ETAPAS ESSENCIAIS DE UM PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O engenheiro de software deverá definir qual o processo de desenvolvimento a ser aplicado em um determinado projeto de software como especificação de requisito não funcional. Inicialmente, terá que identificar quais as atividades que irão compor o desejado processo e, em seguida, definir o sequenciamento das referidas atividades, ou seja, o fluxo do processo.

As atividades típicas que compõem o processo de desenvolvimento de software estão ilustradas na figura 4. O objetivo é ilustrar as atividades mais comuns que compõem os processos de desenvolvimento de software, ou seja, qualquer processo deverá possuir as referidas atividades.

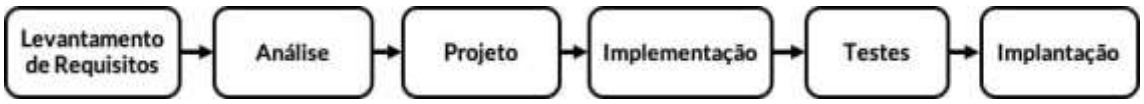

Figura 4 - Atividades típicas de um processo de desenvolvimento de software.

Vamos agora descrever cada uma das atividades comumente previstas em um processo de desenvolvimento de software.

Etapa de Levantamento de Requisitos

A primeira etapa, também denominada de Elicitação de Requisitos, inclui o primeiro desafio do engenheiro de software, entender o problema!

Imagine você como engenheiro de software, projetando um software para o mercado financeiro! O desafio impressiona? Talvez você concorde comigo – “sim”!

Realmente, o desafio é grande, pois o referido engenheiro, normalmente, não entende do ambiente de negócio que o software será implementado. Portanto, terá que se comunicar com diferentes usuários que, muitas vezes, entendem de somente parte do problema; para tal, deverá utilizar diferentes técnicas de levantamento de requisitos, tais como, entrevistas, questionários, leituras de documentos, etnografia e outros.

Nesta etapa, são identificados os requisitos que serão implementados no software projetado. O grande desafio é que o engenheiro de software tenha o mesmo entendimento do negócio que os usuários.

Neste momento, precisamos apresentar a conceituação de requisito:

Os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. Esses requisitos refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema.

SOMMERVILLE, 2007.

A partir dessa conceituação, identificamos a tipificação de requisitos comumente utilizada: requisitos funcionais, não funcionais e de domínio.

REQUISITOS FUNCIONAIS
Estão relacionados com os serviços fornecidos pelo sistema, ou seja, as funcionalidades que estarão disponíveis no software, tal como, a geração de um histórico escolar em um sistema de gestão acadêmico ou geração de uma nota fiscal em um sistema de vendas.

REQUISITOS NÃO FUNCIONAIS
Incluem as restrições operacionais impostas ao software, tais como, o sistema gerenciador de banco de dados, a linguagem de programação, legislação pertinente à *compliance*, entre outros, bem como os requisitos de qualidade, tais como: confiabilidade, manutenibilidade, usabilidade e outros.

REQUISITOS DE DOMÍNIO
Também são conhecidos como “regras de negócio”, que, normalmente, apresentam-se como restrições ao requisito funcional. Como exemplo, temos o cálculo da média para aprovação em uma determinada disciplina, a contagem de pontuação de multas para computo da perda de uma carteira de motorista ou o cálculo dos impostos quando da geração de uma nota fiscal. O não cumprimento de um requisito de domínio pode comprometer o uso do sistema. Cabe destacar que Sommerville (2007) apresenta uma classificação relativa aos níveis de abstração aplicados na descrição dos requisitos, utilizando o termo **requisitos de usuários**, para especificação com alto nível de abstração, e **requisitos de sistema**, para especificação com descrições detalhadas, ou seja, com baixo nível de abstração. Realizada essa primeira classificação, os requisitos de sistema são tipificados em funcionais, não funcionais e de domínio, como anteriormente apresentados.

Muitos estudos destacam a importância desta etapa, pois o mau entendimento de um requisito é propagado nas próximas etapas no processo, ilustrado na figura 4, refletindo de forma negativa no produto software.

Quando da especificação de requisitos, o engenheiro de software firma um **contrato** com os usuários, de modo a definir qual **produto** de software será entregue. A participação dos usuários envolvidos, ou clientes, deve permitir feedbacks sobre defeitos na especificação, evitando a propagação dos referidos defeitos.

Nesta etapa do PDS, normalmente, é realizada uma entrega, ou uma especificação, denominada de Documento de Requisitos, sendo determinado o Escopo do projeto.

A partir desse documento, inicia-se a rastreabilidade dos requisitos, ou seja, a relação com os produtos gerados, e.g. modelos, a partir dos mesmos.

Exemplo

Na etapa de Análise, é gerado o modelo de casos de uso e o modelo de classes; na etapa de Projeto, o modelo de interação; e na etapa de Implementação, a codificação. A rastreabilidade de requisitos garante que as especificações geradas até a codificação estejam de acordo com a documentação de requisitos.

Etapa de Análise

Ainda no contexto da construção de uma casa, quais seriam os requisitos iniciais para que essa construção ocorresse?

Poderíamos imaginar uma descrição especificando que você quer uma casa com uma sala, três quartos, uma suíte, cozinha, piscina, churrasqueira e com aproximadamente 120 m2. Textualmente, temos um documento de requisitos.

E agora, o que fazer com esse documento?

Podemos contratar um arquiteto para desenhar uma planta baixa que atenda aos requisitos descritos. Essa planta é o que chamamos de **modelo**, que representa parte da solução do problema “construir a sua casa”.

A engenharia tem como boa prática a construção de modelos que permitem uma melhor tratativa da complexidade em função de abstrações aplicadas, e.g., o diagrama de casos de uso enfatiza a abstração funcional e o diagrama de sequência, os aspectos dinâmicos do sistema, ou seja, a comunicação entre objetos na realização de um caso de uso. O modelo permite, também, a comunicação entre as partes interessadas do projeto, pois podemos considerar que um diagrama de caso de uso permite uma comunicação, por exemplo, entre analistas e usuários ou entre analistas e gerentes de projeto. Os modelos permitem, ainda, uma economicidade no projeto, pois uma correção em um

modelo de classes na etapa de análise por um erro no entendimento de um determinado requisito evita a propagação desse defeito no código em produção, manutenção essa que seria bem mais custosa. Uma última colocação: os modelos determinam a forma da solução do problema, ou seja, se o diagrama de componentes estabelece três camadas de software, o software implantado deverá refletir a referida especificação.

Na etapa de Análise, as especificações contidas no Documento de Requisitos são convertidas em modelos de análise que incluem artefatos gráficos e textuais. Como exemplo, os requisitos funcionais evoluem para uma especificação gráfica denominada Modelo de Casos de Uso, sendo este composto por diagramas de caso de uso, artefatos gráficos, e descrições de caso de uso, artefatos textuais. Importante destacar que uma descrição de caso de uso permite identificar os diferentes cenários de utilização do referido caso. A figura 5 ilustra um diagrama de casos de uso.

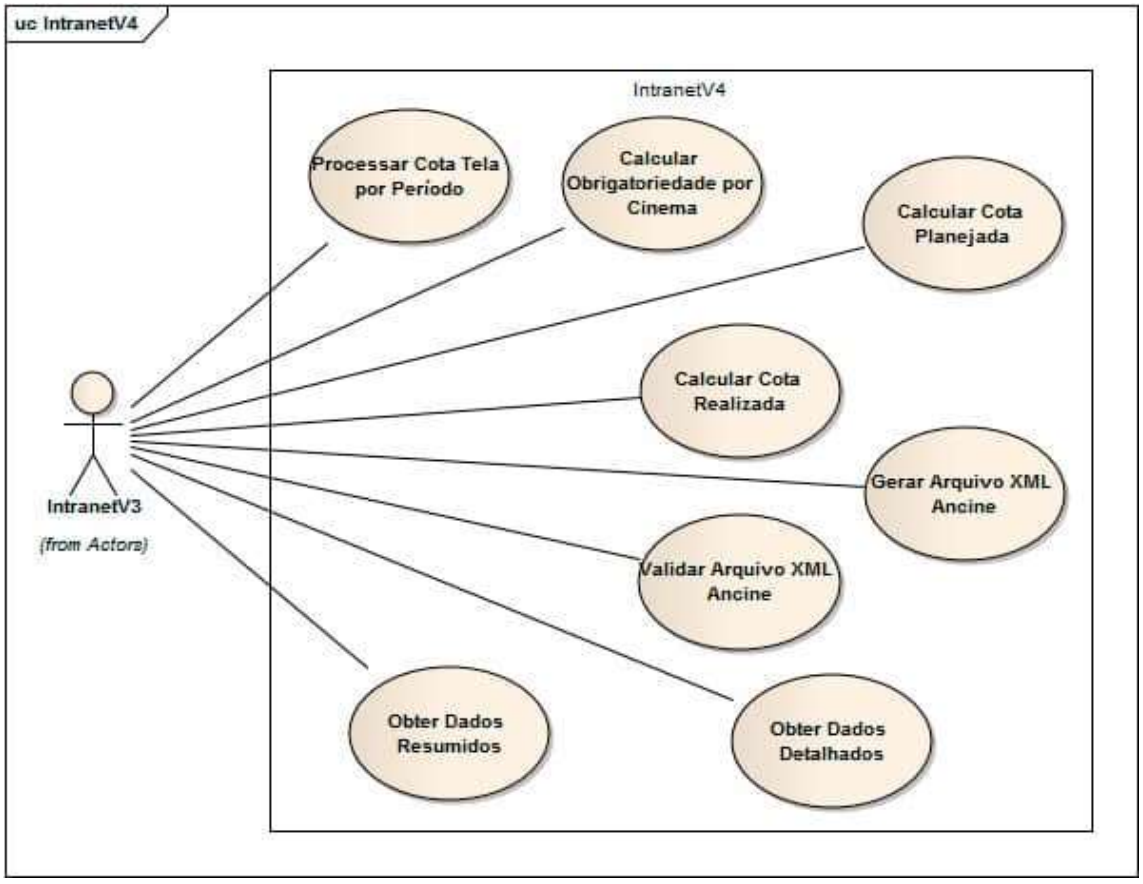

Figura 5 - Exemplo de Diagrama de Casos de Uso.

Nesta etapa, o engenheiro de software aplica um alto nível de abstração de modo a definir “O QUE” o sistema deverá implementar, ou seja, nenhum tipo de restrição tecnológica é considerado, e.g., como ocorre a comunicação entre os objetos que permitirão implementar o caso de uso “Agendar consulta”.

A entrega da referida etapa inclui os modelos denominados “modelos de análise” com alto nível de abstração.

O entendimento do problema modelado está correto?

Em função desse questionamento, o engenheiro de software necessita VALIDAR seus modelos com os usuários, pois, afinal, são estes que entendem do problema. A validação por parte do usuário garante ao engenheiro que o entendimento da solução do problema está correto e, de acordo com as suas expectativas, i.e., de acordo com os requisitos registrados no documento de requisitos.

Os modelos estão corretos e balanceados?

A verificação é uma atividade técnica do engenheiro de software que permite a verificação da correção de cada modelo e o balanceamento entre esses.

A figura 6 ilustra um diagrama de classes, sendo este um artefato do Modelo de Classes que permite identificar os objetos do domínio do problema que serão utilizados na implementação dos casos de uso. Cabe ao engenheiro de software, verificar a correção do modelo e a consistência com o Modelo de Casos de Uso.

Figura 6 - Exemplo de Diagrama de Classes.

Etapa de Projeto

A fase de Projeto permite os refinamentos dos modelos gerados na análise, bem como a construção de novos modelos gerando especificações com menor nível de abstração e que permitam definir “COMO” implementar a solução especificada, ou seja, ao final desta etapa, o nível de detalhamento da especificação permite a implementação da solução.

Dentre as principais atividades, destacamos:

- Refinamento do modelo de classes iniciado na análise.
- Construção do modelo de interação, que permite identificar a comunicação entre objetos que irão permitir a implementação das funcionalidades especificadas pelos casos de uso.
- Mapeamento objeto-relacional, permitindo a geração do modelo lógico do banco de dados, ou seja, identificação das tabelas necessárias ao sistema considerando um requisito não funcional que determine uma tecnologia relacional.
- Desenho dos componentes do sistema e dos nós computacionais necessários para a sua implantação, gerando modelos que determinam a arquitetura do sistema.

Etapa de Implementação

Nesta etapa, ocorre a codificação do software de acordo com os requisitos definidos na etapa de Projeto. Os padrões de projeto devem ser considerados por representarem melhores práticas de implementação do software.

Etapa de Testes

Nesta etapa, são aplicados os denominados testes de software ou testes de validação que permitem verificar se o produto certo está sendo construído.

Os testes validam os componentes individualmente bem como a integração entre eles. Em geral, esta etapa inicia com os testes unitários, em seguida, os

testes de integração e os testes de aceitação ou homologação. Ao final dessa sequência, ocorre a migração do sistema para o ambiente de produção.

Recomendação

Nesta etapa, é fundamental a geração de um Plano de Teste a partir dos casos de testes, que, por sua vez, estão vinculados aos cenários descritos na descrição de cada caso de uso.

Outro aspecto importante é a automação dos testes em função da provável repetição destes em um ciclo iterativo e incremental, onde o software é implementado em versões e, a cada nova versão, os testes anteriores necessitam ser refeitos.

Etapa de Implantação

Nesta etapa, o software é migrado para o ambiente de produção, de acordo com o aval da equipe de qualidade.

Esta etapa pode exigir a geração de manuais de utilização, a migração de dados do ambiente de homologação para o de produção, treinamento de usuários, ou até mesmo a implantação de um *call center* em caso de sistemas complexos assim o exigirem.

FLUXO DE PROCESSO

A especificação de um processo de desenvolvimento de software, lembrando ser este um requisito não funcional, requer a definição de quais atividades irão compor o respectivo processo e como as referidas atividades serão encadeadas, também denominada de Fluxo de Processo ou Ciclo de Vida.

FLUXO DE PROCESSO LINEAR

A figura 7 ilustra o denominado Fluxo de Processo Linear, onde as atividades são executadas em sequência, de modo que cada atividade é realizada por completo uma única vez.

Figura 7 - Fluxo de Processo Linear.

FLUXO DE PROCESSO ITERATIVO

A figura 8 ilustra o denominado Fluxo de Processo Iterativo, onde uma atividade ou um conjunto de atividades podem ser repetidas antes de prosseguir para a seguinte.

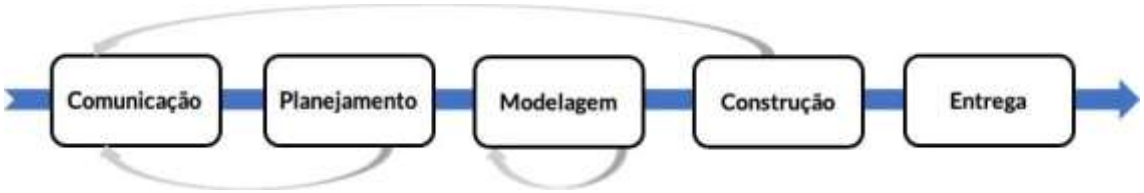

Figura 8 - Fluxo de Processo Iterativo.

FLUXO DE PROCESSO EVOLUCIONÁRIO

A figura 9 ilustra o denominado Fluxo de Processo Evolucionário, onde o sequenciamento de cada fluxo inclui todas as atividades, sendo que cada iteração completa gera uma nova versão do software, ou seja, o software agrega valor às suas funcionalidades a cada ciclo completo.

Figura 9 - Fluxo de Processo Evolucionário.

FLUXO DE PROCESSO PARALELO

A figura 10 ilustra o denominado Fluxo de Processo Paralelo, que permite a execução de uma ou mais atividades em paralelo.

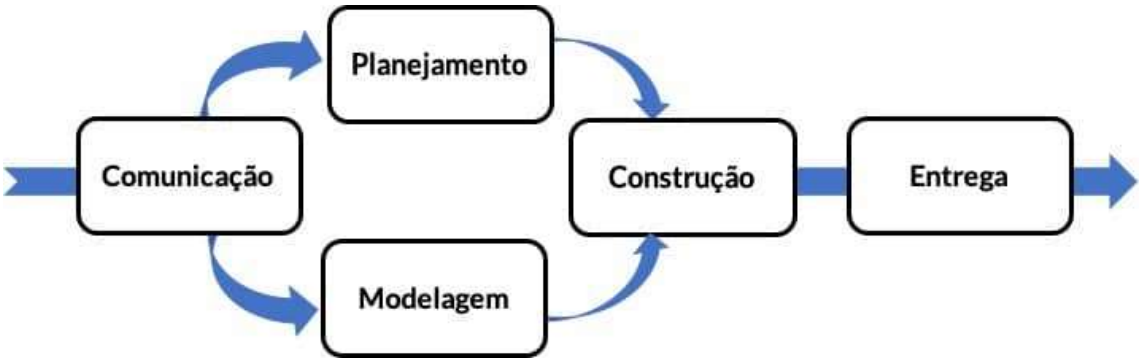

Figura 10 - Fluxo de Processo Paralelo.

MODELO DE CICLO DE VIDA ITERATIVO E INCREMENTAL

A figura 11 ilustra o Modelo de Ciclo de Vida Iterativo e Incremental, onde cada iteração inclui todas as atividades e ocorre o versionamento do produto software gerado. Este modelo corresponde ao Fluxo de Processo Evolucionário. A cada novo ciclo, um subconjunto de requisitos é considerado.

Figura 11 - Modelo de Ciclo de Vida Iterativo e Incremental.

RESUMINDO

Neste módulo, podemos avaliar a importância do processo de desenvolvimento de software. Cabe destacar que processo é a principal camada da Engenharia de Software.

Foram descritas as principais atividades que, genericamente, compõem um processo de desenvolvimento de software, que inclui: levantamento de requisitos, análise, projeto, implementação, testes e implantação.

Cabe ao engenheiro de software determinar as atividades que comporão a especificação de um processo de desenvolvimento de software, sendo o sequenciamento determinado pelo Fluxo de Processo, podendo ser Fluxo de Processo Linear, Fluxo de Processo Iterativo, Fluxo de Processo Evolucionário ou Fluxo de Processo Paralelo.

Lembre-se, não existe engenharia sem processo.