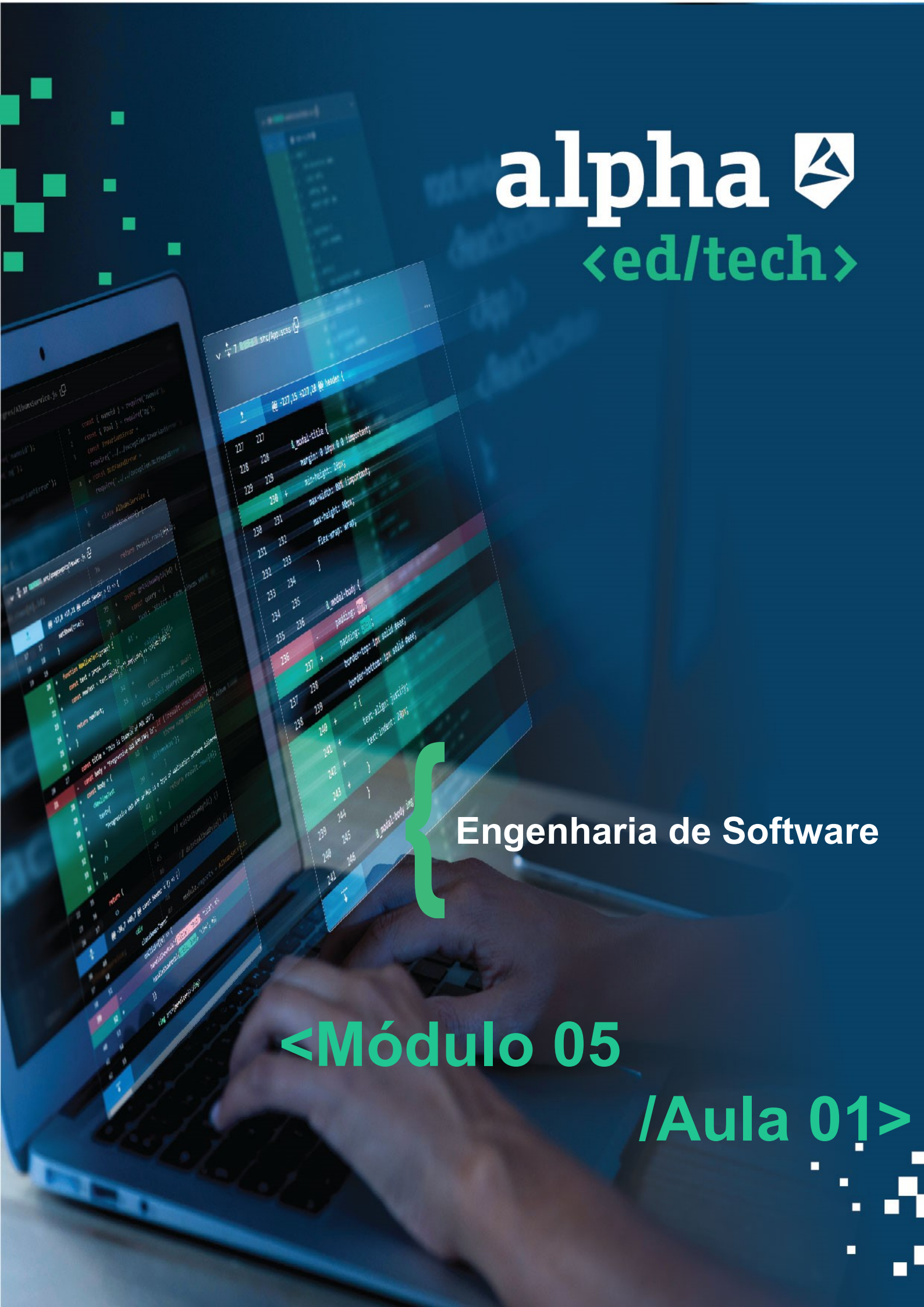

alpha
<ed/tech>
Engenharia de Software
<Módulo 05
/Aula 01>

Engenharia de Software

Histórico e Evolução

Vamos estudar a evolução e história da **modelagem de sistemas**, através das décadas e sobretudo das **mudanças de abordagem**.

Atualmente os profissionais chamados de **"Engenheiros de Software"** foram evoluindo junto com as ferramentas e técnicas para **melhoria do desenvolvimento de sistemas**.

Década de 1960

Fase Zero, nenhuma modelagem, péssima documentação.

Até a década de 1960, os **sistemas eram desenvolvidos pelos cientistas da computação** e **não tinham ferramentas nem técnicas efetivas**. No entanto, **os cientistas da computação**, que muitas vezes **eram os próprios usuários dos sistemas que estavam desenvolvendo**, tinham um **profundo conhecimento dos sistemas** e, portanto, conseguiam criar soluções adaptadas às suas necessidades.

Década de 1970

Durante os anos 70, houve um **aumento na complexidade dos sistemas**, levando à necessidade de abordagens mais estruturadas.

A "**Conferência NATO sobre Desenvolvimento de Software**" realizada em **Garmisch**, Alemanha, em **1698**. O **termo "A Crise do Software"** foi cunhado durante essa conferência.

O autor que introduziu o conceito de **"A Crise do Software"** foi o renomado cientista da computação **Edsger Dijkstra**.

Em seu famoso discurso "A Crise do Software", ele destacou os **desafios crescentes associados ao desenvolvimento de software**, incluindo a complexidade em rápido aumento dos sistemas e a **necessidade de abordagens mais estruturadas e sistemáticas** para lidar com essa complexidade.

Nesse contexto, Dijkstra argumentou que a falta de métodos formais e rigorosos no **desenvolvimento de software** estava levando a problemas significativos na criação de sistemas confiáveis e eficientes. Ele enfatizou a necessidade de disciplina, matemática aplicada e métodos formais para superar os desafios emergentes na criação de software.

Portanto, a Conferência NATO de 1968 e o trabalho de Edsger Dijkstra foram cruciais para destacar os desafios enfrentados pelo campo de desenvolvimento de software e para iniciar discussões sobre a necessidade de abordagens mais estruturadas e científicas na criação de software.

Ver sobre algoritmo de roteamento de Dijkstra na teoria de redes de computadores.

Engenharia de Software

Histórico

Nessa época surgiu a conceituação de que **o desenvolvimento de sistemas deveria ser visto como uma engenharia**, onde o software deveria ter um projeto com diagramas e detalhamento de como seria desenvolvido, além de apresentação a priori de como seria o sistema quando ficasse pronto.

Assim, foi cunhada a disciplina de **Engenharia de Software**.

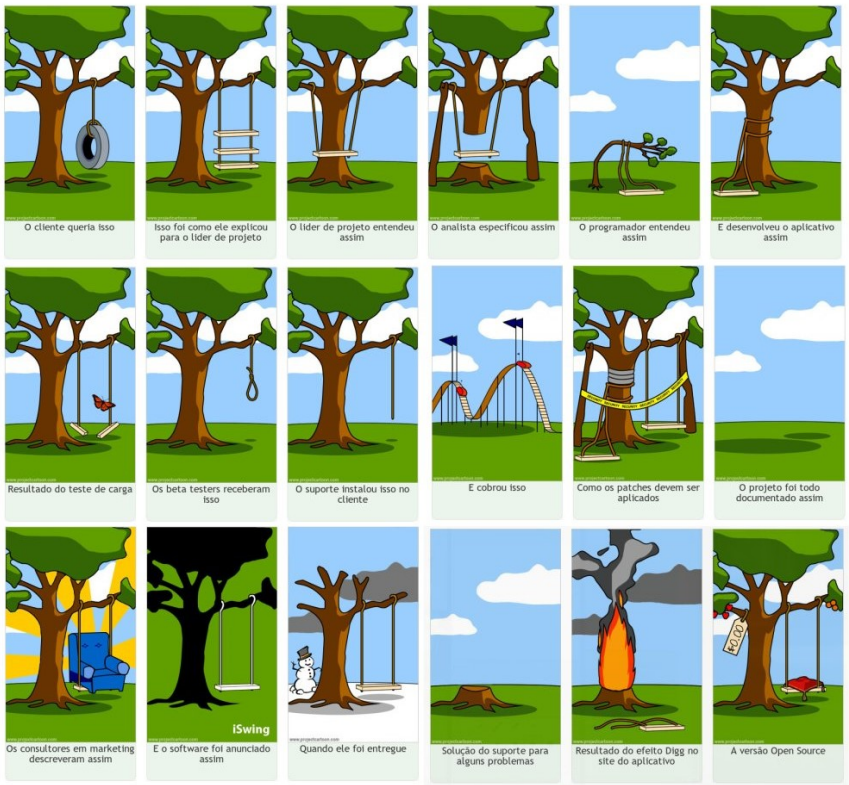
Também **Engenharia de Sistemas**, mas isto é mais genérico, quando envolve software, hardware, treinamento de pessoas etc.

Engenharia de Software surgiu como analogia com a engenharia (em particular a engenharia civil e de construções) que **exige um projeto (*design* e não *project*)** (plantas, desenhos, vistas tridimensionais, maquetes, memorial descritivo, cronograma financeiro com valores de investimento) **que abrange todo um detalhamento do que será feito antecipadamente ao início dos trabalhos**.

Não se concebe que alguém inicie um empreendimento de uma casa, ou uma pirâmide, sem ter o detalhamento preciso do que será edificado, antes de iniciar as obras.

Mas com o software, isto era feito, baseado em um pequeno texto com definições vagas das necessidades a serem atendidas pelo sistema que já se iniciava.

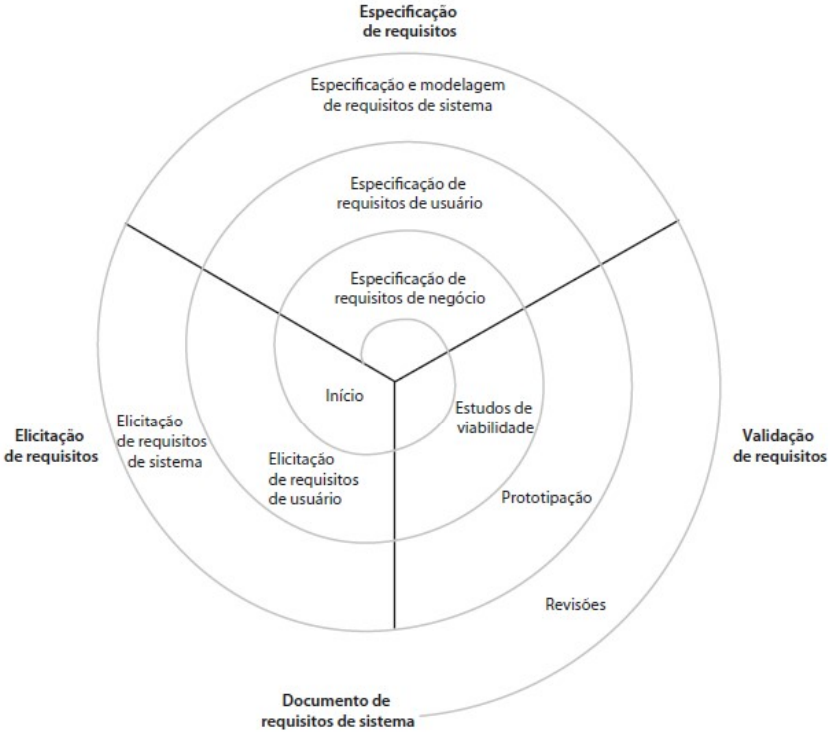
Na entrega do sistema, se descobria que o que foi desenvolvido não atendia as necessidades de quem contratou o serviço.



http://200.17.137.109:8081/novobsi/Members/jones/disciplinas/fundamentos-engenharia-de-software

Engenharia de Software

Engenharia de Requisitos



Fonte: Sommerville, Ian – Engenharia de Software

A conceituação de que o desenvolvimento de sistemas deveria ser tratado como uma engenharia, com ênfase em planejamento, projeto, documentação e processos mais formais, começou a surgir na década de 1960.

Vários autores contribuíram significativamente para essa evolução, destacando-se:

Friedrich Ludwig Bauer: Foi um dos primeiros a usar a analogia com engenharia para descrever o desenvolvimento de software. Em 1968, ele propôs que a construção de software deveria seguir métodos sistemáticos, assim como na engenharia tradicional.
Edsger Dijkstra: Além de sua contribuição para a "Crise do Software", Dijkstra também defendeu práticas mais disciplinadas e métodos rigorosos no desenvolvimento de software. Ele enfatizou a necessidade de abordagens formais e científicas.
Winston Royce: Em 1970, Winston Royce apresentou um artigo seminal chamado "Managing the Development of Large Software Systems," onde introduziu o modelo em cascata (*waterfall model*). Embora tenha reconhecido desafios no modelo, foi um marco ao propor fases distintas no desenvolvimento de software.
Barry Boehm: Conhecido por seu trabalho na década de 1980, Boehm contribuiu com o modelo espiral, uma abordagem iterativa e incremental para o desenvolvimento de software. Ele também enfatizou a importância de gerenciamento de riscos.
Winston Royce: Além de sua contribuição ao modelo em cascata, Royce também introduziu o termo "engenharia de software" em um artigo publicado em 1968, onde delineou princípios e práticas para tratar o desenvolvimento de software como uma disciplina de engenharia.

Esses autores e outros desempenharam papéis fundamentais na formação da Engenharia de Software como uma disciplina distinta, promovendo a ideia de que o desenvolvimento de software deveria seguir princípios e práticas semelhantes aos utilizados na engenharia tradicional.

Engenharia de Software

Na década de 1970, a **Engenharia de Software** começou a reconhecer a necessidade de **abordagens** mais **estruturadas** para lidar com o aumento da complexidade dos sistemas de software e melhorar a **qualidade do desenvolvimento**.

Dois modelos que foram amplamente adotados nesse período para resolver o problema da baixa qualidade:

1. Modelagem de Dados (utilizando os diagramas: DER e MER).
2. Diagramas de Fluxo de Dados (DFD modelagem de processos).

Vamos entender como cada um desses modelos contribuiu para a melhoria da qualidade dos sistemas desenvolvidos..

Modelagem de Dados

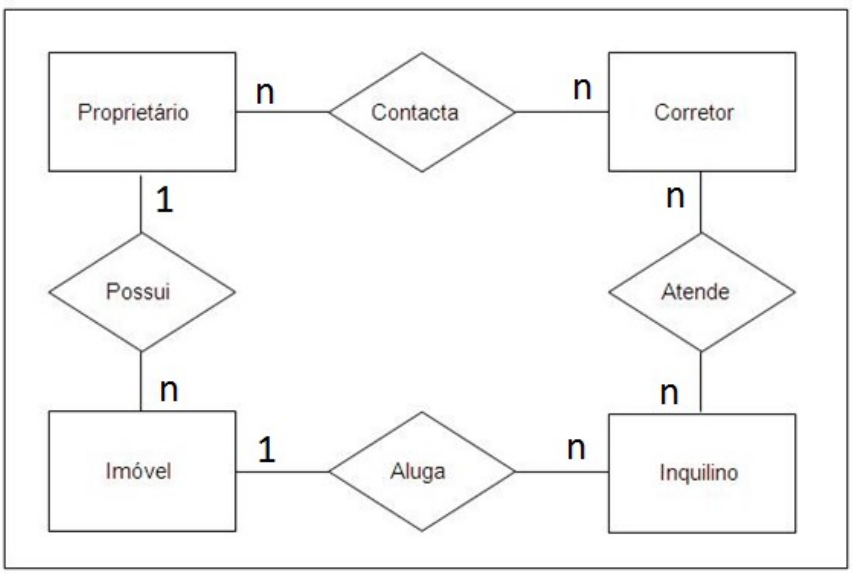
Objetivo:
•Representar visualmente a estrutura dos dados em um sistema.

Ferramentas Principais:
•**Diagrama de Entidade-Relacionamento (DER)**: Este diagrama descreve as entidades (conjuntos de entes ou conceitos) dentro do sistema, os relacionamentos entre elas e a natureza desses relacionamentos.
•**Modelo de Entidade-Relacionamento (MER)**: Proporciona uma visão abstrata e clara das entidades (se tornam tabelas no SQL ou tabelas de memória) e seus relacionamentos, enfatizando a integridade dos dados.

Resolução do Problema:
•Estrutura de Dados Clara: A modelagem de dados fornece uma visão estruturada e organizada dos dados, ajudando a evitar redundâncias e inconsistências.
•Compreensão Visual: Os diagramas permitem que os desenvolvedores e stakeholders visualizem e compreendam facilmente a estrutura do banco de dados.
•Integridade e Relacionamentos Explícitos: Ajuda na manutenção da integridade referencial dos dados, garantindo que as relações entre diferentes partes do sistema sejam explicitamente definidas.

https://pt.wikipedia.org/wiki/Modelo_entidade_relacionamento

https://www.devmedia.com.br/mer-e-der-modelagem-de-bancos-de-dados/14332



Engenharia de Software

Diagramas de Fluxo de Dados (DFD)

Objetivo:
Representar visualmente como as informações fluem dentro de um sistema, entre os módulos do sistema.

Componentes de um DFD:
•**Processos**: Representar visualmente as rotinas e procedimentos executados pelo sistema. São desenhados como retângulos com cantos arredondados ou também representados por elipses (os autores de livros reforçavam que estes diagramas deveriam ser esboçados manualmente, durante a entrevista feita pelos analistas de sistema, ou analistas de O&M, junto ao usuários do sistema).
•**Entidades Externas**: que não são processos ou rotinas do sistema a ser desenvolvido. Os geradores de informação podem ser departamentos que geram informações, empresas externas, outros sistemas que geram ou recebem informações do sistema a ser desenvolvido.
•**Fluxo de Dados**: setas indicando o sentido do "caminho das informações", de qual processo saiu a informação para qual outro processo a informação "fluiu". O fluxo de dados pode ser de qualquer entidade externa. Cada seta que representa um fluxo de dados deve ter um nome no desenho do diagrama e também deve ter uma descrição textual de todos os dados que a seta representa. Por exemplo, se uma seta indica que uma nota fiscal é um fluxo de dados entre o departamento de faturamento para a empresa de contabilidade externa, essa seta deve ter um descritivo de todos os dados que constariam na nota fiscal correspondente.

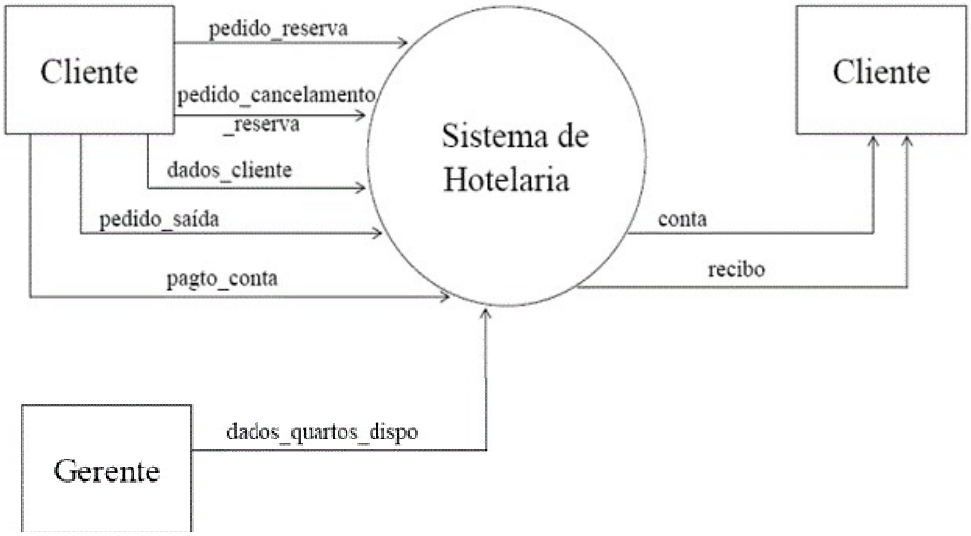
Ferramentas Principais:
•**Diagrama de Fluxo de Dados (DFD)**: Mostra o fluxo de dados entre diferentes partes (módulos, pois nesta metodologia de análise estruturada de sistemas a modularização era fundamental) de um sistema e como esses dados são processados.
•**Diagrama de Contexto**: É um DFD que só representa um único processo, que é o sistema a ser desenvolvido. Mostra todas informações que entram e que saem do sistema.

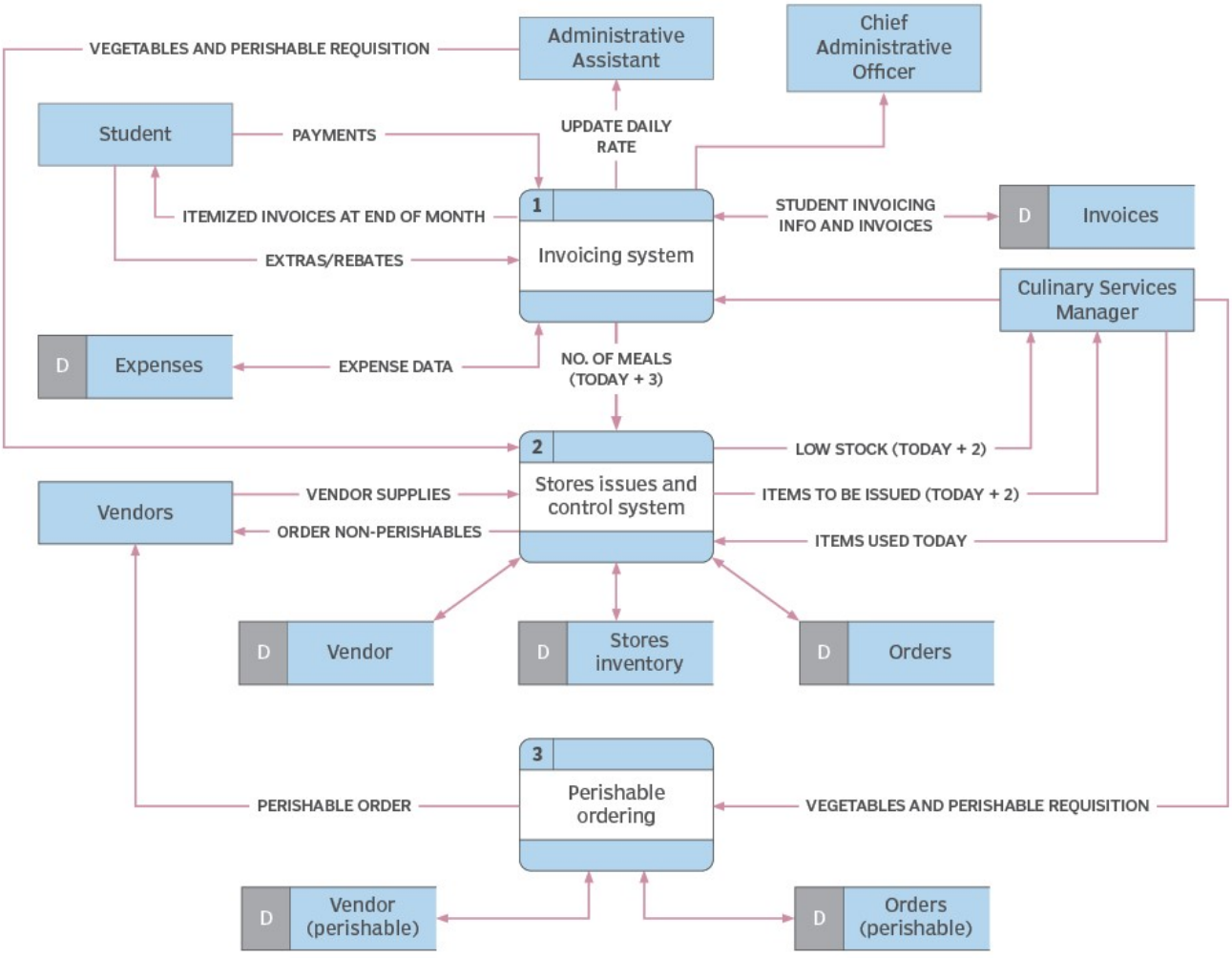
Resolução do Problema:
•Visualização do Fluxo de Informações: Permite entender como as informações se movem entre diferentes partes do sistema, desde a entrada até o processamento e a saída.
•Identificação de Processos: Destaca os processos que manipulam os dados, facilitando a compreensão das funcionalidades do sistema.
•Identificação de Entradas e Saídas: Ajuda a identificar claramente as entradas necessárias e as saídas geradas pelo sistema, contribuindo para a definição precisa dos requisitos.

Esses modelos forneceram uma abordagem mais sistemática e **principalmente visual** para o desenvolvimento de software, abordando questões de qualidade ao fornecer representações claras e compreensíveis da estrutura de dados e do fluxo de informações nos sistemas.

Eles se tornaram fundamentais para a evolução da Engenharia de Software, estabelecendo bases sólidas para práticas mais disciplinadas e eficientes. A ideia disseminada era que o usuário final conseguiria visualizar o sistema final a ser desenvolvido e se imbuir de que o sistema realmente atenderia suas necessidades do negócio.

Engenharia de Software





https://pt.wikipedia.org/wiki/Diagrama_de_fluxo_de_dados
https://pt.wikipedia.org/wiki/Diagrama_de_contexto

Engenharia de Software

Década de 1980

A abordagem de Engenharia de Software começou a ganhar destaque no final da década de 1980. **Métodos formais e técnicas mais estruturadas**, como a Engenharia de Software Orientada a Objetos (início dos anos 90), começaram a ser adotadas.
Ferramentas CASE (*Computer-Aided Software Engineering*) também surgiram, proporcionando um ambiente mais sistemático para o desenvolvimento de software.

Década de 1990

Na década de 1990, a modelagem de sistemas tornou-se mais centrada no usuário com a **ascensão da Engenharia de Requisitos**.

Metodologias ágeis (final de 90 início de 2000), como o **Scrum**, começaram a ganhar popularidade, destacando a importância da comunicação contínua com os clientes.

Década de 2000 até o Presente

Nos últimos anos, as práticas ágeis evoluíram ainda mais, com DevOps integrando desenvolvimento e operações.
Além disso, arquiteturas orientadas a serviços (SOA) e microsserviços ganharam proeminência.
Ferramentas avançadas de modelagem, integração contínua e entrega contínua (CI/CD) também transformaram a maneira como os sistemas são desenvolvidos, testados e implementados.

Os Engenheiros de Software modernos incorporam uma variedade de práticas, incluindo metodologias ágeis, DevOps, e utilizam uma gama diversificada de ferramentas para criar sistemas complexos e escaláveis, adaptados às crescentes demandas tecnológicas.

A modelagem de sistemas continua a evoluir à medida que novas tecnologias e abordagens surgem.

A ideia de objetos, classes, encapsulamento e herança começou a ser explorada na década de 1970, sendo que linguagens como Simula e Smalltalk foram pioneiras nesse conceito.

Engenharia de Software

Análise Estruturada vs. Análise Orientada a Objetos:
Compare os princípios fundamentais da análise estruturada e da análise orientada a objetos. Quais são as principais diferenças e semelhanças entre essas abordagens?

Diagrama de Fluxo de Dados (DFD):
Explique como funciona um Diagrama de Fluxo de Dados (DFD) na análise estruturada. Quais são os símbolos principais e como eles representam as informações e processos no sistema?

Modelagem de Dados:
Quais são os principais elementos da modelagem de dados na análise estruturada? Como os Diagramas de Entidade-Relacionamento (DER) são utilizados nesse contexto?

Dicionário de Dados:
O que é um Dicionário de Dados na análise estruturada? Qual é sua importância na documentação e desenvolvimento de sistemas?

Especificação de Requisitos:
Como a análise estruturada contribui para a especificação de requisitos em projetos de software? Quais são as técnicas comuns usadas nesse processo?

Modelagem Funcional:
Explique o conceito de modelagem funcional na análise estruturada. Como os diagramas de fluxo de dados e os diagramas de árvore de decisão são utilizados nesse contexto?

Ferramentas de Apoio à Análise Estruturada:
Quais são as ferramentas de software disponíveis para apoiar a prática da análise estruturada? Cite exemplos e discuta suas funcionalidades.

Ciclo de Vida de Desenvolvimento de Sistemas:
Como a análise estruturada se integra ao ciclo de vida de desenvolvimento de sistemas? Quais são as fases em que a análise estruturada desempenha um papel crucial?

Problemas e Críticas à Análise Estruturada:
Pesquise sobre críticas e desafios associados à análise estruturada. Quais são alguns dos problemas comuns enfrentados ao aplicar essa abordagem?

Adaptações Contemporâneas:
Como a análise estruturada se adaptou às mudanças nas práticas de desenvolvimento de software contemporâneas, como metodologias ágeis? Quais são as tendências atuais relacionadas à análise estruturada?