Analise e Modelagem de Sistemas

Fundamentos da engenharia de software

Eduardo Furlan Miranda

2025-07-01

Fonte: WERLICH, C. *Análise e Modelagem de Sistemas*.
Londrina: EDE SA, 2020. ISBN 978-85-522-1683-4.

Fundamentos da Engenharia de Software

- Trabalhar na área de engenharia de software é sempre desafiador
  - Um sistema nunca é igual ao outro
  - Constantes novidades tecnológicas
- Necessários organização e muito empenho
- Objetivo: produzir um software de qualidade

Cenário do Cliente: Rede de Postos de Coleta

- Cliente: possui uma rede de postos de coleta de exames laboratoriais no Sul do país
- Sistema utilizado atualmente:
  - Desktop, sem acesso à internet
  - Quase 15 anos de uso
  - Nunca foi atualizado
- A empresa cliente pensa, agora, em atualizá-lo
- Nossa missão inicial como analista de sistemas:
  - Analisar o software do possível cliente
  - Sugerir novas ideias para fechar um contrato

Nossa Missão: Questões Chave para a Análise

- Viabilidade: é mais viável atualizar o sistema existente ou criar um novo?
- Tecnologias: quais as novas tecnologias que podem ser utilizadas em um novo sistema?
- Processos: quais processos deverão ser adotados para manter o software sempre atualizado?
- Profissionais: qual o perfil dos profissionais envolvidos no processo da análise?

O que é Software? Definições e Elementos

- Sommerville (2011) define software como:
  - Programas de computadores com documentação associada
  - Produtos desenvolvidos para um cliente específico ou mercado generalizado
- Pressman (2016) afirma que software é:
  - Produto que profissionais de TI desenvolvem e dão suporte a longo prazo
  - Abrange qualquer tipo de mídia eletrônica
- Software é a união de três elementos:
  - Instruções
  - Estruturas de dados
  - Documentação
- Instruções: quando executadas, fornecem os atributos e funções desejados pelos usuários

Elementos e Evolução do Software

- Estruturas de dados:
  - Possibilitam aos programadores manipular informações
  - De forma mais adequada à necessidade da aplicação
- Documentação:
  - Informação descritiva do software
  - Detalha operação de uso dos programas
  - Inclui diagramas de funcionalidades
- O software não "surgiu" do nada
- Houve um processo evolutivo que profissionais de TI devem conhecer

Cronologia da Evolução do Software

- Década de 1940: O Programa
  - Executável, com controle total sobre o computador
  - Responsável por hardware (Sistema Operacional) e operações matemáticas
- Décadas de 1950 e 1960: Sistemas Operacionais e Linguagens de Programação
  - Sistemas Operacionais: controle do hardware, interface homem-máquina
  - Linguagens de Programação: surgimento de COBOL, LISP, ALGOL, BASIC, etc.

Cronologia da Evolução do Software

- Décadas de 1960 e 1970:
  - Crise do Software: grande crescimento de sistemas
    - Quase não havia planejamento e controle
    - Desenvolvimento informal: atrasos e custos acima do orçamento
    - Surgiu a necessidade de criar métodos de engenharia de software
  - Paradigmas de Programação: criação do conceito de orientação a objetos
  - Sistemas Operacionais mais eficientes: foco na programação desktop
    - Surgiram MS-DOS da Microsoft e Macintosh da Apple

Cronologia da Evolução do Software

- Década de 1980:
  - Computador Desktop: surgimento da Microsoft e evolução dos PCs
  - Unix: sistemas Operacionais em rede avançam pelo mundo
  - Evolução da Internet: começa a despontar como meio de comunicação
- Década de 1990:
  - Internet: amplamente utilizada
  - Linux: surge o núcleo do futuro Linux
  - JAVA: nasce a linguagem JAVA, revolucionando a orientação a objetos

Cronologia da Evolução do Software

- Década de 2000:
  - Sistemas Operacionais Gráficos:
    - Geração Windows da Microsoft (XP, Vista, 70)
    - Versões gráficas do LINUX
  - Internet: softwares que utilizam a web como plataforma
- Década de 2010 e 2020:
  - Computação em Nuvem: utilizada em larga escala por diversos segmentos
  - Aplicativos para Dispositivos Móveis: essenciais com a evolução dos dispositivos
  - Inteligência Artificial: utilização de algoritmos para Deep Learning e Machine Learning

Evolução das Necessidades do Cliente e a Tecnologia

- Aumento da necessidade de melhorar softwares à medida que novas tecnologias se tornam acessíveis
- Necessidades dos clientes se alteram conforme a tecnologia é disponibilizada
- Década de 1990: empresa precisava de software e talvez um site (cartão de visitas)
- Década de 2000: empresa precisava de software com acesso à internet para recursos para clientes
- A partir de 2010: empresa precisava de software, site e aplicativo
  - Serviços acessíveis por diversos dispositivos
  - Armazenamento da base de dados na nuvem
- A partir de 2020: empresa já pensando em utilizar Inteligência Artificial para melhorar processos gerenciais

Software e Sistema: Diferenças e Componentes

- Software (Pressman, 2016):
  - Programa de computador
  - Sequência lógica de instruções a serem seguidas/executadas
  - Manipulação de um determinado conjunto de informações
- Sistema (Sommerville, 2011):
  - Conjunto de componentes inter-relacionados
  - Funcionam de forma unificada para atingir um certo objetivo
  - Exemplos: sistema Operacional, acadêmico, bancário
- Um sistema é formado por:
  - Número de programas separados
  - Arquivos de configurações para eles
  - Documentação específica (estrutura, usuário)
- Um sistema possui um conjunto de partes que interagem visando um objetivo comum
  - É um conjunto de software, hardware e recursos humanos

A Natureza Mutável do Software

- Mudanças sempre ocorrerão ao longo do tempo de criação e uso de um software
  - Durante o desenvolvimento
  - Na fase da entrega
  - Depois de entregue
- Sempre há necessidade de ajustes e correções
- Pode ocorrer a necessidade de incluir novas funcionalidades
  - Muitas vezes requisitadas pelo cliente
- O software passa por uma série de manutenções
  - Devido a novas solicitações do cliente
  - Após diversos ajustes, pode ser necessária a criação de um novo software

Curva de defeitos de um software ao longo d

Quando?	O que?	Descrição
Década de 1940	• Programa	• O programa era executável e tinha controle total sobre o computador, sendo que o software era responsável por todo o funcionamento tanto do hardware (Sistema Operacional) quanto das operações matemáticas que teria que fazer.
Décadas de 1950 e 1960	• Sistemas Operacionais • Linguagens de programação	• Responsáveis pelo controle do hardware (realizava a interface entre o homem e a máquina). • Surgem as linguagens de programação: COBOL, LISP, ALGOL, BASIC, etc.
Décadas de 1960 e 1970	• Crise do software • Paradigmas de programação • Sistemas Operacionais mais eficientes	• Houve um grande crescimento do número de sistemas e quase não havia planejamento e controle de processos de desenvolvimento de software. Devido ao desenvolvimento informal, havia atrasos e os custos superavam o orçamento estimado (surgindo a necessidade de criar métodos de engenharia de software). • Com novas linguagens sendo criadas, nessa época o conceito de orientação a objetos é criado. • O foco, no momento, era a programação desktop, de modo que foi quando surgiram o MS-DOS da Microsoft e o Macintosh da Apple.

Década de 1980	• Computador desktop • Unix • Evolução da internet	• A década foi marcada pelo surgimento da Microsoft e pela evolução dos computadores pessoais – desktop. • O Unix (Sistemas Operacionais em rede) avançaram pelo mundo. • A internet começa a despontar como meio de comunicação.
Década de 1990	• Internet • Linux • JAVA	• A internet é amplamente utilizada. • Surge o núcleo do futuro Linux. • Nasce a linguagem JAVA, que revoluciona o paradigma da orientação a objetos.
Década de 2000	• Sistemas Operacionais gráficos • Internet	• Geração do Windows da Microsoft (Windows XP, Windows Vista, Windows 70) e versões gráficas do LINUX. • Softwares que utilizam a web como plataforma de funcionamento.
Década de 2010 e 2020	• Computação em nuvem • Aplicativos para dispositivos móveis • Inteligência Artificial	• Utilizada em larga escala por diversos seguimentos de empresas. • Com a evolução dos dispositivos móveis os softwares para aplicativos tornaram-se essenciais. • Utilização de algoritmos para a Deep Learning (Aprendizagem Profunda) e Machine Learning (Aprendizado de Máquina).

Curva de Defeitos de um Software ao Longo do Tempo

- Curva Idealizada:
  - Taxa de defeitos alta no início do desenvolvimento
  - Decresce com o tempo e, em seguida, estabiliza
- Curva Real (Pressman, 2016):
  - Não ocorre o comportamento idealizado
  - Acréscimos de funcionalidades ou correções
  - Mudanças introduzidas tornam o software vulnerável
  - Picos no gráfico representam o aumento das taxas de defeitos
- Melhorias rápidas podem levar à negligência da documentação
- Todo o processo de mudança pode acarretar em um software mais frágil

Vida Útil e Obsolescência do Software

- Qual a vida útil de um software?
  - Perspectiva de usabilidade pode ser longa
  - Pode se tornar ultrapassado se mudanças não atendem expectativas
- Software não se desgasta como hardware (Pressman, 2016)
  - Hardware é componente físico, suscetível a fatores ambientais (temperatura, poeira)
- Todavia, softwares podem deteriorar-se com as mudanças implantadas
- Taxa de defeitos no hardware pode ser por fatores ambientais ou picos de processamento do software
  - Recursos de hardware podem não suportar execução

Leis de Lehman para a Evolução de Softwares

- Sommerville (2011) destaca as Leis de Lehman
  - Utilizadas para verificar a dinâmica da evolução dos softwares
- Estudo realizado na década de 1980, ainda aplicável nos dias atuais
- Leis genéricas, utilizáveis em diferentes softwares com processos de:
  - Tomada de decisão
  - Planejamento
  - Desenvolvimento
  - Manutenção

Leis de Lehman: Mudança, Complexidade e Evolução

- 1. Mudança Contínua:
  - O software deve ser ininterruptamente adaptado
  - Caso contrário, torna-se cada vez menos eficaz
- 2. Aumento da Complexidade:
  - A cada mudança, o software torna-se mais complexo
  - Devem ser tomadas medidas para reduzir a complexidade, mantendo a simplicidade
- 3. Evolução (Autorregulação):
  - O processo de evolução de um software é autoajustável
  - Os atributos do sistema quase não mudam entre as versões

Leis de Lehman

- 4. Estabilidade Organizacional:
  - A evolução do software tende a ser constante na sua taxa de desenvolvimento
  - Independentemente dos recursos utilizados
- 5. Conservação da Familiaridade:
  - Durante o uso do software em evolução, a taxa de mudanças tende a ser proporcional ao domínio da equipe
- 6. Crescimento Contínuo:
  - As funcionalidades do software tendem a aumentar conforme novas versões são desenvolvidas
- 7. Declínio da Qualidade:
  - Caso não haja um processo de evolução, a qualidade cairá
- 8. Sistema de Feedback:
  - Os processos de manutenção agregam sistemas de feedback em múltiplos níveis e ciclos
  - Para obter melhorias nos processos e no produto final

Prática da Engenharia de Software

- De acordo com Sommerville (2011), a prática da engenharia de software é necessária por dois fatores:
  - 1. Dependência da Sociedade:
    - Cada vez mais a sociedade e os indivíduos estão dependentes dos sistemas de software
    - Devemos produzir softwares mais confiáveis e de forma mais econômica
  - 2. Custo-Benefício a Longo Prazo:
    - É mais barato utilizar as técnicas da engenharia de software
    - Evitar mudanças em algo já desenvolvido
    - Partes do software podem ser reutilizadas em outros sistemas

Engenharia de Software - Pressman, 2016

- É uma tecnologia em quatro camadas
- Objetiva disciplina, adaptabilidade e agilidade
- 1. Foco na Qualidade:
  - Objetivo final de toda a engenharia de software
  - Gestão de qualidade baseada em comprometimento organizacional total
- 2. Processo:
  - Base da engenharia de software
  - Ligação entre as demais camadas
  - Possibilita desenvolvimento racional e dentro dos prazos
  - Base para controle e gerenciamento de projetos
  - Produção de artefatos (modelos, documentos, relatórios)

(Cont.)

- 3. Métodos:
  - Fornecem as informações técnicas para o desenvolvimento do software
  - Envolvem uma grande quantidade de tarefas:
    - Comunicação e análise de requisitos
    - Modelagem de projetos e codificação
    - Testes e manutenção do software
- 4. Ferramentas:
  - Possibilitam um alicerce automatizado ou semi-automatizado para o processo e métodos
  - Quando integradas, a informação criada pode ser usada por outra ferramenta
  - Estabelece suporte ao desenvolvimento de software, conhecido como Engenharia de Software Auxiliada por Computador (CASE)

Categorias de Softwares - Pressman, 2016

- 1. Software de Sistema:
  - Conjuntos de programas para auxiliar outros programas
  - Exemplos: compiladores, drivers
- 2. Software de Aplicação:
  - Programas independentes para resolver problemas de negócio
  - Facilitam transações comerciais, decisões administrativas
  - Exemplos: planilhas, editores de texto, software exclusivo da empresa
- 3. Software de Engenharia/Científico:
  - Facilitam entendimento e usam grandes quantidades de cálculos
  - Exemplos: aplicações para meteorologia, astronomia, genética
- 4. Software Embarcado:
  - Programado para funcionar apenas para determinado produto
  - Exemplos: forno micro-ondas, geladeira, módulo de carros

(Cont.)

- 5. Software para Linha de Produtos:
  - Projetado para ser utilizado por diversos clientes
  - Concentra-se em nichos de mercado (com ou sem adaptações)
  - Exemplos: software contábil, software de controle de RH
- 6. Software de Aplicações Web/Aplicativos Móveis:
  - Foco em dispositivos com acesso à internet
  - Voltados a navegadores e softwares residentes em dispositivos móveis
  - Exemplos: TV, celulares, tablets
- 7. Software de Inteligência Artificial:
  - Utiliza algoritmos sofisticados para analisar e solucionar problemas complexos
  - Inclui áreas como robótica, reconhecimento de padrões (voz, imagens, sons), redes neurais
- Software Legado (Pressman, 2016):
  - Softwares antigos que continuam sendo utilizados
  - Apresentam baixa qualidade, muita demora e funcionalidades defasadas
  - Documentação deficitária ou inexistente
  - Processo de manutenção caro e muito demorado
  - Sua evolução ideal é a criação de um novo com tecnologias mais recentes

Princípios da Análise de Sistemas

- Fundamentam-se na necessidade de realizar estudos de processos
  - Para encontrar a melhor solução para a criação de um sistema
- Roth, Dennis e Wixom (2014): análise de sistemas baseia-se em
  - Métodos e técnicas de investigação e especificação
  - Para encontrar a melhor solução para problemas ou necessidades computacionais
  - A partir das funcionalidades levantadas pelo analista de sistemas
- Problemas ao não adotar técnicas de engenharia de software na análise (Engholm Jr., 2010):
  - Softwares com manutenção muito onerosa e demorada
  - Falta de padronização na documentação
  - Falta de controle na sequência do que deve ser realizado
  - Previsões imprecisas de prazo de entrega e preço final

Fases dos Processos da Análise de Sistema (Pressman, 2005)

- 1. Análise:
  - Estudos para especificação do software
  - Verificação de viabilidade (custo-benefício)
  - Definição de funcionalidades e escopo
  - Alocação de recursos e orçamento
- 2. Projeto:
  - Definição lógica do software
  - Elaboração de layouts de telas e relatórios
  - Criação de estrutura de banco de dados e diagramas gráficos
- 3. Implementação:
  - Codificação do software
  - Uso de linguagem de programação definida na fase de análise
- 4. Testes:
  - Objetivo: procura de erros
  - Realização de procedimentos que verificam funcionalidades codificadas

(Cont.)

- 5. Documentação:
  - Documentar todos os processos e diagramas
  - Ferramenta de comunicação entre pessoas envolvidas
  - Parte do contrato entre as partes interessadas
- 6. Manutenção:
  - Acompanhamento do software após implantação
  - Visando registrar e corrigir falhas
  - Propor melhorias ou incluir novas funcionalidades
- Processo de Homologação:
  - Aceite oficial do cliente para validar documentos gerados
  - Alterações após aprovação implicam em mudanças de custos e atrasos

O Papel do Analista de Sistemas: Funções e Responsabilidades

- Possui um papel fundamental nos processos da engenharia de software (Sommerville, 2011)

- Responsável por atividades da análise de sistema, como:
  - Pesquisas e planejamentos
  - Coordenação de equipes de desenvolvimento
  - Recomendação de alternativas de software

- Suas tarefas incluem a criação, implementação e implantação de um software
  - Descobrir o que um sistema deverá fazer
  - Entender e avaliar as necessidades e expectativas de cada usuário
  - Organizar, especificar e documentar essas necessidades

O Analista de Sistemas: Habilidades e Atribuições Chave

- Elmasri e Navathe (2011): desenvolve especificações e funcionalidades, customizando soluções
  - Precisa conhecer um pouco de cada área de negócio
  - Proatividade para buscar conhecimento sobre a área do software
- Uma característica importante é ter uma boa visão empresarial
  - Para ajudar nos processos gerenciais da produção do software
- Habilidades desejáveis:
  - Conhecimento tecnológico atualizado
  - Organização e método
  - Visão gerencial
  - Ótimo relacionamento interpessoal
- É a "ponte" entre programadores e usuários finais
  - Interpreta anseios dos usuários
  - Sabe o que é viável para desenvolvimento

Atribuições Detalhadas do Analista de Sistemas

- Interagir com clientes e usuários finais
- Analisar custos e verificar a viabilidade do projeto
- Levantar informações através de entrevistas e requisitos
- Criar a modelagem do software
- Orientar programadores e acompanhar o desenvolvimento
- Acompanhar e executar testes
- Preparar e acompanhar documentação
- Gerenciar eventuais mudanças no projeto
- Determinar padrões de desenvolvimento
- Garantir a qualidade final e conformidade com o solicitado
- Realizar monitoramento e auditorias
- Planejar e aplicar treinamentos para o uso do software
- Implantar o software, acompanhando adaptação e integração
- Proporcionar consultoria técnica
- Pesquisar novas tecnologias e buscar especializações

Solução para o Cliente Laboratorial: Viabilidade e Tecnologias

- Viabilidade: É mais viável criar um novo sistema
  - O software atual é legado e limitado ao ambiente desktop
  - Não é projetado para funcionar com recursos da internet
  - Empresa com filiais necessita de um sistema central integrado via internet
  - Não é viável usar o sistema legado, mesmo sem avaliar funcionalidades
  - Necessidade de propor um novo sistema que suporte transações online

- Tecnologias a serem utilizadas:
  - Trabalho de investigação necessário (visitar unidade, acompanhar coleta, verificar entrega de resultados)
  - Descobrir linguagem e funcionamento do banco de dados
  - Criação de um site e de um aplicativo que permitam:
    - Marcar exame e agendar coleta em casa
    - Acompanhar análise e visualizar/imprimir resultados

Solução para o Cliente (Cont.): Armazenamento, Linguagem e Processos

- Tecnologias a serem utilizadas (Cont.):
  - Armazenamento do banco de dados em nuvem
    - Consultar grandes players (Amazon, Google, Microsoft) para custos
  - Utilização da linguagem JAVA como sugestão
    - Para diminuir os custos para o cliente
- Perfil dos Profissionais Envolvidos:
  - Equipe de desenvolvimento deve ter analistas de sistemas com as seguintes habilidades:
    - Conhecimento tecnológico atualizado
    - Organização e método
    - Visão gerencial
    - Ótimo relacionamento interpessoal
- Processos para Manter o Software Sempre Atualizado:
  - Adotar métodos de engenharia de software
  - Manter um processo contínuo de manutenção para registrar falhas, propor melhorias e incluir funcionalidades
  - Gerenciar eventuais mudanças (ex: uso de códigos de barras, QR-Code)