UNIVERSIDADE FEDERAL FLUMINENSE
HENRIQUE DOS SANTOS GONÇALVES

SOFTWARE: EVOLUÇÃO E LICENCIAMENTO

Niterói
2016

HENRIQUE DOS SANTOS GONÇALVES

SOFTWARE: EVOLUÇÃO E LICENCIAMENTO

Trabalho de Conclusão de Curso submetido ao Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Computação.

Orientador:
Eyder Franco Sousa Rios

NITERÓI
2016

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

G635 Gonçalves, Henrique dos Santos
Software : evolução e licenciamento / Henrique dos Santos Gonçalves. – Niterói, RJ : [s.n.], 2016.
42 f.

Projeto Final (Tecnólogo em Sistemas de Computação) – Universidade Federal Fluminense, 2016.
Orientador: Eyder Franco Sousa Rios.

1. Desenvolvimento de software. 2. Software livre. 3. Sistema de computador. I. Título.

CDD 005.1

HENRIQUE DOS SANTOS GONÇALVES

SOFTWARE: EVOLUÇÃO E LICENCIAMENTO

Trabalho de Conclusão de Curso submetido ao Curso de Tecnologia em Sistemas de Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Computação.

Niterói, 25 de Junho de 2016.

Banca Examinadora:

_____

Eyder Franco Sousa Rios, MSc - Orientador

Universidade Federal Fluminense - UFF

_____

Julliany Sales Brandão, DSc - Avaliadora

CEFET/RJ – Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

_____

Edcarllos Gonçalves Santos, MSc - Avaliador

Universidade Federal Fluminense - UFF

Dedico este trabalho a Deus e minha família.

AGRADECIMENTOS

A Deus, pois sem ele não seria possível chegar à conclusão desse curso.

Ao orientados Eyder Rios, pela atenção e contribuição para o desenvolvimento desse trabalho.

Aos tutores Rodolfo, Cristiano e Rafael Machado do Polo de Belford Roxo, pela dedicação, atenção e orientação nas tutorias prestadas durante o curso.

Aos amigos feitos durante todo esse tempo de formação.

A minha esposa, meus pais e irmã pela paciência, compreensão, colaboração e incentivos dados que ajudaram a chegar ao final desse curso.

“[...] Ebenézer, e disse: Até aqui nos ajudou o Senhor”.

(1Samuel 7:12) Bíblia Sagrada

RESUMO

O presente trabalho realiza um estudo sobre as origens e evolução do *software* e suas formas de licenciamento. Desta forma, é realizada uma análise sobre o papel do *software* dentro do contexto evolutivo do *hardware*, destacando-se como o *software* se originou e como se inseria na alvorada da computação. Através dessa análise é possível observar como o *software* foi evoluindo, se aperfeiçoando até culminar com sua independência ao se desvincular do *hardware*. Nesse processo de desenvolvimento são destacadas duas vertentes relacionadas ao seu uso e licenciamento: o *software* livre e o *software* proprietário. Um estudo sobre o surgimento dessas vertentes é realizado indicando como se consolidaram e suas principais características. Finalmente, é apresentado um quadro comparativo entre essas duas formas de licenciamento em termos dos fatores que influenciam na sua utilização.

Palavras-chaves: *software*, licenciamento de *software*, *software* livre, *software* proprietário, *open source.*

ABSTRACT

This paper conducts a study of the origins and evolution of software and ways of licensing. Thus, an analysis of the role of software within the evolutionary context of the hardware is performed, highlighting how the software originated and how they inserted at the dawn of computing. Through this analysis it is possible to observe how the software was evolving, being perfected to culminate in its independence to the break from the hardware. In this development process are highlighted two aspects related to its use and licensing: free software and proprietary software. A study on the emergence of these strands is performed indicating as consolidated and its main features. Finally, a comparative table between these licensing their ways in terms of the factors that influence their use is presented.

Keywords: software, licensing, free software, proprietary software, open source.

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AGPL - *Affero General Public License*

AT&T - *American Telephone and Telegraph*

EULA - *End User License Agreement*

FDL - *Free Documentation License*

GNU - *Gnu is Not Unix*

GPL - *General Public License*

IBM - *International Business Machines*

LGPL - *Library or Lesser General Public License*

MIT - *Massachusetts Institute of Technology*

MS-DOS - *Microsoft Disk Operating System*

MULTICS - *Multiplexed Information and Computing Service*

PC - *Personal Cumputer*

Q-DOS - *Quick and Dirty Operating System*

VLSI - *Very Large Scale Integration*

WGA - Windows Genuine Advantage

SUMÁRIO

1 INTRODUÇÃO

O *software* é constituído por um conjunto de instruções que gerenciam o *hardware*, controlam seu funcionamento e realizam tarefas específicas. Possui uma história de luta que possibilitou ao longo de sua existência, passar por diversas mudanças. Tais mudanças vão desde sua dependência ao *hardware*, o momento em que se torna independente, a origem de empresas que começaram a produzi-lo, exercendo domínio sobre ele, até o surgimento das licenças que instruem a melhor maneira de usufruí-los [7,9].

Hoje o *software* é um diferencial presente nos computadores. Há diferentes tipos com funções, características específicas e formas de utilização distintas. O usuário que adquire um *software* tem em suas licenças um manual, com orientações de como utilizá-lo corretamente. A decisão na escolha do *software* influenciará na maneira de sua utilização, sendo interessante conhecer as características de seus principais modelos: o livre e o proprietário. Dessa maneira o usuário poderá escolher o que satisfaça melhor às suas necessidades.

Esse trabalho de conclusão de curso apresenta a origem do *software*, sua relação com o *hardware* e sua forma de utilização. Será possível observar que a ação de programar, ou seja, fornecer instruções para uma determinada máquina é bem antiga, tendo origem mesmo antes da existência dos computadores atuais. Esse estudo traz também o processo evolutivo do *software* durante gerações, possibilitando entender seu modelo atual. Para finalizar o trabalho, serão exploradas duas variações de *software*, como se originaram e os principais conceitos de cada uma. A pretensão desse trabalho é propiciar informações sólidas, esclarecedoras e que possibilitam dar a opção de escolha de um *software* que atenda melhor as características de cada usuário.

O trabalho está dividido em seis capítulos. O primeiro é a introdução descrita nesse capítulo. A seguir, o Capítulo 2 apresenta um histórico do *software*, seu conceito e relação com o *hardware*. Especifica os avanços do *hardware* e do *softwa-*

re, acompanhando a evolução da computação eletrônica. O final do capítulo mostra o momento em que o *software* se desvincula do *hardware* e como foi esse processo de tornar-se independente.

O Capítulo 3 apresenta o surgimento do *software* proprietário trazendo uma descrição do seu modelo de licença e alguns conceitos necessários para seu entendimento. Em seguida apresenta como esse tipo de modelo de *software* se consolidou exercendo um domínio sobre quem o utilizava. Por fim é feito uma análise de uma licença proprietária destacando suas características.

O Capítulo 4 apresenta como se originou o *software* livre. Traz um entendimento sobre seus ideais, sua luta pela liberdade do conhecimento, sua metodologia e mostra suas principais características. O fim deste capítulo faz um comparativo entre os modelos de *software* livre e *software* proprietário e apresenta alguns fatores que influenciam na decisão do usuário sobre qual modelo utilizar.

Para finalizar o Capítulo 5 traz a conclusão desse trabalho e o Capítulo 6 suas referências bibliográficas.

2 HISTÓRICO DA EVOLUÇÃO DO *SOFTWARE*

Hoje se utiliza o termo computador para se referir a uma máquina específica e seus componentes. São máquinas que executam tarefas ou cálculos de acordo com um conjunto de instruções. Um computador pode ser programado para aceitar dados de entrada, transformá-los em informações de saída e armazená-los para proteção ou reutilização. Seu funcionamento se dá com base em uma interação de dois elementos principais: o *hardware* e o *software*. O *hardware* está relacionado com toda a parte física do computador, já o *software* basicamente é um conjunto de instruções que diz ao *hardware* o que fazer [11].

É importante entender que desde a máquina mais simples até a mais sofisticada não é importante sozinha, é necessário instruí-la em suas ações. Tais instruções hoje são escritas por desenvolvedores e organizadas de forma lógica, de modo que seus componentes eletrônicos possam executar suas tarefas e atingir os objetivos da aplicação. Existe uma importância mútua entre o *hardware* e o *software*, equipamento e programa. Essa relação é antiga e existe desde mesmo antes do surgimento dos modernos dispositivos eletrônicos [12].

No início de 1800, um tear criado pelo inventor francês chamado Joseph Jacquard pode ser considerado como inicio dessa relação. Tal criação possibilitou o aperfeiçoamento na área da tecelagem. Apesar de ser uma área diferente da computação contribuiu para o desenvolvimento de ideias que serviram de base para o projeto dos computadores modernos.

Jacquard era tecelão e percebeu que o trabalho humano na produção de telas era demorado, o que encarecia o produto final. Com o objetivo de substituir esse esforço produziu um tear dotado de um dispositivo que automatizava o processo de tecelagem de determinados padrões de desenhos. Para executar um traçado, a máquina fiandeira, que tecia o fio, deveria ter um plano ou programa que determinasse a ordem dos fios. O ponto chave desse tear era o uso de uma série de cartões

perfurados, cujos furos eram como instruções configuradas para descrever o modelo que seria produzido [3].

A máquina de Jacquard automatizou o processo manual de tecelagem de padrões agilizando o processo e reduzindo o custo, pois apesar das telas modeladas serem vendidas com altos preços, as longas horas de trabalho não compensavam os lucros. Pode-se considerar, levando a ótica do contexto moderno, que esses padrões perfurados nos cartões eram programas que controlavam o funcionamento da máquina, tornando-a provavelmente o primeiro dispositivo programável.

Um dos últimos e mais importantes trabalhos pioneiros em computação por processos mecânicos foi realizado por um inglês chamado Charles Babbage. Em seu trabalho, Babbage projetou dois tipos de máquinas: a Máquina Diferencial e a Máquina Analítica [4].

A Máquina Diferencial foi idealizada para atender às necessidades da Marinha real inglesa. Em 1823, Babbage se propôs em criar uma máquina que realizasse de forma constante e sem erros o tedioso trabalho de cálculos e que registrasse de forma confiável os resultados. A máquina era acionada por um motor a vapor que após uma série de etapas apresentava uma resposta. Além disso, continha um dispositivo de gravação em uma chapa de cobre, que serviria de matriz para posterior impressão em papel [4].

Babbage percebeu que a tecnologia de seu tempo estava muito aquém do que seu projeto exigia, o que tornava os custos elevados, inviabilizando e fazendo-o abandonar seu projeto. Contudo, Babbage continuou seus estudos e cogitou: se é possível construir uma máquina para executar um determinado tipo de cálculo, por que não seria possível construir outra capaz de qualquer tipo de cálculo? Essa foi a ideia que deu origem ao projeto da Máquina Analítica.

A Máquina Analítica funcionava com base nas instruções de cartões perfurados, assim como o tear de Jacquard. Poderia seguir conjuntos mutáveis de instruções e, portanto, executar diferentes funções que mais tarde seriam chamadas de programa. Babbage percebeu que para criar estas instruções precisaria de um tipo inteiramente novo de linguagem e a imaginou como números, setas e outros símbolos. Ela serviria para Babbage “programar” a Máquina Analítica, com uma longa série de instruções condicionais, que lhe permitiriam modificar suas ações em resposta a diferentes situações [3].

A entrada de dados para a máquina seria feita através de cartões. Mas duas inovações causaram grande impacto. A primeira foi o conceito de “transferência de controle” permitindo a máquina fazer comparações e tomar decisões de acordo com o resultado. A segunda era possibilitar que os resultados dos cálculos alterassem outros números e instruções colocadas na máquina, permitindo modificar seu próprio programa.

A máquina nunca chegou a se tornar uma realidade física, pois seu projeto era muito avançado para a tecnologia da época. Mesmo não estando construída, uma mulher chamada Ada Lovelace, por volta de 1842, desenvolveu uma sequência de instruções introduzindo conceitos que seriam largamente usados na programação de computadores atualmente. Pode-se considerar Ada como a primeira programadora da história da computação.

O próximo passo importante na História da Computação foi dado Herman Hollerith no final do século XIX. Em 1890, ele ganhou uma concorrência para o desenvolvimento de um equipamento de processamento de dados para auxiliar o censo nos Estados Unidos da América daquele ano, inspirado pelos teares de Jacquard, Hollerith desenvolveu o projeto de máquinas que pudesse interpretar, classificar e manipular as somas aritméticas representadas pelas perfurações dos cartões. Ele combinou cartões perfurados com os novos dispositivos eletromagnéticos da época [3].

Atualmente os cartões perfurados não são mais utilizados. Contudo, até meados da década de 1980, eles foram um dos principais elementos de entrada de dados dos computadores digitais [4].

A partir da década de 1930 cientistas começaram a trabalhar com dispositivos de cálculo com algum tipo de sistema de controle automático. Já se dispunha da tecnologia necessária para se construir aquela estrutura imaginada por Babbage. Surgiram os primeiros computadores mecânicos e eletromecânicos e muitos projetos de computadores eletrônicos feitos posteriormente sofreram grandes influências dessas primeiras máquinas [3].

2.1 O *HARDWARE* E *SOFTWARE*

A utilização do termo *hardware* se tornou muito comum na área da computação, porém sua origem não está ligada originalmente aos computadores. De origem inglesa o termo *hardware* está relacionado a equipamentos como: uma calculadora, um celular, um relógio, etc. Entretanto este termo ganhou um significado próprio com os avanços na área da computação, que se apropriou do termo para fazer referência aos componentes físicos do computador.

Já o termo *software* segundo o dicionário Michaellis é, além de outras definições já ditas, “qualquer programa ou grupo de programas que instrui o *hardware* sobre a maneira como ele deve executar uma tarefa, inclusive sistemas operacionais, processadores de textos e programas de aplicação” [5].

Segundo Gates (1995), *software* “é um conjunto abrangente de regras fornecido à máquina para instruí-la sobre como executar determinadas tarefas” [1].

De uma maneira geral o *software* pode ser classificado como de sistema ou de aplicativo. Os *software*s de sistema gerenciam o funcionamento da máquina e envolvem todos os programas relacionados com a coordenação operacional do computador. Nesse contexto inclui-se o Sistema Operacional, que é um conjunto de programas responsável por essa gerência e fundamental para controle de todos os recursos da máquina. Os *software*s aplicativos, que são personalizados ou oferecidos em pacotes, são utilizados para solucionar tarefas reais, resolver um problema em particular ou realizar uma tarefa específica [13].

O uso da palavra *software* para descrever programas de computador apareceu pela primeira vez no ano de 1958 em um artigo de um jornal universitário. Nesse artigo o professor John Wilder Tukey afirma, entre outras coisas, que o *software* são rotinas cuidadosamente planejadas, tão importante para uma calculadora da época como seu *hardware* [6].

Pode-se então fazer uma relação e afirmar que há mais de cem anos atrás já existia o conceito *hardware* e *software*. As instruções contidas nos cartões perfurados do Tear de Jacquard, no programa escrito por Ada para fazer a máquina de Babbage funcionar, nos cartões desenvolvidos por Hollerith para usar em sua

máquina, demostram que equipamento e programa, *hardware* e *software*, são elementos interdependentes.

Com a criação dos computadores eletrônicos, por volta de 1943, foi possível observar a evolução tanto do *hardware* quanto do *software*. Nesse contexto, é interessante entender como ocorreu esse processo e como o *hardware* e o *software* se tornaram o que são hoje. Para melhor entender esse progresso a evolução dos computadores é classificada em gerações, de acordo com o elemento básico utilizado na fabricação das máquinas e as características dos *software*s utilizados.

2.1.1 Primeira geração

A primeira geração de computadores ocorreu durante a década de 1940. Neste período o *hardware* exercia um papel principal, sofrendo contínuas mudanças, enquanto o *software* desempenhava um papel coadjuvante. A programação de computadores era uma arte “secundária” para a qual havia poucos métodos sistemáticos. Durante esse período, era usada uma orientação *batch* (em lote) para a maioria dos sistemas. Nesse processo em lote os dados eram organizados através de uma fila, aguardando seu processamento.

O produto de *software*, ou seja, programas que são desenvolvidos para serem vendidos a um ou mais clientes, estava ainda no seu início. Grande parte do *software* era desenvolvida e utilizada pela própria pessoa ou organização. Caso houvesse falhas o desenvolvedor era responsável pela manutenção [7].

> O ENIAC pesava trinta toneladas e enchia uma sala imensa. Dentro dele, os pulsos computacionais corriam entre 1500 relés eletromecânicos e fluíam através de 17mil válvulas. Para liga-lo, consumiam-se 150 mil watts de energia. E o ENIAC só tinha capacidade de armazenar o equivalente a cerca de oitenta caracteres de informação [1,p.42].

Conforme o relato é possível observar também que nesse período, os computadores eram extremamente volumosos, ocupando salas inteiras e utilizavam quilômetros de cabos e fios. O principal componente dos dispositivos da primeira geração eram as válvulas eletrônicas, que possuíam diversos elementos interligados

permitindo a passagem ou não de energia elétrica. Durante o funcionamento das válvulas consumia-se muita energia, deteriorando-as com muita frequência.

Nessa época predominava o *software* básico, sem muitos recursos e vinculado ao *hardware,* que por sua vez era criado para execução de um único programa. O *software* era distribuído em menor quantidade pelos fabricantes. Cada um possuía seus próprios meios de fabricação, dificultando a migração para outro *hardware* ou *software* devido aos altos custos [4,9].

Pode-se ainda observar essa dependência *software*/*hardware* no seguinte relato:

> Na época, o *software*, assim como o *hardware*, também era caro. Tinha de ser escrito especificamente para cada tipo de computador. E cada vez que o *hardware* do computador mudava, o que acontecia regularmente, o *software* precisava ser quase todo refeito. Os fabricantes de computadores forneciam alguns blocos, padrão de *software* (por exemplo bibliotecas de funções matemáticas) junto com a máquina, mas a maior parte do *software* tinha de ser escrita especificamente para resolver os problemas individuais desta ou daquela empresa. Havia alguns programas gratuitos e umas poucas companhias vendiam *software* de uso geral, porém havia muito poucos pacotes que se pudessem comprar no varejo[1,p.24].

2.1.2 Segunda geração

A segunda geração dos computadores ocorreu entre 1956 e 1963, impulsionada pela invenção do transistor. O transistor realizava as mesmas funções básicas da válvula, porém apresentava maior durabilidade, consumia menos energia e dissipava menos calor, transformando esse componente eletrônico num substituto natural para a válvula. Nasceram novos conceitos entre o homem e a máquina com o surgimento da multiprogramação e os sistemas multiusuários. Técnicas interativas abriram um novo mundo de aplicações e novos níveis de sofisticação de *software* e *hardware*. Com os sistemas de tempo real, dados eram coletados, analisados e transformados com mais rapidez [3,7].

O mercado de *hardware* se desenvolvia cada vez mais, tornando-se na época o grande interesse para a indústria da computação. O *software* não dava lucro imediato, visto que era fornecido juntamente com o *hardware*. O que importava mesmo eram os lucros com a venda de *hardware*.

O conceito de propriedade estava ligado ao *hardware* e havia, na época, um grande monopólio das empresas fabricantes. Uma empresa de *hardware* que possuía mais *software* gratuito disponível vendia mais, porém o *hardware* e *software* não eram portáteis nem interoperáveis em computadores de diferentes fabricantes [9].

2.1.3 Terceira e quarta geração

A terceira geração dos computadores ocorreu entre 1964 e 1970 e foi marcada pela tecnologia de circuitos integrados. O ponto importante no conceito de circuitos integrados é que se podem agregar múltiplos componentes eletrônicos em um único elemento, tornando-o bem menores, com consumo reduzido de energia e dissipando menos calor. O *hardware* passou a ser um produto primário enquanto o *software* oferecia recursos capazes de diferenciá-lo [7].

A quarta geração estende-se até os dias de hoje, é marcada pelas tecnologias VLSI, capazes de armazenar, em um pequeno espaço ou *chip,* grande quantidade de componentes eletrônicos. Surgiram os microcomputadores e computadores pessoais, favorecendo o crescimento de muitas empresas de *software* [3].

2.2 LIBERDADE DO *SOFTWARE*

Até o final da terceira geração de computadores, o *software* era feito para um *hardware* específico com arquitetura fechada ou proprietária. Não era utilizado em máquinas de outros fabricantes, pois usavam uma arquitetura diferente. Na época, a parte principal de um sistema de computação era o *hardware*, e a empresa fabricante tinha total controle, sobre o que podia ser executado nele. Possuía a autonomia de decidir se iria permitir ou não que seu *hardware* executasse *software* de terceiros ou se ela mesma iria desenvolvê-lo. Esse era um modelo estratégico de

propriedade do *hardware*: a empresa fabricante tinha a patente da arquitetura e dos componentes evitando que outros *software*s rodassem em sua plataforma [8].

Em 1969, as vendas de *hardware* eram dominadas pela grande empresa da época, a IBM. É possível observar esse domínio no seguinte relato:

> Na área de mainframes, a IBM era rainha absoluta e os concorrentes achavam difícil superá-la em vendas e fazer frente à alta capacidade de pesquisa e desenvolvimento da empresa. Se um concorrente
> ameaçasse usurpar a coroa, a IBM tinha condições de concentrar seus recursos para tornar a empreitada quase impossível[1,p.72].

As práticas competitivas da IBM no mercado lhe ocasionaram diversos processos. Empresas produziam equipamentos que poderiam ser conectados a instalações IBM. A IBM por sua vez introduzia protocolos e métodos secretos, com objetivo de tornar incompatível o que não fosse de sua produção [14].

Mediante a esse cenário o Departamento de Justiça dos EUA aplicou uma medida contra esse monopólio existente nas vendas, acusando a IBM de vincular o *software* ao *hardware* impedindo clientes de assinarem com concorrentes. Tal medida instituiu que o *software* fosse vendido desvinculado do equipamento, ou seja, sem estar ligado a um *hardware* específico. Nesse caso o *software* não seria mais fornecido junto ao *hardware*, e sim cobrado separadamente pela propriedade intelectual e direitos autorais de seus detentores [9,15].

O processo contra a IBM foi longo e a empresa teve altos gastos com sua defesa. O caso foi encerrado em 1982, a IBM teve que assumir uma série de compromissos, tendo atitudes mais cautelosas visando diminuir esse monopólio [17]. O *software* proprietário ainda não estava em questão, mas devido a essa medida a IBM passou a fornecer o *hardware* incorporando o mínimo de *software*.

A medida da justiça americana impulsionou na década de 70 o surgimento das empresas desenvolvedoras de *software*, que começaram a competir com empresas fabricantes de *hardware*, especializando-se somente em produzir e vender *software* independente da máquina [9].

O *software* era fabricado e fornecido para um mercado variado. Programas para *mainframes* e minicomputadores eram distribuídos para milhares de usuários [8].

As empresas desenvolvedoras de *software* contribuíram muito para o aprimoramento do *software* e sua independência, fazendo com que estes pudessem

ser executados em diferentes plataformas e modelos de computadores. Tendo sido convencidas, ou não, os fabricantes de *hardware* acharam mais conveniente e barato terceirizar a produção de *software*, uma vez que esse deixara de ser seu foco, fortalecendo ainda mais a indústria de *software*.

Esse contexto de liberdade de *software* proporcionou o surgimento de dois grandes grupos ligados ao desenvolvimento de *software*: o do *software* proprietário, baseado em licenças restritivas ao usuário, e o do *software* livre com uma visão colaborativa e de liberdade oposto ao conceito proprietário [9].

3 O *SOFTWARE* PROPRIETÁRIO

Com a consolidação das empresas desenvolvedoras de *software* no mercado houve uma extensão do monopólio dentro do conceito proprietário, que antes era prerrogativa do *hardware*, passou a ser dirigido ao *software* [9].

O *software* proprietário envolve um modelo de desenvolvimento e distribuição baseado em licenças restritivas, relacionadas com sua autoria e propriedade. As licenças funcionam como um contrato jurídico determinando que tipo de relação o usuário terá com o *software* adquirido [8,10].

As instruções contidas no *software* são chamadas de algoritmos, os quais são ocultados pelo modelo de *software* proprietário. Embora seja constituído por informações agrupadas e de se basear em conhecimentos acumulados pela humanidade, a indústria de *software* proprietário se direcionou para tentar bloquear e evitar que o caminho de seu desenvolvimento fosse semelhante ao modelo atual de desenvolvimento do conhecimento científico. Segundo Silveira (2004), a ciência se desenvolve a partir do princípio de compartilhamento e não a partir da ideia de propriedade, é essencialmente social, não se aplicando em seu conhecimento a ideia de apropriação privada.

Quando adquire um *software* proprietário o usuário está comprando uma licença de uso que definirá as melhores formas de utilização e manuseio do *software*[10].

3.1 CÓDIGO FONTE E CÓDIGO BINÁRIO

Um programador elabora o *software* através de um conjunto de instruções escritas em uma linguagem de programação (C, Java, Pascal, etc.) podendo ser lida

e interpretada por qualquer outra pessoa, esse conjunto de instruções é chamado de código fonte. O computador, por sua vez, não é capaz de entender esse tipo de linguagem. Internamente um computador só compreende uma sequência de zeros e uns chamado de código binário. Toda informação, seja ela qual for, precisa ser representada através desse código [16].

Existe um processo chamado compilação, que converte o programa escrito em código fonte para código binário, intermediado por um programa chamado compilador. Durante esse processo, o compilador recebe como entrada os arquivos de código fonte escritos pelo programador e o converte em um arquivo binário, ou programa executável. A compilação, portanto, traduz as instruções de alto nível da linguagem de programação em instruções de linguagem de máquina (ou de baixo nível), que pode ser interpretada pelo computador.

Grande parte dos *software*s proprietários é comercializada e distribuída nesse formato binário que dificilmente pode ser lido e compreendido por uma pessoa. Não tendo acesso ao código fonte, o usuário é restringindo na capacidade de entender e modificar esse *software* adquirido, já licenciado ou patenteado, podendo somente executá-lo no computador [9].

3.2 O MONOPÓLIO

Em 1975 um exemplar da revista “*Popular Eletronics*”, tornara ponto chave para o surgimento de uma grande empresa no ramo de *software* proprietário. Na capa da revista continha a fotografia de um computador, muito pequeno, chamado *Altair* 8800, contendo um microprocessador Intel 8080 como “cérebro” [1].

O *Altair* era programado através de um painel com uma série de interruptores, as respostas vinham por meio de *leds* acesos e apagados em seu painel frontal, não possuía nenhum tipo de interface com o usuário. Foi considerado, na época, como o primeiro computador pessoal comercializado, fazendo parte do início da revolução que os microcomputadores realizaram desde então [1,4].

Um dos grandes problemas do *Altair* era que ele não possuía um *software*. Dois jovens universitários, estudantes de programação enxergaram esse proble-

ma e começaram a desenvolver um *software* que rodasse naquele computador. Foi um processo cansativo, porém cinco semanas após a publicação da revista, *Bill Gates* e *Paul Allen* criaram o BASIC dando origem à *Microsoft*, empresa que posteriormente iria dominar o mercado de sistemas para computadores pessoais.

A *Microsoft* tinha como objetivo escrever e fornecer *software* para microcomputadores, sem se envolver diretamente com a fabricação ou venda de *hardware*. Oferecia *software* a preços baixos, pois acreditava-se que seria possível ter um retorno através das vendas.

Em 1980 a IBM, líder em vendas de *hardware* para computadores de grande porte, almejou entrar no ramo dos microcomputadores em menos de um ano. Para isso precisou abandonar seu modelo tradicional de fabricar o próprio *hardware* e *software* e *a*dquirir componentes já prontos, inclusive o microprocessador, comprado da Intel. Isso possibilitou cortar custos com o desenvolvimento e acelerar o processo, usando uma arquitetura aberta em que outras empresas pudessem copiar.

A IBM precisava de um sistema operacional para seu computador e contratou a *Microsoft* para desenvolvê-lo. Adquirindo um trabalho anterior de um sistema operacional, o QDOS, a *Microsoft* modificou esse sistema transformando-o no Sistema Operacional de Disco da *Microsoft* ou MS-DOS. Em 1981 a IBM lançara no mercado o IBM *Personal Computer*, popularizando o termo PC (*Personal Computer*) para os computadores pessoais. O fato de possuir uma arquitetura aberta permitiu empresas a desenvolver e vender componentes para seu PC, impulsionando seu crescimento no ramo, tornando sua plataforma padrão [1,34].

Essa parceria com a IBM consolidou a *Microsoft* como principal empresa fornecedora de *software* proprietário. A uma taxa bem pequena e única, a *Microsoft* dava o direito à IBM de usar o sistema operacional em tantos computadores que quisesse. Assim, a IBM poderia promover o MS-DOS e vende-lo a preços baixos [1]. No relato seguinte é possível observar o domínio que a *Microsoft* tinha sobre o MS-DOS, apesar de ter uma arquitetura aberta a IBM possuía um *software* totalmente proprietário:

> Nosso objetivo não era fazer dinheiro diretamente com as vendas da IBM, e sim licenciar o uso do MS-DOS a outros fabricantes de computador que quisessem oferecer máquinas mais ou menos compatíveis com o IBM-PC. A IBM podia usar nosso *software* de graça, mas não tinha direito exclusivo de uso nem controle sobre futuros aperfeiçoamentos [1,p.70].

O MS-DOS passou por vários aprimoramentos ao longo do tempo. Em 1983, a *Microsoft* anunciara um sistema operacional com uma interface gráfica bem superior à do MS-DOS, possuía menus, ícones, caixa de diálogos, possibilitando maior interação com o usuário. Dois anos após esse anúncio a *Microsoft* começa a comercializar seu novo sistema operacional chamado *Windows* [18].

O fato de usar o licenciamento para comercializar o sistema operacional, qualquer empresa poderia comprar as licenças para instalá-las e vender suas máquinas já com o *Windows* operando. Toda documentação do *Windows* era disponibilizada gratuitamente, o que possibilitou a *Microsoft* a convencer muitos desenvolvedores a criar *software*s para sua plataforma. Com tudo isso, ligado ainda a uma facilidade de uso, interface atraente, diversos recursos úteis para qualquer tipo de usuário, preços mais acessíveis no mercado, o *Windows* foi se popularizando e se tornou o sistema operacional mais utilizado no mundo [19].

3.3 UMA ANÁLISE DA LICENÇA PROPRIETÁRIA

A licença de um *software* é um contrato que define as condições de uso para um determinado programa. Um *software* proprietário utiliza essa licença para impor ao usuário a maneira de como ele poderá usufruir de seus recursos.

O *Windows* XP, por exemplo, é um sistema operacional da *Microsoft* lançado em 2001, sendo considerado como um dos mais populares da história. Ao analisar alguns pontos da licença dessa versão é possível entender melhor como funciona esse domínio do fabricante [20].

A licença de uso dessa versão segue um padrão da *Microsoft* chamado EULA (*End User License*) ou Contrato de Licença de Usuário Final. Quanto à instalação e utilização ela diz o seguinte:

> **Instalação e Utilização** Você deve instalar, usar, acessar, exibir e executar uma cópia do Produto em um único computador, tais como uma estação de trabalho, terminais ou outros dispositivos ("*Workstation Computer*"). O Produto não pode ser usado por mais de dois (2) processadores ao mesmo tempo em uma única Estação de Trabalho. Você deve permitir um máximo de dez (10) computadores ou dispositivos eletrônicos (cada Dispositivo) pa-

> ra conectar-se à Estação de Trabalho a fim de utilizar os serviços do Produto somente para serviços de Arquivo e Impressão*, Internet Information Services*, e acesso remoto (incluindo compartilhamento de conexão e serviços de telefonia).O máximo de dez conexões inclui quaisquer conexões indiretas feitas através de "*multiplexing*" ou outro *software* ou *hardware* que agregue as conexões. Exceto pelos recursos permitidos pelo *NetMeeting*, *Remote Assistance* e *Remote Desktop* descritos abaixo, você não deve usar o Produto para permitir que qualquer Dispositivo utilize, acesse, exiba ou execute outros *software* executáveis localizados na Estação de Trabalho, e nem deve permitir que qualquer Dispositivo utilize, acesse, exiba ou execute o Produto ou interface de usuário do Produto, a menos que o Dispositivos possua uma licença separada para o Produto [8].

É possível observar que a *Microsoft* impôs um conjunto de exigências para a utilização desse *software*. Apesar de pagar por essa versão o usuário é limitado a utilizá-lo apenas em uma máquina, ainda assim, com uma série de restrições que visam impedir a utilização e compartilhamento dos recursos em grandes proporções [8].

Quando foi lançado, o *Windows* XP veio com uma ferramenta chamada WGA (*Windows Genuine Advantage* ou Vantagens do *Windows* Original), que evita o bom funcionamento do sistema operacional caso perceba que é uma cópia não original. Apesar de não impedir a distribuição ilegal do *software*, caso fosse adquirida uma cópia não original o usuário tinha alguns problemas na utilização. A ferramenta bloqueia diversas funções do sistema, impedindo a atualização periódica e melhorias na segurança [21,22].

Mesmo após a aquisição do sistema operacional a licença cria obrigações ao usuário exigindo a comunicação com a empresa informando-a da instalação do *software*. Com isso poderia bloquear o *software* que não tivesse a licença de uso.

> Ativação Obrigatória Os direitos de licença garantidos pelo EULA são limitados aos primeiros trinta (30) dias após a primeira instalação do Produto, a menos que você forneça as informações exigidas para ativar a sua cópia licenciada da maneira descrita durante a sequência de configuração do Produto. Você pode ativar o Produto através da utilização da Internet ou do telefone; podem ser aplicadas taxas. Você também pode precisar reativar o Produto caso modifique o *hardware* de seu computador ou altere o Produto. Há medidas tecnológicas neste Produto que têm como objetivo prevenir a

> utilização ilegal ou não licenciada do Produto. Você concorda com o fato de termos de utilizar essas medidas [8].

Na aquisição do sistema operacional o usuário recebe uma chave de segurança única, necessária para ativação. Caso a licença não for ativada no prazo de trinta dias, algumas funções do *software* são desativadas e mensagens solicitando a ativação do *Windows* para seu completo funcionamento são exibidas periodicamente. Geralmente quando se adquire um computador o sistema operacional já está instalado e configurado, dispensando qualquer ação do usuário, a não ser que o mesmo tenha problemas com a máquina.

Apesar dessas exigências é possível instalar uma cópia do *Windows* em mais de um computador, utilizando a mesma chave de segurança, porém não é possível ativar duas vezes a mesma chave. Nesse caso os termos do EULA estariam sendo violados, caracterizando pirataria, que é a instalação e utilização de um *software* sem a autorização do fabricante [21].

A engenharia reversa de um *software* trata-se de um estudo feito para analisar suas características, o que ele faz, como se comporta, possibilitando criar uma representação do programa. A licença do *Windows* deixa clara a proibição dessa prática bloqueando o acesso ao código fonte do *software*. Restringindo esse código o usuário possui somente o *software* executável, toda linha de código escrita pelos programadores é mantida em segredo. Uma das principais características do *software* proprietário é perceptível nesse ponto, que é a não transparência do código-fonte, evitando o compartilhamento de conhecimento. O que o cliente estará adquirindo com a compra é somente a licença da utilização do programa, o mesmo continuará sendo da empresa que o desenvolveu.

> LIMITAÇÃO NA ENGENHARIA INVERSA, DESCOMPILAÇÃO OU SEPARAÇÃO Você não deve inverter a engenharia, descompilar ou separar o Produto, exceto e somente ao volume que é, não obstante, expressamente permitido pela lei aplicável a esta limitação [8].

Essa característica evita apenas que outras pessoas ou empresas acessem as linhas de código do *software*, não impede nem bloqueia uma cópia não autorizada. Na verdade, a prática da cópia não autorizada ajudou a popularizar o sistema operacional da *Microsoft*. Com a arquitetura aberta da IBM muitas empresas fabricavam computadores com os mesmos padrões e copiavam ilegalmente o *software* da

Microsoft. Essa junção entre arquitetura aberta do *hardware* e a cópia não autorizada do *software*, contribuiu com a utilização do sistema operacional *Windows* [8].

Foi possível perceber que a licença proprietária impõe condições ao usuário que vão além de apenas usufruir do programa. Utilizar um *software* com esse tipo de licença significa concordar com seus termos e aceitar suas exigências, em alguns casos são tantas que um usuário leigo pode estar violando sem mesmo saber.

É importante deixar claro que embora exista todo esse controle da parte do fabricante, restringindo o uso e protegendo seu *software*, um *software* proprietário, na maioria das vezes, é desenvolvido por grandes empresas. Tais empresas são objetivas em produzir um *software* que supra as necessidades do mercado e atenda os interesses do usuário. Elas oferecem um desenvolvimento centralizado, definindo padrões de instalação e configuração. Além disso, oferecem suporte técnico e dão garantias do bom funcionamento do *software*.

3.4 CARACTERÍSTICAS DO *SOFTWARE* PROPRIETÁRIO

O usuário que adquire um *software* proprietário tem sua utilização controlada por acordos de licenciamentos. Tais licenças são documentos legais que descrevem como o *software* deverá ser utilizado. Entre os *software*s proprietários a variedade de licenças é bem maior, como são de propriedade da empresa, eles possuem um departamento responsável por estipular as variações de licenciamento para cada produto [8].

O *software* proprietário oferece um suporte dedicado, garantindo a segurança, detectando e tratando possíveis erros. Possui fácil instalação, configuração e utilização, sua produção é cuidadosamente planejada visando às necessidades do usuário final. Melhorias, alterações e modificações não podem ser feitas pelo usuário. Na maioria das vezes é necessário pagar à empresa responsável pelo *software* para realizar esse serviço. Os custos com licenças de *software*s proprietários são bem superiores se comparados com as do *software* livre. O acesso ao código fonte é restrito ao usuário, não podendo realizar cópias e distribuí-las para terceiros [26].

4 ORIGEM DO *SOFTWARE* LIVRE

Segundo a *Free Software Foundation* (Fundação do *Software* Livre) é considerado livre qualquer *software* que os usuários possuam a liberdade de executar, copiar, distribuir, estudar, modificar e aprimorar [25]. A metodologia cooperativista dos *softwares* livres de hoje começou de uma forma inconsciente, há muito tempo, com os chamados *softwares* cooperativos ou colaborativos no final dos anos 1950 [9].

> Devido à grande incompatibilidade e falta de conectividade dos computadores e seus *software*s, e ainda a não-disponibilidade de interfaces universalmente aceitas e padronizadas, os usuários gabaritados, principalmente os talentosos pesquisadores da universidades, de posse dos códigos fonte (que na época não era difícil de serem obtidos, e os poucos eram produzidos pelos fornecedores dos computadores), com eu ia dizendo, os pesquisadores e/ou professores por livre iniciativa e necessidade própria, desenvolviam novos *software*s sobre os disponíveis. Criavam, modificavam ou adequavam o anterior, para minimizar as dificuldades e atender suas necessidades, criando novos aplicativos e/ou soluções [9,p.23]

Nesse relato observa-se que ainda não havia a preocupação com direitos de propriedades, licenças e restrições na área do *software*, pois o que prevalecia era o *hardware*. Quem se beneficiava desse princípio de desenvolvimento eram os fornecedores de computadores. Melhoravam e consolidavam seu *software*, tornando-o mais acessível, contribuindo com a venda do *hardware*, grande interesse da época [9].

Dentro desse contexto, na década de 60, um grupo de desenvolvedores se juntou para desenvolver um projeto de um sistema operacional que fosse capaz de comportar centenas de usuários. Esse sistema operacional foi elaborado nos laboratórios Bell da AT&T (*American Telephone and Telegraph*) sendo chamado de Multics e posteriormente abandonado [23].

Ken Thompson, um dos desenvolvedores do Multics, viu a necessidade de continuar com o projeto, criando um sistema operacional com os mesmos propósitos, porém mais simples, originando o Unics posteriormente chamado de Unix.

Uma linguagem de alto nível é uma linguagem de programação que possui estruturas mais próximas da linguagem humana. A utilização de uma linguagem de alto nível especificamente voltada para a implementação do Unix foi um dos principais fatores para o sucesso desse sistema. O Unix possuía características inovadoras no contexto dos sistemas operacionais: simplicidade, escrita quase toda em linguagem de alto nível, permissão de reutilização do código [23].

Controlado pela AT&T, que detinha na época o monopólio na área de telecomunicações nos EUA, o laboratório Bell não tinha permissão comercial sobre o Unix. O processo de desenvolvimento inicial do Unix precisava de melhorias, extensões, portabilidade, interoperatividade e eliminação de erros. Para isso necessitava da colaboração de pessoas criativas e competentes. Buscando aperfeiçoar o sistema, a AT&T disponibilizou seu código-fonte para algumas universidades e organizações poderem usar, operar, modificar e produzir novas extensões do sistema [9].

Nesse contexto de colaboração tecnológica de melhorias e avanços do sistema, em 1984 a AT&T decidiu fechar o código-fonte do Unix, restringindo-o apenas para uso e não mais para pesquisas. O movimento do *software* livre emergiu quando um pesquisador do laboratório de inteligência artificial do MIT (Instituto de Tecnologia de Massachusetts) nos EUA, Richard Stallman, reage à decisão da AT&T de fechar o código fonte do Unix. Stallman inicia um longo processo de desenvolvimento de um novo sistema operacional, o GNU, e posteriormente fundara a *Free Software Foundation* [8].

4.1 O PROJETO GNU

Em 1971, o pesquisador Richard Stallman iniciou sua carreira no MIT. Fazia parte de uma comunidade de compartilhamento de *software*. O termo "*software* livre" ainda não existia, mas Stallman reconhece que o *software* que utilizavam podia ser chamado assim. Segundo ele, "sempre que as pessoas de outra universidade ou empresa queriam portar e usar um programa, tínhamos o prazer de deixá-los". Já na década de 1980, quase todo *software* era proprietário, evitando a cooperação entre os usuários. Stallman era contra o uso sistema proprietário para

ele “o sistema que diz que você não tem permissão para compartilhar ou modificar o *software* – é antissocial, antiético ou simplesmente errado...”.

Após o fim da comunidade da qual fazia parte, Stallman se viu em um momento de reflexão: se juntar ao mundo do *software* proprietário, assinando acordos e prometendo não ajudar outros companheiros ou deixar o ramo da computação, não utilizando suas habilidades na prática proprietária? Stallman, então chegou a seguinte questão: existe um programa que ele pudesse escrever para formar novamente uma comunidade com os princípios da qual fazia parte? A resposta pode ser observada no relato seguinte:

> A resposta foi clara: em primeiro lugar, um sistema operacional era necessário. Esse é o *software* crucial para começar a usar um computador. Com um sistema operacional, você pode fazer muitas coisas; sem ele, o computador mal funciona. Com um sistema operacional livre, poderíamos voltar a ter uma comunidade de *hackers q*ue cooperam — e convidar qualquer pessoa para participar — e todos seriam capazes de usar um computador sem ter que conspirar para privar seus amigos [28].

Em 1984 Stallman deu início ao projeto GNU, com a proposta de construir um sistema capaz de rodar programas e aplicativos Unix, mas que fosse livre e independente de licenças proprietárias de uso. Sua decisão marcou a história da tecnologia da informação.

> Como um desenvolvedor de sistema operacionais, eu tinha as habilidades certas para este trabalho. Assim, ainda que eu não pudesse tomar o sucesso como certo, eu percebi que havia sido eleito para realizar este trabalho. Eu decidi fazer o sistema compatível com o Unix para que ele fosse portável, e para que os usuários do Unix pudessem migrar para ele facilmente. O nome GNU foi escolhido, seguindo uma tradição hacker, como um acrônimo recursivo para “GNU Não é Unix” (do inglês “GNU’s Not Unix”) [28].

Stallman escreveu um texto conhecido como manifesto GNU, nele explicava os motivos da criação do sistema e fazia uma convocação para quem quisesse apoiar ou participar do projeto. Entre outras coisas o texto deixa claro que o GNU não é uma cópia do Unix e sim um sistema compatível que será fornecido gratuitamente para todos que quisessem utilizá-lo. Stallman acreditava nos princípios de colaboração e contribuição era contra a comercialização e criação de licenças que limitavam o usuário na utilização do *software.*

> Eu acredito que a regra de ouro exige que, se eu gosto de um programa, eu devo compartilhá-lo com outras pessoas que gostam dele. Vendedores de *Software* querem dividir os usuários e conquistá-los, fazendo com que cada usuário concorde em não compartilhar com os outros. Eu me recuso a quebrar a solidariedade com os outros usuários deste modo. Eu não posso, com a consciência limpa, assinar um termo de compromisso de não-divulgação de informações ou um contrato de licença de *software*. Por anos eu trabalhei no Laboratório de Inteligência Artificial do MIT para resistir a estas tendências e outras inanimosidades, mas eventualmente elas foram longe demais: eu não podia permanecer em uma instituição onde tais coisas eram feitas a mim contra a minha vontade [29].

Qualquer usuário teria a permissão para modificar o sistema, adequando-o para sua melhor utilização, poderá redistribuir, mas sem restringir caso elaborasse uma nova versão. Nenhuma modificação proprietária seria permitida no sistema. Com o GNU finalizado, qualquer pessoa poderia ter um sistema operacional gratuito com código fonte totalmente disponível, sem precisar pagar por uma licença [29].

Em 1985 o GNU já era utilizável. Com o crescimento e avanço do projeto Stallman fundou a FSF (*Free Software Foundation* – Fundação do *Software* Livre), instituição com o objetivo de buscar mais recursos financeiros e aumentar o desenvolvimento do *software* livre, assumindo sua distribuição. A FSF utiliza diferentes licenças escritas com o objetivo de promover e preservar a liberdade do *software*. São elas: GNU GPL (Licença Pública Geral), LGPL (Licença Pública Geral Menor), AGPL (Licença Pública Geral Affero), FDL (Licença Documentação Livre) [28].

4.2 Características do *software* livre

O termo “*software* livre” não se refere à questão de preço e sim liberdade, um programa é considerado livre atendendo a quatro liberdades específicas na sua utilização:

- Liberdade de executar um programa como o usuário desejar, para qualquer propósito sem necessidade de comunicar ao desenvolvedor.

- Liberdade de estudar como o programa funciona e adaptá-lo às necessidades, tendo possibilidade de usar uma versão modificada no lugar da original.
- Liberdade de redistribuir cópias de modo que possibilite ajudar o próximo.
- Liberdade de distribuir de versões modificadas a outros.

Qualquer *software* que não atende a todas essas liberdades não é considerado um *software* livre. Um *software* livre não está restrito à venda, um usuário pode ou não pagar para obtê-lo, mas uma vez adquirido as quatro liberdades estarão associadas ao programa [30].

A utilização de um *software* livre oferece ao usuário ou organização a flexibilidade de poder modificar o código-fonte e adaptá-lo para que possa atender suas necessidades. Por ter seu código disponível, permite a detecção de falhas e vulnerabilidades, tornando o *software* mais seguro. Consequentemente a correção das falhas é feita em um ritmo superior à do *software* proprietário.

O custo do licenciamento de um *software* livre, se comparado ao *software* proprietário, é bem mais acessível, sendo obtido gratuitamente ou a preços mais baixos. A empresa que utiliza um *software* livre reduz seus gastos na aquisição de licenças, deixando de estar sujeita a alterações e problemas na forma de licenciamento. Poderá instalar, utilizar, modificar, quantas vezes for necessária sem se preocupar em infringir as leis como as impostas por uma licença proprietária. O *software* livre permite a interoperatividade entre sistemas, possibilitando trabalhar com sistemas operacionais diferentes mesmo com as diferenças existentes entre eles.

Após o conhecimento das características das licenças de *software* a tabela 1 retrata um quadro comparativo na utilização de um *software* livre e um *software* proprietário.

Tabela 1: Comparativo *software* livre e *software* proprietário [24].

SOFTWARE PROPRIETÁRIO	*SOFTWARE* LIVRE
Elevados custos na ampliação e modernização geram preços elevados ao usuário final.	Proporciona grande vantagem econômica, pois a aquisição é gratuita ou possuem valores bem mais em conta.
O usuário não paga pelo *software* e sim	Sua filosofia defende que a liberdade e o

por uma licença de utilização. Os direitos comerciais e autorais são preservados e o usuário não tem acesso ao código fonte.	conhecimento não são direitos individuais e sim coletivos. Sua principal licença garante que os trabalhos desenvolvidos não são propriedade de ninguém, o usuário tem livre acesso ao código fonte.
As licenças restringem os direitos do usuário e protegem o fabricante de *software*.	O usuário tem liberdade de executar, copiar, distribuir, estudar e modificar o *software*.
Há uma dependência com a empresa desenvolvedora, pois somente ela é responsável pela manutenção, suporte e atualizações. Na maioria das vezes o usuário paga por esses serviços.	O processo de cooperação e compartilhamento possibilita um rápido processo de descoberta e correção de erros. Há uma variedade de suporte gratuito, disponíveis em sites e fóruns na internet. Usuários contam com comunidades desenvolvedoras que oferecem suporte online.
Seu desenvolvimento é feito por empresas que investem em equipes, tempo e pesquisas.	O *software* está em constante construção coletiva.
Para atender às necessidades do mercado e obter maiores lucros são lançados à curto prazo. Gerando, em alguns casos, instabilidades e erros de programação.	Quando é lançado no mercado, outros desenvolvedores têm acesso ao código, iniciando um processo de verificação e eliminação de erros.
Caso haja necessidade de alterar ou adaptar um *software*, o suporte é limitado a solicitar essa alteração ao fabricante.	Um erro descoberto é rapidamente corrigido pela comunidade de desenvolvimento, só encerrando o trabalho após concluir a correção.
Seu desenvolvimento é centrado no usuário final, logo há uma maior facilidade de instalação, configuração e	Sua documentação é pouco esclarecedora, dificultando a utilização de usuários iniciantes. A instalação e

utilização.	configuração exigem um conhecimento técnico maior, exigindo mais experiência do usuário.
É menos flexível ao usuário.	O usuário pode adaptar o *software* conforme suas necessidades.
Possuem interfaces padronizadas.	Há uma variedade de interfaces, não possuindo um padrão específico.
É vulnerável à invasão de vírus.	É menos vulnerável à invasão de vírus.
Orienta-se em benefício do fabricante.	Ajusta-se em benefício da sociedade.
O *software* está ligado ao computador no qual foi instalado. Caso o usuário mude de máquina sua licença não será mais útil.	O usuário é livre para instalar o *software* quantas vezes quiser, em quantas máquinas for necessário.

4.3 DECISÃO NA ESCOLHA DO *SOFTWARE*

Usar um *software* livre ou proprietário é uma decisão de cada pessoa ou empresa. Alguns fatores como, por exemplo, custo, usabilidade, flexibilidade, segurança confiabilidade, devem ser considerados e influenciarão na escolha do *software* ideal.

No ramo dos computadores pessoais dois sistemas operacionais, um de licença proprietária e outro livre, estão entre os dez mais utilizados em todo mundo. O *Windows*, sistema proprietário da *Microsoft,* é líder absoluto estando presente em quase 90% dos computadores. Já o Linux, sistema operacional livre alternativo ao proprietário da *Microsoft,* é encontrado em aproximadamente 2% [31].

Um dos pontos positivos do Linux é que o sistema é gratuito, enquanto que a licença do *Windows* é paga e só pode ser instalada em um computador. Em relação à segurança, existe uma oferta menor de vírus para o Linux, mas pode-se apresentar tão vulnerável quanto o *Windows*. Tudo depende da atualização e da maneira que o sistema será utilizado. Ao contrário de alguns sistemas o *Windows* possui uma

compatibilidade com diversos *software*s, podendo instalá-los e configurá-los sem maiores problemas. O desempenho do Linux permanece ótimo mesmo quando executado em *hardware*s mais modestos ou computadores mais antigos, possibilitando o usuário ter um sistema atual utilizando poucos recursos. A disponibilidade de games para o sistema Linux é bem menor se comparado ao *Windows* [32].

Não é possível classificar um sistema como melhor ou pior, o melhor sempre será aquele que atender com satisfação as necessidades do usuário.

Na área dos servidores web, *software*s que permitem o compartilhamento de informações pela internet, o *software* livre Apache é um dos mais utilizados seguido pelo IIS da *Microsoft* [33]. Possui grande estabilidade, segurança e flexibilidade, se adaptando a diversas páginas *web*.

O uso de um *software* adequado por uma empresa acarreta em ganhos de produtividade e vantagens no mercado. A escolha desses *software*s deverá ser alinhada e ir de encontro com os objetivos estratégicos da empresa. Alguns critérios podem ser considerados relevantes na hora dessa decisão [27].

- Código aberto: a empresa terá acesso ao código fonte do *software*? O acesso ao código é realmente importante para a empresa? Adotar um *software* com código fechado colocará os negócios da empresa em risco?
- Licenciamento: em que nível a licença restringe os direitos da empresa?
- Confiabilidade: no caso de um *software* livre, existe financiamento para o projeto? Qual o ganho em qualidade e confiança? Caso o *software* seja proprietário, a empresa fabricante possui uma credibilidade no mercado?
- Flexibilidade: é possível criar novas funcionalidades para aprimorar os serviços da empresa?
- Escalabilidade: o *software* acompanha o crescimento da empresa? A escalabilidade é praticável do ponto de vista do custo?
- Tecnologia: o *software* é compatível com os sistemas mais usados atualmente? A equipe possui conhecimento necessário para sua utilização?
- Riscos: a tecnologia adotada é compatível com o mercado? A empresa irá se tornar dependente do *software*? Essa dependência afetará de alguma maneira a empresa?

- Penetração no mercado: quantas empresas utilizam esse *software*? São do mesmo segmento?
- Custo: quais serão os custos iniciais de aquisição, instalação e treinamento? Como serão os gastos rotineiros com suporte e manutenção?
- Suporte: o *software* oferece uma documentação clara para implantação, uso e solução de problemas? Há equipes especializadas para implantação e solução de problemas?
- Manutenção: o *software* possui um bom histórico de atualizações? Há uma manutenção preventiva?
- Confiabilidade técnica: o *software* é estável?
- Segurança: qual o histórico de falhas? Qual tempo médio entre a descoberta e reparação de uma falha? O *software* possui diversos níveis de acesso?
- Performance: o consumo de processamento, memória, espaço em disco é adequado? A linguagem e arquitetura utilizadas são boas? O *software* depende de outros sistemas?
- Funcionalidade: o *software* faz o que é preciso? Atende as necessidades?
- Usabilidade: possui uma interface amigável? É de fácil utilização?

Segundo Moraes e Santaliestra (2008), a implantação de um *software* traz benefícios tangíveis e intangíveis. Os tangíveis que estão relacionados ao custo, suporte, manutenção, etc., acabam sendo mais considerados do que os intangíveis, que envolvem usabilidade, flexibilidade, confiabilidade, etc. Considerar esses dois aspectos em conjunto permite dar uma visão mais realista dos problemas associados ao processo de escolha [27].

5 CONCLUSÕES E TRABALHOS FUTUROS

Pela observação dos aspectos analisados nesse trabalho foi possível conhecer um pouco da história do *software*. Foi visto que o *software* passou por um processo longo de mudanças, lutas, transformações até chegar ao seu estágio atual. Foi abordado o surgimento do *software* proprietário as suas propriedades e o que ele oferece ao usuário. Também foi detalhada a origem do *software* livre, seus ideais e benefícios para quem o utiliza.

O estudo realizado nesse trabalho mostrou que existem vantagens e desvantagens na utilização tanto do *software* livre como do *software* proprietário. Foram descritas as características de cada um desses modelos, conhecendo detalhes de suas histórias e seus conceitos. Através desse estudo percebeu-se que a indústria do *software* proprietário tenta exercer um controle sobre quem utiliza seus programas e o movimento do *software* livre surge com o intuito de diminuir esse domínio, proporcionando liberdade ao usuário.

Esse trabalho possibilitou ao leitor ter um conhecimento maior sobre as licenças de *software*. Pode-se concluir que não há como caracterizar um *software* como melhor ou pior, mas é necessário compreender que o melhor sempre será aquele que atender com amplitude as necessidades de quem o utiliza.

Como trabalhos futuros, pretende-se realizar pesquisas em empresas, organizações e usuários domésticos, identificando suas particularidades e necessidades, para assim, indicá-los e orientá-los na melhor maneira de utilização de um *software*. Pretende-se também, realizar estudos mais aprofundados em temas relacionados a esse assunto somando com o conhecimento adquirido até o momento.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] GATES, Bill. **A estrada do futuro**, São Paulo: Cia das Letras, 1995.

[2] ALCADE, E.; GARCIA, M. & PEÑUELAS, S. **Informática Básica**, São Paulo: McGraw-Hill, 1991.

[3] FONSECA FILHO, Cléuzio. **História da Computação: o caminho do pensamento e da tecnologia**, Porto Alegre: EDPUCRS, 2007.

[4] MONTEIRO, Mário A. **Introdução à Organização de Computadores**, 4ª Edição, Rio de Janeiro: LTC, 2002.

[5] MICHAELIS. **Moderno Dicionário da Língua Portuguesa**. Disponível em: <http://michaelis.uol.com.br/moderno/portugues/index.php> Acesso em: 29 Mar. 2016.

[6] HISTORYOFINFORMATION**. The Origins of The Term "*Software*" Within the Context of Computing**. Disponível em: <http://www.historyofinformation.com/expanded.php?id=936> Acesso em: 24 Mai. 2016

[7] PRESSMAN, Roger S. **Engenharia de *software***, São Paulo: Makron Books, *1995.*

[8] AGUIAR, Vicente M. (Org.). ***Software* livre, cultura hacker e ecossistema da colaboração**, São Paulo : Momento Editorial, 2009.

[9] PACITTI, Tércio. **Paradigmas do *Software* Aberto**, Rio de Janeiro: LTC, 2006.

[10] SILVEIRA, Sérgio Amadeu da. ***Software* livre: a luta pela liberdade do conhecimento**, São Paulo: Fundação Perseu Abramo, 2004.

[11] *MICROSOFT*. **Introdução aos computadores.** Disponível em: < http://*Windows*.*Microsoft*.com/pt-br/*Windows*/introduction-to-computers#1TC =*Windows*-7> Acesso em: 02 Mai. 2016.

[12] DOUGLAS JOSÉ PEIXOTO DE AZEVEDO. **Evolução de *Software*.** Disponível em:< http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.

php?conteudo=299> Acesso em: 02 Mai. 2016.

[13] CAPRON,H.L.; JOHNSON, J.A. **Introdução à informática,** 8ª Edição, São Paulo: Pearson Prentice Hall, 2004.

[14] TIGRE, Paulo Bastos**. Indústria brasileira de computadores; perspectivas até os anos 90**, Rio de Janeiro: Campus, 1987.

[15] FOLHA ON LINE. **Lei Sherman vigora nos EUA desde 1890; conheça alguns processos.** Disponível em: <http://www1.folha.uol.com.br/folha/reuters/ult112u23874.shtml> Acesso em: 06 Mai. 2016.

[16] MARCIO FRAYZE DAVID. **Iniciando no mundo da programação.** Disponível em: <http://www.*hardware*.com.br/dicas/iniciando-mundo-programação.html> Acesso em; 07 Mai. 2016.

[17] P.PIROPO. **A era pc: decisões cruciais.** Disponível em: <http://www.techtudo.com.br/platb/*hardware*/2011/12/08/a-era-pc-decisoes-cruciais/> Acesso em: 06 Mai. 2016.

[18] *MICROSOFT*. **Uma história do *Windows*.** Disponível em:< http://*Windows*.*Microsoft*.com/pt-br/*Windows*/history#T1=era1> Acesso em: 31 Mai. 2016.

[19] RODRIGO GHEDIN. **Com 27 anos, o *Windows* precisa se reinventar para continuar líder.** Disponível em: <http://www.techtudo.com.br/artigos/noticia/2012/11/com-27-anos-o-*Windows*-precisa-se-reinventar-para-continuar-lider.html/> Acesso em: 01 Jun. 2016.

[20] FERNANDO DAQUINO. **O fim de uma era: a trajetória do *Windows* XP [infográfico].** Disponível em:< http://www.tecmundo.com.br/*Windows*-xp/53186-o-fim-de-uma-era-a-trajetoria-do-*Windows*-xp-infografico-.htm> Acesso em: 02 Jun. 2016.

[21] OLIVER HAUTSCH. **O que acontece se o *Windows* for instalado e ativado com a mesma licença em mais de um PC?** Disponível em: <http://www.tecmundo.com.br/*Windows*-7/3101-o-que-acontece-se-o-*Windows*-for-instalado-e-ativado-com-a-mesma-licenca-em-mais-de-um-pc-.htm> Acesso em: 02 Jun. 2016.

[22] ALBERTO LEAL. **Entendendo o WGA (*Windows Genuine Authentication*).** Disponível em: <https://technet.*Microsoft*.com/pt-br/library/cc716511.aspx> Acesso em: 02 Jun. 2016.

[23] TANENBAUM A.S. **Sistemas operacionais modernos**, 3ª Edição, São Paulo: Pearson, 2009.

[24] GARCIA, Mauro neves; SANTOS, Silvana Mara Braga; PEREIRA, Raquel da Silva; ROSSI, George Bedineli. ***Software* Livre em Relação *Software* Proprietário: Aspectos Favoráveis e Desfavoráveis Percebidos por Especialistas.** Gestão & Regionalidade (online), v. 26, p. 106-120, 2010.

[25] GNU. **O que é *software* livre? A definição de *software* livre**. Disponível em: <https://www.gnu.org/philosophy/free-sw.html> Acesso em: 14 Jun. 2016.

[26] RICARDO ANDRADE. **Modelo open-source ou proprietário?** Disponível em: <http://pplware.sapo.pt/informacao/opiniao/modelo-open-source-ou-proprietrio/> Acesso em 13 Jun. 2016.

[27] MORAES, E. A.; SANTALIESTRA, R. **Modelo de decisão com múltiplos critérios para escolha de *software* de código aberto e *software* de código fechado**. Revista Organizações em Contexto, v. 4, n. 7, p. 59-83, 2008.

[28] GNU. **O projeto GNU**. Disponível em: < https://www.gnu.org/gnu/Thegnuproject.html > Acesso em: 14 Jun. 2016.

[29] GNU. **O Manifesto GNU**. Disponível em: <https://www.gnu.org/gnu/manifesto.html> Acesso em: 14 Jun. 2016.

[30] GNU. **O que é *software* livre?** Disponível em: <https://www.gnu.org/philosophy/free-sw.html> Acesso em: 14 Jun. 2016.

[31] NETMARKETSHARE. **Desktop Operating System Market Share.** Disponível em: <https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0> Acesso em: 14 Jun. 2016.

[32] RONALDO PRASS. **O que mudou na velha batalha Linux x *Windows*?** Disponível em: < http://g1.globo.com/tecnologia/blog/tira-duvidas-de-tecnologia/post/o-que-mudou-na-velha-batalha-linux-x-*Windows*.html > Acesso em: 18 Jun. 2016.

[33] NETCRAFT. **September 2015 Web Server Survey.** Disponível em: < http://news.netcraft.com/archives/2015/09/16/september-2015-web-server-survey.html> Acesso em: 18 Jun. 2016.

[34] PEDRO PISA. **A história da *Microsoft*.** Disponível em: < http://www.techtudo.com.br/artigos/noticia/2012/05/a-historia-da-*Microsoft*.html > Acesso em: 09 Jul. 2016.