MODELOS DE PROCESSO DE SOFTWARE E TRATAMENTO DE MUDANÇAS NO SISTEMA

1) Modelos de Processo de Software

Nesta aula, serão tratados modelos de processo de *software*. Esses modelos são utilizados no desenvolvimento de sistemas de maneira elaborada e sistematizada. Eles possuem etapas bem definidas, com características próprias, sendo que alguns deles apresentam pontos comuns em relação aos demais. O uso desses modelos se justifica pois, em muita das vezes, o sistema a ser desenvolvido pode estar definido de forma caótica e as etapas a serem seguidas nesses modelos norteiam as atividades necessárias para o desenvolvimento de *software*. Vale ressaltar que esses modelos são descrições abstratas e genéricas, não descrevendo, de forma específica, como devem ser realizadas as etapas de desenvolvimento do sistema.

Existem diversos modelos de processo. A seguir, serão descritos o Modelo em Cascata, o Modelo em V e o Desenvolvimento Voltado ao Reuso e o Desenvolvimento Incremental. Para maiores detalhes, consulte Pressman (2010) e Sommerville (2011). Grande parte das descrições apresentadas neste trabalho estão baseadas nesses autores.

Nas subseções 1.1, 1.2, 1.3, 1.4. e 1.5, são apresentados, respectivamente, o Modelo em Cascata, o Modelo em V, o Desenvolvimento Voltado ao Reuso, o Desenvolvimento Incremental e o RUP.

1.1) Modelo em Cascata

O Modelo em Cascata, também conhecido como ciclo de vida clássico, é o modelo de processo de *software* mais antigo na Engenharia de *Software*. Nesse modelo, utiliza-se uma abordagem sistemática para a criação de um sistema, de forma que o desenvolvimento de *software* ocorra de maneira sequencial. Sendo assim, uma nova fase no desenvolvimento do sistema deve ser iniciada somente quando a fase que o antecede imediatamente esteja finalizada.

Esse modelo é apropriado para sistemas grandes, cujos requisitos estejam bem definidos. Conforme descrito por Sommerville (2011), esse modelo é composto pelas seguintes fases: i) Definição de requisitos; ii) Projeto de Sistema e Software; iii) Implementação e Teste de Unidade; iv) Integração e Teste de Sistema; e v) Integração e Teste de Sistema. Na Figura 1, podem ser observadas as etapas necessárias para o desenvolvimento de um sistema utilizando o Modelo em Cascata.

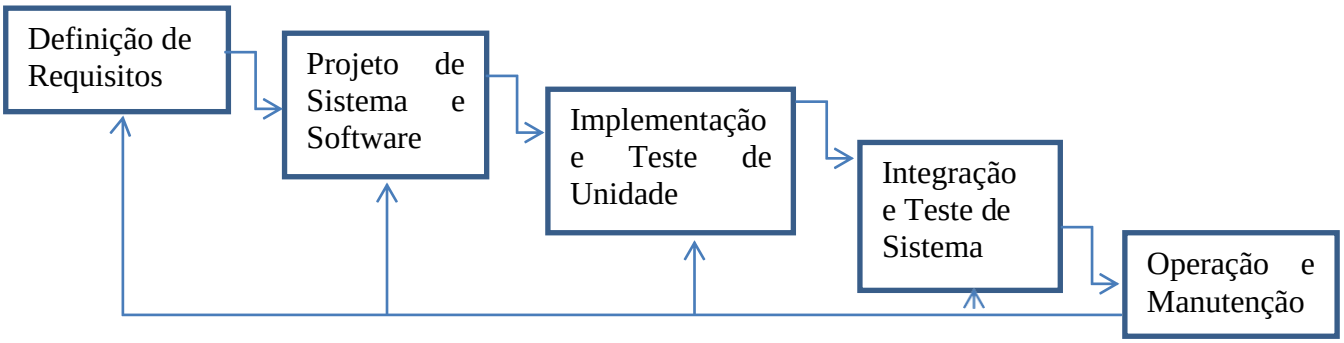

Figura 1: Passos utilizados no Modelo em Cascata. As fases são apresentadas conforme apresentado em Sommervile (2011)

De acordo com Sommerville (2011), os estágios do modelo em cascata podem ser descritos como segue:

1. Na fase de **Análise e definição de requisitos**, os serviços que deverão ser fornecidos pelo sistema devem ser estabelecidos. Essa definição de funcionalidades prestadas ocorre pelo uso de técnicas de levantamento de requisitos. Exemplos de técnicas incluem entrevista, questionário e *brainstorming*. Essas técnicas serão apresentadas em detalhes na próxima etapa desta disciplina. Nessa fase, ocorre uma especificação das funcionalidades esperadas para o sistema.
2. Na fase de **Projeto de sistema e software**, define-se uma arquitetura geral para o sistema criado. O projeto de *software* envolve a identificação e a descrição das abstrações fundamentais para a criação do sistema de software e de seus relacionamentos.
3. No estágio de **Implementação e teste unitário**, são implementadas as funcionalidades do sistema. Nos testes unitários, unidades de código devem ser testadas e deve-se verificar se as mesmas atendem às especificações do sistema.
4. Na fase de **Integração e teste de sistema**, unidades de códigos implementadas na fase 3 são integradas e são realizados testes após essa integração. Esses testes devem ser realizados para verificar se o *software* gerado atende às especificações dos usuários. Após essa etapa, o sistema é entregue ao cliente.
5. No estágio de **Operação e manutenção,** o sistema é instalado e colocado em uso. Nessa etapa, podem surgir erros que não foram detectados durante os testes realizados previamente, em estágios iniciais do ciclo de vida. Além das melhorias do sistema, pode ocorrer uma ampliação dos serviços fornecidos pelo mesmo. Em geral, essa é a fase que demanda mais tempo e recursos financeiros (SOMMERVILLE, 2011).

No desenvolvimento de muitos sistemas, há dificuldade para que o cliente estabeleça, *a priori*, todas as suas necessidades. Além disso, os requisitos podem estar incompletos ou inconsistentes. Por esse motivo, o Modelo em Cascata não é o mais indicado para o desenvolvimento de todo tipo de sistema.

No Modelo em Cascata, as etapas são definidas de maneira rígida e a correção de erros ou a alteração nos requisitos após o início de sua construção pode ser dispendiosa. Sendo assim, ao utilizar esse modelo, há dificuldade de realizar mudanças no sistemas após o início de seu desenvolvimento. Note também que a correção e atualização do *software*, ao utilizar esse modelo, pode requerer a repetição de alguns ou de todos os estágios anteriores do processo, ocorrendo, dessa maneira, um retrabalho para o desenvolvimento do sistema.

1.2. Modelo em V

O Modelo em V representa uma variação para o Modelo em Cascata. Essa variação ocorre à medida em que é realizada uma conexão entre os lados direito e esquerdo do modelo. Essa ligação entre os lados ocorre para que, problemas que sejam identificados em uma atividade de teste, atividades essas que estão identificadas no lado direito do diagrama, possam ser associados ao problema correspondente no lado esquerdo. Dessa maneira, após a correção dos erros, as fases subsequentes são executadas novamente, para a continuidade no processo de desenvolvimento de software.

Note que, nesse modelo, os problemas que surgem em fases prévias à finalização do *software*, podem vir a ser corrigidas assim que detectadas. Assim, em muitos dos sistemas, ao utilizar o modelo em V, os problemas são detectados em fases antecipadas, se comparado ao modelo em cascata.

Conforme descrito em Zoucas (2010), as etapas que compõem esse modelo são: i) Definição de Requisitos; ii) Projeto da Arquitetura (versão preliminar); iii) Projeto da Arquitetura (versão detalhada); iv) Implementação; v) Testes de Unidade, em que partes do sistema são testadas, e de Sistema, em que suas partes constituintes são integradas; vi) Teste de Sistema, após sua finalização; e vii) Teste de aceitação, que ocorre após sua implantação. A descrição aprofundada para essas etapas segue conforme a apresentado na subseção 1.1 deste trabalho, para o modelo em cascata. Na Figura 2, pode-se observar as etapas necessárias para o desenvolvimento de um sistema utilizando o modelo em V.

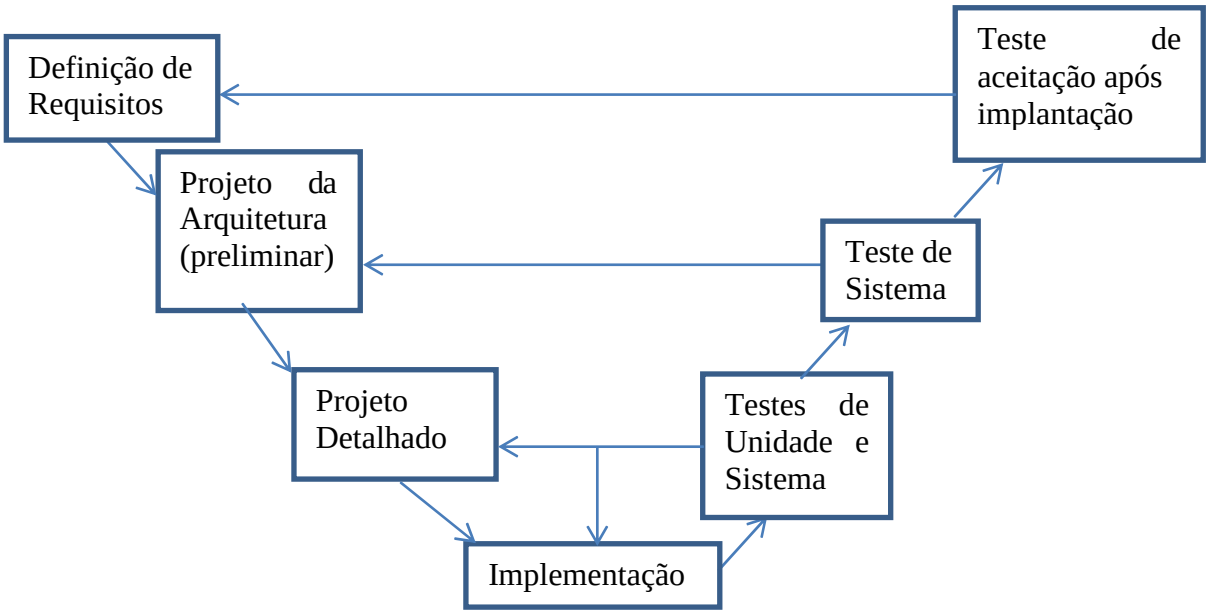

Figura 2: Modelo em V. Adaptado de Zoucas (2010).

1.3) Desenvolvimento voltado ao reuso

Durante o desenvolvimento de um *software*, existe a possibilidade de se reutilizar o sistema, ou partes do mesmo, para o desenvolvimento de outros sistemas. Em alguns casos, as modificações são pontuais para atender diferentes necessidades dos clientes.

No desenvolvimento voltado ao reuso, utiliza-se uma ampla base de componentes de *software* que são reutilizáveis. Dessa maneira, reutilizar trechos de código, em geral, possibilita a criação de sistemas de maneira mais eficiente. Após a seleção de trechos de códigos reutilizáveis, eles podem ser integrados entre si. Note que o reuso de código de maneira informal ocorre independentemente do modelo de projeto utilizado.

Como exemplo, funções como as de cadastro e atualização de clientes e de funcionários e de controle de estoque, são funções comuns a diversos sistemas. Na Figura 3, pode-se observar o esquema geral empregado para o desenvolvimento voltado ao reuso. As etapas que o compõem são: i) Especificação de Requisitos; ii) Análise de Componentes; iii) Modificação de Requisitos; iv) Projeto de sistema com reuso; v) Desenvolvimento e Integração; e vi) Validação de Sistema. Essas etapas são descritas a seguir.

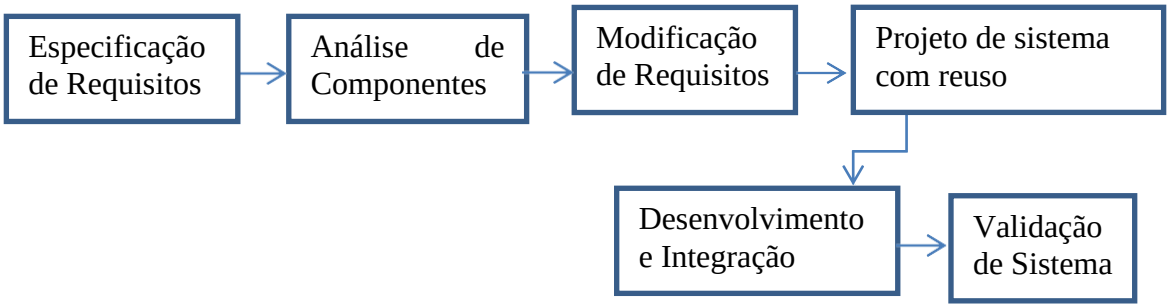

Figura 3: Desenvolvimento Voltado ao Reuso (Sommerville, 2011)

Os estágios utilizados no desenvolvimento voltado ao reuso, conforme explicado em Sommerville (2011), são:

1. Inicialmente, ocorre a **Especificação de requisitos** do sistema. Essa fase é comum aos métodos descritos previamente.
2. Na fase de **Análise de componentes**, ocorre uma busca de componentes que podem ser utilizados no sistema especificado. Mesmo que os componentes não forneçam uma correspondência exata, algumas de suas funcionalidades podem ser reutilizadas.
3. Na fase de **Modificação de Requisitos**, ocorre uma alteração nos requisitos do novo sistema, de forma que os mesmos reflitam os conteúdos disponíveis nos componentes já implementados.
4. Na fase de **Projeto de Sistema com Reuso**, com base nos componentes a serem reutilizados, o sistema a ser criado é projetado. Note que a implementação de novas partes de código pode ser necessária.
5. Na fase de **Desenvolvimento e Integração**, componentes selecionados na etapa 2 são integrados com novos softwares que não haviam foram implementados previamente, gerando um novo sistema.
6. Na fase de **Validação de sistema**, verificam-se e corrigem-se os eventuais erros que venham a surgir.

Ao utilizar o desenvolvimento voltado ao reuso para a criação de sistemas, ocorre uma redução na quantidade de código a ser desenvolvido. Porém, em geral, adequações no sistema são inevitáveis, uma vez que o sistema desenvolvido deve refletir as necessidades do cliente (SOMMERVILLE, 2011).

1.4) Desenvolvimento Incremental

No desenvolvimento incremental ocorre uma implementação inicial do sistema. Em seguida, o usuário verifica o que foi realizado e ocorre um aprimoramento do sistema, corrigindo e/ou inserindo novas funcionalidades. As atividades de especificação,

desenvolvimento e validação ocorrem de maneira simultânea, para que haja um retorno imediato sobre o que foi desenvolvido. O desenvolvimento incremental pode ser observado na Figura 4.

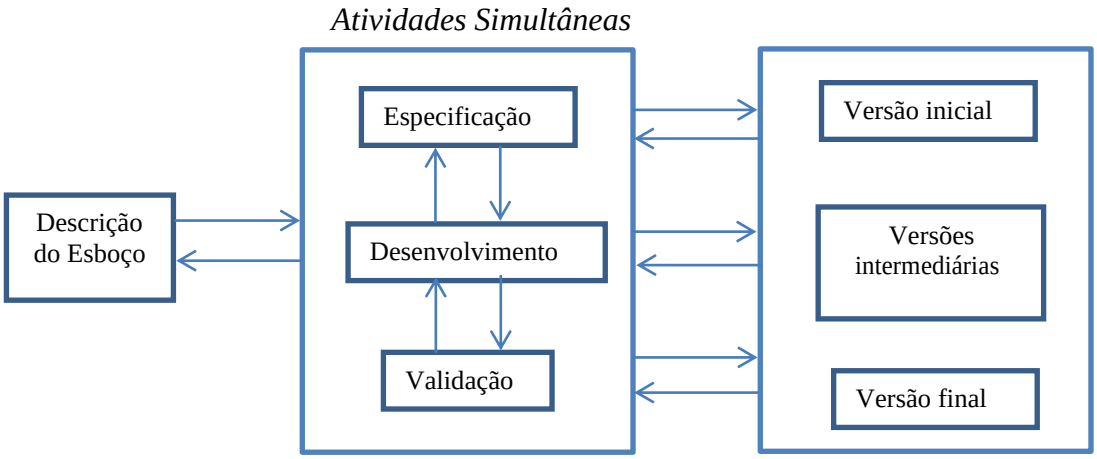

Figura 4 – Desenvolvimento Incremental (Sommerville, 2011)

Note também na Figura 4 que a especificação do sistema pode ser realizada de maneira gradativa. Isso é, à medida que for necessário, novas funcionalidades podem ser inseridas no sistema. Segundo Sommerville (2011), o desenvolvimento incremental, atualmente, é a abordagem mais comum para o desenvolvimento de sistemas aplicativos.

O desenvolvimento incremental apresenta três vantagens de maior relevância quando comparado ao Modelo em Cascata. Conforme apontado por Sommerville (2011), essas vantagens são:

1. Quando necessário, alterações no sistema utilizando o modelo incremental tendem a ser mais facilmente realizáveis, sendo que a análise e a documentação podem ser refeitas com menor esforço que o necessário no modelo em cascata.
2. Os retornos do cliente sobre o sistema gerado podem ser dados mais rapidamente, quando comparado ao modelo cascata. Isso porque os clientes podem comentar as demonstrações de *software* têm acesso ao que foi implementado.
3. O cliente recebe o sistema mais rapidamente, mesmo que nem todas as funcionalidades estejam incluídas.

Porém, ao se utilizar o modelo incremental, alguns problemas podem surgir. Dentre eles, destaca-se que não há uma visualização de todo o processo de desenvolvimento do sistema. Isso porque, como ocorre uma frequente variação nos requisitos do sistema, torna-se inviável realizar a documentação de cada uma das versões do sistema. No modelo em cascata, por exemplo, como o levantamento de requisitos ocorre no início do desenvolvimento do sistema, esse problema não ocorre.

Além disso, por conta das constantes mudanças na fase de levantamento de requisitos,

os sistemas são, frequentemente, mal estruturados, apresentando uma estrutura que tende a corromper-se. Ao longo do tempo, novas incorporações de funcionalidades no sistema tendem a ser custosas, devido a ausência de documentação que pode ocorrer.

1.5. *Rational Unified Process* (RUP)

No modelo de desenvolvimento de software RUP, há uma abordagem disciplinada para a criação do sistema, de forma que tarefas e responsabilidades possuam prazos definidos previamente. Essas fases são: concepção, elaboração, construção e transição. Para cada fase, são reservados os seguintes espaços de tempo: a fase de concepção toma 10%, a de elaboração recebe cerca de 30%, a de construção, 50% e a de fase de transição, 10%. Essas fases serão descritas abaixo.

Esse modelo é aplicável, preferencialmente, a grandes equipes de desenvolvimento e a grandes projetos. Porém, pode ser adaptado para quaisquer projetos.

O processo de desenvolvimento do sistema ocorre de maneira incremental, envolvendo a integração contínua do sistema, para produzir novas versões do sistema. A integração é feita passo a passo no processo de desenvolvimento. Cada uma das fases representam marcos bem definidos, cujos objetivos serão atingidos através da execução de uma ou mais iterações.

O ciclo de vida consiste nas fases conforme descrição abaixo (THIRY, 2007):

1) **Concepção**

- Estabelecimento do escopo do projeto e dos critérios de aceitação;
- Identificação de funcionalidades do sistema e seleção de funcionalidades críticas;
- Estimativa de custo, de cronograma total do projeto e dos riscos potenciais.

2) **Elaboração**

- Garantir que a arquitetura do sistema, os requisitos e os planos estejam estáveis;
- Assegurar que os riscos sejam tratados de forma eficiente e atendam o custo e o cronograma;
- Demonstrar que a arquitetura suportará os requisitos no custo e no cronograma, de forma que os mesmos sejam avaliados em um protótipo.

3) Construção

- Alcançar versões úteis, de forma que o sistema esteja de acordo com as expectativas definidas;
- Completar a análise, *design*, implementação e teste das funcionalidades;
- Certificar que o sistema está pronto para ser utilizado no ambiente do cliente.

4) Transição

- Nessa fase, ocorre o repasse do sistema para o cliente e verifica-se se o sistema foi aceito no ambiente do usuário final.

2) Tratamento de mudanças que podem vir a ocorrer no sistema

Em geral, os sistemas deverão passar por alterações, devido às mudança nos requisitos, à implantação de novas tecnologias ou à necessidade de aumento das funcionalidades do sistema. Segundo Sommerville (2011), qualquer que seja o modelo do *software*, é essencial que o mesmo possa acomodar mudanças quando estiver em desenvolvimento, sem que haja a necessidade de criar um novo projeto. Segundo o mesmo autor, duas abordagens podem ser adotadas para a redução de custos de retrabalho:

1. Prevenção de mudanças: o processo de *software* deve prever as possíveis mudanças antes que seja necessário o retrabalho.

2. Tolerância a mudanças: o processo de desenvolvimento de *software* é projetado de forma a acomodar mudanças com um baixo custo.

Esses modelos são indicados em projetos em que ocorrem mudanças frequentes nos requisitos. Para isso, a especificação é realizada conjuntamente com o sistema que está sendo construído.

A seguir, duas maneiras são descritas para lidar com essas mudanças. Na subseção 2.1, é apresentada a prototipação do sistema. Já na subseção 2.2, é apresentada a entrega incremental.

2.1. Prototipação do Sistema

Na prototipação do sistema, é criada uma versão contendo o sistema, ou parte dele, para que o cliente verifique se suas necessidades estão atendidas e também para ajudar sobre as tomadas de decisão sobre a criação do sistema. Esse método permite que os usuários experimentarem o sistema antes da entrega. Após a avaliação, o cliente retorna um *feedback* sobre o protótipo gerado.

Segundo Sommerville (2011), o número de propostas de mudanças de requisitos a ser

feito após a entrega é, portanto, suscetível de ser reduzido , quando se realiza a construção de um protótipo. Para realizar a prototipação de um sistema, pode-se utilizar, por exemplo:

- Rascunho;
- Apresentação de slides;
- Implementação simples do *software*, etc.

Na Figura 5, pode ser observado um exemplo de protótipo.

Figura 5: Exemplo de Protótipo. Fonte: Linha de Código (2017)

Dentre os tipos possíveis de geração de protótipos, destacam-se:

1) **Prototipação evolucionária:** o início do processo ocorre com um sistema relativamente simples, que é alterado à medida que são descobertos novos requisitos. O protótipo gerado não é descartado.
2) **Prototipação descartável:** em que ocorre a escrita, a avaliação e a modificação do protótipo. A avaliação do protótipo é incluída no documento de requisitos e o protótipo é descartado.

O processo de geração de protótipos colabora também na fase de levantamento de requisitos. Por esse motivo, detalhes sobre prototipação serão apresentados de maneira mais detalhada na próxima aula desta disciplina.

2.2. Entrega Incremental

Na entrega incremental, as atividades do processo de desenvolvimento são realizadas de forma linear, para uma ou mais unidades do *software.* Em cada uma dessas sequências, são

gerados novos incrementos para o sistema, incluindo as funcionalidades requeridas e/ou alterações sugeridas pelo cliente.

Neste modelo, é permitido que mudanças sejam incorporadas nos incrementos posteriores com um custo relativamente baixo, mas mudanças nos requisitos no incremento de *software* atual não podem são aceitas. Após a conclusão de um incremento, o mesmo é entregue e pode ser colocado em operação. Na entrega incremental, a construção do sistema ocorre em etapas, conforme pode ser observado na Figura 6. As fases que o compõem são: i) Comunicação; ii) Planejamento; iii) Modelagem; iv) Construção; e v) Emprego. Essas etapas são descritas adiante.

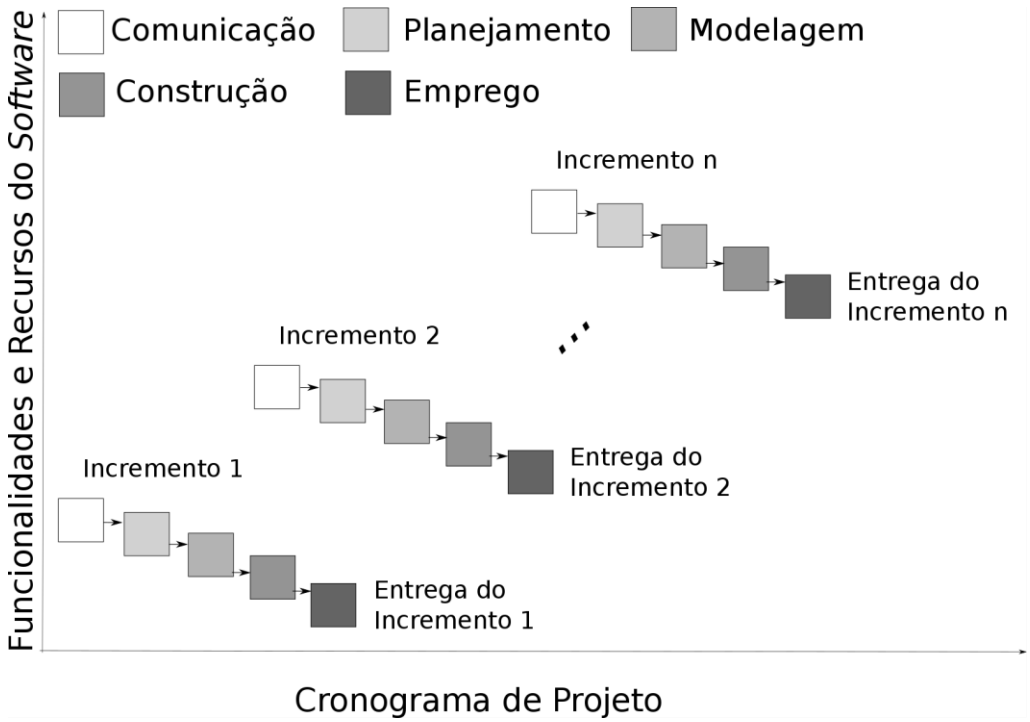

Figura 6 – Entrega Incremental. Baseado em Pressman (2010)

As fases ocorrem conforme descrito abaixo:

1) Na fase de **Comunicação**, ocorre o levantamento de requisitos para o incremento atual do sistema;

2) Na etapa de **Planejamento**, ocorre um projeto do incremento a ser desenvolvido. Descrevem-se, por exemplo, os riscos do *software*, os recursos necessários e é estabelecido um cronograma;

3) Na fase de **Modelagem,** ocorre a criação de um modelo do incremento a ser desenvolvido, de forma que desenvolvedor e cliente entendam os requisitos do *software*;

4) Na fase de **Construção,** o incremento de software é gerado e realizam-se testes nesses incrementos;

5) Na etapa de **Emprego**, o *software* é entregue ao cliente, que o avalia e fornece um *feedback* sobre o incremento desenvolvido.

Um exemplo de incremento de sistema ao utilizar a entrega incremental pode ser observado a seguir.

Exemplo: Desenvolvimento de um sistema de controle de registros acadêmicos
Incremento 1: *Cadastro de professores*
Incremento 2: *Cadastro de alunos*
Incremento 3: *Cadastro de notas*

2.2.1. Vantagens de se utilizar a entrega incremental

Neste modelo, novas funcionalidades do *software* são geradas após a finalização do desenvolvimento do incremento. Dessa maneira, se comparado a modelos como o cascata e o modelo em V, um conjunto de funcionalidades do sistema estará disponível mais cedo. Esses incrementos poderão ser utilizados para a verificação de erros de desenvolvimento. Com isso, há um risco menor de falha no projeto em geral.

Note também que, ao utilizar o desenvolvimento incremental, os serviços do sistema de alta prioridade são entregues primeiro, pois os mesmos são construídos nos incrementos iniciais. Por esse motivo, esses serviços de alta prioridade tendem a receber a maioria dos testes, o que evita a possibilidade de falhas graves em funcionalidades primordiais para o mesmo.

2.2.2. Desvantagens de se utilizar a entrega incremental

O uso desse modelo pode apresentar algumas desvantagens. Dentre elas, pode-se citar a dificuldade de mapear os requisitos em um tamanho adequado, sendo difícil prever o escopo de tempo necessário para que os incrementos sejam realizados.

Além disso, os incrementos devem ser pequenos. Dessa maneira, pode ser difícil identificar recursos necessários para realização dos vários incrementos do sistema, pois a descrição e/ou implementação dos mesmos pode estar incompleta ou inexistente.

Referências Bibliográficas

LINHA DE CÓDIGO. **Criando Protótipos com o SketchFlow Parte 1, 2017.** Disponível em http://www.linhadecodigo.com.br/artigo/2941/criando-prototipos-com-o-sketchflow-parte-1.aspx. Acessado em 20 de julho de 2017.

PRESSMAN, R. S. Engenharia de software . 7. ed. São Paulo: McGraw-Hill, 2010.

SOMMERVILLE, I. Engenharia de Software. São Paulo: Pearson Prentice Hall, 9ª ed., 2011.

THIRY, M. **Engenharia de Software: Introdução**, UNIVALI, 2007.

ZOUCAS, A. C. Análise de Sistemas. Florianópolis, 2010.