APOSTILA DE PROJETO DE SISTEMAS (ENGENHARIA DE SOFTWARE 1)

2020

MATERIAL DAS AULAS - Apostila

PROJETO DE SISTEMAS

Teorias e Conceito de Sistemas

Pensamento Contextual → Holístico ("imagem única" – sintetizar unidades em totalidades organizadas);

Pensamento processual → sistêmico (visão orgânica, lógica de um sistema – coerente com determinada linha de pensamento e/ou de ação);

No desenvolvimento do Pensamento Sistêmico, o aspecto processual foi enfatizado pelo biólogo austríaco Ludwig Von Bertalanffy no final da década de 30, e foi posteriormente explorado na cibernética na década seguinte.

Antes da década de 40, os termos "sistema" e "pensamento sistêmico" tinham sido utilizados por vários cientistas, mas foram as concepções de Bertalanffy de um sistema aberto e de uma teoria geral dos sistemas que estabeleceram o pensamento sistêmico como um movimento científico de primeira grandeza.

Com o forte apoio subseqüente vindo da cibernética, as concepções de pensamento sistêmico e de teoria sistêmica tornaram-se partes integrais da linguagem científica estabelecida, e levaram a numerosas metodologias e aplicações novas - engenharia de sistemas e análise de sistemas.

Cibernética – conceito criado por Norbert Weiner, nos anos 40, para o estudo das funções humanas de controle e da possibilidade de sua substituição por sistemas mecânicos e elétricos. A evolução deste conceito tem sido crítica para a concepção de complexos sistemas informáticos e experiências no âmbito da robótica.

Em 1951, Ludwig von Bertalanffy, publicou a obra " Teoria geral de sistemas".

– Partiu do princípio que um ser vivo não é apenas e simplesmente um aglomerado de elementos, sem integridade e organização. É o organismo um sistema que se mantém num mesmo estado, mas a matéria e a energia que o integram se renovam de uma forma constante, no que chamou de equilíbrio dinâmico do sistema.

CONCEITO

– O nome "sistema" tal qual é conhecido atualmente está relacionado a Teria Geral de Sistemas de Bertalanffy, que pesquisando sobre o comportamento dos organismos vivos constatou que, mesmo com uma grande variedade de formas e de características seres biológicos possuíam pontos em comum.

– Bertalanffy estendeu seus estudos a outros tipos de organismos (sociais, mecânicos, eletrônicos, etc.), verificando que, tal como acontece com os seres vivos, esses organismos não-naturais conservam, igualmente, certas características comuns, não importando sua natureza e complexidade.

– dessa maneira constatou que, independentemente desses organismos vivos e sociais possuírem inúmeros e variados elementos, todos apresentavam uma interação desses componentes com o objetivo de alcançar um determinado propósito, o que em última instância, era a finalidade central desses mesmos organismos

– baseado nessas observações e análises formulou a TGS, identificando os organismos sociais como um sistema visando à realização de objetivos comuns

O que é um Sistema Afinal?

Quantas vezes já nos referimos, ou ouvimos

a palavra sistema:

- *O sistema telefônico ficou mudo*

- *O sistema de coleta de lixo está perfeito*

- *Chefe, cheguei atrasado porque o sistema de trânsito desta cidade está um inferno*

Em qualquer sistema pode-se encontrar elementos característicos vinculados ao seu fim. No caso do sistema de trânsito, temos os veículos, motoristas, pedestre, ruas, guardas, placas, semáforos, etc... Observe que estes elementos interagem entre si (alguns são elementos passivos, outros não). Eles se completam e permitem ao sistema atingir seu objetivo.

Todo e qualquer sistema está inserido em um meio ambiente que o contém, ou seja, tudo que é externo a um sistema é chamada de seu meio ambiente.

Podem ser encontradas várias definições para sistema, as quais, muitas vezes são extremamente amplas, bastante abrangentes, em outros casos, carecem de uma generalização, como nos exemplos a seguir:

Conceito de sistema

"Sistema pode ser definido como um conjunto de elementos interdependentes que interagem com objetivos comuns formando um todo" (BALLESTERO ALVAREZ, 1990:17).

O que são entidades?

São aqueles elementos próprios (característicos, inerentes) do sistema em questão.Qualquer que seja o caso, eles sempre entram com certas características e quando saem, possuem novas características.

Ex.: No sistema educacional, encontramos como entidades os estudantes, professores, livros, administração (funcionários) e equipamentos.

As entidades em trânsito pelo sistema são a energia necessária para a sobrevivência

deste sistema

Módulos

Módulo é a parte do sistema responsável por uma tarefa bem definida e que pode ser acoplado a um sistema para permitir ao mesmo executar a tarefa disponibilizada pelo módulo. É responsável por atividades que satisfaz um assunto bem definido, as atividades do módulo utilizam tarefas e componentes comuns do sistema, um módulo ou vários módulos compõem um Sistema, um módulo também é representado por um grupo de componentes de software que atende a um assunto bem definido. O módulo realiza um assunto no processo de informação que o Sistema propõe atender.

Interdependência

Sendo as entidades responsáveis por apenas uma pequena parte do processo do sistema, implica que o desempenho de uma entidade , depende da outra , e a isto chamamos de *interdependência.*

Acoplamento : dependência

Coesão : qto maior a coesão, menor será seu nível de acoplamento de um módulo.

Subsistema

Um sistema que é parte de um sistema maior.

O sistema nervoso em relação ao corpo humano; o sistema de informações e o sistema gerencial e o sistema de produção e o sistema de comercialização, ambos em relação ao sistema-empresa; sistema de avaliação em relação ao sistema educacional

Tipos de Sistemas

Sistemas Abertos

– Interagem com o ambiente em que estão inseridos.

– Organizações sociais são sistemas abertos

Sistemas Fechados – Ex: Sistema de televisão, pois não muda e não sobre atualizações e não interfere no meio externo. (Com exceção das Smarts).

– Não trocam matéria ou energia com o ambiente em que estão inseridos.

– Raríssimos, podem ser considerados conceituais

Eventos do sistema

objetivos

– que se referem tanto aos objetivos dos usuários do sistema quanto aos do próprio sistema. É a razão de existência do sistema.

entradas

– caracteriza as forças que fornecem ao sistema o material, a informação e a energia para a operação do processo. (dados, energia, matéria)

processo de transformação (processamento):

– a função que possibilita a transformação de um insumo (entrada) em um produto, serviço ou resultado (saída).

saídas:

– que se referem aos resultados do processo de transformação. Podem ser definidas como as finalidades para as quais se uniram objetivos, atributos e relações do sistema. (informação, energia, matéria)

controles e avaliações:

– principalmente para verificar se as saídas estão coerentes com os objetivos estabelecidos. Para controlar avaliar de maneira adequada é necessário uma medida de desempenho do sistema, chamada padrão.

retroalimentação:

– pode ser considerado como a reintrodução de uma saída sob a forma de informação. É uma regulação retroativa desencadeada por uma nova informação, a qual afetará seu comportamento subseqüente.

energia :

– é a capacidade utilizada para movimentar e dinamizar o sistema, fazendo-o funcionar.

Sistemas e Empresa

O CONCEITO DE SISTEMA ABERTO É PERFEITAMENTE APLICÁVEL À ORGANIZAÇÃO EMPRESARIAL

Sistema e empresa: estas palavras estão intimamente ligadas, pois a empresa é um sistema e dentro dela existem diversos sistemas, independentemente do uso ou não da Tecnologia da Informação e seus recursos.

A teoria de sistemas permite reconceituar os fenômenos dentro de uma abordagem global, permitindo a inter-relação e integração de assuntos que são, na maioria das vezes, de naturezas completamente diferentes.

Elementos do sistema

Entender um sistema significa fazer as devidas conexões entre seus elementos, de modo que se ajustem logicamente em um todo.

Análise e Projeto de Sistemas

Problemas no Desenvolvimento de Sistemas

O processo de informatização das atividades administrativas traz inúmeras implicações, que vão desde mudanças nas rotinas de trabalho até reestruturações organizacionais.

Aspectos Relevantes

Ter conhecimento de todas as necessidades nos níveis estratégico, tático e operacional, não apenas nos sistemas atuais mas também, nos sistemas futuros, sejam eles automatizados ou não;

Inadequação dos sistemas é comum porque os profissionais de informática tendem, durante a fase de desenvolvimento dos sistemas, a dar maior atenção aos aspectos pertinentes à eficácia ou ao desempenho, em detrimento dos aspectos relativos à adequação dos sistemas às necessidades dos usuários;

Construção de sistemas efetivos, que sejam úteis para os usuários e para a empresa de modo global.

O Que Vem a Ser Especificação

É o detalhamento do comportamento de uma funcionalidade do sistema.

Problemas de Desenvolvimento de Sistemas

- Produtividade da equipe;
- Correção da especificação;
- Confiabilidade do sistema;
- Manutenibilidade do código-fonte;
- Segurança dos dados;
- Desempenho;
- Portabilidade do sistema;
- Perfil desejável dos analistas de sistemas;
- Aprendizado;
- Comunicação.

Análise de Sistema

Consiste nos métodos e técnicas de investigação e especificação da solução de problemas, a partir de requisitos levantados , para a criação e implementação de um software.

O Perfil do Analista de Sistemas

Habilidades importantes haja vista um bom desempenho nas atividades são elas:

Comunicação;

Capacidade de análise;

Conhecimento da área usuária;

Capacidade de negociação;

Administração de projetos;

Conhecimento técnico.

O Papel dos Profissionais

Analista: especificar quais são os requisitos do sistema do ponto de vista da eficácia, ou seja, garantir que o sistema alcance os objetivos globais da organização.

Projetista: volta-se para a eficiência, isto é, voltado para a obtenção do melhor desempenho individual dos componentes do sistema.

Programador: constrói (implementar) o sistema de acordo com as especificações feitas pelo projetista.

Aprendizado e a Comunicação

Aprendizado: entendimento dos requisitos do sistema.

Comunicação: apresentação da solução encontrada para aprovação dos usuários.

Aprendizado

Dividiu a fase de aprendizado em três grandes etapas. São elas:

Síncrise: processo informal, através do qual tenta-se juntar todos os fatos relevantes a respeito do problema em questão e descobrir os relacionamentos entre estes fatos;

Análise: consiste em identificar as variáveis ou pontos-chave do problema para a construção de um modelo simplificado de sua estrutura, com seus elementos e relações;

Síntese: tem-se a idéia de todo o sistema, quais as suas partes e qual a solução proposta para o problema.

Fase de Comunicação *versus* Problemas

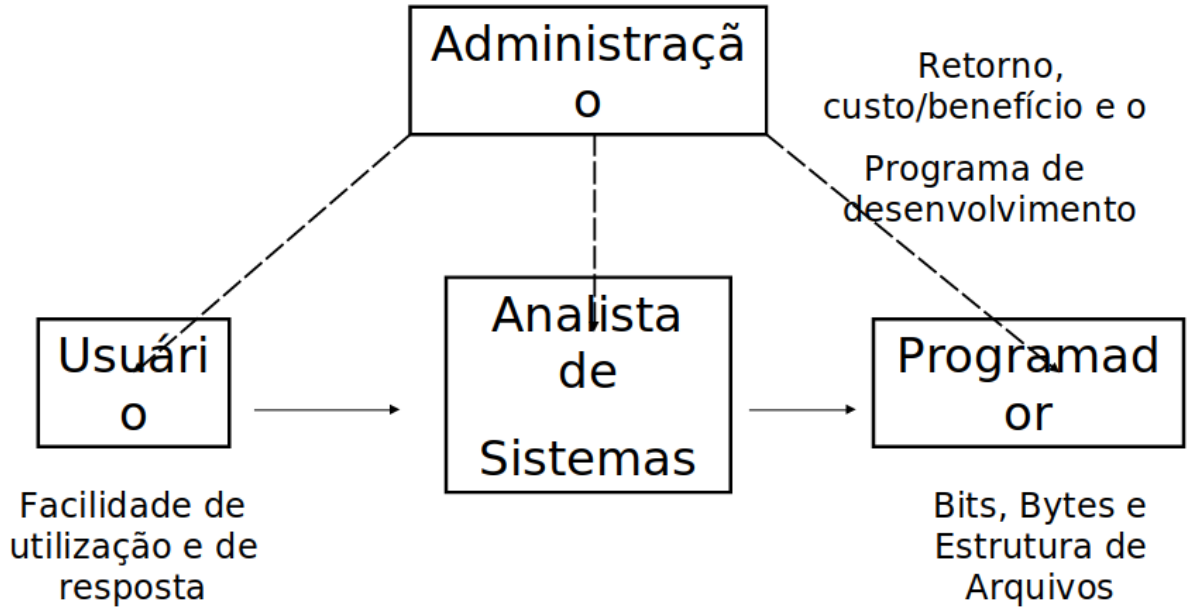
Problemas psicológicos: relacionados com percepção, atenção, motivação, atitudes, memória, hábitos de pensamentos;

Problemas semiológicos: relacionados com o emprego de signos e códigos para comunicar: palavra, gestos, tom de voz, coisas escritas etc.;

Problemas semânticos: relacionados com os significados das palavras, dos objetos e das pessoas, assim como sua interpretação;

Problemas sintáticos: relacionados com a estrutura ou organização dos conteúdos e dos signos;

Problemas cibernéticos: relacionados com a retroinformação e o diálogo, com a quantidade de idéias transmitidas por diversos canais e com a capacidade destes para levarem sinais.



Análise e Projeto de Sistemas

Sistema

É um conjunto de elementos interdependentes, que interagem , formando um todo organizado, visando atingir determinados objetivos.

Ciclo de Vida de Desenvolvimento de Sistemas

Análise de Sistemas: determina-se quais os requisitos do sistema, "o que" o sistema deve fazer, em termos essenciais. Objetiva interpretar e definir uma estrutura para um problema ainda não estruturado. Diz respeito à eficácia sem preocupar-se com a performance;

Projeto de Sistemas: determina-se "como" o sistema funcionará para atender aos requisitos especificados na fase anterior. Objetiva modelar o sistema de modo a implementar a solução idealizada na fase anterior, mas de acordo com os recursos tecnológicos presentes na empresa. Diz respeito à eficiência preocupando-se com a performance;

Implementação de Sistema: efetua-se a construção do sistema de acordo com o modelo de funcionamento especificado na fase anterior. Faz uso dos recursos tecnológicos presentes na empresa para atividades de programação.

Ciclo de Vida do Sistema

Ciclo de vida do software (ISO/IEC 12207)

Um software pode ter um ciclo de vida curto.

O que podemos entender por vida do software?

Não existe sistema pronto e acabado, pois ao longo de sua vida pode exigir:

Manutenção para atender legislação;

Melhorias e/ou implementações;

Eventuais correções de erros.

Fases do ciclo de vida do software

Concepção – nascimento do software;

Construção – análise e programação;

Implantação – testes e disponibilização aos usuários;

Implementação – ajustes após a implantação;

Maturidade – utilização plena;

Declínio – dificuldade de uso;

Manutenção – tentativa de sobrevivência (ajustes e melhorias)e,

Morte – parada definitiva do uso

Metodologias de Desenvolvimento de Sistemas

É um conjunto de regras e padrões que orientam a abordagem utilizada em todas as tarefas associadas com o ciclo de vida do desenvolvimento de sistemas.

Tipos de Metodologia

Entendendo.... Tradicional

Método: pode ser entendido como um procedimento a ser adotado para se atingir um objetivo. O método se vale de um conjunto de técnicas;

Técnica: pode ser entendida como sendo um modo apropriado de se investigar sistematicamente um determinado universo de interesse ou domínio de um problema. Para se expressar uma técnica faz-se uso de uma notação;

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

Metodologias ágeis

Nos últimos 10 anos as metodologias ágeis têm surgido motivadas pela necessidade de buscar alternativas para os modelos tradicionais de desenvolvimento de projetos. Essa necessidade se justifica quando, por exemplo , trabalhamos com projetos cujo escopo e tempo são reduzidos.

Processos de Desenvolvimento de Sistemas

Desenvolvimento

Todos os sistemas bem elaborados passam pelos estágios de:

Concepção: enfoca a questão "o quê?"

Desenvolvimento: enfoca a questão "como?"

Manutenção: enfoca "mudanças" – no sistema e no ambiente

O processo de desenvolvimento efetivo deve considerar:

Relação entre todas as tarefas

Ferramentas

Métodos utilizados

Treinamento

Motivação das pessoas envolvidas.

O Processo de desenvolvimento Software

Um conjunto estruturado de atividades para o desenvolvimento de um sistema de software.

Especificação;

Projeto;

Validação;

Evolução.

Especificação de Software

Processo de engenharia de requisitos

Estudo de viabilidade;

Elicitação e análise de requisitos;

Especificação de requisitos;

Validação de requisitos.

Projeto

É o processo de conversão da especificação de sistema em um sistema executável.

Projetar uma estrutura de software que atenda à especificação.

Transformar essa estrutura em um programa executável.

As atividades de projeto e implementação são fortemente relacionadas e podem ser intercaladas.

Validação

Verificação e validação (V &V) tem a intenção de mostrar que um sistema está em conformidade com a sua especificação e que atende aos requisitos do cliente.

Envolve processos de verificação e revisão, além de testes de sistema.

Esses testes envolvem a execução do sistema com casos de teste que são derivados da especificação de dados reais a serem processados por ele.

Evolução de Software

O software é inerentemente flexível e pode mudar.

Como os requisitos mudam durante as mudanças de circunstâncias de negócio, o software que apóia o negócio deve também evoluir e mudar.

Embora tenha havido uma separação entre desenvolvimento e evolução (manutenção), isso é cada vez mais irrelevante à medida que cada vez menos sistemas são completamente novos.

Modelos Genéricos de processo de software

O modelo cascata

Fases separadas e distintas de especificação e desenvolvimento.

Desenvolvimento evolucionário

Especificação, desenvolvimento e validação são intercalados.

Desenvolvimento orientada a reuso

Desenvolvimento Iterativo

Modelo Cascata (clássico)

O ciclo é representado pelas seguintes fases:

Requisitos:

Definição preliminar do escopo do sistema, restrições e conceitos alternativos.

Análise:

Especificação funcional do sistema (Projeto Lógico);

O ambiente do usuário é modelado através de Diagramas

DFD – Diagrama de Fluxo de dados

DER – Diagrama de Entidade Relacionamento

UML – Linguagem Unificada para Modelagem

Protótipos – apresentar interação usuário→ sistema

Projeto:
Especificação completa da arquitetura de hardware e software,
Estruturas de dados do sistema e caracterização de interfaces;
Determina tarefas que cada pessoa envolvida deverá executar.
Refinamento dos diagramas
Construção de pseudocódigos

Codificação (Implementação):
Codificação e teste individual dos programas

Teste:
Teste dos componentes integrados do sistema
A partir da especificação estruturada (na análise) deve começar os casos de aceite.
Plano de testes – pessoa responsável por testar, comparar resultados obtidos com esperados.
Testes de desempenho – tempo de resposta
Testes de vias normais – rotina correta
Testes de vias de erros – rotina com valores errados.
Final – relatório com resultados obtidos.
Envolvimento com o usuário para aprovação:

Modelo cascata

Implantação:

Entrega da documentação (manuais)

Treinamento dos usuários

Implantação de maneira gradativa.

Acompanhamento pós-implantação.

Operação e Manutenção:

Utilização do sistema e modificações decorrentes de erros, mudança de necessidades, etc.

Desvantagem do modelo Cascata

A principal desvantagem do modelo cascata é a dificuldade de acomodação das mudanças depois que o processo está em andamento. Uma fase tem de estar completa antes de passar para a próxima.

Modelo Evolucionário(prototipação)

Desenvolvimento exploratório

O objetivo é trabalhar com os clientes e desenvolver um sistema final a partir de uma especificação inicial. Deve iniciar com requisitos mal compreendidos e adicionar novas características à medida que forem propostas pelo cliente.

Prototipação

O objetivo é compreender os requisitos de sistema. Deve iniciar com requisitos mal compreendidos para esclarecer o que é realmente necessário.

Problemas

Falta de visibilidade de processo;

Os sistemas são freqüentemente mal estruturados;

Habilidades especiais (por exemplo, em linguagens para prototipação rápida) podem ser solicitadas.

Aplicabilidade

Para sistemas interativos de pequeno e médio portes;

Para partes de um sistema de grande porte (por exemplo, a interface de usuário);

Para sistema com curto ciclo de vida.

Desenvolvimento orientado a reuso

Baseado em reuso sistemático onde sistemas são integrados a partir de componentes

Estágios do processo

Análise de componentes;

Modificação de requisitos;

Projeto de sistema com reuso;

Desenvolvimento e integração.

Esta abordagem está se tornando cada vez mais usada à medida que padrões de componentes têm surgido.

O modelo orientado a reuso tem a vantagem óbvia de reduzir a quantidade de software a ser desenvolvida e, assim, de reduzir custos e riscos. Contudo, as adequações nos requisitos são inevitáveis e isso pode resultar em um sistema que não atenda às reais necessidades dos usuários.

Desenvolvimento Iterativo

Requisitos de sistema SEMPRE evoluem no curso de um projeto e, sendo assim, a iteração de processo, onde estágios iniciais são retrabalhados, é sempre parte do processo dos sistemas de grande porte.

A iteração pode ser aplicada a qualquer um dos modelos genéricos do processo.

Duas abordagens (relacionadas)

Desenvolvimento incremental – quando as fases de especificação, projeto e implementação de software são desdobradas em uma série de estágios;

Desenvolvimento espiral – quando o desenvolvimento do sistema evolui a partir de um esboço inicial, em direção ao sistema final desenvolvido

Desenvolvimento Incremental

Ao invés de entregar o sistema como uma única entrega, o desenvolvimento e a entrega são separados em incrementos, sendo que cada incremento fornece parte da funcionalidade solicitada.

Os requisitos de usuário são priorizados e os requisitos de prioridade mais alta são incluídos nos incrementos iniciais.

Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, embora os requisitos para os incrementos posteriores possam continuar evoluindo.

Vantagens do Des. Incremental

O valor pode ser entregue para o cliente com cada incremento e, desse modo, a funcionalidade de sistema é disponibilizada mais cedo.

O incremento inicial age como um protótipo para auxiliar a elicitar os requisitos para incrementos posteriores do sistema.

Riscos menores de falha geral do projeto.

Os serviços de sistema de mais alta prioridade tendem a receber mais testes.

Modelo Incremental

Essa abordagem permite que as atividades do projeto possam ser subdivididas e que ocorram em paralelo.

Desenvolvimento Espiral

O processo é representado como uma espiral ao invés de uma seqüência de atividades com realimentação.

Cada loop na espiral representa uma fase no processo.

Sem fases definidas, tais como especificação ou projeto, os loops na espiral são escolhidos dependendo do que é requisitado.

Os riscos são explicitamente avaliados e resolvidos ao longo do processo.

Setores do Modelo Espiral

Definição de objetivos

Objetivos específicos para a fase são identificados.

Avaliação e redução de riscos

Riscos são avaliados e atividades são realizadas para reduzir os riscos-chave.

Desenvolvimento e validação

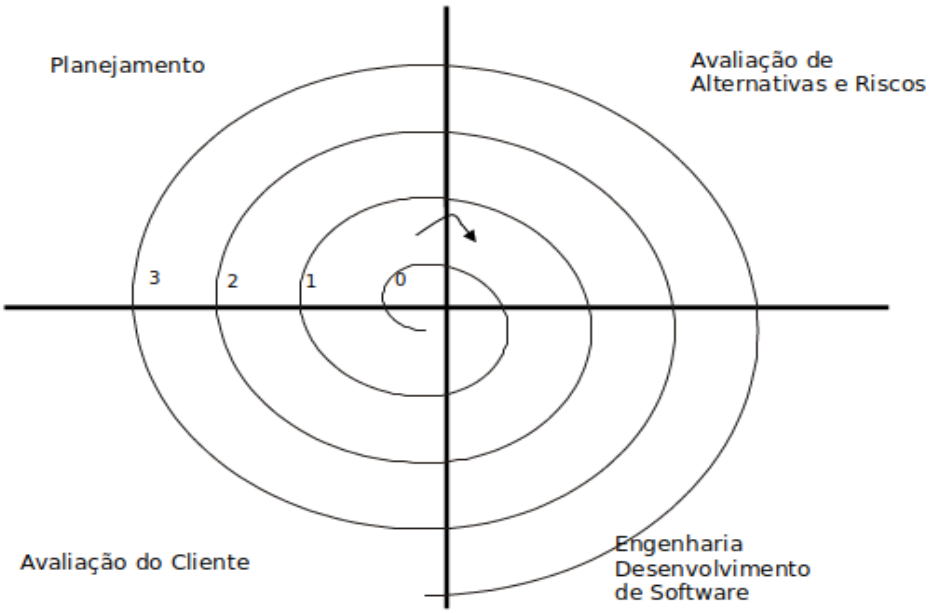
Um modelo de desenvolvimento para o sistema, que pode ser qualquer um dos modelos genéricos, é escolhido.

Planejamento

O projeto é revisado e a próxima fase da espiral é planejada.

Modelo Espiral

É uma aglutinação das melhores características existentes nos modelos antecessores e é apresentado em uma dimensão radial, cujas atividades iniciais encontram-se no centro da espiral, e na medida em que o desenvolvimento do software avança, percorre-se a espiral do centro pra fora.



Padrões de Sistemas Iterativos

RUP

Desenvolvimento Ágil

RUP(Processo Unificado Racional)

É um modelo de processo moderno derivado do trabalho sobre a UML e do Processo Unificado de Desenvolvimento de Software associado.

Normalmente descrito a partir de três perspectivas:

Uma perspectiva dinâmica que mostra as fases ao longo do tempo;

Uma perspectiva estática que mostra atividades de processo;

Uma perspectiva prática que sugere boas práticas.

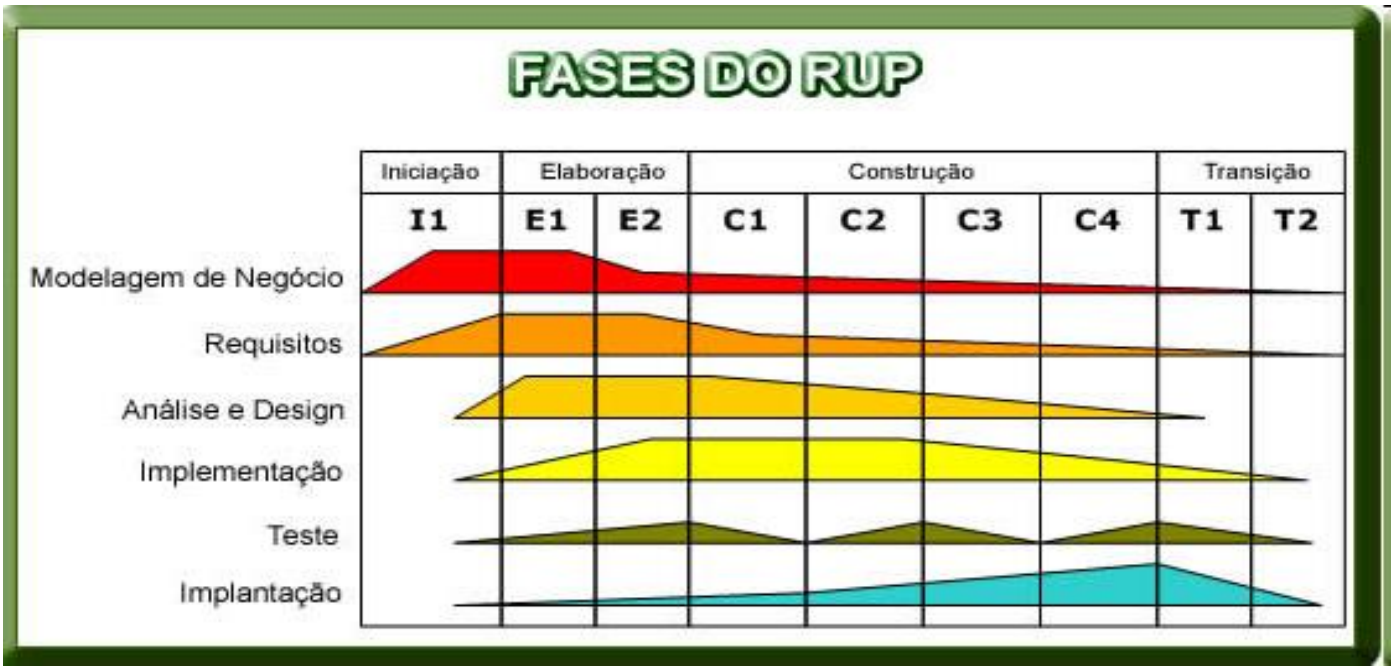
Fases do RUP

Concepção- Estabelecer o business case para o sistema.

Elaboração - Desenvolver um entendimento do domínio do problema e a arquitetura do sistema.

Construção - Projeto, programação e teste de sistema.

Transição - Implantar o sistema no seu ambiente operacional.



Boas Práticas do RUP

Desenvolver o software iterativamente

Gerenciar requisitos

Usar arquiteturas baseadas em componentes

Modelar o software visualmente

Verificar a qualidade de software

Controlar as mudanças do software

Desenvolvimento Ágil

Os princípios do desenvolvimento ágil valorizam:

Garantir a satisfação do consumidor entregando rapidamente e continuamente softwares funcionais;

Softwares funcionais são entregues freqüentemente (semanas, ao invés de meses);

Softwares funcionais são a principal medida de progresso do projeto;

Até mesmo mudanças tardias de escopo no projeto são bem-vindas.

Cooperação constante entre pessoas que entendem do 'negócio' e desenvolvedores;

Projetos surgem através de indivíduos motivados, entre os quais existe relação de confiança.

Design do software deve prezar pela excelência técnica;

Simplicidade;

Rápida adaptação às mudanças;

Indivíduos e interações mais do que processos e ferramentas;

Software funcional mais do que documentação extensa;

Colaboração com clientes mais do que negociação de contratos;

Responder a mudanças mais do que seguir um plano

Métodos ágeis produzem pouca documentação em relação a outros métodos, sendo este um dos pontos que podem ser considerados negativos. É recomendada a produção de documentação que realmente será útil.

Ex.: XP, SCRUM....

XP (Extreme Programming)

Uma abordagem baseada no desenvolvimento e na entrega de incrementos de funcionalidade muito pequenos.

Baseia-se no aprimoramento constante do código, no envolvimento do usuário na equipe e no desenvolvimento e programação em pares.

Princípios Básicos

Feedback rápido

Presumir simplicidade

Mudanças incrementais

Abraçar mudanças

Trabalho de alta qualidade.

Algumas práticas

O desenvolvimento é feito em iterações semanais;

Desenvolvedores e cliente reúnem-se para priorizar as funcionalidades;

O cliente identifica prioridades e os desenvolvedores as estimam;

O cliente recebe novas funcionalidades, completamente testadas e prontas para serem postas em produção (final de cada semana);

A liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando;

Procura facilitar a comunicação com o cliente, entendendo a realidade dele;

A equipe de desenvolvimento é formada pelo cliente e pela equipe de desenvolvimento;

Reuniões em pé para não se perder o foco nos assuntos;

A programação em par/dupla num único computador.

SCRUM

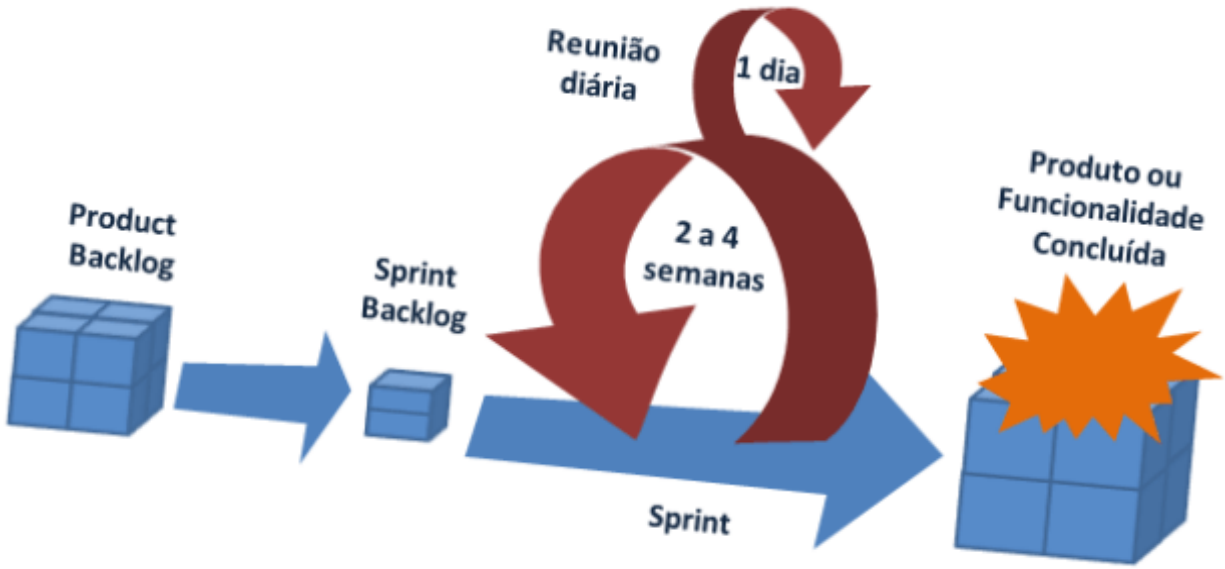
Scrum é um esqueleto de processo que contém grupos de práticas e papéis pré-definidos.

PAPÉIS:

o ScrumMaster, que mantém os processos (normalmente no lugar de um gerente de projeto)

o Proprietário do Produto, que representa os *stakeholders* e o negócio

a Equipe, um grupo multifuncional com cerca de 7 pessoas e que fazem a análise, projeto, implementação, teste etc.



Clientes se tornam parte da equipe de desenvolvimento (os clientes devem estar genuinamente interessados na saída);

Entregas freqüentes e intermediárias de funcionalidades 100% desenvolvidas;

Planos freqüentes de mitigação de riscos desenvolvidos pela equipe;

Discussões diárias de status com a equipe;

A discussão diária na qual cada membro da equipe responde às seguintes perguntas:

O que fiz desde ontem?

O que estou planejando fazer até amanhã?

Existe algo me impedindo de atingir minha meta?

Transparência no planejamento e desenvolvimento;

Reuniões freqüentes com os *stakeholders* (todos os envolvidos no processo) para monitorizar o progresso;

Problemas não são ignorados e ninguém é penalizado por reconhecer ou descrever qualquer problema não visto;

Outros métodos ágeis

FDD – Feature Driven Development

ASD – Adaptative Software Development

CRYSTAL – Família Crystal de Cockburn

ICONIX – Iconix process

DSDM – Dynamic System Development Methodology

OpenUP – Processo Unificado Aberto

- Preza pela qualidade encontrada no modelo RUP, com a agilidade do XP e com as melhores práticas do gerenciamento do modelo SCRUM;

PP – Pragmatic Programming

Inclui um conjunto de melhores práticas de programação, procura abranger o que há de melhor em outros métodos ágeis. Suas práticas tem uma perspectiva pragmática de desenvolvimento iterativo com foco incremental.

1 - FDD – Feature Driven Development

Histórico

A metodologia Ágil Feature Driven Development, ou FDD, como é comumente chamada, foi criada em Singapura nos anos 1990, mais especificamente em meados de 1997/1998. Foi utilizada pela vez para o desenvolvimento de um sistema bancário internacional, considerado inicialmente inviável de ser desenvolvido em um prazo predeterminado. Criada por Jeff De Luca e Peter Coad, A FDD é uma metodologia ágil robusta e muito utilizada.

Características básicas da metodologia

A FDD proporciona uma forma de trabalho que agrada todos os envolvidos no projeto, sugerindo formas de interação e controle fáceis e inteligentes. Os desenvolvedores se sentem muito à vontade durante a implementação, pois FDD possui um conjunto de regras de fácil entendimento e de resultados rápidos, trazendo também vantagens para o cliente.

Beneficio ao cliente por meio de trabalho significativo

Conforme determina o manifesto ágil , essa metodologia também procura realizar seus trabalhos de maneira significativa desde o inicio do projeto de software, ou seja, no inicio da implementação visa sempre agregar valor ao cliente, promovendo a disponibilização daquilo que já foi desenvolvido e testado.

A FDD fornece uma estrutura capaz de atender todos os tipos de equipes de trabalho, desde pequenas até grandes. Oferece um conjunto sólido de atribuições capazes de ser ampliadas, se necessário, mesmo em projetos descentralizados, ou até mesmo em projetos web.

FDD

Software de Qualidade

A produção de software da FDD engloba um conjunto de métricas de qualidade a serem aplicadas durante o desenvolvimento do projeto de software. A FDD enfatiza sempre que o software deve ter qualidade desde o inicio de sua implementação.

Entrega dos resultados frequentes, tangíveis e funcionais.

Durante a fase de implementação, a FDD sugere que sempre que alguma funcionalidade for implementada e testada ela deve ser disponibilizada aos usuários.

Software de Qualidade

A produção de software da FDD engloba um conjunto de métricas de qualidade a serem aplicadas durante o desenvolvimento do projeto de software. A FDD enfatiza sempre que o software deve ter qualidade desde o inicio de sua implementação.

Entrega dos resultados frequentes, tangíveis e funcionais.

Durante a fase de implementação, a FDD sugere que sempre que alguma funcionalidade for implementada e testada ela deve ser disponibilizada aos usuários.

DSDM – Dynamic System Development Methodology

Origens e Características básicas

Dynamic System Development Methodology (DSDM) é uma metodologia de desenvolvimento de projetos de software centrada em estabelecer os recursos e o tempo fixo para o desenvolvimento de um projeto, ajustando suas funcionalidades de maneira a atender os prazos estipulados. Foi criada em 1994 e continua sendo mantida por um consórcio de diversas empresas associadas.

A estrutura da DSDM baseia-se em nove princípios, que são consideradas boas práticas de utilização dessa metodologia. A seguir descrevemos resumidamente esses princípios , com o objetivo de caracterizar essa metodologia.

Princípios da DSDM

Participação ativa dos usuários e stakeholders: todos os envolvidos no projeto e conhecedores do processo devem acompanhar o desenvolvimento a fim de garantir que tudo seja entregue a tempo e de acordo com o solicitado.

Abordagem cooperativa e compartilhada: todas as partes interessadas devem cooperar e assumir o compromisso das entregas de partes do software, bem como escolher a ordem de sua implementação, ou seja, essas escolhas devem ser feitas em comum acordo.

Equipes com poder de decisão: as pessoas envolvidas no desenvolvimento devem ter conhecimento e autonomia suficiente para decidir o destino do sistema; não é tolerável aguardar decisões por um longo período de tempo.

Entregas contínuas que fazem a diferença: é um critério básico da DSDM tentar trazer um retorno ao cliente do projeto desde o inicio, a fim de promover a validação do que está sendo feito (fazer o produto certo).

Desenvolvimento iterativo e incremental: a ideia é convergir o sistema para o negócio e melhorá-lo continuamente em um processo iterativo, buscando e corrigindo os problemas o mais cedo possível.

Feedback: o foco está nas frequentes entregas de produtos de software,permitindo ao usuário colocar suas opiniões e solicitar modificações.

Todas as possíveis alterações durante o desenvolvimento devem ser reversíveis: deve-se permitir que o impacto das modificações seja testado e, caso não surta os efeitos desejados, as modificações possam ser restabelecidas.

Fixar os requisitos essenciais: os requisitos principais devem ser capturados no início, a fim de estabelecer os objetivos gerais. Os requisitos específicos serão norteados pelos principais e sujeitos a modificações, mas sempre focando os objetos.

Teste em todo o ciclo de vida: devido ao prazo normalmente apertado não podemos deixar a fase de testes exclusivamente no final da implementação, devendo ser realizada em todas as fases e componentes do projeto. O teste de regressão é normalmente o mais utilizado pela DSDM devido ao desenvolvimento incremental.

3 - Adaptative Sotware Development -ASD

Origem

Foi criado com base na necessidade de implementar projetos de software de maneira rápida. Ele traz um conjunto de perspectivas embasadas nas metodologias tradicionais, as quais agilizam o desenvolvimento e ao mesmo tempo garantem um software de qualidade para o cliente.

Foi criada por Sam Bayer e James (Jim) Highsmith, em 1977, partindo de anos de experiência em metodologias tradicionais somados às técnicas RAD (Desenvolvimento Rápido de Aplicativos), que utilizam ferramentas de apoio ao desenvolvedor.

Ciclo de vida

A metodologia ASD trata seu ciclo de vida levando em conta três itens: a especulação, a colaboração e o aprendizado.

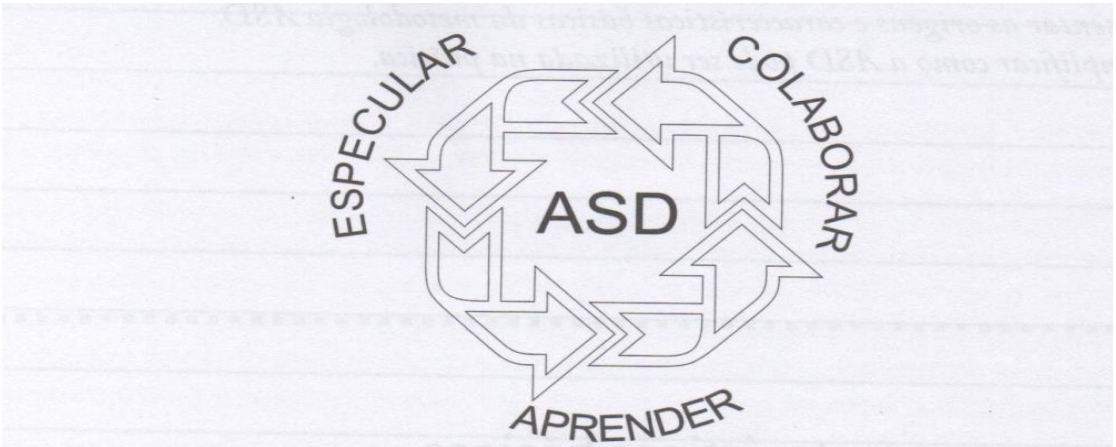


Figura 8.1 – Ciclo de vida ASD. Fonte: Arquivo pessoal do autor.

Fases do ciclo

Especular – O termo especular equivale a observar, indagar, pesquisar; em outras palavras, questionar as causas de algum assunto.

Colaborar – A colaboração, nesse contexto, traduz –se no ato de criação e manutenção de elementos de software capazes de atenderas emergências por parte do cliente. O gerente de projeto, juntamente com os stakeholders, decide as prioridades, ou seja, aquilo que é previsível em curto prazo, fazendo com que sejam gerados benefícios aos usuários.

Aprender – O aprendizado está no fato de que, com o andamento do projeto, o desenvolvedor passa a conhecer os desejos do cliente, adquirindo experiência e domínio do assunto e, consequentemente, da aplicação, além de conhecer antecipadamente os próximos passos a serem desenvolvidos.

Fases

1ª Fase – Levantar requisitos (técnicas) – Brainstorm

2ª Fase – Ficha de História do Usuário – Relato da descrição do requisito

3ª Fase – Protótipos (Lista de RF e NF)

4ª Fase – Cronograma de Desenvolvimento

5ª Fase – Reunião de lançamento do Desenvolvimento

6ª Fase – Reunião da Revisão

7ª Fase – Ciclo de Iteração (Incrementos) – Entrega de valor

8ª Fase – Software produzido

9ª Fase – testes manuais e automatizados (aceitação do cliente)

10ª Fase Termo de encerramento – entrega do software

4 – Família Crystal de Cockburn

Origem e características

Em 1998, Alistair Cockburn criou uma família de metodologias com o objetivo de suprir as necessidades observadas e colaborar para a resolução desses problemas, batizando-a com o nome de família Crystal. Uma família de metodologias com um código genético comum que atende a diferentes tipos e tamanhos de projetos. Essa metodologia foi dividida em cores : quanto mais escuro, mais crítico o sistema seria. Cada cor tem um propósito e elas são separadas de acordo com a criticidade, ou seja, projetos menores envolvem poucos desenvolvedores e, no caso de problemas, o prejuízo tende a ser menor. Já os projetos maiores ou de segurança crítica normalmente envolvem mais profissionais, portanto o prejuízo deve ser muito maior, podendo colocar a vida das pessoas em risco.

Família Crystal de Cockburn

Princípios e filosofia da Crystal (Independente das cores)

1) Trabalho face a face com o cliente;

2) Peso significa custo; (complexidade = custo maior)

3) Usar metodologias diferenciadas para equipes maiores;

4) Mais cerimônias maior criticidade; (mais diálogos)

5) Comunicação eficiente;(feedback)

6) Habitalibidade; (tolerância com seres humanos)

Ciclo de Vida

Cores

Cores	Número de desenvolvedores	Em caso de falha....
Clear	1-6	Perdem dinheiro, mas recuperam facilmente
Yellow	7-20	Perdem dinheiro discretamente
Orange	21-40	Perdem dinheiro substancialmente
Red	41-100	Há perda substancial de dinheiro e, possivelmente, vidas humanas

Outras Funções

A partir da metodologia Orange da familia Crystal, pode ser necessário criar outras funções nas equipes

A) Gerente de projetos

B) Arquiteto de software

C) Facilitador técnico;

D) Especialista em usabilidade

Iconix

HISTÓRICO

Foi criada em 1999 e publicada em 2005 por Doug Rosenberg, fundador e presidente da Iconix Software Engineering Inc., juntamente com Matt Stephens e Mark Colling-Cope, especialistas em projetos de desenvolvimento ágil, ambos com larga experiência em desenvolvimento, Arquitetura, Interação e design de Software.

Iconix diz respeito a uma metodologia dirigida por casos de uso, que tem como objetivo estudar e comunicar o comportamento do sistema sob o ponto de vista de um consumidor ou usuário final. Embora tenha sido criada antes da UML, utilize-se de um de seus diagramas, que permite a compreensão da solução proposta por todas as pessoas, mesmo que não envolvidas no processo de desenvolvimento.

Características

O processo Iconix, além de fazer uso da linguagem de modelagem UML, possui uma característica exclusiva chamada “rastreabilidade dos requisitos”.

Essa metodologia dá um enfoque especial aos requisitos e é perfeitamente adequada a empresas que pleiteiam um certificado de qualidade de software, mais precisamente certificados que exigem como primeiro passo a gerencia de seus requisitos, como CMMI, MPS-BR, etc.

Iconix é flexível e aberta. Utiliza o diagrama de robustez, que representa um diagrama de classes estereotipado.

Fases do Iconix

A) Modelagem do domínio (Fase de entrevistas, questionários..)

B) Identificação do dominio dos objetos quanto ao mundo real (Ficha de requisitos)

C) Comportamento dos requisitos por meio de casos de uso

D) Análise de Robustez

- Diagrama de Robustez

E) Comportamento dos Objetos (diagrama de sequencia, diagrama de atividades)

F) Modelo Estático (digrama de classe)

G) Codificação

h) Testes (caixa preta)

Tabela comparativa

Metodologia	Criado por	Ano	Requisitos	Iterações	Incremental	Validação e teste	Diagramas da UML
XP	Kent Back	1996	Clientes escrevem	1-4 semanas com participação do cliente	Programação em duplas com *refactoring*	Teste de unidade e aceitação	Classes, caso de uso, sequência ou atividades
SCRUM	Jeff Sutherland, Ken Schwaber e Mike Beedle	1995	*Product Backlog*	*Sprint* de 30 dias	*Sprint* Backlog	---	---
FDD	Jeff de Luca e Peter Coad	1997	Artefato ficha	2 semanas por prioridade	Refinamento	Inspeções pelos programadores	Classes, caso de uso, sequência e atividades
ASD	Sam Bayer e James Highsmith	1997	Sessões	4 a 8 semanas por quantidade de requisitos	Implementação das quantidades de requisitos	Testes funcionais	Classes, caso de uso, sequência e atividades
CRYSTAL	Alistair Cockburn	1998	Entrevista, *briefing* e questionários	2 semanas	*Release* em sequência	Testes de regressão	Classes, caso de uso, sequência e atividades
ICONIX	Doug Rosenberg, Matt Stephens e Mark Colling Cope	1999	Entrevista, questionários e atas (rastreáveis)	2 semanas por prioridade	*Releases* em sequência	Testes funcionais, aceitação e de unidade	Classes, caso de uso, sequência ou atividades e robustez
DSDM	Consórcio de empresas associadas	1994	Workshop e estudo de viabilidade	2 semanas MOSCOW	*Timeboxing*	Testes de regressão	Classes, caso de uso, sequência e atividades
APSO	José Henrique T. C. Sbrocco e Paulo Cesar de Macedo	2011	Documento de visão, *Product Backlog*	1 semana (por orientação)	*Sprint Backlog*	Inspeções e testes funcionais	Classes, caso de uso, sequência ou atividades

Tabela 13.1 – Comparação das principais características das metodologias ágeis.

Modelagem de Sistemas

Desenvolveu-se linguagens que incluíam recursos necessários para definir as necessidades do usuário, independente de qualquer restrição relativa à implementação. A linguagem de especificação contorna o termo técnico, de modo a se tornar mais inteligível ao usuário não-técnico e, por outro lado, continue precisa permitindo produzir uma especificação que seja base para um subseqüente projeto operacional dos SI.

Abordagens Utilizadas

Tentar decompor, primeiramente, o problema em subproblemas que possuam menor complexidade que o problema original.

Decompor o problema por pontos de vista diferentes.

Salienta-se que deve utilizar um método de decomposição que garanta a possibilidade, a partir dos subproblemas, reconstituir o todo.

Lembre-se

Antes de construir um sistema de informações, deve-se elaborar um modelo que seja capaz de expressar com a máxima fidelidade possível o conhecimento que se tem do ambiente onde o sistema será implantado, de modo a satisfazer a todos os requisitos necessários. Se, por um lado, o custo de um sistema é função do desempenho de seus componentes, por outro, o valor deste mesmo sistema é função da utilidade que ele tenha para seus usuários. A boa técnica de análise dos sistemas de uma organização preconiza uma boa interação – vale dizer, comunicação entre os usuários e os técnicos de informática.

Utilidade do Modelo

Estabelecer uma visão comum entre usuário e analista do ambiente antes da automação;

Servir como suporte para negociação e especificação de requisitos e possibilidades futuras para o sistema;

Representar, avaliar e refinar conceitos do projeto;

Escalonar a informatização em fases, com produtos bem definidos e dependência mínima entre as fases;

Tratar a complexidade do problema por níveis de abstração, começando pela visão mais abstrata e descendo a visões progressivamente mais detalhadas;

Promover indicações quantitativas do escopo e da complexidade do problema;

Prover facilidades para geração de testes de aceitação.

Modelos

Três perspectivas que complementam-se:

Modelo funcional: apresenta uma visão estruturada das funções ou dos processos que compõem a organização;

Modelo de dados: apresenta uma visão dos dados que serão armazenados;

Modelo de controle: representa as transformações de controle e uma visão do comportamento da organização em relação aos seus diferentes estados válidos, cada sendo caracterizado por uma determinada resposta fornecida quando a organização estiver sujeita a um certo conjunto de estímulos.

Princípios da Modelagem

1°- A escolha dos modelos a serem criados tem profunda influência sobre a maneira como um determinado problema é atacado e como uma solução é definida.

2° - Cada modelo poderá ser expresso em diferentes níveis de precisão.

3° - Os melhores modelos estão relacionados à realidade (o segredo será ter certeza de que sua simplicidade não ocultará detalhes importantes).

4° - Nenhum modelo único é suficiente. Qualquer sistema não trivial será melhor investigado por meio de um pequeno conjunto de modelos quase independentes.

UML (UNIFIED MODELING LANGUAGE) A linguagem unificada de modelagem

CONCEITOS

A UML é a linguagem padrão para especificar, visualizar, documentar e construir artefatos de um sistema e pode ser utilizada com todos os processos ao longo do ciclo de desenvolvimento e através de diferentes tecnologias de implementação.

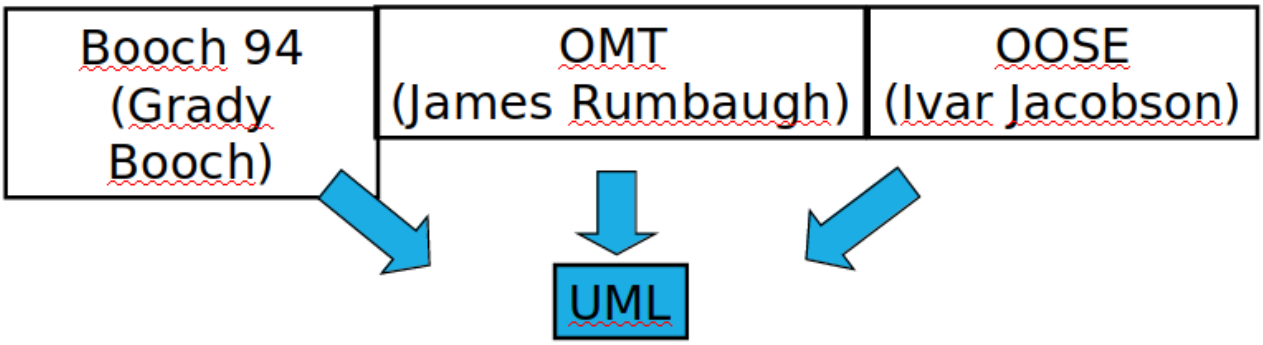
UML oferece uma forma padrão de se desenhar as "plantas" (como em arquitetura) de um sistema de forma a incluir aspectos abstratos (processos de negócio, funcionalidades do sistema) aspectos concretos (classes escritas em determinada linguagem de programação, esquemas de bancos de dados, componentes de software reutilizáveis)

HISTÓRICO

As linguagens de modelagem orientadas a objetos surgiram entre a metade da década de 1970 e o final da década de 1980.

O pessoal envolvido com metodologia, diante de um novo gênero de linguagens de programação orientadas a objeto e de aplicações cada vez mais complexas, começou a experimentar métodos alternativos de análise e projeto.

Na metade da década de 1990, Grady Booch (Rational Software Corporation), Ivar Jacobson (Objectory) e James Rumbaugh (General Electrics) criadores de métodos orientados a objetos, começaram a pegar as melhores idéias e partiram para a criação de uma linguagem unificada de modelagem.



USOS UML

Visualização

Comunicação de modelos conceituais e alguns aspectos de um sistema que não são deduzidos apenas na leitura do código.

Especificação

Construir modelos: precisos, não-ambíguos e completos.

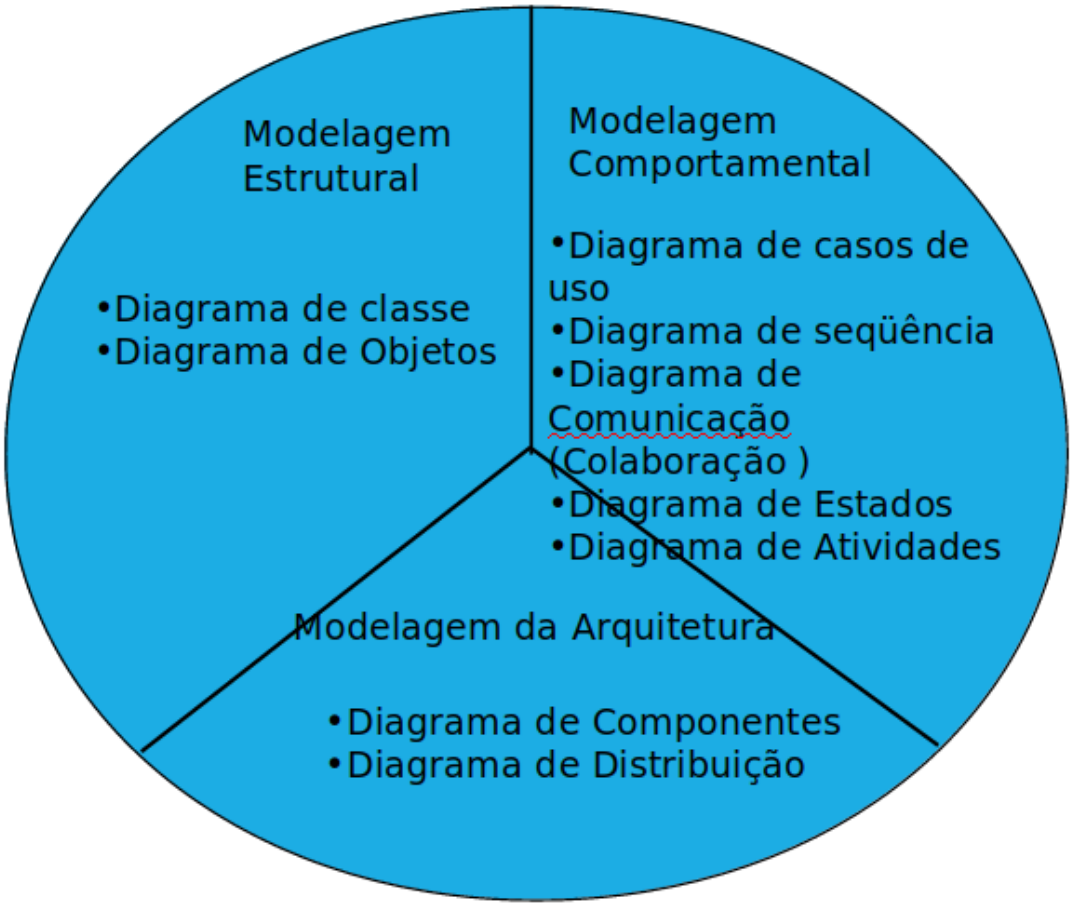
Construção

Não é uma linguagem de programação, mas seus modelos podem ser ligados diretamente a uma.

Documentação

Arquitetura e todos os detalhes documentados, definição de requisitos e testes

PRINCIPAIS DIAGRAMAS



CASOS DE USO

Descrições narrativas de processos do domínio da aplicação

Documentam a seqüência de eventos de um ator (um agente externo) usando o sistema para completar, do início ao fim, um determinado processo

Representação em UML:

Exemplo de um caso de uso de alto-nível:

Caso de uso:	**Comprar Itens**
Atores:	Cliente, Operador
Descrição:	Um Cliente chega no caixa com itens para comprar. O Operador registra os itens e coleta o pagamento. Ao final, o Cliente sai com os

A UML não especifica um formato rígido para os cabeçalhos e a estrutura de um caso de uso
Podem ser alterados de acordo com as necessidades de documentação

Exemplo de um caso de uso expandido:

Caso de uso:	**Comprar Itens com Dinheiro**
Atores:	Cliente (Iniciador), Operador
Propósito:	Capturar uma venda e seu pagamento em dinheiro.
Descrição:	Um Cliente chega no caixa com itens para comprar. O Operador registra os itens e coleta um pagamento com dinheiro. Ao final, o Cliente sai com os itens.
Referencia:	*Funções*: R1.1, R1.2, R1.3, R1.7, R1.9, R2.1

Típica Seqüência de Eventos

Ação do Ator	Resposta do Sistema
1. Este caso de uso começa quando um Cliente chega no caixa com itens para comprar.	

Exemplo de um caso de uso expandido (cont.):

Típica Seqüência de Eventos Ação do Ator	Resposta do Sistema
2. O Operador registra o identificador de cada item. Se há mais de um do mesmo item, o Operador também pode informar a quantidade.	**3.** Determina o preço do item e adiciona informação sobre o item à transação de venda em andamento. Mostra a descrição e o preço do item corrente.
4. Após processar o último item, o Operador indica ao sistema que a entrada de itens terminou.	**5.** Calcula e mostra o valor total da venda.
6. O Operador informa o total ao Cliente.	

Exemplo de um caso de uso expandido (cont.):

Típica Seqüência de Eventos Ação do Ator	Resposta do Sistema
7. O Cliente entrega um pagamento em dinheiro, possivelmente maior do que o valor total.	
8. O Operador registra o valor recebido em dinheiro.	**9.** Mostra o troco devido. Emite um recibo.
10. O Operador deposita o dinheiro e retira o troco devido. O Operador entrega o troco e o recibo ao Cliente.	**11.** Registra a venda no *log* de vendas completadas.
12. O Cliente sai com os itens comprados.	

Exemplo de um caso de uso expandido (cont.):

Seqüências Alternativas

Linha 2: Registro de um identificador inválido. Mostrar erro.

Linha 7: Cliente não tem dinheiro suficiente. Cancelar transação.

Se complexas, seqüências alternativas podem ser expandidas para formar os seus próprios casos de uso

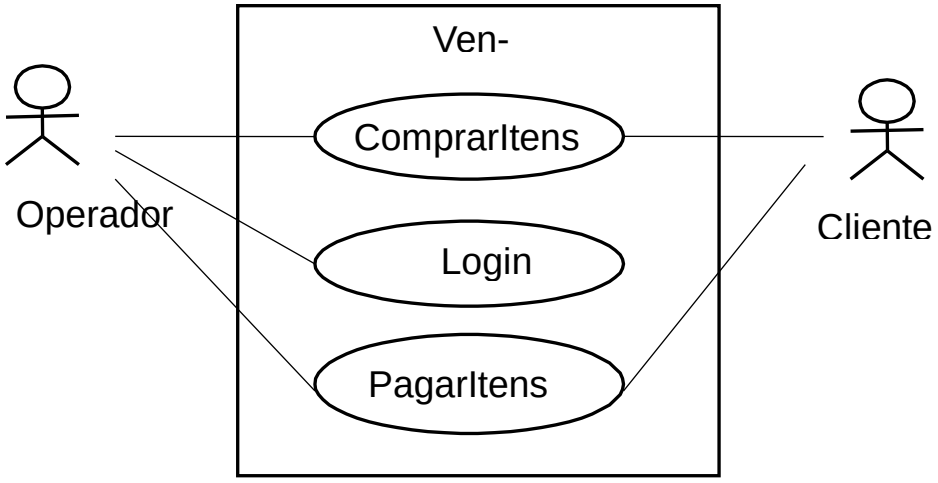
ATORES

Entidades externas ao sistema que de algum modo participam da estória do caso de uso
Estimulam o sistema com eventos de entrada, ou recebem alguma coisa dele
Designados pelo papel que exercem no caso de uso
Ex.: Cliente, Operador, etc.

Representação em UML:

DIAGRAMAS DE CASO DE USO

Ilustram um conjunto de casos de uso e atores para um sistema e os relacionamentos entre eles
Diagrama parcial para o sistema Venda



ORGANIZAÇÃO DE CASOS DE USO

Pacotes

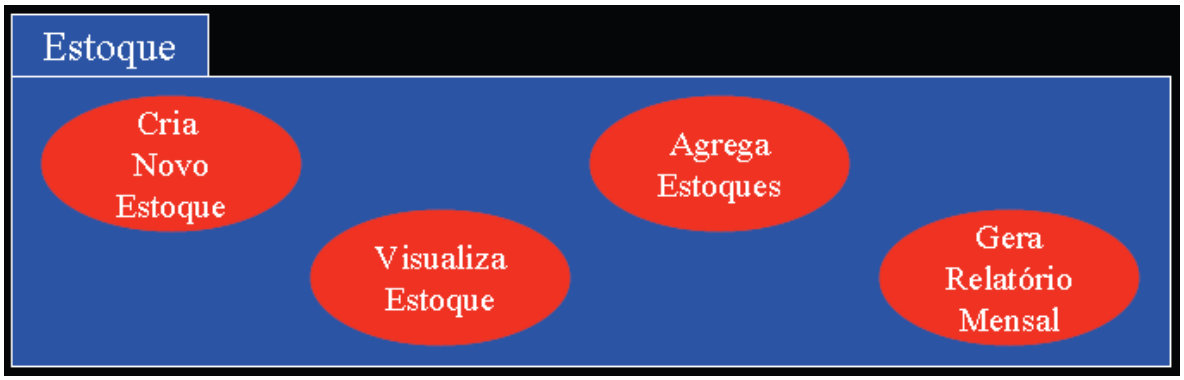
Generalização

Inclusão

Extensão

PACOTES DE CASOS DE USO

Pode ser útil para distribuir trabalho para subgrupos de trabalho.



GENERALIZAÇÃO

Análoga à generalização/especialização de classes.

INCLUSÃO

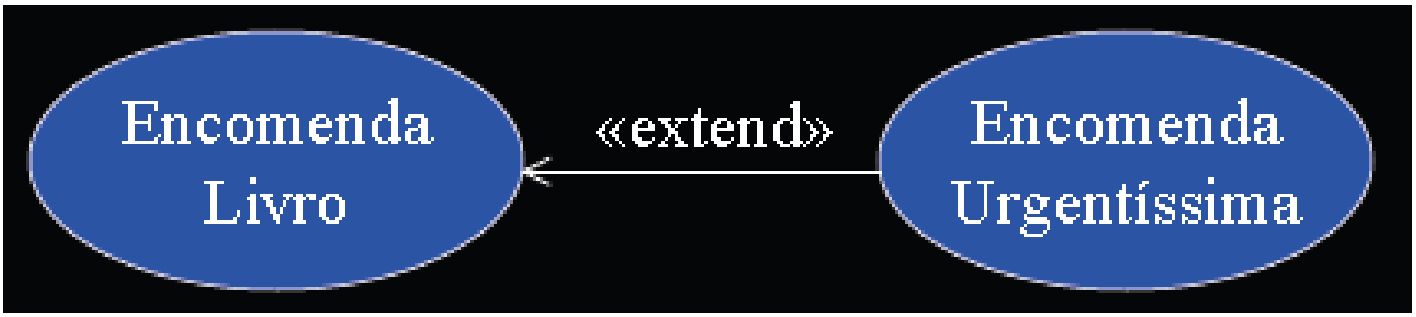
O estereótipo «include» indica que um caso inclui o outro.

Permite fatorar comportamento comum a vários casos.

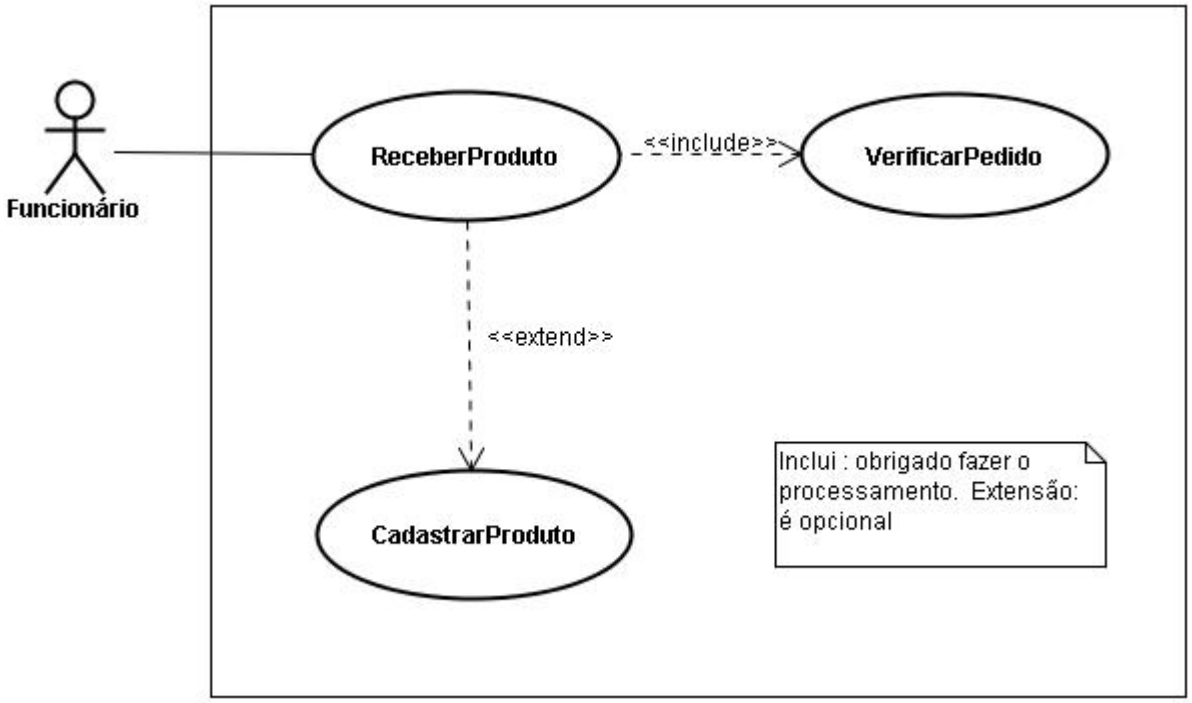
EXTENSÃO

Pode-se usar o estereótipo «extend» para indicar que um caso estende o outro.

Útil para fatorar comportamento incomum/não-padrão.



EXEMPLO



DIAGRAMAS DE INTERAÇÃO

Um *diagrama de interação* ilustra as interações de mensagens entre instâncias (e classes) no modelo de classes

A UML defines dois tipos (equivalentes) de diagramas de interação:

Diagramas de comunicação (Colaboração)

Diagramas de seqüência de eventos

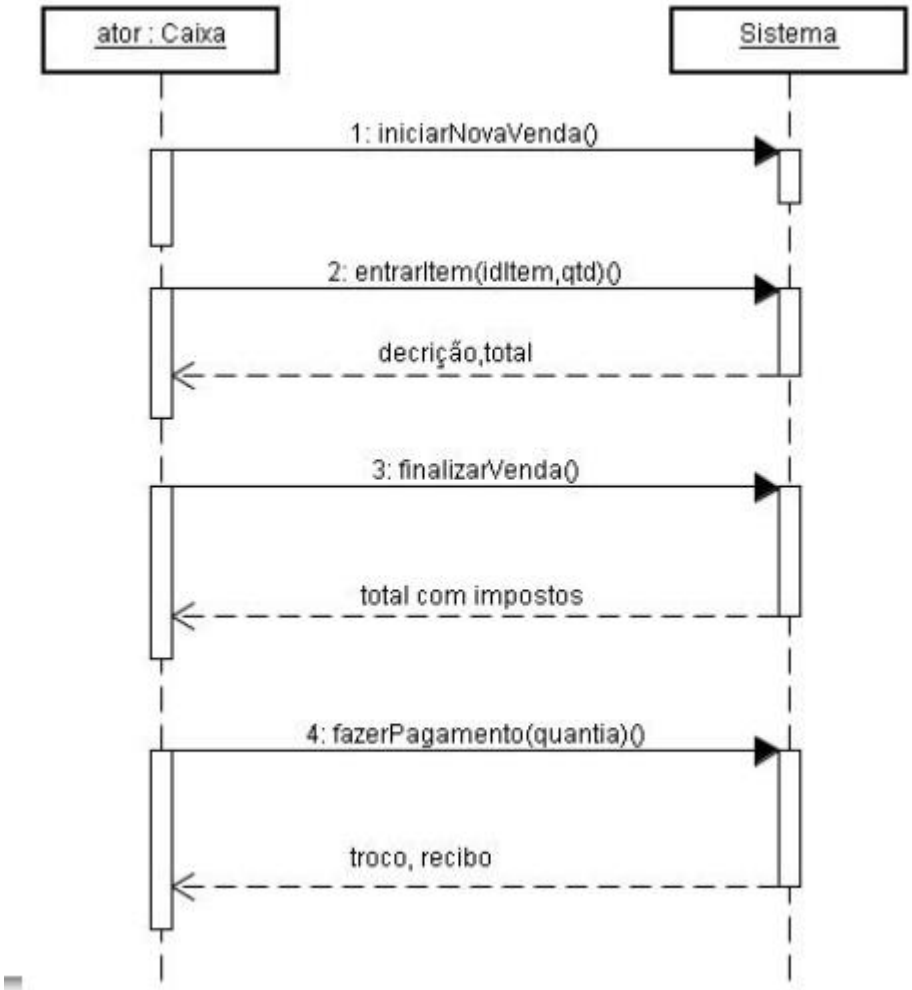
DIAGRAMAS DE SEQÜÊNCIA

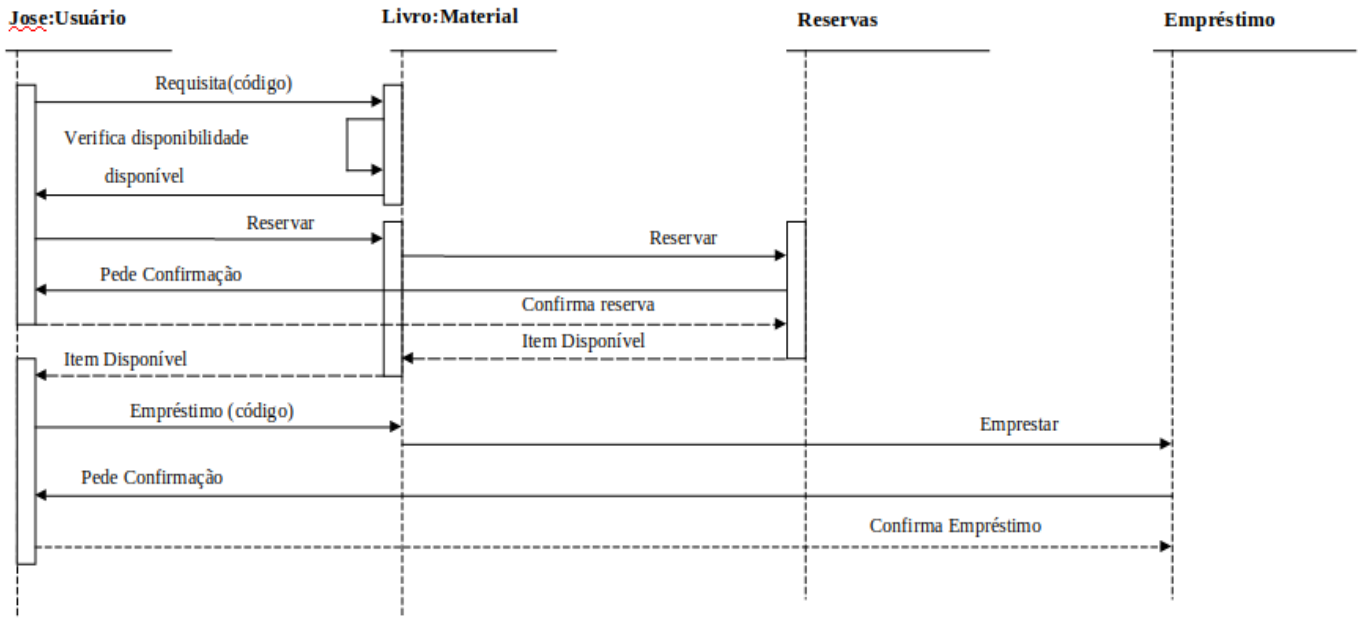
Um *diagrama de seqüência* ilustra a ordem das interações dos atores externos com o sistema e os eventos que eles geram.

Os casos de uso, como vimos, representam um conjunto de cenários que descrevem os diferentes processos que ocorrem no sistema.

O diagrama de seqüência de eventos permite modelar estes processos através da troca de mensagens (eventos) entre os objetos do sistema.

Os objetos são representados por linhas verticais e as mensagens como setas que partem do objeto que invoca um outro objeto.


ator : Caixa
Sistema
1: iniciarNovaVenda()
2: entrarItem(idItem,qtd)()
decrição,total
3: finalizarVenda()
total com impostos
4: fazerPagamento(quantia)()
troco, recibo



DIAGRAMAS DE CO MUNICAÇÃO

Os diagramas de comunicação possuem essencialmente, a mesma informação que um diagrama de seqüência de eventos, mas que é apresentada de uma outra forma.

Este diagrama também mostra as mensagens sendo trocadas entre as classes, mas agora em um diagrama onde são apresentados os relacionamentos entre as classes, servindo de caminho para as mensagens.

Os diagramas de comunicação também servem para descrever os cenários identificados pelos casos de uso, e podem ser traçados em conjunto com o diagrama de classes.

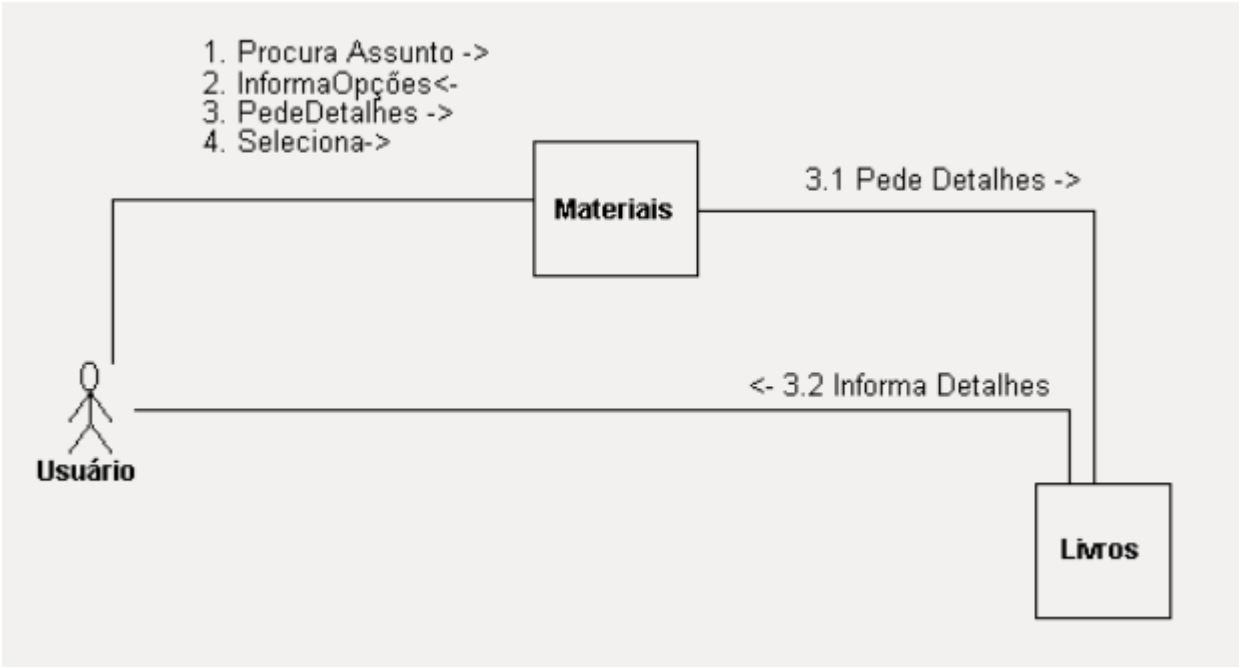


DIAGRAMA DE ESTADOS

Os diagramas de transição de estados mostram a dinâmica interna de uma classe. Apenas os eventos e estados de uma única classe são apresentados neste diagrama.

Uma classe pode ter vários estados, caracterizados por situações em que a classe se encontra.

O diagrama de estados pode possuir ainda estados especiais como o estado inicial e o estado final e outros estados de controle internos.

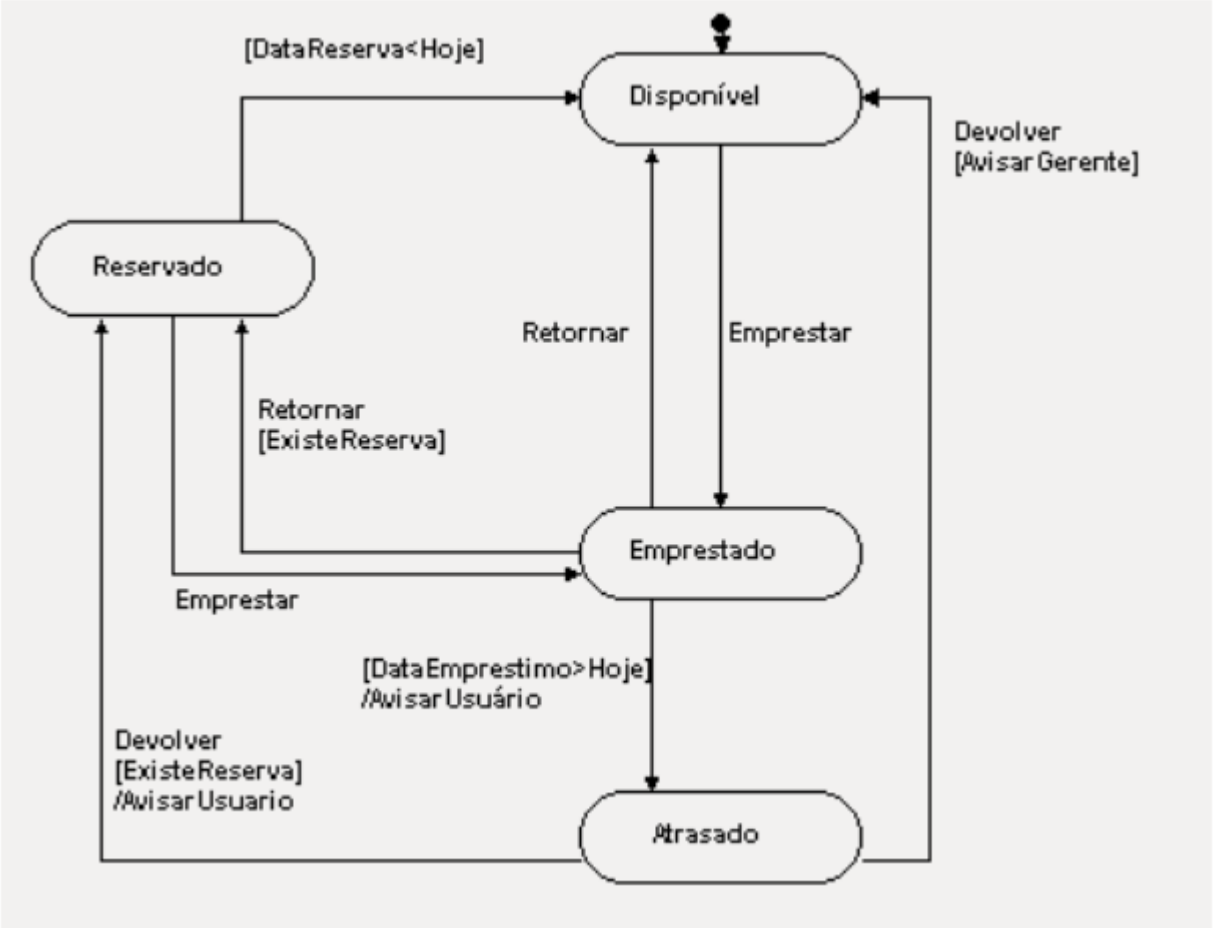


DIAGRAMA DE ATIVIDADES

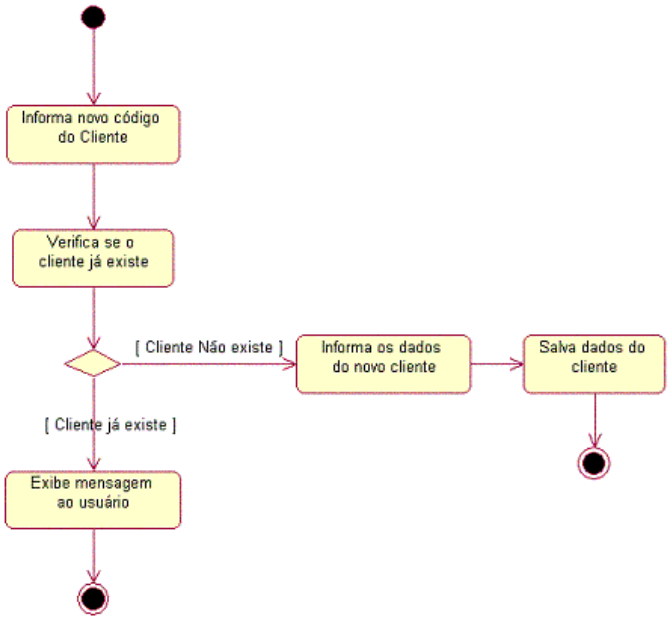
É um tipo específico de Diagrama de Estados

• Útil para modelar fluxo de trabalho.

• Representa as atividades que afetam o estado do sistema e os fluxos que levam de uma atividade a outra.

• É usado para modelar processos de negócio.

EXEMPLO DE DIAGRAMA DE ATIVIDADE



DIAGRAMAS DE CLASSE

Um *diagrama de classe* ilustra as especificações de software para as classes e interfaces do sistema

Inclui:

Classes, associações e atributos

Interfaces (com operações e constantes)

Métodos

Informação sobre o tipo dos atributos

Navegabilidade

Dependências

Os diagramas de classe podem exibir nas fases iniciais da análise apenas o nome das classes, e em uma fase seguinte os atributos e operações.

Finalmente, em uma fase avançada do projeto pode exibir os tipos dos atributos, a visibilidade, a multiplicidade das relações e diversas restrições.

O diagrama de classes, ao final do processo de modelagem, pode ser traduzido em uma estrutura de código que servirá de base para a implementação dos sistemas.

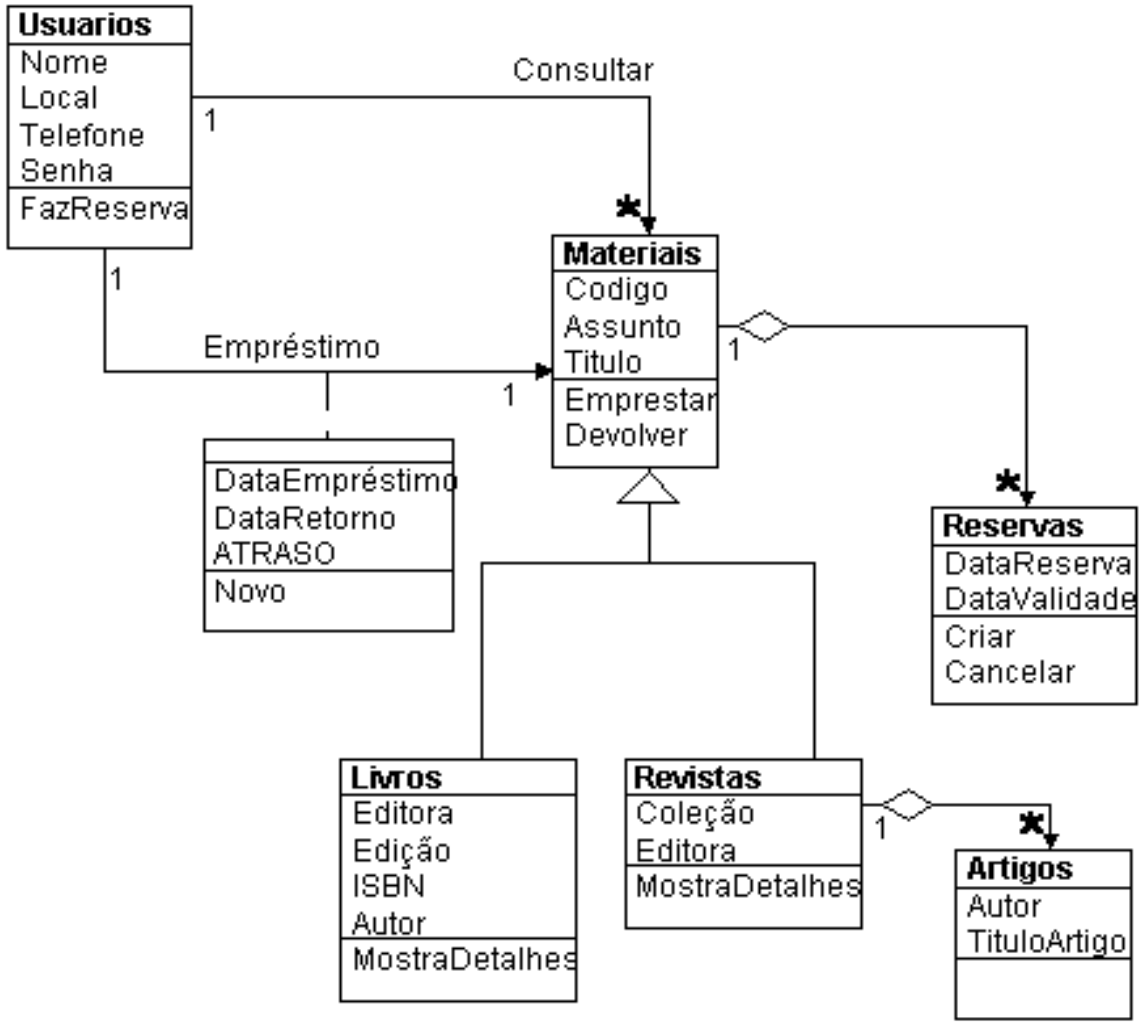


DIAGRAMA DE COMPONENTES

Os diagramas de componentes mostram os elementos reutilizáveis de software e sua interdependência.

Um componente é formado por um conjunto de classes que se encontram implementadas nele. Um componente, assim como as classes que ele possui, dependem funcionalmente das classes de outro componente.

O diagrama de componentes mostra esta dependência.

No diagrama de componentes também é possível mostrar a configuração de um sistema de software mostrando, graficamente, a dependência entre os diversos arquivos que compõem o sistema.

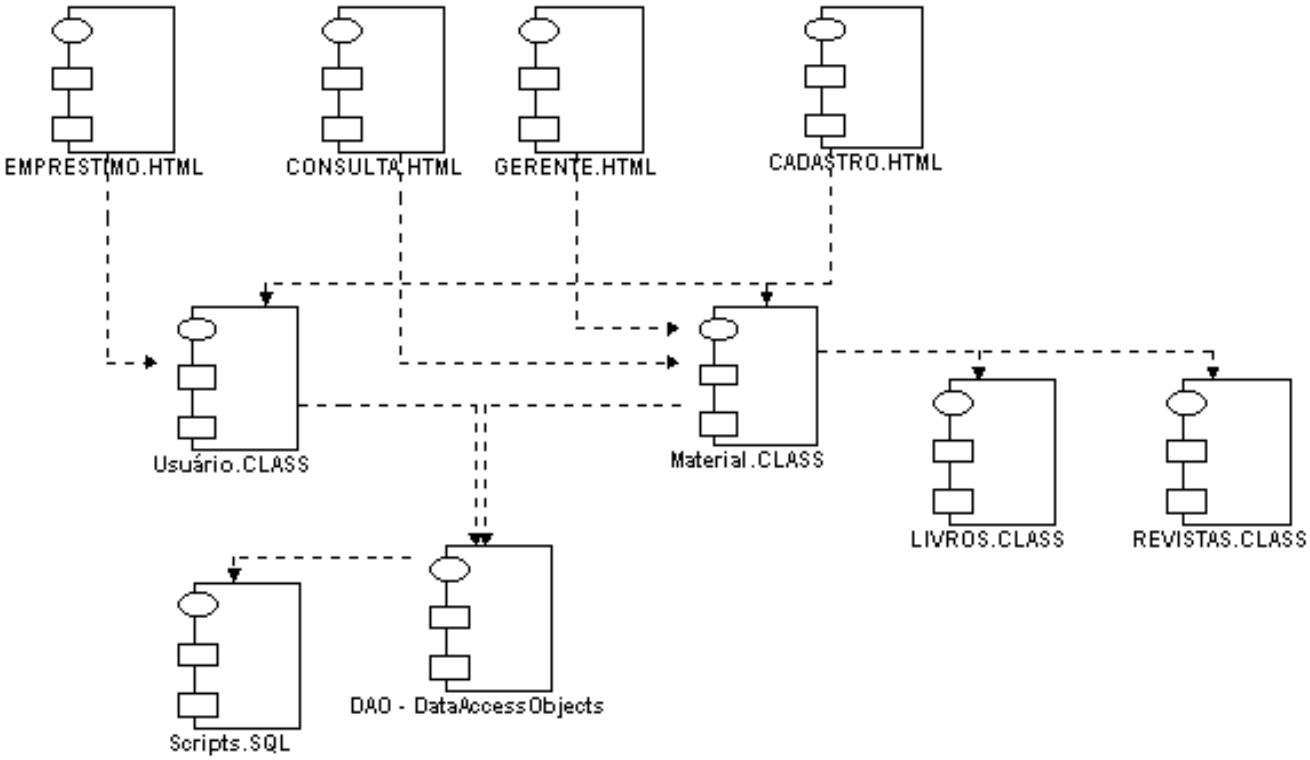
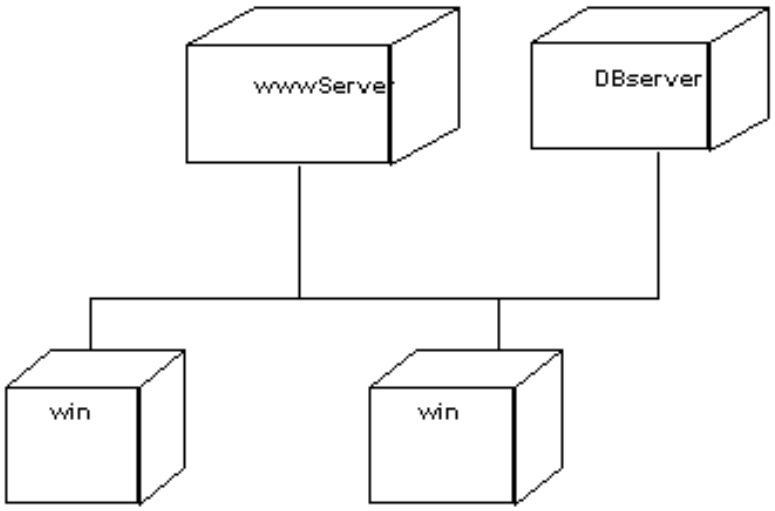
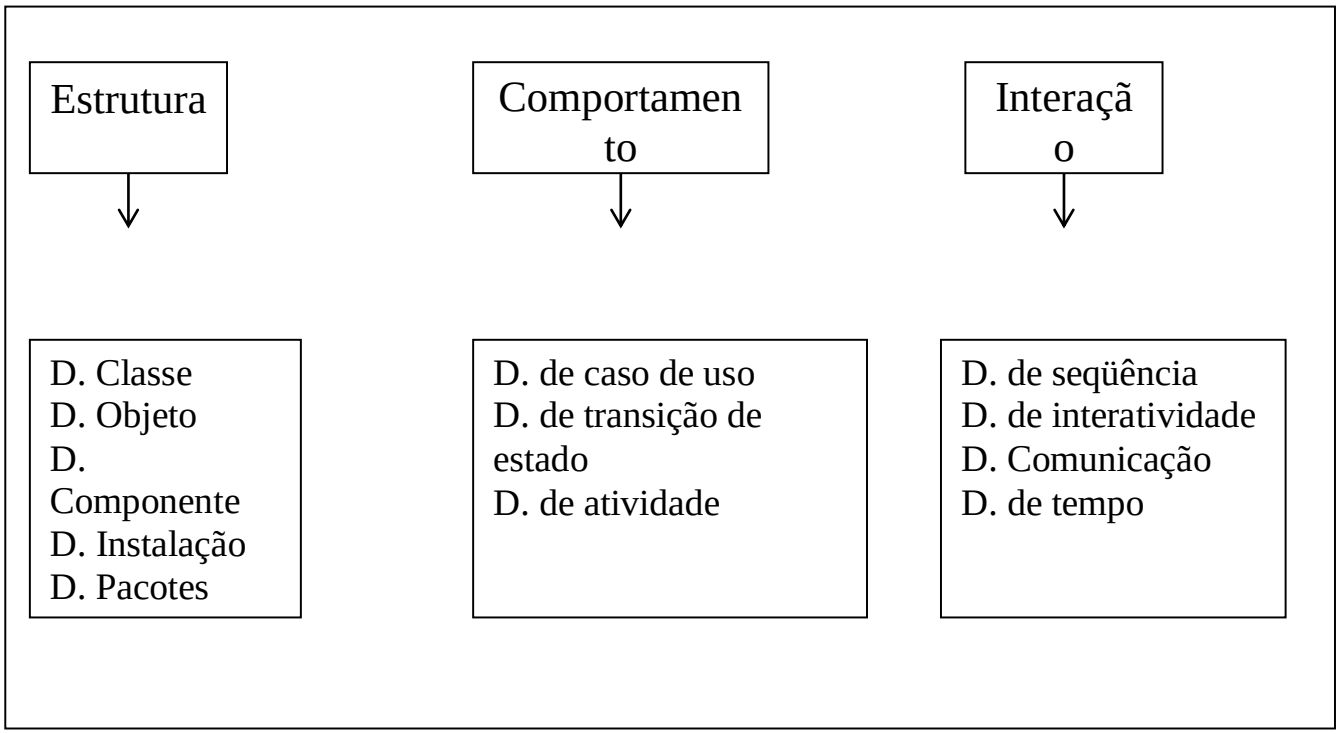


DIAGRAMA DE DISTRIBUIÇÃO

Os diagramas de distribuição mostram a distribuição de hardware do sistema, identificando os servidores como nós do diagrama e a rede que relaciona os nós. Os componentes de software vão estar mapeados nestes nós.



UML 2.0



O QUE MUDA?

Acrescenta mais 4 diagramas:

Interatividade - são variações de "*diagrama de atividades*". Nele, sequências formam um fluxo de atividades, mostrando como elas trabalham em uma sequência de eventos.

Tempo - apresenta o comportamento dos objetos e sua interação em uma escala de tempo, focalizando as condições que mudam no decorrer desse período.

Estrutura - destina-se a descrição dos relacionamentos entre os elementos. Utilizado para descrever a colaboração interna de classes, interfaces ou componentes para especificar uma funcionalidade

Pacotes - ou diagrama de módulos, definido pela UML descreve os pacotes ou pedaços do sistema divididos em agrupamentos lógicos mostrando as dependências entre estes, ou seja, pacotes podem depender de outros pacotes.

Análises Estruturada x Essencial

Técnica da Análise Essencial

Uma de suas propostas fundamentais é usar os eventos como base para o particionamento dos sistemas;

A técnica da Análise Essencial pode ser designada como uma evolução bem sucedida da Análise Estruturada. Segundo E. Yourdon é a Análise Estruturada Moderna.

Análise Estruturada: Ênfase na perspectiva das funções;

Modelagem de Dados: Interesse centrado na Análise de Dados.

Obs.: Controle e comportamento do sistema em relação ao tempo não obteve atenção necessária.

Especificação de Sistema

Por qual perspectiva se deveria começar a especificação de um sistema?

Pelos dados?

Pelas funções?

Deve-se começar a especificação de um sistema pela identificação dos **eventos** que o afetam.

Abstração

É o processo através do qual detalhes são ignorados para nos concentramos nas características essenciais;

Representa os objetivos de acordo com o ponto de vista e interesse de quem representa.

Obs.: descrevendo-se um automóvel, do ponto de vista de um observador externo, identifica-se a cor,o número de portas, o tipo das rodas e pneus. Quando identifica-se o automóvel apenas a partir destas características externas estar-se-á fazendo uma **abstração** pois uma série de detalhes **não** estão sendo descritos.

Vantagens da Análise Essencial em Relação à Análise Estruturada

Análise Essencial:

Inicia-se pelo Modelo Essencial, o que equivale, na Análise Estruturada, a partir diretamente do modelo lógico proposto;

aborda três perspectivas: funções, dados e controles.

Análise Estruturada:

Aborda duas perspectivas do sistema: funções e dados.

A **Análise Estruturada** propõe um particionamento através da abordagem *top-down;*

A **Análise Essencial** propõe outra forma de particionamento, baseada nos eventos facilitando a identificação das funções e entidades que compõem o sistema;

A **Análise Essencial** permite a construção dos modelos de dados e de funções concomitantemente, garantindo correspondência entre os dois modelos.

Decomposição da Modelagem Proposta pela Análise Essencial

Níveis

Essencial (Modelo Essencial)

Grau de abstração totalmente **independente** de restrições tecnológicas;

Dedica-se ao conhecimento da essência do sistema, sem preocupar-se com a implementação (se manual ou automatizada) e nem que tipo de HW ou SW será utilizado;

Corresponde ao **modelo lógico** da Análise Estruturada.

Implementação (Modelo de Implementação)

Grau de abstração totalmente **dependente** de restrições tecnológicas;

Deriva-se do Modelo Essencial;

Diz respeito à implementação do sistema preocupar-se com a implementação (se manual ou automatizada) e nem que tipo de HW ou SW será utilizado;

Modelo Essencial

Indica o que o sistema deve fazer, mencionando o mínimo possível sobre como o sistema será implementado;

Pressupõe a existência da tecnologia perfeita e que esta pode ser obtida a custo zero.

Divide-se em dois modelos:

Modelo Ambiental;

Modelo Comportamental.

Modelo Ambiental: volta-se para fora do sistema, para o ambiente em que está inserido. Representa a interface do sistema com o mundo exterior. Mostra a interação do sistema com os elementos externos a ele.

Modelo Comportamental: volta-se para dentro do sistema, para o comportamento de suas partes internas. Mostra como ele deve reagir internamente a estímulos do ambiente externo.

Modelo Comportamental

Descreve o comportamento, do interior do sistema, necessário para interagir com o ambiente com sucesso.

Componentes:
Diagrama de Fluxo de Dados (DFD);
Diagrama de Entidade Relacionamento (DER);
Diagrama de Transição e Estado (DTE);
Dicionário de Dados;
Miniespecificações.

Modelo Ambiental

O sistema possui dois pontos de vista. São eles:

Ponto de vista interno: é um conjunto de elementos completamente inter-relacionados de modo a constituir um todo organizado;
Ponto de vista externo: é um todo organizado, que se relaciona com o meio ambiente e responde dinamicamente a estímulos vindos de seu exterior.

Modelo Ambiental *versus* Relevâncias na Especificação

Onde irá operar?

Quais necessidades irá atender?

O que é parte integrante do sistema?

O que não é parte integrante do sistema?

Com quem interage?

Quais são as entradas?

De onde vêm as entradas?

Quais são as saídas?

Para onde vão as saídas?

Quais são as finalidades que o sistema deve atender?

Quais estímulos deve reagir?

Modelo Ambiental *versus* Relevâncias

Deve ser aderente às necessidades dos usuários e, portanto, deve ser expresso de uma forma fácil de ser compreendida por eles, para que possa ser entendido por uma equipe multidisciplinar;

Os termos usados na descrição devem ser os mais familiares possíveis aos usuários;

Deve-se evitar todo e qualquer jargão técnico de informática.

Componentes do Modelo Ambiental

Declaração dos objetivos do sistema;

Lista dos eventos que afetam o sistema;

Diagrama de contexto do sistema.

Declaração dos Objetivos do Sistema

Antes de pensar em desenvolver qualquer sistema algumas indagações são necessárias tais como:

Qual é a finalidade do sistema?

A que ele se propõe?

Que problemas ele deverá resolver?

Que requisitos devem ser atendidos?

O que muda com a sua implantação?

Algum outro sistema será por ele substituído?

Obs.: as respostas a essas perguntas não podem incluir recursos tecnológicos; restringi-se apenas à essência do problema e não pode descer a detalhes que digam respeito a opções de tecnologia.

Deve concentrar-se no "que" o sistema faz e não em "como' o faz.

Declaração textual, breve e direta dos objetivos do sistema;

Não pretende detalhar toda a funcionalidade do sistema, apenas relatar, resumidamente, o que se espera usufruindo do sistema;

Pode conter os benefícios tangíveis e quantificáveis que serão obtidos com o sistema;

Volta-se para a alta gerência e outros que não estejam diretamente envolvidos com o desenvolvimento do sistema.

Exemplificando a Declaração dos Objetivos do Sistema

O sistema SAC (Sistema de Atendimento ao Cliente) propõe-se a diminuir o tempo médio de atendimento ao cliente;

O sistema de gestão de estoque é garantir que a quantidade de produtos no armazém esteja sempre entre os valores máximo e mínimo estipulados.

Lista de Eventos do Sistema

Os eventos constituem a parte fundamental de um sistema;

As **finalidades** do sistema são atender a determinadas **necessidades**. Estas necessidades são decorrentes de eventos que ocorrem no mundo exterior ao sistema.

Análise de Eventos

Um sistema pode ser visto como uma "caixa preta" que, a partir de certos estímulos externos predeterminados , produz respostas esperadas.

Para cada função a ser executada por um sistema tem de haver um estímulo responsável pela sua ativação.

Para descobrimos as funções de um sistema, primeiro deve-se descobrir os estímulos que chegam ao sistema.

Para cada estímulo a chegar no sistema deve haver a ocorrência de um evento no mundo externo ao sistema.

O evento é quem provoca o estímulo a quem cada função deve reagir.

Cada ocorrência de um estímulo provoca mudança de estado do sistema.

Análise Essencial

Propõe o particionamento do sistema por **eventos**;

O sistema deve responder com eficácia a todos os **estímulos** a que for submetido;

A cada estímulo, o sistema deve reagir produzindo uma **resposta** predeterminada.

Evento: Informalmente, é um acontecimento do mundo exterior que requer do sistema uma resposta;

Obs.: o evento deve ser representado por uma frase que expresse um acontecimento (ex.: " Aluno se matricula na disciplina" é o evento; " Matrícula-do-aluno" é o fluxo).

Estímulo: É um ativador de funções. É a forma como o evento age sobre o sistema;

Resposta: É o resultado gerado pelo sistema devido à ocorrência de um evento;

Evento Externo: É um acontecimento independente que ocorre fora do sistema e provoca um estímulo que faz com que a função seja executada dentro do sistema;

Obs.:Todo evento é do tipo EXTERNO.

Vale Ressaltar

Um fluxo de dados é sempre representado por um substantivo, enquanto o evento é sempre representado por uma frase que expressa um acontecimento.

Ex.: "Professor corrigi a avaliação do aluno" é o evento ao passo que "correção da avaliação" é um fluxo de dados.

Evento não é o estímulo. Os eventos são a causa do surgimento dos estímulos no sistema

Ex.: "Professor corrigi a avaliação do aluno" é o evento ao passo que "correção da avaliação" é um fluxo de dados.

Classificação dos Eventos

São classificados de acordo com o tipo de estímulo que eles provocam no sistema.

Evento Orientado por Fluxo de Dados (F);

Evento Orientado por Controle (C);

Evento Orientado por Tempo (Evento Temporal) (T).

Evento orientado por fluxo de dados

É aquele em que o estímulo é a chegada ao sistema de um fluxo de dados enviado por uma **entidade externa**;

Obs.: Nem todo fluxo de dados que chega ao sistema serve de estímulo relativo a um evento. Pode ser apenas uma informação complementar à execução de uma função.

Ex.: Uma função encarregada de cadastrar os fornecedores toda vez que chega um pedido de cadastramento.

O **evento** é o " pedido de cadastramento" feito pela entidade externa Fornecedores;

O **estímulo** é " o pedido enviado".

Evento Orientado por controle

É aquele em que o estímulo é a chegada ao sistema de um fluxo de controle;

Gera um fluxo de controle que pode ser enviado por uma **entidade externa** ou gerado por uma **função interna** do sistema;

Só tem dois valores possíveis (binário);

Ex.: diretoria autoriza o pagamento de uma fatura.

O **evento** é " autorização de pagamento" pela entidade externa Diretoria;

O **estímulo** é o " sim, pagamento autorizado".

Evento orientado por tempo

É aquele em que o estímulo é a chegada ao sistema da informação de tempo decorrido;

A ação é executada a partir de uma função interna do sistema;

Ex.: É hora de emitir relatório mensal de vendas. Hoje é o último dia do mês.

O **evento** é "identificação do último dia do mês";

O **estímulo** é " sim, último dia do mês atingido".

Como Desenhar um Diagrama de Contexto

Desenhar um único processo ou função para representar o sistema inteiro;

Desenhar todas as entidades externas que se comunicam com o sistema;

Para cada entidade externa, desenhar o fluxo de dados que mostra sua comunicação com o sistema.

Lista de eventos em forma tabular

Nº do Evento	Nome do Evento	Tipo de Evento	Estímulo	Ações	Respostas
(1)	Cliente entrega pedido	(F)	Pedido	Registrar pedido	*(Pedido registrado)*
(2)	Cliente cancela pedido	(F)	Pedido de cancelamento	Cancelar pedido	*(Pedido cancelado)*
(3)	Cliente envia pagamento	(F)	Cheque de pagamento	Emitir recibo de pagamento	Recibo de pagamento *(Fatura paga)*
(4)	Fornecedor solicita cadastramento	(F)	Pedido de cadastramento de fornecedor	Cadastrar fornecedor	*(Fornecedor cadastrado)*
(5)	É hora de emitir relatório de vendas	(T)	*(a hora de emitir relatório de vendas)*	Emitir relatório de vendas	Relatório de vendas
(6)	É hora de verificar pedidos em atraso	(T)	*(a hora de verificar pedidos em atraso)*	Verificar pedidos em atraso	Pedidos em atraso
(7)	Direção autoriza pagamento de fornecedor	(C)	Autorização de pagamento	Gerar pagamento	Pagamento de fornecedor *(duplicata paga)*
(8)	2º cheque sem fundos é emitido	(C)	*(informação de que é o 2º cheque sem fundos)*	Registrar cheque no Serviço de Proteção ao Crédito)	Relatório p/ o Serviço de Proteção ao Crédito SPC *(Cheque no SPC)*
(9)	Nível de ressuprimento é atingido	(C)	*(informação de que foi atingido o nível de ressuprimento)*	Emitir encomenda ao fornecedor	Encomenda de material ao fornecedor *(encomenda)*

Diagrama de Contexto do Sistema

Mostra de forma gráfica toda a interação entre sistema e mundo externo;

Seus componentes são:

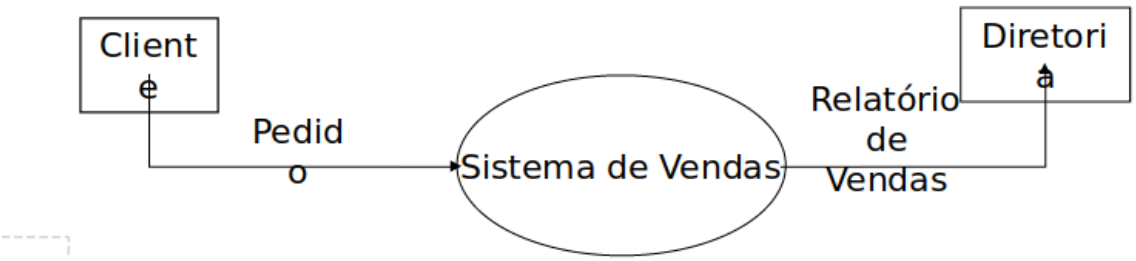
Processos ou funções;

Entidades externas;

Fluxo de dados quer sejam de entrada e saída.

Representação do Diagrama de Contexto

Nº do evento	Nome do evento	Tipo de Evento	Estímulo	Ações	Respostas
(1)	Cliente faz pedido	(F)	Pedido realizado	Registrar pedido	*(Pedido registrado)*
(2)	É hora de emitir relatório de vendas	(T)	———	Emitir relatório de vendas	Relatório de vendas



Relevâncias do Diagrama de Contexto

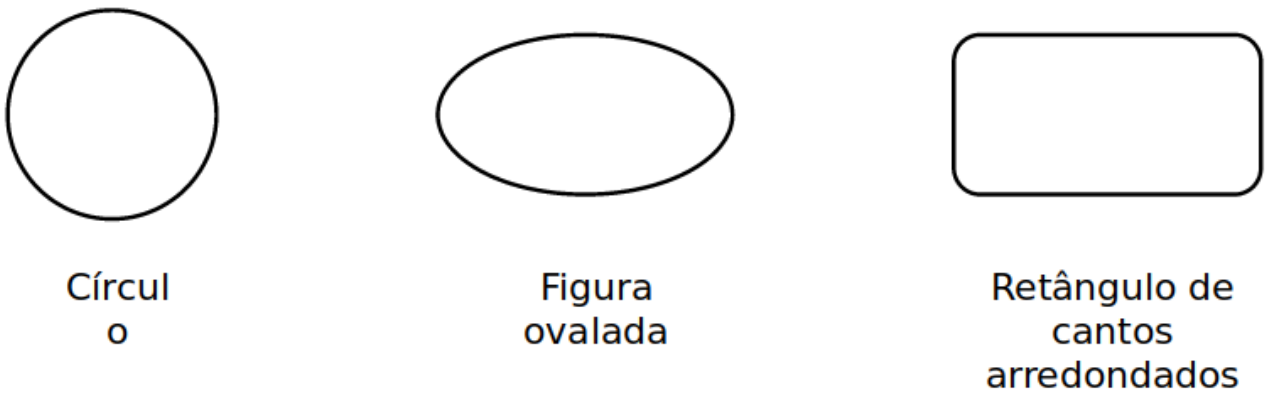
Quando o evento fornece uma **resposta** do tipo **interna** não há fluxo de saída no sistema;

Quando o **evento** é do tipo **temporal** não há fluxo de entrada no sistema e sim só o fluxo de saída haja visto, não haver estímulo;

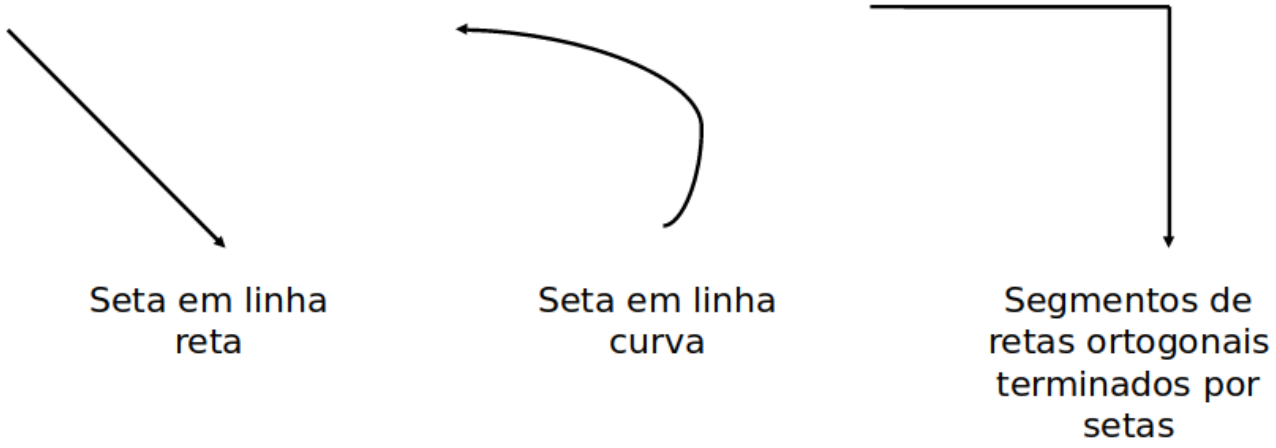
Não pode haver fluxo de dados partindo diretamente de uma Entidade Externa em direção a outra;

Não devem aparecer depósitos de dados.

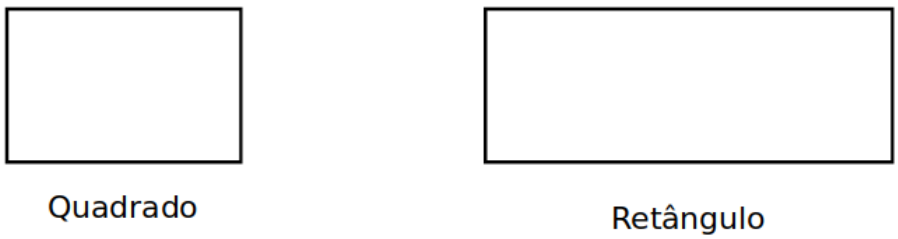
Função ou Processo



Fluxo de Dados



Entidade Externa



Obs.: Para duplicação de entidades externas, costuma-se acrescentar ao símbolo tantas linhas diagonais quantas forem as repetições utilizadas.

Lista de Eventos

Evento nº 01: A secretaria cadastra os períodos letivos;

Evento nº 02: A secretaria cadastra novos cursos;

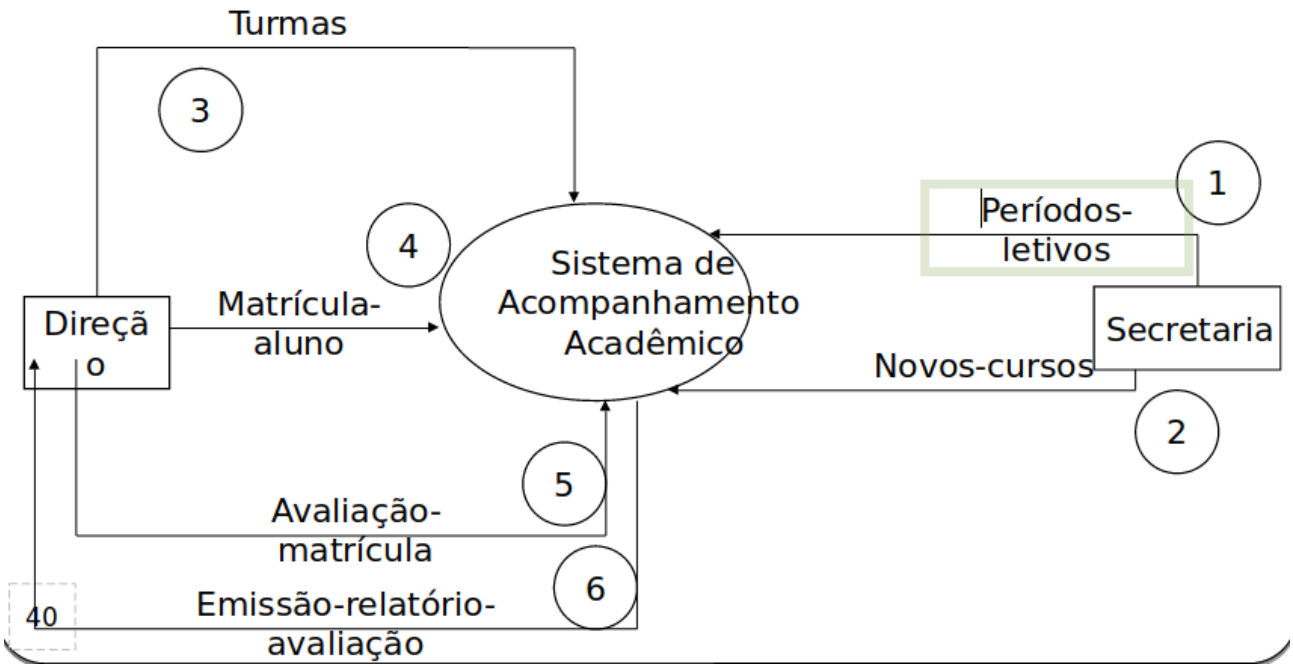
Evento nº 03: A direção da escola cadastra turmas;

Evento nº 04: A direção da escola efetua a matrícula do aluno numa turma;

Evento nº 05: A direção da escola efetua a avaliação da matrícula;

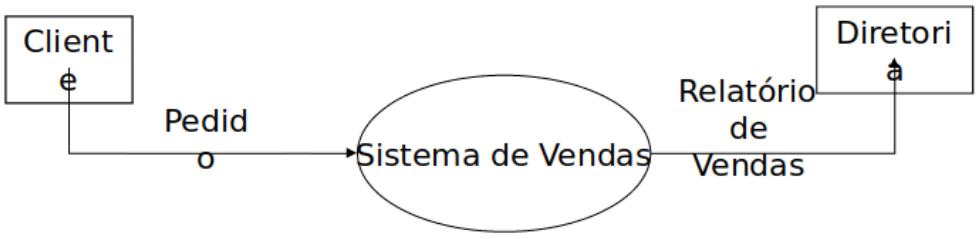
Evento nº 06: É hora de emitir relatório de avaliação para a direção da escola.

Diagrama de Contexto do Sistema



Representação do Diagrama de Contexto

Nº do evento	Nome do evento	Tipo de Evento	Estímulo	Ações	Respostas
(1)	Cliente faz pedido	(F)	Pedido realizado	Registrar pedido	*(Pedido registrado)*
(2)	É hora de emitir relatório de vendas	(T)	———	Emitir relatório de vendas	Relatório de vendas



Modelo Essencial

Modelo Ambiental: Definido do ponto de vista *externo*, descreve o sistema visto pelo lado de fora, no melhor estilo de um mecanismo do tipo estímulo-resposta, mostra o que faz e o que não faz parte do sistema,

preocupa-se em delimitar fronteiras, qual é o universo de interesse, qual é o domínio das mudanças que são desejadas para o novo sistema em desenvolvimento.

Modelo Comportamental: Definido do ponto de vista *interno*, é o modelo do interior do sistema. Descreve de que maneira o sistema, enquanto um conjunto de elementos inter-relacionados, reage, internamente, como um todo organizado, aos estímulos do exterior. Preocupa-se em mostrar quais as ações que o sistema deve executar para responder adequadamente aos eventos previstos no modelo ambiental, que é seu ponto de partida.

Modelo Comportamental

Descreve o comportamento, do interior do sistema, necessário para interagir com o ambiente com sucesso.

Componentes:

Diagrama de Fluxo de Dados (DFD);

Diagrama de Entidade Relacionamento (DER);

Dicionário de Dados;

Diagrama de Transição e Estado (DTE);

Miniespecificações.

Quando pensa-se em decompor um sistema, pensa-se em dois tipos de componentes:

Funções e Dados.

Anteriormente é importante saber:

O que é produzido pelo sistema?

A que estímulos o sistema deve responder?

Obs.: os dados armazenados e as funções são meios para atingir-se o verdadeiro objetivo do sistema, que é apresentar as respostas adequadas ao ambiente em que está contido. Sendo assim, a decomposição deve ser feita a partir dos eventos.

Diagrama de Fluxo de Dados (DFD)

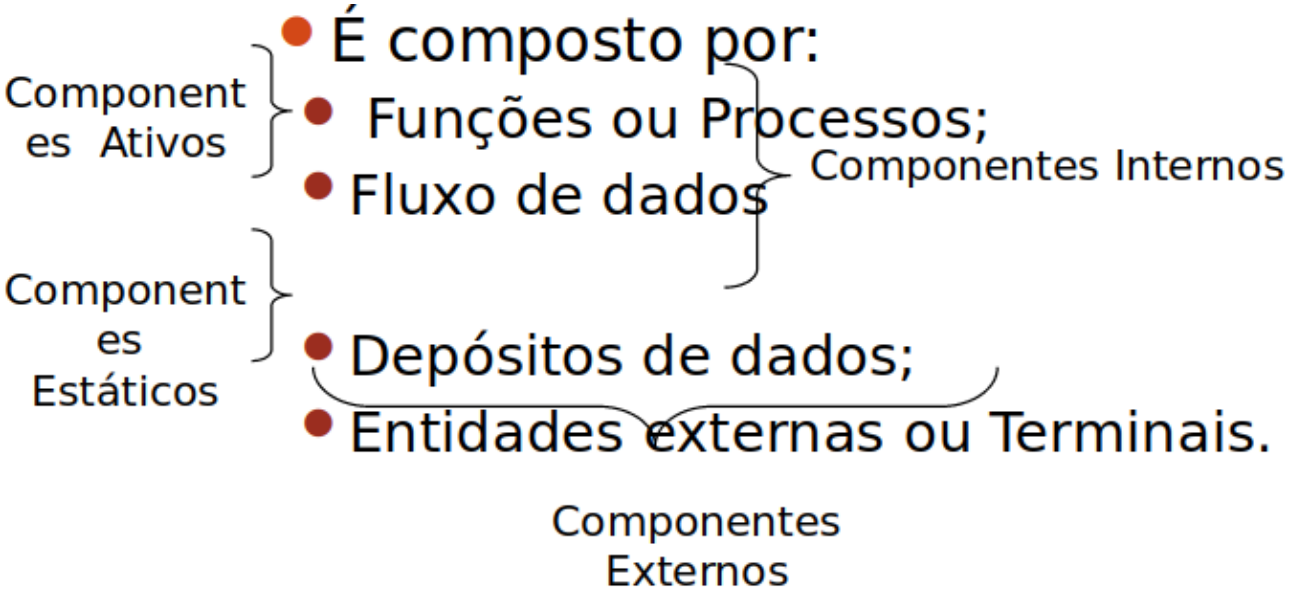
É uma forma gráfica de mostrar a interdependência das **funções** que compõem um sistema, apresentando **fluxos de dados** entre elas;

Ferramenta utilizada na modelagem comportamental e funcional.

Mostra ainda:

Arquivos lógicos de dados chamados de **Depósito de dados**;

Entidades externas ou **Terminais** tanto à origem dos fluxos de dados que chegam ao sistema, como ao destino dos fluxos que dele partem.



Função

Pode-se comparar ao conceito de **caixa-preta** onde:

Há ligação de entrada e de saída da caixa;

Conhecem-se os elementos de entrada da caixa;

Conhecem-se os elementos de saída da caixa;

Sabe-se o que a caixa realiza (*o que* a caixa faz para que, a partir dos elementos de entrada, sejam produzidos os elementos de saída;

Não precisa conhecer *como* a caixa realiza suas operações e nem em que ordem.

Pode ser entendida como um componente de um sistema onde somente os dados de entrada e os dados de saída são conhecidos;

Não se conhece explicitamente nada a respeito do processo interno de transformação dos dados de entrada em dados de saída;

Representam as ações que o sistema executa;

Nem todas as funções são automatizadas;

Pode-se usar a palavra função como sinônimo da palavra processo.

O Fluxo de Dados

São condutos que levam informação de um ponto do sistema para outro;

Mostram como os dados fluem através do sistema, o caminho percorrido pelos dados no sistema;

O fluxo de dados liga:

Uma entidade externa a uma função;

Uma função a uma entidade externa;

Uma função a um depósito de dados;
Um depósito de dados a uma função;
Uma função a outra função.

Os Depósitos de Dados

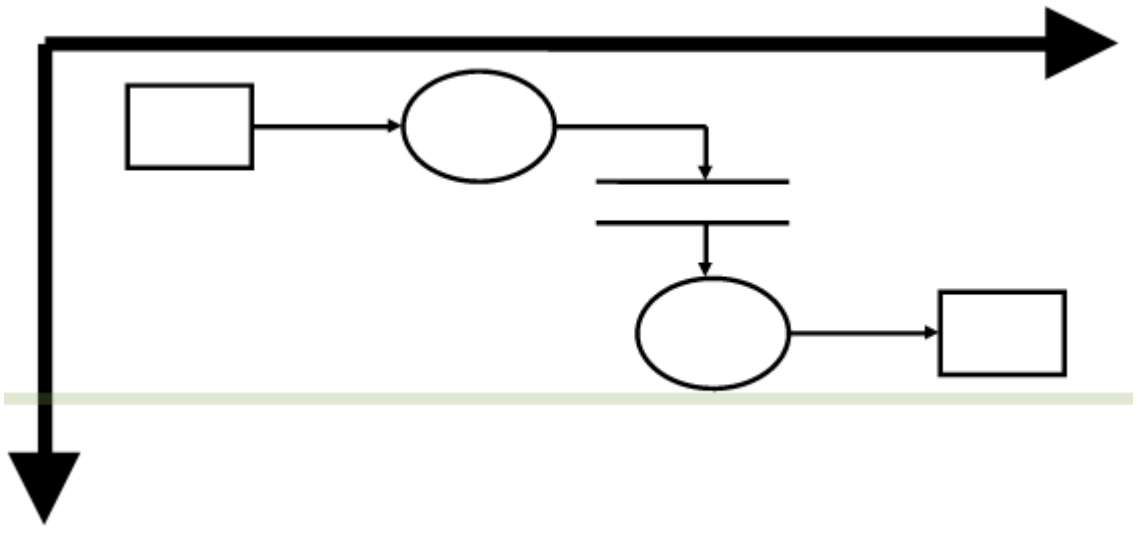
É um conjunto de dados armazenados;
Constituem a memória do sistema;
Representa-se os dados em estado de repouso;
Notação de representação:
Duas retas paralelas
Um retângulo aberto do lado direito

As Entidades Externas ou Terminais

Ambiente de um sistema pode ser entendido como o meio externo ou em torno do sistema;
Delimita-se pelos elementos externos que exercem influência sobre o comportamento do sistema;
São as fontes ou destinos dos fluxos de dados que chegam e saem do sistema;
Mostram as interfaces do sistema com o ambiente em que ele está inserido.

Regras para a Construção de DFDs

Ver slides anteriores para a notação da:
Função ou processo;
Fluxo de dados;
Entidade externa ou terminal;
Depósito de dados.
Obs.: quando for inevitável o cruzamento de linhas (fluxos), um artifício que pode ser usado é o de desenhar um arco.
Orientação geral do layout de um DFD:
O fluxo geral dos dados deve seguir uma orientação de cima para baixo e da esquerda para a direita;



Orientação geral do layout de um DFD:

As entidades externas devem ficar, preferencialmente, nas bordas do desenho;

Os depósitos de dados devem estar posicionados:

Se forem de entrada: à esquerda ou acima da função que recebe seus dados;

Se forem de saída: à direita ou abaixo da função que lhe fornece dados.

Os processos podem estar ligados:

Diretamente, através de fluxo de dados;

Indiretamente, com um depósito de dados entre eles.

Obs.: o processo de origem deve ficar à esquerda ou acima do processo de destino.

DFD de Resposta ao Evento

Nº do evento	Nome do evento	Tipo de Evento	Estímulo	Ações	Respostas
(1)	Cliente faz pedido	(F)	Pedido realizado	Registrar pedido	*(Pedido registrado)*

Observações:

- A ação do evento é o processo do DFD;
- O estímulo do evento é a entrada do DFD;
- A resposta interna do evento vai para um depósito de dados do DFD;
- A resposta externa do evento vai para uma entidade externa do DFD;
- Quando a seta do fluxo sai do depósito para a bolha representa uma leitura de dados;
- Quando a seta do fluxo sai da bolha para o depósito representa uma gravação, atualização ou exclusão de dados.

Modelagem de Dados

É um método de análise de sistemas que busca especificar, a partir dos dados relevantes, que estejam associados ao domínio de conhecimento analisado, a perspectiva dos dados, permitindo organizá-los em estruturas bem definidas, estabelecer regras de dependência entre eles, produzindo um modelo expresso por uma representação, ao mesmo tempo descritiva e diagramática.

Relevância dos Dados

Há uma pesquisa dos mesmos associados a um sistema de informação;

Tipos de dados e propriedades são definidos construindo um modelo de dados;

Defini-se os relacionamentos entre os tipos de dados.

Quando faz-se os fatos acima citados adequadamente, com envolvimento do usuário, os dados necessários ao contexto analisado são encontrados e definidos.

Níveis de Abstração

Modelo conceitual dos dados;

Modelo lógico dos dados;

Modelo interno dos dados;

Modelo externo dos dados.

Modelo Conceitual dos Dados

Chamado **nível conceitual**, que procura espelhar a realidade, independentemente de restrições de implementação, segundo uma visão global, integrada, do sistema sob análise;

Nível independente dos recursos de HW e SW existentes no ambiente sob análise;

É constituído de uma coleção de entidades que representa os conceitos permanentes ao contexto do sistema e seus relacionamentos;

A descrição é denominada de **esquema conceitual**;

Chama-se, abreviadamente, de **modelo de dados**.

Modelo Lógico dos Dados

Chamado **nível lógico**, voltado para as características lógicas, ou seja, deve atender as necessidades da aplicação;

Ocorre a implementação parcial do BD. Nesta etapa não será definida as características específicas dos atributos. Ex.: Tipo de dado e tamanho;

Nível de descrição do BD é independente dos recursos de HW existentes no ambiente sob análise;

A descrição é denominada de **esquema lógico**.

Modelo Interno dos Dados

Representa o **nível interno**, também chamado **nível físico** corresponde à forma com que os dados são implementados nos dispositivos de armazenamento existentes no ambiente de instalação do BD, através de vários tipos de registros armazenados, sendo consistente com os requisitos de processamento e procurando o uso mais econômico dos recursos computacionais;

A descrição é denominada de **esquema interno**.

Modelo Externo dos Dados

Representa a visão dos dados que são de interesse de um usuário específico;

Entende-se como sendo um subconjunto do modelo lógico;

A descrição é denominada de **esquema externo.**

Modelo Conceitual de Dados

O valor é em função de sua aderência à realidade do mundo que ele se propõe a representar;

Esta técnica deve procurar capturar os conceitos semânticos do ambiente a ser modelado, bem como expressar-se em uma linguagem, notação, que facilite a percepção do usuário a respeito da organização da realidade que o cerca;

O método deve permitir que se possa estabelecer agrupamentos das entidades do mundo real sob diversas formas de agregação. Ex.: por sistemas, funções, órgãos ou por qualquer outra forma de agregar entidades inter-relacionadas.

Requisitos *versus* Método de Modelagem Conceitual de Dados

Expandir a habilidade do analista no reconhecimento dos requisitos de informação;

Capacidade de capturar ao máximo a semântica do mundo real durante o próprio processo de modelagem;

Facilidade de entendimento, através de uma representação pragmática, que possa, inclusive, ser entendida por usuários não-técnicos de Informática;

Possibilidade de mapeamento, transformação, "direto" do modelo conceitual para o modelo lógico relacional;

Possibilidade de uma abordagem que contemple a modelagem de dados de maneira integrada com a modelagem das funções;

Permitir a derivação do modelo de funções a partir do modelo de dados e vice-versa.

Análise Orientada a Objetos

Introdução

Cenário

Mudança do enfoque das funções para os dados

Preocupação em modelar de forma mais detalhada o sistema

Análise mais próxima da realidade

Facilidade na comunicação com o usuário

Objetos como entidades do mundo real

Objetos com estrutura e comportamento e que se comunicam

Dificuldades em fazer alterações nas estruturas de dados nas abordagens tradicionais

"Se eu alterar a definição desse dado, quais programas serão afetados?"

Trabalha com conceitos já conhecidos - Modularidade, Abstração, Encapsulamento, Mascarar informações, etc

Orientação a objetos apesar de antiga não era utilizada por falta de pessoas treinadas, interesse em manter a cultura adquirida, ferramentas imaturas. Isso começa a se resolver.

O mundo real é composto por objetos. Cada objeto tem propriedades e comportamentos. Então por que não desenvolver programas que simulem no computador os objetos do mundo real com suas propriedades e comportamentos?

Segundo Yourdon, "Um sistema construído usando um método Orientado a Objetos é aquele cujos componentes são partes encapsuladas de dados e funções, que podem herdar atributos e comportamentos de outros componentes da mesma natureza, e cujos componentes comunicam-se entre si por meio de mensagens."

Objetivo

Encontrar os objetos, organizá-los, descrever como interage através de mensagens, definir operações de seus comportamentos.

Estruturada x Essencial x Orientada a Objeto

Nos métodos tradicionais de análise, o comportamento do sistema e seus dados eram considerados separadamente. Com orientação a objetos, comportamento e dados são integrados, assim encapsulando detalhes internos de um objeto dos demais.

Análise	Enfoque
Estruturada e Essencial	Conjunto de programas que executam processos sobre dados
Orientada a	Conjunto de "entidades" que têm características e

Características

Concentra-se nos aspectos essenciais do objeto sem detalhamento, focando em suas características e o que ele faz;

Impede que um sistema se torne tão interdependente que uma pequena alteração ou implementação resulte em grandes alterações em sua estrutura;

Combina estrutura (dados) e comportamento (funções) em um único objeto;

Compartilha elementos estruturais e de comportamento com objetos de níveis inferiores;

Enfatiza a estrutura de objetos ao invés da estrutura de procedimentos, ou seja, o que o objeto é e não como ele é utilizado.

Por que usar Orientação a Objetos?

Atualmente temos ferramentas completas para sua utilização (integrando especificação e implementação)

Praticamente todas as ferramentas novas de programação permitem suporte a sua utilização

Qualidade melhor do software (se usada corretamente)

Produtividade em função do reuso

Produção de códigos mais fáceis de serem entendidos

Adequada para a construção de sistemas distribuídos e para aplicações voltadas a Internet

Permite acesso controlado às informações

Dificuldade

Usuários não pensam seus problemas de forma orientada a objetos

Requisitos não são orientados a objetos

Conceitos

Objeto: Desde a mais tenra idade formamos conceitos. Cada conceito é uma idéia ou um entendimento pessoal que temos de nosso mundo. Os conceitos que adquirimos nos permitem dar um sentido e raciocinar sobre as coisas de nosso mundo. Essas coisas as quais nossos conceitos se aplicam são denominados **objetos**. Um objeto pode ser real ou abstrato, ex.:

Uma fatura, uma organização, um avião...

Classe: é a representação do objetos com seus atributos e métodos.

Atributos → são características que descrevem o objeto.

Ex: Objeto: cão → possui nome, idade, peso,cor dos olhos, comprimento dos pêlos, cor dos pelos, etc.

Métodos ou serviços → as ações que um objeto pode executar.

Ex: latir, comer,sentar, dormir ,etc.

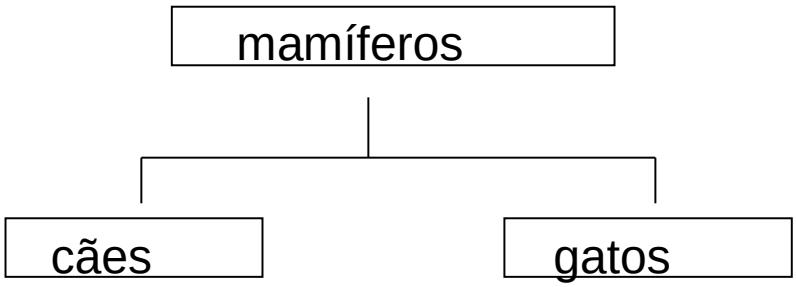
Abstração →significa que, só deve ser representado aquilo que vai ser usado.Pelo princípio da abstração nos objetos são representadas somente as características que são relevantes para o problema em questão

Ex: Cor dos olhos(toda pessoa tem) – não é relevante para um sistema de folha de pagamento.

Encapsulamento →os dados e os processos que tratam esses dados estão " encapsulados" numa única entidade. Os objetos agem como uma "caixa preta", você utiliza sem precisar saber como ele funciona internamente.

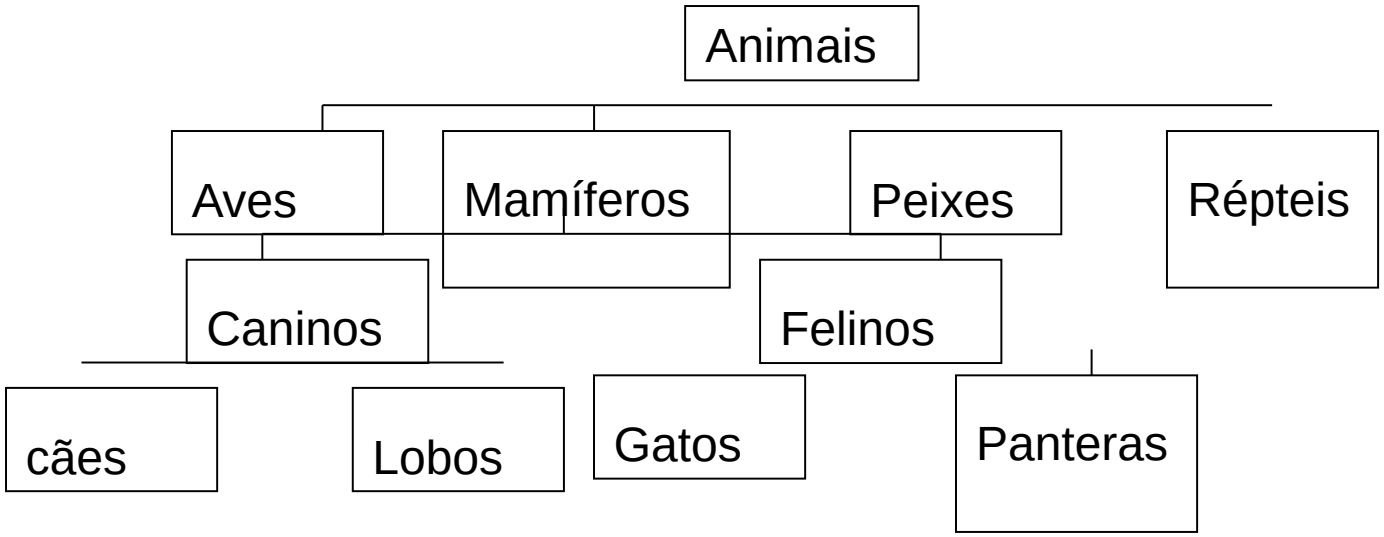
Hierarquia de classes – classe que tem características comuns e que podem fazer parte de uma classe (categoria) maior.

Ex.



Classes ancestrais – classes das quais as outras dependem.

Classes descendentes – as classes originadas a partir de outra.

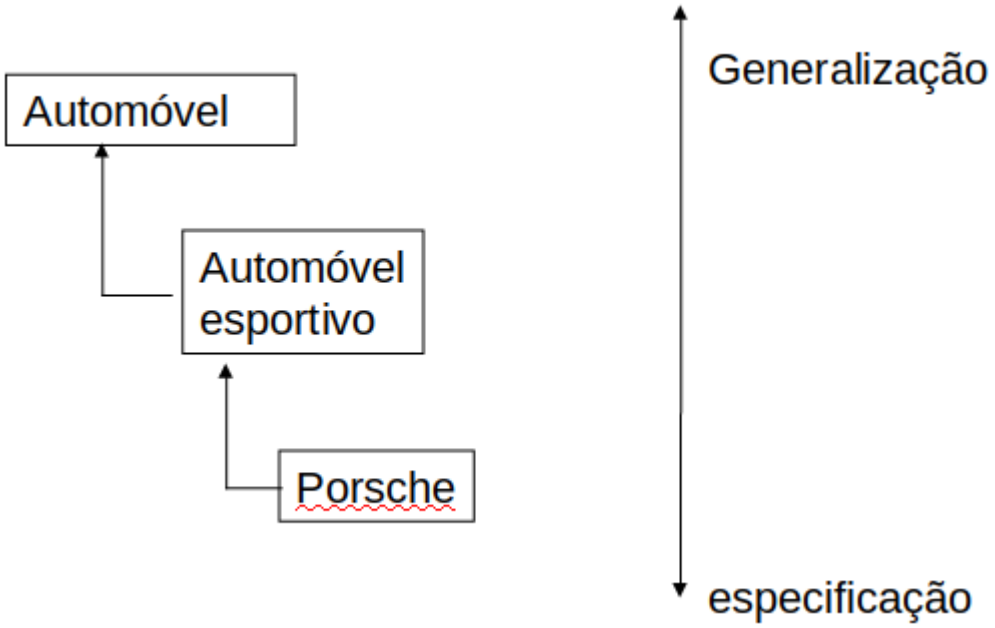


Classes Puras ou abstratas – são classes cujos objetos nunca são instanciados diretamente, mas sempre por uma classe descendente dela.

Herança – significa que todos os atributos e métodos programados no ancestral já estarão automaticamente presentes em seus descendentes sem necessidade de reescrevê-los.

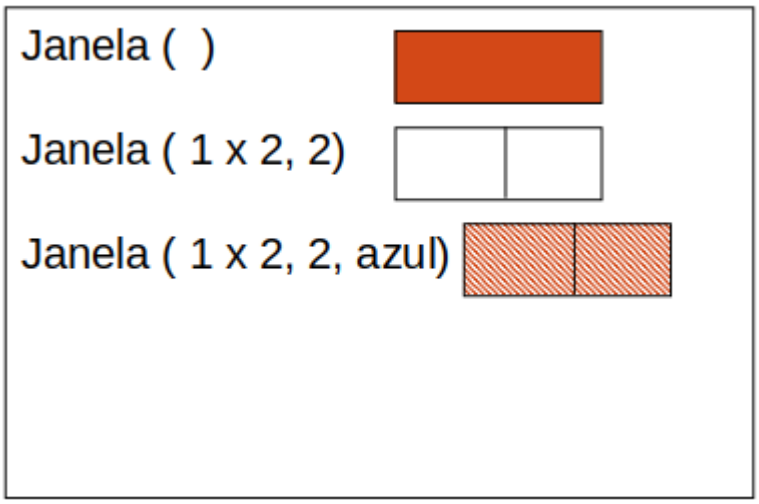
Ex: uma pessoa pode ser um estudante ou um professor,sendo assim todos os atributos e métodos programados na classe pessoa automaticamente passarão para as classes estudante e professor.

Herança



Conceitos

Polimorfismo – é o princípio relacionado com as diferentes formas de um objeto. Operacionalmente, o polimorfismo pode ser visto conforme o exemplo abaixo, onde se pode instanciar o objeto janela de várias formas:

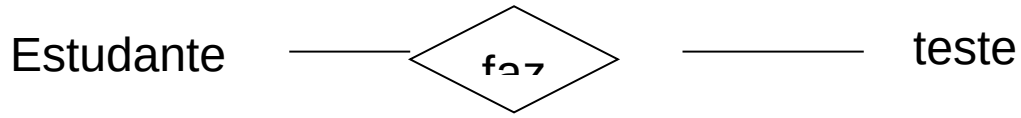


Todo – Parte (Agregação/Composição) –

Agregação : é um mecanismo que permite a construção de uma classe agregada a partir de outras classes componentes. Ex. Casa (classe pai) e tijolos(classe filho)

Composição: a parte não existe sem o todo: Ex: A empresa é composta por funcionários.

Associação – é usada para agrupar certos objetos que ocorrem em algum ponto no tempo ou sob circunstâncias similares. Na AOO , a associação é modelada através de uma conexão de ocorrências. Uma conexão de ocorrência é um relacionamento que um objeto precisa ter com outro(s) objeto (s), para cumprir suas responsabilidades. Ex:



Identificando Classes e Objetos

Os objetos modelam quase todos os aspectos identificáveis do domínio do problema: entidades externas, coisas,ocorrências,papéis, unidades organizacionais, lugares e estruturas; todos podem ser representados como objetos. Os objetos podem ser:

Coisas : elementos que fazem parte do domínio da informação do problema, tais como relatórios, cartas

Papéis: desempenhados por elementos que interagem com o sistema. Ex.: vendedor, cliente,gerente…

Unidades organizacionais: pertencem a organização.Ex: equipe

Lugares: auxiliam na definição do contexto: Ex. piso da fábrica

Estruturas: definem classes relacionadas com o sistema: Ex.: veículos, computadores, sensores….

Identificando objetos

Lembre-se: Objeto é um conceito, uma abstração ou uma coisa, com limites e significados bem definidos, em relação ao problema considerado.

Um objeto é geralmente identificado por um substantivo.

Um objeto contém estrutura e comportamento.

Cada objeto tem sua identidade

Dois objetos são distintos, mesmo que eles apresentem as mesmas características.

Ex: 1 dezena de automóveis Astra

Cada automóvel é um objeto!

Todos tem a mesma característica!

Classes são fábricas de objetos…

Comunicação entre objetos

Mensagens:

Elemento usado para prover a comunicação entre objetos

Na prática, mensagens são implementadas como ativações de uma função definida no objeto chamado, onde:

Nome é o nome da função.

A informação é a lista de parâmetros.

Requisitante é o objeto que realizou a chamada.

As conexões de mensagens são utilizadas apenas quando o projetista do modelo deseja ilustrar uma interação de objetos em um determinado comportamento do sistema. Assim, esse tipo de conexão não é utilizado no projeto do sistema, onde o foco é a modelagem dos objetos, e não seus comportamentos.

Modelagem de um sistema Orientado a objeto

Passos para uma Modelagem OO:

Identificar as classes e os objetos;

Identificar as estruturas e os relacionamentos;

Identificar os atributos e os métodos importantes.

Exemplo de modelagem de um sistema para uma escola

A relação de informações que poderão ser consideradas

neste sistema são:

Um curso pode ser formado por uma ou muitas disciplinas diferentes;

Uma disciplina poderá fazer parte de nenhum ou até muitos cursos;

Cada disciplina deverá ser ministrada por apenas um professor, podendo o professor

ministrar uma ou muitas disciplinas diferentes;

Um curso não poderá ter mais de 40 nem menos de 20 alunos matriculados;

Para cada turma de alunos, deverá haver uma sala de aula;

Um aluno poderá se matricular em nenhum ou até muitos cursos;

Havendo grupos de pesquisa, cada grupo não poderá ter mais de um professor ;

Havendo grupos de pesquisa, cada grupo poderá ter nenhum ou até 2 alunos

bolsistas.

1) Identificar as Classes e os Objetos

Para identificar as classes de um sistema, é importante ter em mente o conceito de que a classe é a representação de um objeto, e que, por sua vez, pode representar qualquer coisa, inclusive um aluno, uma turma, ou uma sala de aula.

Na escola em questão , os alunos fazem seus cursos matriculando-se em uma turma. Os cursos da escola são definidos como um conjunto de disciplinas lecionadas separadamente, e cada disciplina é ministrada por somente um professor.

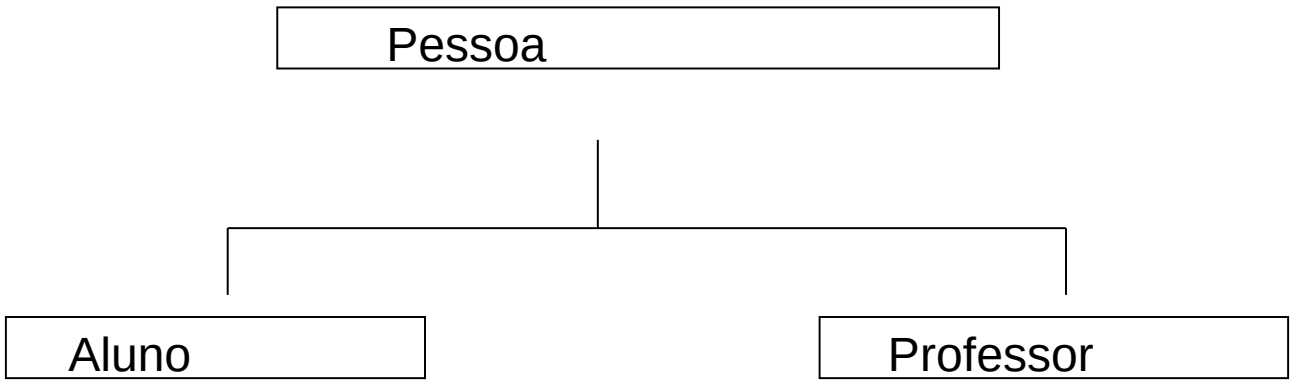
Cont..

Cada turma possui uma sala de aula definida, onde acontecerão as aulas do curso.Diante desse posicionamento , precisamos de objetos como *Curso* e *Disciplina*. Sabendo que os cursos ocorrem em salas de aula, poderemos transformar a sala de aula em objeto do sistema também.

Da mesma forma, outras questões também deverão conduzir a criação de outros objetos como *Turma* e *Matrícula*.

2) Identificar as Estruturas e os relacionamentos

A estrutura mais característica desse nosso exemplo é, sem dúvida, a estrutura tipo Generalização – Especialização (GE).

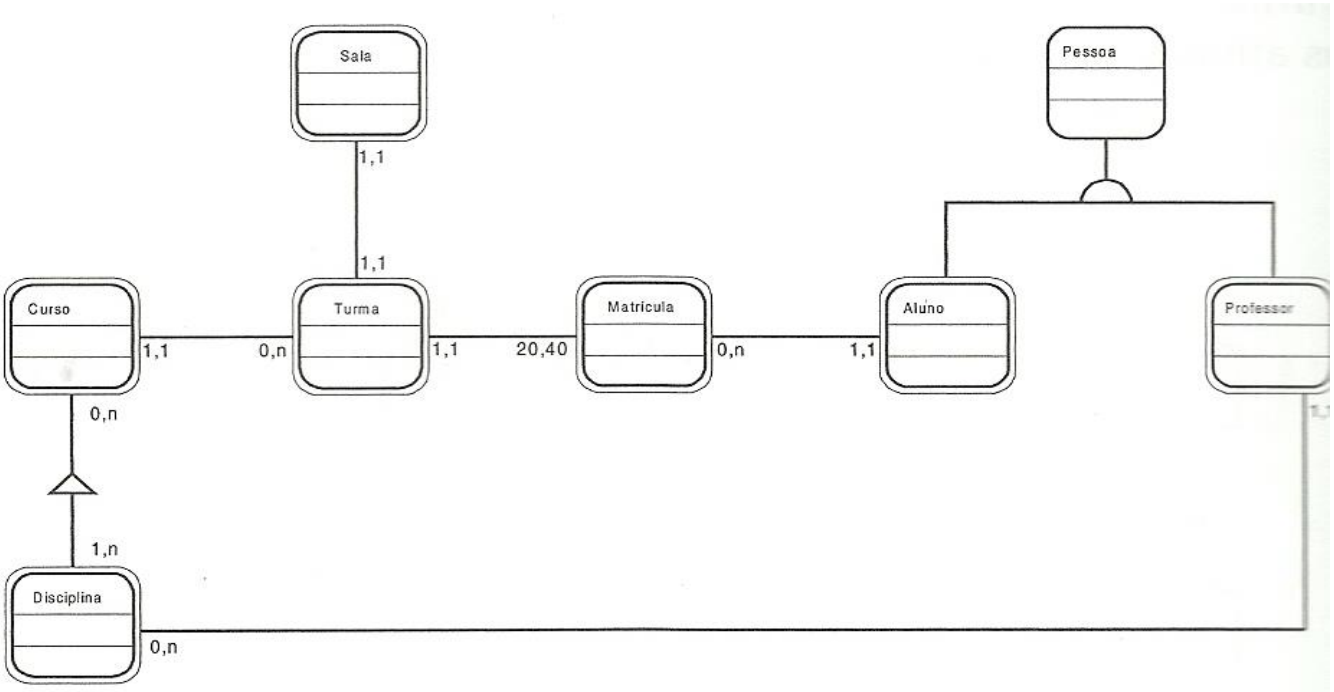


Temos ainda uma estrutura Todo-Parte (TP) entre os objetos *Curso* e *Disciplina*. Cada objeto Curso deverá ser composto por pelo menos um ou até muitos objetos Disciplina, e cada objeto Disciplina poderá ou não fazer parte de um Curso.

Os relacionamentos, também conhecidos como conexões de ocorrências, completam o sistema quando definem as correlações entre os objetos. Um curso poderá definir a existência de nenhuma ou até muitas turmas. Para cada objeto Turma existente, deverá haver uma sala de aula disponível para as atividades de ensino.

As matriculas são associadas as turmas e não aos cursos, pois são as turmas que representam o curso em andamento. Cada turma deverá ter um mínimo de 20 alunos e, no máximo, 40 alunos matriculados.

Modelagem com estrutura e relacionamentos



3) Identificar os atributos e os métodos

Muitos atributos das classes e objetos são identificáveis facilmente já durante a sua concepção, como o atributo *Nome* da classe *Pessoa*.

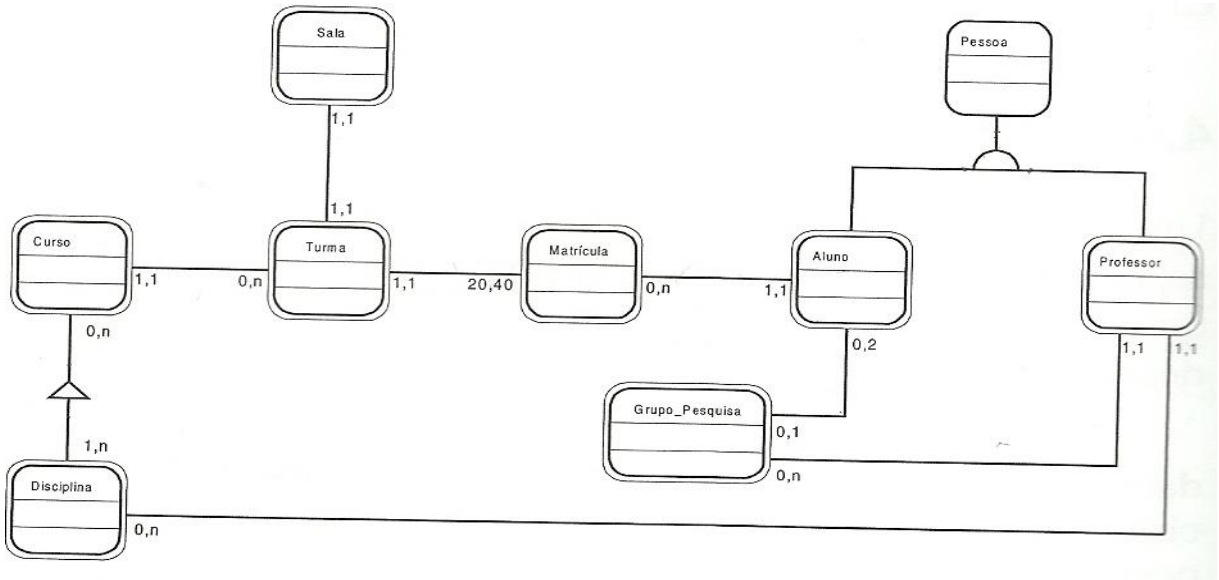
É importante lembrar que devemos ser flexíveis quando estamos construindo um sistema, pois o objetivo que devemos ter em mente é o sistema e a sua análise adaptada à cultura da empresa.

Novos objetos

Um sistema Orientado a Objetos permite que novos objetos sejam agregados ao sistema sem prejudicar os demais objetos que o compõem e, também, sem prejudicar as inter-relações desses objetos.

Um surgimento de um novo objeto chamado Grupo_Pesquisa. Esse novo objeto pode ser composto por um professor e, possivelmente, até 2 alunos.

Modelagem Completa



A análise, o projeto e a programação são atividades distintas

A análise OO se preocupa com a modelagem dos objetos para o domínio da aplicação.

O projeto OO se preocupa com o desenvolvimento de um modelo de sistema que implemente os requisitos definidos pela AOO.

A programação OO se preocupa com a implementação do Projeto OO usando uma linguagem de programação OO.

Análise de Requisitos

O tratamento da informação é um requisito que fundamenta o processo de desenvolvimento de software antes da solução de tecnologia a ser aplicada.

Cada projeto deve ter suas fases de desenvolvimento adequadas às necessidades de tratamento da informação.

CONCEITOS

Requisito é (são):

"Descrições das funções e das restrições de um sistema"

"Definição detalhada, matematicamente formal, de uma função do sistema"

Sommerville p. 82

Requisito é (são):

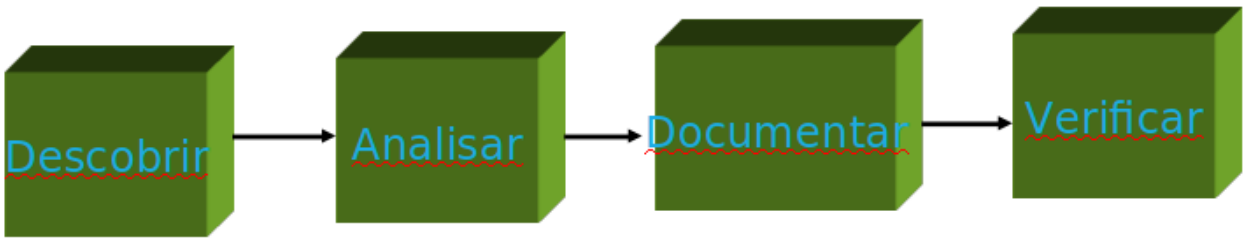
"uma descrição dos principais recursos de um produto de software, seu fluxo de informações, comportamento e atributos. Fornece uma estrutura básica para o desenvolvimento de um produto de software. O grau de compreensibilidade, precisão e rigor da descrição fornecida por um documento de requisitos de software tende a ser diretamente proporcional ao grau de qualidade do produto resultante"

Peters p. 102

Requirements engineering

Engenharia de Requisitos é :

"O processo de (em relação aos requisitos):"



Sommerville p. 82

Engenharia de Requisitos é:

"Estabelecer quais funções são requeridas pelo sistema e as restrições sobre a operação e o desenvolvimento do sistema"

Sommerville p. 46

Engenharia de Requisitos é:

"Um processo que envolve todas as atividades exigidas para criar e manter o **documento de requisitos de sistema**"

Sommerville p. 103

Engenharia de Requisitos objetiva:

Fornecer métodos para compreender a natureza de um problema

Estabelecer com exatidão o que um sistema deve fazer

Sommerville p. 82

Levantamento e Análise de Requisitos

Obtenção de requisitos

1) Entrevistas (Não estruturada, estruturada e semi-estruturada, Focus Group)

2) Observação

3) Questionário

ENTREVISTAS

Faça perguntas sucintas e diretas. Perguntas longas são difíceis de lembrar;

Evite sentenças compostas dividindo-as em duas ou mais perguntas separadas;

Evite utilizar jargões ou linguagens que o entrevistado possa desconhecer, mas acabar não admitindo por vergonha;

Busque a neutralidade nas perguntas;

Solicite que colegas revisem as perguntas;

O entrevistado está prestando um favor, portanto, tente tornar a atividade mais agradável possível e fazer com que ele sinta-se confortável.

REALIZANDO A ENTREVISTA

Os passos seguintes ajudarão a realizar uma boa entrevista:

Sessão de Introdução – Apresentação e explicações dos motivos da entrevista. Questões éticas podem ser envolvidas e, se for o caso, pedir autorização para gravar a entrevista.

Sessão de Aquecimento – Perguntas fáceis e que não "intimidam" devem ser realizadas primeiramente.

Sessão de Principal – Apresentar as questões em uma sequência lógica, procurando deixar as mais difíceis para o final.

Sessão de descanso – Algumas poucas questões fáceis, para dissipar eventuais tensões.

Sessão de encerramento – Agradecimentos ao entrevistado e procedimentos finais.

Seja profissional e trate a entrevista com responsabilidade e seriedade.

OBSERVAÇÃO

Envolve ver, ouvir e registrar tarefas significativas do usuário;

Não deve fazer que o usuário alterem seu comportamento na frente do observador;

Observar diversas situações (normalidade, críticas, aprendizado etc.);

Escolha do método de registro e testar antes;

Reforçar a mensagem de que se trata de conhecer uma situação ou procedimento e não de avaliar o desempenho na atividade;

Fotos do local ajudarão a análise do trabalho;

O relatório deve ser elaborado assim que a observação terminar.

QUESTIONÁRIO

Questões projetadas a fim de obter informações específicas.

Diferentes tipos de respostas: sim/não, múltipla escolha, respostas textuais.

Impressos, via e-mail ou em um website ou sistema on-line.

A eficiência do método depende do planejamento do questionário.

Atenção: o que as pessoas dizem nem sempre são o que fazem.

Questões básicas: Informações demográficas (gênero, idade, formação), experiência do usuário. Descobrir a diversidade dentro de um grupo.

Após as questões genéricas, são colocadas as questões específicas, que contribuirão para o estabelecimento dos requisitos.

REQUISITOS

O requisito é uma condição cuja exigência deve ser satisfeita.

Se a condição é produzir algo, diz-se que o requisito é funcional

Se a condição é caracterizar algo (propriedade, comportamento, restrição, etc,...), diz-se que o requisito é não-funcional.

Requisitos funcionais correspondem à listagem de todas as coisas que o sistema deve fazer;

Requisitos não funcionais são restrições e qualidades que se coloca sobre como o sistema deve realizar seus requisitos funcionais;

CLASSIFICAÇÃO DE REQUISITOS NÃO FUNCIONAIS

Usabilidade:requisitos que selecionam ou afetam a usabilidade do sistema. Exemplos incluem a facilidade de uso e a necessidade ou não de treinamento dos usuários.

Confiabilidade: Tratamento de falhas, possibilidade de previsão, não erros de programação;

Desempenho: Velocidade, eficiência, precisão, tempo de recuperação, tempo de resposta, uso de recurso, etc;

Configurabilidade: O que pode ser configurado pelos usuários do sistema;

Portabilidade:restrições sobre a plataforma de hardware e de software nas quais o sistema será implantado e sobre o grau de facilidade para transportar o sistema para outras plataformas.

Segurança: Permissões de usuários do sistema;

REQUISITOS

Requisitos funcionais evidentes são efetuados com conhecimento do usuário;

Requisitos funcionais ocultos são efetuados pelo sistema sem o conhecimento explícito do usuário;

Descrever requisitos funcionais e requisitos não-funcionais requer tratar dois aspectos: primeiro, "Produzir"; segundo, "com Qualidade", as duas faces da moeda aplicáveis à Engenharia de Software.

O processo de produção de software depende da definição clara de qual produto construir.

Esta definição fundamenta-se no conhecimento do problema e na viabilização de oportunidade de negócio com o uso de tecnologia da informação.

A estratégia é o tratamento multidisciplinar da informação de requisitos obtida do ponto de vista dos stakeholder (fonte de informação) para o entendimento e atendimento às necessidades.

TABELA DE REQUISITOS FUNCIONAIS

Código do requisito funcional (Ex.: F1, F2, F3, ...).

Nome do requisito funcional (especificação curta).

Descrição (especificação longa e detalhamento do requisito).

Categoria funcional: evidente ou oculto.

Código do requisito não funcional (Ex.: NF1.1, NF1.2, ... NF2.1, NF2.2, ...).

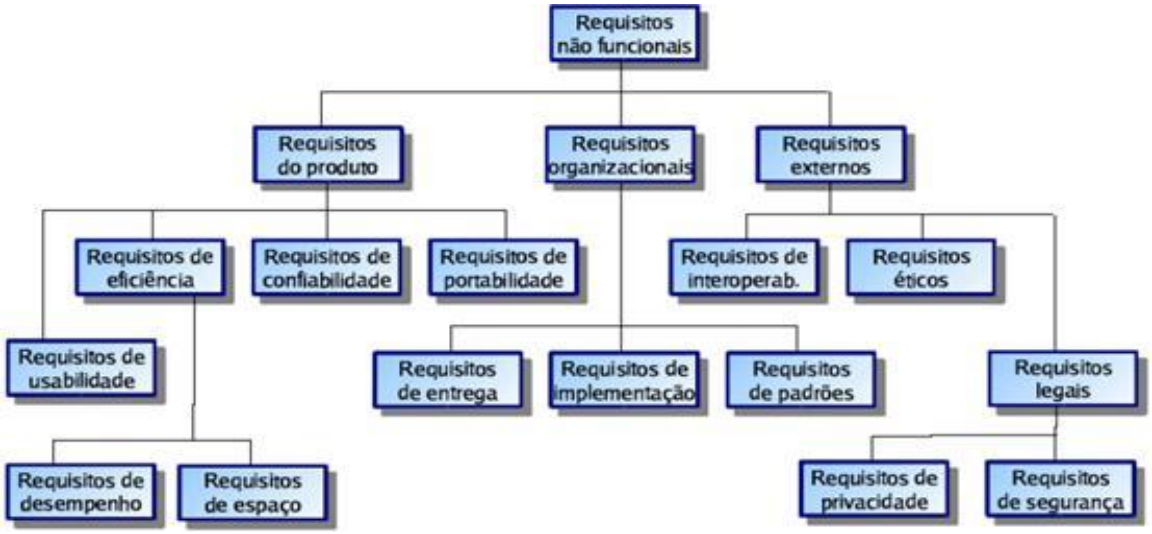
Nome do requisito não funcional (especificação curta).

Restrição: especificação do requisito não funcional.

Categoria: tipo de restrição: segurança, performance, compatibilidade, etc.

Obrigatoriedade: se o requisito é desejável ou obrigatório.

REQUISITOS NÃO FUNCIONAIS



DESAFIOS DA ANÁLISE DE REQUISITOS

Como descobrir os requisitos;

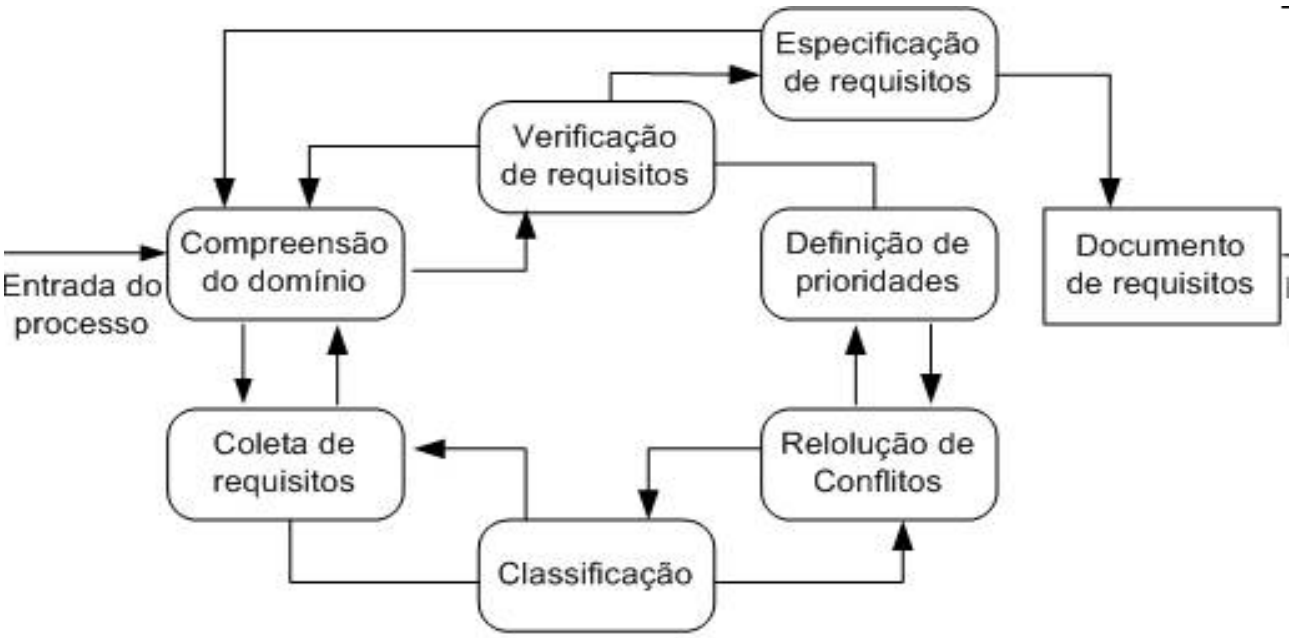
Como comunicar os requisitos para as outras fases ou equipes do projeto;

Como lembrar dos requisitos durante o desenvolvimento e verificar se foram todos atendidos

Como gerenciar a mudança

Técnicas e Tipos de Requisitos

Processo de levantamento de requisitos



Dificuldades

1) Cliente/usuário não sabem o que querem, ou não sabem expressar o que querem.
2) Expressão de requisitos em seus próprios termos.
3) Sobre um mesmo problema: Requisitos diferentes para diferentes usuários.
4) Um stakeholder errado afetará em perda de tempo e dinheiro para ambas as partes envolvidas no desenvolvimento do sistema.

Técnicas

Levantamento Orientado a Ponto de Vista

Por que há diferentes tipos de usuário final

Por que usuários tem interesses diferentes em requisitos

Sommerville p. 106

Perspectiva de cada pessoa sobre o sistema

(Pressman p. 242)

Usuários num Sistema de uma clinica médica

Para levantar os pontos de vista, realiza-se :

Entrevistas com os usuários

Reuniões

Obtém-se Serviços do sistema

Entrada de dados

Requisitos não funcionais

Eventos de controle

Exceções

Clínica Médica - Identificar Pontos de vista e Serviços

Paciente

Realizar Consulta/Exame

Receber Laudo

Ser atendido com seu convênio

Realizar pagamento (caso atendimento particular)

Clínica Médica - Identificar Pontos de vista e Serviços

Recepcionista

Cadastrar Paciente (Dados cadastrais)

Verificar se paciente cadastrado

Agendar Atendimento

Checar guia de atendimento (caso de convênios)

Preencher atendimento (Paciente, convênio, serviço, médico)

Confirmar Atendimento

Emitir recibos/formulário de entrega de resultado

Cenários

Cenários são textos ou narrativas sobre pessoas e suas atividades, criados com o intuito de apresentar o conceito de novos produtos.

Essa construção textual permite inseri-los dentro de uma situação plausível mesmo que hipotética, identificar potenciais problemas, antecipar necessidades e até propor soluções alternativas para os problemas levantados.

Ambiente: descreve um estado inicial do ambiente onde o episódio acontece, caracteriza se o ambiente fisicamente, como as pessoas estão nele presentes.

Atores ou agentes: aqueles que participam do episódio descrito interagem com o ambiente influenciando ou sendo influenciado.

O roteiro: seqüência de ações e eventos representando o que os atores fazem durante o episódio, o que lhes acontece e que mudanças ocorrem no ambiente.

Clínica Médica

Cenários para atendimento de Paciente

Ambiente

Recepção de uma clínica, há um computador com um sistema de atendimento instalado.

Atores

Paciente

Recepcionista

Roteiro

1. Paciente solicita atendimento entregando cartão de convênio e uma guia
2. Recepcionista:
   1. Recebe Cartão de convênio e guia
   2. Checa se convênio e serviços são credenciados
   3. Checa se paciente já cadastrado
   4. Cadastra paciente
   5. Cadastra Atendimento e Confirmar
   6. Emitir Formulário de Recebimento de laudo
   7. Entrega formulário para o Paciente

Clínica Médica

Cenário Negativo

Paciente solicita atendimento entregando cartão de convênio e uma guia

Recepcionista:

Recebe Cartão de convênio e guia

[Convênio e serviços são credenciados, mas não há médicos para atendimento de tal serviço.] [Paciente não cadastrado e esqueceu CPF.] [A emissão de Formulário de Recebimento de laudo não acontece devido a problema na impressora] [Criança trazida pelo paciente desconecta cabo do computador]

Storyboard

É uma representação das interações entre os usuários e o sistema em seu ambiente de trabalho. Ela corresponde ao detalhamento de um cenário de uso especificado para o sistema, consistindo em uma sequência de desenhos representando não só esboços de telas, mas também elementos do contexto (usuário, equipamento, móveis, telefones, pessoas, etc.).

O propósito da técnica é obter uma reação antecipada dos usuários com relação aos conceitos propostos para uma aplicação.

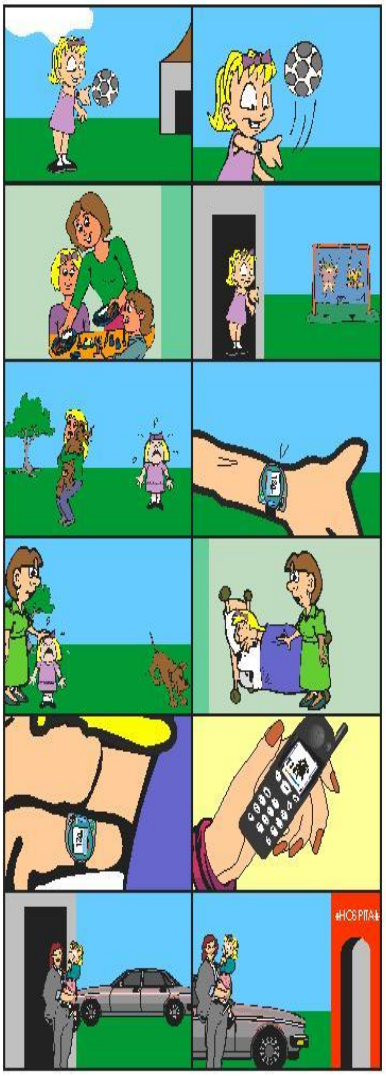
• Amigável, informal e interativa.

• Ótima para validar interfaces do sistema.

• Fácil de criar e fácil de modificar.

Exemplo de storyboard

Positivo



Bianca é uma criança de 2 anos, que tem problemas de coração. Todos os dias ao sair a mãe de Bianca ajusta sua pulseira para receber as mensagens de 3 em 3 horas.
Ela precisa ficar monitorando Bianca devido o problema que ela tem, pois não confia em deixar por conta da Babá.Certo dia Carla(mãe) saiu para o trabalho e deixou Bianca com D. Marleide(babá). Ao sair conversou com a babá e pediu que ficasse observando Bianca para evitar qualquer problema.Até o meio dia tudo ocorria bem, todos almoçaram e voltaram para o trabalho, por volta das 14h50min Bianca ao sair para o jardim levou um grande susto, o cachorro do vizinho tinha conseguido entrar em seu jardim, logo começou a gritar, D. Marleide (babá) quando chegou para socorrer Bianca já estava bem agitada e com fortes palpitações, D.Marleide não se preocupou em avisar a Carla, então tentou acalmar a criança e após para dormir. Durante o sono Bianca começou a ter arritmia, mas devido ao sistema de monitoramento Carla recebeu a mensagem que Bianca esta com os batimentos cardíacos muito alto, entrou em contato urgentemente com D. Marleide para dar os primeiros socorros e deixou o trabalho às pressas, e conseguiu rapidamente levar sua filha para o hospital.

Negativo

Uma situação muito preocupante aconteceu com a família de Bianca. Bianca é uma criança de 2 anos, que tem problemas de coração. Tudo teve início quando Bianca ao brincar pela manhã sem querer deu um impacto muito grande na sua pulseira de monitoramento, nada foi percebido, pois a empregada estava preparando o almoço quando tudo aconteceu e a mãe de Bianca (Carla) estava trabalhando. Até o meio dia tudo ocorria bem, todos almoçaram e voltaram para o trabalho, por volta das 14h50min Bianca ao sair para o jardim levou um grande susto, o cachorro do vizinho tinha conseguido entrar em seu jardim, logo começou a gritar, D. Marleide (empregada) quando chegou para socorrer Bianca já estava bem agitada e com fortes palpitações, D.Marleide não se preocupou em avisar a Carla, pois Bianca estava com o sistema de monitoramento ativado, então tentou acalmar a criança e a pôs para dormir. Durante o sono Bianca começou a ter arritmia, mas devido ao impacto no sistema de monitoramento Carla não estava recebendo mensagens, quando D. Marleide percebeu, entrou em contato urgentemente com Carla que teve que deixar o trabalho às pressas, já que o sistema de monitoramento não estava funcionando no momento então rapidamente levou sua filha para o hospital.

Storyboard

Na prática, não há regras ou restrições para a aplicação da técnica. • Entretanto, storyboards podem ser categorizadas em três tipos, dependendo do modo de interação com o cliente:

• Passiva

• Ativa

• Interativa

Storyboard Passiva

Conta-se uma história para o cliente

• Sketches, figuras, apresentações PowerPoint, etc.

• O analista de requisitos executa o papel do sistema e simplesmente "navega" com o cliente pelo storyboard.

• "Quando você faz assim, acontece isso..."

Storyboard Ativa

Tenta-se fazer com que o cliente veja "um filme que ainda não foi feito".

• Animação ou automação.

• Provê uma descrição automatizada do comportamento do sistema numa utilização típica ou num cenário operacional.

Storyboard Interativa

Deixa-se o cliente experimentar o sistema de forma realística e prática.

• Requer a participação intensiva do usuário.

• "Demonstração" do sistema.

StoryBoard

Nos 3 casos, o storyboarding exercita 3 elementos essenciais em qualquer atividade:

• Quem são os envolvidos (atores)

• O que acontece com os envolvidos (comportamento)

• Como isto acontece (descrição da interação)

Etnografia

Etnografia é uma técnica de observação

Objetiva compreender requisitos sociais/organizacionais

Analista se insere no ambiente no qual o sistema será utilizado e observa o trabalho diário e anota

Ajuda a descobrir requisitos implícitos

Requisitos descobertos com eficácia com a etnografia

Técnica de etnografia:

Identificar as áreas do usuário a serem observadas

Obter aprovação da gerência

Obter os nomes e funções das pessoas chave que estão envolvidas no estudo de observação

Explicar a finalidade do estudo

Etnografia – Desvantagens

Consumir bastante tempo

Analista ser induzido as erros em suas observações

Prototipação

Protótipo tem por objetivo explorar aspectos críticos dos requisitos de um produto

O protótipo é indicado para estudar as alternativas de interface do usuário

problemas de comunicação com outros produtos

a viabilidade de atendimento dos requisitos de desempenho.

Prototipação – benefícios

reduções dos riscos na construção do sistema;

O uso de protótipo auxilia na elicitação e validação dos requisitos de sistema;

A prototipação pode ser utilizada para elicitar requisitos quando há um alto grau de incerteza ou quando é necessário um rápido feedback dos usuários.

Prototipação de Baixa Fidelidade

Prototipação de Alta Fidelidade

Tipos de Requisitos

Requisitos do Usuário

Declarações, em linguagem natural e também diagramas/formulários sobre as funções que o sistema deve fornecer e as restrições sob as quais deve operar.

Descreve requisitos ... de modo compreensível pelo usuários do sistema que não tem conhecimento técnico detalhados.

Especificam comportamentos externos do sistema

Requisitos de Sistema

Descrições detalhadas dos requisitos do usuário

Podem servir de base para o contrato, contendo especificações concretas e consistentes

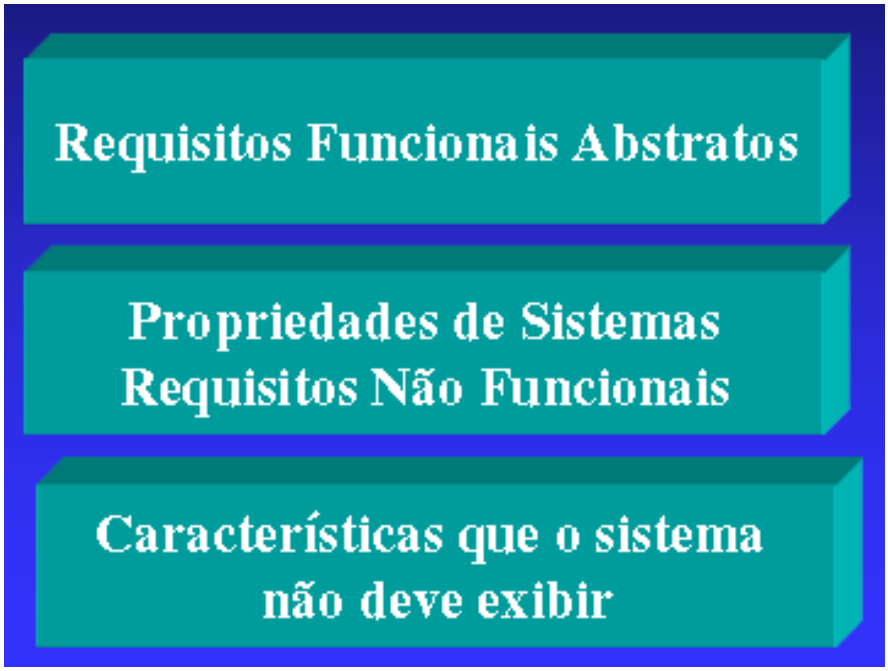
Base para o projeto de sistemas

Define o que o sistema deve fazer e não como deve ser implementado

Sommerville p. 91-95

Requisitos de Sistema

Classificação



Sommerville p. 26-27, Peters p. 102