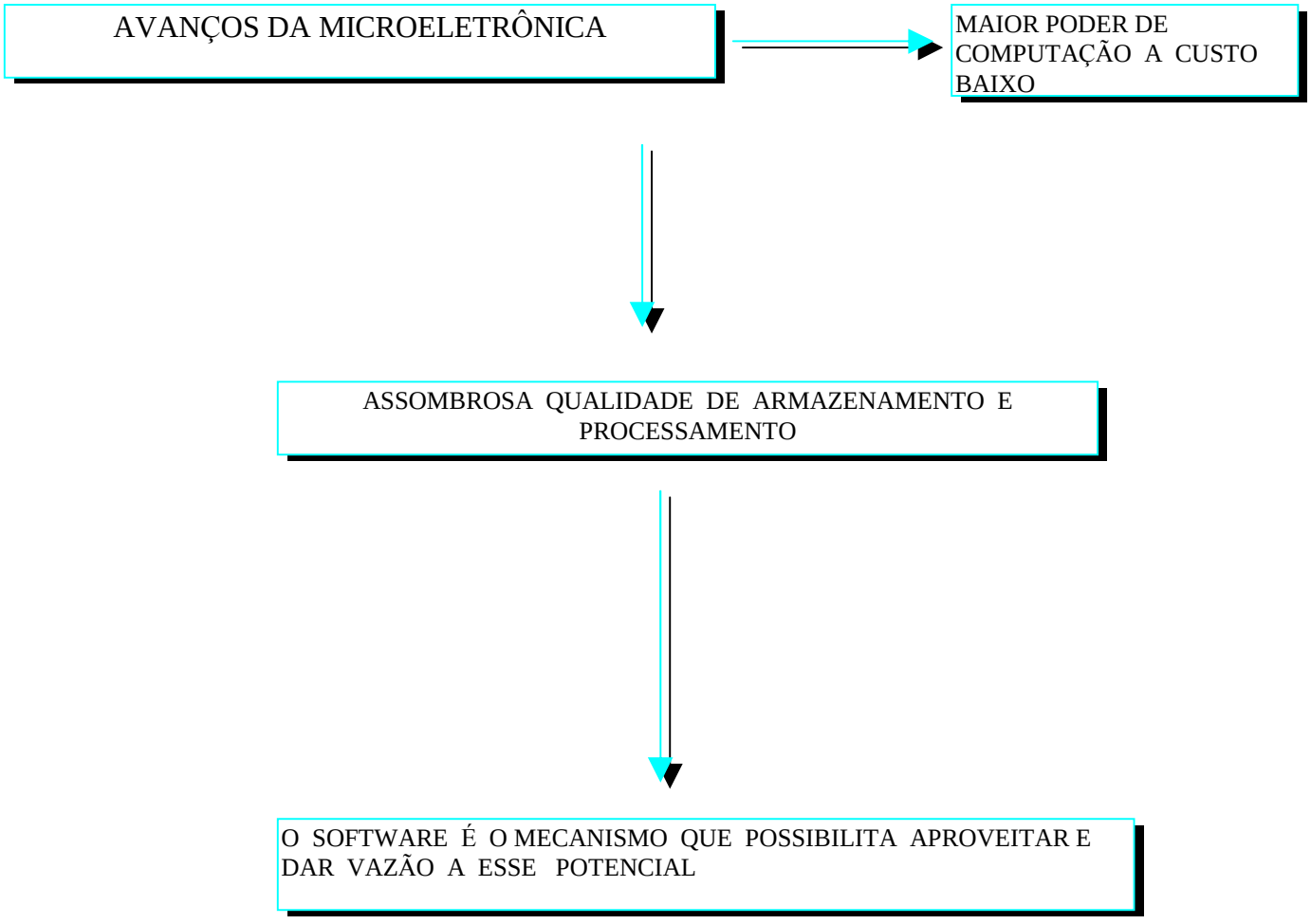
UNIVERSIDADE VEIGA DE ALMEIDA
AVALIAÇÃO DE SISTEMAS
PROFESSORA ROSA AMELITA SÁ MENEZES DA MOTTA

EMENTA

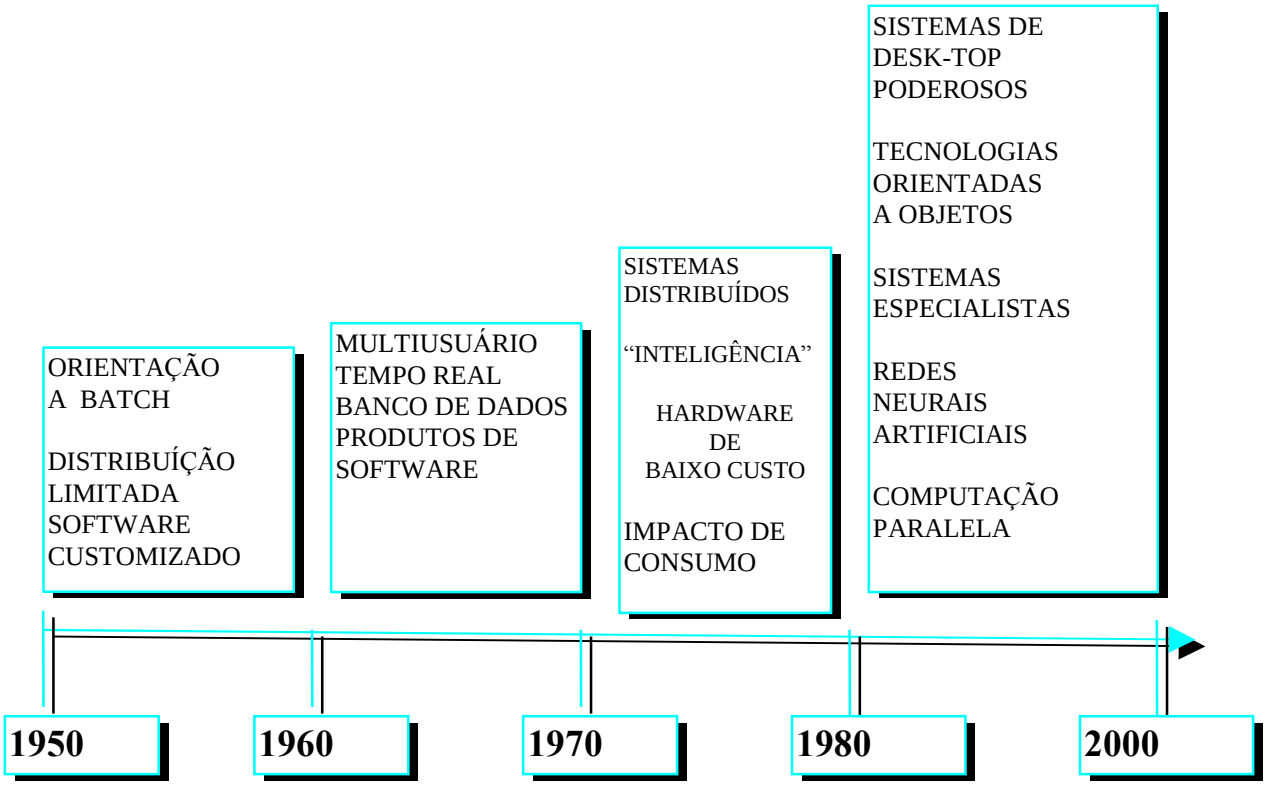
O DESENVOLVIMENTO DE SISTEMAS
A ORIENTAÇÃO A OBJETOS E O DESENVOLVIMENTO DE SISTEMAS
ADMINISTRAÇÃO DE PROJETOS: MÉTRICAS E ESTIMATIVAS
QUALIDADE DE SISTEMAS
AMBIENTES DE PROGRAMAÇÃO E A QUALIDADE DE SISTEMAS
NOVAS PERSPECTIVAS EM ENGENHARIA DE SOFTWARE

O DESENVOLVIMENTO DE SISTEMAS

A IMPORTÂNCIA DO SOFTWARE



EVOLUÇÃO

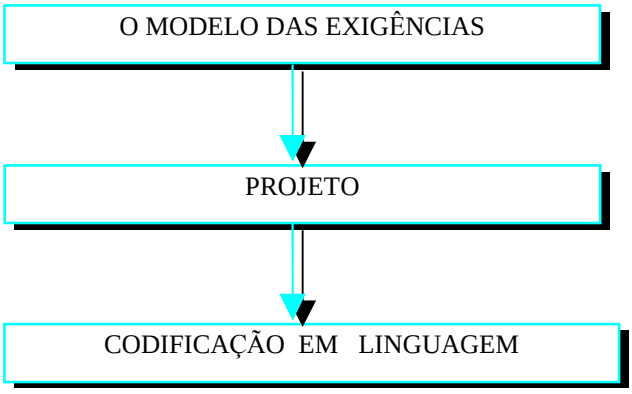

ORIENTAÇÃO
A BATCH
DISTRIBUÍÇÃO
LIMITADA
SOFTWARE
CUSTOMIZADO
MULTIUSUÁRIO
TEMPO REAL
BANCO DE DADOS
PRODUTOS DE
SOFTWARE
SISTEMAS
DISTRIBUÍDOS
"INTELIGÊNCIA"
HARDWARE
DE
BAIXO CUSTO
IMPACTO DE
CONSUMO
SISTEMAS DE
DESK-TOP
PODEROSOS
TECNOLOGIAS
ORIENTADAS
A OBJETOS
SISTEMAS
ESPECIALISTAS
REDES
NEURAIS
ARTIFICIAIS
COMPUTAÇÃO
PARALELA
1950
1960
1970
1980
2000

CARACTERÍSTICAS E COMPONENTES

- O SOFTWARE É DESENVOLVIDO OU PROJETADO POR ENGENHARIA, NÃO MANUFATURADO NO SENTIDO CLÁSSICO (NÃO É PROCESSO MECÂNICO)
- O SOFTWARE NÃO SE DESGASTA
- A MAIORIA DOS SOFTWARES É FEITO SOB MEDIDA EM VEZ DE SER MONTADO DE COMPONENTES EXISTENTES

COMPONENTES DO SOFTWARE

SÃO CRIADOS POR MEIO DE UMA SÉRIE DE CONVERSÕES QUE MAPEIAM AS EXIGÊNCIAS DO CLIENTE PARA CÓDICO EXECUTÁVEL EM MÁQUINA



AS LINGUAGENS EM USO SÃO:

- LINGUAGENS DE MÁQUINA :

- LINGUAGENS DE ALTO NÍVEL: Pascal, C, ADA, C++, Object Pascal, Eiffel, LISP, PROLOG, etc...
- E NÃO PROCEDIMENTAIS: Linguagens de Banco de Dados

EXIGÊNCIA: REUSABILIDADE

PROBLEMAS E CAUSAS

AS ESTIMATIVAS DE PRAZO E DE CUSTO SÃO FREQUENTEMENTE IMPRECISAS

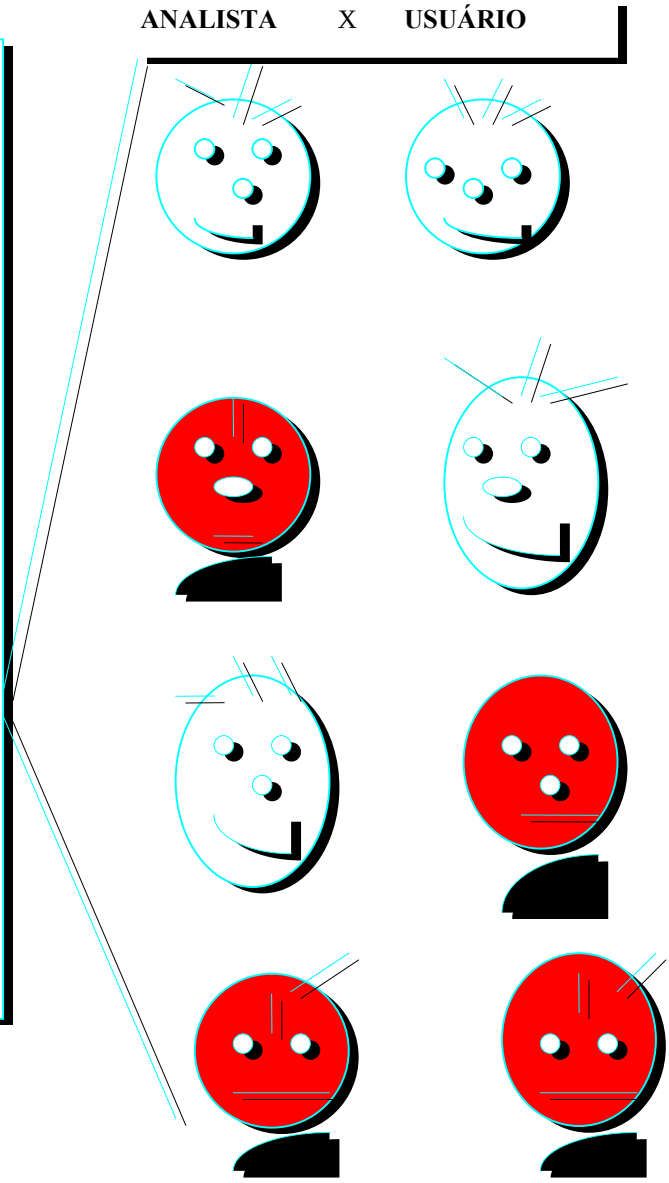
A PRODUTIVIDADE DAS PESSOAS DA ÁREA DE SOFTWARE NÃO TEM ACOMPANHADO A DEMANDA POR SERVIÇOS

A QUALIDADE DE SOFTWARE É MENOS QUE A ADEQUADA

NÃO SE DEDICA TEMPO À COLETA DE DADOS (ERRA-SE NO PLANEJAMENTO)

INSATISFAÇÃO DO CLIENTE COM O SISTEMA PRONTO (COMUNICAÇÃO ENTRE O CLIENTE E O DESENVOLVEDOR É FRACA)

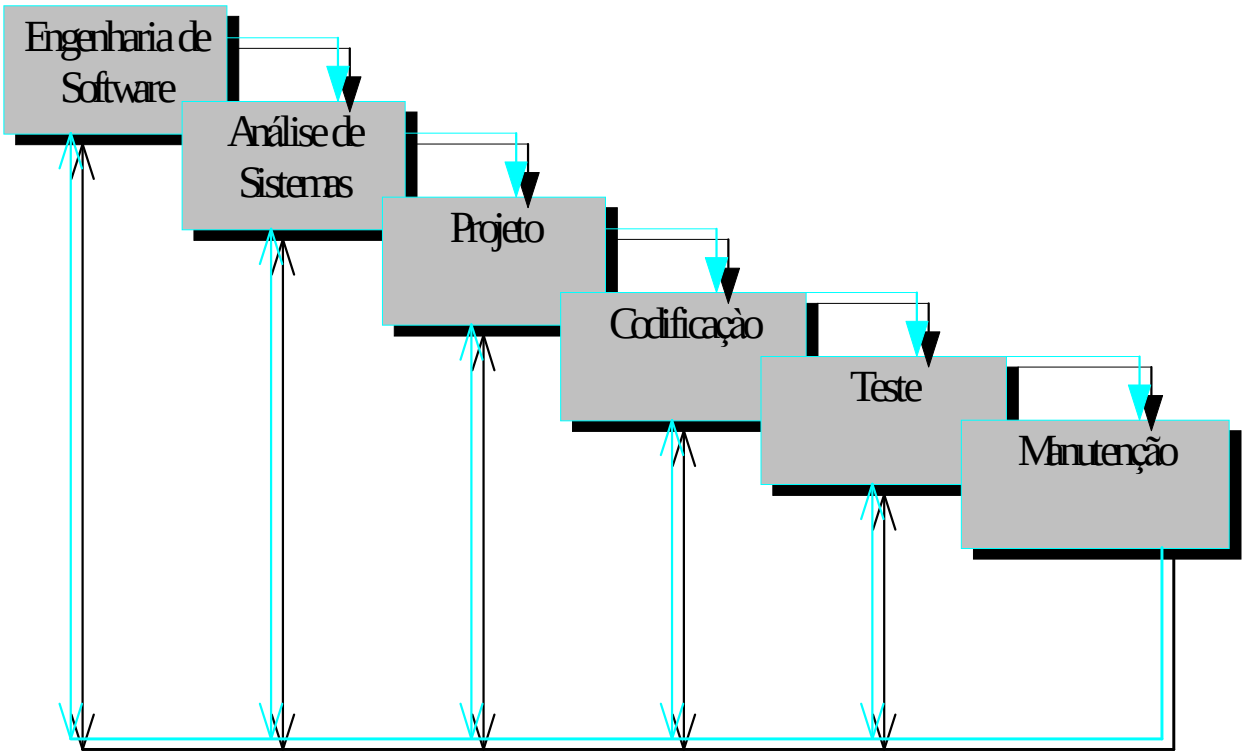
DIFICULDADE EM MANTER O SOFTWARE EXISTENTE



CAUSAS

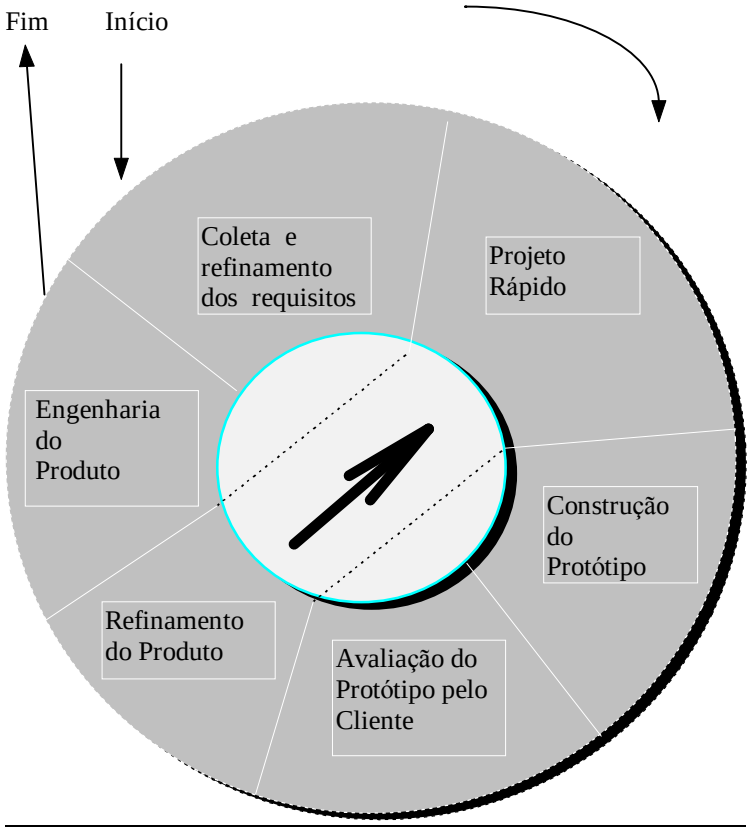
"EXPERIÊNCIA POUCO MAIS DE 40 ANOS"
"PROFISSIONAIS DE INFORMÁTICA COM POUCO TREINAMENTO FORMAL EM TÉCNICAS PARA DESENVOLVIMENTO DE SOFTWARES"
"GERENTES SEM BACKGROUND"
"TUDO DEVE SER FEITO PARA ONTEM"

CICLO CLÁSSICO DA ENGENHARIA DE SOFTWARE

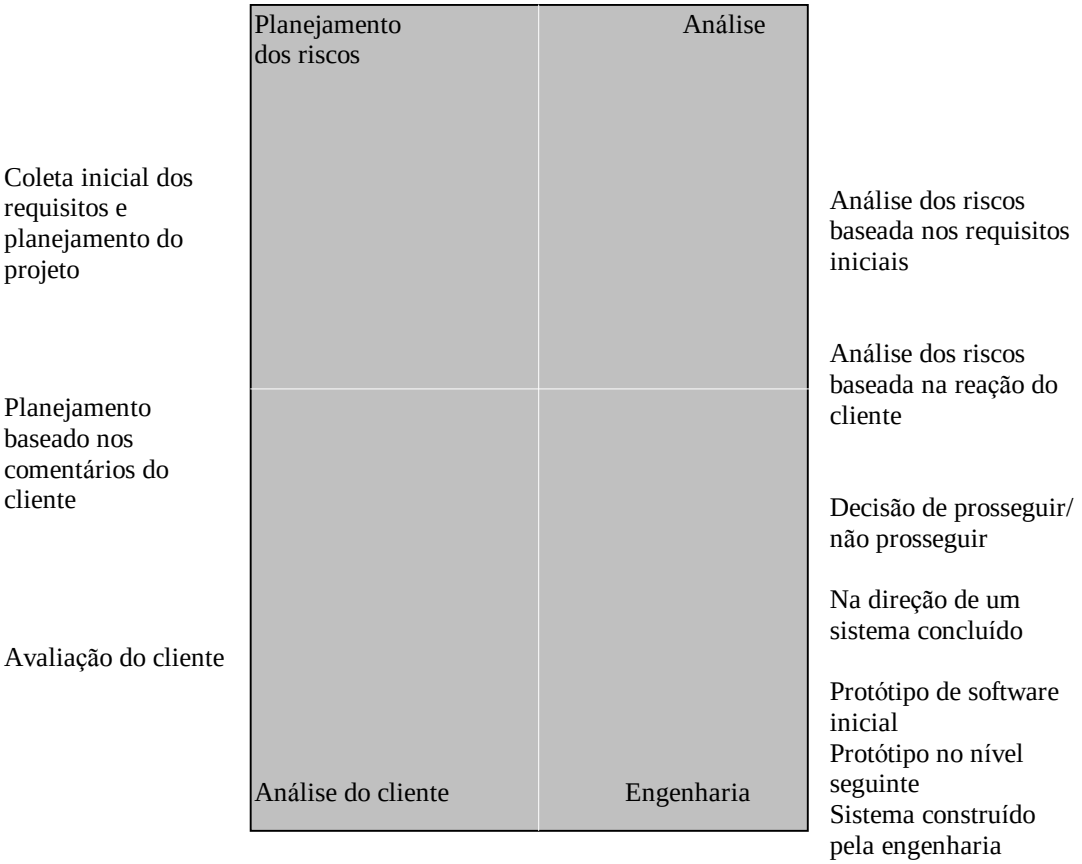


PROTOTIPAÇÃO

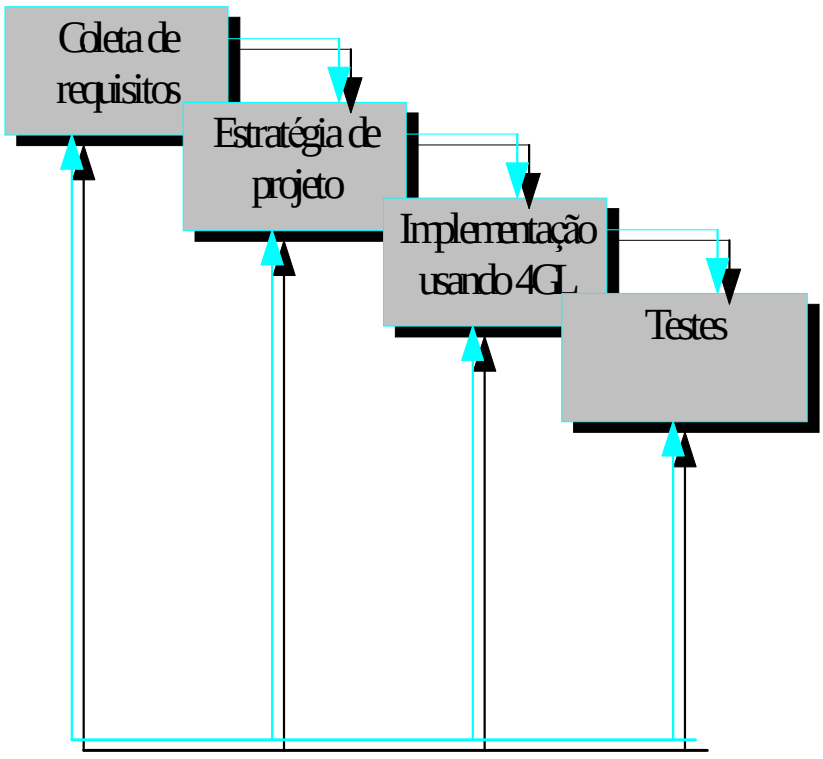
PROCESSO QUE CAPACITA O DESENVOLVEDOR A CRIAR UM MODELO DO SOFTWARE QUE SERÁ IMPLEMENTADO.



O MODELO ESPIRAL



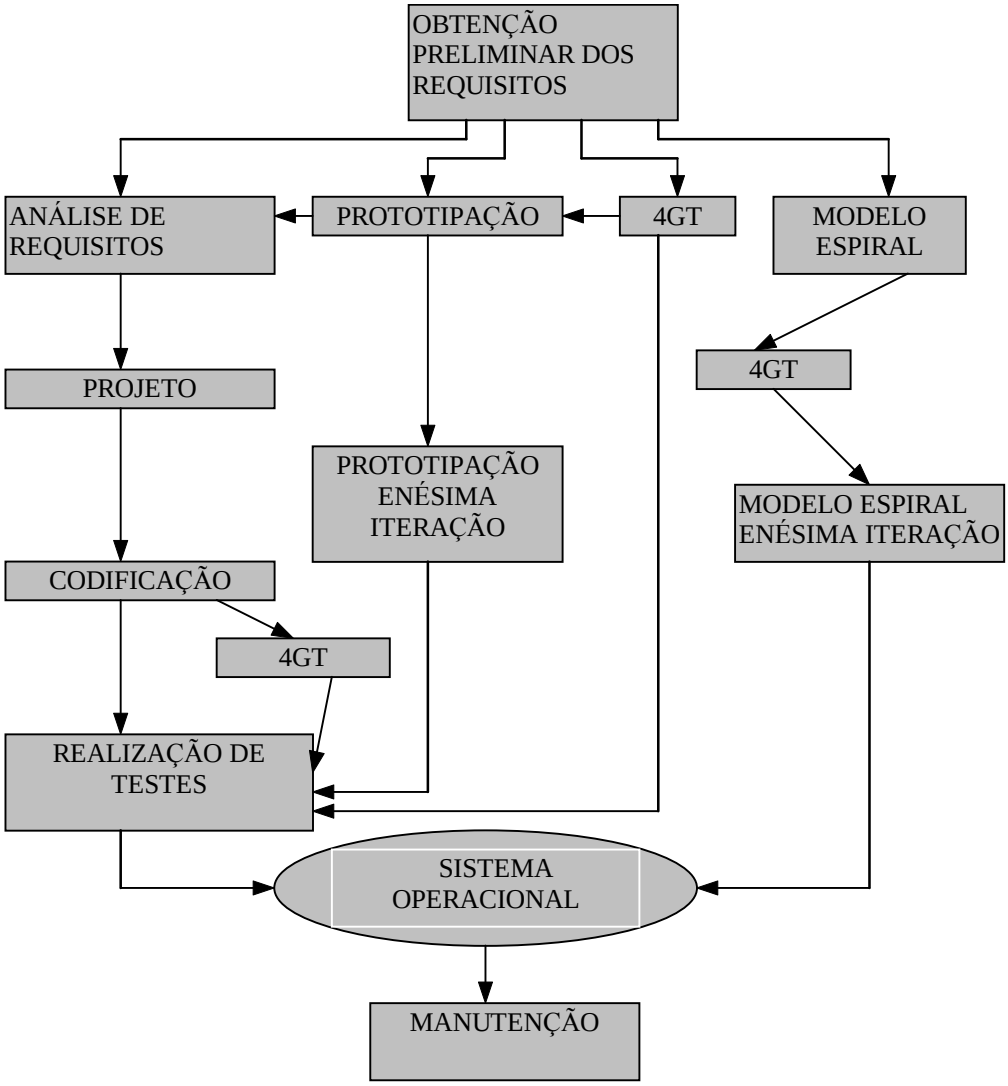
TÉCNICAS DE QUARTA GERAÇÃO



As novas ferramentas CASE agora suportam o uso das 4GT.

Geração de Código Automática

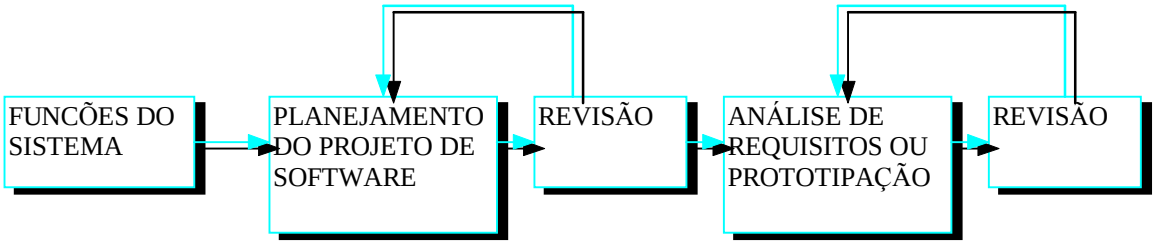
COMBINANDO PARADIGMAS



FASES DO DESENVOLVIMENTO DE SISTEMAS

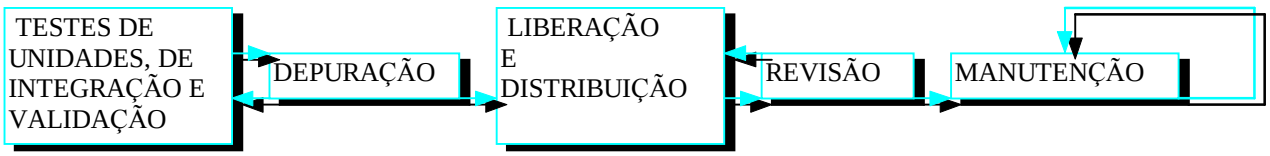
por Roger S. Pressman

Fase de Definição=> planejamento do sistema: descrição do escopo, análise do esforço, análise de riscos, levantamento dos recursos exigidos, estimativas de custos e de prazos. O objetivo é fornecer uma indicação da viabilidade do software; fase de análise e requisitos do software: a análise forma do domínio da informaçào é utilizada para estabelecer modelos de fluxo de dados e da estrutura da informação. Alternativamente pode ser feito um protótipo. Estes modelos são detalhados para se tornar uma **especificação do software**, que é o **documento** produzido com resultado desta fase.



Fase de Desenvolvimento=>descrição de estrutura modular, definição de interfaces, uma estrutura de dados é estabelecida. Uma especificação de projeto é produzida. E a codificação é realizada.

Fase de Verificação, Liberação e Manutenção=>realização de testes para descobrir o máximo de erros. Faz-se a manutenção do software ao longo da sua vida útil.



AS ETAPAS DO DESENVOLVIMENTO DE SISTEMAS

(continuação) por S. Pompilho

Análise de Sistemas=>determinação de quais os requisitos do sistema. O **que** o sistema deve fazer.

Projeto de Sistemas=>determinação de **como** o sistema funcionará para atender aos requisitos especificados na fase de análise.

Implementação de Sistemas=>construção efetiva do sistema.

Metodologias de Desenvolvimento maneira de se utilizar um conjunto coerente e coordenado de métodos para atingir um objetivo. Em outras palavras, a metodologia deve definir quais as fases de trabalho previstas no desenvolvimento de sistemas.

Método é um procedimento a ser adotado para se atingir um objetivo.

Técnica é um modo apropriado de se investigar sistematicamente um determinado universo de interesse ou domínio de um problema. Ex: análise estruturada, análise essencial e projeto estruturado.

Ferramentas

Diagrama de Fluxo de Dados
Diagrama Entidade Relacionamento
Diagrama de Transição de Estados

PRODUÇÃO DE MODELOS. **EXEMPLOS**: MODELO FUNCIONAL, MODELO CONCEITUAL DE

Notação é um conjunto de caracteres, símbolos e sinais formando um sistema convencionado de representação.

METODOLOGIA DEVE ESTABELECER QUAIS OS PONTOS DE CONTROLE E PADRÕES DE QUALIDADE

AS ETAPAS DO DESENVOLVIMENTO DE SISTEMAS
(continuação)

TÉCNICAS	**ABORDAGENS**	**FERRAMENTAS**
ANÁLISE TRADICIONAL	FUNCIONAL	TEXTOS E FLUXOGRAMAS
ANÁLISE ESTRUTURADA	FUNCIONAL DADOS	DIAGRAMA DE FLUXO DE DADOS DIAGRAMA DE ESTRUTURA DE DADOS NORMALIZAÇÃO DICIONÁRIO DE DADOS
ANALISE ESSENCIAL	FUNCIONAL DADOS CONTROLE	TABELA DIAGRAMA E FLUXO DE DADOS DIAGRAMA DE ENTIDADE-RELACIONAMENTO DIAGRAMA DE TRANSIÇÃO DE ESTADOS

O DESENVOLVIMENTO DE SISTEMAS

Modelagem de Dados Conceitual e seus Elementos

Entidade

É uma representação de um objeto do mundo real que tem muita importância para a vida do sistema e que independe da existência de quaisquer outros elementos. Segundo Setzer, uma entidade pode ser a representação de um ser, de um fato, de uma coisa, etc...

EXEMPLO:

1- Considere a informação a seguir para um determinado sistema de controle acadêmico com o objetivo de gerar um relatório contendo para cada alunos disciplinas que este faz.

"Maurício cursa Pascal."

2- Considere as informações a seguir para um determinado sistema de controle da venda de produtos e uma loja com o objetivo de gerar um relatório contendo para cada cliente os produtos comprados.

"O **cliente** Luiz comprou 2 quilos do **produto** açúcar."
"A **cliente** Marieta comprou um quilo de farinha de trigo."

Relacionamento

Segundo Setzer, é uma estrutura abstrata que indica a associação entre elementos de duas ou mais entidades. Um relacionamento binário é um par ordenado (e1, e2), onde *e1* e *e2* são respectivamente os elementos de *E1* e *E2* (que são só conjuntos de entidades envolvidas).

Um relacionamento é dependente das entidades, as quais associa.

Entidade-tipo

É um conjunto de entidades da **mesma natureza** ou **características**. Por exemplo, alunos pode ser o nome dado ao conjunto de entidades do tipo aluno.

Relacionamento-tipo

É um conjunto de relacionamentos da **mesma natureza**. Isto é, um conjunto de relacionamentos que tenham o mesmo significado semântico. Cursam pode ser o nome dado ao conjunto dos relacionamentos do mesmo tipo cursa.

Atributo

É uma característica (ou propriedade) de uma entidade ou relacionamento. Exs: nome do aluno, endereço de um cliete, etc. Os atributos são escolhidos de acordo com os objetivos de cada sistema.

Descrição do Mini-mundo

Deseja-se construir um sistema de ganhos dos proprietários de lojas de um novo shopping.

Cada loja, identificada pelo número de box e razão social, possui um ou mais proprietários, identificados por nome e telefone. Cada proprietário recebe os lucros mensais em função da participação em cada uma de suas lojas.

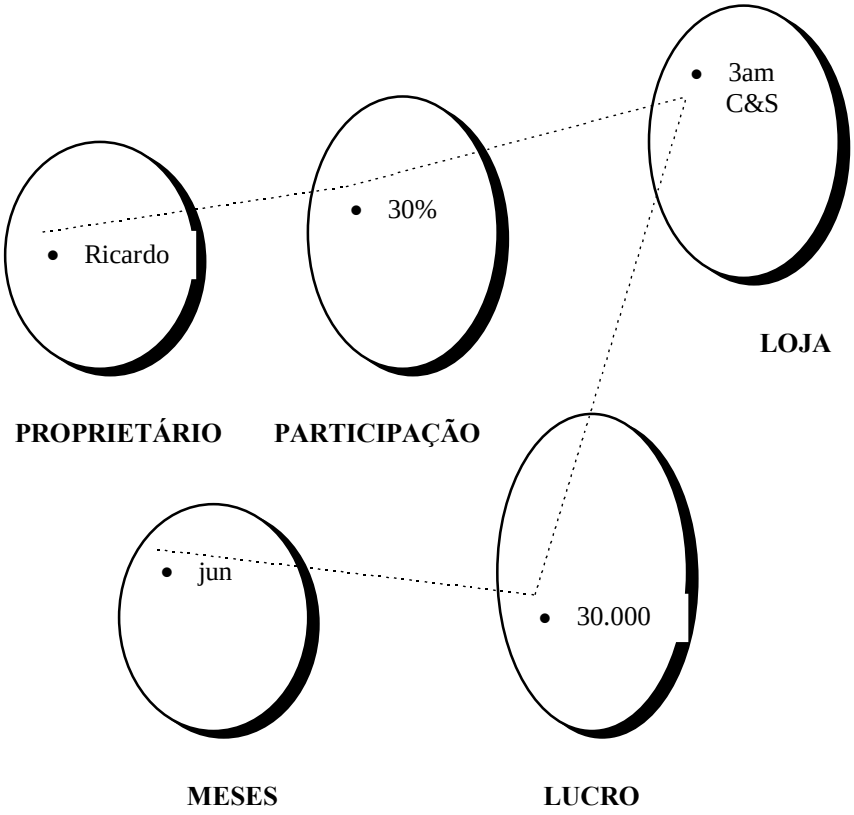
Deseja-se construir um sistema que:

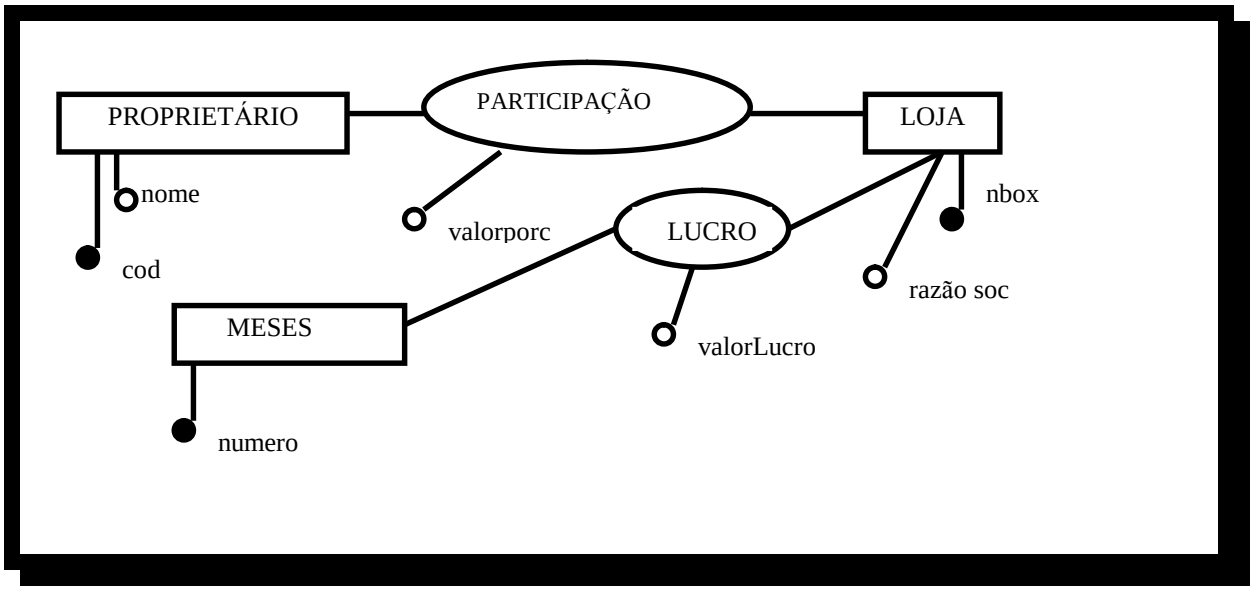
registre a participação de cada proprietário em suas lojas
e o lucro mensal recebido de cada loja, com o objetivo de saber o fator de crescimento de um determinado mês em relação ao anterior ou o total recebido no ano corrente.

" A loja de box 3am e razão social C&S obteve um lucro de 30 mil dólares no mês de junho."

Participação (proprietário, valorporc, loja)

Lucro (loja, valorlucro, meses)





Notação do MER pelo Peter Chen

Restrições de Integridade

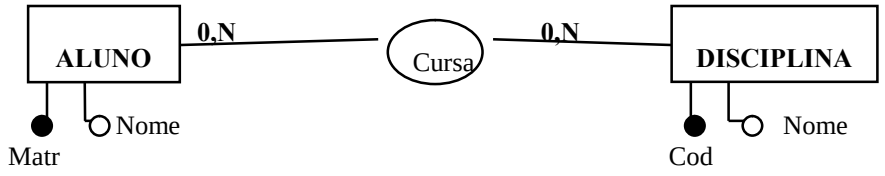
O **modelo conceitual de dados** deve conter as **restrições**, isto é, as normas ou leis que regem a realidade em estudo.

Essas restrições preservam a integridade desta realidade ao se automatizar a solução para o problema.

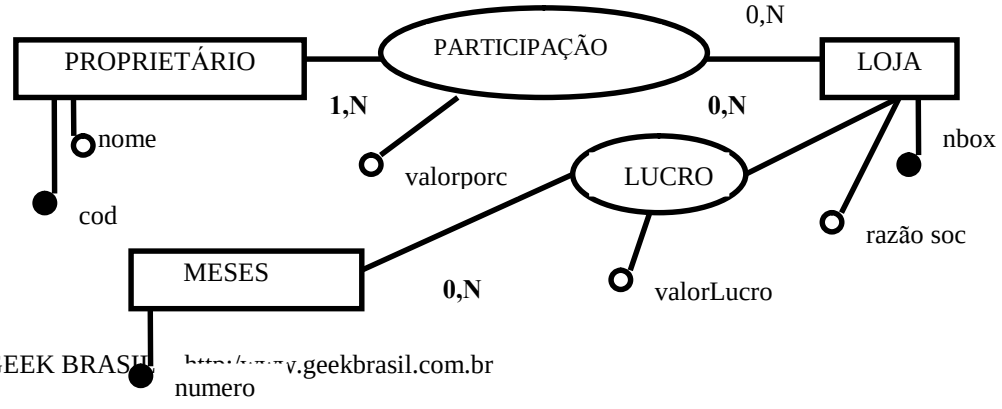
Por exemplo, uma realidade acadêmica, pode sofrer as seguintes restrições: um professor de um determinado curso só pode lecionar, no máximo, três disciplinas; um professor só pode estar lecionando disciplinas para as quais esteja habilitado; um aluno pode estar matriculado em nenhuma disciplina e, no máximo, em N=7.

Cardinalidade

Cardinalidade de uma entidade-tipo X é o número mínimo e máximo de vezes que uma entidade "e" da entidade-tipo X pode estar associado a um relacionamento-tipo R.



O MODELO DE DADOS CONCEITUAL COM CARDINALIDADES



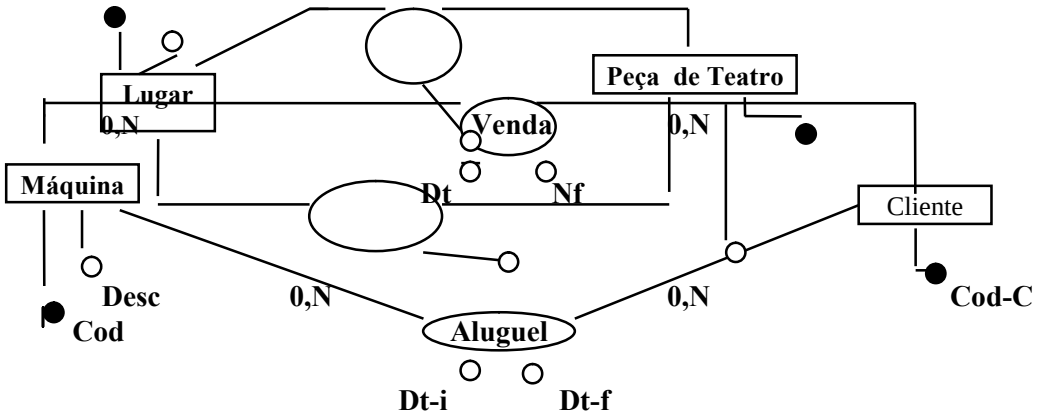
RI ER-Restriçã o de Integridade por Existência de Relacionamento

RI NER-Restriçã o de Integridade por não Existência de um Relacionamento

Cod O,N O,N
Desc Venda

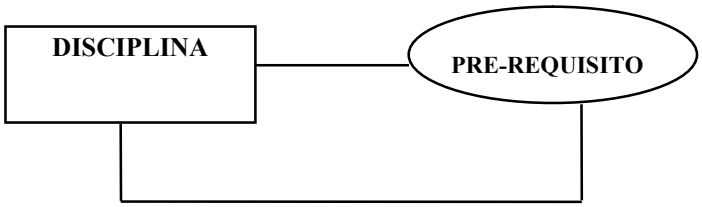
Data-Venda Cod

O,N O,N
Reserva
Data-Reserva. Nome



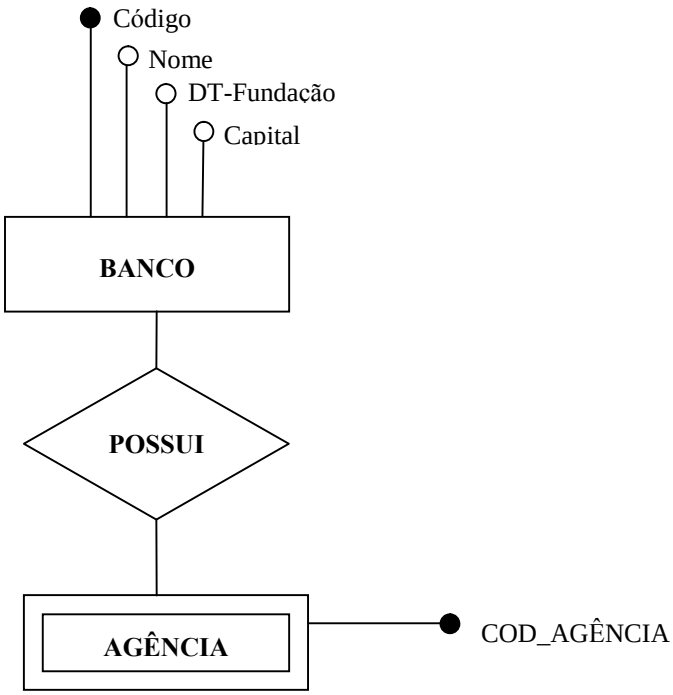
Auto-Relacionamento

Uma entidade de um determinado tipo relaciona-se com uma entidade do mesmo tipo



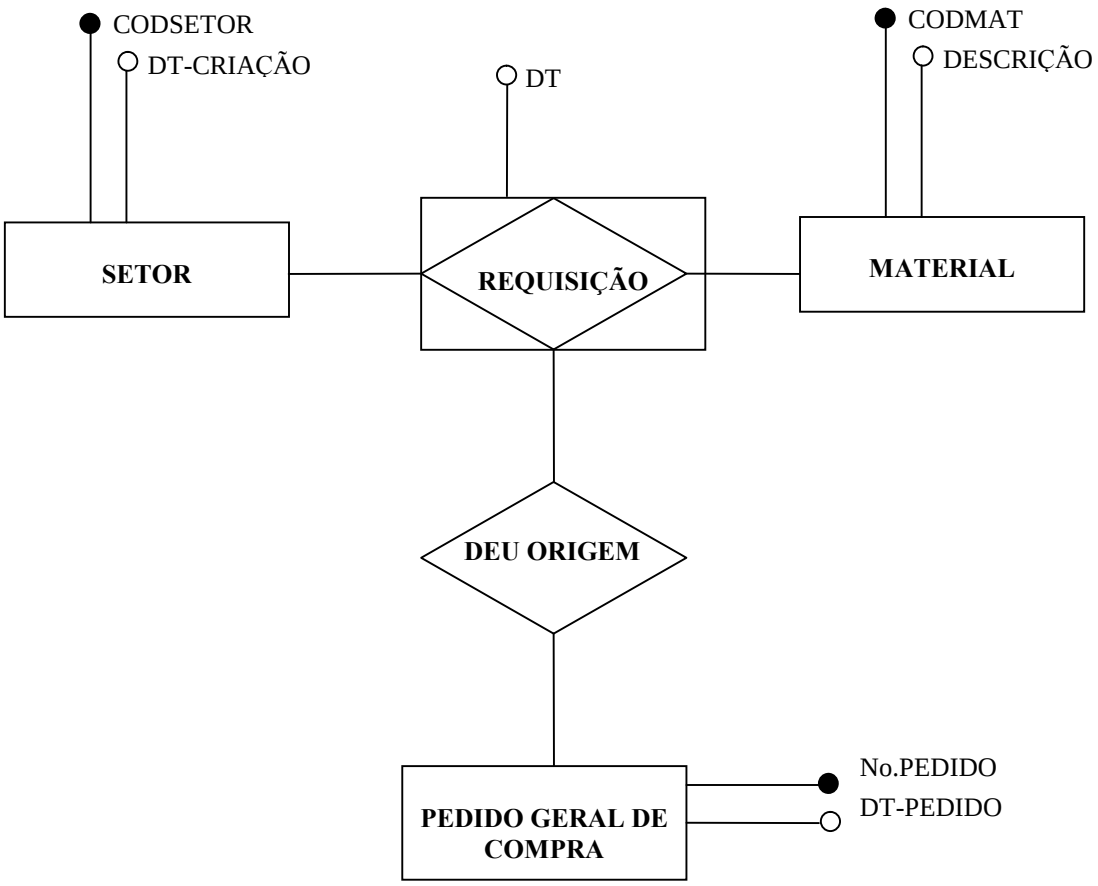
Entidades Fracas

São entidades que dependem de outras para existir e/ou ser indentificadas.



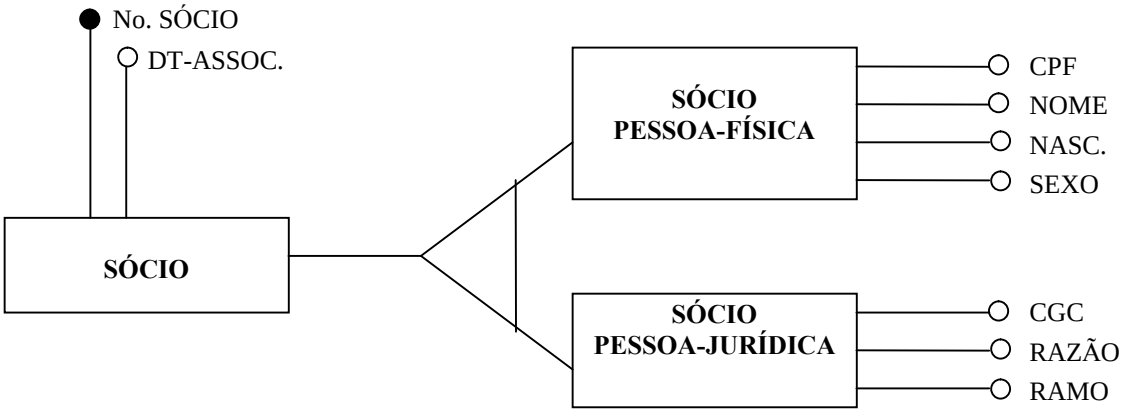
Entidade associativa ou agregada

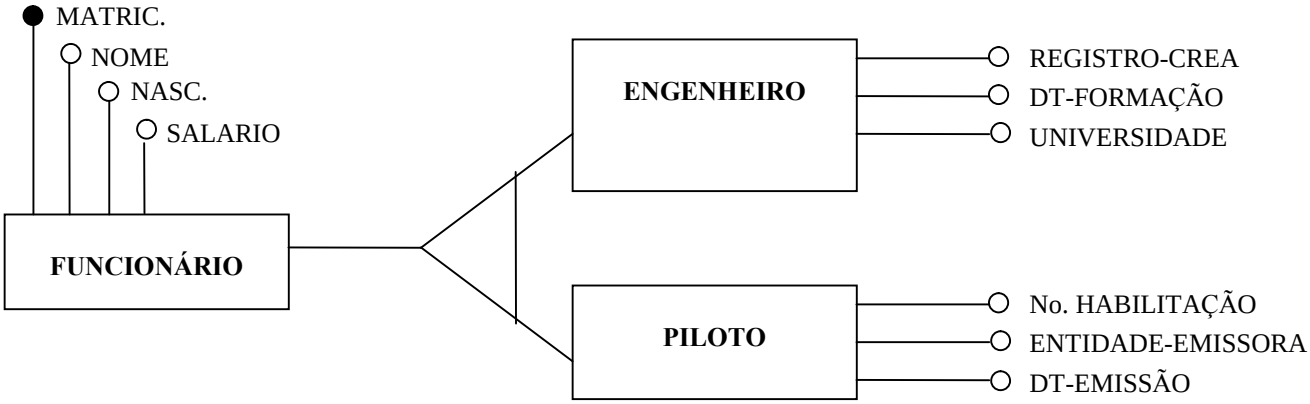
Uma entidade é dita associativa quando não existe por si só. Sua existência está condicionada à existência de duas ou mais entidades, a partir das quais é concebida.
Resulta da associação entre duas ou mais entidades, a partir das quais é concebida.



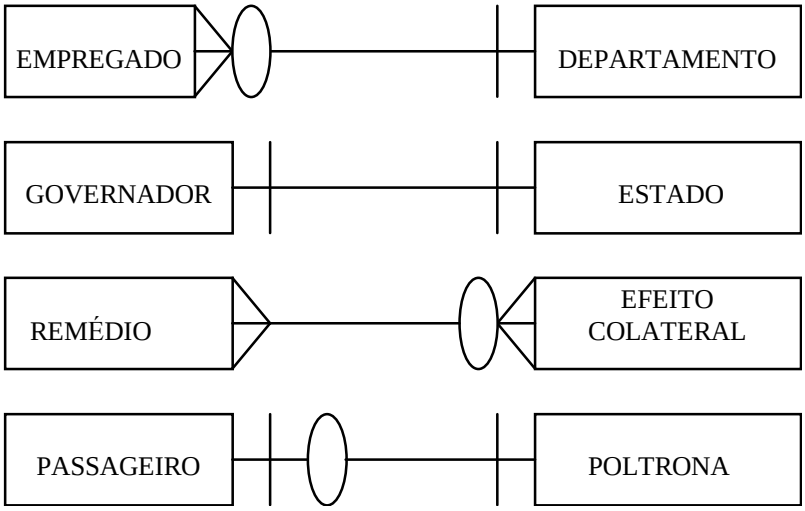
Uma entidade da Entidade-tipo "PEDIDO GERAL DE COMPRA" não deve ser associada nem com "SETOR" nem com "MATERIAL" separadamente, mas com o relacionamento "REQUISIÇÃO". Neste caso, este relacionamento é tratado como se fosse uma entidade, chamada **entidade associativa** ou **agregada.**

Generalização/Especialização ou Supertipo/Subtipo:





ALTERNATIVA DE NOTAÇÃO "PÉ DE GALINHA"



MINI-MUNDOS PROPOSTOS COMO ESTUDOS DE CASO

1)

Um restaurante necessita de automatizar algumas de suas atividades. Desta forma, solicitou um sistema que controlasse o pedido de encomendas feitas pelo cliente, através do telefone, o fornecimento e a compra de ingredientes para fazer os pratos e a composição de cada prato.

Toda encomenda feita pelo cliente, naturalmente, possui um número para identificá-la. Ao fazer uma encomenda, o cliente informa seu nome, endereço, telefone e os pratos que deseja, com a respectivas quantidades. Por exemplo, a encomenda 123 do cliente Pedro é constituída de 3 saladas mistas e 2 frangos grelhados simples. Cada prato possui o seu preço unitário.

Um dos objetivos do sistema é registrar **para cada prato, os ingredientes que o compõem**, com as respectivas quantidades. Ou seja, um pudim é composto de duas latas de leite condensado.
Outro objetivo é gerar uma listagem, contendo para cada fornecedor, os ingredientes que fornece. E, também, interessa que seja gerado um relatório contendo, para cada encomenda, o nome do cliente, o endereço, o telefone e os pratos pedidos com as respectivas quantidades e preços. O sistema deve registrar a compra de ingredientes, guardando o número da nota fiscal, a quantidade comprada de cada ingrediente, a data de compra e o nome do fornecedor, a fim de contabilizar o custo com a compra de ingredientes.

2) Numa empresa que deseja automatizar algumas de suas atividades, existem duas categorias de pessoas: os empregados e os dependentes destes empregados. Os empregados, por sua vez, podem ser classificados como assalariados, permanentes e temporários. Todos os empregados são cadastrados com nome, endereço, telefone do escritório e salário. Dos assalariados, interessa guardar o nível de salário e o nível de bonificação. Do empregado permanente, interessa guardar seu título e do temporário, de onde veio e quando tempo está emprestado. Somente os empregados permanentes podem chefiar um departamento, mas todos, obrigatoriamente, estão vinculados a um. Um empregado pode supervisionar outro e todos os empregados, de alguma maneira participam(trabalham em projetos). Todo projeto tem um gerente, que pode ser qualquer empregado.

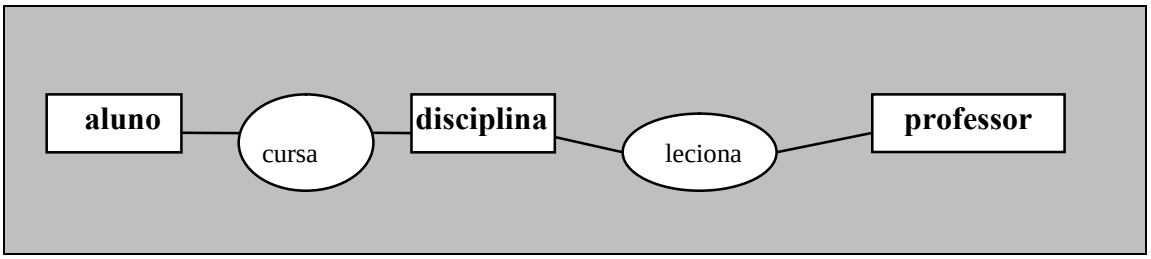
Os objetivos do sistema são: 1)gerar uma listagem contendo, para cada departamento, seu código, nome, o nome e endereço de seus empregados; 2)gerar uma listagem contendo, para cada departamento, o nome de seu chefe, com a data em que iniciou na chefia; 3)gerar um relatório, contendo para cada projeto, seu código, nome, data de início, duração provável em meses, o nome do gerente e os empregados que trabalham no mesmo e 4)gerar uma listagem contendo, para cada empregado temporário, seu nome, telefone , de onde veio e a data de início do seu empréstimo e quanto tempo deve ficar emprestado, para se ter uma idéia de quanto tempo mais podemos contar com ele na empresa.

A IMPORTÂNCIA DA MODELAGEM DE DADOS E DA ANÁLISE DAS FUNÇÕES

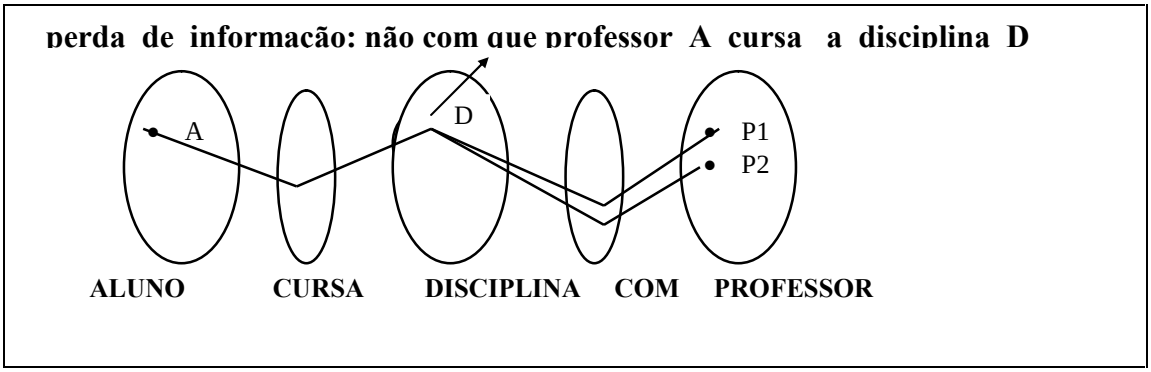
Perda de Informação

Suponha que se tenha modelar um sistema acadêmico com o objetivo de saber para um determinado aluno, as disciplinas que cursa e com que professor ele.

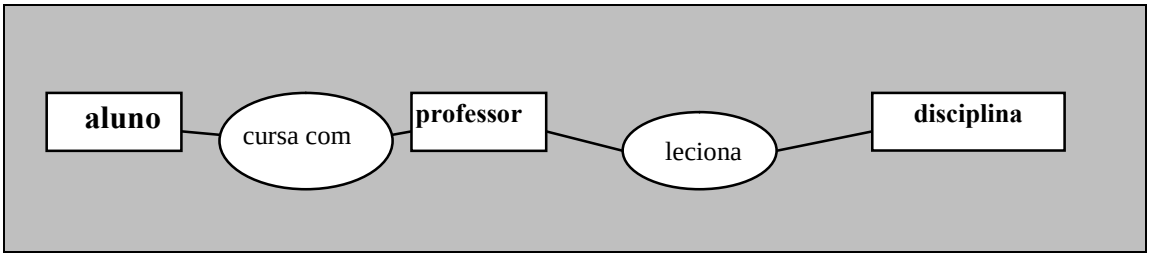
SE a situação **for**: uma disciplina é lecionada por apenas um professor e um professor só leciona uma disciplina.



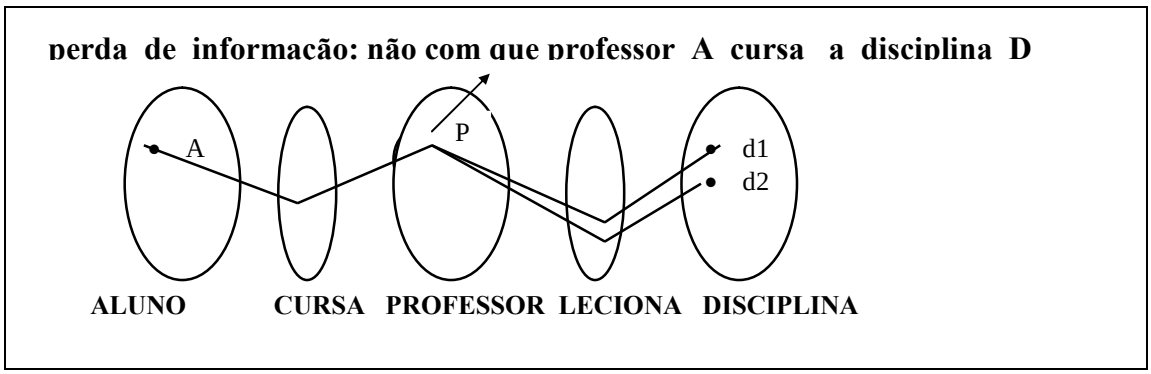
SE a situação **for**: uma disciplina pode ser lecionada por vários professores, mas um professor leciona apenas uma disciplina. Neste caso, suponha que se queira aproveitar o modelo de dados acima:



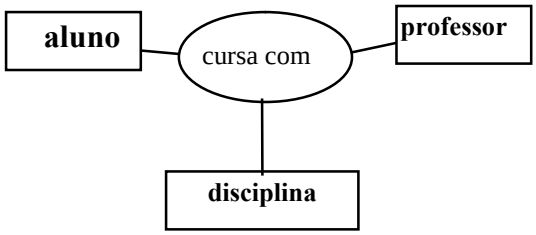
Solucão:



SE **for** a situação em que: um professor pode lecionar várias disciplinas e uma disciplina pode ser lecionada por vários professores.
Neste caso, também, suponha que se queira aproveitar o modelo de dados anterior: ocorrerá a perda de informação: não se saberá qual a disciplina que o aluno fez. Então, a solução será um relacionamento triplo.



Solução:



Restrições de Integridade e Operações

OPERAÇÕES sobre o Modelo de Dados: inclusão
exclusão
consulta
alteração

1. Cardinalidade e operação de inclusão de entidades

min igual a 0: não precisa criar relacionamento.

Exemplo:

```
insDisciplina(ent:codDisc:t-cod,
                  nomeDisc:t-nome,
                  chDisc:t-ch;
              sai:codRet:{ok, disciplina já existe}
              )
```

descrição: verificar se a disciplina existe. Se não existir, incluir a disciplina com os atributos fornecidos como entrada.

min # 0 :tem que criar relacionamento

Exemplo:inserir proprietário do sistema de shopping

```
insProprietário(ent:codProp:t-cod,
                    nomeProp:t-nome,
                    endProp:t-end,
                    porcent:t-port,
                    codLoja:t-loja  ;
                sai:codRet:{ok, proprietário já existe,
                            loja não existe}
               )
```

descrição: verificar se o proprietário existe. Se não existir, verificar se a loja existe. Se existir, incluir o proprietário com os atributos fornecidos como entrada e criar o relacionamento do tipo **participação** entre o proprietário e a loja fornecidos.

2. Cardinalidade e a exclusão de entidades

min igual a 0 ou 1, com a entidade associada com min igual a 0: desfazer os relacionamentos e excluir apenas a entidade em questão.

Exemplo:exclusão de disciplina de um sistema acadêmico.

```
excDisciplina(ent:codDisc:t-cod,
              sai: codRet:{ok, disciplina não existe}
              )
```

descrição: verificar se a disciplina existe. Se não existir, verificar se existem relacionamentos, se existirem, desfazê-los e excluir a disciplina, cujo código foi fornecido.

min igual a 0 ou 1, com a entidade associada com min igual a 1: verificar se tem que excluir as entidades que estão associadas, depois de desfazer relacionamento e, por fim, excluir a entidade em questão.

Exemplo:excluir loja do sistema de shopping

excLoja(**ent**:codLoja:t-cod;**sai**:codRet:{ok, loja não existe})

descrição: verificar se a loja existe. Se existir, verificar se existem relacionamentos do tipo lucro. Se existirem, desfazê-los. Verificar se existem relacionamentos do tipo participação. Se existirem, para cada proprietário associado à loja em questão, verificar se ele tem mais de um relacionamento desse tipo. Se tiver, desfazer o relacionamento. Do contrário, desfazer o relacionamento e excluir o proprietário em questão. Ao final, excuir a loja cujo código é fornecido com entrada.

3.**RI ER e RI NER e as operações**:

Exemplo: considere o modelo de dados em que um professor tem possuir habilitação em uma disciplina para poder lecioná-la.

insLeciona(**ent**:codDisc:t-cod,codProf:t-prof;
sai: codRet:{ok, disciplina não existe, professor não existe, habilitação não existe})

descrição: verificar se o professor existe. Se existir, verificar se a disciplina existe. Se existir, verificar se existe relacionamento do tipo **habilitação**. Se existir, incluir o relacionamento leciona entre professor e disciplina.