# A PROBLEMÁTICA DO DESENVOLVIMENTO DE SOFTWARE: CRISE OU CALAMIDADE CRÔNICA?

ADEMILSON ANGELO CABRAL
Discente da AEMS
Faculdades Integradas de Três Lagoas

DIEGO BEZERRA DA SILVA
Discente da AEMS
Faculdades Integradas de Três Lagoas

ALAN PINHEIRO DE SOUZA
Docente da AEMS
Faculdades Integradas de Três Lagoas
Mestre em Informática
Área de Sistemas de Informação

A utilização do termo “crise” associado ao processo de desenvolvimento de *software* teve início na década de 60 e tornou-se mais recorrente na década seguinte. Considerando que a engenharia de *software* foi criada para combater as causas que levaram à criação da expressão “crise de *software*”, este trabalho busca reconhecer origens da problemática e debater como a engenharia de *software* busca alcançar o seu objetivo e reconhecer caminhos adotados para superar a crise. Portanto, realizou-se um levantamento bibliográfico visando identificar um possível panorama do desenvolvimento de *software* nos últimos anos. As conclusões da pesquisa apontam avanços no processo de desenvolvimento de *software*, porém, existem ainda problemas a serem superados.

Palavras-Chaves: crise, engenharia de *software*, modelagem, qualidade, *software*.

## 1. INTRODUÇÃO

Nos anos 60, a expressão “crise de *software*” começou a ser utilizada para denominar o conjunto de problemas encontrados no processo de desenvolvimento de *software*. Entretanto, mesmo com o surgimento da engenharia de *software*,

muitos desses problemas persistiram e são recorrentes até nos dias atuais. Esse trabalho realizou um levantamento bibliográfico de modo a estabelecer panorama a respeito da situação do desenvolvimento de *software* ao longo das últimas décadas.

O trabalho está estruturado em quatro seções. A primeira seção destaca a introdução de modo a passar uma visão geral do projeto e organização do trabalho. A segunda seção discursa sobre a crise de *software* e suas causas. A terceira seção aborda os objetivos de *software* e suas contribuições para superar as causas da crise. A quarta seção mostra as conclusões da pesquisa.

## 2. CRISE DE SOFTWARE

*Software* de computador é o produto que profissionais de *software* constroem e, depois, mantêm ao longo do tempo (PRESSMAN, 2011). Esse produto abrange programas executáveis, código-fonte, documentação na forma impressa ou virtual, entre outros. O desenvolvimento de *software* não é uma tarefa trivial.

À medida de sua evolução, vários problemas não previstos surgiram, e comprometeram a qualidade do processo de desenvolvimento e do produto final. Segundo (REZENDE, 2005), a crise de *software* ocorre quando o sistema não satisfaz todos os envolvidos no projeto, sejam clientes, desenvolvedores e/ou usuários. Os problemas podem ocorrem em diferentes etapas do processo de desenvolvimento: análise, projeto, construção, implantação ou manutenção.

Na literatura, diversos autores mencionam o termo “crise de *software*”. Alguns autores, tal como, Pressman (2011), complementam que essa crise acompanha o processo de desenvolvimento há mais de 40 anos. Essa argumentação gera uma contradição de termos, pois esse cenário extrapola a configuração de uma crise. Na verdade, existe uma calamidade crônica associada ao processo de *software*.

O estudo *Chaos Report* da *Standish Group* realizou um levantamento a cerca de projetos na área de tecnologia da informação. Esses dados são utilizados com frequência para quantificar a crise do *software*. O resultado final dos projetos é classificado como fracasso, modificado ou sucesso. Na Figura 1 é mostrada a distribuição do resultado final desses projetos, na qual observa-se que uma minoria dos esforços de desenvolvimento de tecnologias consegue ser completada com sucesso. Apesar de avanços, esse cenário negativo é recorrente há quase duas décadas.

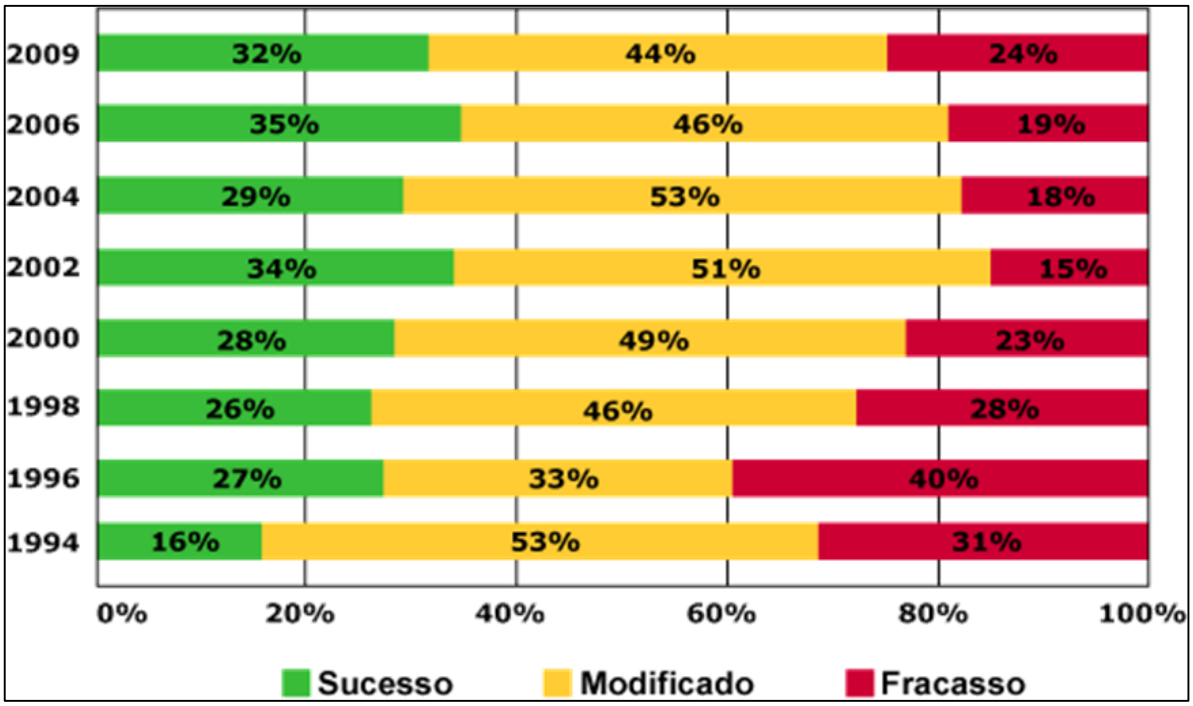


**Figura 1 - Distribuição do resultado final de projetos na área de tecnologia.**

Segundo as informações do *Gartner Group Research*, os principais problemas do desenvolvimento de *software* são aqueles apontados na sequência. Na Figura 2 encontra-se a porcentagem de ocorrência desses problemas.

- ☑ Cumprimento dos prazos;
- ☑ Custos elevados;
- ☑ Manutenção de sistemas em uso;
- ☑ Conflito entre desenvolvimento e manutenção;
- ☑ Qualificação dos profissionais.


**Figura 2 - Principais problemas do desenvolvimento de *software*.**

Outros problemas comuns são sistemas desenvolvidos incorretamente a partir de especificações corretas; sistemas desenvolvidos corretamente a partir de especificações erradas ou incompletas; e corte deliberado no escopo do projeto por limitações de prazo ou verba (SOMMERVILLE, 2007; PAULA FILHO, 2000). *Software* de má qualidade está presente em todas as organizações que usam computadores, provocando horas de trabalho perdidas, dados de negócios são perdidos ou corrompidos, oportunidades de vendas desperdiçadas, com elevados custos com manutenção e baixa satisfação dos clientes.

Por um lado, clientes culpam os desenvolvedores, argumentando que práticas descuidadas levam a um *software* baixa qualidade. Os desenvolvedores culpam os clientes e outros interessados, argumentando que datas de entregas absurdas e um fluxo contínuo de mudanças os forçam a entregar *software* antes de estarem completamente validados. Esses cenários não estão associados somente aos programas que não funcionam, abrangem sistemas que estão sendo construídos e implantados, além de projetos que foram descontinuados.

Segundo Brooks (1987), a compreensão dos fatores que levam ao estabelecimento de um cenário de crise necessita da compreensão da natureza do *software* e como o seu desenvolvimento vem sendo tratado ao longo dos anos. Essa análise envolve aspectos intrínsecos que caracterizam o *software*, por exemplo, complexidade; necessidades de modificações; adaptabilidade; rápida evolução tecnológica; baixo custo de manufatura, entre outros. A maioria das ferramentas ou tecnologias não produziu aumento significativo na produtividade do desenvolvimento de *software*, pelo fato de não resolverem problemas associados a alguns desses aspectos básicos do *software*.

## 3. ENGENHARIA DE SOFTWARE

Há décadas a crise do *software* instiga e inspira pesquisadores e profissionais a criarem propostas de soluções visando combatê-la e tornar projetos de criação de *software* mais produtivos e previsíveis; e com resultados de melhor qualidade. Uma dessas propostas é a engenharia de *software*. Segundo o IEEE (*Institute of Electrical and Electronic Engenieering*), a engenharia de *software* é a aplicação sistemática, disciplinada e com abordagem quantitativa para desenvolvimento, operação e manutenção de *software*.

Na visão de Rezende (2005), a engenharia de *software* é uma metodologia que desenvolve soluções profissionais utilizando-se dos recursos de *software*, observando os padrões requeridos de qualidade, produtividade e efetividade. Pode ser compreendida como uma disciplina que reúne metodologias, métodos e ferramentas a serem utilizados, desde a percepção do problema até o momento que o sistema desenvolvido deixa de ser operacional, visando resolver problemas inerentes ao processo de desenvolvimento e ao produto de *software* (CARVALHO; CHIOSSI, 2001).

Para o desenvolvimento de um *software* de qualidade, a engenharia de *software* propõe a utilização de processos, modelagem e gestão da qualidade. Esses conceitos fornecem um roteiro e técnicas para criação de *software* de qualidade. Na sequência, cada um desses conceitos e técnicas será discutido para o alcance de um entendimento amplo sobre a possibilidade de alcance de soluções:

- ✓ **Processos de *Software***

Na visão de Pfleeger (2004), um processo é uma série de etapas que envolvem atividades, restrições e recursos para alcançar uma saída desejada. A sua importância está na consistência e estrutura agregadas a um conjunto de atividades. Essas características são úteis quando é conhecido como fazer algo bem e deseja-se garantir que outras pessoas o façam da mesma maneira. Além disso, os processos permitem capturar experiências e passá-las adiante.

No contexto da engenharia de *software*, um processo não é uma prescrição rígida de como desenvolver um *software*. Ao contrário, é uma abordagem adaptável que possibilita a equipe de *software* escolher o conjunto apropriado de ações e tarefas para o desenvolvimento com qualidade e satisfação daquelas pessoas envolvidas no projeto e/ou que utilizarão o sistema quando finalizado.

- ✓ **Modelagem de *Software***

O objetivo da modelagem é a criação de modelos para uma melhor compreensão do que será realmente construído. Os modelos são úteis para elaboração da estrutura do *software*, além de permitir visualização, especificação, construção e documentação dos artefatos de projeto (DEBONI, 2003).

A construção de diagramas contribui essencialmente para desenvolver um *software* que satisfaça aos propósitos pretendidos pelo cliente e com qualidade

duradoura (BOOCH; RUMBAUGH; JACOBSON, 2000; FURLAN, 1998). Além disso, a modelagem agrega outros benefícios como a possibilidade de desenvolver um sistema de forma mais rápida e eficiente, com o mínimo de desperdício e retrabalho de *software* (FOWLER, 2005; MATOS, 2002).

Os modelos de *software* devem ser capazes de apresentar as informações que o *software* transforma, a arquitetura e as funções que possibilitam as transformações, as características que os usuários desejam e o comportamento do sistema na medida em que as transformações acontecem. Esses modelos devem cumprir objetivos em diferentes níveis – primeiro, descrevendo o *software* ao ponto de vista do cliente e, posteriormente em um termo mais técnico.

- ✓ **Gestão da qualidade**

O clamor por qualidade de *software* começou quando as empresas perceberam que estavam tendo prejuízos com *software* em razão do não cumprimento com as funcionalidades e características prometidas para os sistemas junto aos clientes (PRESSMAN, 2011; PAULA FILHO, 2000).

Em um nível mais pragmático, Garvin (1984) sugere que a qualidade é um conceito complexo e multifacetado e pode ser descrito segundo diferentes visões:

- **Visão transcedental:** sustenta que qualidade e algo que se reconhece imediatamente, mas que não consegue ser definida explicitamente;
- **Visão do usuário:** vê a qualidade em termos das metas específicas de um usuário final. Se um produto atende essas metas, ele apresenta qualidade;
- **Visão do fabricante:** define qualidade em termos de especificação original do produto. Se um produto atende as especificações, ele possui qualidade;
- **Visão do produto:** sugere que a qualidade está diretamente interligada a características interessantes, por exemplo, funções e recursos;
- **Visão baseada em valor:** mede o quanto um cliente estaria disposto a pagar por um produto.

Na visão de Pressman (2011), não é suficiente afirmar que qualidade é importante, características de qualidade precisam ser explicitamente definidas. Sendo assim, um conjunto de tarefas de desenvolvimento devem ser estabelecidas visando garantir que todo produto oriundo do trabalho de engenharia de *software* exiba um nível alto de qualidade. As atividades de controle e garantia de qualidade e

a adoção de métricas para quantificar e aperfeiçoar o processo de *software* são partes importantes desse processo de gestão (SOMMERVILLE, 2007).

## 4. CONCLUSÃO

Apesar dos processos, métodos, ferramentas e pesquisas sobre o desenvolvimento de *software* com qualidade, nos dias de hoje ainda existe *software* sendo desenvolvidos e outros já instalados que apresentam problemas de qualidade. A engenharia de *software* até o momento parece ser a cura de todo o mal, no entanto, observa-se que essa crise de *software* iniciou na década de 60 e até então não se obtiveram soluções definitivas para o problema.

Em razão desse cenário problemático associado ao desenvolvimento de software, que persiste por mais de quatro décadas, pode-se considerar a existência de uma configuração de calamidade crônica. A crise ocorreria caso esse cenário problemático se estabelecesse por um período de tempo limitado ou passageiro. Além disso, ainda não surgiram respostas definitivas para a solução de todos os problemas, de modo que essa problemática persistirá ainda por algum tempo.

As empresas ou equipes de desenvolvimento precisam adotar processos que a engenharia de *software* sugere como um padrão para a criação de um produto de qualidade. É importante destacar que esses processos precisam passar por modificações ao longo do tempo para atender as necessidades de cada empresa ou equipe. Provavelmente, esse seja um caminho para solução dos problemas ou, pelo menos, diminuição do índice de *software* de baixa qualidade.

## REFERÊNCIAS BIBLIOGRÁFICAS


BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: Guia do Usuário**. 12ª Ed., Rio de Janeiro: Campus, 2000.

BROOKS, F. ***No Silver Bullet: Essence and Accidents of Software Engineering***. IEEE Computer, Vol. 20 (4), p. 10–19, 1987.

CARVALHO, A.; CHIOSSI, T. **Introdução à Engenharia de Software**. São Paulo: Unicamp, 2001.

DEBONI, J. **Modelagem Orientada a Objetos com UML**. São Paulo: Futura, 2003.

FOWLER, M. **UML Essencial**. 3ª Ed., São Paulo: Bookman, 2005.

FURLAN, D. **Modelagem de Objetos Através da UML**. São Paulo: Makron Books, 1998.

GARVIN, D. ***What Does 'Product Quality' Really Mean?*** Sloan Management Review, Fall, p. 25–45, 1984.

MATOS, A. **UML: Prático e Descomplicado**. São Paulo: Érica, 2002.

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.

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

SOMMERVILLE, I. **Engenharia de Software**. 8ª Ed., São Paulo: Addison Wesley Brasil, 2007.