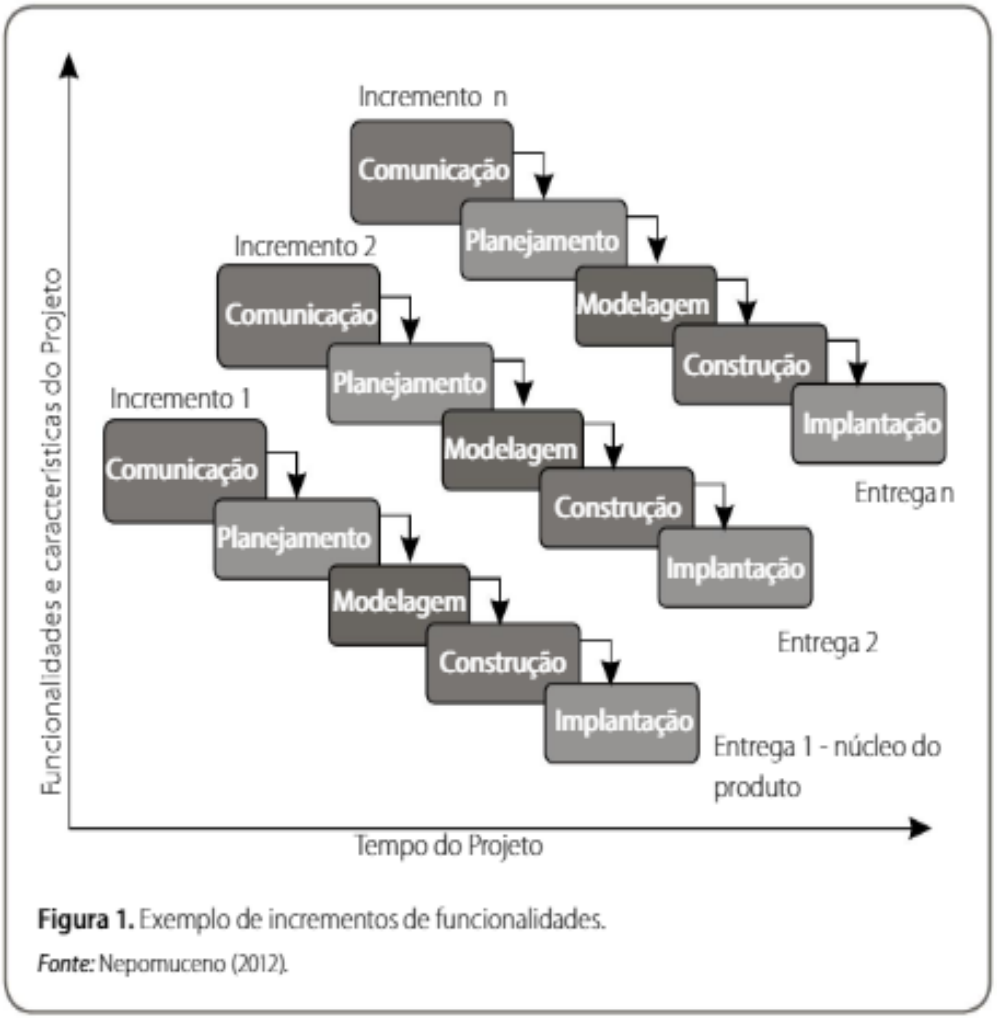
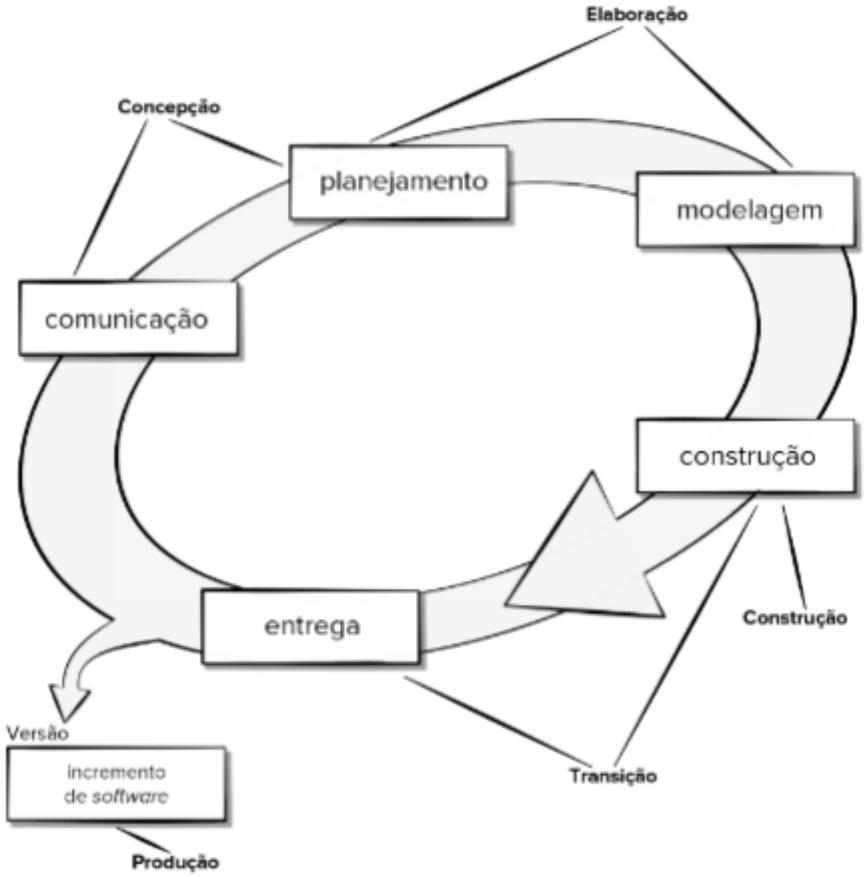


Figura 1. Exemplo de incrementos de funcionalidades.

Fonte: Nepomuceno (2012).

- (+) reduçãos dos custos com manuençãodo sistema (identificação dos erros)
- (+) Melhor controle de cronograma
- (+) Melhor probabilidade de atendimento dos requisitos dos clientes
- (-) Alguma dificuldade de gerenciamento (fases podem estar ocorrendo em simultâneo)
- (-) Necessidade que o cliente esteja disposto a prover *feedback* constante

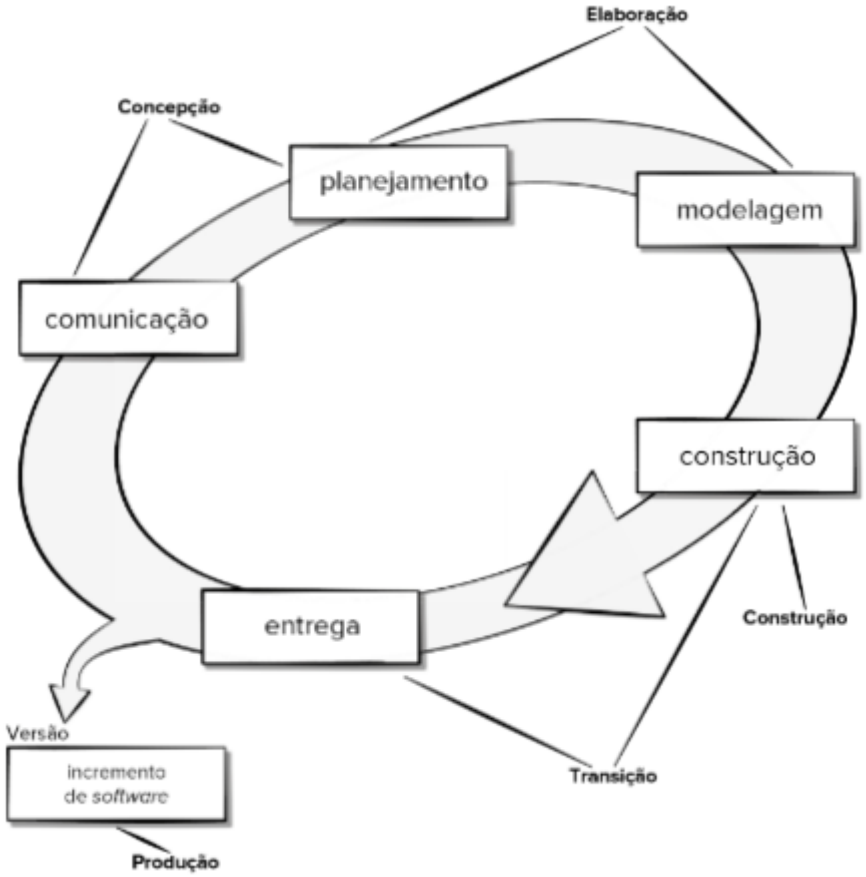
Modelos de processo unificado



- Tentativa de aoroveitar os melhores recursos dos modelos tradicionais acrescentanto aspectos de agilidade.
- (+) Documentação de qualidade enfatizada.
- (+) Envolvimento contínuo do cliente.
- (+) Acomoda mudanças de requisitos.
- (+) Funciona bem para projetos de manutenção.

Fonte: PRESSMAN (2016)

Modelos de processo unificado



- (-) Os casos de uso nem sempre são precisos.
- (-) Complicada integração de incremento de software.
- (-) Fases sobrepostas podem causar problemas.
- (-) Requer equipe de desenvolvimento especializada.

Fonte: PRESSMAN (2016)

RESUMO

Tabela 2.1 Comparação entre modelos de processo

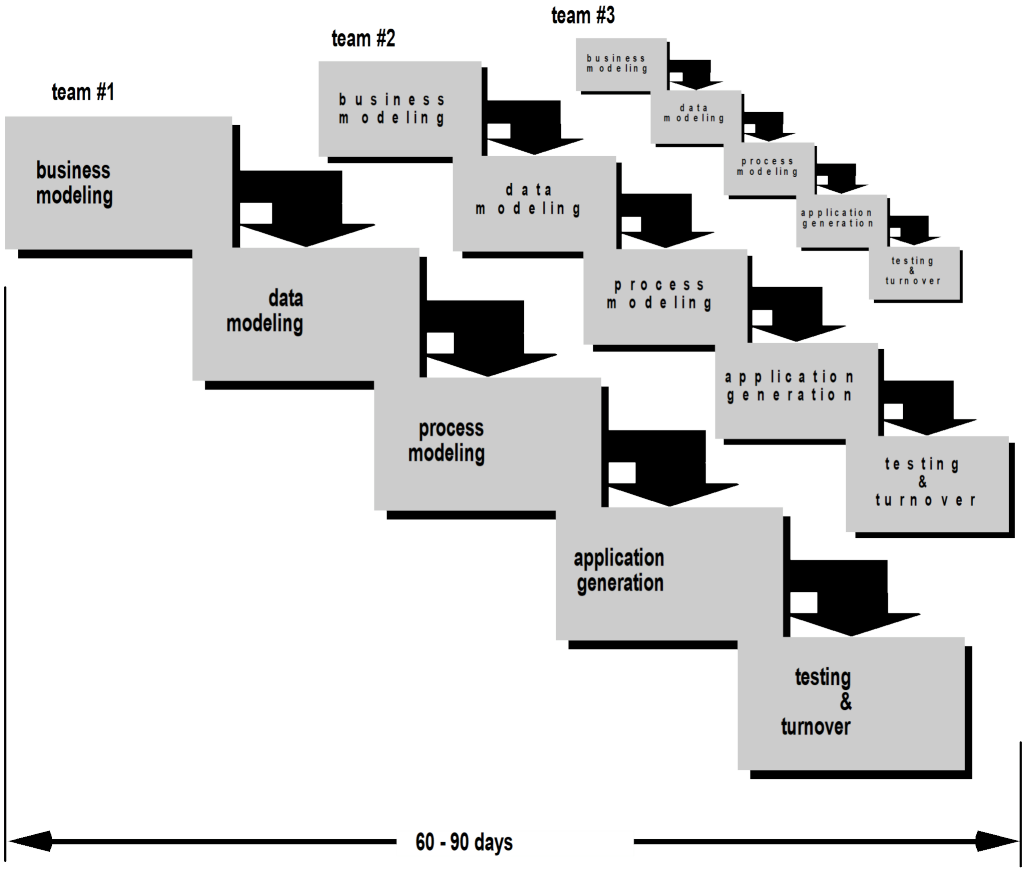
Prós do modelo cascata	É fácil de entender e planejar. Funciona para projetos pequenos e bem compreendidos. A análise e o teste são simples e diretos.
Contras do modelo cascata	Não se adapta bem a mudanças. O teste ocorre nas fases finais do processo. A aprovação do cliente vem no final.
Prós da prototipação	O impacto das alterações aos requisitos é reduzido. O cliente se envolve bastante e desde o início. Funciona bem para projetos pequenos. A probabilidade de rejeição do produto é reduzida.
Contras da prototipação	O envolvimento do cliente pode causar atrasos. Pode haver a tentação de "embalar" o protótipo. Desperdiça-se trabalho em um protótipo descartável. É difícil de planejar e gerenciar.
Prós do modelo espiral	Há envolvimento contínuo dos clientes. Os riscos de desenvolvimento são gerenciados. É apropriado para modelos grandes e complexos. Funciona bem para artefatos extensíveis.
Contras do modelo espiral	Falhas de análise de risco podem fadar o projeto ao fracasso. O projeto pode ser difícil de gerenciar. Exige uma equipe de desenvolvimento especializada.
Prós do Processo Unificado	A documentação de alta qualidade é enfatizada. Há envolvimento contínuo dos clientes. Adapta-se a alterações aos requisitos. Funciona bem para projetos de manutenção.
Contras do Processo Unificado	Os casos de uso nem sempre são precisos. A integração de incrementos de *software* é complicada. A sobreposição das fases pode causar problemas. Exige uma equipe de desenvolvimento especializada.

Fonte: PRESSMAN (2016)

MODELOS DE PROCESSO DE SOFTWARE

MODELOS ÁGEIS

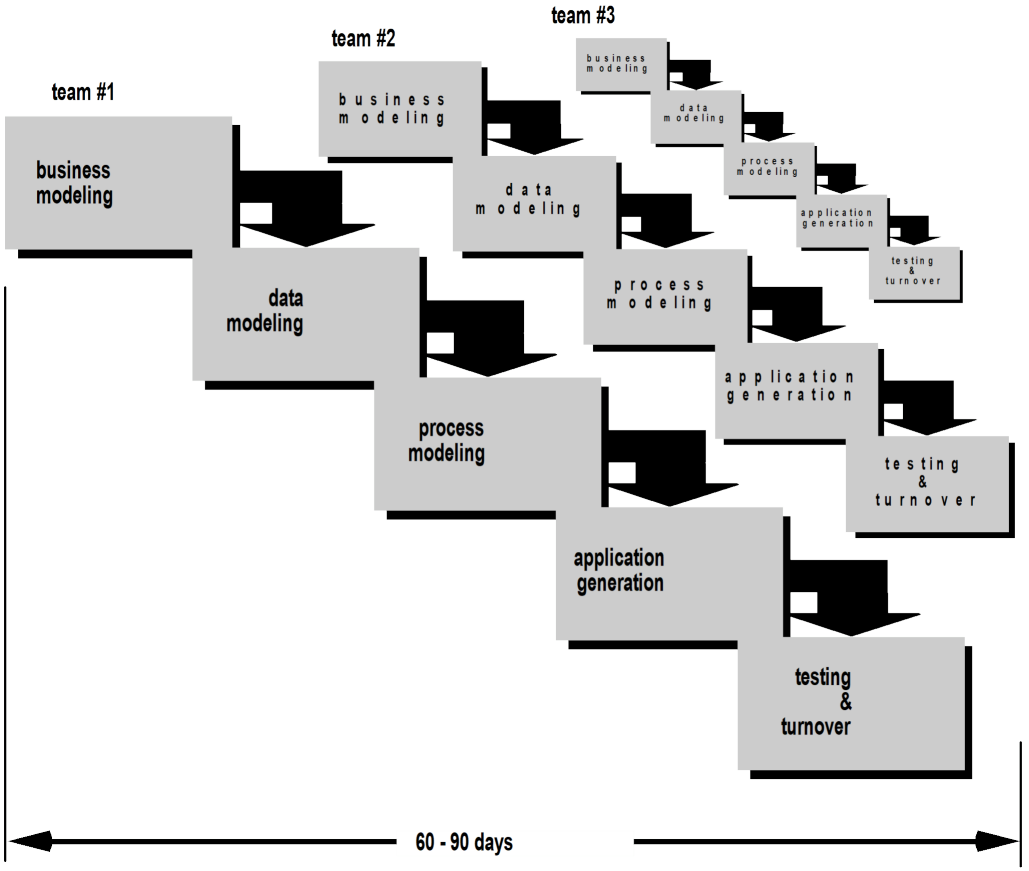
Modelos de processo de SW – RAD(Rapid Application Development)



Fonte: PRESSMAN (2002) **RAD**

- Modelagem do negócio:
  - Que informação dirige o processo de negócio? Que informação é gerada? Quem a gera? Para onde vai a informação? Quem a processa?
- Para projetos grandes, mas mensuráveis, o RAD exige recurso humanos suficientes para criar um número adequado de equipes
- Exige desenvolvedores e clientes extremamente compromissados; caso contrário, os projetos falharão
- Não adequado quando riscos técnicos forem elevados
- Se o sistema não puder ser adequadamente modularizado, a construção dos componentes será problemática

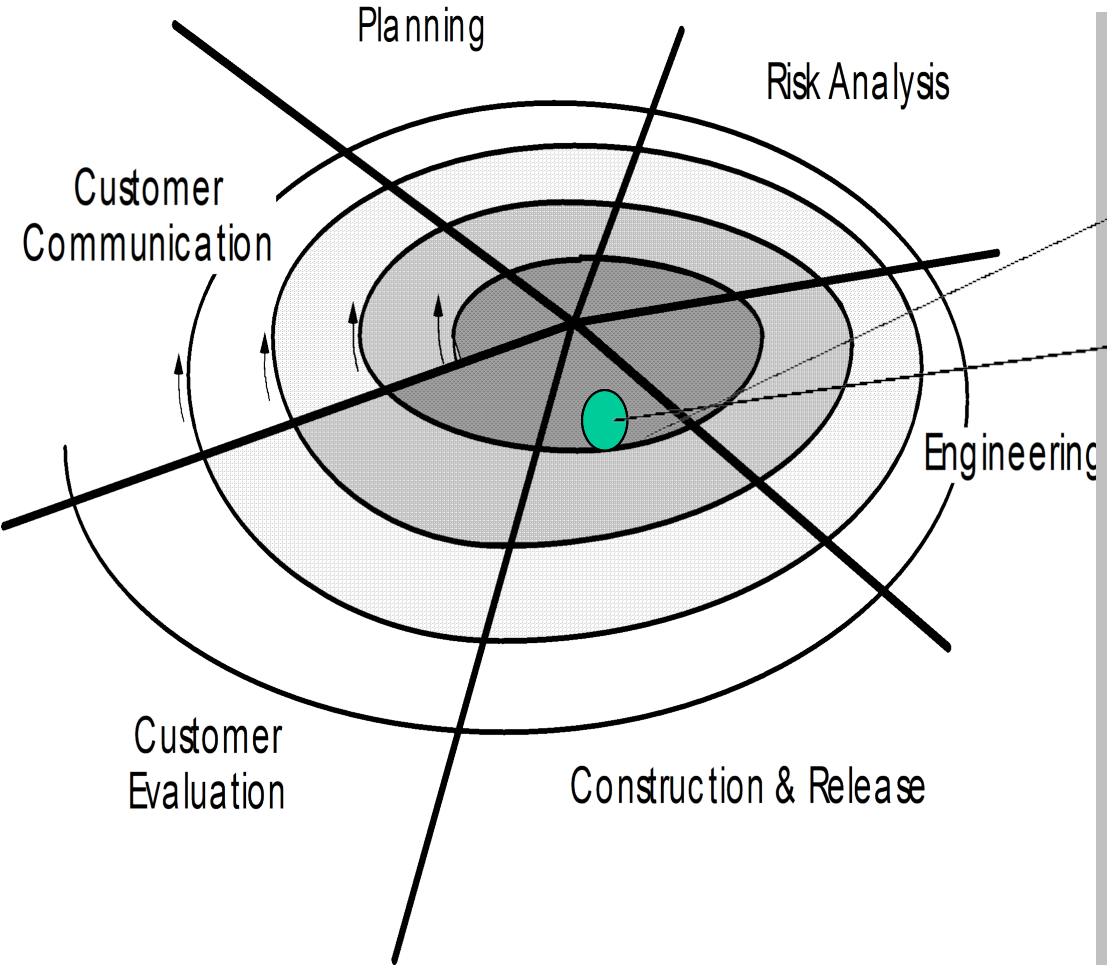
Modelos de processo de SW – RAD(Rapid Application Development)

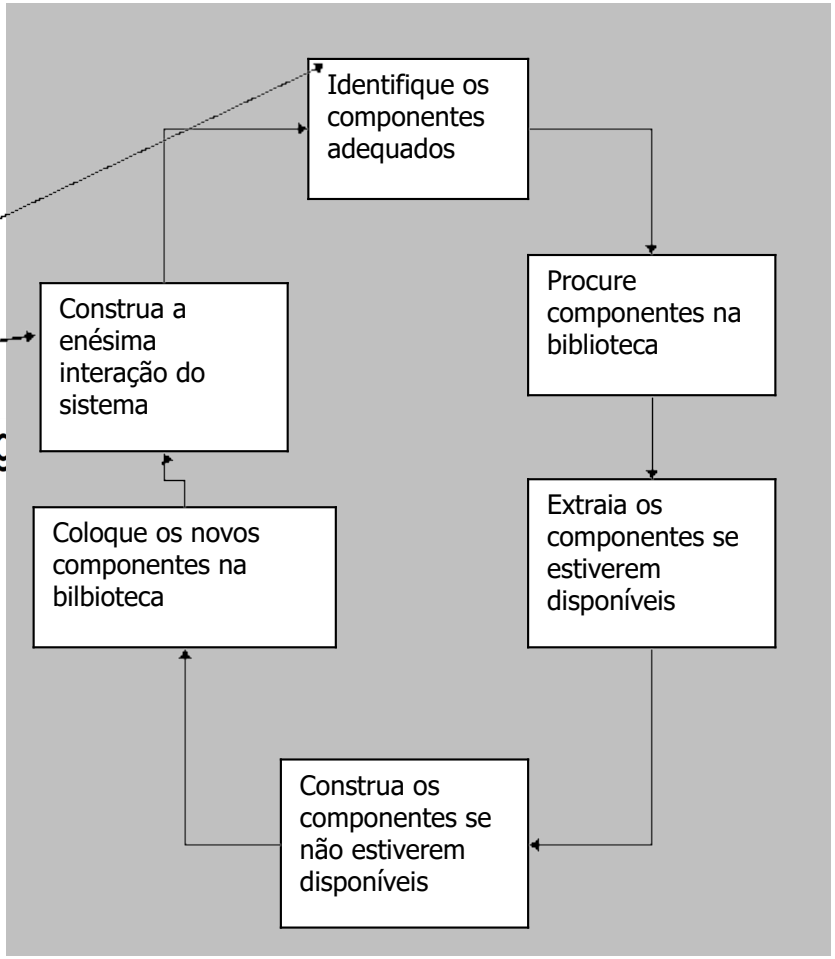


- Modelagem dos dados:
  - Conj. de objetos de dados que são necessários para dar suporte ao negócio. Atributos e relações
- Modelagem do processo
  - Objetos dos dado são transformados para conseguir o fluxo de informação necessários para implementar uma função do negócio → descrições de processamento
- Geração da aplicação
- Teste e reuso

Fonte: PRESSMAN (2002) **RAD**

Modelos de processo de SW
Modelo baseado em componentes





Fonte: PRESSMAN (2002)

Referências

- PRESSMAN, R. S; MAXIM, B. R. Engenharia de Software: Uma Abordagem Profissional. 8. ed. McGraw-Hill, 2016.
- SOMMERVILLE, I. Engenharia de Software. 9. ed. Pearson, 2011.