Processo de *Software*

Processo de *software*

- A Engenharia de *Software*, na quase totalidade de sua existência, focou-se em traduzir sistemas de informação já existentes para o computador.
- Por isso, a literatura especializada dá maior relevância ao processo de desenvolvimento em si.
- Isto é justificado porque, se os processos de negócio já existem e são maduros, então o resultado final fica a cargo da implementação correta do processo.

Métodos orientados a planos × métodos ágeis

- Nos últimos 20 anos, há um grande embate dentro da Engenharia de *Software* entre métodos orientados a planos e os novos métodos ágeis.
- Os métodos orientados a planos são métodos dentro do processo de desenvolvimento de *software* mais antigos e que se baseiam no máximo planejamento.
- Abordaremos aqui os processos de *software* de acordo com este paradigma.

Atividades do processo de *software*

Os diferentes modelos de processo de desenvolvimento de *software* baseiam-se geralmente nas seguintes atividades:

- especificação;
- projeto e implementação;
- validação;
- evolução.

Cada uma destas atividades é composta por outras inúmeras atividades, como documentação e gerenciamento de configuração de *software*.

Modelos de processo

Devido à complexidade do processo de *software*, pode-se representá-lo através de modelos, representações simplificadas do processo. Alguns dos principais modelos são:

- cascata;
- desenvolvimento incremental.

Os modelos também são chamados de paradigmas de processo.

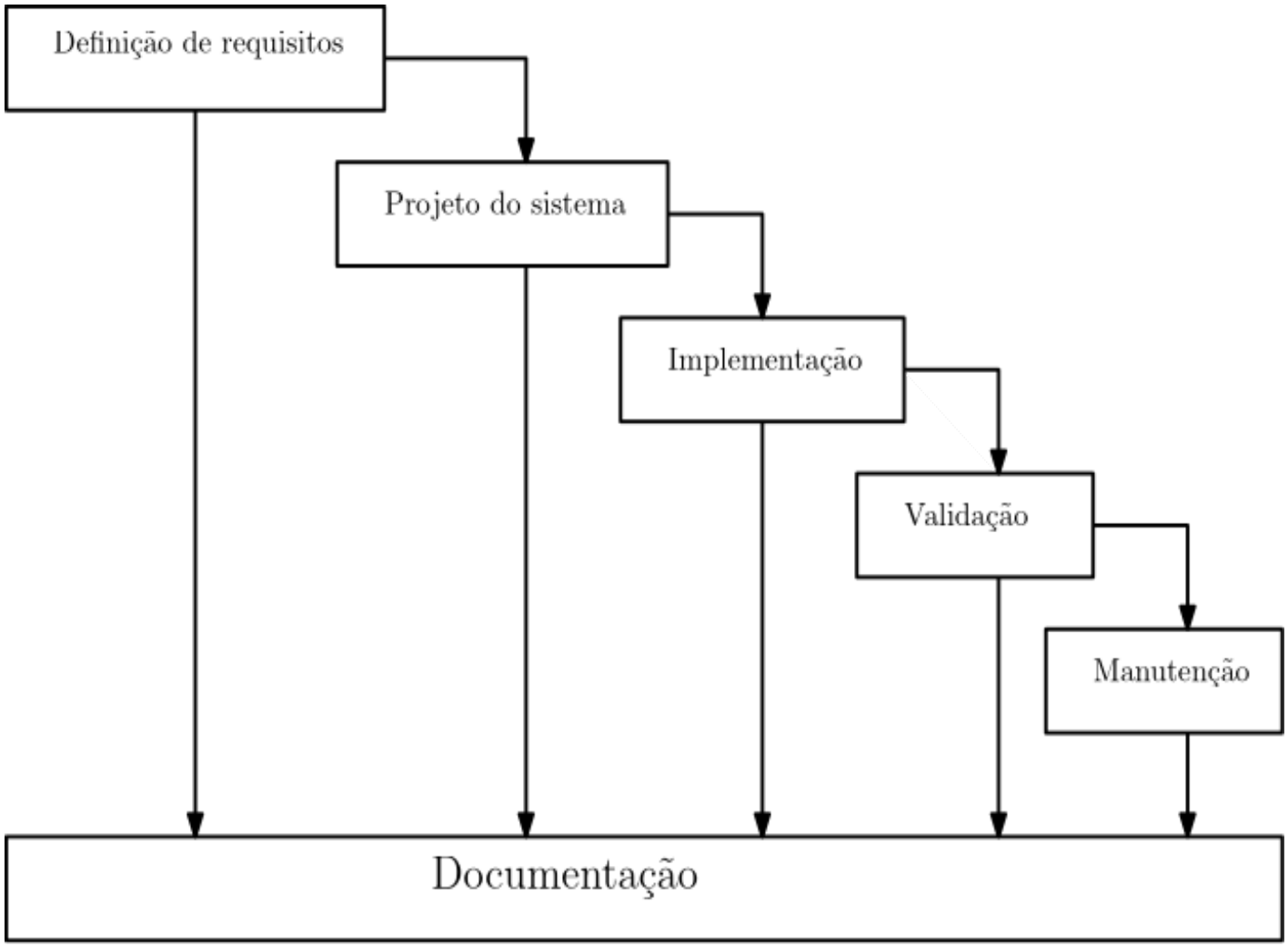
Modelo cascata

- O modelo cascata é o modelo clássico para o desenvolvimento de *software*.
- Ele e o Rational Unified Process (RUP) são os modelos mais emblemáticos dos métodos orientados a planos.

Modelo cascata

- É formado por atividades encadeadas, onde uma atividade só pode ser executada caso uma atividade anterior seja terminada.
- Exige um extenso planejamento prévio das atividades.
- É composto pelas seguintes atividades:
  - análise e definição de requisitos;
  - projeto de sistema e *software*;
  - implementação e teste unitário
  - teste de integração e de sistema;
  - operação e manutenção.

Modelo cascata



Modelo cascata

- Cada fase do modelo cascata gera documentos que validam a fase.
- Erros ou modificações nos requisitos fazem com que o processo retorne a etapas anteriores.
- As fases iniciais não produzem *software*, apenas documentação.
- O cliente só recebe um produto funcional ao fim das etapas.

Modelo cascata

- O modelo não reage bem a mudanças nos requisitos devido à sua natureza conservadora.
- Projetos reais nem sempre seguem o modelo à risca.
- Nem sempre todos os requisitos são conhecidos no início do projeto.
- Em um ambiente de negócios dinâmico tende a ser ineficiente.

Modelo cascata

- Alguns contratos, como os firmados com governos, exigem ainda um processo em cascata.
- *Softwares* embarcados e sistemas críticos o cascata encontra bastante espaço.
- *Softwares* produzidos em trabalhos acadêmicos, onde a equipe de pesquisa é também a de desenvolvimento, o cascata também é utilizado, mesmo que de maneira menos formal.

Desenvolvimento incremental

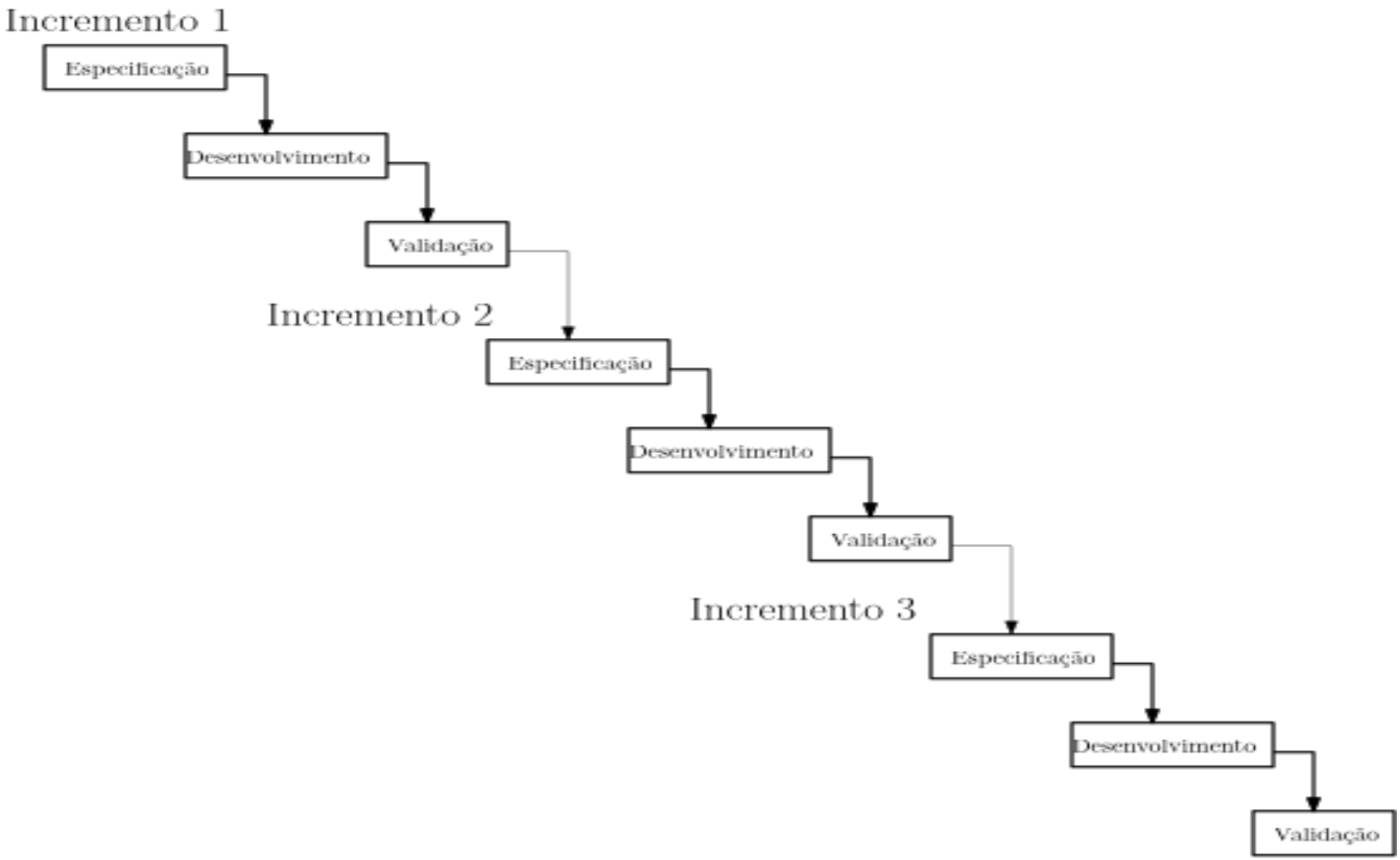
- O desenvolvimento incremental divide o processo de desenvolvimento de *software* em iterações.
- Em cada iteração, a versão do *software* é posta para validação com o cliente.
- As versões intermediárias não possuem todas as funcionalidades requeridas, apenas as necessárias para se obter um *software* funcional.
- Este modelo serve tanto para abordagens orientadas a planos quanto para métodos ágeis.

Desenvolvimento incremental

Cada iteração é dividida nas seguintes etapas:

- especificação;
- desenvolvimento;
- validação.

Desenvolvimento incremental



Desenvolvimento incremental

- Muito mais dinâmico e capacidade de tolerar erros e mudanças que o modelo cascata.
- Os custos das mudanças de requisitos são reduzidos.
- O *feedback* constante do cliente ajuda a prevenir erros de identificação de requisitos.
- É o modelo adotado em larga escala pela Engenharia de *Software* moderna, em especial na criação de produtos de *software* (próxima aula) e nos métodos ágeis.

Desenvolvimento incremental

- Há a necessidade de mensurar, de maneira correta, o tamanho das iterações.
- Iterações longas degeneram o processo em um modelo em cascata.
- Iterações curtas geram *overhead* com mais validações e planejamentos, além de gerar prazos mais curtos à equipe.

Desenvolvimento incremental

- Do ponto de vista da gerência de projetos, o processo, em si, torna-se menos visível.
- Exige maturidade organizacional e o uso de boas práticas de arquitetura de *software*.
- Alguns contratos são difíceis de adequar ao desenvolvimento incremental, pois exigem a documentação completa e total dos requisitos antes da implementação.

Mudanças nos requisitos

- Devido a complexidade do *software*, requisitos podem ser modificados.
- A mudança nos requisitos é geralmente custosa, pois pode demandar retrabalho.
- Há dois modos comuns de se lidar com mudanças de requisitos:
  - prototipação;
  - entrega incremental;

Prototipação

- O protótipo pode ser tanto um *software* quanto qualquer outra ferramenta demonstrativa.
- Geralmente é utilizado na engenharia de requisitos para elucidar e validar os requisitos.
- Pode ser utilizado também na fase de desenvolvimento para testar possíveis soluções de problemas complexos.
- Também é uma técnica bastante utilizada para o desenvolvimento de *interfaces*.
- Tem como vantagem poder ser associada a outros modelos, servindo como ferramenta de apoio.

Prototipação

- Tenha-se um *software* a ser desenvolvido em que há determinada necessidade de performance no acesso aos dados.
- Pode-se criar um ou mais protótipos rapidamente a fim de testar se um SGBD ou modelagem consegue atender a tal demanda.

Prototipação

- Todo protótipo deve possuir um objetivo e o seus componentes definidos.
- Coisas como tratamentos de erros podem ser ignoradas caso não façam parte do objetivo.
- Os protótipos devem ser descartáveis, pois seu objetivo não é implementar a funcionalidade totalmente.
- Nem sempre os protótipos atendem a padrões de qualidade ou podem ser integrados ao sistema.
- De modo algum pode ser tomado como uma versão para entrega.

Entrega incremental

- A entrega incremental é ligada ao desenvolvimento incremental.
- Nela o cliente identifica as funcionalidades e as priorizam.
- As funcionalidades mais importantes são entregues primeiro.
- Assim que é entregue, o *software* já é implanetado no ambiente de produção.

Entrega incremental

- Esta abordagem pode acabar motivando clientes e desenvolvedores porque cada iteração permite que ambos visualizem com clareza o andamento do processo de desenvolvimento.
- O *software*, logo ao fim da primeira iteração, está gerando retorno.
- Realiza número maior de testes sobre as funcionalidades mais importantes, pois elas estarão em mais etapas de validação.
- Pode diminuir o retrabalho devido a mudanças.
- Permite a todos os envolvidos entenderem melhor o *software*.

Entrega incremental

- Nem sempre é possível implementar a entrega incremental. Alguns contratos exigem que todos os requisitos sejam detalhados no próprio contrato.
- Quando o sistema a ser desenvolvido substitui outro, a entrega incremental pode não funcionar, já que não conseguirá substituir o *software* anterior de maneira completa.

Referências

- Sommerville, Ian. Software Engineering - Global Edition. 10ed. 2016. Pearson Education.