Aula 1

Paulo Cézar Stadzisz
stadzisz@utfpr.edu.br
Sala: Lit - Bloco D - **2° Andar**
Fone: 3310-4764

[1]Conteúdo:

1. Introdução a Engenharia de Software.
    a. 3 semanas -> 12 horas
2. Processos de Desenvolvimento de Software.
    a. 7 semanas -> 28 horas
    b. 1° nota parcial (50%)
        i. Prova Escrita =~ 3 de Outubro
        ii. Trabalhos -> 4 horas
3. Especificação de Requisitos.
    a. 8 semanas -> 72 horas
    b. 2° nota parcial (50%)

Referências:

- Roger Pressman
- Sommerville
  (VERGONHA /\, não cite, pior que tecmundo)

[1]

Engenharia de Software

É a área de conhecimento que envolve o emprego de métodos, processos, técnicas, conceitos (teorias), modelos (padrões), linguagens (notações), e ferramentas para o desenvolvimento sistemático **de software**.
-> Definição generalizada de Engenharia

Área do conhecimento - Engenharia de software por si só é uma área do conhecimento. Ela é completa (como Medicina).

Método - Método é quase a mesma coisa que processo, mas é algo mais conceitual/teórico.

Técnica - Forma de fazer algo pequeno (uma parte de um processo).

Conceitos (Teorias) - São usados para ajudar no desenvolvimento do software, “descartado” no projeto finalizado.

Modelos (Padrões) - Reuso de soluções já conhecidas (Exemplo: Carro, 90% das peças reusadas). Reutilização de Software.

Ferramentas - Compiladores, Editores de Texto, ferramentas de modelagem UML, ferramentas de gestão de requisitos, etc.

ACM/IEEE

1. Computação/Computing
    a. Grande area temática
    b. Áreas de Conhecimento
        i. Engenharia de Software -> Engenheiro/a de Software
        ii. Engenharia de Computação (Computer Engineering)[2]
        iii. Ciência de Computação -> Cientista de Computação
        iv. Tecnologia de Informação (Information Technology)
        v. Sistemas de Informação (Information System)

- A Engenharia de software é a área de conhecimento mais mal compreendida!

CMU -> Software Engineering Institute - CMMI (Programa de qualidade)
Softex - Instituto -> IMPS-BR (Melhoria do Processo de Software - BR)

CMMI - Pesado para empresas pequenas.
IMPS-BR - Para empresas pequenas (Apesar de poder ser usada para grandes), conhecida mundo a fora.
CERTICS - foi criada para comprovar se um s*oftware* é resultado de desenvolvimento e inovação tecnológica no País.

Maior empresa de Software do Brasil - TOTVS

TIVIT
STEFANINI
CINQ

ABES (Associação Brasileira de Engenharia de Software)

[2] Curiosidade: Engenharia de Computação != Engenharia de Computador

Qual a importância de SW?

- Vital, tem em todo lugar

Problemas na produção de software

- Baixa produtividade
  - LOC = Lines Of Code
  - É uma atividade intelectual complexa
  - Falta de ferramentas mais eficazes
- Altos custos
  - Devido a baixa produtividade
- Complexidade crescente
  - Reduz a produtividade mas aumenta as linhas de código
- Pouca mão de obra qualificada
- Em média 50% dos projetos são falidos

Aula 3

O que é software?

- Arquivos de código fonte (Programas)
- Bases ou Estruturas de Dados
- Arquivos Executáveis
- Documentação Técnica (Modelos, Especificações, API, Ferramentas, Ex: Windows, entender o funcionamento do código fonte)
- Capital Intelectual Humano
- Informações de Marketing (Clientes, Posicionamento, Concorrentes)
- Direitos (Marcas, Registros, Patentes) -> Não p/ Direito Autoral (Intransferível)

Evolução das tecnologias do Software

https://www.draw.io/#G0B3TiTcAbl0LPcDd0blRVd0RMY2M

Aula 4

Processos
|
Métodos: *O&M*
| **Organização de Métodos**

Programas de Qualidade
|
ISO9000

Processos de Desenvolvimento de Software
|
Métodos
|
Ciclo de vida

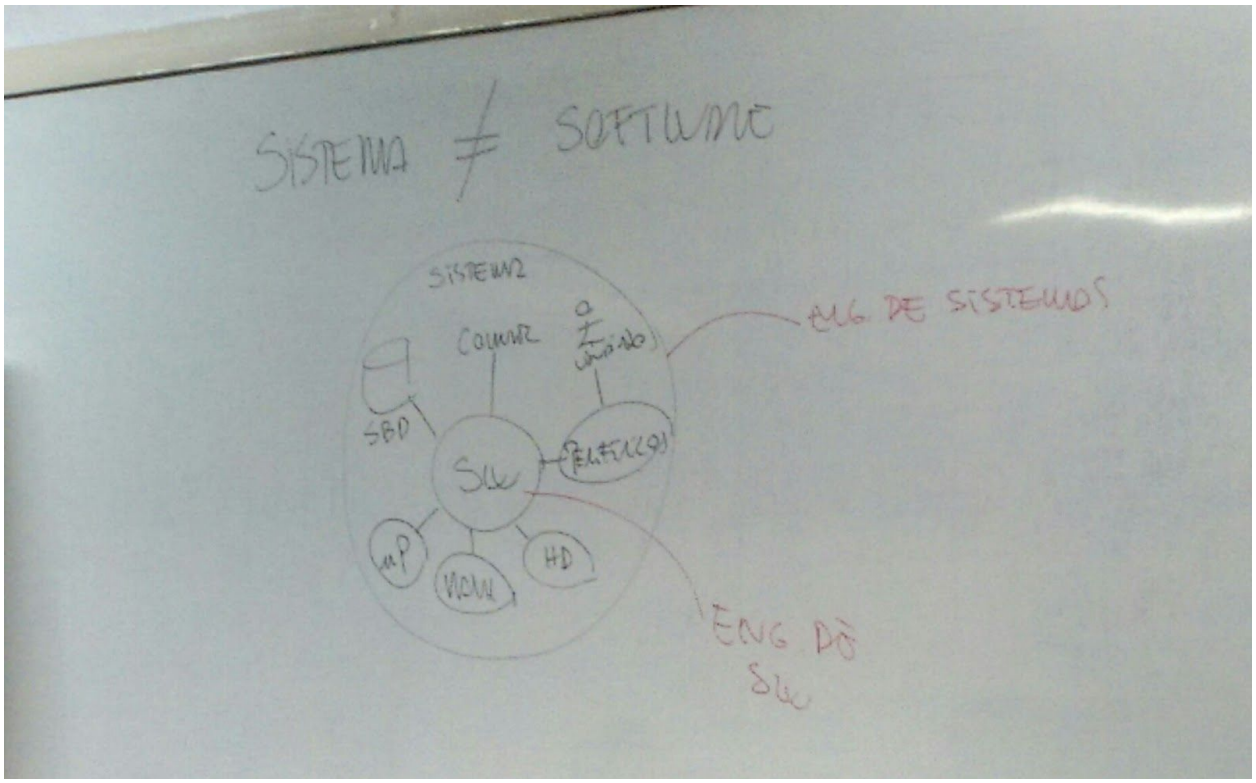
Ciclo Econômico de Software

Sistema -> Conjuntos de partes (componentes) que se relacionam (interagem) para atingir um objetivo comum que dá identidade ao conjunto.

TOP-Down -> Do mais abstrato ao mais técnico.

BOTTOM-Up -> é a “colcha de retalhos” do sistema para dar rumo a sistemas mais complexos. Tornando assim, a cada passo, os sistemas originais em subsistemas de um sistema final maior. Um processamento de baixo para cima é um tipo de processamento de informação baseado em dados de entrada vindos do meio ao qual o sistema pertence para formar uma percepção. Informação entra nos olhos e uma direção

(entrada), e é transformada em imagem pelo cérebro que pode interpretar e reconhecer como uma percepção (saída). Numa abordagem de baixo para cima os elementos básicos são inicialmente descritos em detalhes. Esses elementos são associados para formar um subsistema maior, que então pode ser associado a outros elementos em muitos outros níveis eventualmente até completar o nível mais alto do sistema objetivo. Essa estratégia se assemelha a um modelo de "semente", de forma a começar pequeno com elementos básicos e ir crescendo ao longo de completações e associações.



Incose -> 1990

BPM : Business Process Modeling
SSA = System Safety Analysis

IEC EN 50.128

|

Regra de software para área ferroviária

FIMEA - Failure Modes and Effects Analysis

Exemplo de FIMEA:

N°	Modo	Efeito	Classe
1	Falha no sensor de aproximação	Choque mecânico com a porta fechada	C
2	Falha na abertura da porta	Idem	C
3	Falha no sensor de passagem	Impressa o usuário	C

Pascal's Wager

System Requirements

|

Requisitos de Sistema

|

{O que o sistema deverá fazer

{ O que o sistema terá como propriedades

|

Especificação

{ - Textual (frases)

{ - Gráfica/Textual

System Architecture

|

Descrição estrutural/organizacional do sistema e de suas partes

Design

|

{ - Pensamento

{ - Modelagem

Arquitetura de SW (UML)

- Diagrama de Pacotes
- Diagrama de Componentes
- Diagrama de Classes

Arquitetura de Sistemas

- Diagrama de Blocos (Sysml)

System functional Modeling

- Modelos Funcionais
- Redes de Petri
- Máquina de Estados
- { - Diagrama de Atividades
- { - Diagrama de Transição de Estados

Padrões (Standard)

- Requirement Specification Standard
- Design Standard
- Coding Standard
- Notification Standard

Exemplos de Planos:

- PSAC - Plan For Software Aspects of Certification
- Software Requirements Specification Plan
- Software Development Plan (Processo de Desenvolvimento)

- Software Certification Plan (Testes, revisões, inspeções -> Qualidade Técnica)
- Software Quality Assurance Plan (Garantia de Processos)
- Software Configuration Management Plan
- Software Tools Qualification Plan

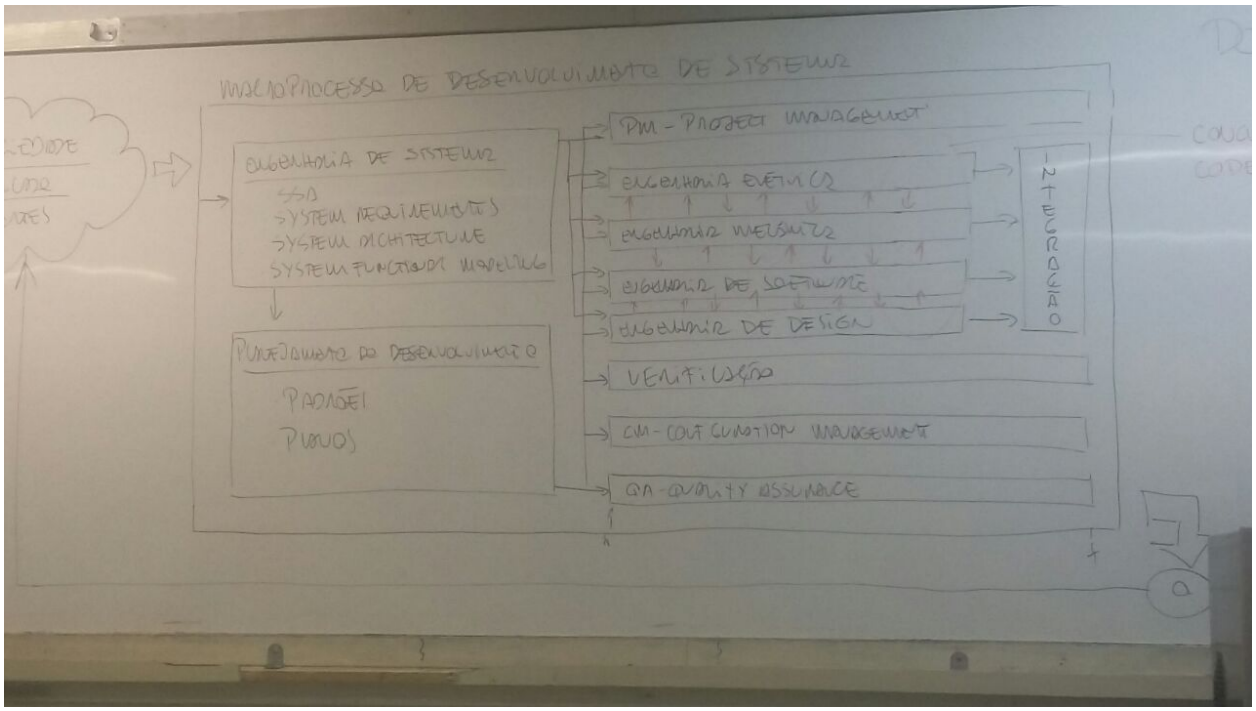
CM

- Version Control
- Branches
- Baselines
- Tasks
- Change Management

\/

Synergy

- IBM



Processo de Desenvolvimento de Software

| |

| Organização de Atividades para Desenvolvimento sistemático de um

| Software

|

Metodologia, metodo ou ciclo de vida

- Processo

- Antigos
- Atuais
- Novas (por vir)

- Waterfall
- Prototipação
- Modelo em V
- Refinamentos (Espiral)
- Processo Unificado (NUP)
- Métodos Ágeis (XP, Scrum, Lean)
- Harmony
- Model Driven
- Axiomatic Design

Atividades dos processos:

- Especificaçao de Requisitos
- Projeto (Design) - Inclui a modelagem
- Construção
- Verificação

Modelo Clássico (Waterfall ou Cascata)

Waterfall:

| |

| Processo sequencial

Aspectos positivos:

- Simples e intuitivo
- Fácil de gerenciar
- Possível conforto de trabalho

Aspectos Negativos

- Alto Custo/Esforço para retornos
- Alongamento de prazos
- Alto risco de falha dos requisitos

-> E -> P-> C -> V

Processos de desenvolvimento de Software

- Processo clássico
  - Waterfall
    - Sequencial
    - Fácil de entender e gerenciar
    - Alonga o desenvolvimento
      - Deixa muito distante a especificação da verificação.
      - Risco de não atender as necessidades do cliente.

Protótipos -> Prototipação

Objetivo: Permitir validar (parcialmente) os requisitos.

Regras:

- Investir o mínimo possível na construção.
- Jogar fora depois de usar.

Modelo ou processo em V “Cascatão”

Especificação de requisitos -> Projeto -> Construção -> Testes unitários -> Testes de integração -> Testes de aceitação -> Software

Mesmas vantagens e desvantagens do modelo em cascata. Pode-se usar prototipação

Caixa Preta = Sem código
Caixa Branca = Com código

Classes de Equivalência
Ex: int -> 1 byte = 1 bit sinal e 7 bits para dados
Int -127 …….0……..127

Casos de teste:
Soma(-5, -7)
Soma(-5, 0)
Soma (-5, +6)
Soma (0, 0)
Soma (0, 5)
Soma (5, 6)

- Processo por refinamentos sucessivos - espiral - (Gave e Sanson) - 1980

- Em lugar de revisar completamente cada atividade antes de passar para a próxima, avança-se um pequeno passo (incremento) de refinamento em todas elas simultaneamente gerando uma nova versão evoluída do software.
- Cada refinamento é atendido ciclo ou iteração
- Verificação frequente ajuda a garantir que o produto ao final estará próximo do “desejado” pelo cliente/mercado.

- Processo em espiral
    - Verificação frequente ajuda a garantir que o produto final estará próximo do “desejado” pelo cliente/mercado
    - Vantagens:
        - Reduz o risco e aumenta as chances de sucesso por meio de verificações frequentes.
        - Tendência de ser mais curto do que os processos sequenciais (20/30%)
        - Desenvolvimento em “time” (equipe)
            - Maior corresponsabilidade
                - Maior cooperação
                    - Sinergia
    - Desvantagens:
        - Pode gerar dificuldades na equipe heterogênea (trabalhar com pessoas de design).
        - Dificuldade de entendimento do processo e de sua gestão.

<u>Abordagens para o espiral</u>

- Abordagem em amplitude
    - Abordagem top-down
    - Iterações a cada nível do diagrama
    - Cada iteração é uma descrição completa do sistema

        - Versões melhoradas do sistemas a cada iteração
    - Abordagem em profundidade
        - Segunda iteração (tudo pela esquerda)
        - Sempre completa um ramo e não um nível
        - A folha é um item pronto
    - Abordagem mista
        - Em amplitude até um ponto e após profundidade.
        - Garante que o projeto não terá tanta falha

<u>Rational Unified Process (RUP)</u>

- 1997
    - Grady Booch - OOD (Object Oriented Design)
    - James Rumbaugh - OMT (Object Modeling Technique)
    - Ivar Jacobson - Objectory
- Os três criaram a Rational -> Comprada pela IBM em 2002
    - UML
    - RUP
    - Livros
    - Rose
    - RequisitePro

- TeleLogic -> Comprada pela IBM em 2009
    - Door
    - Rhapsody
    - Synergy

- RUP - Processo mais completo (rico) com relação ao SEBOK (Software Engineering Body of Knowledge).

- Elementos inovadores:
    - Conceito de Fases de Desenvolvimento
    - Curvas de Esforço

TABELA SHOW LÁ

Métodos Ágeis

- Processos iterativos e incrementais em profundidade
- Baseados no conceito de “Agilidade”
    - Valores definidos no “Manifesto Ágil”.
        - Indivíduo e interações mais que métodos e ferramentas
        - Software funcionando mais que documentação técnica
        - Colaboração com o cliente mais que contratos
        - Adaptação a mudanças mais do que planos.

Principais Métodos

- XP - Extreme Programming - Melhorar a qualidade e maior dinâmica
- Scrum
- Lean

Aula 31/10-

Episódio III - Especificação de requisitos de SW

- Definição precisa dos objetivos, propósitos ou do “o quê” pretende-se alcançar com um desenvolvimento de software;
- Literatura: 50% dos projetos falham em não especificar;
    - Feeling do professor: 70% ->SW. Para outros projetos -> 90%;
- Os requisitos são provenientes dos clientes/mercado;
    - Mas quem especifica é o Engenheiro/Analista;

Qual o Problema?

- O Cliente não sabe o que quer; ***<- NÃO É VERDADE***
- O Cliente não sabe dizer o que quer; ***<- NORMAL***
- Os requisitos mudam com muita frequência; ***<- NÃO É VERDADE***

- O engenheiro/analista pode definir os requisitos baseado na sua experiência: ***<u><- NÃO</u>***
    - Jim Turley -> Designing Crappy Products;
- Especificação de requisitos superficial, incompleta ou inconsistente; ***<u><- NÃO É VERDADE PARA NÓS</u>***

<u>Ideias para enfrentar os problemas:</u>

1. GET INVOLVED;
2. Faça uso de técnicas e ferramentas:
    a. IEEE850;
    b. Doors;
    c. RequisitePro;
3. Entender/mapear as razões para os requisitos;

<u>Definição de requisitos</u>

- Um requisito é uma exigência (obrigação) imposta pelo cliente sobre o software a ser desenvolvido.

<u>Tipos (ou categorias) de requisitos</u>

- Requisitos funcionais
    - Requisito que descreve o que o software deverá fazer (entregar)
    - Exemplos:
    -
- Requisitos não-funcionais
    - Descreve característica, propriedades ou restrições.
    - Exemplo:
    - Ser amarelo;
    - Ser escrito em .NET;
    - Ter botões arredondados.
    - Classes:
    - Econômicos: custo, preço, prazo;
    - Desempenho: taxa, tempo, rendimento
    - Tempo real

        - RTOS
            - VxWork - Wind River
            - Linux embedded
            - Windows embedded
            - Android
            - QNX
            - X-Kernel - UTFPR
    - Ambiente
        - Biblioteca
        - Ferramentas
        - Linguagens de modelagem
    - Estéticos
        - Cor
        - Linguagens de design
    - Estima
        - Marca
        - Referências

<u>Formas de especificação</u>

- Textual (criar uma sentença em linguagem natural para cada requisito).
    - Ex: O software deve gerar um alarme quando o sensor detectar uma temperatura maior do que 60 graus Celsius.
    - IEEE 850
- Modelos
    - Diagrama de requisitos (SYSML)
    - NOPSRS - UTFPR
- Métodos formais

<u>Hierarquização de Requisitos</u>

- RF001 - O SW deverá imprimir um relatório de vendas

    - RF001.1 - O SW deverá permitir selecionar o período para o relatório de vendas
        - RF001.1.1 - O SW deverá apresentar uma tela de configuração de datas para o relatório
        - RF001.1.2 - O SW deverá checar a consistência das datas indicadas
    - RF001.2 - O SW deverá carregar os dados de vendas da base de dados
        - RF001.2.1 - O SW deverá estabelecer conexão segura com o banco de dados.
        - RF001.2.2 - O SW deverá realizar uma Query de busca pelos dados de acordo com as datas do relatório.
    - RF001.3 - O SW deverá imprimir o relatório com os dados das vendas

High level requirements: raíz
Intermediate level requirements: meio
Low level requirements: folhas

IMAGEM ^^ - by alves

<u>Nomenclatura para requisitos</u>

- Usar um código: RF e RNF (FR E NFR) Numeração e uma sentença com uma oração em linguagem natural
- Requisitos Funcionais
    - Sujeito - O software
    - Verbo
    - Complemento

Elaborar a especificação de requisitos para um software editor de textos simples (tipo notepad)

- O software deverá mostrar (ação) condição

- O software deverá ser/ter (propriedade) condição

Exercício:

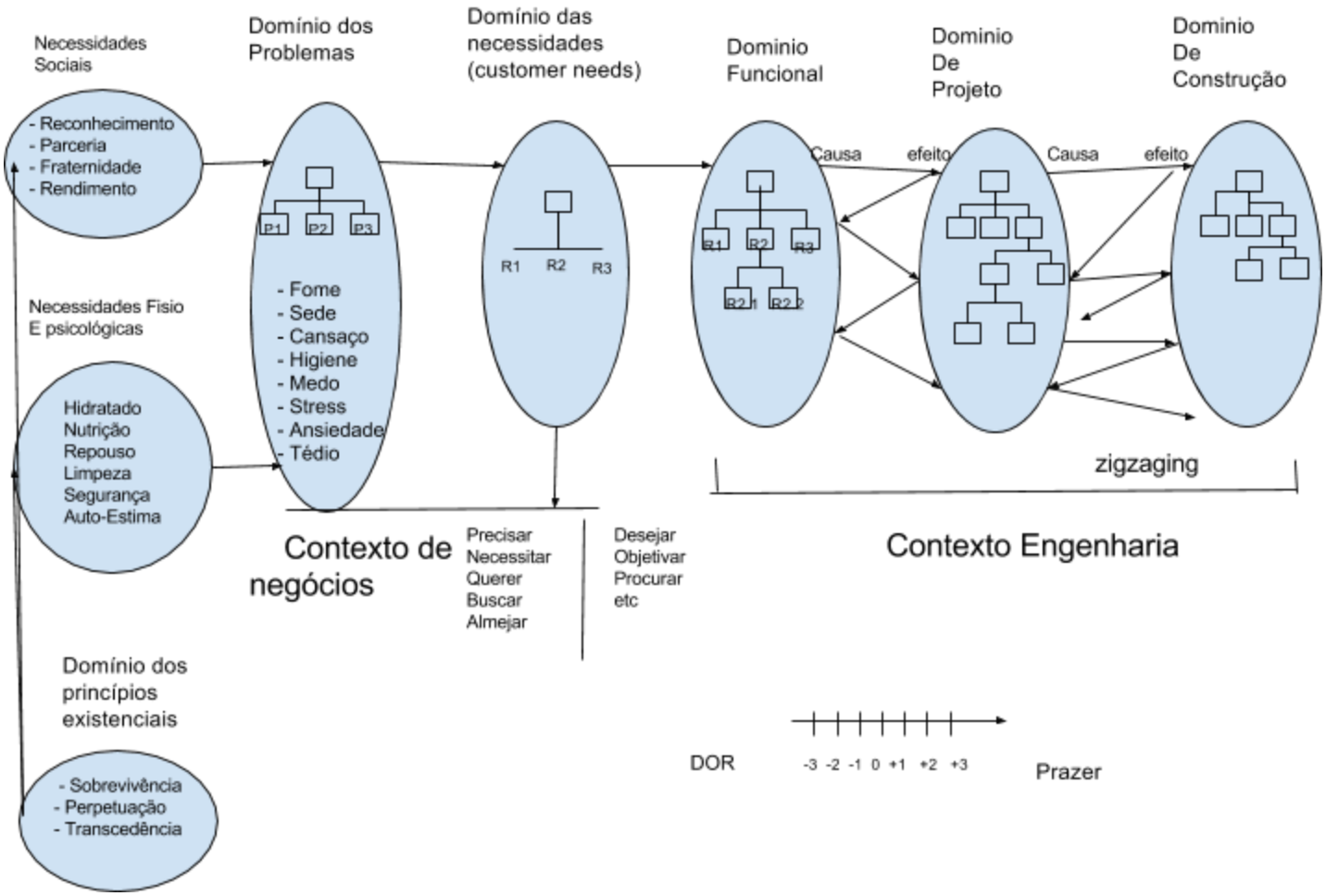
Requisitos Funcionais:

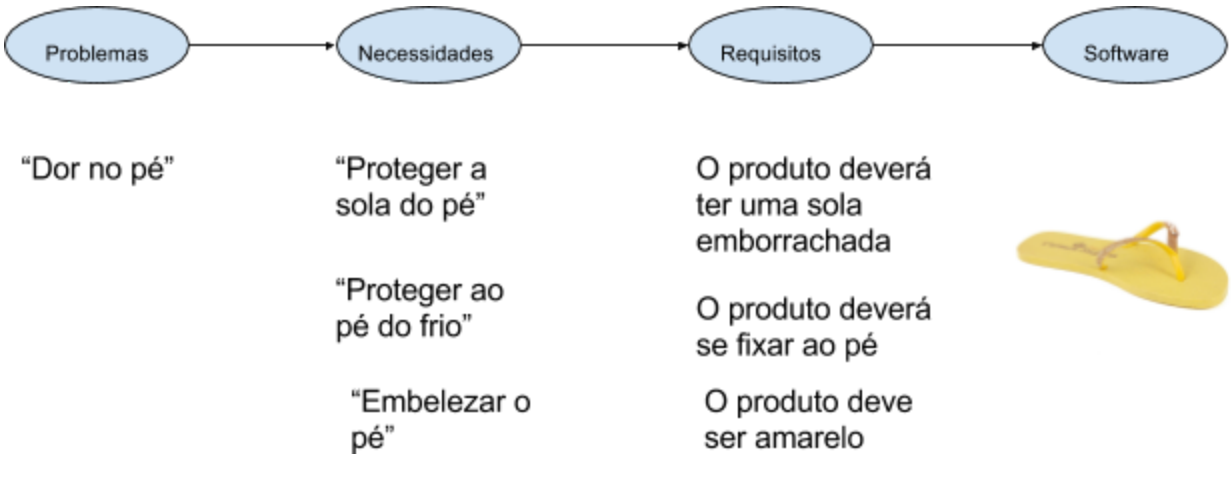
RF01 - O Software deverá ler caracteres do teclado
RF02 - O software deverá mostrar ~~(ecoar)~~ cada caractere lido do teclado na posição atual do cursor no monitor
RF03 - O software deverá carregar em memória um texto de um arquivo armazenado em um sistema de arquivos e indicado pelo usuário.
RF04 - O software deverá permitir ao usuário gravar o texto editado em um arquivo em um sistema de arquivos.
RF05 - O software deverá permitir o usuário apagar o caractere à esquerda do cursor de edição quando for pressionada a tecla Backspace.
RF06 - O software deverá permitir ao usuário alterar a posição do cursor de edição para aquela apontada pelo cursor do mouse quando for clicado seu botão esquerdo.
RF07 - O software deverá permitir ao usuário mover o cursor de edição com as teclas direcionais.
RF08 - O software deverá permitir ao usuário copiar um texto marcado.
RF08.1 - O software deverá permitir ao usuário marcar um texto.
RF08.1.1 - O software deverá permitir ao usuário iniciar a marcação de um texto pressionando a tecla Shift.
RF08.1.2 - O software deverá permitir ao usuário estender a indicação de um texto com as setas direcionais.

RF08.2 - O software deverá permitir ao usuário comandar a cópia do texto marcado.
RF08.3 - O software deverá permitir inserir uma cópia do texto indicado na posição atual do cursor de edição.

Requisitos Não Funcionais:

RNF01 - O software deverá ter como padrão a codificação Unicode.
RNF02 - O software deverá ter interfaces com teclado, mouse, monitor e sistema de arquivos.
RNF03 - O software deverá ser desenvolvido em até 12 meses.
RFN04 - O software deverá funcionar em Windows 10 e Linux Manjaro 3.1
RFN05 - O software deverá ser programado em C++.
RFN06 - O software deverá ter cor de fundo da tela em verde ou vermelho.





Necessidades do Cliente (Customer’s Needs)

- Codigo: N01, N01.1, etc..
- Sentença:

SUJEITO +	VERBO DE FALTA +	FONTE +	OBJETO (OUTCOME)
{ O CLIENTE }	{ PRECISA }	{ SOFTWARE }	{ INFORMAÇÃO }
{ O DIRETOR }	{ NECESSITA }	{ SISTEMA DE INF }	{ ENTRETENIMENTO}
{ A EMPRESA }	{ QUER }	{ SISTEMA DE COMP }	{ CONSTRUIR / CRIAR}
{ O MERCADO}	{ DESEJA }	{ ERP }	{ CONTROLE }
	{ PROCURA }		
	{ BUSCA }		
	{ OBJETO }		

O diretor de vendas precisa de um software para saber o total mensal das vendas.
O diretor de RH deseja um software para controlar o horário de entrada e saída dos funcionários.
O assistente de recepção necessita um software para entreter os visitantes enquanto aguardam.
O médico busca um software para criar as receitas de medicamentos durante uma consulta.

EXERCÍCIO:

1) Determine uma ou mais necessidades para o software a partir dos seguintes requisitos:
    a) RF01: O software deverá autenticar o usuário
        i) O cliente necessita de um software para controlar o acesso do usuário.
        ii) O cliente deseja um software para saber quais usuários usaram o sistema.
        iii) A empresa necessita de um software para controlar o acesso ao sistema.
    b) RF02 - O Facebook deverá permitir postar uma foto
        i) O usuário precisa de um software para criar posts com imagens.
    c) RF03 - O software deverá permitir ao usuário cadastrar um novo livro
        i) O usuário necessita de um software para criar cadastro com informações sobre um livro.
        ii) O mercado bibliotecário precisa de um software para saber quais livros possui.

Problemas do Cliente (Customer's Problems)

- Código: P01, P01.1
- Sentença:

SUJEITO +	VERBO DE INTENSIDADE +	OBJETO +	PENALIZAÇÃO
{ O CLIENTE }	{ DEVE }	{ VERBO +	{ PREJUÍZO }

{ O DIRETOR }	{ SOFRE A	COMPLEMENTO }	{ DOR }
{ A EMPRESA }	EXPECTATIVA}		{ SOFRIMENTO}
{ O MERCADO}	{ SOFRE A		{ ONUS }
	ESPERANÇA }		

Classes de Intensidade

- Obrigações (Deveres)
    - Coisas que devem ser feitas ou exercidas sob pena de prejuízos, sofrimentos, etc.
- Expectativas
    - Coisas não-obrigatórias mas que gera penalizações
- Esperanças
    - Coisas que não se aguarda, mas que poderia ocorrer

P1 - O diretor de rh deve pagar os funcionários até o último dia útil do mês, sob pena de multa sindical e trabalhista.
P2 - A escola deve garantir que todos os alunos saíram antes de fechar, sob pena de negligência na guarda da crianças.
P3 - O diretor de marketing é obrigado a contatar seus clientes semestralmente, sob pena de não cumprir a política de relacionamento da empresa.

- Um diretor de estoque tem preocupações com relação às suas responsabilidades. Determine dois problemas que ele possa ter.
    - O diretor de estoque deve garantir que não falte peças para utilização, sob pena de perder parte da lucratividade.
    - O diretor de estoque deve garantir a integridade peças, sob pena de ressarcir o valor dela.
    - O diretor de estoque deve garantir que não repasse informações erradas sobre a quantidade do estoque, sob pena de perder vendas.
    - Professor:
    - O diretor de estoque deve garantir as reservas mínimas dos produtos, sob pena de incapacidade de entrega.
    - O diretor de estoque deve assegurar que os itens de estoque não sejam danificados, sob pena de perda financeira e incapacidade de entrega.