RELATÓRIO CIENTÍFICO: FUNDAMENTOS DA ENGENHARIA DE SOFTWARE E ANÁLISE DE SISTEMAS

RESUMO

Este relatório aborda os fundamentos da Engenharia de Software e da Análise de Sistemas, disciplinas cruciais para o desenvolvimento tecnológico contemporâneo. Discute a evolução do software, sua importância estratégica e suas características intrínsecas, diferenciando-o do hardware. Apresenta os conceitos essenciais da Engenharia de Software e da Análise de Sistemas, delineando suas atividades, tipos de software e o papel do analista de sistemas. Em seguida, explora os diversos modelos de processos de software, categorizando-os em prescritivos (Cascata, Incremental, Evolucionários), especializados (como RUP) e ágeis (XP e Scrum), detalhando suas características, vantagens e aplicações. A compreensão desses modelos é destacada como fundamental para o sucesso na produção de software de qualidade.

Palavras-chave: Engenharia de Software; Análise de Sistemas; Modelos de Processos de Software; Metodologias Ágeis; Software.

1 INTRODUÇÃO

A tecnologia permeia cada vez mais o cotidiano da sociedade, com avanços em áreas como computação em nuvem, realidade virtual e reconhecimento facial. Neste cenário, o software emerge como um componente fundamental, sendo a "peça-chave" que capacita o funcionamento do hardware e impulsiona a inovação. A complexidade e a ubiquidade dos sistemas modernos exigem uma abordagem estruturada e disciplinada para o seu desenvolvimento.

Este relatório, fundamentado nas discussões apresentadas no livro "Introdução à Engenharia de Software e à Análise de Sistemas" de Claudia Werlich, visa explorar a evolução histórica do software, sua importância, as distinções entre software e sistema, e as disciplinas de Engenharia de Software e Análise de Sistemas. Adicionalmente, serão detalhados os principais modelos de processos de software, essenciais para a gestão eficaz de projetos e a produção de soluções tecnológicas robustas e confiáveis.

2 A EVOLUÇÃO E A IMPORTÂNCIA DO SOFTWARE

O software passou por uma notável evolução, desde programas rudimentares que controlavam diretamente o hardware nos anos 1940, passando pela era da internet e da linguagem Java nos anos 1990, até a atual era da **computação em nuvem, aplicativos móveis e Inteligência Artificial (IA)**. Essa trajetória demonstra a crescente complexidade e a relevância estratégica que o software adquiriu.

Uma característica intrínseca do software é que ele **não se "desgasta" fisicamente como o hardware**, mas pode "deteriorar" ao longo do tempo devido a mudanças e manutenções, gerando novos erros. As Leis de Lehman, formuladas nos anos 1980, corroboram essa observação, indicando que o software demanda **mudança contínua** para não se tornar obsoleto, e que cada alteração tende a elevar sua complexidade. É crucial distinguir software de sistema: o **software** refere-se ao programa e suas instruções, enquanto o **sistema** constitui um conjunto mais abrangente, englobando software, hardware e a interação humana para alcançar um objetivo específico.

3 ENGENHARIA DE SOFTWARE E ANÁLISE DE SISTEMAS

Diante da crescente dependência de software confiável e da necessidade de otimização de custos no desenvolvimento, a **Engenharia de Software** surge como uma disciplina essencial. Ela se concentra na **qualidade, processo, métodos e ferramentas** empregadas na produção de software. As atividades primordiais da Engenharia de Software incluem **comunicação, planejamento, modelagem, construção e entrega**.

Os tipos de software são variados, abrangendo desde software de sistema (como compiladores e drivers) e de aplicação (editores de texto), até software embarcado (presente em dispositivos como micro-ondas), de engenharia/científicos, e, de forma proeminente, as **aplicações web/móveis e as baseadas em Inteligência Artificial**. Um desafio persistente na área é o **software legado**, que se refere a sistemas antigos e defasados, frequentemente carentes de documentação adequada ou completa, tornando sua manutenção custosa e problemática, e muitas vezes recomendando sua substituição por soluções mais modernas.

A **Análise de Sistemas** representa a etapa fundamental de investigação e especificação, visando identificar a solução mais apropriada para um problema de negócio. Segundo Pressman (2005), as fases principais da Análise de Sistemas são: **Análise, Projeto, Implementação, Testes, Documentação e Manutenção**. É imperativo que, durante essa fase, o domínio da informação seja claramente estabelecido, as funcionalidades sejam bem definidas e o comportamento do software seja representado por meio de diagramas.

O **Analista de Sistemas** desempenha um papel de "ponte" entre os desenvolvedores e os clientes. Suas responsabilidades incluem pesquisa, planejamento, coordenação de equipes, levantamento de requisitos, modelagem do software, acompanhamento do desenvolvimento e testes, gerenciamento de mudanças, garantia de qualidade e, por vezes, treinamento de usuários. Para tanto, esse profissional deve possuir organização, visão gerencial e manter-se continuamente atualizado tecnologicamente.

4 PROCESSOS E MODELOS DE PROCESSOS DE SOFTWARE

Um **Processo de Software** é definido como um conjunto de atividades inter-relacionadas que culminam na produção de um software. A adoção de um processo padroniza a geração de produtos, facilita a reutilização de componentes, preserva o conhecimento institucional e mitiga riscos. As atividades fundamentais de um processo genérico são: **especificação, projeto e implementação, validação e evolução**. Pressman (2016) detalha essas atividades como comunicação, planejamento, modelagem, construção e entrega.

Os **Modelos de Processos de Software** são guias que estruturam o fluxo de atividades, as interações entre elas, os artefatos produzidos e a organização do trabalho. Eles são cruciais para conferir estabilidade, controle e organização ao ciclo de desenvolvimento. Existem três categorias principais de modelos:

4.1 Modelos Prescritivos (ou Tradicionais)

Caracterizados por serem sequenciais e possuírem diretrizes bem definidas.

- **Cascata:** Considerado o modelo mais antigo e sistemático. Nele, cada fase (Análise, Projeto, Implementação, Testes, Manutenção) é concluída antes que a próxima se inicie. Apesar de sua simplicidade de gerenciamento, pode ser lento, e o cliente só visualiza o produto final após um longo período.
- **Incremental:** Este modelo foca na entrega de versões funcionais parciais do software em "incrementos". O cliente fornece feedback sobre cada incremento, permitindo que o software evolua progressivamente com novas versões.
- **Evolucionários:** Produzem versões do software que se tornam progressivamente mais completas.
    - **Prototipação:** Envolve a criação de uma versão inicial (protótipo) para ser testada pelo cliente, possibilitando feedback rápido e auxiliando na compreensão dos requisitos.
    - **Espiral:** Combina a natureza iterativa da prototipação com o controle do modelo Cascata, enfatizando o gerenciamento de riscos em cada ciclo.

4.2 Modelos Especializados

Utilizam características dos modelos prescritivos, adaptando-as para abordagens específicas. Exemplos incluem os Modelos Baseados em Componentes (que promovem a reutilização de partes do software), e o **Processo Unificado (RUP)**, que se destaca por sua natureza iterativa e incremental.

4.3 Modelos de Desenvolvimento Ágil

Surgem como uma resposta à rápida mudança tecnológica, priorizando flexibilidade e entregas rápidas. Os princípios ágeis incluem o **envolvimento contínuo do cliente, entregas incrementais, a valorização de indivíduos e interações sobre processos e ferramentas, software funcionando sobre documentação abrangente, colaboração com o cliente sobre negociação de contratos, e resposta a mudanças sobre seguir um plano**.

- **XP (Extreme Programming):** Enfatiza o feedback constante, desenvolvimento incremental, comunicação ativa, programação em pares e a presença do cliente na equipe de desenvolvimento.
- **Scrum:** É um *framework* iterativo e incremental amplamente utilizado para gestão de projetos. Seus componentes incluem o **Backlog** (uma lista priorizada de requisitos), **Sprints** (períodos curtos de trabalho que resultam em uma entrega funcional) e **Reuniões Scrum diárias (Daily Meetings)**, focadas no progresso, obstáculos e planos. As funcionalidades iniciais são frequentemente descritas como "Histórias", e

um **Quadro Scrum** (com categorias como "To Do", "In Progress", "Testing", "Done") auxilia na organização visual do fluxo de trabalho.

Todo o conceito dos modelos ágeis é fundamentado no **Manifesto Ágil**, que estabelece esses valores e princípios para uma abordagem mais adaptativa ao desenvolvimento de software.

5 CONCLUSÃO

A Engenharia de Software e a Análise de Sistemas são pilares indispensáveis na construção de soluções tecnológicas no cenário atual. A compreensão da evolução do software, suas características únicas e a distinção entre software e sistema é crucial. Ambas as disciplinas fornecem as ferramentas e metodologias para transformar ideias em produtos funcionais e de qualidade.

A escolha do modelo de processo de software mais adequado é um fator determinante para o sucesso de qualquer projeto. Não existe um modelo universalmente "melhor", mas sim aquele que melhor se alinha ao tipo de software a ser desenvolvido e às características da equipe envolvida. Independentemente do modelo adotado, a **comunicação eficaz e o trabalho em equipe** são considerados indispensáveis para a entrega de um produto de sucesso. A constante atualização sobre as tendências e metodologias é vital para os profissionais da área, garantindo a capacidade de inovar e responder aos desafios do mercado.

REFERÊNCIAS

WERLICH, Claudia. Introdução à Engenharia de Software e à Análise de Sistemas. *In*: **Documento sem título.pdf**. [S. l.]: [s. n.], [entre 2005 e 2016]. (Informação extraída de excertos do documento fornecido).

PRESSMAN, Roger S. *Engenharia de Software*: uma abordagem profissional. [S. l.]: [s. n.], 2005.

PRESSMAN, Roger S. *Engenharia de Software*: uma abordagem profissional. [S. l.]: [s. n.], 2016.