Análise e Modelagem de Sistemas

Software

Softwares são programas de computadores com uma documentação associada e [que] os produtos de software podem ser desenvolvidos para um determinado cliente ou para um mercado mais generalizado.

Pressman (2016) afirma que um software de computador é um produto que profissionais da área da Tecnologia da Informação (TI) desenvolvem e para o qual darão suporte a longo prazo. Além disso, abrange qualquer tipos de mídia eletrônica, consistindo na união de três elementos:

- **Instruções:** Quando executadas, fornecem os atributos e funções de desempenhos desejados pelos usuários.
- **Estrutura de Dados:** Possibilitam aos programadores manipular as informações de forma mais adequada conforme a necessidade da aplicação.
- **Documentação:** É toda a informação descritiva do software, a qual detalha a operação de uso dos programas, diagramas de funcionalidades, etc.

Evolução do Software

A medida que novas tecnologias se tornam populares, maiores são as necessidades da produção de software para atender as demandas da sociedade.

- **Na Década de 1990:** Uma empresa precisava de um software e (talvez) de um site para que ficasse conhecida na internet, o site era utilizado como um cartão de visitas.
- **Na Década de 2000:** A mesma empresa precisava de um software com acesso à internet para disponibilizar recursos para seus clientes.
- **A Partir de 2010:** A empresa precisava de um software, um site e um aplicativo para que seus serviços pudessem ser acessados por diversos dispositivos e para que o armazenamento da base de dados estivesse na nuvem.
- **A Partir de 2020:** A mesma empresa já está pensando em utilizar a inteligência artificial para melhorar seus processos gerenciais.

As mudanças sempre ocorrerão ao longo do tempo de criação e uso de um software: Durante o desenvolvimento, na fase da entrega e depois de entregue. Sempre há necessidade de ajustes e correções ou ainda pode ocorrer a necessidade de incluir novas funcionalidades ao software, as quais são, muitas vezes, requisitadas pelo cliente.

Além disso, o software passa por uma série de manutenções (isso acontece pois são realizadas novas solicitações do cliente) e, após realizar diversos ajustes (que podem levar a novos problemas), é possível que haja a necessidade de se criar um novo software.

Prática da Engenharia de Softwares

De acordo com Sommerville (2011), a prática da engenharia de software se faz necessária porque cada vez mais a sociedade (e os indivíduos que a compõem) estão dependentes dos sistemas de software e, neste contexto, devemos produzir softwares mais confiáveis e de forma mais econômica, e a longo prazo é mais barato utilizar as técnicas da engenharia de software para evitar mudanças

em algo que já tenha sido desenvolvido e principalmente que partes do software possam ser reutilizadas em outros sistemas.

Conforme Pressman (2016), a engenharia de software é uma tecnologia em quatro camadas que objetiva a disciplina, a adaptabilidade e a agilidade. As quatro camadas são:

- **Foco na Qualidade:** É o objetivo final de toda a engenharia de software, toda a gestão de qualidade (das demais camadas) deve ser fundamentada em um comprometimento organizacional com a qualidade total de todas as etapas de desenvolvimento.
- **Processo:** É a base da engenharia de software, o processo é a ligação entre as demais camadas, é o que mantém as camadas de tecnologia coesas, possibilitando o desenvolvimento de forma racional (e dentro dos prazos preestabelecidos) e é a base para o controle e gerenciamento de projetos de software. É nesta camada que são produzidos diversos artefatos (modelos, documentos, relatórios, formulários, etc.).
- **Métodos:** Fornecem as informações técnicas para o desenvolvimento do software e envolvem uma grande quantidade de tarefas (comunicação, análise de requisitos, modelagem de projetos, codificação, testes e manutenção do software).
- **Ferramentas:** Possibilitam um alicerce automatizado ou semiautomatizado para o processo e para os métodos, quando as ferramentas são integradas de modo que a informação criada possa ser usada por outra ferramenta, é estabelecido um suporte ao desenvolvimento de software, conhecido como engenharia de software auxiliado por computador.

A engenharia de software, segundo Sommerville (2011), preocupa-se com todos os aspectos de produção do software.

Enquanto isso Pressman (2016) enfatiza que essa área engloba processos, métodos e ferramentas que possibilitam a construção de sistemas, incorporando cinco atividades específicas nesses processos: Comunicação, planejamento, modelagem, construção e entrega.

Pressman (2016) destaca sete grandes categorias de softwares e desafia a constante evolução das práticas empregadas na engenharia de software:

- **Software de Sistema:** São conjuntos de programas com finalidade específica para auxiliarem outros programas, por exemplo: Compiladores, drivers, etc.
- **Software de Aplicação:** Programas independentes que têm a finalidade de resolver um problema específico de negócio, facilitando transações comerciais, decisões administrativas, e que podem ser: Planilhas de cálculos, editores de textos, software exclusivo da empresa (criado e desenvolvido para a empresa).
- **Software de Engenharia/Científico:** Programas que facilitam o entendimento e usam grandes quantidades de cálculos, geralmente utilizados em aplicações para meteorologia, astronomia, genética, etc.
- **Software Embarcado:** Programado de forma que só funciona para determinado produto (forno micro-ondas, geladeira, módulo de carros).
- **Software para Linha de Produtos:** Projetado para ser utilizado por diversos clientes (com ou sem adaptações), concentrando-se em nichos de mercado, por exemplo: Software contábil, software de controle de RH, etc.
- **Software de Aplicações Web/Aplicativos Móveis:** Foco em dispositivos com acesso à internet e voltados a navegadores e softwares residentes em dispositivos móveis: TV, celulares, tablets, etc.
- **Software de Inteligência Artificial:** Utiliza algoritmos sofisticados para analisar e solucionar problemas complexos, inclui as seguintes áreas: Robótica, reconhecimento de padrões (voz, imagens, sons), redes neurais, etc.

Softwares Legados

Segundo Pressman (2016), são softwares antigos que continuam sendo utilizados, apresentam baixa qualidade, muita demora e funcionalidades que não são mais utilizadas pela empresa ou que são muito defasadas. Um software legado pode ter uma documentação deficitária ou inexistentes, fazendo com que o processo de manutenção seja caro e muito demorado. O processo de evolução desse tipo de software é a criação de um novo com tecnologias mais recentes.

Aplicativos

Software projetado para funcionar em uma determinada plataforma móvel. Os aplicativos possuem uma interatividade maior com os usuários, apresentam recursos disponíveis pelo fornecedor da plataforma móvel (iOS, Android, etc.) e também recursos baseados na web, permitindo uma vasta opção de informação e compartilhamento. Uma vantagem dos aplicativos é o fornecimento de recursos de armazenamento persistente dentro da plataforma. Um aplicativo pode acessar diretamente ferramentas de hardware, disponíveis no dispositivo, trazendo novas opções de programação utilizando, por exemplo, a câmera ou o GPS do dispositivo.

Princípios da Análise de Sistemas

Os princípios da análise de sistemas fundamentam-se na necessidade de realizar estudos de processo para encontrar a melhor solução para a criação de um sistema. Conforme Roth, Dennis e Wixom (2014), a análise de sistemas baseia-se em métodos e técnicas de investigação e em especificação para encontrar a melhor solução para algum problema ou necessidade computacional de determinada área de negócio, a partir das funcionalidades levantadas pelo analista de sistemas.

Em uma visão mais generalizada das fases que envolvem os processos da análise de sistema, destacam-se, conforme Pressman (2005):

- **Análise:** Nesta fase são realizados estudos que objetivam a especificação do software, de modo a verificar a viabilidade (custo-benefício), definir as funcionalidades que o software deverá possuir e realizar o escopo, alocando recursos e realizando o orçamento do software. O resultado desta fase será utilizado nas próximas etapas.
- **Projeto:** Nesta etapa há uma preocupação com a definição lógica do software, são elaborados os layouts de telas e relatórios e são criados a estrutura do banco de dados e os diagramas gráficos para o desenvolvimento do software.
- **Implementação:** Nesta fase é realizada a codificação do software por meio de uma linguagem de programação (definida na fase de análise).
- **Testes:** Objetivando a procura de erros, nesta fase, são realizados procedimentos de testes que verificam as funcionalidades dos itens codificados.
- **Documentação:** Trata-se de documentar todos os processos (de todas as fases) e diagramas produzidos, são utilizados documentos padronizados (e personalizados por cada empresa de desenvolvimento), que servem como ferramenta de comunicação entre as pessoas envolvidas no desenvolvimento e também como parte de contrato entre as partes interessadas na produção do software.
- **Manutenção:** Esta fase consiste em fazer o acompanhamento do software após ser implantado e entrar em funcionamento (durante um período), visando a registrar e corrigir falhas, propor melhorias ou incluir novas funcionalidades.

As fases apontadas por Pressman (2005), devem apresentar um processo de homologação para validar os documentos gerados. O cliente deve estar ciente de que uma vez aprovada uma fase, caso seja alterada, haverá mudanças nos custos e implicará no dimensionamento do tempo (atrasos).

O Papel no Analista de Sistemas

O analista de sistemas é o profissional responsável por realizar atividades da análise de sistema como: Pesquisas, planejamentos, coordenação de equipes de desenvolvimento e recomendação de alternativas de software de acordo com as necessidades de desenvolvimento ou de solução para problemas de negócios.

Possui como tarefas a criação, a implementação e a implantação de um software, de maneira que deve, primeiro, descobrir o que um sistema deverá fazer e depois entender e avaliar as necessidades e expectativas de cada usuário do software, a fim de que estas sejam organizadas, especificadas e documentadas.

Cabe ao analista de sistemas colher informações com os usuários que utilizarão o software, interpretar essas informações e repassá-las aos programadores de forma técnica para que seja criado um software que atenda as expectativas do cliente e dos usuários.

O profissional desse ramo precisa conhecer um pouco de cada área de negócio e, caso não tenha o domínio necessário sobre algum tema, deve ter a proatividade de procurar o máximo de conhecimento sobre a área que o software irá abranger.

Uma característica importante do analista de sistemas é ter uma boa visão empresarial para ajudar nos processos gerenciais da produção do software. As habilidades desejáveis para um analista de sistema são: Conhecimento tecnológico atualizado, organização e método, visão gerencial, ótimo relacionamento interpessoal, entre outras.

Até aqui vimos que um software precisa evoluir para não ficar obsoleto e a necessidade de incluir práticas de engenharia de software e princípios da análise de sistemas no desenvolvimento de um software. Além disso, vimos o papel do analista de sistemas em todo esse processo.

Para colaborar na ampliação de seus conhecimentos a respeito do desenvolvimento de um software, o artigo apresentado a seguir trata de um mecanismo de visualização da evolução de software capaz de ser integrado em um processo de desenvolvimento distribuído, apresentando uma ferramenta por meio da qual desenvolvedores espalhados geograficamente podem compreender a evolução de seus projetos e identificar, através de uma visão global do projeto, diferentes características relevantes do projeto.

Processo de Software

O processo de software é uma abordagem adaptável que possibilita à equipe de desenvolvimento escolher os processos que melhor se enquadram na filosofia da empresa (de desenvolvimento) com o foco na qualidade do produto, no prazo de entrega e na redução de custos (Pressman, 2016).

É um conjunto de atividades e resultados que estão relacionados, que levam à produção e ao resultado de um software desenvolvido. Um processo de software pode conter diversas atividades que normalmente são: Especificação, projeto, implementação, validação, manutenção e evolução (Sommerville, 2011).

Um processo de software possui inúmeras entradas e saídas. O processo se constitui em uma série de atividades que serão executadas de forma padronizada, agrupadas em fases (essas atividades mudam conforme há a troca de fase), sendo que, em cada fase, serão definidos: As responsabilidades (quem fará o quê), prazos de entrega e como o objetivo será alcançado.

Estrutura de Processo Genérico de Softwares

O processo de software adotado em uma empresa pode ser completamente diferente de outra empresa, cada qual procura encontrar e estabelecer atividades que visam aumentar a qualidade e baixar o custo de produção do software produzido. Independente do modelo de processo de software adotado pela empresa de desenvolvimento, todos utilizam uma estrutura de processo genérico de software, com atividades preestabelecidas.

As atividades de um determinado processo de software constituem um conjunto mínimo para se obter um produto de software (o software finalizado e entregue ao cliente). Em um processo genérico de software, os processos podem ser diferentes, mas podemos identificar quatro atividades fundamentais em toda a produção de software, conforme Sommerville (2011):

- **Especificação:** Definição do que será desenvolvido, suas restrições e funcionalidades.
- **Projeto e Implementação:** Definição do que será desenvolvido, suas restrições e funcionalidades.
- **Validação:** Verificação se o que foi construído atende as solicitações do cliente.
- **Evolução:** Evolução do software para que acompanhe as alterações pelo cliente.

Cada atividade do processo genérico de software é composta por um conjunto de atividades da engenharia de software. Pressman (2016) afirma que uma metodologia genérica da engenharia de software é composta de cinco atividades, que são:

- **Comunicação:** Com a intenção de entender os objetivos do projeto, a comunicação entre os envolvidos é a primeira ação primordial, para entender os requisitos (as funcionalidades) do software a ser realizado.
- **Planejamento:** É realizado um “mapa”, um plano do projeto do software a ser realizado, descrevendo as tarefas técnicas, os riscos, os recursos, os produtos resultantes e um cronograma de trabalho (para acompanhar o desenvolvimento do software).
- **Modelagem:** São criados modelos (diagramas) para melhor entendimento das necessidades do software, os modelos são utilizados para realizar a codificação do software e para validação das partes envolvidas no projeto.
- **Construção:** Realização da codificação (baseada nos modelos criados anteriormente), nesta fase também são realizados os testes para validar os códigos de programação gerados.
- **Entrega:** O software é entregue parcialmente ou na sua totalidade, onde o cliente realzia testes e fornece um feedback, nesta fase são realizadas adaptações e correções no software por um determinado período (acordado entre as partes).

Fluxo de Processo

As atividades metodológicas devem ter uma série de tarefas que darão suporte no acompanhamento e controle do projeto, controlando os riscos, fazendo revisões técnicas, etc. O fluxo de processo é usado para descrever como as atividades metodológicas de cada processo são organizadas.

Os fluxos de processo podem ser:

- **Fluxo de Processo Linear:** Caracteriza-se por realizar em forma sequencial as cinco atividades metodológicas apontadas por Pressman (2016), inicializando pela comunicação e terminando na fase da entrega.
- **Fluxo de Processo Interativo:** Possui como característica a repetição de uma ou mais atividades antes de avançar para a próxima atividade.
- **Fluxo de Processo Evolucionário:** As atividades são executadas de modo circular, cada ciclo envolve as cinco atividades, gerando uma versão mais completa (é um processo incremental).
- **Fluxo de Processo Paralelo:** As atividades são realizadas de forma paralela, onde duas ou mais atividades podem ser executadas simultaneamente, por exemplo, a atividade de comunicação pode ocorrer em paralelo com a atividade de análise.

Planejamento de um Software

O desenvolvimento de um software requer muito planejamento e um software nunca é igual ao outro. Pressman (2016) afirma que projetos diferentes exigem conjuntos de tarefas e modelagem das atividades do processo de software diferenciados. Os analistas de sistemas determinam o conjunto de tarefas baseados nos problemas e nas características do projeto que será executado.

Observe um conjunto de tarefas (atividades) na fase de planejamento de um software:

Fase	Atividades	Resultados
Planejamento	Levantamento de requisitos	Documentação do levantamento de requisitos
	Especificação dos requisitos	Documentação da especificação de requisitos
	Estimativa de prazos	Plano de ação para determinar os prazos
	Estimativa de recursos	Alocação de recursos para criação do software

Modelo de Processo de Software

Modelo de processo de software é uma descrição simplificada do processo que especifica as atividades para o desenvolvimento, define os produtos de cada atividade, determina os papéis dos envolvidos no desenvolvimento, oferecendo um roteiro para a engenharia de software (Sommerville, 2011).

A qualidade do software produzido é diretamente influenciada pelos padrões de qualidade impostos durante os processos de software (durante a produção do software) sendo necessário estabelecer procedimentos e padrões para garantir a qualidade dos processos.

O processo de software pode passar por uma série de critérios preestabelecidos que ajudam a garantir a integração e validação entre as atividades do processo de software. Pressman (2016) destaca uma série de abordagens de avaliação e aperfeiçoamento dos processo de software:

- **SCAMPI:** A abordagem SCAMPI (Método Padrão CMMI de Avaliação para Aperfeiçoamento de Processo da CMMI – Standard CMMI Appraisal Method for Process

Improvement) fornece um modelo de avaliação do processo em cinco etapas – início, diagnóstico, estabelecimento, atualização e aprendizado.

- **CBA IPI:** A abordagem CBA IPI (CMM Bases Appraisal for Internal Process Improvement – Avaliação para o Aperfeiçoamento do Processo Interno Baseado na CMM (Capability Maturity Modelo – Modelo de Maturidade em Capacitação)), fornece uma técnica de diagnóstico para avaliar a maturidade relativa de uma organização de software. O método possui a capacidade de identificar pontos fortes e fracos dos processos e viabiliza a possibilidade de priorização das melhorias mais relevantes.
- **SPICE:** A abordagem SPICE (ISO/IEC15504) é um padrão que define um conjunto de requisitos para avaliação do processo de software, possui a finalidade de auxiliar as organizações no desenvolvimento de uma avaliação objetiva da eficácia de um processo de qualquer software. Este método fornece uma estrutura para a avaliação de processos de software e esta estrutura pode ser empregada nas organizações envolvidas na produção de um software.
- **ISO 9001:2000 para Software:** A abordagem ISO 9001:2000 para software, é um padrão genérico aplicável a qualquer organização que precise aplciar um padrão global de qualidade em seus produtos, sistemas ou serviços. Existem vários modelos de referências para ajudar a garantir a qualidade e que são aplicáveis a um software, alguns destes modelos são: ISO/IEC 9126, a ISO9000, a ISO9001, a ISO/IEC12207.

Os erros que ocorrem durante o processo de software podem ser controlados utilizando uma abordagem metodológica. O analista de sistemas deve estar atento ao surgimento de novas metodologias, testá-las e, se forem apropriadas, utilizar durante o processo de software. O objetivo é criar um software com qualidade com o mínimo de erros e com a aprovação do cliente.

Vimos as atividades do processo de software, as quais estão relacionadas com o planejamento para o desenvolvimento do produto (o software).

Para contribuir na ampliação de seus conhecimentos a respeito dos conceitos sobre sistemas, consulte o capítulo 2: “Os Paradigmas na Engenharia de Produção”, presentes no livro: Sistemas de Produção – Conceitos e Práticas para Projeto de Gestão da Produção Enxuta, de Antunes.

Modelos de Processo de Softwares

Durante a fase de desenvolvimento de um software existem muitas tarefas, que devem ser distribuídas entre os membros da equipe. Sommerville (2011) destaca que dentre as atividades do processo de software estão: O estudo da viabilidade, a análise dos requisitos, a especificação, a arquitetura de software, implementação (codificação), testes, documentação, suporte e treinamento e manutenção.

Para o gerenciamento das atividades de processo de software são utilizados os modelos de processos de software.

Modelo de processo de software é um guia exclusivo para as atividades da engenharia de software, definindo um fluxo de todas as atividades, ações e tarefas, o nível de interação entre as atividades, os artefatos que serão produzidos e a organização do trabalho que deve ser realizado (Pressman, 2016).

Existem diversos modelos de processos de software que possuem características diferentes. Destacam-se os seguintes: Modelos prescritivos, modelos de processos especializados e modelos de desenvolvimento ágil.

Modelo de Processo Prescritivo

Também conhecido como modelos de processos tradicionais. Esse tipo de modelo prescreve os relacionamentos, ou seja, como os elementos dos processos são interligados, com a finalidade de estruturar e ordenar o desenvolvimento de um software.

As tarefas ocorrem de forma sequencial, com diretrizes bem definidas, uma vez que indicam uma série de atividades metodológicas, ações, tarefas, artefatos, garantias de qualidade e mecanismos de controle de mudanças para cada projeto. Para casa modelo de processo prescritivo também é indicado um fluxo de processo (Pressman, 2016).

Alguns dos modelos de processos prescritivos são os seguintes: Modelo cascata, modelo incremental, modelos evolucionários (divididos em: Prototipação e espiral) e modelos concorrentes.

Modelo Cascata

Conhecido como ciclo de vida clássico de um sistema ou abordagem top-down, o modelo cascata possui enfoque sistemático e sequencial dos processos, cada fase é iniciada somente após a conclusão da fase anterior.

O modelo em cascata é considerado o modelo mais tradicional e simples, com especificação das atividades de forma clara, além de ser uma base para modelos que surgiram posteriormente e de fácil gerenciamento. Todavia, o desenvolvimento de um software pode se estender ao longo de meses, dependendo da complexidade do projeto, uma vez que as tarefas são realizadas de forma sequencial e o atraso em uma das etapas reflete nas demais.

Modelo Incremental

É um modelo iterativo que visa, a partir de requisitos iniciais, criar pequenas versões do software, que vão sendo entregues ao cliente, e posteriormente expandir o software em novas versões até o sistema ideal ser totalmente construído (Sommerville, 2011). Nesse modelo, uma versão é um incremento, e cada incremento (ou versão) incorpora parte da funcionalidade requisitada pelo cliente.

No modelo incremental ao invés do cliente receber o software em uma única entrega, ele receberá “pedaços” do software (versões), a cada incremento, até que o software seja desenvolvido por completo. Esses “pedaços” são módulos que acrescentam, ou melhoram as funcionalidades do sistema. Cada incremento (entrega) é lançado como uma nova versão, do software, até que se atinja a versão final.

Modelo Evolucionário

Esse modelo produz uma versão cada vez mais completo do software. Esse tipo de modelo é iterativo e evolui ao longo do tempo, o que se alinha perfeitamente a um projeto do software, pois os requisitos do negócio e do produto não são estáveis, eles mudam (evoluem) e o software pode ser desenvolvido pensando nesta evolução (Pressman, 2016). No modelo de processo evolucionário aparecem dois modelos: Prototipação e Espiral.

O modelo de prototipação, começa a partir da comunicação, identificando quais são os objetivos e as funcionalidades do software. No planejamento rápido são determinados os requisitos que serão modelados (e quais serão “deixados” para serem implementados mais tarde) e que já podemo ser

modelados e construídos. Após a entrega, o cliente dará um feedback e a equipe de desenvolvimento irá aprimorar os requisitos.

O modelo espiral (um tipo de modelo evolutivo) é iterativo como a prototipação, mas utiliza os aspectos sistemáticos e controlados do modelo cascata (Pressman, 2016). O objetivo do modelo espiral é fornecer um rápido desenvolvimento de versão, que a cada ciclo possa gerar versões mais completas.

Modelos Concorrentes

São representados graficamente por uma série de tarefas e técnicas maiores e estados associados a elas e que são utilizados como um paradigma para o desenvolvimento de aplicações cliente/servidor. Os modelos concorrentes são utilizados em projetos que envolvem diferentes equipes de desenvolvimento e, conforme Pressman (2016), os planos de projeto devem ser considerados documentos vivos e a evolução de cada processo deve ser avaliada com frequência e revisada levando em consideração as alterações. Os modelos de processos concorrentes podem ser aplicados a diversos tipos de desenvolvimento de softwares e, diferentemente do modelo cascata, ele não segue uma sequência de atividades, mas estabelece uma rede de atividades que se movimentam de uma atividade para outra.

Modelo de Processo Especializado

Esses modelos utilizam muitas das características de um ou mais modelos de processos prescritivos e são utilizados quando existem a necessidade de uma abordagem mais especializada de engenharia de software. Conforme Pressman (2016), eles são:

- **Modelo Baseado em Componentes:** São utilizados em projetos de software de prateleira (software de linha), compreende aplicações de componentes de software previamente empacotados (e vendidos em partes ou completo). São desenvolvidos para poderem ser reutilizados em outros projetos. Um componente é uma parte do independente do software e pode ser trocado ou alterado.
- **Modelo de Métodos Formais:** Compreende um conjunto de atividades que levam à especificação matemática forma do software, fornecendo mecanismos para a descoberta e a eliminação de muitos problemas como a ambiguidade, incompletude e inconsistência. Servem como base para fazer a averiguação do código de programação com o objetivo de descobrir erros (que passariam despercebidos).
- **Desenvolvimento de Software Orientado a Aspecto:** Fornece um processo e abordagem metodológica para definir, especificar, projetar e construir aspectos. O código do software é separado de acordo com sua importância (classes orientadas a objetos é um exemplo de aspecto) e os requisitos são modelados transcendendo várias funcionalidades do sistema.
- **Modelo de Processo Unificado:** Conhecida também como RUP (Rational Unified Process), aproveita as características dos modelos de processo tradicionais (prescritivos), mas implementa alguns princípios da metodologia ágil (abordada mais adiante nesta seção), é considerado um modelo iterativo e incremental.
- **Modelos de Processos Pessoal e de Equipe:** O cerne do desenvolvimento de software está diretamente ligado a toda equipe de desenvolvimento. No modelo de processo de software pessoal é enfatizada a medição pessoal do que foi produzido (do artefato gerado e a qualidade resultante). O modelo de processo de software de equipe objetiva a criação de uma equipe autodirigida, que se organiza por si mesma com a finalidade de produzir um software com alto padrão de qualidade.

Modelo de Desenvolvimento Ágil

O desenvolvimento ágil procura resolver alguns problemas da engenharia de software, oferecendo benefícios importantes. As etapas de levantamento de requisitos, análise e projeto são muito demoradas e, de acordo com Pressman (2016), o desenvolvimento ágil é uma resposta ao rápido desenvolvimento de software (os clientes querem resultados rapidamente) e aplicativos (onde novas funcionalidades são agregadas, na medida que surgem novas ideias ou necessidades), tornando desta maneira o desenvolvimento mais flexível e atendendo as necessidades do cliente com mais rapidez.

O princípio do desenvolvimento ágil é focado nas entregas, priorizando também a comunicação entre os envolvidos de forma ativa e contínua para realizar entregas incrementais (procurando a satisfação do cliente) (Pressman, 2016).

Segundo Pressman (2016), o processo de desenvolvimento ágil visa reduzir drasticamente a documentação, tornando o processo de desenvolvimento flexível e reduzindo a burocracia (presente em outros modelos de processos de softwares). Nesse universo, dois métodos ágeis se destacam: XP (Extreme Programming) e Scrum.

Método (ou Metodologia) XP

O feedback é constante, a abordagem de desenvolvimento é incremental e a comunicação entre os envolvidos é primordial. Quatro atividades metodológicas precisam ser seguidas: O planejamento, o projeto, a codificação e os testes, o desenvolvimento do software deve ser padronizado e com o trabalho sendo realizado em pares e o cliente (um representante) deve estar sempre presente para esclarecer dúvidas (ele faz parte da equipe de desenvolviment).

Metodologia Scrum

Determina um processo de desenvolvimento iterativo e incremental, e pode ser utilizado em processos gerenciais. Esse método define um conjunto de regras e práticas de gestão, para alcançar o sucesso dos projetos como, por exemplo, o trabalho em equipe e comunicação melhorada. O scrum possui as seguintes atividades metodológicas: Requisitos, análise, projeto, evolução e entrega, e em cada atividade ocorrem as seguintes tarefas principais:

- **Backlog:** Lista com prioridades dos requisitos (das funcionalidades) do projeto, na qual um item pode ser adicionado ou eliminado a qualquer momento (essas são as alterações) e o gerente do produto deve registrar e atualizar as prioridades.
- **Sprints:** São unidades de trabalho para atingir um requisito (estabelecido no backlog) e precisa ser ajustado dentro do time box (janela de tempo) para definir os prazos de entrega.
- **Reunião de Planejamento:** Reunião na qual o product owner (dono do produto) prioriza os itens do product backlog e a equipe seleciona as atividades que ela será capaz de implementar durante o sprint que se inicia.
- **Reuniões Scrum:** Reuniões breves, geralmente de 15 minutos, chamadas daily meeting e realizadas diariamente (geralmente no início da manhã), nesta reunião são realizadas três perguntas chaves (para cada integrante da equipe: O que você realizou desde a última reunião de equipe? Quais foram os obstáculos encontrados? O que planeja realizar até a próxima reunião?

- Quando o projeto inicializa, são definidas as ideias e funcionalidades iniciais do produto a ser desenvolvido, estas ideias são chamadas histórias e o conjunto de todas as histórias forma o product backlog.
- As reuniões são conduzidas pelo scrum master que conduz o processo e realiza avaliações das respostas de cada integrante da equipé, detectando de forma precoce eventuais problemas, como atrasos ou dificuldade de entendimento de algum requisito.
- No término da sprint os requisitos são concluídos e o funcionamento é avaliado, melhorando o processo para a sprint sequencial. Cada sprint se encerra com um incremento ao produto (ou product backlog).

Na metodologia scrum sempre é montado um quadro (board) para acompanhar as tarefas. No quadro são inseridas as atividades stories, to do, in progress, testing e done.

Esse quadro pode ser adaptado para a realizada de cada equipe de desenvolvimento.

Existem ainda outros métodos de desenvolvimento ágil, Pressman (2016) lista os seguintes: Método de desenvolvimento de sistemas dinâmicos (DSDM), modelagem ágil e processo unificado ágil. Todos os métodos ágeis enfatizam a colaboração humana e a auto-organização como elementos chaves.

Vimos os modelos de processos de software e seus diferentes tipos: Modelos prescritivos, modelos especializados e modelos ágeis. Todos os modelos apresentados são amplamente utilizados nas empresas de desenvolvimento de software. São fatores determinantes para a escolha do tipo de modelo a ser usado: O tipo de software que deverá ser produzido e a experiência dos analistas de sistemas (envolvidos nos projetos).

Para conhecer mais sobre o processo de software para metodologias ágeis, consulte o capítulo 1: Processo de Software, no livro: Engenharia de Software (Recurso Eletrônico) de Vettorazzo.

Fundamentos de Processos de Negócio

Nesta aula vamos compreender os fundamentos de processos de negócios, entendendo os conceitos envoltos às áreas de negócios, entendendo os conceitos envoltos às áreas de negócios, estrutura e classificação de processos e tipos de visão.

Processo de Negócio

Cada empresa tem uma estrutura, atividades e, por consequência, áreas diferentes, portanto, é fundamental analisar cada organização com bastante cautela para compreender e determinar quais áreas existem e como elas se relacionam.

O papel de integração da área TI às demais áreas de negócio é bastante significativo, pois somente desta forma será possível evitar as falhas no desenvolvimento de software, para que este não pareça desconectado do negócio.

A área de TI, atualmente, contribui de forma estratégica para as organizações, e auxilia no desenho e gestão de processos adequados que permitem a entrega de soluções mais efetivas aos clientes.

Área de Negócio

As áreas de negócio são aquelas que têm por objetivo dar prosseguimento à missão organizacional, por meio de produção de bens ou serviços que atenderão às necessidades do cliente externo. Tais atividades são determinadas como atividades essenciais, pois estão diretamente ligadas à atividade central (core business) da organização.

Quando falamos em áreas de negócios, não podemos nos esquecer do processo de negócio, mas antes é necessário lembrar do conceito de processo.

Processo trata uma sequência lógica e estruturada de tarefas que apresentam uma entrada – input – de diversos elementos, que são processados e geram saídas – outputs – (Chiavenato, 2014).

Os inputs, entradas do processo, envolvem, basicamente, a área de recursos humanos (disponibilidade de pessoas), máquinas e equipamentos (desde um computador a um maquinário produtivo), materiais (insumos de produção ou não), recursos financeiros, informações (do cliente, do mercado, do estoque), procedimentos (como deverá ser executado). Por sua vez, os outputs representam o resultado gerado a partir do processamento de todos os recursos de entrada que têm por objetivo gerar um produto ou serviço (Paim et al., 2009).

Estrutura de Processos

A complexidade do processo infere diretamente no seu grau de dificuldade de modelagem e gerenciamento, sendo necessário realizar sua documentação adequada, que trata o detalhamento de como as tarefas e atividades devem ser executadas, a quem cabe a execução das tarefas para que o resultado esperado seja atingido. Vale ressaltar que um processo simples também deve ser documentado, pois favorece a padronização e pode gerar interferência em outros processos.

Em relação ao processo de decomposição, ele pode ser intitulado de macroprocesso, processo e subprocesso.

Os macroprocessos tratam os processos mais abrangentes da organização, que são subdivididos em processo, que por sua vez são fracionados em subprocesso. Exemplo: Uma montadora automobilística que possui a fabricação de veículo (macroprocesso), possui diversos processos, sendo um deles a produção, que tem como subprocessos pintura e montagem, entre outros.

Classificação dos Processos

O processo de negócio representa a consolidação de atividades/tarefas que visam atingir um resultado que demonstre valor agregado ao cliente. Por meio de um bem ou serviço, ele realiza o sequenciamento de atividades/tarefas de forma lógica a fim de desenhar como o trabalho deverá ser executado. Os processos de negócios são classificados, conforme suas características, em processos primários, processos de suporte e processos de gerenciamento (Brocke e Rosemann, 2013):

- **Processos Primários:** Os processos primários estão vinculados ao core business (negócio principal) da organização, ou seja, possuem vínculo com as atividades essenciais, são aqueles que agregam valor ao cliente final e, por consequência, os resultados deles demonstram o grau de satisfação do cliente. Ajudam a traduzir a missão da organização e podemos caracterizar como exemplos a produção dos produtos, a entrega ao cliente, o marketing, entre outras atividades voltadas à agregação de valor ao cliente.
- **Processo de Suporte:** Os processos de suporte suportam as ações dos processos primários e de gerenciamento, isto é, apoiam os dois outros grupos de processos. Auxiliam os processo primários a realizarem as entregas que agregam valor ao cliente, mas não entregam valor

diretamente para o cliente final. Em outras palavras, agregam valor ao processo e não ao cliente. As áreas de recursos humanos, de tecnologia da informação, de contabilidade, de contas a pagar representam áreas de apoio aos processos primários, mas não estão diretamente conectados ao cliente final. Primeiramente deve ocorrer um alinhamento organizacional para que todos os colaboradores estejam “remando” para a mesma direção, e posteriormente terem desenhos claros dos processos primários, de suporte de gerenciamento.

- **Processo de Gerenciamento:** Os processos de gerenciamento existem para que ocorra o acompanhamento dos resultados, consegue determinar se eles são satisfatórios ou não e, por consequência, determinam melhorias que indiretamente agregarão valor ao cliente, porém, por meio de melhoria nos processos e não em agregação de valor direta ao cliente. Tem como alvo fazer com que os objetivos organizacionais sejam atingidos. O monitoramento e controle acontecem por meio de gerenciamento de performance (desempenho), por exemplo, que servem para acompanhar, medir ou controlar o andamento dos processos primários e de suporte. O gerenciamento de processo é de grande relevância para organização, pois acarreta em benefícios que gerarão impactos diretos no cliente.

Tipos de Visão

A visão funcional da organização está ligada à sua estrutura hierárquica, e isto quer dizer que trata um modelo de visualização vertical. Dessa forma, os processos são vistos por departamento e cada um gerencia um recursos específico de sua área (De Sordi, 2018). Essa visão não trabalha a conectividade entre as áreas de negócios, portanto, cada área é percebida isoladamente como se não houvesse conexão com as demais áreas. Os objetivos são vistos de forma isolada, cada departamento pensa apenas nos seus objetivos e não foca o objetivo global e, por consequência, os desempenhos são avaliados de forma departamental. O foco principal da visão está no desenvolvimento de competências funcionais na equipe, e isto quer dizer, que se privilegia uma visão restrita e não ampla. Como resultado, o reconhecimento se torna individualizado e restrito, sem avaliar se a atividade conseguiu agregar valor ao cliente ou ao processo que suporta.

O fluxo de informação ocorre em um padrão hierárquico, ou seja, de cima para baixo e, portanto, não gera um processo de comunicação horizontal, mas apenas vertical. As diretorias e gerências funcionam de forma individualizada, tornando muito mais difícil atender às necessidades do cliente, pois a visão é hierarquizada e estrutural.

A visão de processos ponta a ponta traz uma visão bastante ampla, pois trafega e visualiza a conexão entre todos os departamentos, isto é, em uma perspectiva horizontal. Envolve questões como tempo, custos, capacidade, qualidade, o que permite compreender a contribuição dada por cada parte para atender às necessidades do cliente. Permite uma visualização nos diferentes níveis e representa uma forma de agregar valor ao cliente. Essa visão vai além das funções departamentais, e conecta todos os departamentos que estão vinculados a um determinado processo. Pode ser ainda intraorganizacional, que além de conectar os departamentos conecta elementos externos à organização (ABPMP, 2013).

A integração horizontal existente entre as diretorias e gerências tem maior possibilidade de atender às necessidades dos clientes de forma satisfatória.

Lembre-se: Para a modelagem de sistema acontecer de forma adequada é necessária a integração entre as áreas.

Portanto, podemos concluir que o desenho de processo adequado determinará o sucesso do desenvolvimento de sua atividade e, portanto, é necessário focar em uma coleta de dados adequada para que possa atender aos requisitos do cliente de forma satisfatória.

O artigo intitulado As Empresas são Grandes Coleções de Processos traz à tona a complexidade existente nos processos organizacionais e reforça os conteúdos abordados ao longo da seção, desde a definição de processo, passando pelas áreas de negócios, sua classificação, e enfatiza os aspectos que permeiam a visão por processos.

Na sequência, o artigo Transparência de Processo Organizacionais corrobora para que seja percebida a importância de se ter processos de negócio que demonstrem transparência.

Modelagem de Processos de Negócio

Nessa aula vamos compreender os conceitos vinculados à modelagem de processos de negócios, a notação BPMN (Business Process Modeling Notation) e seus elementos que são utilizados para formar o desenho de processos de negócio, a cadeia de valores, o fluxo de trabalho e documentação de processos de negócios.

Modelagem de Processos

Modelagem, segundo o dicionário Michaelis (2020), significa ato ou resultado de modelar e aplicada ao campo da informática trata da criação de modelos. Podemos entender esses modelos como representações em escala reduzida, isto é, a simplificação de algo real.

Processo de negócio é traduzido como uma sequência de atividades executadas para atingir um objetivo (resultado) que agregue valor ao cliente.

A modelagem pode acontecer por meio de uma representação simples, que é composta por uma quantidade de elementos e áreas de negócio reduzidas, ou complexa, com uma grande quantidade dos mais variados elementos e com muitas áreas envolvidas. Os modelos podem ser matemáticos, gráficos, descritivos ou uma combinação de alguns ou de todos, e são utilizados para organizar, aprender, prever, medir, explicar, verificar e controlar (ABPMP, 2013).

Motivos para realizar o processo de modelagem:

- **Melhorar Processos:** Avaliar e redesenhar processos visando melhor desempenho e atendendo melhor às demandas dos clientes internos/externos.
- **Eliminar ou Automatizar Processos:** Criar processos mais ágeis e eficazes que permitam custos reduzidos.
- **Documentar Processos:** Para que a organização possua informação uniforme e que todos os seus membros, por meio da documentação, possam compreender e realizar as tarefas ou atividades necessárias.

Técnicas de Modelagem

Há diversas técnicas de modelagem, algumas são utilizadas para fins específicos e outras trazem uma aplicação mais ampla, porém as mais difundidas são:

- **BPMN (Business Process Modeling Notation):** O BPMN atua com diagrama único BPD (Business Process Diagram) que permite desenhar os mais diversos tipos de modelagem de processo (Brocke e Rosemann, 2013).
- **UML (Unified Modeling Language):** Dá suporte ao desenvolvimento de softwares e, é uma linguagem de representação gráfica especificada, é independente da metodologia de

modelagem de processos adotada, sendo apenas um conjunto de convenções de modelagem (Valle e Oliveira, 2013).

- **IDEF (Integrated Definition):** O IDEF permite a modelagem de requisitos para sistemas, as técnicas IDEF0 e IDEF3 são utilizadas para modelagem de processos de negócios.
- **EPC (Event-Driven Process Chain):** O EPC visa a modelagem com base no controle de fluxo de atividades e suas dependências. Tem foco essencialmente para descrição de processos.

Tipos de Abordagem:

- **Abordagem Bottom Up (De Baixo pra Cima):** Parte do detalhamento de tarefas e atividade e depois se estabelece uma visão macro da empresa, isto é, caminha do nível mais baixo (micro) para o mais alto (macro) da organização.
- **Abordagem Bottom Down (De Cima para Baixo):** Ocorre de maneira inversa, primeiro se tem a visão macro (geral da organização) e posteriormente se atinge a visão do processo (tarefas e atividades).

Modelagem BPMN (Business Process Modeling Notation)

É uma técnica de fácil compreensão, pois atua com notações mais simples e que podem ser facilmente compreendidas, pode ser utilizado por todos os envolvidos nos processos de negócio e permite modelagem de todo o tipo de processo (compras, vendas, empréstimos, manutenção, distribuição, desenvolvimento de produtos ou serviços, entre outros).

O BPMN se apresenta no formato de linhas paralelas e cada linha representa um papel diferente a ser desenvolvido na realização do trabalho. É composto por elementos básicos e específicos.

Veja a seguir cada uma desses elementos.

Os elementos básicos que compõe o BPMN são divididos em atividade, evento, gateway e conectores.

Atividades

Também é possível verificar o desdobramento das atividades e os subprocessos, são diversos os desdobramentos existentes. No interior de cada elemento há indicações de especificidade do elemento.

O processo é um conjunto de objetos gráficos compostos por tarefas e subprocessos, isto é, ele não é representado por um único elemento, mas um grupo deles.

Subprocessos:

- Colapsado é adicionado do símbolo “+” que indica outro nível de detalhes. Expandido contém um processo de negócio.
- Transacional têm diversas atividades que devem ser completadas ou canceladas.
- Com marcador de loop podem ser utilizados para executar atividade até que a condição seja satisfeita.
- Com marcador de instâncias múltiplas executa diversas atividades até que todas sejam satisfeitas.

- Com marcador compensação serve para compensar atividade já executada do processo, isto é, desfaz a atividade.

Eventos

O evento trata algo que ocorre durante o processo de negócio e afeta o fluxo do processo. Há três tipos de eventos: Os de início (círculo com contorno claro), os intermediários (círculo duplo), que pode ser utilizado para enviar uma informação, e os de encerramento (círculo com contorno escuro).

Todos os eventos apresentam uma indicação (representação gráfica) no centro do elemento. Nos eventos de início e intermediário essas representações significam os disparadores, e nos eventos de fim são os resultados.

Gateway

Os gateways são filtros de decisão, eles separam e juntam os fluxos. Caso o fluxo não precise ser controlado não há a necessidade deste elemento.

Tipos de gateways:

- **Gateway Exclusivo Baseado em Dados:** Tem como caminhos possíveis “sim ou não” em resposta a uma pergunta, portanto, trata uma decisão com escolha de apenas uma alternativa.
- **Gateway Baseado em Evento:** Depende de uma resposta externa ao processo para determinar o ponto de desvio. Exemplo: Uma cotação é enviada a um cliente que pode responder (mensagem) com “sim” ou “não” e com base nesta resposta é determinado o caminho a ser tomado.
- **Gateway Inclusivo:** Que depende de mais de uma condição para dar sequência na atividade, ou seja, ele não trabalha com “sim” e “não”, mas com a satisfação de duas ou mais condições para dar andamento na tarefa.

Conectores

Os conectores servem para dar direção ao fluxo e podem ser divididos em três modalidades: Sequência do fluxo, fluxo da mensagem e associação de elementos.

Swinlanes

O conceito de swinlanes serve para ajudar a dividir e organizar as atividades e é dividido em: Pool (piscina) e lane (raia):

- Pools devem ser utilizados quando se envolvem duas ou mais entidades de negócios ou atores determinando quem faz “o quê”.
- Lane é a separação das atividades associadas para um papel específico, isto é, são utilizadas para representar um ator do processo.

Artefatos

Os artefatos que são elementos que contribuem para que sejam mostradas informações além da estrutura básica do diagrama.

Artefatos:

- **Objeto de Dados:** É utilizado para agregar informação ao processo, isto é, trata-se de um conjunto de informações referentes a uma atividade específica, por exemplo, a atividade “emitir pedido” é um documento que possui uma série de detalhes.
- **Grupo:** Serve para dar destaque a um grupo de atividades, ou seja, coloca em ênfase um grupo de atividades.
- **Anotação:** Traz comentários do processo que ajudam a entender a tarefa, mantendo o exemplo da atividade “emitir pedido” teríamos como anotação verificar “impressora para que o pedido seja impresso”.

Modelo de Processos

Para a ABPMP (2013), o modelo de processo é composto por ícones que representam atividades, tarefas, decisões e podem conter informações. Esse modelo de processo pode ser representado por um diagrama, mapa ou modelo:

- **Diagrama:** Retrata apenas os principais elementos do fluxo, porém, não mostra detalhes menores relativos ao fluxo de trabalho. Ele contribui para entender as principais atividades do processo.
- **Mapa:** Além do conteúdo do diagrama, agrega mais detalhes acerca do processo, mostra os principais componentes do processo e apresenta maior precisão que um diagrama, e agrega mais detalhes acerca dos atores, eventos e resultados.
- **Modelo:** É mais completo, segundo a ABPMP (2013), pois cria uma representação do negócio (atual ou futuro), de todos os envolvidos (pessoas, informação, instalações, automação, finanças e insumos) e dos fatores que afetam o comportamento do processo.

O BPMN define e utiliza um único modelo de diagrama, trata-se do Business Process Diagram (BPD), ou o Diagrama de Processo de Negócio (DPN). Ele representa a saída gráfica de um modelo de processo no BPMN e é capaz de retratar diversos tipos de modelagem.

Não se esqueça que o BPMN nada mais é que uma notação evoluída de um fluxograma e que serão utilizados apenas os elementos que se fizerem necessários ao longo do processo de modelagem.

Cadeia de Valor

A cadeia de valor é um dos recursos que as organizações utilizam para que consigam manter a vantagem competitiva frente aos concorrentes. As empresas podem possuir uma ou mais cadeias de valor, que representam os processos centrais que definem a empresa, portanto há variação de empresa para empresa. A cadeia de valor depende do alinhamento entre todas as áreas organizacionais para que ocorra viabilidade na realização de todos os processo com a maior eficácia possível.

A cadeia de valor é um elemento primordial para a análise e modelagem de processos de negócio, pois contribui para a compreensão dos objetivos organizacionais e definições de processo primários (principais), de apoio (suporte) e de gerenciamento.

Fluxo de Trabalho

Ao realizar a modelagem de processos de negócio dentro da cadeia de valores estabelecida pela organização é possível vislumbrar um fluxo de trabalho que entregará ao cliente o valor agregado necessário. O fluxo de trabalho nada mais é que a consolidação de atividades em uma área

funcional com foco em eficiência e a modelagem mostrará o trabalho como um fluxo que descreve o relacionamento de cada atividade com as demais atividades executadas na área funcional (ABPMP, 2013).

Documentação

Após a realização de mapeamento e modelagem dos processos será necessário gerar e disponibilizar a documentação necessária às áreas envolvidas em cada processo de negócio. A documentação tem por objetivo permitir a compreensão do estado atual (as is), bem como subsidiar informações para o diagnóstico que permitam vislumbrar mudanças nos processos (to be).

A documentação de análise permite elucidar uma visão geral do ambiente de negócios, para determinar o motivo pelo qual cada processo existe, registrar os processos mostrando suas interações e subprocesso, demonstrar o fluxo de trabalho (atividades realizadas dentro da área funcional), compreender os requisitos de medição de desempenho nos processos, motivos para que existam essas lacunas, compreensão de regras documentadas e não documentadas que afetam as atividades, identificação de tecnologia de informação utilizadas e em quais processos, onde os dados são coletados, armazenados e acessados, política de auditoria interna, oportunidade de melhoria e benefícios e riscos e seus impactos no processo (ABPMP, 2013).

Vimos até aqui os aspectos da modelagem de processos de negócio, possibilitando a compreensão de que a utilização de todos os elementos da modelagem contribui de forma significativa para que haja um alinhamento facilitado entre todos os envolvidos no processo, independentemente do nível organizacional, pois o entendimento da notação se torna bastante facilitada.

O texto Gestão de Processos de Negócio: Um Estudo de Caso da BPMN em uma Empresa do Setor Moveleiro mostra a aplicação prática da notação em questão em três processos de negócios do setor específico e demonstra as percepções dos indivíduos com relação à utilização dos modelos da BPM. Com essa leitura é possível perceber a importância do BPMN e como aplicá-lo de forma efetiva.

Gerenciamento de Processos de Negócio

O gerenciamento de processos de negócio (BPM) é uma atividade de extrema importância para todas as organizações e em todas as suas áreas, pois é essa atividade que permite a identificação de gaps (lacunas) de resultados e, a partir daí permite a criação de um processo de melhoria contínua. A criação de KPIs (Key Performance Indicator) por área permite o monitoramento permanente de cada área que faz parte do processo de negócio e, por isso, é importante lembrar que os indicadores de desempenho das áreas devem contribuir para a melhor execução do processo como um todo, isto é, todas as etapas estão ligadas e se relacionam.

O BPM pressupõe a integração entre as áreas, a sua comunicação, a troca de informações e os tipos de relacionamento existentes, fortalecendo uma visão horizontal. Ele é necessário para olhar o processo de um nível mais elevado do que o de execução do trabalho, pois apenas vendo o todo do processo será possível entender a relação entre as áreas e subdividi-lo por subprocessos que serão executados por diversas atividades (fluxos de trabalho) dentro das áreas funcionais.

Governança dos Processos

Para que o BPM funcione de forma adequada, elementos como cultura, clima, estruturas, tecnologia, políticas devem ser compreendidos para que ocorra a governança dos processos, isto é, todas as ações são direcionadas pelas mesmas regras e diretrizes com o intuito de se atingir os objetivos organizacionais. A governança deve incorporar a ideia de controle e da prestação de

contas, o que para a gestão dos processos torna-se perfeitamente adequado e extremamente necessário (Araújo, Garcia, Martines, 2017).

Os processos de negócio estão embasados em algumas metodologias de mudanças (Brock e Rosemann, 2013).

O processo de evolução da melhoria dos processos, tem início em uma visão de simplificação do trabalho, evoluindo para os controles de qualidade, seis sigma, lean, gestão de negócios e tecnologia da informação. A soma de todos esses elementos culmina no BPM, e vinculado a ele temos o BPMS como suporte tecnológico (BPMS: Business Process Management Suit or System, que no português significa sistema de gerenciamento de processos de negócio, ou seja, sistema (software) que permite a realização do mapeamento, execução e monitoramento dos processos organizacionais).

BPMS (Business Process Management Suit ou System

O BPMS é uma ferramenta que permite mapear, executar e monitorar os processos funcionais com o intuito de dar-lhes uma visão de processo ponta a ponta, ou seja, contribuem para a automatização das ações e do fluxo de informações existentes nos processos.

Para Araújo, Garcia e Martines (2017), o BPMS é considerado uma evolução do workflow (fluxo de trabalho), pois é capaz de integrar diversos workflows. Por conta disso, o BPMS traz uma visão muito mais ampla e permite que ocorra integração com sistemas legados (sistemas antigos que permanecem em operação).

O BPM conta com um conjunto de atividades, tais como modelagem, análise, desenho, medição de desempenho e transformação de processos, a fim de assegurar um ciclo que gere processos de negócios alinhados com a estratégia da empresa e com o foco do cliente.

Gestão de Negócios

Para Valle e Oliveira (2013), os processos de negócio devem estar alinhados com os objetivos e estratégias organizacionais.

O desdobramento ocorre a partir da entrega de valor ao cliente (valor que deve ser percebido por ele) por meio de produtos/serviços, que devem ser alinhados aos objetivos organizacionais, ou seja, a empresa toda deve trabalhar em prol dos objetivos. Os objetivos são desdobrados em processos e seu gerenciamento, que visam a melhor forma de atender as necessidades do cliente e, por consequência, aos objetivos organizacionais.

A visão de gestão de negócios tem uma relação íntima com a função administrativa, que segundo Chiavenato (2014) estabelece padrões de desempenho que sejam mensuráveis e que possam ser comparados com os resultados reais por meio de monitoramento para que, se necessário, sejam tomadas medidas corretivas com o intuito de atingir os objetivos propostos. Ainda segundo o autor, o controle deve abranger todos os níveis organizacionais e se divide em controles estratégicos, controles táticos e controles operacionais.

Process Owner

O process owner, nada mais é que o dono do processo de negócios. Normalmente, é um gerente que tem essa responsabilidade permanente sobre o processo, diferentemente de um gerente de projetos, cuja responsabilidade termina em dado momento.

O gestor de processos tem algumas atribuições principais, determinadas para garantir que todos os recursos necessários sejam disponibilizados ao longo do processo de negócio, além de avaliar de forma contínua os resultados do processo, viabilizar treinamento adequado de toda equipe e, em conjunto com a equipe, avaliar, redefinir e implementar mudanças que se façam necessárias no processo (Sordi, 2018).

Diversas são as atribuições do gestor de processos, porém, as ações não dependem exclusivamente dele. A visão dos profissionais envolvidos no processo contribuirá de forma significativa para a avaliação e a implementação de mudanças que se fizerem necessárias ao processo, portanto, uma cultura participativa terá grande relevância para o bom andamento do processo e para o bom atendimento das demandas dos clientes.

Gestão de Mudança

A mudança gera desconforto e traz impactos negativos aos processos de gestão organizacionais. Para o BPM a disponibilidade para a mudança é extremamente necessária, já que a ideia é voltar-se para a melhoria contínua dos processos de negócio. Essa característica deve ser “enraizada” em todos os colaboradores da organização, para que o processo de mudança ocorra de forma mais simplificada.

No geral os métodos BPM não têm uma abordagem de gestão de mudanças e isso se dá porque são desenvolvidos com foco em conteúdo e não em mudança de comportamento.

O gestor de processo também atua como gestor de mudanças e/ou gestor de conflitos, pois apesar de gerenciar um processo, a organização ainda continua tendo uma estrutura funcional, o que gera, muitas vezes, conflitos de interesse (Brocke e Rosemann, 2013). Já a equipe de processos de negócio deve ter a capacidade de identificar e compreender o problema e criar soluções para problemas identificados, bem como aproveitar oportunidades que surgirem no processo.

Recursos Tecnológicos (Softwares)

Eles são muitos e podem contribuir para o BPM e todos eles têm como objetivo final entregar relatórios que permitam a identificação de cenários e, por consequência, determinar um plano de melhoria de processo, ou seja, devem apresentar dashboards (painéis de desempenho) para os interessados. Vale lembrar que existem softwares simples e extremamente arrojados (complexos).

São exemplos de softwares BPM o Casewire, Aris Platofrm, WebSphere Business Modeler, Aqualogic BPM Studio, Visio, Bizagi Modeler, Bonita Open Solution, Comindware. Frequentemente surgem novas ferramentas que podem ser incorporadas na modelagem e gerenciamento de processos.

Um sistema BPMS tem como finalidade modelar processo e fluxo de trabalho, definir regras, simular operações de negócios, automatizar processos, acompanhar o desempenho, monitorar e controlar as atividades.

Normalmente é necessário estabelecer interfaces para que os dados necessários sejam fornecidos dentro das regras estabelecidas. Em empresas que utilizam SOA (Service-Oriented Architecture ou Arquitetura Orientada a Serviços), o processo pode ser simplificado por meio de adaptadores, que ajudam a definir a integração, e os sistemas, que fornecem troca de dados entre as aplicações. No geral, os processos de negócio são construídos pela notação BPMN quando se usa um modelo BPMS.

O modelo conceitual do BPMS valoriza os investimentos já realizados em softwares pelas organizações envolvidas com o processo de negócio, diferentemente da estratégia da reengenharia de uma década atrás, que apregoava o descarte e a substituição dos sistemas de informação legados pelo sistema ERP (De Sordi, 2018).

Metas

O estabelecimento de metas deve levar em conta o método SMART – acrônimo de Specific (específico), Measurable (mensurável), Attainable (alcançável), Relevant (relevante) e Timely (temporal). Esse método foi criado por Peter Drucker e tem como objetivo, no momento da construção das metas, realizar questionamentos usando o significado de cada letra:

- **S (Specific):** As metas devem ser específicas, portanto, deve ter clareza nos seus objetivos e ações.
- **M (Measurable):** É importante que sejam mensuráveis, isto é, possíveis de serem medidas.
- **A (Attainable):** Devem ser alcançáveis, ou seja, devem estar dentro da realidade.
- **R (Relevant):** Precisam ser relevantes para a organização, pois não se deve perder tempo com o que não é importante.
- **T (Timely):** Devem ser temporais, isto quer dizer que deve ser possível apurar seus resultados dentro de um período de tempo.

Ao modelar um processo de negócio e pensar no seu processo de controle, Gerenciamento de Processos de Negócio, é necessário que se faça o questionamento indicado sugerido por Drucker para cada KPI planejado. É preciso que os indicadores de desempenho sejam claros, estejam ligados aos objetivos organizacionais e tenham método de apuração definido. O método não foca indicadores simples de serem medidos ou gerenciados, mas sim, a importância de serem medidos e gerenciáveis.

O domínio das ferramentas e técnicas, sem dúvida, é fundamental para o profissional da área de tecnologia da informação. Porém, a compreensão das relações e comportamentos humanos se tornam cada dia mais relevantes para o sucesso da área.

O artigo Avaliação do BPM e BPMS no setor de manutenção de uma IES mostra o passo a passo de toda a cadeia BPM e demonstra a importância e funcionalidades da BPMN e BPMS. Ainda é demonstrada a abordagem prática da aplicação de ferramentas.

Na sequência, o texto Análise Comparativa Entre Ferramentas Gratuitas de Gestão e Automação de Processos (BPMS), traz um comparativo que nos ajuda a entender o que deve ser avaliado na escolha de uma ferramenta BPMS, principalmente as gratuitas. Aproveite a abordagem para aprofundar seus conhecimentos na vantagem das ferramentas e sua aplicabilidade no dia a dia.

O Processo de Engenharia de Requisitos

Engenharia de Requisitos

Conforme Sommerville (2011), engenharia de requisitos é o processo de descrever todas as funcionalidades que um sistema deve possuir, bem como descrever todos os serviços e as restrições de seu funcionamento, refletindo diretamente as determinações dos clientes (usuários do software projetado) e contribuindo, de forma significativa, para o desenvolvimento de um produto final com qualidade.

O objetivo da engenharia de requisitos, para Pressman (2016), é proporcionar a todos os envolvidos no desenvolvimento do sistema uma mesma compreensão por escrito do problema, para isso é utilizada uma série de elementos (artefatos) que garante a qualidade do que foi especificado.

Definir os requisitos de um sistema são fundamentais para o sucesso do software.

O que são os Requisitos de um Sistema?

Os requisitos de um sistema abrangem a função que o software deve ter, a qualidade que ele deve apresentar, as especificações dos serviços que ele deve oferecer, as restrições que deve possuir, as características gerais e as restrições que devem ser atendidas no seu processo de desenvolvimento.

Os requisitos podem evoluir e podem ser modificados no decorrer do desenvolvimento do software, eles não são estáticos e sofrem atualizações constantes, além disso, devem ser documentados para fins de controle. Paula Filho (2019) destaca os seguintes fatores que contribuem para a evolução dos requisitos:

1. Descobrimento de defeitos e inconformidades nos requisitos originais.
2. Falta de detalhamento nos requisitos originalmente elencados.
3. Modificações incontornáveis no projeto (por exemplo, alterações de legislação, moedas, impostos).

Classificação de Requisitos

Um fator de destaque na produção de um requisito é determinar a prioridade que ele deve ter. Essa prioridade possui denominações diferenciadas de acordo com a literatura consultada, e cada empresa pode adotar a sua terminologia. Os requisitos são classificados como: Essencial, importante ou desejável.

- **Essencial:** Um requisito classificado como essencial é de extrema importância (é fundamental) para o software ser executado. Sem esse requisito, o software ficará incompleto. O sistema não poderá ser implantado ou concluído caso um requisito essencial não esteja totalmente realizado.
- **Importante:** Um requisito classificado como importante deve ser realizado, mas não é essencial para que o software seja implantado. Esse requisito será realizado, mas pode ficar em segundo plano, pois é desejável, porém não imprescindível.
- **Desejável:** Um requisito classificado como desejável é aquele que não é imprescindível para o software estar concluído, é algo opcional que pode ou não ser realizado e que até pode ser eliminado no decorrer do desenvolvimento do sistema.

Descrição dos Requisitos

Na engenharia de requisitos é importante fazer uma distinção dos diferentes tipos de descrições dos requisitos conforme afirma Sommerville (2011):

- **Requisitos de Usuário:** Descrevem os requisitos funcionais e os não funcionais do sistema. Utilizando uma linguagem acessível aos usuários do sistema (clientes), demonstram as funções e as restrições que o sistema deverá possuir. Como resultado será produzido um documento que não contém detalhamento técnico do sistema e possui como finalidade a comunicação entre os desenvolvedores e clientes.

- **Requisitos de Sistema:** Especificam detalhes do sistema (apoiados nos requisitos de usuários). Resulta em um documento que pode servir como parte do contrato entre os envolvidos no desenvolvimento do sistema. É o ponto que marca a fase inicial de um projeto.

Tipos de Requisitos

Determinar o tipo de requisito em um sistema pode ser uma tarefa extenuante, e um requisito pode ser classificado inicialmente como de um tipo e, no decorrer no projeto, pode sofrer alterações e receber outra classificação, podendo gerar, ainda, uma série de novos requisitos.

Os requisitos de um sistema podem ser classificado como:

- **Requisitos Funcionais:** Determinam de forma clara e precisa as funcionalidades específicas do que o sistema deve ou não realizar. Eles definem os objetivos específicos do sistema, ou seja, o que ele deve possuir ao final de seu desenvolvimento (Sommerville, 2011). Esse tipo de requisito deverá conter todas as funções e informações fornecidas pelo cliente antes da construção do software. Usamos a sigla RF (Requisito Funcional) seguida de uma numeração para indicar o requisito.
- **Requisitos Não-Funcionais:** Estabelecem restrições sobre as funcionalidades do sistema, por meio do estabelecimento de padrões específicos de desenvolvimento, plataforma, tempos de resposta e restrições de acessos, estão relacionados às qualidades que o sistema deverá apresentar. Possuem a sigla RFN e especificam critérios para qualificar os funcionais. Esse tipo de requisito aponta para questões relacionadas à qualidade, performance, confiabilidade, usabilidade, implementação, etc. Um requisito não funcional pode gerar uma grande quantidade de requisitos funcionais.
- **Requisitos de Domínio:** Determinam as características do domínio do sistema, refletindo as características do sistema, podem estabelecer restrições aos requisitos funcionais ou podem indicar cálculos específicos sobre determinado requisito. Eles descrevem as características e estabelecem ressalvas aos requisitos funcionais, indicando, por exemplo, um cálculo obrigatório para o requisito ser validado. Eles podem ser novos requisitos funcionais ou alguma restrição (complementar) de algum requisito funcional.

Atividades do Processo

De acordo com Pressman (2016), as atividades do processo da engenharia de requisitos envolvem a coleta de informações sobre o software (sistema) a ser realizado, a análise do problema e, em seguida, a descrição dos requisitos, classificação e priorização, sendo que logo após isso é gerada a especificação desses requisitos.

1. Análise do problema.
2. Levantamento dos requisitos.
3. Descrição dos requisitos.
4. Classificação dos requisitos.
5. Priorização e negociação dos requisitos.
6. Especificação dos requisitos.

Todas as atividades que englobam a engenharia de requisitos possuem como finalidade a produção de um documento de requisitos, como afirmam Sommerville (2011) e Pressman (2016), englobando as seguintes etapas:

1. **Concepção:** Determina-se o escopo geral do sistema e todos os envolvidos.

2. **Elicitação:** Faz-se o levantamento dos requisitos funcionais e não funcionais, utilizando técnicas como entrevistas, observação e pesquisa.
3. **Elaboração:** Detalha-se cada requisito (levantado e escrito em linguagem natural) para transformá-los em modelos conceituais (UML – Linguagem Unificada de Modelagem), eliminando erros, esquecimentos, duplicidades e inconsistências.
4. **Negociação:** Identificam-se os requisitos conflitantes, sendo realizada uma negociação entre as partes envolvidas para modificar e/ou eliminar o requisito.
5. **Especificação:** Transformam-se os requisitos ao se observar a visão do sistema (do desenvolvimento da solução).
6. **Validação:** Realiza-se a homologação dos requisitos pelos usuários envolvidos, verificando se todos os requisitos forma atendidos (na visão do cliente).
7. **Gerenciamento:** É a verificação constante (no decorrer dos processos) de que os requisitos estão de acordo com o escopo do projeto e a garantia da rastreabilidade, que é o gerenciamento das eventuais modificações que um requisito pode sofrer.

Os requisitos são a essência de qualquer software. Antes de sair desenvolvendo algum sistema, é necessário criar uma lista de funcionalidades e características que ele deve possuir (esses são os requisitos iniciais), mas não paramos nessa etapa.

Portanto, podemos concluir que o desenho de processos adequado determinará o sucesso do desenvolvimento de sua atividade e, portanto, é necessário focar em uma coleta de dados adequada para que possa atender aos requisitos do cliente de forma satisfatória.

Elicitação, Especificação e Validação de Requisitos

Engenharia de Requisitos

A engenharia de requisitos é um processo que compreende todas as atividades que contribuem para a produção de um documento de requisitos e sua manutenção ao longo do tempo. Os procedimentos básicos de levantamento e análise de requisitos de um sistema, propostos por Sommerville, contém as seguintes tarefas:

- **Concepção ou Compreensão do Domínio:** É estabelecer um entendimento básico sobre o problema a ser resolvido. Todos os stakeholders devem ter a compreensão exata do que será realizado e os limites (do que será e do que não) que serão realizados.
- **Coleta de Requisitos e Classificação dos Requisitos (Elicitação):** São todas as atividades para conseguir elencar o máximo de requisitos dos stakeholders. Os requisitos são classificados em funcionais ou não funcionais.
- **Negociação dos Requisitos:** Nesse processo (já com os requisitos listados) é fundamental determinar se há requisitos em conflitos com outros. Para resolver o conflito, as partes envolvidas devem realizar análises, verificando se todos os requisitos estão de acordo com as exigências dos stakeholders e definindo prioridades, determinando os requisitos mais importantes e que merecem total atenção para o sucesso do projeto.
- **Validação dos Requisitos:** É realizada uma verificação geral dos requisitos, proveniente de um “termo de aceite” em que todas as partes envolvidas (os stakeholders do sistema projetado) concordam e validam os requisitos.
- **Documentação:** Resultado final da engenharia de requisitos, quando é gerada uma documentação com todas as especificações dos requisitos funcionais e requisitos não funcionais, é o principal meio de comunicação entre o analista de sistemas ou engenheiro de software e o cliente.

Processo de Levantamento e Análise de Requisitos

A partir da compreensão do domínio do que será efetivamente realizado no sistema, é realizada a coleta e a classificação de requisitos, com a finalidade de determinar o que realmente será realizado no sistema, classificando para melhor controle. A negociação serve para estabelecer os ajustes necessários e ajudar a determinar o que será prioritário no desenvolvimento. A especificação é o início da documentação dos requisitos, e na validação é realizada uma checagem geral de todos os requisitos, tendo como objetivo a documentação de requisitos.

O levantamento e análise de requisitos, proposto por Sommerville (2011), é um processo de validação continuada. Basta alguma alteração em um requisito para que o ciclo deve ser repetido, de forma iterativa e contínua, até a finalização do projeto.

Elicitação de Requisitos

Na engenharia de requisitos, a elicitação de requisitos é descobrir (extrair de algo ou alguém) o máximo de informações para estabelecer os requisitos de determinado sistema, sendo essa uma das primeiras etapas da engenharia de requisitos (Pressman e Maxim, 2016).

O analista de sistemas não faz a elicitação de requisitos sozinho, esse processo envolve diversas pessoas – todos os envolvidos – conhecidas como stakeholders. De acordo com Pressman e Maxim (2016) a elicitação de requisitos combina elementos para solucionar algum problema e para isso é necessária uma abordagem colaborativa dos envolvidos.

A elicitação de requisitos tem por objetivo conseguir o máximo de requisitos do sistema a ser desenvolvido, destacando as seguintes técnicas:

- **Pesquisa:** Envolve a observação de como funciona a rotina dos processos do sistema e de outros softwares utilizados, visando procurar requisitos que não foram explicitamente solicitados. Envolve também a pesquisa pela tecnologia solicitada.
- **Entrevista:** Geralmente é guiada por um questionário para saber as necessidades que o sistema deverá suprir. Nessa etapa é importante saber ouvir e marcar o máximo de informações obtidas.
- **Reuniões:** Usando técnicas como o brainstorming para descobrir requisitos que ainda não foram determinados e resolver requisitos conflitantes que apareceram nas entrevistas.
- **Documentos:** Coleta de documentos que possam auxiliar na clareza das funcionalidades do sistema, como: Relatórios, planilhas, papéis de controle, cadernos de anotações, etc.
- **Etnografia:** É a observação e análise de como os usuários finais realmente trabalham (diferente do que venham a dizer), gerando requisitos importantes para o sistema.

Especificação dos Requisitos

A especificação de requisitos é o meio de comunicação entre o analista de sistemas e os programadores que desenvolverão o software. É preciso especificar os requisitos de forma que não haja duplicidade de interpretações, pois o programador utilizará a especificação gerada para programar exatamente o que estará na especificação. A especificação de requisitos descreve todas as funcionalidades e suas restrições dos requisitos funcionais e dos requisitos não funcionais (geralmente em tabelas ou documentos específicos) e pode utilizar diagramas de caso de uso para ajudar na comunicação ou ainda fazer uso da prototipagem.

Diagramas de Caso de Uso

Os casos de uso são diagramas que compõem a linguagem unificada de modelagem, conhecida como UML (Unified Modeling Language). Usada de forma simplificada, é uma ótima ferramenta de comunicação nas atividades do desenvolvimento, conforme afirma Sommerville (2011).

Prototipagem

A prototipagem é a criação de uma versão menor do sistema a ser desenvolvido e tem como princípio a verificação de custo-benefício, em que a experiência do usuário é uma parte fundamental do desenvolvimento do protótipo (Paula Filho, 2019).

Negociação e Monitoramento de Requisitos

A negociação de requisitos tem como finalidade desenvolver um plano de projeto que atenda às demandas dos envolvidos (stakeholders) e, ao mesmo tempo, analisar as restrições no que diz respeito ao orçamento, pessoal, tecnologia e/ou tempo, impostas à equipe de desenvolvimento do software (Pressman e Maxim, 2016).

O monitoramento de requisitos é um processo que consiste em garantir que o escopo do software desenvolvido seja realizado. A cada alteração (em um ou mais requisitos) deve-se garantir a rastreabilidade das alterações, utilizando alguma ferramenta de controle, por exemplo:

- Determinar um status do requisito (proposto, em progresso, em alteração, adiado, excluído, aprovado, etc.).
- Criar uma matriz de rastreabilidade, para facilitar o gerenciamento dos requisitos, sendo que nessa matriz deverão constar todos os requisitos, suas dependências (quais requisitos dependem dos requisitos em questão), o status do requisito, quem alterou o requisito, quem aprovou o requisito e, principalmente, as datas que esses fatos ocorreram.

Validação dos Requisitos

O processo de validação dos requisitos determina que a especificação é consistente com a definição dos requisitos, assegurando que os requisitos propostos atenderão às necessidades impostas pelo cliente (Pfleeger, 2004).

O objetivo da validação de requisitos é encontrar erros nos requisitos documentados. Exemplos típicos de erros nos sistemas são inconsistências, contradições, duplicidade, ambiguidades, incompletudes e imprecisões (Pressman e Maxim, 2016).

Checklist

Uma ferramenta que pode auxiliar na garantia da qualidade da validação de requisitos é o checklist, uma lista de perguntas elaborada e que servirá para analisar cada requisito do sistema. Essa técnica visa:

1. A descoberta de erros em vários níveis: Função, lógica, implementação.
2. A verificação se o sistema possui os requisitos especificados.
3. A garantia de que o software desenvolvido foi implementado de acordo com padrões previamente impostos.

O levantamento de requisitos é um processo demorado (e caro), mas um detalhe despercebido pode causar danos gigantescos.

Modelagem de Requisitos

A fim de concluir o entendimento sobre os requisitos, podemos destacar que eles não só descrevem o fluxo de informação que entra e sai de um sistema e a transformação dos dados no sistema, mas descrevem também todas as restrições quanto ao seu comportamento e desempenho, conforme afirma Pfleeger (2004). Os requisitos permitem:

1. A explicação (na visão dos desenvolvedores), o entendimento de como os clientes querem que o sistema funcione.
2. Informar as funcionalidades e atributos que o sistema deverá possuir.
3. Informar a equipe de testes aquilo que deverá ser validado com o cliente.

Existem diversas técnicas de modelagem de requisitos, a primeira, conforme Sommerville (2011), é fazer uma separação entre os requisitos funcionais e requisitos não funcionais, determinando um equilíbrio entre ambos. O ideal é agrupar os requisitos conforme os seus objetivos, suas prioridades e seus tipos. Após o agrupamento dos requisitos funcionais com prioridade essencial (sem esse tipo de requisito, o software não estará apto a funcionar).

Elicitação de Requisitos

A elicitação de requisitos (também conhecida como levantamento de requisitos) procura identificar o problema a ser resolvido e todo pessoal envolvido (stakeholders), procurando combinar a solução dos problemas encontrados, com a negociação (do que será realizado) e finalizando com a especificação dos requisitos. A documentação a ser produzida na fase da elicitação de requisitos pode ser:

- **Lista de Funcionalidades:** Identificadas em entrevistas individuais e/ou reuniões em grupos.
- **Casos de Uso:** Com o auxílio da UML podemos exemplificar ações do sistema.
- **Cenários de Uso:** É uma descrição narrativa textual, em linguagem natural que descreve uma determinada situação de uso do sistema.

Técnicas de Modelagem de Requisitos

- Técnica REMO (sigla em inglês de Requirements Elicitation oriented by business process Modeling).

É utilizada na fase de elicitação de requisitos e permite integrar a modelagem de processos de negócios (no desenvolvimento do sistema) usando a notação BPMN (Businnes Process Model and Notation – Modelo e Notação de Processos de Negócio) com a elicitação de requisitos (Vieira, 2012).

A técnica REMO permite que a extração de requisitos seja retirada dos diagramas de processos de negócios, apoiados por um conjunto de heurísticas. Ela é composta por duas fases:

1. Foca no entendimento do contexto para conhecer o domínio do problema do software (que será produzido). Deverá ser elaborado um documento (pelo analista de sistemas) contendo informações importantes para entender o domínio do software e que são: Problemas e

necessidades, papéis envolvidos nos processos, recursos necessários e disponíveis e diagramas de processos de negócios.

2. Se concentra nos requisitos, na extração e descrição dos requisitos do sistema.

Requisito Gerado a partir da Modelagem de Processos de Negócio

- Linguagem de Modelagem SysML (System Modeling Language).

É open source (código aberto), derivada da linguagem UML e especificada pelo Object Management Group, SysML suporta as seguintes atividades: Especificação, análise, design, verificação e validação de sistemas.

Nesta modelagem podemos reutilizar vários diagramas da UML, e foi criado o diagrama de requisitos, que possui como principal vantagem a demonstração dos requisitos e suas relações. O diagrama de requisitos do SysML é baseado em textos e os relacionamentos entre os requisitos (HAUSE, 2006).

Na figura são ilustrados três requisitos. O requisito “Processo de Matrícula” possui duas dependências, os requisitos “Entrada de Dados” e “Disponibilização das Disciplinas” precisam ser elaborados após a abertura do requisito “Processo de Matrícula”.

- Linguagem UML (Linguagem de Modelagem Unificada).

A UML é amplamente utilizada na modelagem de requisitos. Possui muitos diagramas que podem ser utilizados de forma isolada (um diagrama para representar algo específico) ou podemos usar um dos vários diagramas que compõe a UML, por exemplo: Caso de uso (o mais utilizado para modelar os requisitos), diagrama de sequência, diagrama de atividades.

Cada empresa pode adotar uma ou várias técnicas de modelagem de requisitos, com a finalidade de produzir uma documentação ao final da modelagem.

Documentação da Elicitação de Requisitos

Registra os principais tópicos que dizem respeito diretamente ao que o sistema deverá realizar e determinar em quais condições as soluções serão realizadas, como afirma Sommervile (2011).

O documento gerado pode ser de forma sequencial (em forma de lista ou tópicos), onde os requisitos são descritos sem muita complexidade, porém de forma clara, objetiva e completa. As lacunas desse documento (se existirem e ou na medida que forem encontradas) serão preenchidas durante a fase de especificação de requisitos.

O detalhamento de cada requisito pode ser necessário caso evidencie dúvidas. Na figura, observe o detalhamento do requisito funcional que requer um melhor entendimento, sendo necessária nova documentação.

A documentação produzida nesta fase começa nos registros das reuniões com todo o pessoal envolvido na elicitação de requisitos e começa a ser elaborado o documento de visão do sistema.

Documentação da Especificação de Requisitos

Os requisitos funcionais e os requisitos não funcionais são documentados. Nesta etapa podem ser utilizados diagramas de casos de uso (UML) e realizados protótipos de parte do sistema. Na documentação deverão ser descritos todos os passos das funcionalidades e das restrições do requisito.

A documentação gerada da especificação de requisitos não segue um padrão preestabelecido e as empresas adaptam os formulários da documentação de acordo com suas necessidades internas de desenvolvimento. A figura a seguir ilustra uma configuração básica da documentação da especificação de requisitos.

Outra forma de especificar requisitos é a partir de diagramas de caso de uso. Pressman (2016) afirma que um cenário de uso (ou especificação do caso de uso) é um detalhamento do caso de uso. Esse detalhamento ajuda a modelar o requisito especificado e será a principal forma de comunicação entre a equipe de desenvolvimento, ou seja, entre o analista de sistemas (que fez a modelagem) e o programador (que programará o requisito modelado).

A especificação ou descrição de um caso de uso possui como objetivo informar quais os atores (pessoas ou sistemas relacionados com o sistema) interagem com as funcionalidades específicas que estão sendo modeladas.

Não há um modelo padrão de especificação, é totalmente adaptável às necessidades das empresas, mas é aconselhável que seja escrito de forma simplificado para facilitar seu entendimento. Portanto, podemos notar que existe grande quantidade de formulários e diagramas que ajudam no processo de modelagem de requisitos.

Atualmente, a área de TI pode projetar software de diversos tamanhos e das mais variadas utilizadas, entretanto a engenharia de requisitos sempre estará presente. Um exemplo é o artigo Estabelecimento de Requisitos para Aplicações de Realidade Aumentada, no qual é demonstrado como a engenharia de requisitos pode auxiliar nas aplicações de RA (realidade aumentada).

Paradigma

Na engenharia de software, consideramos um paradigma como um modelo que já foi testado e segue alguns princípios para a resolução de um problema computacional. Há uma grande vantagem em seguir um modelo, pois facilita o desenvolvimento e a compreensão da solução encontrada.

“Um paradigma de programação é um padrão de resolução de problemas que se relaciona a um determinado gênero de programas e linguagens” – Tucker e Noolan (2010).

Historicamente, a busca por padrões foi um processo evolutivo na construção de programas e sistemas.

- **Crise do Software:** Período entre 1968 e 1969 – Busca um padrão de desenvolvimento de software, um paradigma. Utilização de uma estrutura adequada para construção de algoritmos, com o objetivo de melhorar os padrões e processos de programação e facilitar o entendimento pelos programadores quando eles realizassem uma manutenção (alteração).
- **Padrão Estruturado:** 1970 – Paradigma estruturado passou a ser utilizado pelos desenvolvedores. O padrão estruturado era adequado às linguagens existentes.
- **Orientação a Objetos:** Com o surgimento de ambientes gráficos, o crescimento dos sistemas, que tornaram-se cada vez maiores e mais complexos, a crescente necessidade de integração entre sistemas, a solicitação, dia após dia, de melhorias para serem incrementadas nas aplicações existentes. Diante desses fatores, o padrão estruturado se mostrou ineficiente

para algumas aplicações e assim, surgiu o paradigma orientado a objetos, que se tornou muito utilizado a partir de 1997, com a linguagem UML (Unified Modeling Language).

Com o paradigma orientado a objeto surgiu não só um novo padrão para o desenvolvimento de software, mas também uma nova forma de pensar como modelar os problemas do mundo real.

Esse conceito também foi estendido para a linguagem de programação, como a linguagem Smalltalk 80 e posteriormente outras duas importantes linguagens de programação orientadas a objetos, a Eiffel e a Java.

Conceitos Básicos POO (Programação Orientada a Objetos)

Dessa forma, surgiram outros importantes conceitos básicos de orientação a objetos. Entre os conceitos e princípios básicos de POO:

Abstração

A modelagem do objeto inicia pela abstração, que é quando reconhecemos os objetos. Suponha que você ouviu o termo cadeira: Você pensa na ideia de como é uma cadeira, e isso é uma abstração.

Classe

É a representação da abstração do objeto com suas características e comportamentos.

- **Atributos:** Denominações técnicas para as características.
- **Métodos:** Ações ou comportamentos.

Exemplo: Classe Cadeira, atributos: A classe Cadeira tem pés, rodinhas, encosto, assento e cor. Métodos: Ela move o encosto, eleva o assento e anda.

Objeto

É a materialização de forma única de uma classe, e a esta materialização damos o nome de instância da classe. Portanto, um objeto é uma instância de uma classe. Os objetos são únicos e distinguem-se pelos valores dos seus atributos e ações (comportamentos), apesar de pertencerem a uma mesma classe.

Herança

Aplicamos o conceito de herança quando elaboramos uma classe que herda as características e operações de outra classe (classe: Cadeira, herda atributos: Encosto, assento e pés).

Durante o processo de modelagem serão refinadas e construídas subclasses que poderão herdar as características e comportamentos da classe genérica.

- **Superclasse:** É a classe genérica (Cadeira).
- **Subclasse:** É a classe que herda as características da superclasse (CadeiraGiratoria).

Exemplo: Nova classe CadeiraGiratoria, acrescentando a ela o atributo rodas e os métodos andar, girar, mover assento.

Encapsulamento

Na POO, o encapsulamento tem capacidade de tornar a visibilidade das informações e os detalhes da implementação dos métodos de uma classe oculta ou restrita. Isola os atributos da classe e suas ações, garantindo a integridade e a ocultação dos dados e ações. O relacionamento entre as classes passa a ser feito por mensagens.

Exemplo: Você pressiona o comando abrir do objeto controle remoto do portão eletrônico e ele transmite a mensagem para o portão que executa o pedido, mas você não precisa ter conhecimento de como isso é feito.

No diagrama de classe usamos o símbolo “-” (um traço) para indicar que o atributo ou método está encapsulado. Isto significa que somente o objeto da classe pode modificá-los ou acessá-los. Já o símbolo “+” (adição) indica que o atributo ou método é visível por outros objetos, ou seja, é público.

Polimorfismo

Em uma classe, um método (operação) pode aparecer diversas vezes, pois pode ter comportamentos diferentes nas subclasses. Ou seja, objetos de um mesmo tipo de classe que apresentam comportamentos diferentes.

Exemplo: Duas casas têm portões eletrônicos, porém em uma casa o portão abre para cima, e na outra, o portão é de correr na horizontal.

A orientação a objetos tornou-se um paradigma muito utilizado no desenvolvimento de software e você pôde ter contato com os pilares deste paradigma: Abstração, classe, objeto, herança, encapsulamento e polimorfismo.

Desde o início da orientação a objeto, muitos trabalhos têm sido desenvolvidos e outros paradigmas vêm surgindo, portanto, quanto mais você dominar este tema mais preparado você estará para desenvolver boas práticas de análise e desenvolvimento.

Modelo do Processo Unificado

Processo Unificado (PU)

O processo unificado (PU) surgiu para melhorar o desenvolvimento de softwares com o foco na A/POO (Análise e Projeto Orientados a Objetos). Este modelo de desenvolvimento de software é iterativo e adaptativo, permitindo produzir um sistema de grande porte como se fossem vários pequenos sistemas, o que diminui o risco do projeto.

Foco

Segundo Jacobson, Booch e Rumbaugh (2000), o PU é definido por três aspectos chaves:

1. **Dirigido por Caso de Uso:** O caso de uso é o fio condutor do PU, como afirmam Jacobson, Booch e Rumbaugh (2000). Nesta fase devemos conhecer o quê os futuros usuários necessitam e desejam. Devemos nos atentar para que os casos de usos respondam o que cada usuário necessita e não apenas as funções que o sistema precisa ter. Desta maneira o modelo

de caso de uso guiará o desenvolvimento no seu projeto (design), na implementação e nos testes, avançando através de uma série de fluxo de trabalho (workflow).

2. **Centrado na Arquitetura:** Visa dar ao engenheiro de software uma visão abrangente do sistema, como no caso do projeto de um carro, no qual há o design, a mecânica, o projeto elétrico, aerodinâmico, etc. A arquitetura do software também deverá prover ao desenvolvedor estas visões generalizadas, tais como as necessidades dos usuários e objetivos estratégicos da empresa.
3. **Iterativo e Incremental:** A iteração consiste em dividir o projeto em subprojetos menores e o resultado de uma iteração produz um incremento. O processo iterativo no PU é controlado, isso reduz os riscos de aumento de custos com incrementos específicos, os prazos são melhor controlados e também acelera o ritmo de desenvolvimento, pois reconhece um aspecto ignorado: Dificilmente em um sistema complexo, no início, é possível definir totalmente os requisitos e necessidades do usuário.

Ciclo de Vida

O ciclo de vida do PU é uma série de repetições ao longo da vida do sistema, sendo que cada ciclo completo resulta em uma versão do software, por sua vez cada ciclo é composto por 4 fases:

- **Concepção:** Definirá a visão geral do projeto, o escopo e os requisitos iniciais.
- **Elaboração:** É uma visão mais refinada dos requisitos e da arquitetura, análise de riscos e estimativas.
- **Construção:** É o momento de desenvolvimento do sistema, começando pelos elementos mais fáceis, e inicia-se a preparação para a implementação.
- **Transição:** É a fase de implantação do sistema, ou seja, a entrega.

Na figura a seguir o ciclo de desenvolvimento (concepção + elaboração + construção + transição) termina com a entrega de uma versão do sistema, pronta para ser implementada em produção. Todo esse ciclo é composto de muitas iterações, segundo Larman (2007).

Elementos de um Processos

O PU descreve um processo por quatro elementos básicos: Papel (quem), artefato (o quê), atividade (como) e disciplina (quando):

- **Papel:** (Worker, trabalhador). É a identificação das responsabilidades de cada indivíduo (quem faz o quê), o papel do trabalhador naquele momento, o ator da cena. Ao longo do processo, um worker pode ter várias responsabilidades e desenvolver uma série de atividades.
- **Artefato:** É o termo utilizado para identificar qualquer produto de trabalho, seja código, esquema de banco de dados, diagramas, modelos, etc. É o produto que o worker gera.
- **Atividade:** É a tarefa executada pelo worker com o objetivo de produzir ou alterar um artefato.
- **Disciplina:** (Workflow, fluxo de trabalho). É a sequência de atividades que gera um resultado significativo. Os fluxos estão associados a três perspectivas: Dinâmica (tempo), estática (atividades) e boas práticas. Em cada uma encontramos um conjunto de disciplinas.

Workflow (Disciplinas)

Segundo Jacobson, Booch e Rumbaugh (2000), as disciplinas fundamentais, também chamadas fluxo de trabalho do PU são:

- **Modelagem de Negócio:** O objetivo é documentar quais os objetivos de negócio estão envolvidos no processo e são analisados, buscando entender como o negócio deve apoiar os processos associados. Muitos projetos podem optar por não fazer modelagem de negócios. Os casos de uso de negócios são utilizados nessa fase.
- **Requisitos:** Tem por objetivo descrever o que o sistema deve fazer. Tem como base os casos de uso para identificar os requisitos. Nessa fase surge um importante elemento na engenharia de software, os atores (os atores envolvidos devem ser identificados, todos os quais representam os usuários e qualquer outro sistema que venha a interagir com o sistema em desenvolvimento). O foco está nas funcionalidades do sistema e tanto desenvolvedores como o cliente deverão concordar com a descrição dos requisitos.
- **Análise e Projeto:** Estas disciplinas geram os modelos do sistema objetivando as implementações futuras. Em conjunto, essas fases melhoram a compreensão dos requisitos funcionais, definido nos casos de uso. Segundo Larman, “a análise enfatiza a investigação do problema e dos requisitos, em vez de uma solução” (Larman, 2005), ou seja, como será usado e quais serão as suas funções, deixando para a implementação o modo como fazer isso. O projeto dá ênfase à solução conceitual, ou seja, cria os modelos para satisfazer os requisitos. O projeto é para o desenvolvedor e não para o usuário. Consiste em subsistemas bem definidos, com classes estruturadas em pacotes e deverá representar os componentes a serem implementados.
- **Implementação:** O seu foco está direcionado à construção do software, o desenvolvimento do código, a famosa “mão na massa”. Também prepara os primeiros testes, chamados beta teste.
- **Testes:** Nesta etapa deve ser descrito quais os procedimentos a serem avaliados. Segundo as recomendações do documento RUP: Boas Práticas Para o Desenvolvimento de Software (IBM, 2005), os objetivos do teste são: Verificar a interação entre objetos e componentes, verificar se todos os requisitos foram implementados corretamente e identificar e garantir que os defeitos sejam resolvidos antes da implantação do software.
- **Implantação:** Nesta etapa deve ser descrito quais os procedimentos a serem avaliados. O objetivo do fluxo de trabalho da implantação é entregar o produto com êxito para os usuários, e envolve a aceitação forma do software pelo cliente. Entre as atividades do fluxo de trabalho da implantação temos a distribuição, instalação, migração dos dados e de software já existente.

As demais disciplinas estão associadas ao suporte e foram influídas no Processo Unificado Racional (RUP) para suprir disciplinas não contempladas pelo PU, são elas: Gerenciamento de mudanças e configurações, gerenciamento de projeto e ambiente. Basicamente visam a maturidade do projeto.

Na figura a seguir você pode observar como as disciplinas são desenvolvidas ao longo das fases do ciclo de vida do desenvolvimento e nas diversas iterações (minissistemas). Cada ponto de iteração é um marco no PU e finaliza com a entrega de um incremento. Neste ponto é avaliado se os objetivos foram alcançados e, se necessário, ajustes são realizados. Embora para cada iteração todas as disciplinas podem estar presentes, o tempo dedicado a cada uma, muda de iteração para iteração.

O processo unificado foi um marco na engenharia de software, e voltado para o paradigma orientado a objeto, sendo uma característica importante o fato de ser iterativo e incremental, assim o usuário não aguarda a conclusão total do software para iniciar o uso do sistema e, ao término do desenvolvimento, praticamente não há erros.

O principal objeto do PU ou do RUP é o desenvolvimento de um produto de qualidade.

Vale a pena você conhecer o relator da Standish Group sobre alguns fatores críticos de sucesso para projetos de TI.

A indústria de software cresceu e está em todos os negócios e as empresas de softwares precisaram obter alguma certificação para que seus produtos fossem aceitos pelos clientes ou para exportação de seus softwares. O CMMI (Capability Maturity Model Integration) é uma destas certificações e está associado diretamente a melhorar o setor de desenvolvimento. Seus resultados foram tão bons que se expandiu para outros negócios, não somente de software. Para saber mais sobre CMMI, pesquise por CMMI Institute (2020).

A certificação CMMI era apenas empresarial, mas há pouco tempo o CMMI Institute criou uma certificação pessoal, que passou a ser necessária aos membros das empresas que desejam ter os níveis mais altos do padrão CMMI, logo, é bom conhecer esse tema.

Como essa certificação é muito cara, a Softex (Associação Promoção da Excelência do Software Brasileiro), um órgão do MCTIC (Ministério da Ciência, Tecnologia, Inovação e Comunicações) do governo brasileiro, desenvolveu o MPS-BR, uma certificação para a melhoria do processo de software brasileiro. É uma certificação reconhecida internacionalmente com um custo viável para as empresas nacionais. Pesquise esse programa de melhoria para saber mais.

Métodos Orientados a Objetos

UML (Linguagem Unificada de Modelagem)

Com a introdução do paradigma orientado a objeto, surgiu a necessidade de métodos específicos voltados para análise e projetos orientados a objetos (A/POO).

A UML (Linguagem Unificada de Modelagem ou Unified Modeling Language) foi a linguagem que unificou os métodos para modelagem de software orientado a objetos. Essa linguagem permitiu aos desenvolvedores e engenheiros de software um padrão único para elaborar a modelagem de programas e projetos de desenvolvimento.

Diagramas

A UML faz uso de uma linguagem gráfica, o que nos permite visualizar com mais facilidade os objetos e suas interações (relacionamentos), bem como construir, especificar e documentar os artefatos gerados por um software.

A versão 2.5 apresenta 14 diagramas subdivididos em duas categorias: Diagramas de estrutura e diagramas de comportamento.

Diagramas de Estrutura

Representam as estruturas estáticas do sistema por meio de objetos, relações e atributos. Eles proporcionam uma visão de arquitetura e os aspectos funcionais globais do sistema, além de conceitos de implementação. Seu foco não são os detalhes nem o tempo, pois seu objetivo é mostrar como os objetos se relacionam.

Diagramas de Comportamento

Representam os aspectos dinâmicos do sistema, que são as mudanças que ocorrem no sistema com o passar do tempo, por meio da comunicação entre os objetos e das mudanças de estados internos e/ou externos no sistema. Entre os diagramas de comportamento há uma subdivisão, denominada diagrama de interação, em que encontramos quatro diagramas.

A existência de tantos diagramas tem como objetivo permitir visões múltiplas do sistema a ser modelado, como afirma Guedes (2011).

Vamos conhecer um pouco mais sobre alguns dos diagramas mais importantes da UML:

- **Diagrama de Caso de Uso:** O diagrama de caso de uso fornece uma visão geral dos objetivos que os usuários (os atores) desejam alcançar utilizando o sistema. Os elementos mais importantes são os atores, os relacionamentos e o fluxo de eventos.
- **Diagrama de Classe:** O diagrama de classe, segundo Guedes (2011), "define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe tem, além de estabelecer como as classes se relacionam e trocam informações entre si".

A representação da classe é um retângulo dividido em três partes: O nome da classe, os atributos e os métodos.

Um mecanismo de extensibilidade da UML bastante utilizado nos diagramas de classe são os estereótipos. A finalidade de um estereótipo é permitir classificar elementos do diagrama que tenham algo em comum entre si. Os estereótipos podem ser definidos pelo desenvolvedor ou predefinidos. Os estereótipos predefinidos são nativos na linguagem. A UML dispõe de 50 estereótipos predefinidos, segundo a especificação OMG (2017), mas os estereótipos predefinidos mais comuns são três:

- **«entity»:** Estereótipo identidade, que identifica classes de persistências. Essas classes armazenam dados recebidos pelo sistema.
- **«boundary»:** Estereótipo fronteira, o qual identifica uma classe de fronteira. Essas classes servem de comunicação entre atores externos e o sistema.
- **«control»:** Estereótipo de controle. Esse estereótipo geralmente é formado por classes que indicam regras de negócio.

As classes relacionam-se entre si, e os tipos de relacionamentos possíveis especificados na UML (OMG, 2017) são:

- **Dependência:** É o tipo de relacionamento mais fraco entre duas classes, chamado de relação semântica entre duas classes, na qual uma alteração na classe independente pode afetar a classe dependente.
- **Associação:** Este é o tipo de relacionamento mais comum, e indica que a classe A tem uma relação com a classe B. É um tipo de relacionamento genérico.
- **Agregação:** É uma associação específica, em que a classe filha pode existir independentemente da classe pai.
- **Composição:** É uma associação específica em que, se o objeto da classe pai é destruído o outro objeto associado, não fará sentido a existência da classe filha.
- **Generalização/Especialização:** Indica que a classe filha herda as características da classe pai, também conhecida como especialização da classe.
- **Multiplicidade:** Indica quantas instâncias dos objetos estão envolvidos na associação.

Diagrama de Sequência

O diagrama de sequência mostra a interação entre os participantes do cenário ao longo da vida. Ele apresenta, como o próprio nome diz, a sequência dos passos realizados pelos objetos.

Diagrama de Atividade

O diagrama de atividade tem por objetivo descrever os passos que devem ser seguidos para a execução de uma determinada atividade. Esse diagrama assemelha-se muito com as funções de um fluxograma.

Os elementos básicos do diagrama de atividade são: Ações (atividades), sentinela (desvios), estados inicial e final, barra de bifurcação e barra de junção.

Diagrama de Máquina de Estados

Esse diagrama visa mostrar a transição de um estado a outro dos objetos do sistema. É muito semelhante ao diagrama de atividade.

As transições internas de estado são três:

- **Entry:** Indica ações internas do estado executada pelo objeto ao assumir um estado.
- **Do:** São ações internas do estado executadas durante o período que o objeto se encontra em um estado.
- **Exit:** Neste caso, são ações executadas pelo objeto quando ele sai de um estado.

Vimos os principais diagramas propostos pela UML, que trouxeram para engenharia de software o instrumental para modelar os programas e projetos de acordo com o paradigma orientado a objeto.