Plano de curso – DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA

0000 – ENGENHARIA DE SOFTWARE I – PRESENCIAL - 80 Aulas

Competências Profissionais desenvolvidas neste componente
• Especificar os requisitos, projetar e documentar soluções de software baseadas no conhecimento apropriado de teorias, modelos e técnicas, observando as necessidades dos projetos. • Modelar e implantar processos de negócio, propor soluções de TI a fim de aumentar a competitividade das organizações.

Objetivos de Aprendizagem:

- Identificar as características de Sistemas de Informação, seus tipos, viabilidade técnica, características de custo, valor e qualidade da informação.
- Explicar as características de um sistema, seus componentes e relacionamentos.
- Compreender o ciclo de vida utilizando concepções do modelo cascata.
- Utilizar conceitos da UML na análise de requisitos e na elaboração de diagramas focando na modelagem de sistemas.

Ementa: Introdução à Análise de Sistemas. Modelos de Ciclo de Vida de Software. Modelos de Processos de Desenvolvimento de Software (Modelo em Cascata, Espiral e Prototipagem). Definição e classificação de Requisitos de Software (funcionais e não funcionais). Técnicas de Levantamento de Requisitos. Modelo de Negócios aplicado ao levantamento de Requisitos (Canvas). Estudo de Viabilidade. Técnicas de documentação. Metodologias para desenvolvimento de sistemas.

Metodologia proposta: Aulas Expositivas. Aprendizagem Baseada em Projetos/Problemas. Sala de Aula Invertida. Estudo de Caso Real. Nesta disciplina o professor é responsável por desenvolver um projeto Interdisciplinar integrando as disciplinas de Desenvolvimento Web I e Design Digital, seguindo o Manual de Projetos Interdisciplinares expedido pela CESU.

Instrumentos de avaliação:

Avaliação Formativa: Exercícios para prática. Análise e Resolução de Problemas acompanhado de rubrica de avaliação. Análise da documentação do projeto interdisciplinar.

Avaliação Somativa: Provas. Projetos. Avaliação em pares e Trabalhos Interdisciplinares. Validação do projeto para inclusão no Portfólio Digital do aluno.

Bibliografia Básica:

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. 3 ed. Rio de Janeiro: Elsevier, 2015.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software**. 8 ed. São Paulo: McGraw Hill Brasil, 2016.

SOMMERVILLE, Ian. **Engenharia De Software**. 10 ed. São Paulo: Pearson Brasil, 2019.

Bibliografia Complementar:

LARMAN, Craig. **Utilizando UML e padrões**. 3 ed. Porto Alegre: Bookman, 2007.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação**. 3 ed. Rio de Janeiro: Brasport, 2005.

WASLAWICK Raul. **Análise e Projeto de Sistemas de Informação Orientados a Objetos.** 2 ed. Rio de Janeiro: Elsevier, 2010.

Observações a considerar:

1-) Comunicação de alunos com alunos e professores:

- Um e-mail para a sala é de grande valia para divulgação de material, notícias e etc.
- Criação de grupo nas redes sociais também é interessante.
- Grupo em WhatsApp também é muito interessante.

2-) Uso de celulares:

Para o bom andamento das aulas, recomendo que utilizem os celulares em *vibracall*, para não atrapalhar o andamento da aula.

3-) Material das aulas:

A disciplina trabalha com NOTAS DE AULA que são disponibilizadas ao final de cada aula.

4-) Prazos de trabalhos e atividades:

Toda atividade solicitada terá uma data limite de entrega, de forma alguma tal data será postergada ou seja, se não for entregue até a data limite a mesma receberá nota 0.

5-) Qualidade do material de atividades:

- Impressas ou manuscritas:
  Muita atenção na qualidade do que será entregue, atividades sem grampear, faltando nome e número de componentes, rasgadas, amassadas, com rebarba de folha de caderno e etc. serão desconsiderados por mim.

- Digitais:
  Ao enviarem atividades para o e-mail da disciplina, SEMPRE no assunto deverá ter o nome da atividade que está sendo enviada, e no corpo do e-mail deverá ter o(s) nome(s) do(s) integrante(s) da atividade, sem estar desta forma a atividade será DESCONSIDERADA.

6-) Critérios de avaliação:

Cada trimestres teremos as seguintes formas de avaliação:
- 1 avaliação teórica;
- 1 avaliação prática (a partir do 2º trimestre, e as turmas do 2º módulo em diante);
- Seminários;
- Trabalhos teóricos e/ou práticos;
- Assiduidade;
- Outras que se fizerem necessário.

Caso o aluno tenha alguma menção I em algum dos trimestres será aplicada uma recuperação, que poderá ser em forma de trabalho prático ou teórico.

1. Introdução a Engenharia de sistemas

Engenharia de sistemas é um campo interdisciplinar da engenharia que foca no desenvolvimento e organização de sistemas artificiais complexos.

Uma definição de Engenharia de Sistemas provém do INCOSE (*International Council of Systems Engineering*) e conceitua que "a Engenharia de Sistemas é uma abordagem interdisciplinar que torna possível a concretização de 'Sistemas' de elevada complexidade. O seu foco encontra-se em definir, de maneira precoce no ciclo de desenvolvimento de um sistema, as necessidades do usuário, bem como as funcionalidades requeridas, realizando a documentação sistemática dos requisitos, e abordando a síntese de projeto e a etapa de validação de forma a considerar o problema completo"

A Análise do problema é o processo de compreensão do problema do mundo real e das necessidades do usuário, juntamente com a proposição de soluções que enderecem essas necessidades.

É comum existirem várias soluções e o trabalho do desenvolvedor consiste exatamente na exploração de todas elas e na identificação daquela que melhor se ajuste ao problema real.

Observação: É importante notar que nem sempre é necessário que exista um problema no sentido estrito, pois, muitas vezes, o desenvolvimento de sistemas é motivado pela vontade de aproveitar oportunidades determinadas pela tecnologia.

Isto coloca o binômio (automatização X inovação) em foco. Em outras palavras, problema e oportunidades são dois lados da mesma moeda. Para fugir desta "neurose", os desenvolvedores podem se concentrar na análise do problema.

O objetivo da análise do problema é ganhar uma melhor compreensão do problema a ser resolvido, antes de o desenvolvimento se iniciar.

Um problema pode ser definido como a discrepância entre como as coisas são e como se espera que elas sejam.

Nesta visão, existe uma série de maneiras de endereçar o problema além daquela óbvia, de desenvolver um (novo) sistema:

- proporcionar aperfeiçoamentos de sistemas já existentes;
- proporcionar soluções alternativas que não impliquem o desenvolvimento de um novo sistema;
- proporcionar treinamento adicional sobre soluções cobertas, mas não dominadas pelo público-alvo.

O objetivo do exercício de resolução de problemas consiste em ganhar uma melhor compreensão do problema a ser resolvido, antes de o desenvolvimento realmente começar. Os passos específicos para alcançar esta meta são os seguintes:

1) ganhar concordância mútua em relação à definição do problema;
2) compreender a raiz das causas, ou "o problema por trás do problema";
3) Identificar os stakeholders e os perfis de usuário;
4) definir a fronteira da solução do sistema;
5) identificar as restrições impostas sobre a solução.

2. Definição do problema

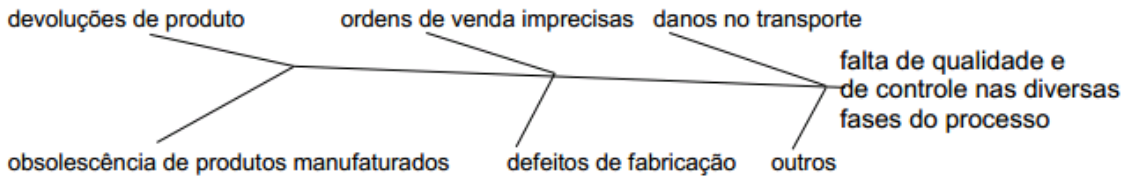
O problema pode ser definido por meio da utilização do formulário de definição de problemas, descrito a seguir.

Elemento	Descrição
O problema de...	Descrição do problema
afeta...	Identificação dos *stakeholders*
e resulta em...	Descrição do impacto na atividade, no mundo real, dos *stakeholders*
Poderia ser resolvido por meio de...	Descrição de uma possível solução
com os benefícios...	Descrição dos benefícios esperados

2.1 Identificação da raiz do problema

Com o intuito de descobrir "o problema por trás do problema", pode ser utilizada uma técnica do "Diagrama de espinha de peixe".

Exemplo: Imagine uma companhia que realize vendas pela Internet de itens de vários tipos de utilidade doméstica. Os problemas levantados numa primeira conversa informal podem levar à identificação do problema principal. Isto pode ser visto no seguinte Diagrama de espinha de peixe instanciado:



Às vezes não é tão fácil identificar a raiz do problema e, nesses casos, é necessário investigar profundamente cada um dos fatores e atribuir a cada um deles fatores de peso, até se chegar à causa principal.

2.2 Identificação do problema

Uma vez identificado o problema real, cada uma das causas que contribuem para gerar o problema deve ser examinada e, aquelas que tivessem solução computacional devem ter a solução desenvolvida.

Cabe chamar a atenção para o fato de que uma solução computacional pode corresponder, por exemplo, à reelaboração de um formulário de vendas de um sistema já existente.

No exemplo recém descrito, poderia se chegar a esta solução a partir da aplicação do formulário de definição de problemas, por exemplo, à causa “ordens de venda imprecisas”.

2.3 Identificação dos stakeholders

O termo stakeholders se refere a todos os potenciais interessados pelo sistema a ser desenvolvido. Em outras palavras, qualquer um que possa vir a ser materialmente afetado pela implementação de um novo sistema ou aplicação.

Os stakeholders usuários potenciais do sistema são fáceis de serem identificados. Os restantes, usuários indiretos, devem ser procurados na fronteira do ambiente onde a atividade à que o sistema dará apoio é realizada.

Os stakeholders podem ser classificados em 4 categorias, associadas ao sistema e não à atividade-fim da organização:

contribuintes: atores - são os que de fato executam a ideia principal e produzem os resultados; responsáveis – aqueles que concebem a ideia principal são os que mais contribuem com o sistema e mais se beneficiam dele;

fontes: clientes – aqueles que recebem os produtos indiretos do sistema; fornecedores – responsáveis por proporcionar as condições necessárias para o funcionamento do sistema;

mercado: parceiros – colaboram compartilhando recursos e juntando forças para resolver o problema de forma colaborativa; competidores – representam um desafio na medida em que eles concorrem ou entram em conflito com o sistema a ser desenvolvido;

comunidade: legisladores – aqueles responsáveis pelo estabelecimento das regras, sejam elas oficiais ou protocolos sociais; espectadores - compreende a comunidade que receberá os ganhos e ou as perdas decorrentes da utilização do sistema a ser desenvolvido e implantado.

Existem algumas perguntas que podem facilitar a tarefa de identificação dos stakeholders:

Quem (são e ou) serão os usuários do sistema?
Quem são os clientes (comprador) do sistema?
Quem mais será afetado pelas saídas que o sistema vai produzir?
Quem vai avaliar e aprovar o sistema quando ele estiver implementado e implantado?
Há qualquer outro tipo de usuário interno ou externo do sistema cujas necessidades devam ser consideradas?
Quem irá fazer a manutenção do sistema?
Tem mais alguém que possa vir a se importar com o que está sendo definido?

Exemplos de Perfis de usuário e stakeholders para o problema recém discutido.

Perfis de usuário direto: funcionários que entram os pedidos de compra, supervisor de pedidos de compra, pessoal do controle da produção, empregado que elabora a nota de compra.

Outros stakeholders: diretor e equipe da TI da organização, diretor financeiro, gerente de produção.

2.4 Exercícios de fixação do conteúdo.

1-) Utilizando o formulário de definição do problema, faça o levantamento do seguinte problema: Um almoxarifado de pequenas peças(cerca de 500.000 peças já existem em estoque) onde todo o processo é manual, e que 2 almoxarifes, e 1lider interagem nesta área, fornecendo peças para 20 líderes de uma produção.

2-) Elabore e descreva uma situação problema(implementação de um sistema), onde você possa identificar a raiz do problema através do diagrama de “Espinha de peixe”.

3-) Usando a situação problema do exercício anterior identifique os stakeholders.

Ao final, subam a tarefa em formato .doc na plataforma *Teams* em tarefas, o prazo será até a próxima aula dia 15/02/2022.

3. Definição das fronteiras do sistema

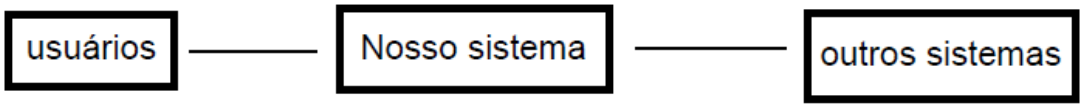
Após identificar todos os perfis envolvidos, devemos definir um sistema que possa endereçar o problema.

Ao considerarmos uma solução potencial, devemos determinar os seus contornos, ou seja, a fronteira entre a solução e o mundo real que a rodeia.

Neste intuito, dividimos o mundo em duas classes de coisas:

1. nosso sistema;
2. coisas que interagem com nosso sistema.

Numa primeira visão, a perspectiva do sistema poderia ser assim representada:



Esta divisão determina atores e papéis associados a eles em relação ao sistema.

Conceito: Um ator é qualquer pessoa externa ao sistema que interage com ele.

A identificação dos atores é uma atividade não trivial. Algumas perguntas ajudam a identificá-los.

Quem vai suprir, usar ou remover informação no ou do sistema?

Quem vai operar o sistema?

Quem vai fazer a sua manutenção?

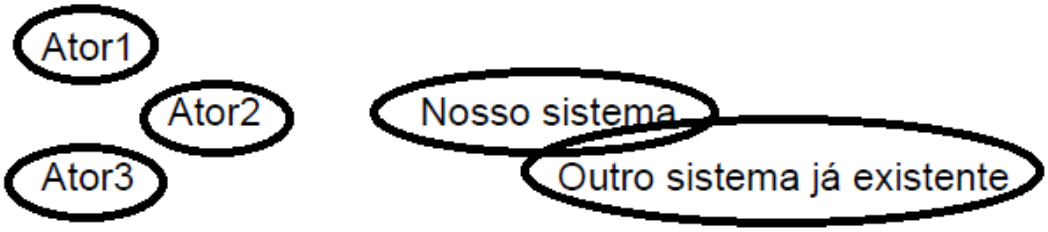
Onde o sistema vai ser usado?

Onde o sistema pega a informação? De onde o sistema retira a informação que necessita?

Que outros sistemas vão interagir com o sistema em desenvolvimento?

Após a obtenção das respostas às questões anteriores, uma visão mais apurada das fronteiras do sistema poderia ser:

O Nosso sistema pode ter diversos atores (ator1, ator2, ator3,...) interagindo com ele e pode, simultaneamente, ter uma interseção com um outro sistema computacional X.



Neste caso, a solução proposta seria composta de um novo sistema a ser desenvolvido e uma modificação a ser efetuada num sistema existente.

4. Determinação das restrições impostas ao sistema

Conceito: Uma restrição é uma limitação imposta ao grau de liberdade que o projetista tem no desenvolvimento da solução do problema.

Existem uma série de fontes potenciais de restrições que devem ser consideradas:

Fonte	Questões típicas
econômica	*Quais as restrições financeiras aplicáveis?* *Há custos de produtos vendidos ou alguma consideração sobre a composição de preço?* *Há questões sobre licenciamento envolvidas?*
política	*Há aspectos políticos internos ou externos que devam ser considerados?* *Há algum problema ou aspecto crítico entre departamentos?*
tecnológica	*Estamos limitados na nossa escolha de tecnologias?* *Precisamos nos ater a trabalhar sobre plataformas ou tecnologias já utilizadas?* *Temos proibição de usar alguma nova tecnologia?* *Alguém espera que a gente utilize alguns pacotes de software específicos?*
de sistemas	A solução a ser construída será parte do sistema existente? Devemos manter compatibilidade com as soluções existentes (internas ou externas)? Quais os sistemas operacionais e os ambientes aos que o nosso sistema deve dar suporte?
ambiental	*Existem restrições ambientais ou regulatórias?* *Há restrições legais?* *Quais são os requisitos de segurança?* *Que outros padrões devemos seguir?*
temporal e de demais recursos	*Tem um cronograma definido?* *Estamos limitados aos recursos existentes?* *Podemos utilizar mão de obra externa?* *Podemos estender os recursos? Em caso afirmativo, de forma transitória ou permanente?*

Uma tabela ajuda a registrar as restrições para posterior exame:

Risco	Restrição	*Rationale* (motivo)
procedimentos	Uma cópia em papel do recibo deve ser mantida pelo prazo de um ano.	O risco de perda de dados eletrônicos no período de implantação é grande.
Desperdício (equipamento)	...	
Desperdício (humano		
...		

5. Modelagem do negócio

Há uma série de questões cujas respostas devem ficar claras antes de partir para o desenvolvimento de um sistema:

Por que construir um sistema?

Onde ele será instalado?

Como podemos determinar qual a funcionalidade máxima para cada nó particular do sistema?

Quando devemos usar passos de processamento “manual” (humano!) ou outros caminhos que não o de sistemas computacionais?

Quando devemos considerar a reestruturação da organização como passo prévio à resolução do problema?

A técnica de modelag
em do negócio nos ajuda a resolver estas questões.
Os objetivos principais da modelagem do negócio são os seguintes:
1) entender a estrutura e a dinâmica da organização onde o sistema vai se inserir;
2) assegurar que os clientes, os usuários os desenvolvedores tenham consenso sobre o problema;
3) entender como empregar novos sistemas para potencializar a produtividade e quais os sistemas existentes podem vir a ser afetados pelo novo sistema.

6. Escolha da técnica certa

A indústria definiu como padrão o Unified Modeling Language (UML).
Conceito: A UML é uma linguagem para visualizar, especificar, construir e documentar os artefatos de um sistema substancialmente de software.
A UML proporciona um conjunto de elementos, notações, relacionamentos e regras de uso que podem ser utilizadas no processo de modelagem de sistemas computacionais.

7. Análise de Requisitos de Sistemas

Roger S. Pressman em *"Engenharia de Software"* explica:

"Uma compreensão completa dos requisitos de software é fundamental para um bem-sucedido desenvolvimento de software. Não importa quão bem projetado ou quão bem codificado seja, um programa mal analisado e especificado desapontará o usuário e trará aborrecimentos ao desenvolvedor."

A tarefa de análise de requisitos é um processo de descoberta refinamento, modelagem e especificação. O escopo do software, inicialmente estabelecido pelo engenheiro de sistemas é refinado durante o planejamento do projeto de software, é aperfeiçoado em detalhes. Modelos do fluxo de informação e controle exigido, comportamento operacional e conteúdo de dados são criados. Soluções alternativas são analisadas e atribuídas a vários elementos de software. Tanto o desenvolvedor como o cliente desempenha um papel ativo na análise a especificação de requisitos. O cliente tenta reformular um conceito de função e desempenho de software, às vezes nebuloso, em detalhes concretos. O desenvolvedor age como indagador, consultor e solucionador de problemas.

A análise e especificação de requisitos podem parecer uma tarefa relativamente simples, mas as aparências enganam. O conteúdo de comunicação é muito elevado.

Abundam as chances de interpretações errôneas e informações falsas. A ambiguidade é provável. O dilema com o qual se defronta um engenheiro de software pode ser mais bem entendido repetindo-se a declaração de um cliente anônimo: "Sei que você acredita que entendeu o que acha que eu disse, mas não estou certo deque percebe que aquilo que ouviu não é o que eu pretendia dizer..."

A análise de requisitos é uma tarefa da engenharia de software que efetua a ligação entre a alocação de software em nível de sistema e o projeto de software (Figura a seguir). A análise de requisitos possibilita que o engenheiro de sistemas especifique a função e o desempenho do software, indique a interface do software com outros elementos do sistema e estabeleça quais são as restrições de projeto que o software deve enfrentar. A análise de requisitos permite que o engenheiro de software (muitas vezes chamado de analista nesse papel) aprimore a alocação de software e construa modelos do processo, dos dados e dos domínios comportamentais que serão tratados pelo software.

A análise de requisitos proporciona ao projetista de software uma representação da informação a da função que pode ser traduzida em projeto procedimental, arquitetônico e de dados. Finalmente, a especificação de requisitos proporciona ao desenvolvedor e ao cliente os critérios para avaliar a qualidade logo que o software for construído.