PROCESSO DE *SOFTWARE*

Allan Senna Costa dos Santos
Discente do Curso Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas (AEMS)

Jhonatan Ricardo Ferraris da Silva
Discente do Curso Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas (AEMS)

Alan Pinheiro de Souza
Docente do Curso Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integradas de Três Lagoas (AEMS)
Mestre em Informática
Área de Sistemas de Informação

André Aparecido Leal de Almeida
Docente do curso Engenharia da Computação
Faculdades Integradas de Três Lagoas (AEMS)

RESUMO
Processo de *software* está associado à engenharia de *software* que tem como objetivo obter resultados satisfatórios dentro do desenvolvimento de *software* com qualidade e cumprindo custos e prazos previstos. Este trabalho busca demostrar como é feito o processo de *software* e seus benefícios com isso mostrando alguns de seus modelos cada um com a suas características específicas que mostram como se desenvolver um produto da melhor maneira possível para se atingir o resultado esperado pelo contratante do serviço. Portanto, realizou-se um levantamento bibliográfico visando identificar um possível panorama do desenvolvimento de *software* com qualidade e estabilidade.

PALAVRAS-CHAVE: Processo de *Software*; Modelos; Qualidade; *Software.*

INTRODUÇÃO

Engenharia de *software* é uma abordagem disciplinada para o desenvolvimento de *softwares* com o objetivo de construí-los com alta qualidade e de se evitar problemas tradicionais de desenvolvimento como atraso na entrega e estrapolação de custos. Nesse contexto, o conceito de processo é importante para estabelecer uma prática de trabalho organizada, controlada, padronizada e com a otimização de recursos.

As pessoas envolvidas em desenvolver um produto dentro de uma organização adaptam o seu processo com características especificas para se chegar a um produto final com maior qualidade e estabilidade. Mesmo existindo processos de *software* diferentes, há quatro atividades fundamentais entre todos os processos. Segundo Sommerville (2003, p.36), essas atividades são: especificação de *software*, projeto e implementação de *software,* validação de *software* e evolução de *software*.

O trabalho está estruturado em duas seções. A primeira seção discursa sobre o processo de *software* e sua importância. A segunda seção mostra os modelos de processo. As seções finais apresentam as considerações finais e as referências bibliográficas adotadas na pesquisa.

1. PROCESSO DE *SOFTAWARE*

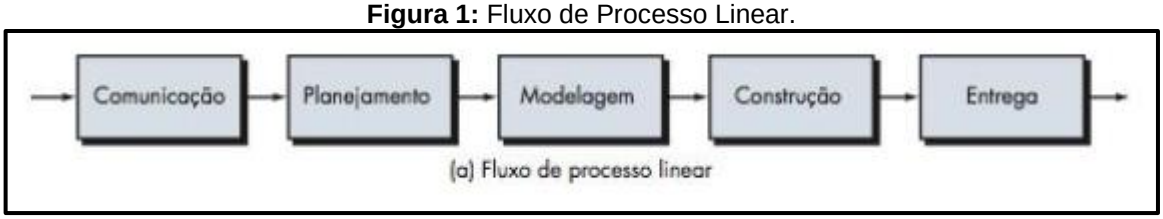
Em uma visão geral, processos podem ser definidos como um conjunto de passos parcialmente ordenados usados para atingir uma meta que está ligada a um ou mais resultados finais que são produtos da execução do processo. Na visão de Paula Filho (2000, p.23), um processo é definido quando tem documentação que detalha: o que é feito (produto), quando (passos), por quem (agentes), as coisas que usa (insumos) e as coisas que produz (resultados).

Um processo é um conjunto de etapas que envolvem atividades, restrições e recursos para alcançar uma saída desejada. A sua importância esta na consistência e estrutura agregadas a um conjunto de atividades. Isso se torna útil quando é conhecido como fazer algo bem e deseja-se que outras pessoas executem igualmente (PFLEEGER, 2004 *apud* CABRAL; SILVA; SOUZA, 2014, p.122).

Segundo Pressman (2011, p.53), uma metodologia de processo genérica estabelece cinco atividades metodológicas: comunicação, planejamento, modelagem, construção e entrega. Além disso, mais um conjunto de atividades de apoio, as atividades guarda-chuva, elas são aplicadas ao longo do projeto e ajudam a administrar riscos e garantir a qualidade. Um fluxo de processo descreve como são organizadas essas cinco atividades metodológicas, conforme pode ser observado em (PRESSMAN, 2011, p.54):

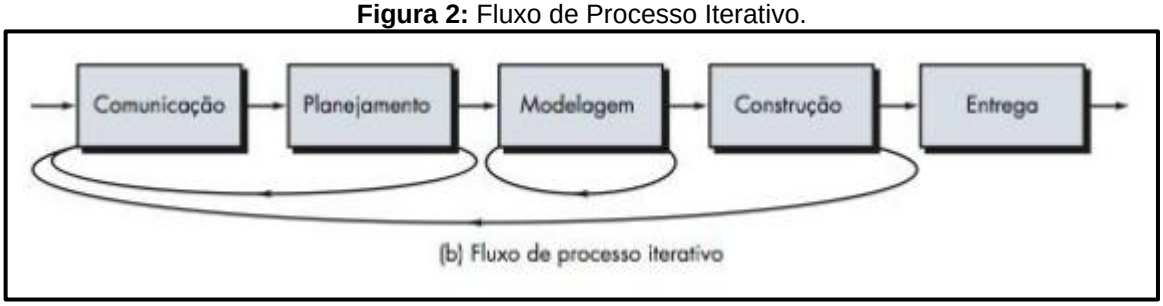
- **Linear:** executa as cinco atividades de forma sequencial começando com a comunicação e terminando na entrega como mostra a (Figura 1);
- **Iterativo:** repete um ou mais dos cinco passos antes de ir em frente como apresenta a (Figura 2);
- **Evolucionário:** executa as atividades de forma circular e a cada volta traz uma versão mais completa do *software* como é indicado na (Figura 3);
- **Paralelo:** executa uma ou mais atividades em paralelo como exibido na (Figura 4).

Figura 1: Fluxo de Processo Linear.



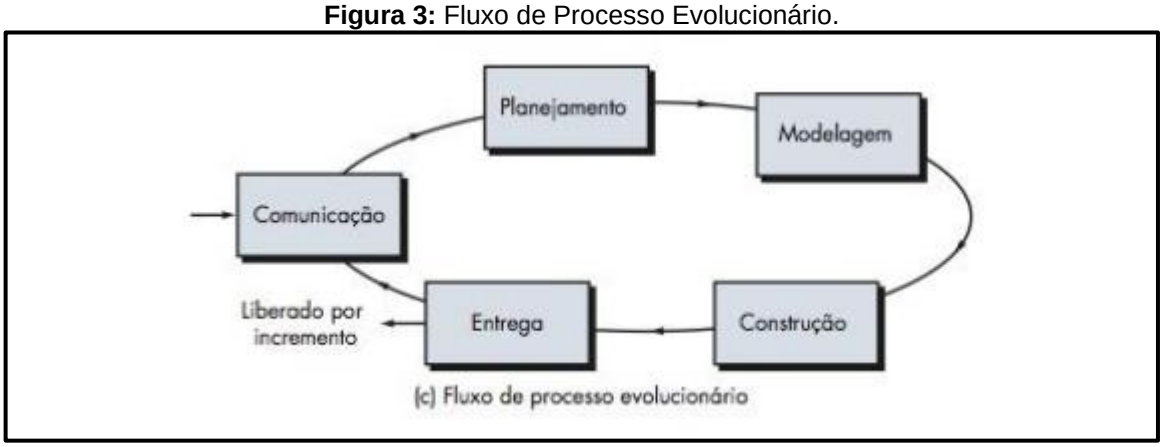
Fonte: Retirado de Pressman (2011, p.54).

Figura 2: Fluxo de Processo Iterativo.



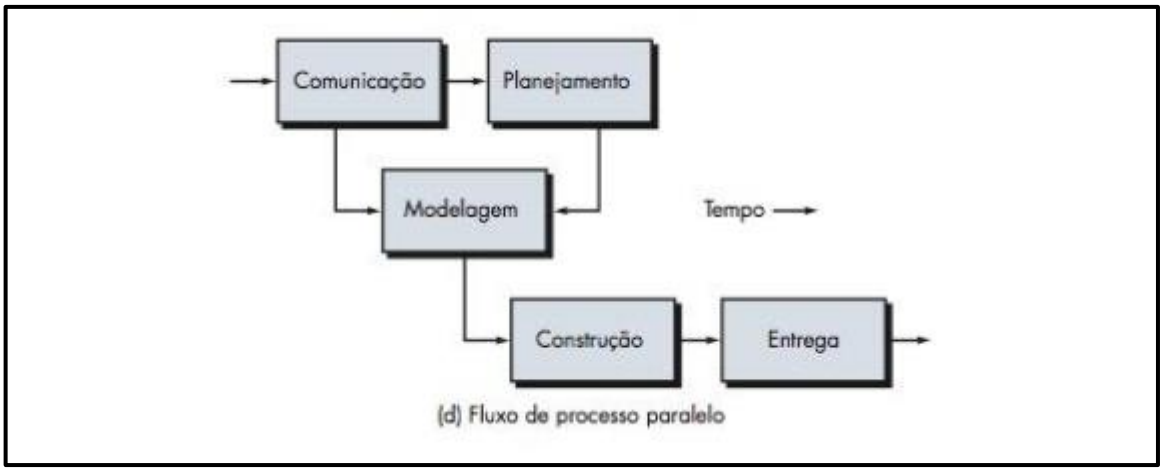
Fonte: Retirado de Pressman (2011, p.54).

Figura 3: Fluxo de Processo Evolucionário.



Fonte: Retirado de Pressman (2011, p.54).

Figura 4: Fluxo de Processo Paralelo.



Fonte: Retirado de Pressman (2011, p.54).

Esses fluxos de processo são usados dentro da ideia de modelos de processo que foram criados para auxiliar os desenvolvedores a ter uma base para se começar a projetar seu *software*. Segundo Pressman (2011, p.59), esses modelos tradicionais proporcionaram uma considerável contribuição quanto à estrutura utilizável no trabalho de engenharia de *software* e forneceram um roteiro razoavelmente eficaz para as equipes de *software*.

2. MODELOS DE PROCESSO

Em uma visão geral, modelos de processos podem ser vistos como um esboço dos objetos e atividades envolvidas no processo de *software*, oferecendo uma forma mais fácil de representar o processo de *software* e a evolução do projeto De acordo com Sommerville (2004, p.36), os processos de *software* são muito complexos, além disso, existe uma grande diversidade de processos de *software*, não há um processo ideal para se desenvolver os projetos, e cada modelo tem suas vantagens e desvantagens. Embora existam todas estas diferenças entre os modelos há quatro atividades fundamentais entre todos eles que são:

- **Especificação de *software*:** onde define a funcionalidade do *software* e restrições em sua operação;
- **Projeto e implementação de *software*:** deve-se desenvolver o *software* da forma que cumpra tudo o que foi especificado;

- **Validação de *software*:** precisa ser validado para mostrar autenticidade e mostrar para o cliente que cumpre tudo o que deseja;
- **Evolução de *software*:** o *software* precisa ter sempre atualizações para melhor atender o cliente.

Segundo Pressman (2011, p.59), todos os modelos podem conter estas quatro atividades, porém, cada um deles da ênfase diferente a elas e define um fluxo de processo que chama cada atividade, seguindo esse contexto alguns desses modelos são:

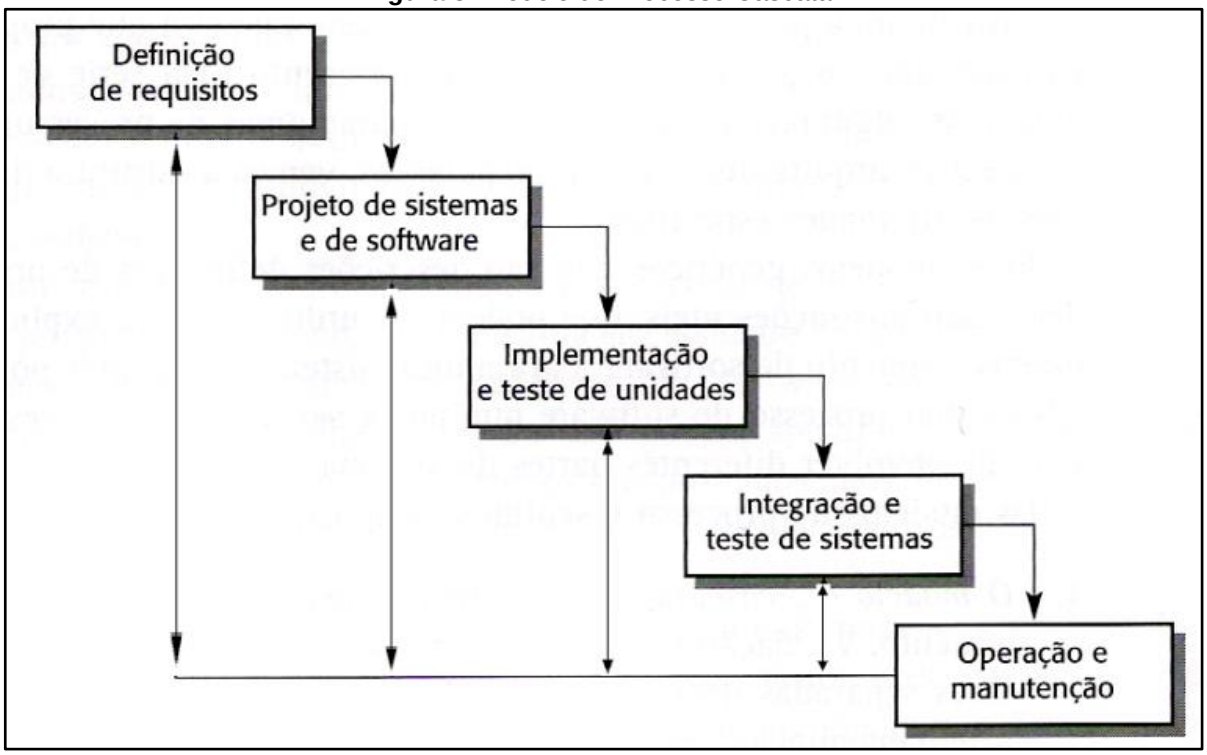
✓ Modelo cascata

Segundo Rezende (2005, p.130), no modelo cascata, os processos obedecem a uma sequência obrigatória de desenvolvimento de *software*, ele não permite o retorno nas sequencias tornando o desenvolvimento linear assim podendo ter consequências ruins para o usuário final como pode ser observado na (Figura 5). Segundo Sommerville (2004, p.37-38), os principais estágios do modelo retratam as atividades de desenvolvimento fundamentais:

- **Análise e Definição de Requisitos:** as funções, as restrições e os objetivos do sistema são estabelecidos pro meio da consulta aos usuários do sistema. Em seguida, são definidos em detalhes e servem como uma especificação do sistema;
- **Projeto de Sistemas e de *Software*:** o processo de projeto de sistemas agrupa os requisitos em sistemas de *hardware* ou de *software*. Ele estabelece uma arquitetura do sistema geral. O projeto de *software* envolve a identificação e a descrição das abstrações fundamentais do sistema de *software* e suas relações;
- **Implementação e Teste de Unidades:** durante esse estágio, o projeto de *software* é compreendido como um conjunto de programas ou unidades de programa. O teste de unidades envolve verificar que cada unidade atenda a sua especificação;
- **Integração e Teste de Sistemas:** as unidades de programa ou programas individuais são integrados e testados como um sistema completo a fim de garantir que os requisitos de *software* foram atendidos. Depois dos testes, o sistema de *software* é entregue ao cliente;

- **Operação e Manutenção:** Normalmente, esta é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em operação. A manutenção envolve corrigir erros que não foram descobertos em estágios anteriores do ciclo de vida, melhorando a implementação das unidades de sistema e aumentando as funções desse sistema à medida que novos requisitos são descobertos.

Figura 5: Modelo de Processo Cascata.



Fonte: Retirado de Sommerville (2004, p.38).

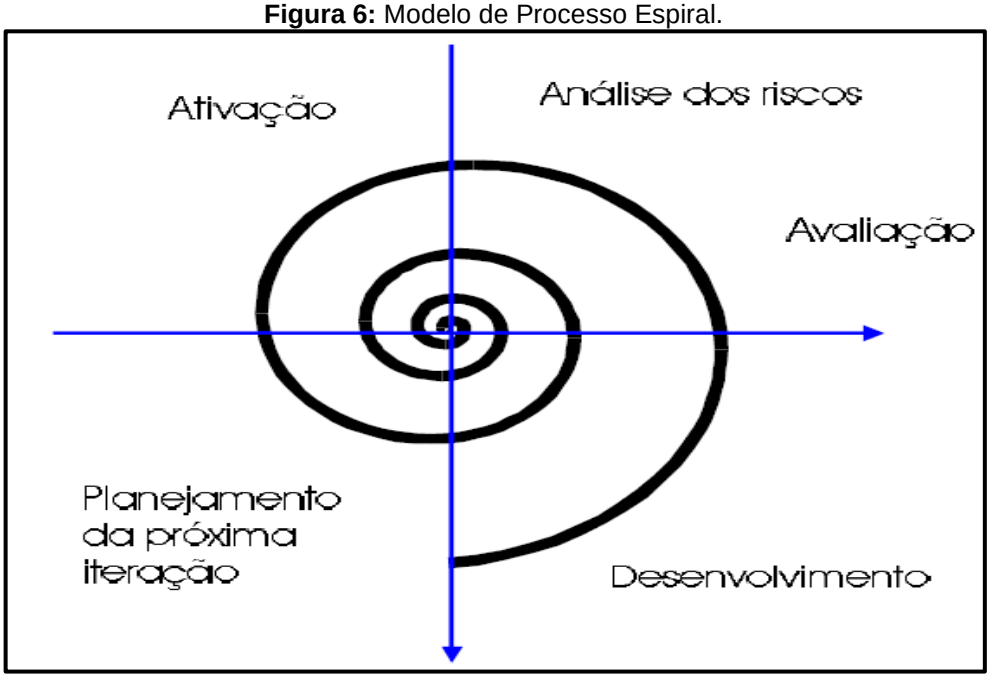
✓ Modelo espiral

Segundo Pressman (2011, p.65), o espiral é um modelo de processo evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos do modelo cascata como mostra a (Figura 6). Além disso, o modelo espiral possui potencial para desenvolver rapidamente versões mais completas do *software*.

O processo espiral se move a partir do seu centro no sentido horário. Cada rodada representa uma versão melhorada do projeto apresentado na rodada anterior. Assim, pode-se adaptar o processo espiral ao processo completo de um *software,* conforme pode ser observado em (HIRAMA, 2012, p.33).

Na visão de Paula Filho (2000, p.25), o modelo espiral permite a construção do produto com prazo curto, novas características e recursos que são agregados na medida em que a experiência descubra a sua necessidade. O principal problema dessa abordagem é o requerimento de uma gestão muito sofisticada para que o modelo seja previsível e confiável.

Figura 6: Modelo de Processo Espiral.



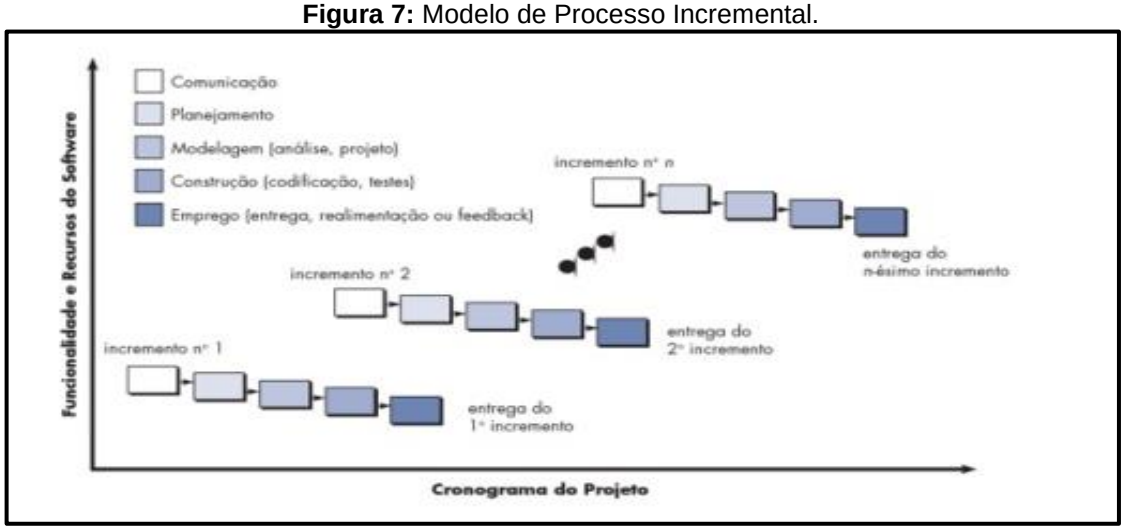
Fonte: Retirado de Paula Filho (2000, p.25).

✓ Modelo de processo incremental

De acordo com Ramos (2006, p.42), processo incremental corresponde à ideia de “aumentar (ou alagar) pouco-a-pouco” o âmbito do sistema. Como por exemplo, uma mansão que foi construída a partir de incrementos em uma primeira casa com apenas algumas divisões.

O modelo incremental combina elementos dos fluxos de processo lineares e paralelos na visão de Pressman (2011, p.61), ele aplica uma sequencia linear, de forma escalonada, à medida que o tempo avança ele gera um novo incremento como é indicado na (Figura 7) de um modo parecido aos incrementos gerados por um fluxo de processos evolucionários.

Figura 7: Modelo de Processo Incremental.



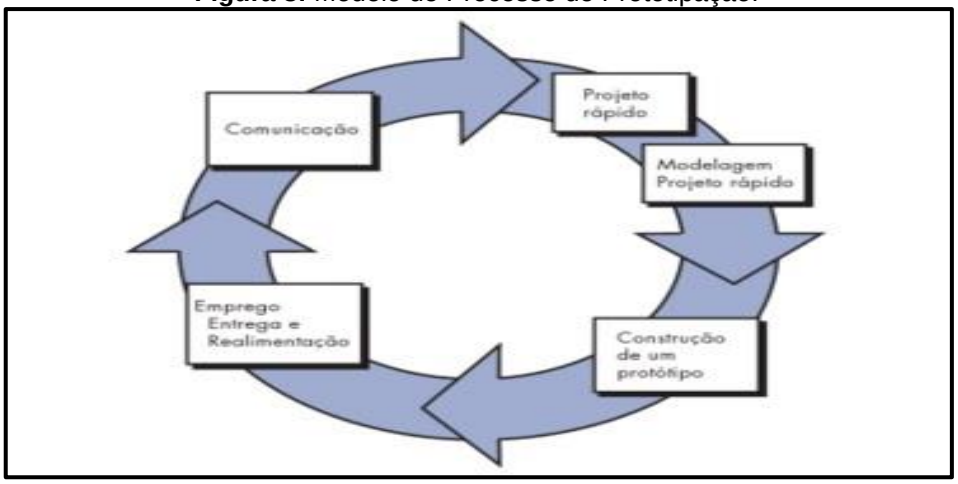
Fonte: Retirado de Pressman (2011, p.61).

✓ Modelo de prototipagem evolutiva

De acordo com Paula Filho (2000, p.25), neste modelo a espiral é usada não para desenvolver o produto completo, mas para construir uma série de versões provisórias que são chamadas de protótipos. Os protótipos cobrem cada vez mais requisitos, até que se atinja o produto desejado.

Segundo Pressman (2011, p.63), o modelo começa com uma reunião com os envolvidos para definir objetos gerais do *software*, identificar as necessidades já conhecidas e esquematizar as áreas que ainda faltam definições. Uma interação de prototipagem é planejada rapidamente e ocorre a modelagem. Um projeto rápido leva à construção do protótipo a ser implantado e avaliado pelo cliente, conforme visto na Figura 8.

Figura 8: Modelo de Processo de Prototipação.



Fonte: Retirado de Pressman (2011, p.63).

CONSIDERAÇÕES FINAIS

Apesar dos métodos e pesquisas sobre o desenvolvimento de software com qualidade, ainda existe *softwares* sendo entregues com defeito sem qualidade e extrapolando os prazos previstos, Mas o processo de *software* parece ser uma forma de resolver esses problemas dando uma “luz” para como se desenvolver um produto com qualidade e eficiência.

Mesmo com os processos de *software*, os problemas ainda continuam e já existem de longa data, talvez ainda seja necessária uma maior evolução nos métodos usados no desenvolvimento de *software* levando a mais algum tempo para que se ache uma solução definitiva.

O processo de *software* tem como objetivo evitar problemas que ocorrem no desenvolvimento de software nas empresas, por exemplo, extrapolação nos custos, tempo para entrega do projeto, perda de qualidade e muitos outros, Um processo padroniza etapas no desenvolvimento para que o produto final tenha um resultado satisfatório com qualidade, estabilidade e durabilidade.

Uma dessas etapas é a escolha de um modelo de processo a ser seguido. Cada um deles tem suas características especificas assim tornando-se necessário uma análise minuciosa nas abordagens existentes para que se tenha uma certeza de qual melhor modelo irá se encaixar no processo de desenvolvimento do produto pretendido e, se necessário, ocorrer modificações no modelo escolhido para uma melhor efetividade na sua aplicação.

Dessa forma, espera-se obter uma solução para que se chegue a um produto final no prazo e com qualidade e estabilidade desejados. O não alcance desses objetivos é um problema constante no processo de desenvolvimento de *software*.

REFERÊNCIAS BIBLIOGRÁFICAS

CABRAL, A.; SILVA, D; SOUZA, A. **A Problemática do Desenvolvimento de *Software*: Crise ou Calamidade Crônica?** Revista Conexão Eletrônica, p.118-125, 2014.

HIRAMA, K. **Engenharia de *Software*, Qualidade e Produtividade com Tecnologia.** Rio de Janeiro: Elsevier Editora, 2012.

PAULA FILHO, W. **Engenharia de *Software*: Fundamentos, Métodos e Padrões.** 2ª Ed., Rio de Janeiro: LTC, 2000.

PFLEEGER, S. **Engenharia de Software: Teoria e Prática.** 2ª Ed., São Paulo: Pearson Prentice Hall, 2004.

PRESSMAN, R. **Engenharia de *Software*: Uma Abordagem Profissional.** 7ª Ed., São Paulo: McGraw-Hill, 2011.

RAMOS, R. **Treinamento Prático em UML.** São Paulo: Digerati Books, 2006.

REZENDE, D. **Engenharia de *Software* e Sistemas de Informação.** 3ª Ed., Rio de Janeiro: Brasport Livros e Multimídia, 2005.

SOMMERVILLE, I. **Engenharia de *Software*.** 6ª Ed., São Paulo: Addison Wesley Brasil, 2003.