ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

- **#OBJETIVOS DA DISCIPLINA**

- O estudante deverá ser capaz de criar uma documentação completa de um produto de software, através de exercícios práticos na criação de artefatos: SWOT, 5W2H, EAP, TAP, BPMN, modelagem de diagramas, preenchimento de matrizes de rastreabilidade e preenchimento de documentação de requisitos.

- E também executar as melhores práticas de engenharia de software que apoiem o desenvolvimento estratégico de soluções com qualidade.

ENGENHARIA DE SOFTWARE

- **#EMENTA**

- Enfatizar a importância do conhecimento técnico que os profissionais da área de T.I. devem ter com relação às ferramentas da Engenharia de Software.

- Apresentar e aplicar de forma prática e detalhada todos os aspectos relacionados ao desenvolvimento completo do software, englobando, entre outros tópicos, o levantamento e especificação de requisitos, as modelagens (BPMN, caso de uso, diagramas, demais), análise de portabilidade, planejamento, desenvolvimento e documentação de todas as etapas.

ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

Objetivo: Proporcionar ao aluno a oportunidade de participar, conhecer e aplicar na prática os recursos da Engenharia de Software na construção de uma solução de T.I.

Justificativa: Preparar o aluno para o mercado de trabalho, com maturidade nos processos de desenvolvimento de um software.

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

Características: Interdisciplinares

Cuidados: Conhecer a regra de negócio do cliente; estar atualizado sobre os conceitos (Requisitos, GP, IHC); atenção com (planejamento, organização, comprometimento, pés no chão, seriedade e profissionalismo).

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

Cronograma: Deve ser criado conforme as definições no EAP e TAP;

Artefatos: O aluno deverá apresentar: SWOT; 5W2H; EAP; TAP; BPMN; Levantamento de necessidades; Levantamento de dados / requisitos; modelagem de Caso de Uso; diagramas (atividade, estado e componentes); matriz de rastreabilidade; Portabilidade;

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

Cliente: A coordenação do curso indicará qual será a empresa/organização;

Então, bom trabalho!

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

1º Passo:

Estudar a missão, visão e valores da empresa ‘cliente’, bem como das empresas no mesmo segmento de mercado.

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

2º Passo:

- Analisar a viabilidade para desenvolvimento do projeto: análise econômica; viabilidade; inovação; Onde o produto será utilizado?; Qual é o impacto do software na sociedade?; Quando desenvolver para Mobile, Web ou desktop?; Como desenvolver?; O que o software deverá produzir?; O que levar em consideração ao desenvolver um software? (Layout, Padronização, LP, BD, Redes, Segurança, Qualidade, Público, Plataforma, SO, Multiusuário, Pensar no hoje e no amanhã).

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

- **3º Passo:**

- Elaborar uma quantidade de questionamentos sobre a aplicação do projeto e esclarecê-los através de entrevistas e etnografia com os profissionais ligados à área da pesquisa. Isto proporcionará um melhor entendimento da regra de negócio, o que facilitará a criação do BPMN.

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

- **4º Passo:**

- Preencher os documentos: SWOT, 5W2H, EAP e TAP. Como sugestão, estudar alguns artigos sobre o tema do projeto, isto ajudará na construção dos objetivos, justificativa, etc.

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

- **5º Passo:**

- Criar a modelagem de BPMN, de acordo com os relatos do cliente, funcionário da área ou pessoa que conheça o processo completo. O BPMN é uma modelagem processual, ou seja, rotina de execução. Utilizar corretamente todos os artefatos, o que facilitará a construção do diagrama de Caso de Uso.

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

- **6º Passo:**

- Criar o diagrama de Caso de Uso, tendo como base o layout do BPMN. Neste 'passo' irão surgir inúmeras dúvidas, o que necessitará do preenchimento do documento de levantamento de necessidades e também do documento de levantamento de dados / requisitos.
- Com estes dois documentos preenchidos corretamente, a criação do Caso de Uso se tornará totalmente 'sistêmica', ou seja, representará a ação de todos os atores e seus processos. Vale ressaltar, que haverá muitas alterações no BPMN .... esta é uma das funções da Engenharia de Software.
- Neste passo, cuidado com a utilização do Include e Extend, pois um equívoco poderá comprometer toda a funcionalidade do projeto.

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

- **7º Passo:**

- Neste passo, o engenheiro de software deverá identificar na modelagem de Caso de Uso os processos de tomada de decisão, e assim, detalhar os mesmos através de novos diagramas: (atividade, estado e componentes) – o que proporcionará uma visão minuciosa do funcionamento do processo.

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO - Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

- **8º Passo:**

- O quanto é importante para o desenvolvedor um documento como as matrizes de rastreabilidade. O engenheiro deverá preencher o documento com muita atenção, porque serão representados os requisitos organizacionais, ou seja, requisitos que proporcionam a funcionalidade através de execuções obrigatórias. No documento, deverão ser inseridos apenas os requisitos e casos de uso que possuem Regra de Negócio.

ENGENHARIA DE SOFTWARE

#GUIA DO PROJETO

Fonte: PMBOK (Project Management Body Of Knowledge) = (Guia do Conhecimento em Gerenciamento de Projetos)

ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

1.Métodos de Conversação:

- O método de conversação fornece um meio de comunicação verbal entre duas ou mais pessoas. Sendo uma forma natural de expressar as necessidades, ideias e responder às perguntas, é bastante eficaz para identificar e compreender as necessidades do entrevistado. Proporciona a comunicação verbal entre um ou mais participantes e ajuda a comunicação eficaz com os mesmos. Esses métodos fornecem a maneira natural de expressar as necessidades e as ideias e identificar os requisitos do produto.

1. **1-Entrevistas (Interviews):** A entrevista é uma das técnicas tradicionais mais simples de utilizar e que produz bons resultados na fase inicial de obtenção de dados. Convém que o entrevistador dê espaço ao entrevistado para esclarecer as suas necessidades. É uma discussão do projeto desejado com diferentes grupos de pessoas.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

Fatores positivos:

1) Com um plano geral bem elaborado, o analista terá facilidade em descobrir que informação o usuário está mais interessado e usar um estilo adequado ao entrevistar;
2) Poder alterar o curso da entrevista de forma a obter informações sobre aspectos importantes que não tinham sido previstos no planejamento da entrevista;
3) Poder alterar a ordem sequencial das perguntas;
4) Poder eliminar perguntas anteriormente planejadas;
5) Poder incluir perguntas que não estavam na programação da entrevista;
6) Poder motivar o entrevistado no decorrer do processo;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

Fatores negativos:

1) Podem ocorrer desvios de curso, no decorrer da entrevista;
2) Consumir mais tempo e recursos com sua realização;
3) Tratamento diferenciado para os entrevistados;
4) É necessário ter um plano de entrevista para que não haja dispersão do assunto principal e a entrevista fique longa, deixando o entrevistado cansado e não produzindo bons resultados;
5) O usuário tem dificuldade de concentração em reuniões muito longas;
6) O entrevistado pode não saber expressar corretamente suas necessidades ao analista.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

1.2-WorkShop: Trata-se de uma técnica de elicitação em grupo usada em uma reunião estruturada. Devem fazer parte do grupo uma equipe de analistas e uma seleção dos stakeholders que melhor representam a organização e o contexto em que o sistema será usado, obtendo assim um conjunto de requisitos bem definidos.

Fatores positivos:

1) Obtêm um conjunto de requisitos bem definido;
2) Trabalho em equipe tornando o levantamento de requisitos mais eficaz;
3) Baixo custo e resposta relativamente rápida;
4) Tempo de obtenção de informações é reduzido.

Fatores negativos:

1) Por ser realizado por convocação por dia e horário, pode ocasionar problemas no presencial dos stakeholders;
2) Não abre caminho para ideias externas além da equipe de analistas; Dados excessivamente agregados.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

1.3-BrainStorming: É utilizado normalmente em workshops. Após os workshops serão produzidas documentações que refletem os requisitos e decisões tomadas sobre o sistema a ser desenvolvido. Seu objetivo é uma apresentação do problema/necessidade a um grupo específico, requerendo assim soluções.

Fatores positivos:
1) Várias pessoas pensam melhor do que uma (grupo pensante);
2)Rompe a inibição de ideias;
3) Generaliza a participação dos membros do grupo.

Fatores negativos:
1)Disponibilidade de todos pode inviabilizar o levantamento de dados.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

1.4-Questionários: Diferente da entrevista, essa técnica é interessante quando temos uma quantidade grande de pessoas para extrair as mesmas informações. As questões são dirigidas por escrito aos participantes com o objetivo de ter conhecimento sobre opiniões das mesmas questões. São autoaplicáveis pois o próprio informante responde.

Fatores positivos:
1) Atinge um grande número de pessoas; Menores custos;
2)Permite que os participantes respondam no momento em que acharem convenientes;
3) Questões padronizadas garantem uniformidade.

Fatores negativos:
1) Não há garantia de que a maioria dos participantes responda o questionário;
2) Os resultados são bastantes críticos em relação ao objetivo, pois as perguntas podem ter significados diferentes a cada participante questionado.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

1.5-Grupo Focal (Focus Group): É um grupo de discussão informal e de tamanho reduzido (até 12 pessoas), com o propósito de obter informação qualitativa em profundidade. As pessoas são convidadas para participar da discussão sobre deter- minado assunto.

Fatores positivos:
1) Baixo custo, resposta rápida e Flexibilidade;
2) Obtêm informações qualitativas em curto prazo;
3) Eficiente para esclarecer questões complexas no desenvolvimento de projetos;

Fatores negativos:
1) Exige facilitador/moderador com experiência para conduzir o grupo; Não garante total anonimato;
2) Depende da seleção criteriosa dos participantes;
3) Informações obtidas não podem ser generalizadas.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos
Fonte: Bruno Conde Perez Brum e Leandro Pena

2.Métodos de Observação:
- Utilizado para a compreensão do domínio da aplicação, observando as atividades humanas.

2.1-Etnografia (Ethnographic Study): É uma análise de componente social das tarefas desempenhadas numa dada organização. É utilizado para desenvolver um entendimento completo e detalhado.

Fatores positivos:
1) Capacidade de observar o comportamento do ambiente, gerando maior profundidade no conhecimento.
2) Apoia-se no comportamento real;
3) Permite uma abordagem integral.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos
Fonte: Bruno Conde Perez Brum e Leandro Pena

2.1-Etnografia (Ethnographic Study): É uma análise de componente social das tarefas desempenhadas numa dada organização. É utilizado para desenvolver um entendimento completo e detalhado.

Fatores negativos:
1) Dificuldades para analisar e interpretar situações;
2) A amostra pode ser reduzida;
3) Requer treinamento especializado;
4) As observações podem ter uma interpretação complicada

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

2.2 Observação (Observation): A técnica resume-se em visitar o local em foco com a finalidade de observação do mesmo. Permitindo assim, coletar informações de acordo com o cotidiano das operações e execução dos processos diários do local.

Fatores positivos:
1) Capaz de captar o comportamento natural das pessoas;
2)Nível de intromissão relativamente baixo;
3) Confiável para observações com baixo nível de inferência.

Fatores negativos:
1)Polarizada pelo observador;
2)Requer treinamento especializado;
3)Efeitos do observador nas pessoas;
4) Não comprova/esclarece o observado; 5)Número restrito de variáveis.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos
Fonte: Bruno Conde Perez Brum e Leandro Pena

2.3 Protocolo de Análise (Protocol Analysis): Análise de protocolo é uma forma de levantamento de requisitos no qual o analista analisa as partes interessadas quando estão envolvidas em algum tipo de tarefas.

Fatores positivos:
- 1) Processo de extração de registro de tarefas via áudio, vídeo ou notas escritas.

Fatores negativos:
- 1) o analista deve ter conhecimento sufi- ciente sobre domínio atual, a fim de compreender melhor as tarefas.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

- **3. Métodos Analíticos:**
- Conjunto de técnicas para analise de documentação e conhecimento existentes com o intuito de adquirir requisitos através do levantamento de informação pertinentes ao sistema a ser especificado, como por exemplo, domínio do negócio, fluxos de trabalho e características do produto.

Fatores positivos:
1)O estudo do conhecimento de especialistas leva a um processo sucessivo de aumento de maturidade e qualidade;
2)Reutilização de informação já disponível salva tempo e custo;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

- **3. Métodos Analíticos:**
- Conjunto de técnicas para analise de documentação e conhecimento existentes com o intuito de adquirir requisitos através do levantamento de informação pertinentes ao sistema a ser especificado, como por exemplo, domínio do negócio, fluxos de trabalho e características do produto.

Fatores negativos:
1)Requer dados empíricos, documentação e a opinião de especialistas, e sem estes, não é possível identificar os requisitos;
2)Podem levar a restrição da visão do produto final;
3)Lida com informação antiga, e com isso pode levar a replicação de erros existentes;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

3.1-Reuso de Requisitos: Estudo e reutilização de especificações e glossários referente a projetos de sistemas legados ou sistemas de mesma família (com funcionalidades de negócio similares

Fatores positivos:

1) Economia de tempo e dinheiro: Estudos tem mostrado que sistemas similares podem reutilizar acima de 80% de seus requisitos; Pode levar a uma reutilização adicional de outros itens em outras atividades do ciclo de vida de desenvolvimento (ex.: reuso do design de componentes já existentes, testes e código fonte);

2) Redução de risco: Requerimentos reutilizados tem uma chance maior de serem compreendidos pelos stakeholders visto que já são conhecidos de certa forma;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

3.2-Estudo de Documentação / Analise de Conteúdo: Estudo e reutilização de documentação de diferentes naturezas, para a identificação de requisitos a serem implementados no sistema que se está modelando. Uma grande variedade de documentação pode ser analisada incluindo estrutura organizacional da empresa, padrões de mercado, leis, manuais de usuário, relatório de pesquisas de mercado, glossário de termos de negócio, etc.

Fatores negativos:

1)Documentos com problemas podem levar a uma falha na definição dos requisitos;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

3.3-Laddering: É um método de entrevistas estruturadas, um-a-um, utilizado para o levantamento de conhecimento (o que é importante e por que) de especialistas, e que consiste na criação, revisão e modificação da hierarquia de conhecimento dos especialistas geralmente na forma de diagramas hierárquicos (ex.: diagrama em árvore).

Fatores positivos:

1) Cobre um amplo domínio de requisitos;
2) Necessita de menos tempo para a preparação e execução das sessões de levantamento;
3) Necessita de menos experiência para a execução das sessões de levantamento;
4) Provê um formato padrão que é adaptável para a automação computadorizada;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

3.3-Laddering: É um método de entrevistas estruturadas, um-a-um, utilizado para o levantamento de conhecimento (o que é importante e por que) de especialistas, e que consiste na criação, revisão e modificação da hierarquia de conhecimento dos especialistas geralmente na forma de diagramas hierárquicos (ex.: diagrama em árvore).

Fatores negativos:

1) Não é capaz de extrair todos os tipos de requisitos;
2) Necessita da execução combinada de outras técnicas de levantamento de requisitos para sua complementação em determinados domínios;
3) Não é compatível com todo e qualquer domínio de requisitos, sendo necessário a verificação de sua adequação ao levantamento a ser feito;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

3.4-Sorteio de Cartões: Utilizado para capturar informações e ideias sobre estrutura de requisitos de especialistas de domínio. Neste método um conjunto de cartões é distribuído em um grupo de stakeholders onde cada cartão é impresso com a descrição das entidades do domínio.

Fatores positivos:

1) Ajuda os stakeholders a levantar os conceitos do domínio e distinguir entre problemas de alto e baixo nível;
2) O resultado do método pode ser utilizado como insumo para outros métodos de levantamento de requisitos;

3.5-Repertory Grid: Método onde os stakeholders são questionados sobre atributos e valores destes, referentes a uma série de entidades. Com esta informação é montada uma matriz de entidade X atributo.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

4.Métodos Sintéticos:

- Algumas vezes em projetos complexos um único método de levantamento de requisitos não é suficiente para capturar os requisitos detalhadamente. Para solucionar este problema os analistas de requisitos tentam utilizar diferentes métodos de levantamento de requisitos. Por exemplo, em alguns casos é utilizado o método de entrevista antes de se fazer um estudo etnográfico. Ao invés de utilizar a combinação de diferentes técnicas de levantamento de requisitos, é possível utilizar métodos sintéticos, que são formados pela combinação das outras técnicas em uma única.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

4.1-Sessões JAD/RAD: Consiste em workshops e sessões de grupo nos quais stakeholders e analistas de requisitos se encontram para discutir as características desejadas do produto. Seu objetivo é envolver todos os stakeholders importantes no processo de levantamento, através de reuniões estruturadas e com foco bem definido. Depende diretamente do grau de envolvimento dos stakeholders bem como do líder das sessões JAD.
O processo JAD consiste em três fases principais: customização, sessões e agrupamento.
Na customização, o analista prepara as tarefas para as sessões como organizar os times, preparar o material, etc. Na fase de sessões, o analista marca uma ou mais reuniões com os stakeholders. No inicio da sessão JAD o engenheiro de requisitos provê uma visão genérica sobre o sistema e a discussão com os stakeholders continua até o fim do levantamento de requisitos. Na fase de agrupamento todos os requisitos levantados nas fases anteriores são convertidos em documentos de especificação de requisitos.

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

4.1-Sessões JAD/RAD:

Fatores positivos:
1) As discussões que ocorrem na fase de sessões são altamente produtivas porque resolvem dificuldades entre as partes enquanto se dá o desenvolvimento do sistema para a empresa;
2) Melhor aplicado para grandes e com- plexos projetos;

Fatores negativos:
1)Somente projetos que possuem pelo menos uma das características abaixo podem utilizar o JAD:- Possuir alto número de stakeholders responsáveis por departamentos Cross na empresa;- Primeiro projeto na empresa o qual é considerado crítico para o futuro da mesma;
2) Requer mais recursos se comparado à métodos tradicionais;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

4.2 Prototipação: Utilizado no estágio inicial do projeto. Ajuda aos stakeholders a desenvolver uma forte noção sobre a aplicação a qual ainda não foi implementada, que através da visualização da mesma eles podem identificar os reais requisitos e fluxos de trabalho do sistema. É muito utilizado quando os stakeholders são incapazes de expressar os seus requisitos ou se os mesmos não têm nenhuma experiência com o sistema.

Fatores positivos:

1) Permite alcançar um feedback antecipado dos stakeholders;
2) Redução de tempo e custo de desenvolvimento devido a detecção dos erros em uma fase inicial do projeto;
3) Prove alto nível de satisfação dos usuários devido a sensação de segurança ao ver algo próximo do real;

Fatores negativos:

1) Demanda um alto custo de investimento, em relação à outros métodos, para ser realizado;
2) Demanda um tempo maior para sua realização devido a complexidade do sistema e a limitações técnicas;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

4.3-Questionário de Ambiente: Permite aos analistas o real entendimento das necessidades dos stakeholders com a coleta detalhada de informações através de observação e interação com as pessoas no ambiente de trabalho. Alguns profissionais são escolhidos e acompanhados a fundo para o completo entendimento de suas práticas de trabalho.

Fatores positivos:
1) Permite um levantamento profundo e detalhado das necessidades dos stakeholders;
2) Pode ser utilizado para resolver problemas extremamente complexos;

Fatores negativos:
1) Dependendo dos processos de trabalho, necessita de uma grande quantidade de tempo e pessoas para ser executado;

ENGENHARIA DE SOFTWARE

#Técnicas de Elicitação de Requisitos

- Fonte: Bruno Conde Perez Brum e Leandro Pena

4.4-Storyboards: São sessões interativas que descreve uma sequência de atividades e eventos para um caso em específico para um processo genérico que é esperado que o sistema automatize.

Fatores positivos:

1) Método muito eficiente no esclarecimento de requisitos relacionados a processos, fluxos de dados e tarefas;
2) Método relativamente barato de ser executado;

ENGENHARIA DE SOFTWARE

Gestão participativa = SWOT

Matriz SWOT (Strengths, Weaknesses, Opportunities e Threats)

Fatores Internos: Pontos Fortes e Pontos Fracos.

A análise deve acontecer no presente, na situação a qual o processo se encontra no momento.

Fatores Externos: Oportunidades e Ameaças.

A análise deve acontecer no futuro, no que potencialmente poderia acontecer se a oportunidade ou ameaça se concretizasse.

Finalidades da Análise SWOT

- Avaliar os Ambientes Internos e Externos, proporcionando subsídios para a formulação de estratégias e otimizar seu desempenho.
- Identificar os pontos fortes e fracos, assim como as oportunidades e ameaças dos quais o processo está exposto.

Gestão participativa = SWOT

Matriz SWOT (Strengths, Weaknesses, Opportunities e Threats)

Definição do ambiente interno:

- O ambiente interno propõe a identificação dos pontos fortes (strengths) e também dos pontos fracos (weaknesses) em relação aos concorrentes e ao mercado.
- Nesta etapa devem ser estudados o contexto da empresa e as ações a serem realizadas.
- É importante considerar que toda característica como força ou fraqueza é altamente relativa e alterável, podendo ser enquadrada na medida do seu comportamento.

Definição de ambiente externo:

- A análise externa tem como objetivo a identificação das oportunidades e ameaças que num determinado momento se colocam diante do processo.
- Por isso, é necessário haver uma prevenção por parte dos gestores em relação aos impactos positivos e negativos que o processo possa a vir receber.
- Todas as previsões efetuadas possuem reflexo natural sobre o plano estratégico.

Gestão participativa = 5W2H

A planilha **5W2H** é uma ferramenta administrativa que pode ser utilizada em qualquer empresa a fim de registrar de maneira organizada e planejada como serão efetuadas as ações, assim como por quem, quando, onde, por que, como e quanto irá custar para a empresa.

Gestão participativa = 5W2H

Podemos dizer que o **5W2H** se trata de uma ferramenta administrativa que pode ser utilizada por toda e qualquer empresa. Sua análise possui a finalidade de auxiliar na elaboração de planos de ação, como uma espécie de check-list que aumenta a clareza do colaborador sobre suas atividades.

Gestão participativa = 5W2H

Toda empresa precisa de uma boa gestão para crescer e se desenvolver com sucesso, e o **método 5W2H** é uma ferramenta que auxilia o empreendedor na hora de organizar suas ideias. ... Este método se destaca das demais metodologias de gestão por sua simplicidade na hora da elaboração de um plano estratégico.

Gestão participativa = 5W2H

O **5W2H** é um **formulário** para execução e controle de tarefas onde são atribuídas as responsabilidades e determinado como o trabalho deverá ser realizado, assim como o departamento, motivo e prazo para conclusão com os custos envolvidos.

Gestão participativa = 5W2H

O **5W2H** é uma ferramenta de gestão muito utilizada para **planejamento** e acompanhamento de atividades necessárias para o atingimento de um resultado desejado. O **plano de ação** permite o acompanhamento da execução das atividades mais importantes para se atingir determinados objetivos e metas.

Gestão participativa = 5W2H

O método serve para organizar as práticas de investigação, consistindo na seleção das técnicas (conjuntos de procedimentos destinados a produzir certos resultados ao nível da recolha e tratamento da informação de pesquisa), no controle da sua utilização, permitindo, ainda, a integração dos resultados parciais obtidos.

Gestão participativa = 5W2H

Alcançar grandes resultados, ascender na carreira, realizar metas e objetivos com efetividade. ... Nessa perspectiva, elaborar uma planilha 5W2H é uma forma de separar as etapas de elaboração da execução, obtendo um estudo mais detalhado de todas as atividades necessárias para atingir o objetivo.

Gestão participativa = 5W2H

Como utilizar?

Antes de utilizar o **5W2H** é preciso que você estabeleça uma estratégia de ação para identificação e proposição de soluções de determinados problemas que queira sanar. Para isso pode-se utilizar de brainstorm para se chegar a um ponto comum. É preciso também ter em conta os seguintes pontos:

Gestão participativa = 5W2H

Tenha certeza de estar implementando ações sobre as causas do problema, e não sobre seus efeitos;

Tenha certeza que suas ações não tenham qualquer efeito colateral, caso contrário deverá tomar outras ações para eliminá-los;

É preciso propor diferentes soluções para os problemas analisados, certificando-se dos custos aplicados e da real eficácia de tais soluções.

Gestão participativa = 5W2H

5W

WHAT – O QUE SERÁ FEITO (ETAPAS)?

WHY – POR QUE SERÁ FEITO (JUSTIFICATIVA)?

WHERE – ONDE SERÁ FEITO (LOCAL)?

WHEN – QUANDO SERÁ FEITO (TEMPO)?

WHO – POR QUEM SERÁ FEITO (RESPONSABILIDADE)?

2H

HOW – COMO SERÁ FEITO (MÉTODO)?

HOW MUCH – QUANTO CUSTARÁ FAZER (CUSTO)?

Gestão participativa = 5W2H

QUESTÃO PROBLEMA – 1 = _____							
Ação	(What?)	(Why?)	(Where?)	(When?)	(Who?)	(How?)	(How much?)
1							
2							
3							

Gestão participativa = 5W2H

Ao planejar determinada atividade gerencial, você deve responder às 7 perguntas citadas no slide anterior com clareza e objetividade. Logo após, você deverá elaborar uma tabela explicativa sobre tudo o que foi planejado.

Gestão participativa = 5W2H

What – O que será feito?

O primeiro questionamento de um plano estratégico por meio do 5W2H é o que será feito ou planejado na empresa. Como objetivo temos em nosso exemplo a intenção de aumentar o número de clientes da empresa. Nesse caso, a meta a ser concluída é exatamente essa.

Gestão participativa = 5W2H

Why – Por que será feito?

Considerando esse segundo questionamento, podemos resumir que a meta deverá ser concluída para que a empresa tenha inúmeros benefícios, como melhor posicionamento frente ao mercado, maior lucratividade, expansão do negócio, mais envolvimento com seus clientes, reconhecimento na área de atuação, entre outros.

Gestão participativa = 5W2H

Where – Onde será feito?

Em nosso exemplo podemos dizer que há várias opções de onde esse planejamento estratégico será aplicado. Se a empresa atua apenas em uma região, talvez a melhor opção é atrair mais clientes daquela mesma região, principalmente se a modalidade de negócio não permitir envolvimento com outras regiões do país ou do mundo. Mas dependendo da modalidade de negócio, do porte da empresa e das intenções, é possível alcançar novos clientes em outras regiões do país ou até mesmo do mundo.

Gestão participativa = 5W2H

When – Quando será feito?

Nesta etapa é definido o prazo para que esse objetivo será realizado. No caso do nosso exemplo, podemos definir um limite de 10 meses para que o plano de ação seja totalmente concluído.

Gestão participativa = 5W2H

Who – Por quem será feito?

Ao chegar nesta etapa, é necessário avaliar novamente o que será feito, quando, onde, entre outras características, para definir quais serão as pessoas envolvidas nesse objetivo. Dependendo até mesmo de como o objetivo será encarado as pessoas envolvidas com essa ação também poderão variar. No caso da expansão da empresa em relação ao número de clientes, talvez seja necessário adotar mais funcionários para atendimento, ou deixar os esforços dessas aplicações para a equipe de marketing, entre outros setores e funcionários.

Gestão participativa = 5W2H

How – Como será feito?

Considerando tudo que já definimos até aqui, podemos dizer que o plano de ação poderá ser executado por mais de um setor. O marketing é o setor que mais influenciará nessa questão de conquistar mais clientes, por isso, podemos concluir que a aplicação desse objetivo poderá ser feita por meio de estratégias de marketing voltadas para a atração de novos clientes.

Gestão participativa = 5W2H

How much – Quanto irá custar?

Por fim, é definido o valor de todo esse planejamento e ações relacionadas ao objetivo traçado. Em nosso exemplo, podemos definir diversos valores considerando as variações de acordo com as decisões que serão tomadas. O mesmo é aplicado a qualquer plano de ação, que deve contar com um balanço de tudo que será gasto para que o objetivo seja alcançado.

Gestão participativa = 5W2H

Para aplicar todos esses conceitos em sua empresa, você simplesmente terá que mapear esse planejamento para os objetivos traçados por sua equipe. Sendo assim, você deverá realizar todas as questões do planejamento 5W2H e apontar as respostas de acordo com as ações planejadas pela equipe.

Gestão participativa = 5W2H

Ação	O quê? (What?)	Por que? (Why?)	Onde? (Where?)	Quem? (Who?)	Quando? (When?)	Como? (How?)	Quanto? (How much?)
1	Refazer a estrutura de redes, que hoje é dividida entre alunos e professores em duas redes.	Para simplificar a rede e melhorar o fluxo de banda de internet.	No setor de Infraestrutura.	Técnico de redes.	Média de 4 meses para implantação.	-Desfazendo a rede atual. -Licitando mais técnicos. -Levantando os materiais necessários. -Licitando compras. -Aplicando os serviços.	Resultado de licitação.
2	Contratar um pacote maior de dados.	Para melhorar a disponibilidade do sinal da banda para alunos e professores.	No setor de Infraestrutura.	Tesouraria da faculdade.	Média de 2 semanas	-Licitando pacotes de bandas de internet de diferentes empresas. -Comprando através de licitações.	Resultado de licitação.
3	Aplicar fibra ótica na rede existente e melhorar roteadores para suportar a mesma.	Porque a internet é mais rápida e flui melhor através da fibra ótica.	No setor de infraestrutura.	Técnico de redes.	Média de 2 meses para implantação.	-Licitando mais técnicos. -Levantando os materiais necessários. -Licitando compras. -Aplicando os serviços.	Resultado da licitação.
4	Remover limitação de velocidade atual e aplicar limite de consumo de banda.	Para facilitar o uso dos alunos e professores e bloquear aqueles que usufruem de forma indevida.	No setor de infraestrutura.	Técnico de redes.	Média de 1 mês para configuração.	-Licitando mais técnicos. -Aplicando os serviços.	Resultado da licitação.

Gestão participativa = 5W2H

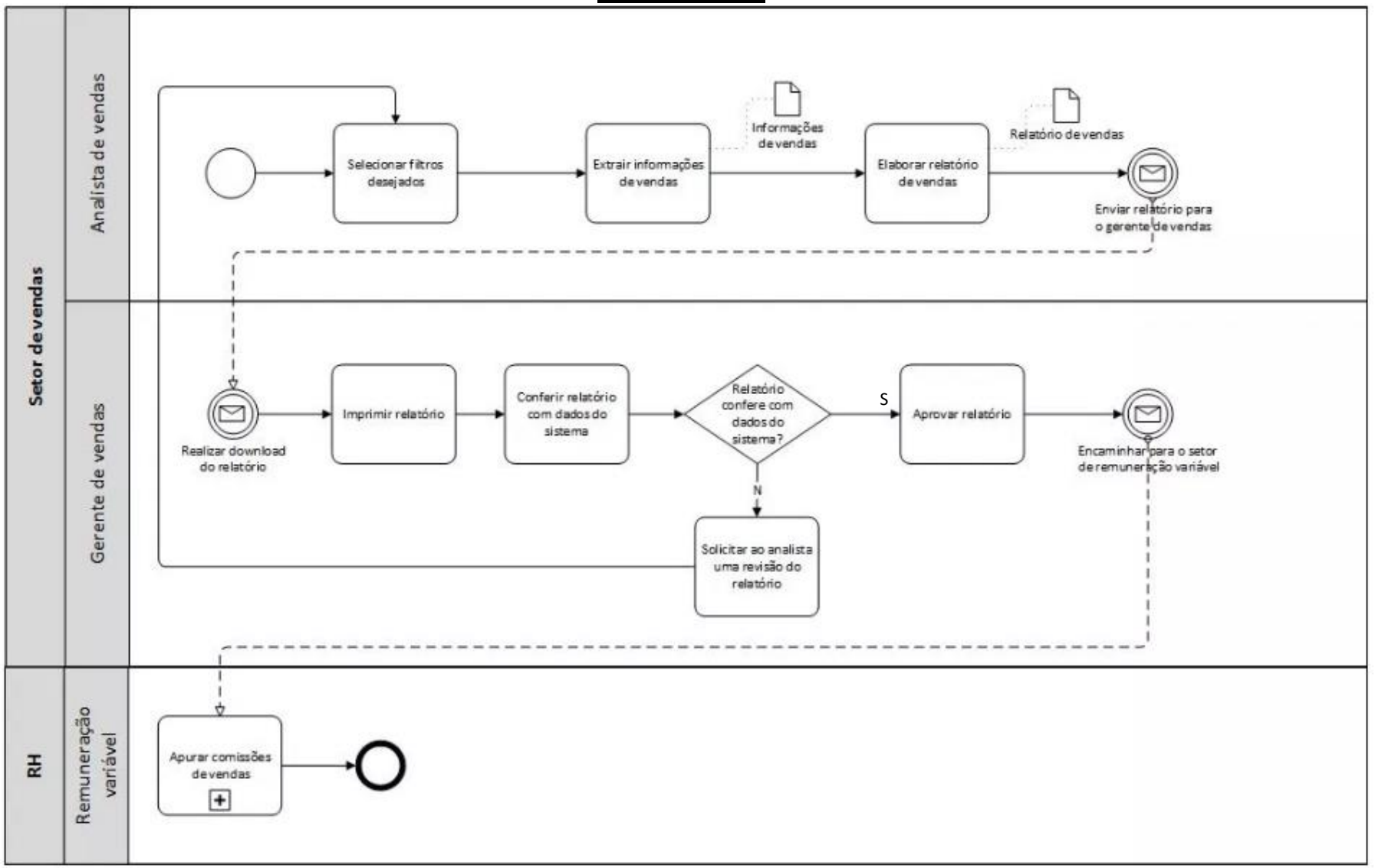
Ação	O quê? (What?)	Por que? (Why?)	Onde? (Where?)	Quem? (Who?)	Quando? (When?)	Como? (How?)	Quanto? (How much?)
1	Aumentar o número de laboratórios disponíveis para as turmas.	Porque com mais salas, facilitará o uso do laboratório do térreo. Para que outros alunos possam utilizar o laboratório.	Na unidade I e II algumas salas de informática podem ser construídas	Cabe ao departamento de computação realizar essa tarefa	O projeto deve ser implantado na virada do semestre, enquanto os alunos estiverem de férias.	Realizando a instalação de computadores, componentes, rede dentre outros recursos computacionais	Através de licitação para que uma empresa forneça o requerido
2	Definir o uso do laboratório de informática do térreo apenas para o uso individual	Para que os alunos tenham um espaço para realizar as atividades a qualquer momento	O laboratório já se encontra na unidade II	A coordenação da instituição	Nas férias.	Obedecendo e seguindo uma decisão hierárquica	Sem custos
3	Utilizar os notebooks nas turmas de sistemas de informação	Para que todas as turmas de SI utilizem os laboratórios disponíveis em algumas disciplinas	Nas próprias salas das turmas	O departamento de computação	Nas férias, pois o cronograma é definido com antecedência.	Realizando algumas atividades podem ser realizadas por grupos de alunos.	Custos apenas se o aluno quiser possuir um notebook, pois seu uso não será obrigatório
4	Planejar as disciplinas para que o número de turmas que necessitem utilizar os laboratórios seja igual	Porque a grade curricular não leva em conta a necessidade de utilização dos computadores pelas turmas no mesmo momento.	Na coordenação	Todos os departamentos em conjunto	Durante as reuniões antes do inicio dos semestres	Facilitando a ação da coordenação que deve trabalhar em conjunto com o departamento para definição do cronograma e agendamento das salas	Tempo e cronogramas

ENGENHARIA DE SOFTWARE

BPMN

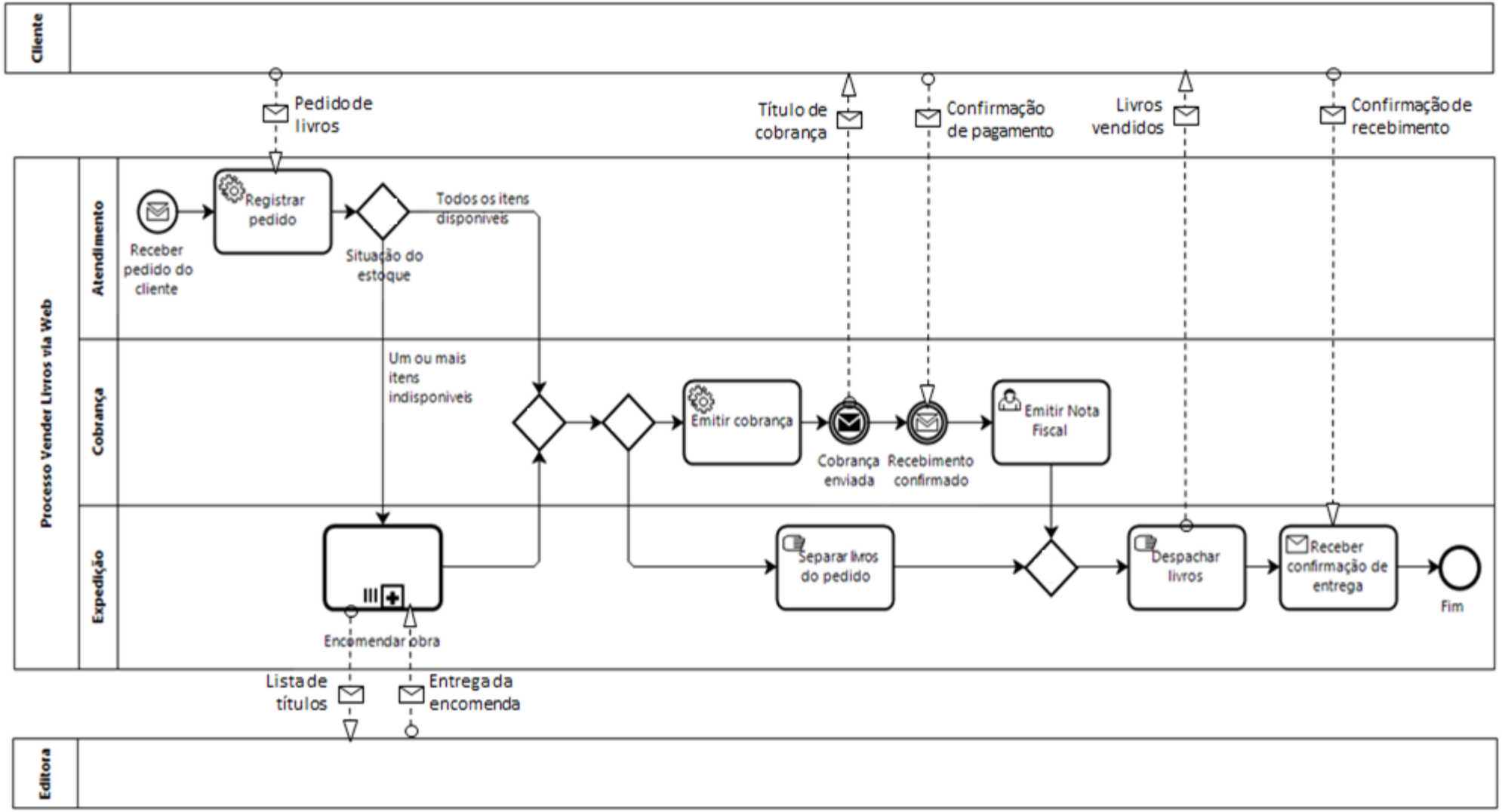
ENGENHARIA DE SOFTWARE

Exemplo 1:



ENGENHARIA DE SOFTWARE

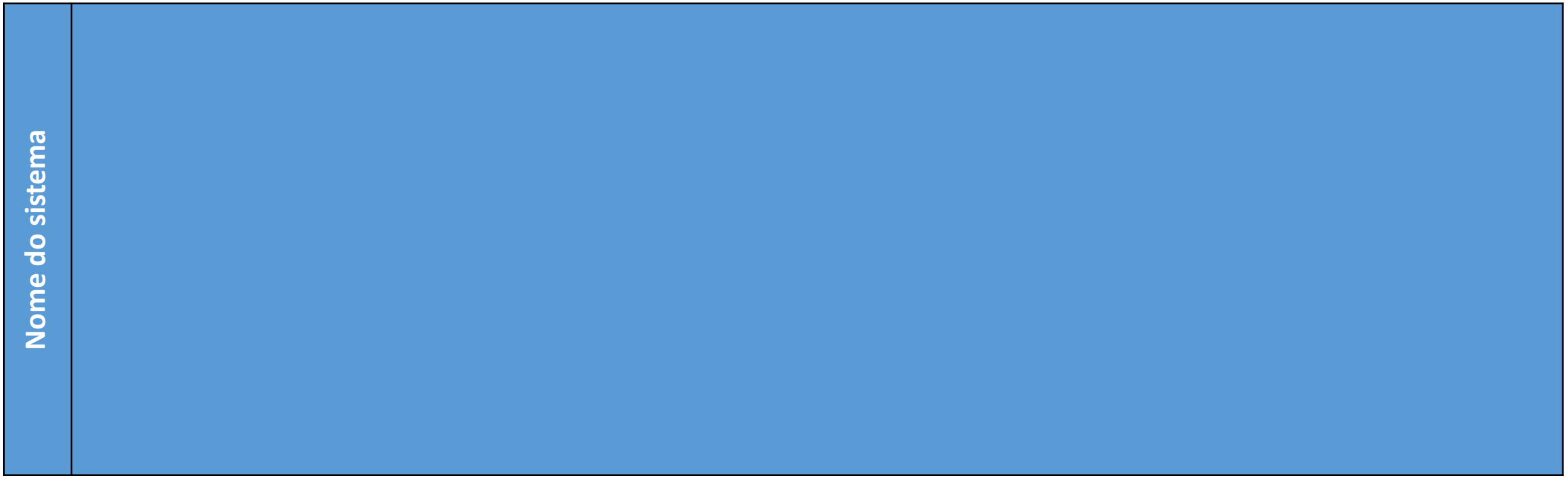
Exemplo 2:



ENGENHARIA DE SOFTWARE

Na prática: definir uma ferramenta (BPMN.io, Bizagi, Lucidchart, Visio)

1º criar a piscina



ENGENHARIA DE SOFTWARE

Na prática: definir uma ferramenta (BPMN.io, Bizagi, Lucidchart, Visio)

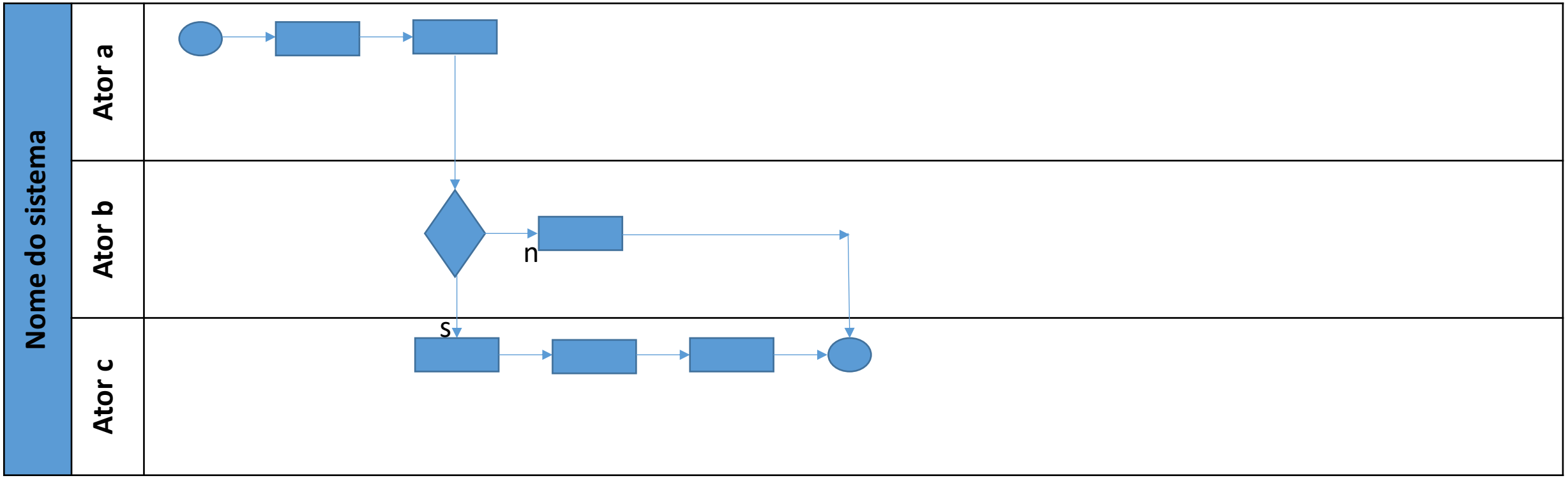
2º criar a raias (depende da complexidade do sistema)

Nome do sistema		
	Ator a	
	Ator b	
	Ator c	

ENGENHARIA DE SOFTWARE

Na prática: definir uma ferramenta (BPMN.io, Bizagi, Lucidchart, Visio)

2º criar a raias (depende da complexidade do sistema)



ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

- **Business Process Model and Notation (BPMN) em (*Modelo e Notação de Processos de Negócio*) --** É um modelo da metodologia de gerenciamento de processos de negócio e trata-se de uma série de ícones padrões para o desenho de processos, o que facilita o entendimento do usuário.
- A modelagem é uma etapa importante da automação, pois é nela que os processos são descobertos e desenhados.
- É nela também que pode ser feita alguma alteração no percurso do processo visando a sua otimização.
- A notação também pode ser utilizada para a modelagem de Arquitetura de Processos.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

- Foi desenvolvido pela Business Process Management Initiative (BPMI) para unificar a forma com que as empresas faziam a modelagem de seus processos, pois cada organização possuía sua própria notação, dificultando a vida de usuários e clientes.
- Atualmente é mantido pelo Object Management Group já que as duas organizações se fundiram em 2005.
- O BPMN, desde o início, foi apoiado por várias empresas de renome mundial no segmento de modelagem de processos, sendo uma resposta independente de fornecedor de solução à demanda de modelagem de processos.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

O Modelo e Notação de Processos de Negócio é um padrão para modelagem de processos de negócios e fornece uma notação gráfica para a especificação de processos de negócios em um *Business Process Diagram* (BPD), ou *Diagrama de Processos de Negócio*, baseado em uma técnica de fluxograma muito semelhante ao de diagramas de atividades da Unified Modeling Language (UML).

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

O objetivo do BPMN é de apoiar a gestão de processos de negócios tanto para usuários técnicos e usuários de negócios, fornecendo uma notação que é intuitiva para os usuários corporativos ainda capaz de representar a semântica complexa do processo. A especificação BPMN também fornece um mapeamento entre os gráficos da notação para as construções subjacentes de linguagens de execução, particularmente a Business Process Execution Language.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

O principal objetivo do BPMN é fornecer uma notação padrão que seja facilmente compreensível por todos os intervenientes do negócio. Estas partes interessadas no negócio incluem os analistas de negócios que criam e refinam os processos, os desenvolvedores técnicos responsáveis pela implementação dos processos e os gerentes de negócios que monitoram e gerenciam os processos.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

A notação BPMN também é útil para definir melhorias em processos, documentar processos existentes, identificar e automatizar processos, além de servir como uma linguagem comum para fazer a ponte de comunicação entre o design de processos de negócios e a implementação.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

Atualmente existem vários padrões concorrentes para linguagens de modelagem de processos de negócio utilizadas por ferramentas de modelagem e processos. A adoção generalizada do BPMN ajudará a unificar a expressão de conceitos básicos de processos de negócio (por exemplo, os processos públicos e privados, coreografias), bem como conceitos avançados de processos (por exemplo, tratamento de exceção, a compensação de transações).

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

- A notação BPMN também é útil para definir melhorias em processos, documentar processos existentes, identificar e automatizar processos, além de servir como uma linguagem comum para fazer a ponte de comunicação entre o design de processos de negócios e a implementação.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

- Softwares podem exigir o cumprimento das conformidades estabelecidas, o que é totalmente compatível com a especificação. Softwares apenas parcialmente compatíveis com os termos de conformidade só podem exigir conformidade referente a parte do software que foi baseada nesta especificação, mas não pode exigir a observância da conformidade com a especificação toda.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

A especificação define quatro tipos de conformidade:

- Conformidade da Modelagem de Processo;
- Conformidade da Execução de Processo;
- Conformidade da Linguagem de Execução de Processo;
- Conformidade da Modelagem de Coreografia.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

A implementação que atende a conformidade do tipo Modelagem de Processo não é requisito obrigatório para a conformidade Modelagem de Coreografia e vice-versa. Da mesma forma, a implementação observando a conformidade Execução de Processo não é requisito obrigatório para a conformidade Modelagem de Coreografia.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

O BPMN será obrigado a suportar apenas os conceitos de modelagem que são aplicáveis aos processos de negócios. Isto significa que outros tipos de modelagem feitos por organizações sem propósitos corporativos estará fora de alcance para o BPMN.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

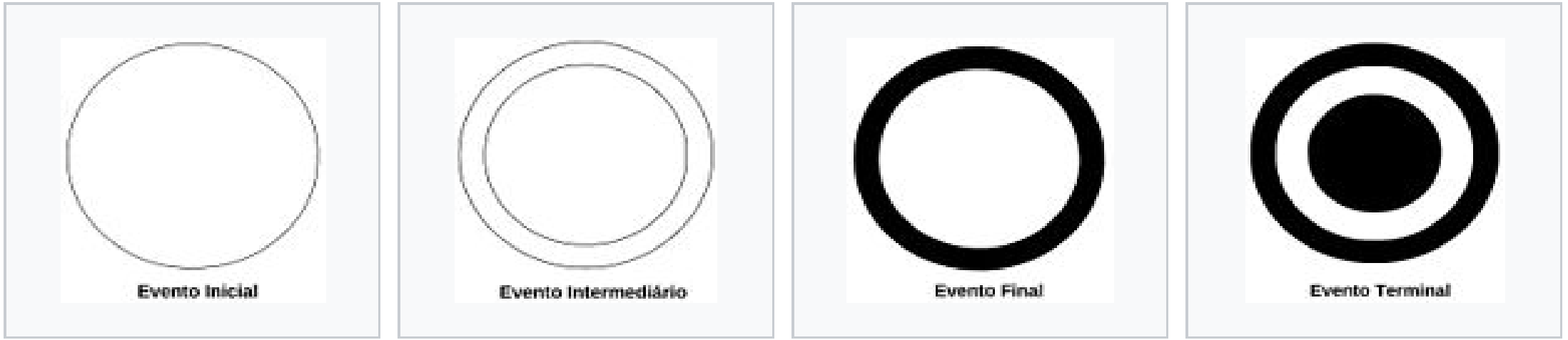
Por exemplo, a modelagem dos seguintes **não** será uma parte da BPMN:

- estruturas organizacionais
- avarias funcionais
- modelos de dados
- Além disso, apesar do BPMN mostrar o fluxo de dados (mensagens) e a associação de artefatos de dados para atividades, ele não é um diagrama de fluxo de dados.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

Objetos de Fluxo - são os principais elementos descritivos dentro do BPMN e consistem de três elementos essenciais: (Eventos)



ENGENHARIA DE SOFTWARE

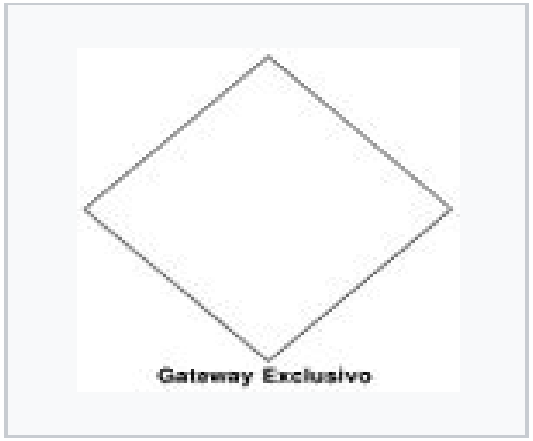
#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

Objetos de Fluxo - são os principais elementos descritivos dentro da BPMN e consistem de três elementos essenciais: (Atividades)

ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

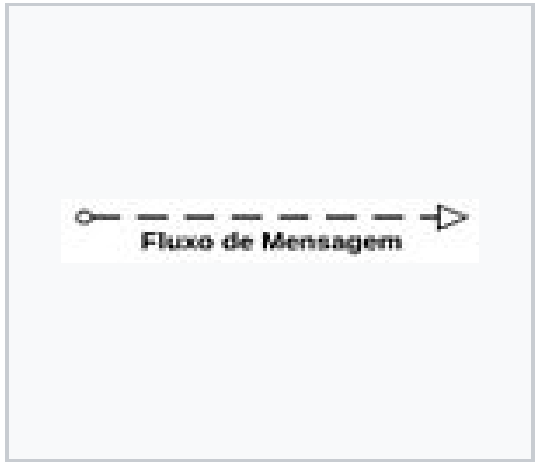
Objetos de Fluxo - são os principais elementos descritivos dentro da BPMN e consistem de três elementos essenciais: (Gateways)



ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

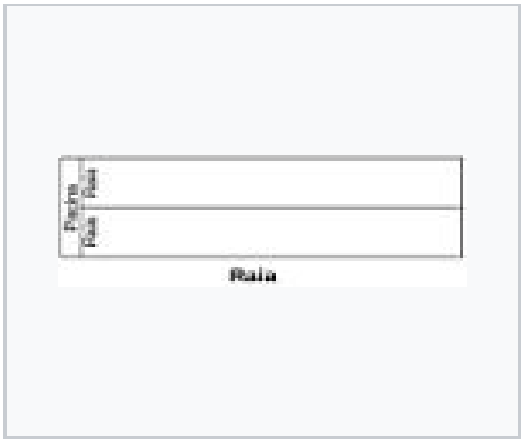
Objetos de Conexão - Conectam os objetos de fluxo uns com os outros ou com outras informações (Fluxo de Sequência, Fluxo de Mensagem, Associação);



ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

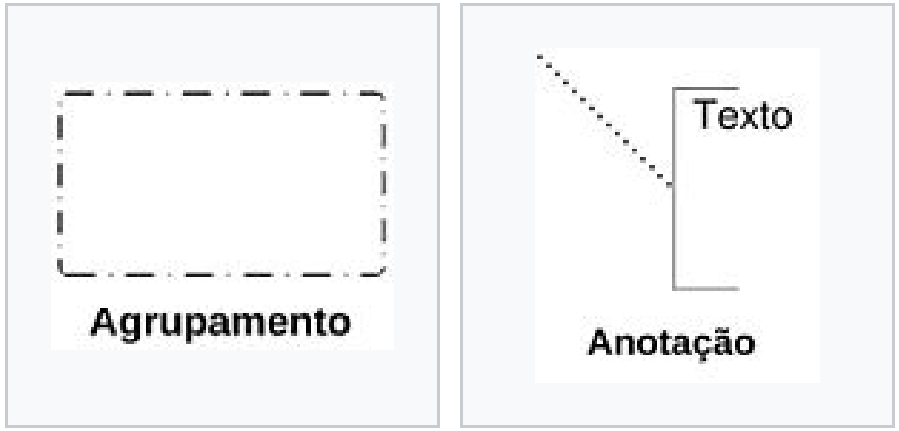
Swim lanes - A Piscina (Pool) é a representação gráfica da Colaboração entre Participantes. Um Participante representa uma entidade parceira específica (por exemplo, uma empresa) e/ou uma entidade genérica (por exemplo, um comprador, vendedor ou fabricante). Uma Colaboração é uma coleção de mensagens trocadas entre Participantes (Pool, Lane);



ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

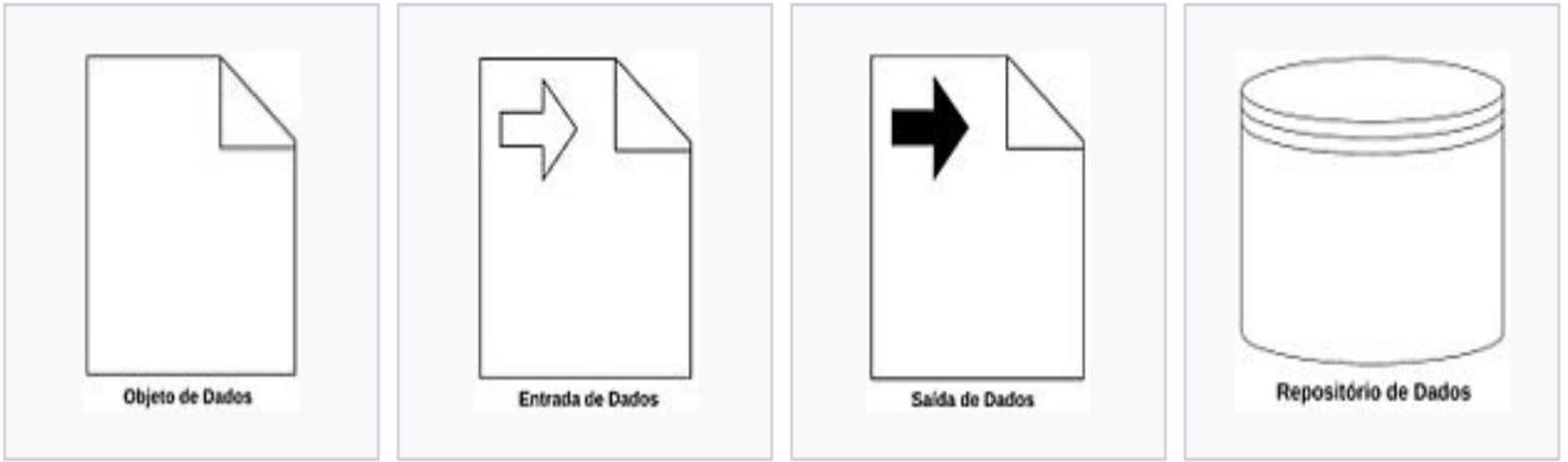
Artefatos - Artefatos são usados para fornecer informações adicionais sobre o processo (Objeto de Dados, Grupo, Anotação);



ENGENHARIA DE SOFTWARE

#BPMN – Fonte adaptada da obra de Vinicius Nóbile de Almeida

Dados - São elementos que armazenam ou transmitem dados durante a execução do processo. Esses elementos são semelhantes a uma variável de linguagem de programação.



ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

- Sobre as swim lanes, existem dois tipos a analisar:
- **Piscinas** – representam processos e participantes no processo.
- **Raias** – cada piscina possui várias raias, que simbolizam os papeis, áreas e responsabilidades no processo.
- Segundo o conceito BPMN, os artefatos trazem um maior nível de detalhe ao diagrama, pois permitem que informações extras sejam trazidas.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

- Principais símbolos usados na notação BPMN 2.0, divididos em 4 tipos principais:

- **1-Eventos:** indicam eventos exteriores ao processo que o influenciam.
- **2-Conectores:** elementos de ligação da sequência dos fluxos de trabalho.
- **3-Atividades:** representam o trabalho que será realizado.
- **4-Gateways:** mostram a ramificação e a reunião do fluxo de tarefas.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

- Para todos os tipos de eventos, via de regra, uma linha fina de contorno significa início (pode ser usada também a cor verde), uma linha dupla mostra um evento intermediário (pode ser usada a cor azul) e uma linha grossa indica um evento final (pode ser usada a cor vermelha), com algumas exceções em que não haveria sentido haver um evento final desse tipo.
- É importante lembrar que dependendo do evento, diferentes consequências ocorrerão tanto no início, como na fase intermediária ou final do processo, segundo a notação BPMN 2.0.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo simples: Normalmente é utilizado para representar o início manual de um processo.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo mensagem: O processo é iniciado com a chegada de uma mensagem recebida.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo tempo: o processo é iniciado por uma condição de tempo.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo condicional: uma condição lógica determina o início do evento.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo sinal: um sinal vindo de outro processo inicia este processo.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo múltiplo: um de muitos eventos possíveis iniciam o processo.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo múltiplo paralelo: múltiplos eventos devem ocorrer para iniciar o processo.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Muitos dos gatilhos (ícones) dos eventos iniciadores podem ser encontrados também nos eventos intermediários, que servem para modelar os eventos que ocorrem durante a execução do processo.

Veja alguns exemplos de eventos intermediários segundo a notação BPMN 2.0

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo intermediário (genérico): Esse evento não possui uma ação definida, mas representa na modelagem alguma mudança de estado no processo.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo mensagem: Pode ser utilizado para troca de mensagens entre duas piscinas. Na implementação de automatização também é usado para envio e recebimento de e-mails, chamadas web services e outras funções disponíveis via conectores.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo temporizador: Pode ser utilizado como um evento de borda em uma tarefa para definir fluxos de exceção. Também é utilizado para estabelecer uma restrição no fluxo (por exemplo “Aguardar 1 dia”).

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo link: É utilizado para representar graficamente uma continuidade de um fluxo de sequência.

O evento que inicia o “go to” deve ser do tipo “lançamento” e o evento link que recebe o redirecionamento deve ser do tipo “captura”.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo sinal: Pode ser utilizado para fazer o broadcast de um sinal ou para ouvir um broadcast em um sinal de um evento de borda. Sinais são uma forma de comunicação desacoplada entre processos de negócio.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo condicional: Assim como o evento intermediário temporizador, pode ser utilizado como evento de borda (contido em uma tarefa) para modificar o fluxo normal, ou fora de tarefas para representar uma restrição.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo paralelo: Semelhante aos eventos condicional e timer, porém com a possiblidade de conter vários intermediários, onde todos itens contidos devem ser atendidos para que o paralelo seja disparado.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

1-Eventos:

Início de processo múltiplo: Semelhante ao paralelo, porém dispara o evento quando apenas um dos intermediários contidos for atendido.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

2-Conectores:

- Ligam diferentes elementos em um fluxo BPMN.
- Os objetos de fluxo são uma simbologia BPMN que precisa conectar-se entre si de alguma forma, e isso se dá através dos objetos de conexão.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

2-Conectores:

- **Fluxo de sequência** – mostra em que ordem as atividades são executadas, e é simbolizado por uma linha cheia e uma seta adiante.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

2-Conectores:

Fluxo de mensagens – indica quais as mensagens que fluem entre dois processos/piscinas, e é representada por uma linha tracejada, um círculo aberto e uma seta aberta no fim.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

2-Conectores:

Associação – conecta os artefatos aos objetos de fluxo, e é simbolizado por uma linha tracejada.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

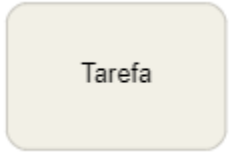
- São as tarefas que devem ser realizadas. Podem ser discriminadas com pequenos símbolos BPMN no canto superior esquerdo do retângulo que as representam.
- Devem ser escritas com o verbo no infinitivo.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

Tarefa simples: Uma tarefa representa um trabalho realizado no processo. Pode ter associado um formulário para entrada de dados.

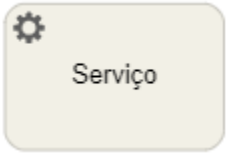


ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

Tarefa de serviço: Executa um web service e é utilizado para implementar integrações com sistemas de informação.

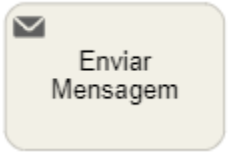


ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

Tarefa de envio de mensagem: Envia uma mensagem para outra piscina ou processo e avança automaticamente para a próxima tarefa, que normalmente é uma tarefa de recebimento ou um evento intermediário de captura de mensagem.

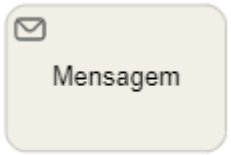


ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

Tarefa de recebimento de mensagem: Aguarda o recebimento de uma mensagem a partir de outra piscina ou processo. Normalmente está posicionado após uma tarefa de envio de mensagem ou evento intermediário de lançamento de mensagem.



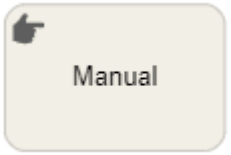
ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

Tarefa manual: Representa uma tarefa realizada por uma pessoa que não utiliza um sistema de workflow.

Exemplo: Servir café.

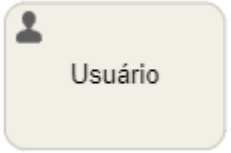


ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

Tarefa de usuário: Representa o trabalho realizado por um usuário de um sistema conectado ao engine de workflow. Exemplo: Cadastrar funcionário.

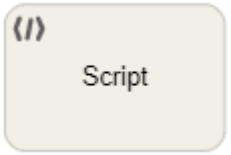


ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

Tarefa de script: Executa uma sequência de comandos utilizando o próprio motor de processos. Pode ser usado, por exemplo, para rodar um script Powershell.

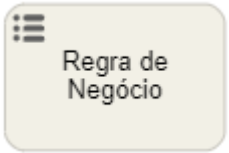


ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

3-Atividades

- **Tarefa de regra de negócio:** Aciona uma regra de negócio que retorna um valor para comparação. Pode ser realizado por meio de uma chamada de web service.



ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

4-Gateways

- **Exclusivo:** o fluxo segue por apenas um dos fluxos de saída. Pode ser utilizado para representar um desvio no fluxo.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

4-Gateways

Paralelo: o fluxo se divide em outros que ocorrem em paralelo.

Paralelo

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

4-Gateways

Inclusivo: o fluxo segue por uma condição inclusiva, ou seja, para cada fluxo de sequencia de saída é avaliada uma fórmula e, se for retornado o valor verdadeiro, então o caminho é ativado. Este tipo normalmente demanda o acréscimo de um segundo gateway inclusivo para representar a sincronização. Falamos sobre isso no nosso **curso de modelagem BPMN.**

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

4-Gateways

- **Complexo:** Controla condições complexas de divergência e também convergência.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

4-Gateways

Intermediário exclusivo baseado em eventos: usado sempre para dividir o fluxo iniciando um processo devido a ocorrência exclusiva de um de múltiplos eventos. É muito usado para receber uma mensagem a partir de outra piscina.

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

4-Gateways

Inicial exclusivo baseado em eventos: usado sempre para dividir o fluxo iniciando um processo devido a ocorrência exclusiva de um de múltiplos eventos.

Evento inicial exclusivo

ENGENHARIA DE SOFTWARE

#BPMN – Fonte: https://www.heflo.com/pt-br/bpm/notacao-bpmn/

Dicas:

1-Defina as responsabilidades: use a piscina e as raias para agrupar as tarefas e definir os responsáveis por elas;

2-Adicione um iniciador: ele indicará a forma como o processo será iniciado, pode ser o recebimento de um email, por exemplo;

3-Acrescente as tarefas e desvios: esses elementos determinam a lógica de seu processo, o caminho que as atividades seguem;

4-Marque o fim do processo: um evento deve indicar o fim do processo. Por exemplo, a encomenda enviada;

5-Revise o BPMN --- 0 cliente precisa verificar e validar;

ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

- Software:
- São programas para computadores e equipamentos eletrônicos com documentação associada (requisitos, modelos de projeto e manuais de usuário, etc.).
- É o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais.
- Eles refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

- Software:
- Os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais.
- O termo requisito pode ser usado com variações que vão desde uma simples declaração de um serviço que o sistema deve fornecer, ou uma restrição do sistema, até a uma definição formal e detalhada de uma função do sistema.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

- Requisitos funcionais: correspondem à listagem de tudo o que o sistema deve fazer. São declarações das funções que o sistema deve fornecer, como ele deve reagir a entradas específicas e como deve se comportar em determinadas situações. Podem, inclusive, declararem o que o sistema NÃO deve fazer.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

- Requisitos do usuário: correspondem aos requisitos abstratos de alto nível. São declarações escritas em linguagem natural, ou representadas em diagramas, sobre as funções e restrições do sistema.
- Requisitos de sistema: indicam uma descrição detalhada do que o sistema deverá fazer, assim como das restrições. O documento de requisitos pode ser chamado de especificação funcional.
- Requisitos de domínio: são requisitos que se originam do domínio de aplicação do sistema e que refletem características desse domínio. Podem ser funcionais ou não-funcionais.
- Requisitos evidentes: são efetuados com conhecimento do usuário. Corresponderão aos eventos e respostas do sistema (troca de informações).
- Requisitos ocultos: são efetuados pelo sistema sem o conhecimento explícito do usuário.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

- Requisitos não-funcionais: são restrições colocadas sobre como o sistema deve realizar seus requisitos funcionais. Por exemplo: regras de negócio, restrições de tempo, restrições sobre o processo de desenvolvimento, etc.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

- Requisitos externos: abrange todos os requisitos procedentes de fatores externos ao sistema e a seu processo de desenvolvimento. Ex.: requisitos de interoperabilidade, requisitos legais, requisitos éticos (garantir que o sistema será aceitável para seus usuários e o público em geral).
- Requisitos de produtos: especificam o comportamento do produto. Ex.: requisitos de desempenho (rapidez com que o sistema deve operar, memória requerida), requisitos de confiabilidade (taxa aceitável de falhas), requisitos de portabilidade, requisitos de facilidade de uso.
- Requisitos organizacionais: procedentes de políticas e procedimentos nas organizações do cliente e do desenvolvedor. Ex.: padrões de processo a serem utilizados, os requisitos de implementação (linguagem de programação, método do projeto a ser utilizado), requisitos de fornecimento (quando os produtos e documentos devem ser entregues).

ENGENHARIA DE SOFTWARE

#Perguntas – Fonte:

Documento de requisitos: Como preencher este importante artefato?

ID: (RF001 ou RNF001)	**Nome do Requisito:** (cadastrar alunos)
Descrição →	O sistema deverá cadastrar os novos alunos.....
Categoria: (evidente/oculto)	**Prioridades:** (essencial/importante/desejável)
Informações →	Descrever os campos
Regra de Negócio (se existir) →	O sistema não poderá deixar a inclusão de usuários com o mesmo CPF. O sistema deverá deixar habilitada a opção para incluir, alterar e excluir um aluno sem alterar o número do CPF. No ato do cadastro a informação de Aluno Ativo deverá estar marcada, podendo ser alterada posteriormente. O Aluno só poderá ter um Avaliador.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

Documento de requisitos: Como preencher este importante artefato?

- **ID do Requisito:** Os requisitos devem receber um identificador único. A numeração inicia com o identificador [RF001] ou [RNF001] e prossegue sendo incrementada à medida que forem surgindo novos requisitos.
- **Nome do Requisito:** O nome deve representar com clareza a ação do Requisito e deve ser escrito com o verbo no infinitivo, exemplo: (cadastrar alunos).
- **Descrição:** Deve ser altamente detalhada --- ressaltar a funcionalidade do requisito, exemplo: (O sistema deverá cadastrar os novos alunos conforme descrição e orientação da secretaria acadêmica, atendendo as normas do CPS).
- **Categoria:** Indicar se o Requisito é evidente (são efetuados com conhecimento do usuário) ou oculto (são efetuados pelo sistema sem o conhecimento explícito do usuário), exemplo: (login). **ATENÇÃO:** Se o Requisito for Não Funcional é preciso citar em **Categoria:** segurança, performance ou compatibilidade.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

Documento de requisitos: Como preencher este importante artefato?

- **Prioridades dos requisitos:** Descrevem a relevância do Requisito para o sistema. Podem ser:
- **Essencial:** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
- **Importante:** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- **Desejável:** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte:

Documento de requisitos: Como preencher este importante artefato?

- **Informações:** É muito importante que o Analista de Negócio (pessoa que está realizando a elicitação de requisitos) faça a citação de todos os campos necessários e obrigatórios deste Requisito que serão tratados na IHC/BD, exemplo: (nome, CPF, RG, telefone, cidade, demais --- TODOS).
- **Regra de Negócio:** São “declarações sobre políticas ou condições que devem ser satisfeitas”. É uma restrição imposta pelo negócio que regulamenta o comportamento de um procedimento operacional do negócio, exemplo: (O sistema não poderá deixar a inclusão de usuários com o mesmo CPF).

ENGENHARIA DE SOFTWARE

#Perguntas – Fonte:

Verifique as Funcionalidades?

1. Onde os softwares serão usados?
2. Qual o impacto do software na sociedade?
3. Quando desenvolver para mobile, web ou desktop?
4. Como desenvolver?
5. O que o software produz?
6. O que levar em consideração ao desenvolver um software? (Layout, Padronização, LP, BD, Redes, Segurança, Qualidade, Público, SO, Multiusuário, Pensar no hoje e no amanhã)

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- O valor de um produto vem de suas características. Uma **característica de software** (*software characteristic*) é definida no Glossário do IEEE (*IEEE Standard Glossary of Software Engineering Terminology*) [IEEE90] como um traço, qualidade ou propriedade de um produto de software. Quando uma característica é distintiva de um item, ela será chamada de **característica especial**.
- Tratando-se de software, costuma-se dividir as características especiais em:
- •**funcionais**, que representam os comportamentos que um programa ou sistema deve apresentar diante de certas ações de seus usuários;
- •**não funcionais**, que quantificam determinados aspectos do comportamento.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Por exemplo, em um terminal de caixa automático, os tipos de transações bancárias suportadas são características funcionais. A facilidade de uso, o tempo de resposta e o tempo médio entre falhas são características não funcionais.
- Os **requisitos** são as características que definem os critérios de aceitação de um produto. A engenharia tem por objetivo colocar nos produtos as características especiais que são requisitos. Outras características podem aparecer acidentalmente, mas os produtos não devem ser desenhados para incluí-las, já que, normalmente, toda característica extra significa um custo adicional de desenho ou de fabricação.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- **Especificação dos requisitos**
- Os requisitos podem ser dos seguintes tipos:
- •Os requisitos **explícitos** são aqueles descritos em um documento que arrola os requisitos de um produto, ou seja, um documento de **especificação de requisitos** ou equivalente.
- •Os requisitos **normativos** são aqueles que decorrem de leis, regulamentos, padrões e outros tipos de normas a que o tipo de produto deve obedecer requisitos organizacionais.
- •Os requisitos **implícitos** são expectativas dos clientes e usuários, que são cobradas por esses, embora não documentadas e satisfazem os requisitos externos.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Uma especificação de requisitos pode conter requisitos incompletos, inconsistentes ou ambíguos.
- Alguns desses problemas decorrem da própria linguagem natural, que normalmente é usada para expressá-los.
- Outros decorrem de técnicas deficientes de levantamento e especificação dos requisitos.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- **Engenharia dos requisitos**
- Um dos problemas básicos da engenharia de software é o levantamento e a documentação dos requisitos dos produtos de software.
- Quando esse levantamento é bem-feito, os requisitos implícitos são minimizados.
- Quando a documentação é bem-feita, os requisitos documentados têm maiores chances de serem corretamente entendidos pelos desenvolvedores.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- **Engenharia dos requisitos**
- Algumas técnicas de análise dos requisitos ajudam a produzir especificações mais precisas e inteligíveis.
- O conjunto das técnicas de levantamento, documentação e análise forma a **engenharia dos requisitos**, que é uma das disciplinas da engenharia de software.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Infelizmente, muitos clientes não entendem a necessidade de especificações de requisitos.
- Pior ainda, muitos desenvolvedores de software e, pior de tudo, muitos gerentes também não.
- É uma situação tão absurda quanto querer resolver um problema sem escrever o respectivo enunciado: existe grande risco de resolver-se o problema errado.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Por outro lado, é possível também a existência de requisitos que não correspondam às necessidades reais dos clientes e usuários.
- Essa falha de engenharia de requisitos indica que não foi feita uma análise do valor de cada requisito, do ponto de vista da missão que o produto deve cumprir.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Cabe aos engenheiros de software insistir sempre na elaboração de uma boa especificação de requisitos, ou conjunto equivalente de enunciados. Faz parte do seu trabalho convencer clientes e usuários de que:
- •a explicitação dos requisitos é indispensável, devendo-se registrar esses requisitos em um ou mais artefatos oficiais;
- •ela não representa custos supérfluos, mas investimentos necessários, que se pagam com altos juros;
- •a participação dos usuários na engenharia de requisitos é fundamental para que as suas necessidades sejam corretamente atendidas pelo produto;
- •um bom levantamento de requisitos custa tempo e dinheiro;
- •a ausência de um bom levantamento de requisitos custa muito mais tempo e dinheiro.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- Um problema comum no desenvolvimento de software é a **instabilidade dos requisitos**, que ocorre quando clientes e usuários trazem novos requisitos, ou alterações de requisitos, quando o desenvolvimento já está em fase adiantada. A instabilidade dos requisitos costuma ter um custo muito alto; geralmente significa perder trabalho já feito, desfazer algumas coisas e remendar outras. Na engenharia de software, a instabilidade dos requisitos é tão danosa quanto nas outras engenharias. Quando se muda a planta de um edifício durante a construção, geralmente é preciso desfazer parte do que já foi construído, e o remendo raramente é satisfatório.

ENGENHARIA DE SOFTWARE

#Requisitos – Fonte: Wilson de Pádua Paula Filho (E.S. produtos)

- A boa engenharia de requisitos reduz a instabilidade desses requisitos. Entretanto, alterações dos requisitos são às vezes inevitáveis. A engenharia de requisitos é sujeita a limitações humanas, e, mesmo que o levantamento seja perfeito, podem ocorrer alterações de requisitos por causas externas aos projetos. Por exemplo, nos seguintes casos, requerendo alterações nos relatórios que o produto deve atender:
- •a legislação, as necessidades e objetivos do cliente ou os processos de negócio mudam no meio do projeto;
- •os requisitos são inicialmente nebulosos porque as necessidades não são bem entendidas, e passam a ser entendidos melhor depois que liberações do produto são experimentadas pelos usuários;
- •a cultura da organização cliente dificulta o levantamento de requisitos firmes e precisos, e ela está disposta a pagar por isso.
- A **gestão dos requisitos** é a disciplina da engenharia de software que procura manter sob controle o conjunto dos requisitos de um produto, mesmo diante de algumas alterações inevitáveis.

ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

- **#Estrutura Analítica de Projetos (EAP)**

- É um processo de subdivisão das entregas e do trabalho do projeto em componentes menores e mais facilmente gerenciáveis. É estruturada em árvore exaustiva, hierárquica (de mais geral para mais específica) orientada às entregas, fases de ciclo de vida ou por sub-projetos (deliverables) que precisam ser feitas para completar um projeto.

ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- O objetivo de uma EAP é identificar elementos terminais (os produtos, serviços e resultados a serem feitos em um projeto). Assim, a EAP serve como base para a maior parte do planejamento de projeto.
- A ferramenta primária para descrever o escopo do projeto (trabalho) é a estrutura analítica do projeto (EAP).

ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- Como construir uma EAP: A EAP deve ser completa, organizada e pequena o suficiente para tornar possível a medição do progresso, mas não detalhada o suficiente para se tornar, ela mesma, um obstáculo à realização do projeto. Uma boa heurística a seguir é a regra do 8-80: exige-se que um pacote de trabalho ocupe entre 8 e 80 horas de duração. É uma das partes mais importantes no plano do projeto.

ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- Ela serve como entrada para o desenvolvimento da agenda, atribuir funções e responsabilidades, gerir riscos, entre outros.
- Na internet é possível acessar alguns sites para montagem gratuito de uma EAP ou WBS, tais como: http://www.wbstool.com/.

ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- Não há regras para os níveis de decomposição. Cada gerente de projeto ou membros da equipe encarregados da decomposição devem usar o bom senso de parar no nível no qual o custo de acompanhar o pacote seja inferior ao benefício de controle.

ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- A Regra 100%... estabelece que a EAP inclui 100% do trabalho definido pelo escopo do projeto e captura todas as entregas – internas, externas, intermediarias – de forma completa, incluindo o gerenciamento do projeto. A regra dos 100% é um dos mais importantes princípios que guia o desenvolvimento, decomposição e avaliação da EAP.
- A aplicação desta regra vale para todos os níveis na hierarquia: a soma de todos os trabalhos dos níveis "filhos" deve ser igual a 100% do trabalho representado pelo "pai" e a EAP não deve incluir qualquer trabalho que saia do escopo existente do projeto, isto é, ele não pode incluir mais do que 100% do trabalho.

ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- Planeje entregas, não planeje ações: Se o projetista da EAP (Estrutura Analítica de Projeto) tenta capturar qualquer detalhe orientado a ação na EAP, ele irá incluir ações de mais ou de menos. Ações demais excederão 100% do escopo do pai e ações de menos cairão abaixo dos 100% do escopo do pai. A melhor forma de ser aderente a Regra dos 100% é definir os elementos da EAP em termos das entregas ou resultados.

ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- Desenvolvimento orientado a aspectos utiliza-se de uma técnica similar a qual emprega uma estrutura de decomposição de aspectos. Quando um projeto provê serviços profissionais, uma técnica comum é capturar todas as entregas planejadas para criar uma EAP orientada à entrega.
- EAP que subdividem o trabalho em fases do projeto (por exemplo: Fase Projeto Preliminar, Fase projeto Critico) devem assegurar que as fases sejam claramente separadas para uma entrega (por exemplo: um documento de revisão de projeto preliminar, ou um documento aprovação da revisão projeto crítico).

ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- Nível de detalhe (granularidade) e elaboração progressiva: Uma questão a ser respondida no projeto de qualquer EAP é quando parar de quebrá-lo em elementos menores.

- Se os elementos finais da EAP são definidos de forma muito abrangente, não deve ser possível rastrear eficientemente a performance do projeto.

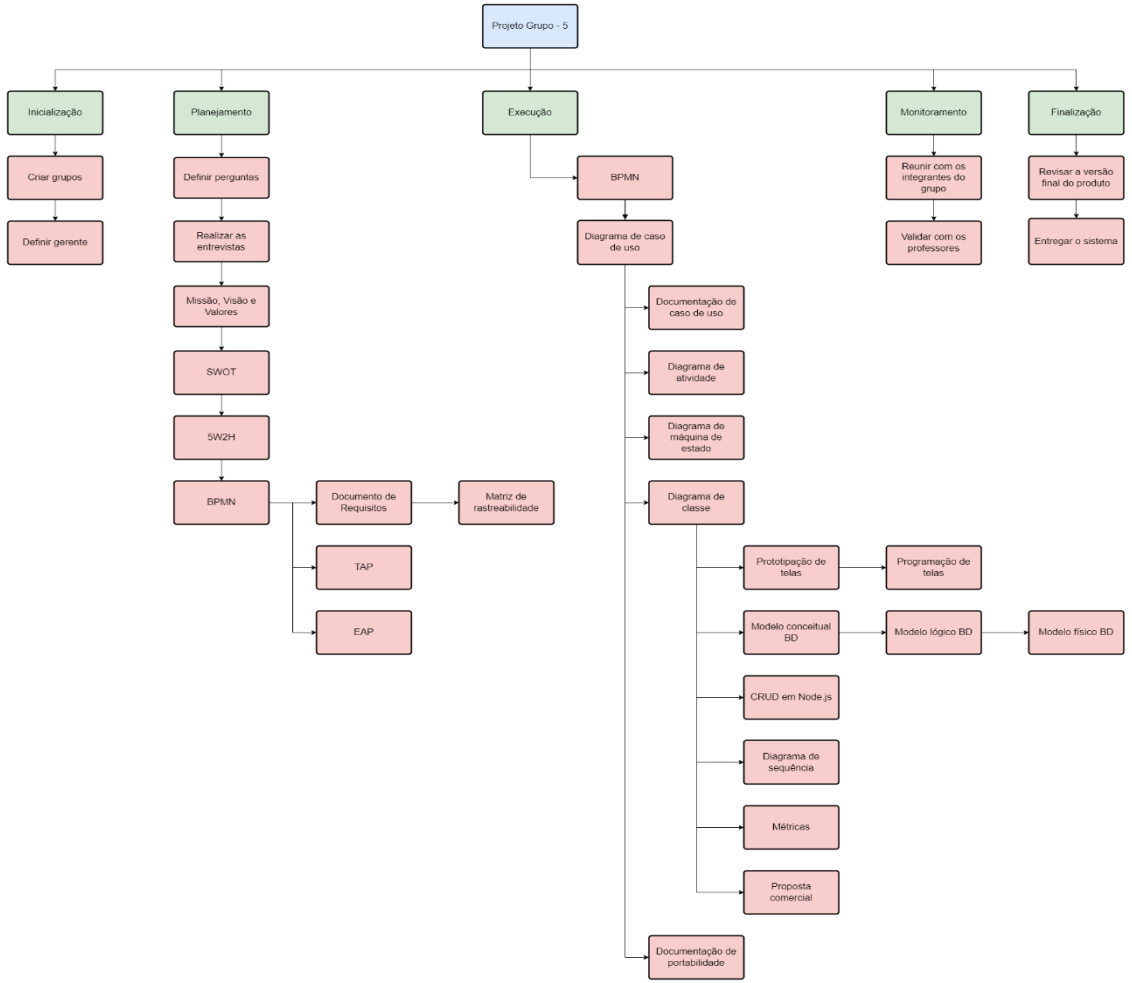
ENGENHARIA DE SOFTWARE

- **# Estrutura Analítica de Projetos (EAP)**

- A EAP pode ser expressa até algum nível de interesse. Três níveis são o mínimo recomendado, com níveis adicionais para, e somente para, itens de alto custo ou de alto risco, e dois níveis de detalhe em casos como o de engenharia de sistemas ou gestão do programa, com os exemplos que mostram padrões de E.A.P. com profundidade variável.

ENGENHARIA DE SOFTWARE

- **<u>#(EAP) ..... Exemplo:</u>**



ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

- O termo de abertura do projeto, também conhecido como TAP, é um documento formal onde contém informações detalhadas sobre o projeto, este é o documento no qual se autoriza o início do projeto.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

- O termo de abertura do projeto estabelece uma parceria entre a organização executora e a organização solicitante. No caso dos projetos externos, um contrato formal é normalmente a forma preferida de estabelecer um acordo.
- Neste caso, a equipe do projeto torna-se o fornecedor que responde às condições de uma oferta de compra de uma entidade externa.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

- Um termo de abertura do projeto é também usado para estabelecer acordos internos no âmbito de uma organização para garantir a entrega nos termos do contrato. O termo de abertura do projeto aprovado inicia formalmente o projeto.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

- O gerente de projeto é identificado e designado o mais cedo possível, preferivelmente enquanto o termo de abertura está sendo desenvolvido e sempre antes do início do planejamento.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

- • O processo de desenvolver um documento que formalmente autoriza a existência de um projeto e dá ao gerente do projeto a autoridade necessária para aplicar recursos organizacionais às atividades do projeto.
- • O principal benefício deste processo é um início de projeto e limites de projeto bem definidos, a criação de um registro formal do projeto, e uma maneira direta da direção executiva aceitar e se comprometer formalmente com o projeto.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

Termo de Abertura do Projeto (TAP)

- O documento do Termo de Abertura do Projeto tem como objetivo formalizar o início do projeto e descrever todas as suas etapas, justificativas, premissas, riscos, restrições, marcos e partes interessadas (COUTINHO, 2020, *online*).
- Serão abordadas todas as etapas a serem seguidas para o desenvolvimento do *software* e compreensão das necessidades da empresa. Para descrição será feita a análise de cada tópico a fim de justificar sua importância para o projeto e integrante do grupo.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

1. Situação atual

- Atualmente, não existe padronização no encaminhamento de solicitações. Como observado no levantamento de questionário aos *stakeholders*, os principais meios de comunicação são o celular, gmail e ramal de ligações o que deixa as informações avulsas e de difícil compreensão. Dessa forma, a definição de metas e desempenho esperado não fica tão claro para os funcionários.

- Os prazos definidos, bem como o retorno desses processos, podem prejudicar o andamento de outro setor por sua demora ou falta de dados. Seria necessário então, um maior tempo para confirmar com os responsáveis, tornando todo o processo complexo e demorado.

- Um ponto positivo é que a empresa possui um forte espírito de equipe e uma cultura que favorece a implementação de novos procedimentos internos. A boa relação entre os colaboradores pode conduzir a uma mudança mais natural na maneira de estabelecer a resolução de problemas dentro da equipe.

- Com a análise das respostas também foi possível notar que às vezes é necessário o deslocamento excessivo dentro do prédio. O que pode ser apenas uma confirmação, acarreta a perda de produtividade ou atraso na entrega de procedimentos por não achar o responsável procurado ou demorar nesta locomoção.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

2. Justificativa do Projeto

- A justificação do projeto se na baseia em criar uma solução sistêmica para auxiliar o cumprimento de prazo de entrega de procedimentos e organização de autorizações, gerando maior facilidade de retorno dos dados e no gerenciamento do grande fluxo de informações passado entre colaboradores e setores da empresa.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

3. Propósito do Projeto e Metas

- O projeto tem como objetivo otimizar o tempo de resposta entre setores e colaboradores para que ambas as partes se beneficiem desta troca de informações. Essa comunicação deve ter suas informações armazenadas de forma centralizada, ou seja, utilizando o *software* como intermédio para ações de autorização, abertura de procedimento internos e esclarecimentos.

- Outro ponto a ser melhorado, diz respeito à forma como as informações são passadas, tornando-as mais claras e de fácil interpretação. Para isso, busca-se o

- preenchimento de campos obrigatórios na abertura de um chamado, garantindo que os envolvidos possam realizar o seu retorno dependendo das dúvidas presentes, ou falta de informações.

- Com o passar do tempo, esse retorno irá criar um padrão a ser seguido pela empresa ou equipe, dependendo da maneira como é conduzido, nível de prioridade, autorizações a serem confirmadas, definição de metas e prazos.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

3. Propósito do Projeto e Metas

- Por fim, foi observado que uma das grandes barreiras para tornar os procedimentos mais rápidos e eficientes, é a necessidade de deslocamento dos colaboradores para confirmação de dados ou documentos enviados com determinada liderança ou setor. O sistema fornece uma plataforma com diversas opções de comunicação, seja por *chat*, vídeo ou ligação, para que seja evitada a locomoção desnecessária, principalmente para situações de fácil resolução.
- O objetivo é tornar a comunicação interna de empresas mais eficiente e clara, com foco na organização de dados (solicitações, arquivos etc.) ou autorizações passadas por meio de banco de dados. Com base nas descrições anteriores as metas visadas são:

- Possibilitar a definição de prioridade do chamado e controle de *status* pelas lideranças.
- Criar sistema de acompanhamento do chamado aberto de forma que priorize o prazo de resolução.
- Garantir segurança e integridade nos dados enviados, principalmente para documentos e autorizações.
- Tornar a comunicação entre os colabores e setores diferentes da empresa mais claros e centralizados (*chat*, vídeo e áudio), de forma que se torne uma ferramenta padrão.
- Manter a interação e troca de experiência entre os colaboradores, mesmo evitando o deslocamento pelo prédio da empresa.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

4. Descrição do Projeto

- Este projeto está dividido em 5 etapas. A primeira se concentra na Pesquisa de Mercado e seguimento de atuação da empresa Cliente e criação de formulário de perguntas para a melhor compreensão da atual situação dos processos realizados e de pontos a serem trabalhados.
- A segunda etapa é responsável pelo registro das respostas fornecidas pelos colaboradores, reconhecimento de fraquezas e forças da empresa, definição de melhorias e validações com a criação do SWOT, 5W2H e BPMN.
- Em terceiro é a elicitação de Requisitos Funcionais e Não Funcionais com documentação, bem como a criação de Diagramas de Casos de Uso, Atividade e Sequência, juntamente com suas revisões.
- A quarta etapa possui foco na prototipação de telas seguindo os requisitos pré-estabelecidos, escolha do banco de dados e linguagem que melhor se adequem ao desempenho da solução sistêmica, também levando em consideração a experiência na utilização, segurança e produtividade.
- Na quinta e última etapa é feita a definição de funcionalidades e a apresentação da proposta comercial no final do projeto.

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

5. Premissas

- Premissa de um *software* intuitivo e com ótima operabilidade
- Cumprir com as atribuições e prazos estipulados
- Desenvolver o projeto visando a fácil implantação e manutenção, para auxiliar no cumprimento dos prazos de solicitações ou autorizações
- Validação dos artefatos produzidos com integrantes do projeto e orientador
- Revisão periódica da documentação para seu aprimoramento

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

6. Restrições

- Prazo de entrega do projeto
- Capacitação técnica dos colaboradores
- Tempo disponível dos membros de equipe
- Considerar recursos de *software* e *hardware* disponíveis pela empresa

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

7. Stakeholders

- Partes interessadas como colaboradores e lideranças da empresa

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

8. Riscos

- Segurança e integridade dos dados e grande fluxo de informações
- Concorrência de empresas que já utilizam sistemas de controle de chamados como diferencial
- Falta de acompanhamento do cronograma de entrega dos artefatos
- Falta de qualidade na entrega de artefatos ao decorrer do projeto, com documentação insuficiente
- Falta de coerência nos diferentes registros ao longo do desenvolvimento da aplicação

•

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

9. Marco

- O cronograma foi estipulado para as entregas iniciais, desde sua concepção, até a data de entrega do documento e apresentação do sistema, seguindo a data de início do projeto no dia 13/08/22 e a data de entrega da documentação até o dia 15/11/23

ENGENHARIA DE SOFTWARE

#TAP (Termo de Abertura do Projeto)

10. Responsabilidades

- Nome do gerente e dos demais alunos e suas responsabilidades

ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#Gestão da Qualidade - (Fonte: RAUL, Wazlawick – Eng. de Software – 11.3)

- **11.3.1. Walkthrough**
- Walkthrough é uma forma de avaliação do produto que utiliza uma equipe de especialistas na qual cada um faz uma análise prévia do produto. Depois, um grupo de pessoas reúne-se por cerca de duas horas para trocar impressões sobre o produto e sugerir melhorias.
- O produto tanto pode ser BPMN, UC, código-fonte como também modelos de classes, diagramas dos mais diversos tipos, como casos de uso, máquina de estados, modelos de negócio etc.

ENGENHARIA DE SOFTWARE

#Gestão da Qualidade - (Fonte: RAUL, Wazlawick – Eng. de Software – 11.3)

- **11.3.1. Walkthrough**
- Além dos analistas, a reunião de walkthrough deve contar preferencialmente com desenvolvedores e usuários, que poderão apresentar rapidamente respostas a eventuais dúvidas, como “este requisito devia ter sido implementado dessa forma mesmo?”.
- Ao término da reunião, os participantes votam pela aceitação do produto, pela aceitação com modificações parciais ou pela rejeição. Sempre que houver modificações recomendadas, o produto deverá passar por um novo walkthrough.

ENGENHARIA DE SOFTWARE

#Gestão da Qualidade - (Fonte: RAUL, Wazlawick – Eng. de Software – 11.3)

- **11.3.1. Walkthrough**
- É importante mencionar que a reunião deve seguir estritamente o planejamento inicial, mantendo a discussão produtiva e objetiva. O objetivo principal é avaliar os defeitos, e não os desenvolvedores. Além disso, a reunião não será usada para corrigir defeitos, apenas encontrá-los. O processo de reparação vai ocorrer depois.
- Erros triviais, como erros ortográficos ou estéticos, não necessitam de discussão. Apenas os erros mais graves.

ENGENHARIA DE SOFTWARE

#Gestão da Qualidade - (Fonte: RAUL, Wazlawick – Eng. de Software – 11.3)

- **11.3.1. Walkthrough**
- Em relação à psicologia das reuniões, devem ser observados os perfis a seguir:
- • Programadores gênios: especialmente aqueles gênios arrogantes, impacientes e de mente estreita, podem causar problemas. Devem ser valorizados, pois são capazes de detectar defeitos com facilidade (e alimentam seu ego com isso). O coordenador da sessão deve ter humildade e controle para não iniciar discussões com eles, nem deixar que outros o façam, pois isso tornará o trabalho improdutivo.

ENGENHARIA DE SOFTWARE

#Gestão da Qualidade - (Fonte: RAUL, Wazlawick – Eng. de Software – 11.3)

- **11.3.1. Walkthrough**
- Em relação à psicologia das reuniões, devem ser observados os perfis a seguir:
- • Pessoas defensivas e inseguras: deve-se ter cuidado com essas pessoas, pois frequentemente se sentirão atingidas pessoalmente pelas críticas feitas aos seus artefatos. Também é preciso tomar muito cuidado para que seja mantida a discussão sobre o produto, e não sobre os desenvolvedores. A reunião de walkthrough não é o momento para tentar resolver a vida deles.

ENGENHARIA DE SOFTWARE

#Gestão da Qualidade - (Fonte: RAUL, Wazlawick – Eng. de Software – 11.3)

- **11.3.1. Walkthrough**
- Em relação à psicologia das reuniões, devem ser observados os perfis a seguir:
- Conservadores: também poderão causar problemas algumas vezes, pois buscam se manter fiéis às tradições estabelecidas. Deve-se dar atenção às suas opiniões, porque a área de programação é muito sujeita a modismos, mas deve-se também procurar evitar que iniciem discussões improdutivas.

ENGENHARIA DE SOFTWARE

#Gestão da Qualidade - (Fonte: RAUL, Wazlawick – Eng. de Software – 11.3)

- **11.3.1. Walkthrough**
- Em relação à psicologia das reuniões, devem ser observados os perfis a seguir:
- Alienados: não estão interessados no mundo real e primam mais pelo processo do que pelo produto, podendo tornar-se um incômodo sério. O coordenador sempre deve ter em mente que o processo só é útil quando ajuda a produzir da melhor forma possível. Regras existem porque há objetivos a alcançar, e não porque foram ditadas por algum guru da computação, mas, muitas vezes, os alienados não percebem isso.

ENGENHARIA DE SOFTWARE

#Gestão da Qualidade - (Fonte: RAUL, Wazlawick – Eng. de Software – 11.3)

- **11.3.1. Walkthrough**
- Em relação à psicologia das reuniões, devem ser observados os perfis a seguir:
- Enfim, muitos problemas interpessoais poderão surgir nas primeiras reuniões. Por isso, é necessário que o coordenador seja experimentado e competente na condução das reuniões para que, com o tempo, a equipe aprenda a se manter estritamente focada em seu objetivo, que é a detecção de defeitos nos artefatos.

ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

CONCEITO:

- É uma linguagem visual utilizada para modelar sistemas, sejam eles computacionais ou não.

IMPORTÂNCIA:

- É a linguagem padrão de modelagem de software adotada internacionalmente pela indústria de Engenharia de Software.

CARACTERÍSTICAS:

- A UML é a ferramenta ideal para conceber, compreender, testar, validar, arquitetar e ainda identificar todos os possíveis comportamentos do sistema.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

ESCLARECIMENTOS:

- A UML não é uma linguagem de programação e sim uma linguagem de modelagem, cujo objetivo é auxiliar os engenheiros de software a definir as características do software, tais como seus requisitos, seu comportamento, sua estrutura lógica, a dinâmica de seus processos e até mesmo suas necessidades físicas em relação ao equipamento sobre o qual o sistema será implantado.
- A UML é independente tanto de linguagem de programação quanto de processos de desenvolvimento. Isso quer dizer que a UML pode ser utilizada para a modelagem de sistemas, não importa qual a linguagem de programação será utilizada na implementação do sistema, ou qual a forma (processo) de desenvolvimento adotado.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

HISTÓRIA DA UML:

- Embora a concepção da UML tenha sido influenciada por vários métodos de análise e projeto orientados a objeto, há particularmente três notações das quais a UML é derivada:
- 1. OOD (Projeto Orientado a Objetos) – Grady Booch
- 2. OMT (Técnica de Modelagem de Objetos) – James Rumbaugh
- 3. OOSE (Engenharia de Software Orientada por Objetos) – Ivar Jacobson.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- Em 1997, a OMG (Object Management Group) tornou a UML uma linguagem de modelagem padrão para aplicações orientadas a objeto. A UML utiliza vários diagramas de modelagem, onde o objetivo é fornecer múltiplas visões do sistema a ser modelado. Cada diagrama analisa o sistema sob determinada ótica.
- É importante destacar que, embora cada diagrama tenha sua utilidade, nem sempre é necessário modelar um sistema utilizando todos os diagramas, pois alguns deles possuem funções muito específicas (GUEDES, 2005, p.7).

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

Diagrama de Caso de Uso

- Esse diagrama procura demonstrar o comportamento externo do sistema, procurando apresentar o sistema através de uma perspectiva do usuário, demonstrando as funções e serviços oferecidos e quais usuários poderão utilizar cada serviço.
- O diagrama de Caso de Uso (Use Case) é um elemento gráfico usado para modelar o modo como às pessoas esperam usar um sistema. O diagrama descreve quem serão os usuários relevantes, os serviços que eles exigem do sistema e os serviços que eles precisam oferecer ao sistema.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- Casos de Uso são utilizados para identificar as regras do negócio e é uma excelente forma de entender o ponto de vista do usuário simplesmente pelo fato de que modela o que ele precisa executar.
- Internamente, um caso de uso é uma sequência de ações que permeiam a execução completa de um comportamento esperado para o sistema, ou seja, possibilita identificar os fatores críticos de sucesso do sistema.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- O diagrama de Caso de Uso pode ser aplicado a muitos tipos de desenvolvimento, incluindo sistemas manuais, mas é usado mais comumente para sistemas e sub-sistemas.
- Um caso de uso representa quem faz o que com o sistema, sem considerar o comportamento interno do sistema, ou seja, não deve se preocupar com o “como” das funções.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- Um caso de uso modela um objetivo que o sistema precisa realizar a fim de que tenha sucesso. Acima de tudo, um diagrama de Caso de Uso deve ser simples, porém não pode ser incompleto. Resumindo, os casos de uso têm por objetivo:
- Descrever um conjunto de atividades de um sistema sob o ponto de vista de seus atores;
- Decidir e descrever os requisitos funcionais do sistema;
- Fornecer uma descrição clara e consistente do que o sistema deve fazer;
- Delimitar o contexto de um sistema.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Identificar um caso de uso, portanto, envolve:**
- Descobrir um ator (alguém que influencia ou é influenciado pelo sistema);
- Verificar para esse ator ações das quais ele participaria;
- Agrupar tais ações de forma que possuam um nome em comum (geralmente um verbo no infinitivo)

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

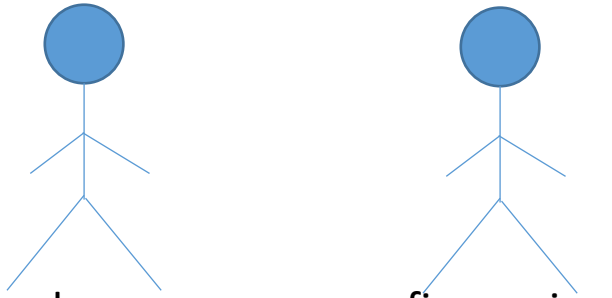
- Elementos do diagrama de Caso de Uso
- Seis elementos de modelagem compõem o diagrama de Caso de Uso:

1. atores
2. processos
3. associações
4. relacionamentos (<<inclusão>> e <<extensão>>)
5. generalização
6. documentação

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Ator:** Um papel desempenhado por uma pessoa, sistema, dispositivo ou mesmo uma empresa, que possui interesse na operação bem-sucedida do sistema.
- O termo ator refere-se a um tipo de usuário, onde esses usuários são pessoas que utilizam o sistema. Porém os usuários podem ser outros sistemas, dispositivos ou ainda empresas que trocam informações. Um ator é um papel que uma entidade, externa ao sistema, desempenha em relação ao sistema.
- Vários tipos de ícones podem representar o papel de ator no sistema. Por exemplo: Professor, Aluno, Financeiro, Secretaria.....



ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Caso de Uso:** Identifica um comportamento-chave do sistema. Sem esse comportamento, o sistema não preencherá os requisitos do ator. Cada caso de uso expressa um objetivo que o sistema precisa alcançar e/ou um resultado que ele precisa produzir.
- O foco está na finalidade e não na implementação. As sequências de ações realizadas por um caso de uso são as interações com os atores, e não os processos internos. Por exemplo:

ENGENHARIA DE SOFTWARE

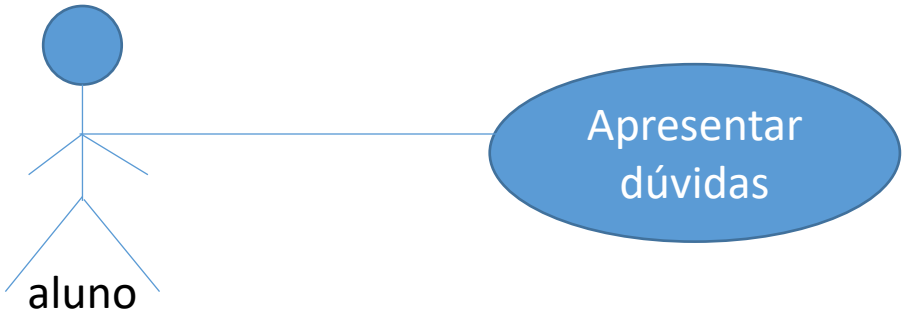
#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Existem dois tipos de casos de uso:**
- Primários: são aqueles que representam os objetivos principais do sistema. Exemplos: Realizar Venda, Matricular Aluno, etc.
- Secundários: são aqueles que não trazem benefícios diretos para os atores, mas que são necessários para que o sistema funcione adequadamente. Exemplo: proporcionar manutenção de cadastros, atualizar valores, etc.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Associação:** Identifica uma interação entre atores e casos de uso. O relacionamento é representado por uma linha entre um ator e um caso de uso. Cada associação torna-se um diálogo que deve ser explicado em uma narrativa de caso de uso. Diferentes atores podem acessar o mesmo caso de uso.
- O fato de um ator estar associado a um caso de uso significa que esse ator interage (troca informações) com o sistema. Um ator pode se relacionar com mais de um caso de uso do sistema.



ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Relacionamento include (inclusão):** O relacionamento de inclusão existe somente entre casos de uso. Este relacionamento é utilizado quando existe uma situação ou rotina comum a mais de um Caso de Uso.
- Quando isso ocorre, a documentação dessa rotina é colocada em um Caso de Uso específico para que outros Casos de Uso utilizem-se desse serviço, evitando-se descrever uma mesma sequência de passos em vários Casos de Uso.

ENGENHARIA DE SOFTWARE

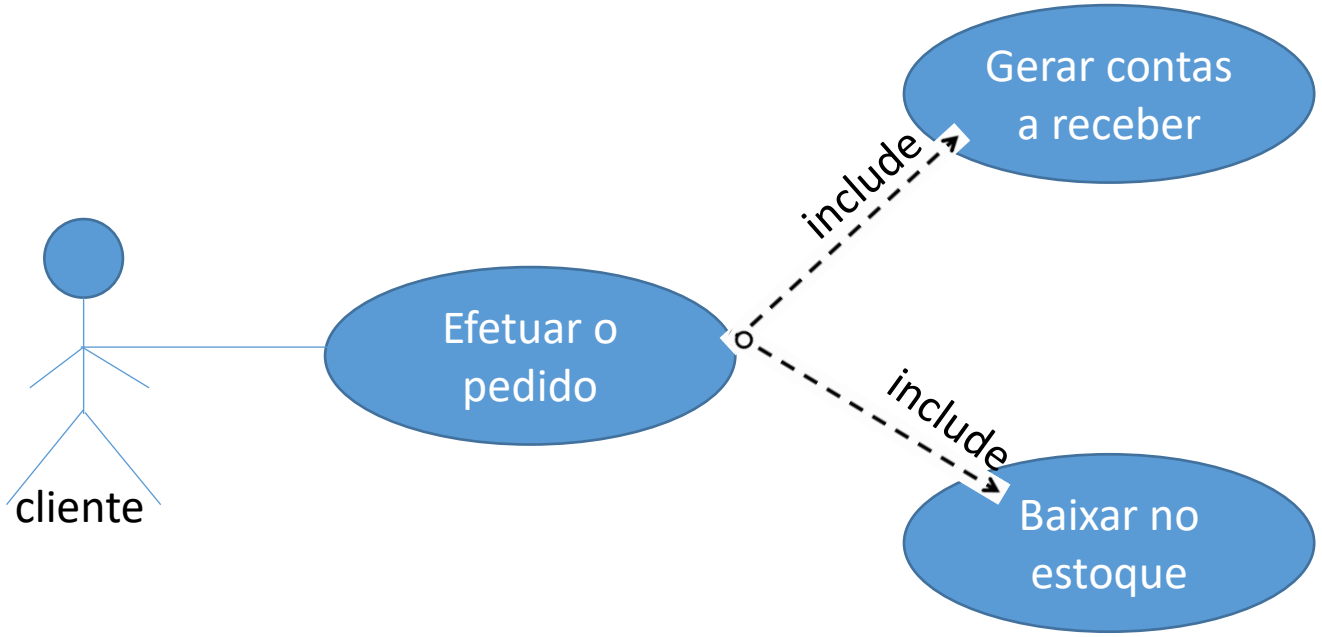
#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- Identifica um caso de uso reutilizável, que é incorporado incondicionalmente na execução de outro caso de uso. Os relacionamentos de Inclusão indicam uma ***obrigatoriedade***, ou seja, quando um determinado Caso de Uso possui um relacionamento de Inclusão com outro, a execução do primeiro obriga também a execução do segundo.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- Veja o exemplo:



ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Relacionamento extend (extensão):** Este relacionamento é utilizado para descrever cenários ***opcionais*** ou ***excepcionais*** de um Caso de Uso, que somente ocorrerão se uma determinada condição for satisfeita.
- Identifica um caso de uso reutilizável, que interrompe condicionalmente a execução de outro caso de uso para aumentar sua funcionalidade. Dessa forma, quando um ator opta por executar a sequência de interações definidas no caso de uso extend, este caso de uso é executado e o fluxo de interações volta ao caso de uso anterior.

ENGENHARIA DE SOFTWARE

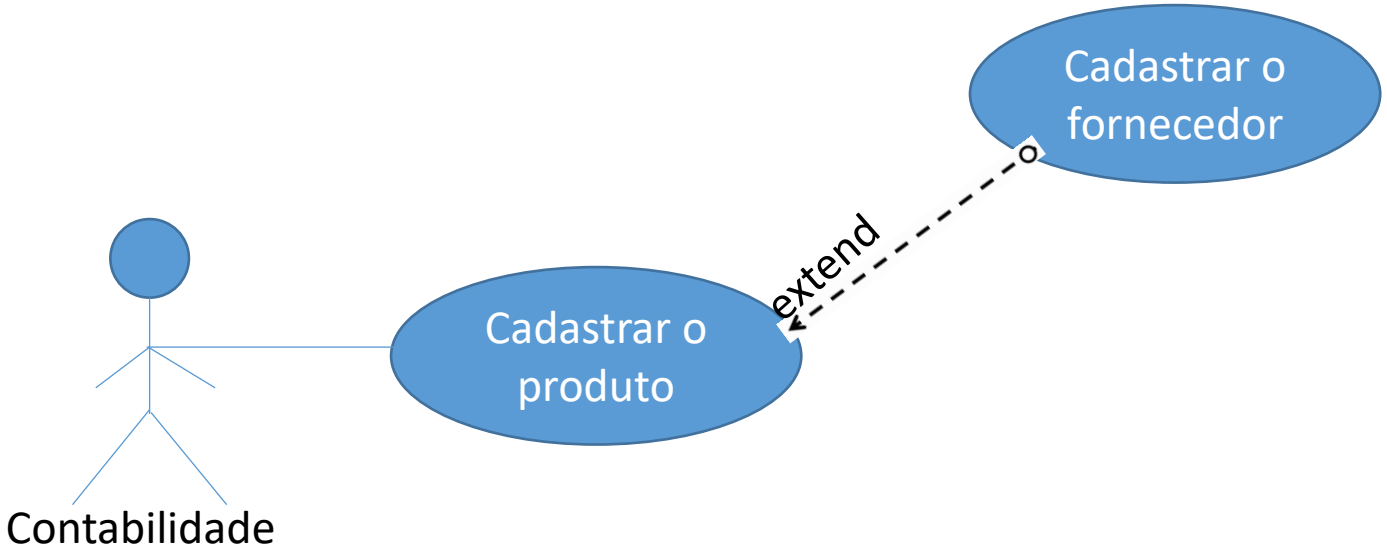
#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- A seta é desenhada da extensão para o caso de uso em execução. À medida que o caso de uso evolui e novas extensões são desenvolvidas, o caso de uso básico não precisa ser alterado a cada extensão nova ou revisado.
- O caso de uso que chama verifica uma condição para determinar se a extensão deve interromper o caso de uso que chama.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- Veja o exemplo:



ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Generalização:**
- A generalização é um relacionamento entre um ator A e um ator B. O ator A possui alguns casos de uso e ainda herda todas as características do ator B. Esta representação é indicada através de uma seta.
- Um objeto mais específico pode ser usado como uma instância do ator A. A generalização, também chamada de herança, permite a criação de elementos especializados em outros.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Documentação do Caso de Uso:**
- A documentação de um Caso de Uso costuma descrever por meio de uma linguagem simples a função do Caso de Uso, destacando quais Atores interagem com o mesmo, quais etapas devem ser executadas pelo Ator e pelo sistema para que o Caso de Uso execute sua função, quais parâmetros devem ser fornecidos e quais restrições e validações o Caso de Uso deve possuir.
- Uma narrativa de caso de uso é um documento escrito que explica um caso de uso em um comportamento do sistema, com início, meio (diálogo) e fim (término). A narrativa do caso de uso normalmente inclui os seguintes elementos:

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Nome do Caso de Uso (ID):** Deve ser uma ação sistêmica e escrito com o verbo no infinitivo
- **Descrição:** Breve descrição do Caso de Uso, ou seja, qual o objetivo principal do Caso de Uso
- **Atores Principal e Secundário:** Atores (papéis) que participam do Caso de Uso
- **Pré-condições:** Assim como as suposições, as pré-condições descrevem um estado do sistema que precisa ser verdadeiro antes que possa usar o caso de uso. Mas, ao contrário das suposições, essas condições são testadas pelo caso de uso antes de fazer algo mais. Se as condições não forem verdadeiras, o caso de uso não será executado.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Fluxo Principal (Cenário Principal):** O cenário do caso de uso refere-se a uma descrição passo a passo da interação entre o usuário (um ator ou outro caso de uso) e o caso de uso em execução. O fluxo principal descreve o que normalmente acontece quando o caso de uso é realizado.
- **Fluxo Alternativo (Cenário Alternativo):** Esses fluxos podem ser utilizados para descrever o que acontece quando o ator faz uma escolha alternativa, diferente da descrita no fluxo principal, para alcançar seu objetivo. Fluxos alternativos também podem ser utilizados para descrever situações de escolha exclusivas entre si (em que há diversas alternativas e apenas uma deve ser realizada).

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Pós-condições:** Descrevem o estado que o sistema alcança após o caso de uso terminar.
- **Regras de Negócio (Restrições/Validações**): Contêm as consistências que devem ser validadas durante o processo. As restrições são muitas vezes chamadas de Regras de Negócio. Exemplos: Um professor só pode estar lecionando disciplinas para as quais esteja habilitado; Senhas devem ter, no mínimo, seis caracteres, entre números e letras.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language)

#Exemplo de documentação

Caso de Uso – Cadastrar Aluno	
ID	UC 001
Descrição	Este caso de uso tem por objetivo cadastrar um novo aluno no sistema atendendo o RF 001
Ator Primário	Avaliador
Pré-condição	Ter perfil de avaliador no sistema
Cenário Principal	1. O use case inicia quando o avaliador seleciona o campo de cadastrar aluno 2. O sistema carrega o formulário para cadastro do aluno 3. O avaliador informa o CPF do aluno 4. O sistema consulta o Banco de Dados e verifica. 5. O sistema valida o CPF 6. O avaliador preenche os campos Nome, Sobrenome, Data de Nascimento, RG, Endereço, Numero da Casa, CEP, Cidade, Estado, Telefone, Dia de vencimento 7. O sistema deixa marcada a opção de Cliente Ativo 8. O sistema preenche o Avaliador do aluno logado 9. O sistema grava a Data de Inclusão 10. Uma mensagem de confirmação do cadastro é mostrada
Pós-condição	Não possui
Cenário Alternativo	*a – Em qualquer momento o atendente pode sair do sistema 5a – CPF já se encontra cadastrado
Inclusão	UC 034 – Solicitar Dados
Extensão	UC 029 – Solicitar Cadastramento

ENGENHARIA DE SOFTWARE

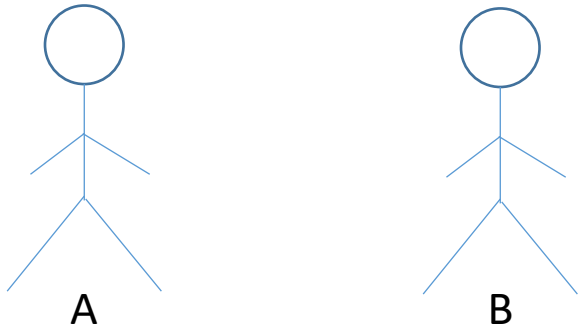
#UML – (Unified Modeling Language)

Revisando .....

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Ator:** Um papel desempenhado por uma pessoa, sistema, dispositivo ou mesmo uma empresa, que possui interesse na operação bem-sucedida do sistema.



ENGENHARIA DE SOFTWARE

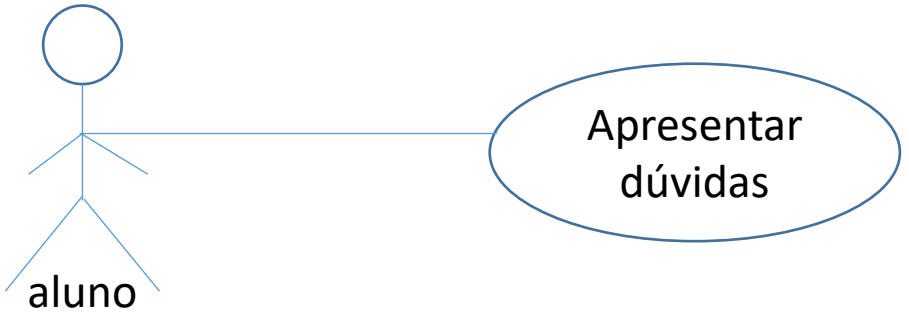
#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Caso de Uso:** Identifica um comportamento-chave do sistema. Verbo no infinitivo.

ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

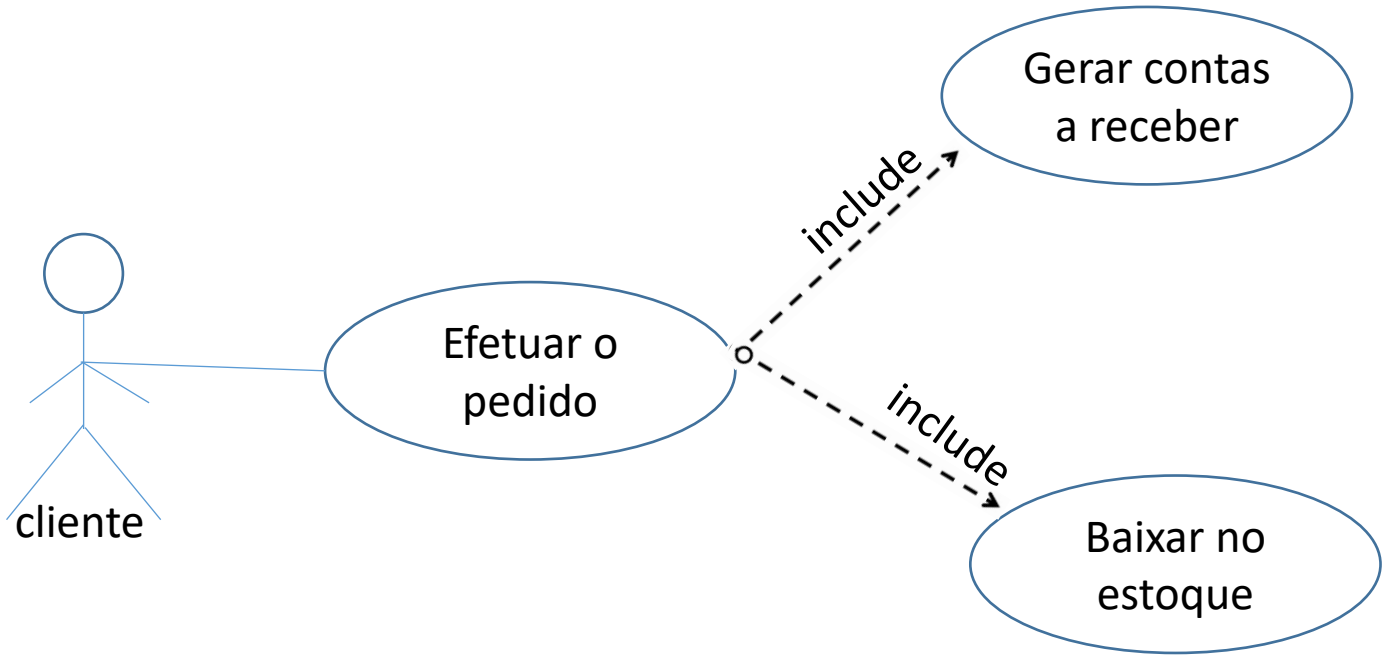
- **Associação:** Identifica uma interação entre atores e casos de uso. O relacionamento é representado por uma linha entre um ator e um caso de uso.



ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

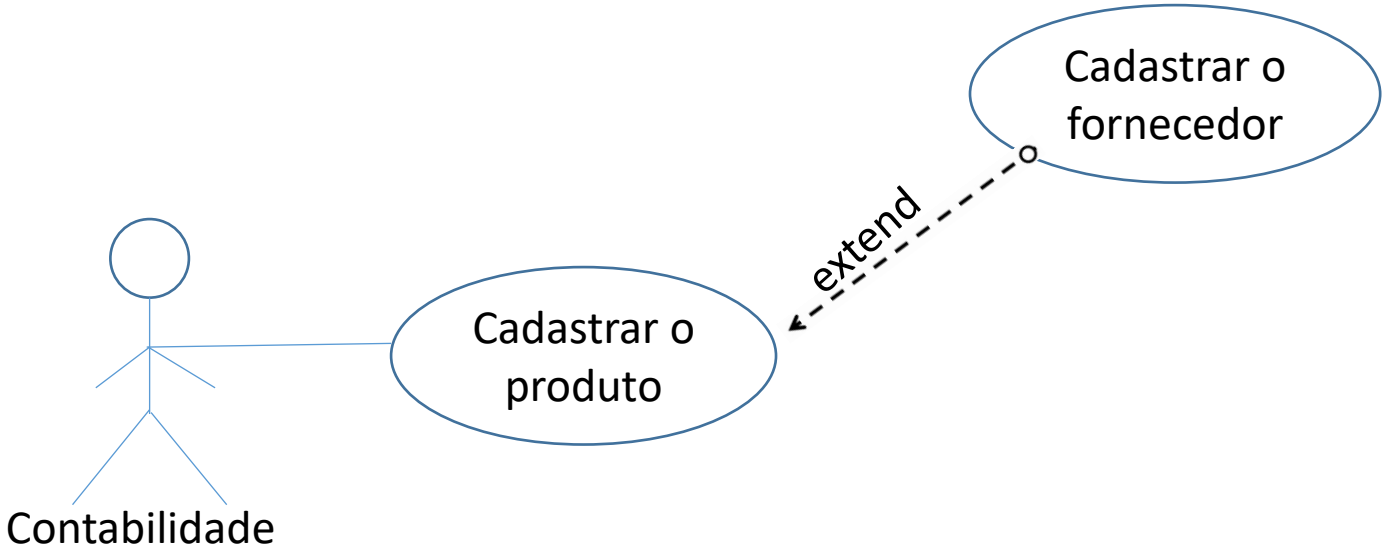
- **Relacionamento include (inclusão):** O relacionamento de inclusão existe somente entre casos de uso. Ação obrigatória.



ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Relacionamento extend (extensão):** Este relacionamento é utilizado para descrever cenários ***opcionais*** ou ***excepcionais*** de um Caso de Uso, que somente ocorrerão se uma determinada condição for satisfeita.

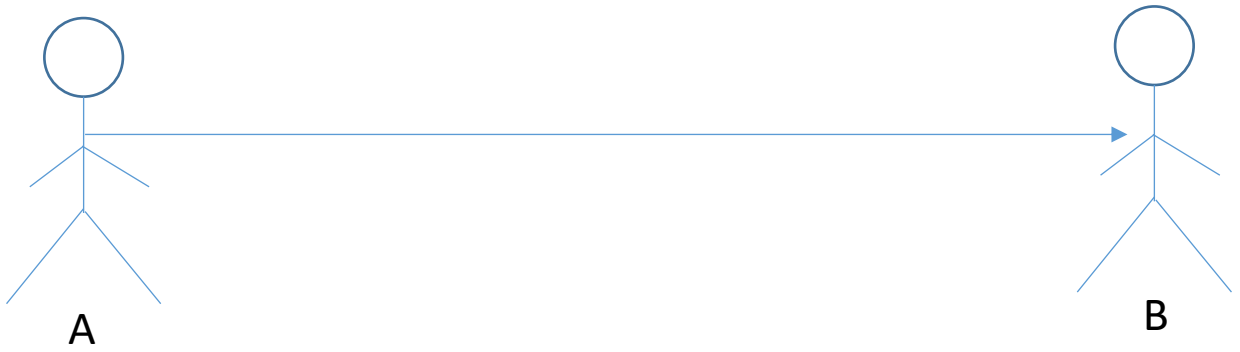


ENGENHARIA DE SOFTWARE

#UML – (Unified Modeling Language) – Fonte: Pressman, Roger S. – 8ª ed. – 2016

- **Generalização:**

- A generalização é um relacionamento entre um ator A e um ator B. O ator A possui alguns casos de uso e ainda herda todas as características do ator B.



ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#Diagrama de Atividades

- **Conceito:** representa o fluxo de relacionamentos entre os processamentos.
- **Importância:** contempla a ação dos requisitos (funcionais e não funcionais).
- **Aplicabilidade:** descreve a ação de cada RF-RNF utilizado no sistema. Proporciona ao implementador/modelista uma visão geral de cada ação.
- **Atenção:** atividades, ação, decisão, relatório.

ENGENHARIA DE SOFTWARE

#Diagrama de Atividades

- Os **diagramas de atividades** são as representações mais próximas de alguns diagramas tradicionais, como os fluxogramas e os diagramas de fluxo de dados (DFDs). O elemento básico é a **atividade**, uma especificação de comportamento executável como execução sequencial e concorrente de unidades subordinadas, que podem incluir atividades aninhadas, até chegar às **ações**, que representam atividades no nível do modelo.

ENGENHARIA DE SOFTWARE

#Diagrama de Atividades

- Os diagramas de atividades são formados por **nodos de atividade** unidos por **fluxos**. Nodos de atividade incluem ações; nodos estruturados, que contêm outros nodos; **nodos de controle**, que coordenam o fluxo entre outros nodos; e **nodos de objeto**, que representam os objetos consumidos e produzidos por ações. Note-se, portanto, que nodos de objetos representam **instâncias de classes**, e não diretamente as próprias classes.

ENGENHARIA DE SOFTWARE

#Diagrama de Máquina de Estados

- **Conceito:** é o estado em que o objeto se encontra durante o processo de execução do sistema.
- **Importância:** descreve a ação do objeto no momento de sua execução – portanto, um mesmo objeto poderá ser representado de diferentes estados (Include, Extend).
- **Aplicabilidade:** merece uma atenção especial – sistemas críticos.
- **Atenção:** estado, condição, transição, condição, estado final.
- **Ex.:** semáforo; controle de vôo (não deixar que aviões parem em uma mesma pista); sistema bancário (não deixar que os correntistas saquem ao mesmo tempo de uma mesma conta).

ENGENHARIA DE SOFTWARE

#Diagrama de Máquina de Estados

- **Estado** é uma condição ou situação, durante a vida de um objeto, na qual ele satisfaz a alguma condição, executa alguma atividade ou espera por algum evento. **Transição** é um relacionamento entre dois estados, que indica que um objeto no primeiro estado passa para o segundo estado quando certos eventos ocorrem, ou certas condições são satisfeitas.

ENGENHARIA DE SOFTWARE

#Diagrama de Máquina de Estados

- Uma **máquina de estados** é um grafo de estados e transições, que especifica a sequência de estados que um objeto percorre durante sua vida, em resposta a eventos. Máquinas de estados são representadas por **diagramas de estado**, herdados de notações bem anteriores à UML.
- As máquinas de estado são frequentemente vinculadas a classes, para descrever o comportamento das instâncias em relação aos eventos recebidos.

ENGENHARIA DE SOFTWARE

#Diagrama de Sequencia

- **Conceito:** Representa o fluxo dos processos, principalmente a comunicação entre eles.
- **Importância:** De acordo com a complexidade dos D. Classes e D. Objetos - os D. Seq. representam as suas relações.
- **Aplicabilidade:** representa interações entre objetos e cenários – propriedades e métodos.
- **Atenção:** esta interação são mensagens entre os casos de uso.
- **Composição:** objetos, módulos e linha de vida/cronograma.
- **Ex.: (**aluno -> solicitar documentação -> secretaria); (secretaria -> solicitar dados -> aluno); (aluno -> informar dados -> secretaria); (secretaria -> expedir documento -> aluno).

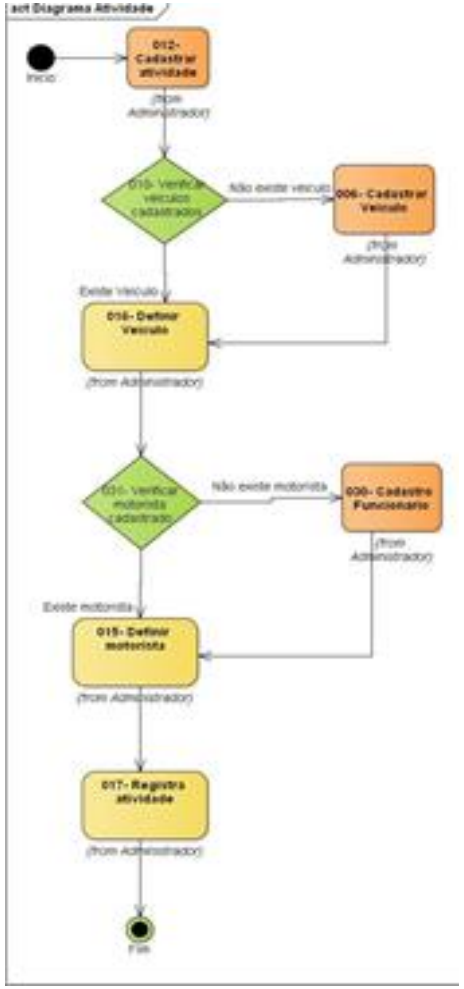
ENGENHARIA DE SOFTWARE

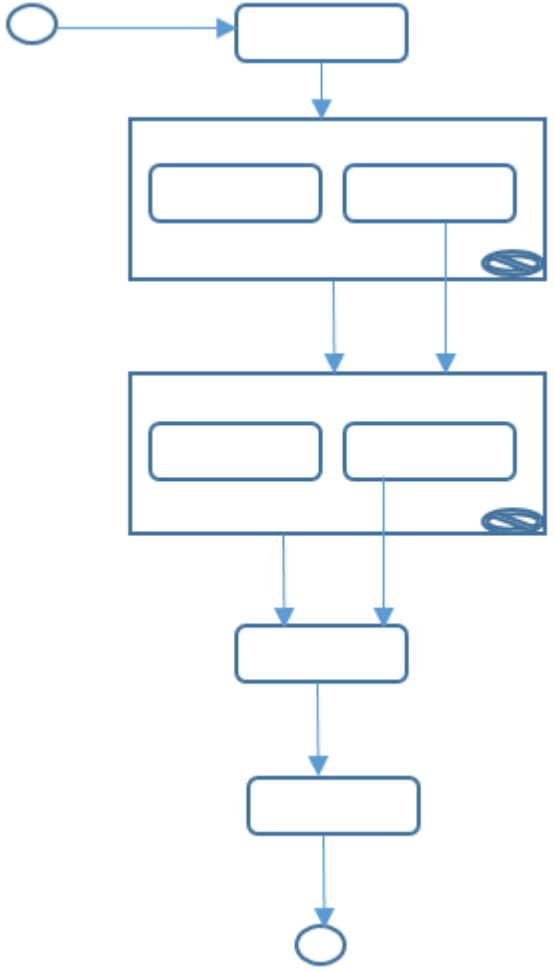
#Diagrama de Sequencia

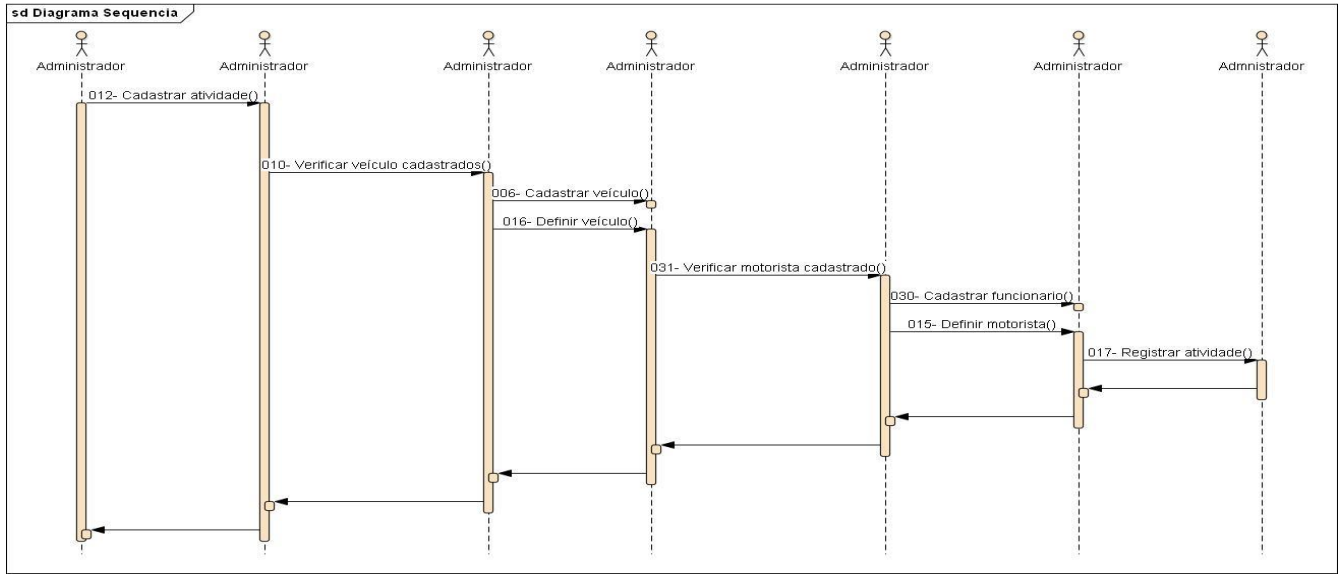
- Os diagramas de sequência enfatizam o ordenamento temporal das ações. Eles são construídos de acordo com as seguintes convenções:
- •linhas verticais, chamadas de **linhas da vida**, representam os papéis e os respectivos objetos;
- •setas horizontais representam as **mensagens** trocadas entre os objetos;
- •rótulos das setas indicam os nomes das operações invocadas pelas mensagens;
- •a posição na vertical mostra o ordenamento relativo das mensagens;

ENGENHARIA DE SOFTWARE

#Diagramas - exemplos

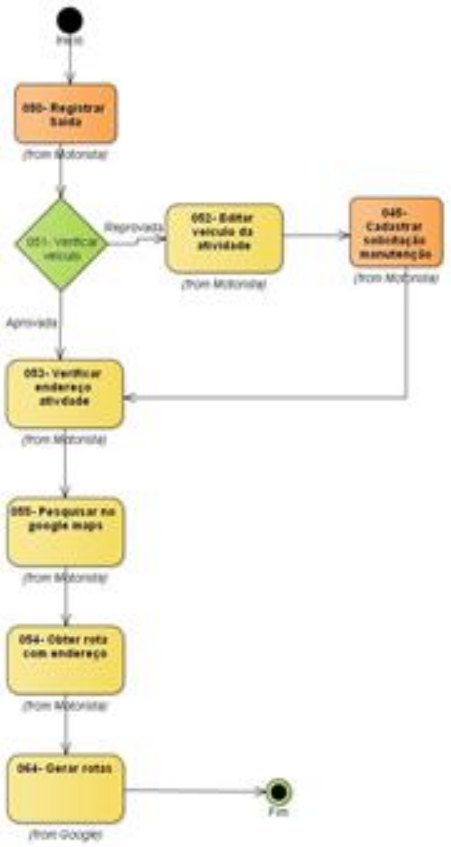


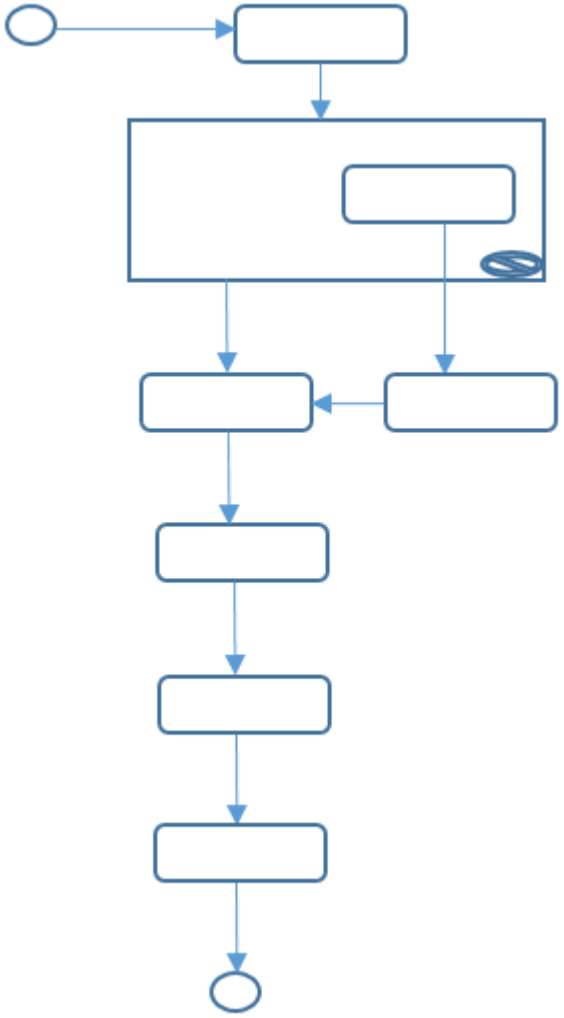


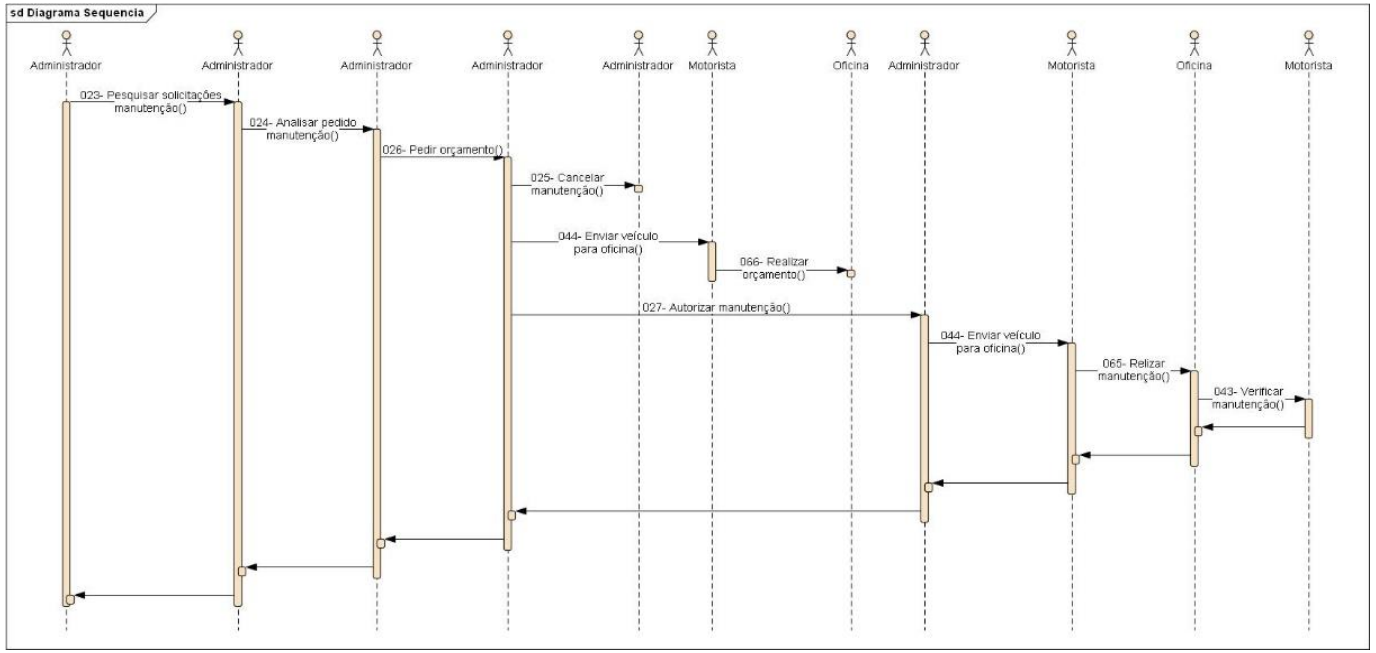


ENGENHARIA DE SOFTWARE

#Diagramas - exemplos

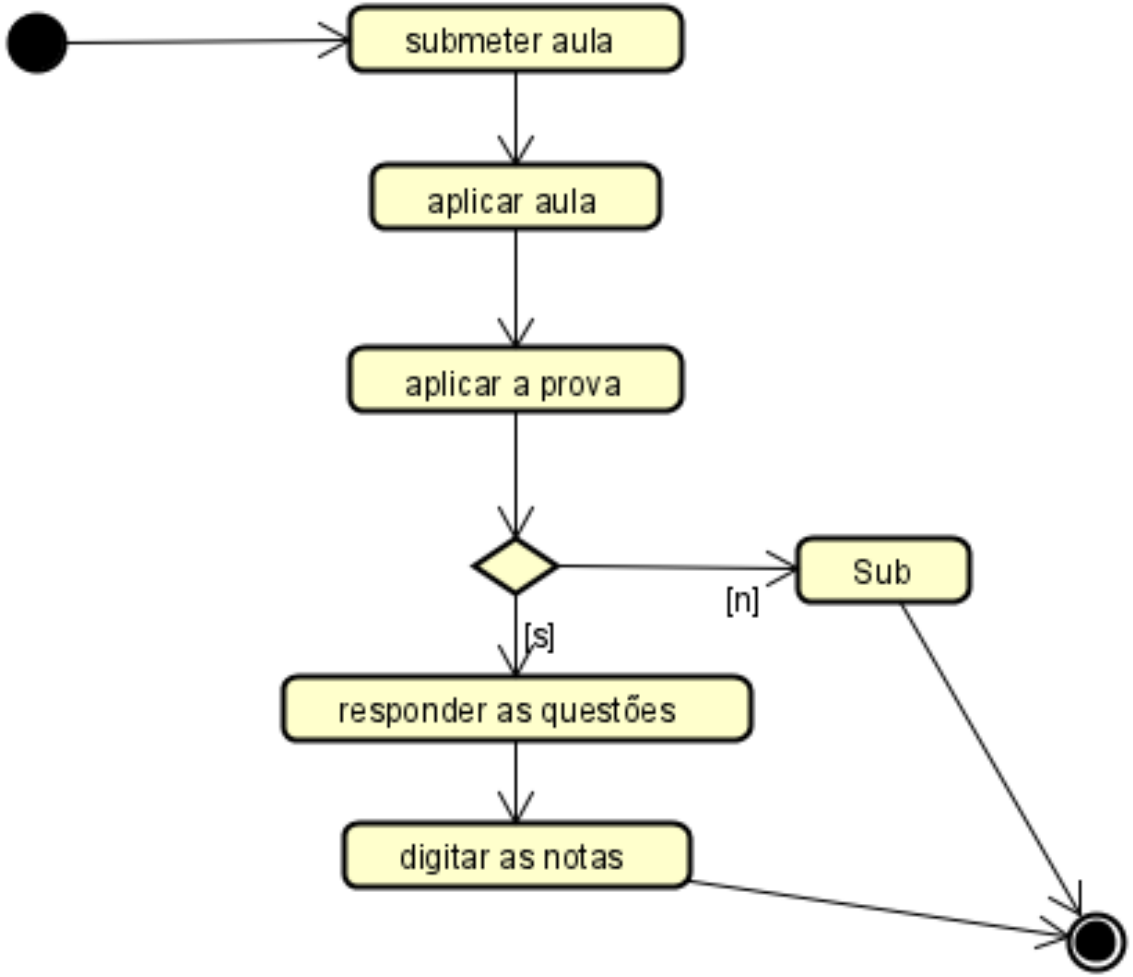






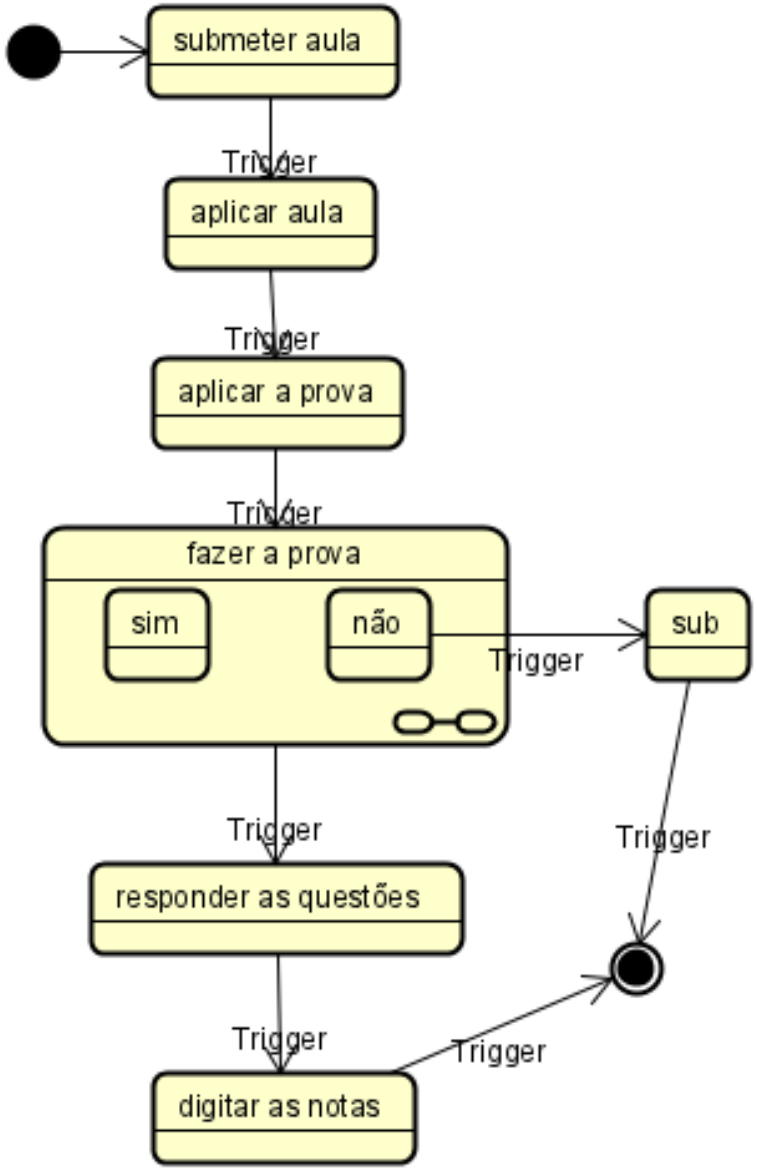
ENGENHARIA DE SOFTWARE

#Diagrama de Atividades



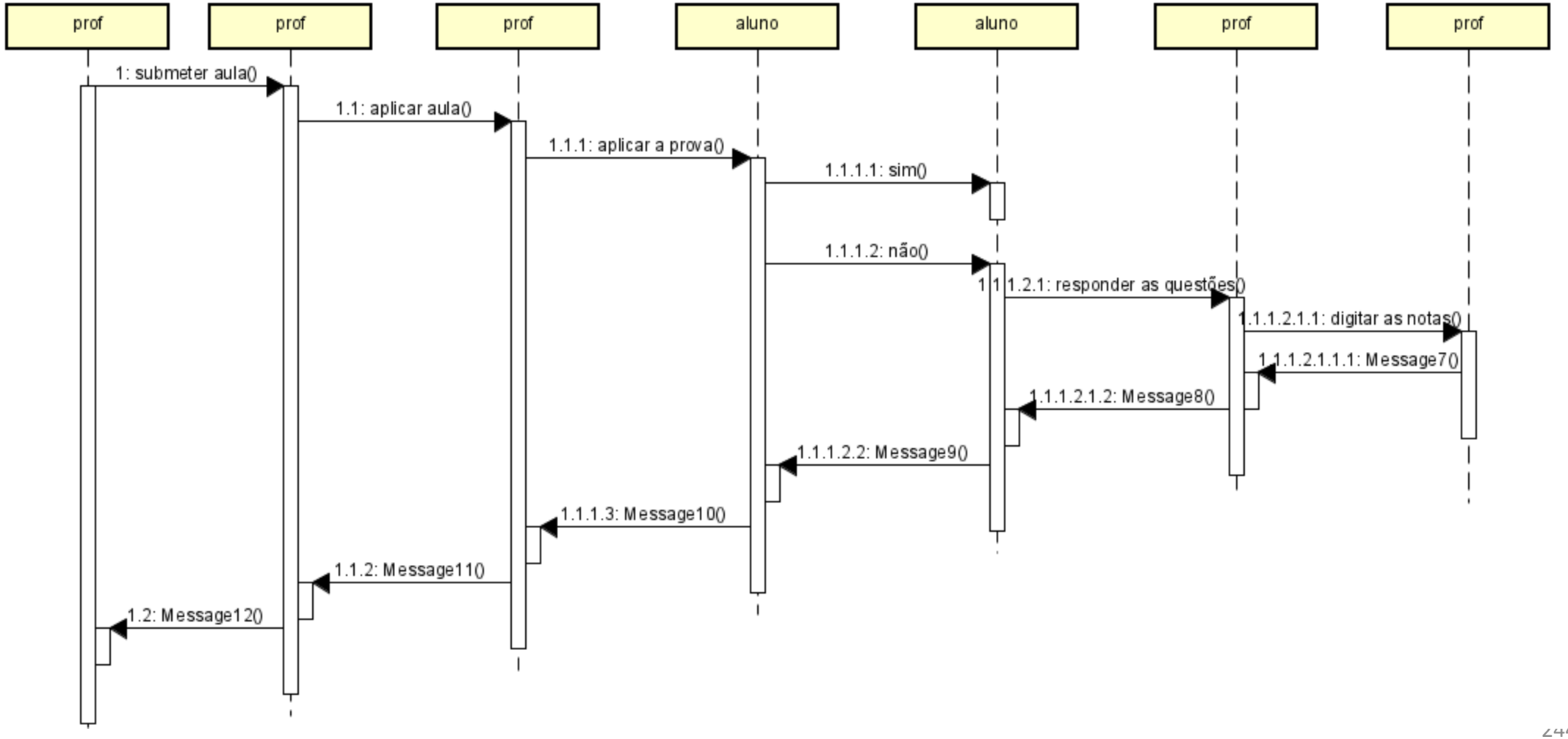
ENGENHARIA DE SOFTWARE

#Diagrama de Máquina de Estados



ENGENHARIA DE SOFTWARE

#Diagrama de Sequencia



ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

O que é uma matriz de rastreabilidade?

- Têm como principal objetivo a documentação e a definição das funcionalidades dos ICS do projeto, atendendo às expectativas dos interessados.
- A matriz de rastreabilidade é o modelo utilizado para visualizar a evolução de diferentes requisitos e seus pontos de cruzamento / utilização.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Como elaborar uma Matriz de Rastreabilidade?

- 1 – A primeira etapa da Matriz de Rastreabilidade é identificar quais são as necessidades e demais particularidades do projeto.
- 2 – Documentação de requisitos.
- 3 – Classificação dos requisitos.
- 4 – Regras de Negócio
- 5 – Caso de Uso
- 6 – Documentação da Matriz de Rastreabilidade.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

O que é uma matriz de rastreabilidade?

- A Rastreabilidade é definida como a técnica usada para prover relacionamento entre requisitos e regras, bem como entre requisitos e caso de uso. Ela auxilia na compreensão dos relacionamentos existentes entre requisitos do software ou entre artefatos de requisitos.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Para que serve a matriz de rastreabilidade?

- Uma matriz de rastreabilidade é o modelo utilizado para visualizar a evolução de diferentes requisitos e seus pontos de cruzamento.
- É natural que haja um refinamento dessas necessidades, porém a matriz não deixa a origem e o motivo do requisito isolado.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Um dos benefícios mais importantes de qualquer iniciativa de gerenciamento de requisitos dentro de uma metodologia para gestão de projetos está no gerenciamento e controle de mudanças no escopo desses projetos.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Com mecanismos eficazes para elicitar, documentar e controlar nossos requisitos, conseguimos garantir que temos um retrato adequado do escopo do nosso projeto no seu início e que conseguiremos acompanhar a evolução desse escopo de forma adequada. Uma das ferramentas de que dispomos para nos auxiliar nesse acompanhamento é a matriz de rastreabilidade.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

O que é uma matriz de rastreabilidade?

- Antes de tratarmos da matriz de rastreabilidade propriamente dita, precisamos definir claramente o que entendemos por rastreabilidade. Segundo o dicionário, a palavra rastrear significa “seguir o rastro ou pegada de”.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

O que é uma matriz de rastreabilidade?

- Aplicada ao gerenciamento de requisitos, significa registrar e manter o relacionamento entre os objetos que estamos gerenciando. Podemos manter a rastreabilidade entre requisitos e regras, por exemplo, para procurar saber especificamente qual objetivo de negócio cada requisito contribui para atender.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio))

A ISO recomenda o gerenciamento da rastreabilidade dos requisitos como uma das tarefas da área de conhecimento “Gerenciamento e Comunicação dos Requisitos”. O seu objetivo, é “criar e manter relacionamentos entre objetivos de negócios, requisitos, outros entregáveis da equipe, e componentes da solução para apoiar a análise de negócios e outras atividades”.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio))

A matriz de rastreabilidade surge então como uma ferramenta para facilitar a visualização dos relacionamentos entre requisitos e outros artefatos ou objetos.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Tipos de matriz de rastreabilidade

- É possível gerar matrizes de vários tipos. Roger Pressman sugere algumas no seu “Software Engineering” (ele usa o nome tabela e não matriz, mas o resultado é o mesmo):
- Matriz de rastreabilidade entre funcionalidades: mostra o relacionamento entre partes do sistema visíveis aos clientes/usuários.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Matriz de rastreabilidade de fontes: permite identificar a fonte, isto é, a origem de cada requisito.

- Matriz de rastreabilidade de dependências: essa é a forma mais comum da matriz, e identifica os relacionamentos entre os requisitos. Por ser a mais comum, quando dizemos apenas “matriz de rastreabilidade” geralmente estamos nos referindo à matriz de rastreabilidade de dependências.
- Matriz de rastreabilidade de subsistemas: relaciona os requisitos pelos subsistemas a que estão relacionados.
- Matriz de rastreabilidade de interfaces: identifica como os requisitos se relacionam com as interfaces internas e externas do sistema.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

- Modelo de matriz de rastreabilidade
- Qualquer que seja o tipo de matriz, ela sempre segue o mesmo modelo. Basicamente, coloca-se os objetos sendo rastreados nos eixos de uma tabela e marca-se os pontos de intersecção. No caso mais comum, da matriz de rastreabilidade entre requisitos ou de dependências, repete-se os requisitos nos eixos horizontal e vertical.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

É possível manter a matriz de rastreabilidade manualmente em uma planilha, mas é fácil perceber como isso rapidamente se torna inviável para sistemas um pouco mais complexos. Nesses casos, muitas ferramentas de software para gerenciamento de requisitos montam as matrizes automaticamente e as mantém atualizadas conforme atualizamos o banco de requisitos.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Imagine agora que na metade do desenvolvimento do seu projeto você recebe uma solicitação de mudança que envolve alterar um determinado requisito. Sem uma matriz de rastreabilidade, você pode não perceber todo o impacto dessa mudança no seu sistema e acabar tomando decisões equivocadas por não poder realizar uma análise de impacto completa e confiável.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Com a matriz, facilmente conseguimos identificar quantos e quais requisitos são afetados por qualquer alteração no sistema, e assim tornamos nossa avaliação de impacto muito mais eficaz.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

Rastreabilidade

- O enunciado dos requisitos é rastreável se permite a fácil determinação dos antecedentes e consequências de todos os requisitos. Dois tipos de rastreabilidade devem ser observados:
- •**rastreabilidade para trás:** deve ser possível localizar a origem de cada requisito. Deve-se sempre saber por que existe cada requisito, e quem ou o que o originou. Isso é importante para que se possa avaliar o impacto da mudança daquele requisito, e dirimir dúvidas de interpretação.
- •**rastreabilidade para a frente:** deve ser possível localizar quais os resultados do desenvolvimento que serão afetados por requisito. Isso é importante para garantir que os itens de análise, desenho, código e testes cubram todos os requisitos e nada mais do que eles, e para localizar os itens que serão afetados por uma mudança nos requisitos.

ENGENHARIA DE SOFTWARE

Matriz de Rastreabilidade (Requisitos x Regra de Negócio)

	RN 01	RN 02	RN 03
RF 001	X		
RF 003		X	
RF 004			X
RF 008	X		X

ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE

#Portabilidade

O F. A. P. (Formulário de Análise de Portabilidade) é um documento que possibilita a padronização no processo de levantamento de itens e necessidades estruturais.

Manter um roteiro para entrevistar o cliente é algo essencial. Esta lista de checagem apontará os diagnósticos técnicos, as abordagens de hardware e os recursos humanos.

ENGENHARIA DE SOFTWARE

#Portabilidade

Esta entrevista deve priorizar o foco na abstração do potencial de tecnologia que a empresa possui, visto que o sucesso das funcionalidades sistêmicas dependem deste conjunto de tecnologias:

Sistema – Hardware, Banco de Dados, Redes e Licenças de Software

... Importante ressaltar = a equipe de Recursos Humanos e Processos

ENGENHARIA DE SOFTWARE

#Portabilidade

- **F. A. P. (Formulário de Análise de Portabilidade)**

- **Todos os grupos (gerentes) deverão preencher este formulário**

- **O F.A.P. deve representar:**
  - **O que a empresa possui de tecnologia**
  - **As características de configuração para que a solução funcione eficientemente**
  - **Descrição dos usuários**

ENGENHARIA DE SOFTWARE

#Portabilidade

- **1. Dados da empresa**
  - **Nome da empresa (cliente):**
  - **Nome do contato:**
  - **Telefone:**

ENGENHARIA DE SOFTWARE

#Portabilidade

2.Infraestrutura

- **Hardware - Redes de Dados (quantidade e descrição)**
  - **Pontos de acesso**
  - **Meio de transmissão (cabeada, wi-fi)**
  - **Velocidade**
  - **Equipamento (qtdd de hubs, switches)**

ENGENHARIA DE SOFTWARE

#Portabilidade

3.Infraestrutura

- **Hardware - Redes elétrica (quantidade)**
  - **Estabilizadores**
  - **Nobreaks**

ENGENHARIA DE SOFTWARE

#Portabilidade

4.Infraestrutura

- **Hardware – Computadores Pessoais (quantidade e descrição)**
- **Arquitetura, processador, memória, HD**
- **Servidores (descrever as características – se web, arquivos, e-mail, domínio, BD)**

ENGENHARIA DE SOFTWARE

#Portabilidade

5.Infraestrutura

- **Hardware – Periféricos (quantidade e descrição)**
- **Impressora**
- **Scanner**
- **outros**

ENGENHARIA DE SOFTWARE

#Portabilidade

6.Licenças

- **Software – quantidade e descrição**
- **Sistemas operacionais**
- **Aplicativos (sistemas comerciais)**
- **Antivírus**
- **ERP**
- **Firewall**

ENGENHARIA DE SOFTWARE

#Portabilidade

7.Recursos Humanos

- **Quantidade de usuários**
- **Nível de alfabetização digital**
- **Se existe mudança constante de usuário**

ENGENHARIA DE SOFTWARE

#Portabilidade

8.Descreva a configuração mínima para que a solução (que o grupo está desenvolvendo) funcione eficientemente. (Arquitetura, processador, memória, HD) + (Servidores - descrever as características – se web, arquivos, e-mail, domínio, BD).

ENGENHARIA DE SOFTWARE

#Portabilidade

9.Nome dos componentes do grupo (entrevistadores)

ENGENHARIA DE SOFTWARE