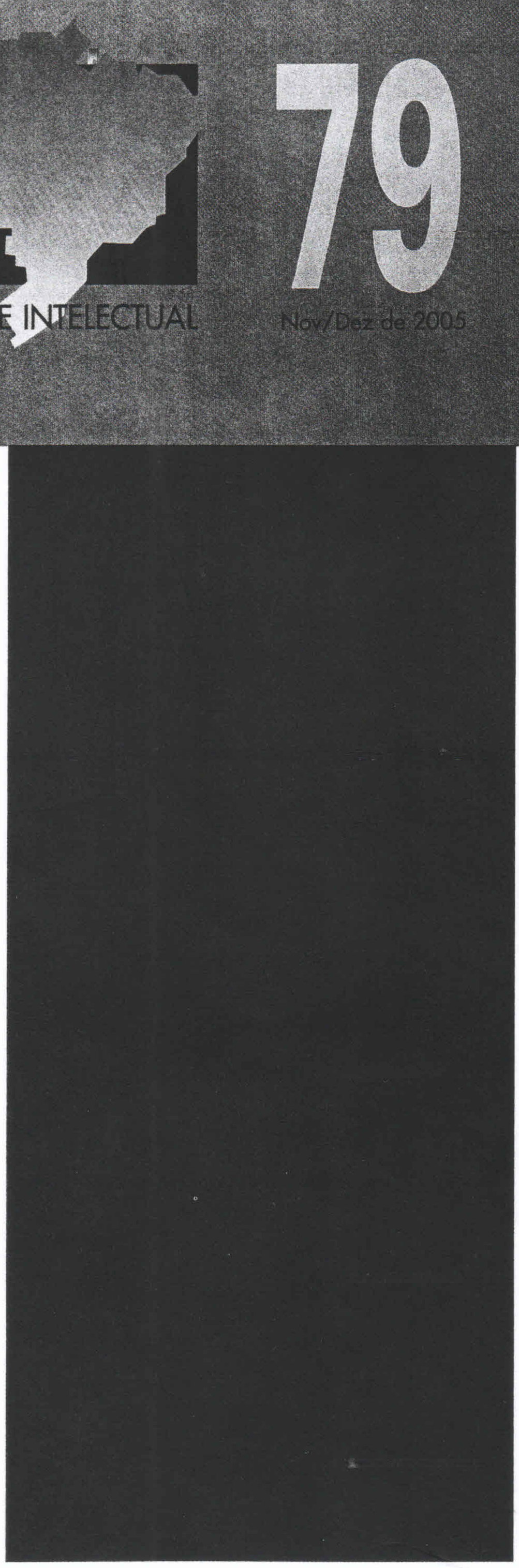
REVISTA DA
ABPI
ASSOCIAÇÃO BRASILEIRA DA PROPRIEDADE INTELECTUAL

79

Nov/Dez de 2005



A NATUREZA JURÍDICA DO SOFTWARE À LUZ DA LINGÜÍSTICA[1]

Eduardo Tomasevicius Filho
Bacharel em Direito pela USP e Doutorando em Direito Civil na mesma Universidade. Advogado.
Co-Autor do livro "Código Civil Comparado" (2ª edição, São Paulo: LTr, 2003)

Sumário: 1. Introdução - 2. Histórico do Problema - 3. Tratamento Legislativo da Matéria pelos Países - 4. Análise Técnica do Software - 5. Análise Lingüística do Software - 6. O Software na Perspectiva da Lingüística - 7. Conclusão - 8. Referências Bibliográficas

1. Introdução

O objetivo desse artigo consiste no estudo da natureza jurídica do software à luz da lingüística.

Entre as décadas de 1960 a 1980, o debate sobre a proteção jurídica destas obras intelectuais foi intenso entre os juristas no mundo inteiro. A informática era uma novidade na vida das pessoas e das empresas, e a necessidade de proteção dos investimentos feitos em pesquisa e desenvolvimento do software exigiu diversos estudos sobre o tema.[2]

Apesar de se ter chegado à conclusão de que o melhor sistema de proteção seria o direito de autor, a lingüística pôde enriquecer a compreensão da natureza jurídica destas obras. Deseja-se, pois, contribuir para uma melhor sustentação da natureza jurídica do software.

A metodologia empregada nesse estudo consiste na análise do software a partir da visão técnica do mesmo, para, em seguida, apontar, com base na lingüística, as semelhanças que estas obras têm com as demais obras tradicionalmente protegidas pelo direito de autor.

2. Histórico do Problema

O primeiro pedido de registro de um software de que se tem notícia foi em 1961 nos Estados Unidos. Evidentemente, não se sabia naquela época como se deveria proteger essas criações intelectuais, nem qual seria a melhor forma para essa proteção.

Em 1971 a Organização Mundial da Propriedade Industrial (OMPI), atendendo a um pedido da ONU, constituiu um grupo consultivo de técnicos para discutir a proteção de softwares e preparar um relatório sobre qual seria sua melhor forma de proteção.[3]

Esse grupo elaborou em 1978 as Disposições-Tipo da OMPI (WIPO Model Provisions on the Protection of Computer Software), que traziam as seguintes definições:

a) Software: conjunto de instruções que, uma vez incorporados a um suporte legível pela máquina, possa fazer com que uma máquina seja capaz de processar informações, realize ou obtenha uma função, uma tarefa ou um resultado específico.

b) Descrição do programa: apresentação completa de procedimentos em forma verbal, esquemática ou outra, suficientemente detalhada para determinar um conjunto de instruções que constituam o software correspondente.

c) Material auxiliar: todo material distinto de um software ou de uma descrição de um programa, criado para facilitar a compreensão ou a aplicação de um software, por exemplo, descrições de problemas de instruções para o usuário.

De acordo com o inciso "iv" do artigo I das Disposições-Tipo, só se considerava programa de computador o software e o material auxiliar. Além disso, essas Disposições-Tipo traziam regras sobre o prazo de proteção ao software e o acesso à justiça em caso de violação. Por não mencionarem explicitamente qual era o regime aplicável (direito de autor, propriedade industrial, concorrência desleal, contratos, etc), entendeu-se que se tratava de um direito *sui generis*. No entanto, essa idéia não foi incorporada pelos países. A partir da década de 1980, passou a entender-se que o melhor regime seria o direito de autor e que, portanto, não seria necessário uma convenção internacional, ou um direito *sui generis*, que tivesse por objeto a proteção jurídica do software.[4]

---

1. Este artigo é adaptação do trabalho financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (1999-2000), sob orientação do Prof. Fábio Maria De-Mattia.
2. Kindermann, Manfred. "O direito do autor internacional e a proteção do software. Histórico, situação atual e fatos novos", tradução de Bruno Jorge Hammes. in Estudos [illegible] 1989, p. 69
3. Baptista, Luiz Olavo. "A Proteção dos Softwares em Direito Comparado e Internacional". in Revista de Direito Mercantil, São Paulo. vol. 22, nº 50, abr/jun 1983, p. 27
4. Kindermann, Manfred. op.cit. p. 73

Os argumentos das principais correntes serão expostos a seguir:[5]

2.1. Direito de Autor

Esta corrente sustentava que o direito de autor seria o melhor sistema de proteção, porque o software consiste numa obra intelectual, original, que resulta do esforço de seu criador.[6] Como o direito de autor protege a forma da expressão consubstanciada num *corpus mechanicus* (papel, meios magnéticos, ópticos, etc), independentemente do caráter estético da obra, o software atenderia aos requisitos estabelecidos para proteção.[7]

Além disso, os direitos morais do autor (de paternidade, de nominação, ao inédito, à integridade, ao arrependimento, à modificação) poderiam ser estendidos ao criador do software.

Tendo em vista a facilidade em sua reprodução, o direito de autor preservaria melhor o segredo da criação, inclusive por não ser obrigatório o registro, e ser protegido desde o seu lançamento no mercado. Por último, o direito de autor está assentado em larga experiência legislativa, doutrinária e jurisprudencial.[8]

No entanto, a principal crítica contra a proteção por meio do direito de autor foi que este sempre esteve voltado à proteção de obras intelectuais, que se dirigem aos sentidos e ao espírito dos destinatários, indeterminados da mensagem, a qual não se aplicaria ao software, pois não pode ser percebido pelos sentidos humanos[9], e que o software não comportaria direitos morais de autor.

2.2. Propriedade Industrial (Patentes)

Considerando-se a natureza da obra tecnológica, a finalidade essencialmente comercial e o caráter utilitário deste tipo de produção intelectual, a proteção, mediante a concessão de patentes, poderia ser uma alternativa à proteção do software.

Sob esse ponto de vista, o software seria um processo de resolução prática de determinados problemas por parte do hardware, não se confundindo com os resultados que o homem deseja. Além disso, quando em código objeto, e expresso em linguagem legível por máquina e não em linguagem humana. Logo, seria uma obra puramente técnica, destinada ao hardware, não ao ser humano.[10]

O problema dessa concepção é que o software poderia ser considerado um método, que, até os dias atuais, não é patenteável.[11] Além disso, o software não é o próprio processo produtivo, pois mesmo quando o programa se destina a este, é a máquina quem o realiza.[12]

2.3. Direito *Sui Generis*

Os defensores do direito *sui generis* de proteção do software desejavam criar um mecanismo que tivesse regras específicas para a proteção do software. Essa corrente atraiu defensores, porque o regime de proteção pelas patentes de invenção não se adequava à proteção de "métodos" e o direito de autor tradicionalmente se destinava à proteção de obras artísticas, literárias e científicas, voltadas para os seres humanos, e concedia proteções que não eram interessantes à indústria do software, como o caso dos direitos morais de autor.

Como relatou à época Luiz Olavo Baptista,[13]

"(...) a matéria evolui, e evolui sem levar em conta interesses poderosos como os da IBM e dos grandes fabricantes que melhor se acomodariam em um regime *sui generis*, dos consumidores ignorados em quase todas as legislações e abandonados pelo novo projeto do executivo brasileiro, ao contrário daqueles oriundos do Senado (Chiarelli) e da Câmara (Cristina Tavares) e dos países que vêem mais uma prática monopolística crescer".

3. Tratamento Legislativo da Matéria pelos Países

Com o crescimento e desenvolvimento da informática, os países foram obrigados a legislar sobre a proteção jurídica do software, exigindo-se a tomada de posição sobre qual sistema seria adotado. O primeiro país em que se legislou sobre o assunto foi nas Filipinas em 1972. Desde então, verificou-se a tendência de proteção pelo direito de autor, seja através da emenda das legislações de direito de autor para incluir o software como obra protegida, ou pela criação de leis especiais.

Nos Estados Unidos, como já dito acima, o *Copyright Law of United States* não trazia o software como obra protegida por essa legislação. Somente em 1978 a Comissão Nacional sobre Novos Usos Tecnológicos das Obras Protegidas por *Copyright* dedicou-se a estudar os efeitos da proteção de direito de autor ao *software*. Naquela época não se sabia quais tipos de *softwares* deveriam ou não ser protegidos, pois se tinha em mente a distinção entre código e o *corpus mechanicus*. Em 1980, o Congresso Norte-Americano aprovou o *Computer Software Copyright Act*, estendendo ao software a proteção pelo direito de autor norte-americano.

Na França, por seu turno, também não havia uma legislação que cuidasse da proteção jurídica do software. Assim, a lei francesa de di-

---

5. As outras correntes eram a) desnecessidade de proteção; b) proteção contratual; c) concorrência desleal; d) direitos conexos. Cf. Manso, Eduardo Vieira. *A Informática e os Direitos Intelectuais*. São Paulo: RT, 1985
6. Hammes, Bruno Jorge. "O Programa de Computador segundo a Lei nº 7.646, de 18.12.1987". in Estudos Jurídicos. São Leopoldo, v. 21, nº 52, mai/ago 1988, p. 24
7. Neto Lobo, Paulo Luiz. "Direito de Autor de Software", in Revista Forense, Rio de Janeiro, v.85. n.305. jan./mar. 1989, p. 86
8. Santos, Manoel Joaquim Pereira dos. "A proteção adequada ao 'software'". in Revista de Direito Civil, Imobiliário, Agrário e Empresarial. São Paulo, São Paulo, v.11. nº 40. abr./jun. 1987, p. 134
9. Gomes, Orlando. "A proteção dos softwares". in Gomes, Orlando (org). *A proteção jurídica do software*. Rio de Janeiro: Forense, 1985, p. 6
10. Santos, Manoel Joaquim Pereira dos. op.cit. p. 135
11. Baptista, Luiz Olavo. "A proteção...". p. 31
12. Ascensão, José de Oliveira. "Software e Direito Autoral". in Gomes, Orlando (org). *A proteção jurídica do software*. Rio de Janeiro: Forense, 1985, p. 59
13. Baptista, Luiz Olavo. "Proteção Jurídica do Software - Novos Desenvolvimentos", in Revista Forense, Rio de Janeiro. v.84. n.301. jan./mar. 1988. p. 52

reito de autor (Lei nº 298, de 1957) foi emendada pela Lei nº 660 de 1985, para proteger o *software*, ao incluir no artigo 1º, alínea 5, do título 1, a proteção do software pelo sistema de direito de autor.

A Diretiva 91/250 da União Européia, de 14 de maio de 1991, disciplinou a proteção jurídica do *software*. Acerca de questões técnicas referentes ao mesmo, esta Diretiva estabeleceu, entre outras coisas, que:

a) o termo *programa de computador* significa igualmente os trabalhos preparatórios de concepção e desenvolvimento de um programa, sob a condição de que permitam posteriormente a realização de um programa de computador;

b) para evitar ambigüidade, só a expressão de um programa de computador é protegida e que as idéias e os princípios que são a base dos diferentes elementos de um programa, bem como as interfaces, não são protegidos pelo direito de autor em virtude da presente diretiva;

c) as idéias e os princípios que são a base da lógica, dos algoritmos e das linguagens de programação não são protegidos em virtude da presente diretiva.

O artigo 1º da Diretiva prevê que a proteção jurídica estende-se a toda forma de expressão de um programa de computador, ou seja, tanto os códigos, escritos em uma linguagem de programação, quanto o programa armazenado sobre um *corpus mechanicus*, são protegidos juridicamente; por outro lado, as idéias e os princípios que são a base de algum elemento que componha um programa de computador não são protegidos.

Na Itália, a fim de harmonizar sua legislação com a Diretiva 91/250, emendou-se a Lei nº 633, de 22 de abril de 1941, que trata da proteção dos direitos de autor de obra intelectual por meio do Decreto nº 518, de 29 de dezembro de 1992.

Embora não existisse na Itália qualquer disposição acerca da proteção jurídica do *software* até à promulgação do Decreto nº 518, a jurisprudência tinha decidido que a sua proteção deveria ser feita pelo direito de autor, e não pela propriedade industrial.[14] O artigo 1º do Decreto nº 518 adicionou os programas de computador como obras a serem protegidas pela lei de direito de autor.

Além disso, o autor de *software* na Itália passou a ter direitos semelhantes aos direitos morais de autor de obras literárias e artísticas, tais como o direito de impedir que se modifique a sua obra, ou impedir que se pratique qualquer ato que venha a prejudicar sua honra ou reputação. Pode, também retirar a obra de circulação ou manter a obra inédita, de acordo com os artigos 20 a 22 da Lei nº 633.

O Brasil teve duas leis sobre proteção jurídica do software. A primeira delas foi a Lei nº 7.646/87 e a segunda, a Lei nº 9.609/98.

A Lei nº 7.646 foi elaborada para atender às reclamações dos Estados Unidos, que desejavam um sistema de proteção dos softwares no Brasil. Esta lei foi revogada pela Lei nº 9.609, a qual veio adequar o regime de proteção à abertura do país para o comércio internacional de bens, serviços e propriedade intelectual ocorrida na década de 1990 no âmbito da Organização Mundial do Comércio.

O artigo 2º de ambas as leis estabelecem que a proteção do software no Brasil seria feita pelo regime conferido às obras protegidas pelo direito de autor, ressalvadas as modificações necessárias que ambas as leis traziam para efetivar tal proteção.

4. Análise Técnica do Software

A fim de que se possa compreender a natureza jurídica do software, é imprescindível entender o funcionamento do computador, para que, numa etapa seguinte, se faça a análise do software numa perspectiva da lingüística.

4.1. O Computador

Um computador é uma máquina processadora de dados. Sua finalidade é dar-lhes um tratamento racional, fornecendo, ao final, um resultado em função dos mesmos. O funcionamento dos atuais computadores é mais fácil de ser compreendido quando se conhece a evolução dessas máquinas.

Os primeiros computadores foram construídos na Idade Moderna. Blaise Pascal foi o inventor da primeira máquina automática de

14. Kindermann, Manfred. op.cit, p. 81

calcular, constituída de engrenagens, muito parecida com os atuais odômetros dos veículos, que simulava o funcionamento de um ábaco. Nesse caso, os dados eram os números a serem somados ou subtraídos, que, processados pelas engrenagens, produziam um resultado no final da operação. O computador de Pascal tinha uma estrutura rígida, por realizar apenas essas operações matemáticas.[15]

Joseph Marie Jackuard construiu o primeiro computador programável. Era um tear controlado por cartões perfurados, cujo funcionamento mecânico era semelhante às caixinhas de música. As estampas dos tecidos confeccionados por esse tear eram resultado da seqüência dos furos dos cartões nele introduzidos.[16] As informações representadas pelos furos em uma determinada seqüência no cartão eram processadas e produziam o resultado desejado: as estampas nos tecidos confeccionados. Embora tivesse uma estrutura fixa, a possibilidade de troca desses cartões conferia uma maior versatilidade a esse computador em relação ao computador de Pascal.

A primeira aplicação dos computadores no tratamento de um grande número de informações ocorreu no final do século XIX, nos Estados Unidos, no recenseamento da população. Os censos, naquela época, levavam cerca de dez anos para serem concluídos. Hermann Hollerith, funcionário do Departamento de Recenseamento dos Estados Unidos, percebeu que as respostas às perguntas do censo eram muito simples: ou sim ou não. Desse modo, Hollerith utilizou cartões perfurados, muito semelhantes aos cartões utilizados por Jackuard em seu tear, para coletar estas informações e uma máquina capaz de lê-las e tabulá-las. Esta máquina ficou conhecida como Máquina de Recenseamento ou Tabuladora. Seu uso no censo de 1890 permitiu a sua conclusão em três anos, não só pela velocidade, mas também pela redução dos erros. Em 1895, Hollerith tentou, pela primeira vez, dar uma aplicação comercial à sua máquina, utilizando-a na contabilidade das Ferrovias Centrais de Nova York. Em 1896, Hollerith fundou a Tabulating Machines Company, empresa esta que, em 1924, transformou-se na IBM.[17]

Em 1940, John W. Mauchly e J. Presper Eckert Jr, junto com cientistas da Universidade da Pensilvânia, construíram o primeiro computador eletrônico, o ENIAC (Electronic Numerical Integrator and Calculator), que entrou em funcionamento em 1945. Foi um projeto do Exército dos Estados Unidos para o cálculo de trajetórias de projéteis. Nesse caso, o aumento da versatilidade do computador decorria da modificação constante de sua estrutura, o que exigia várias pessoas que interligavam suas partes por meio de cabos elétricos.[18]

Nessa mesma década, o matemático húngaro John von Newmann desenvolveu a idéia de software – embora não tivesse utilizado esse termo – e descreveu o fundamento teórico da construção de um computador eletrônico, que acabou sendo denominado Modelo de von Newmann. Ele demonstrou que um computador poderia ter uma estrutura simples e fixa, capaz de executar qualquer tipo de instrução dada sem a necessidade de modificar sua estrutura a todo momento, como ocorria com o ENIAC, que necessitava de homens interligando fios de um ponto a outro. Von Newmann contribuiu com uma nova concepção de organização e construção de computadores. Suas idéias tornaram-se fundamentais para a indústria da informática, pois resultaram em computadores mais rápidos, flexíveis e eficientes.[19]

Com o advento da Eletrônica, foi possível criar computadores eletrônicos em chips. Em razão de suas reduzidas dimensões, sua produção em larga escala, aplicação genérica e extrema rapidez, permitiu-se a fabricação de computadores pessoais (PC's), levando, enfim, à massificação do uso do computador.

A partir das idéias de Von Newmann, um computador passou a ter duas partes fundamentais: a parte física, denominada hardware, e a parte lógica, denominada software.

4.2. O hardware

O hardware é a parte física do computador. É composto pelos dispositivos de entrada de dados, pelas memórias e pela unidade central de processamento (CPU).

15. Alcalde, E; Garcia, M; Penuelas, S. *Informática Básica*. São Paulo: Makron Books, 1991, p. 10
16. Alcalde, E; Garcia, M; Penuelas,, S. *op.cit.* p. 11
17. Alcalde, E; Garcia, M; Penuelas,, S. *op.cit.* p. 14
18. Alcalde, E; Garcia, M; Penuelas,, S. *op.cit.* p. 18
19. Alcalde, E; Garcia, M; Penuelas,, S. *op.cit.* p. 18

Os dispositivos de entrada de dados são aqueles que permitem ao usuário inserir informações no computador. São eles o teclado, o *mouse* e o *scanner*. Os dispositivos de saída de dados são aqueles que fornecem ao usuário do computador os dados processados. São eles o monitor e a impressora. Há também os dispositivos mistos, que tanto inserem dados no computador, quanto os envia, por exemplo, o modem.

As memórias armazenam dados. Podem ser comparadas a uma estante com prateleiras de igual tamanho, enumeradas em ordem crescente. Podem ser divididas de acordo com o meio de armazenamento e à volatilidade dos dados. Quanto ao método de armazenamento, há os meios magnéticos (fitas cassete, disquetes e disco rígido), os meios mecânicos (cartões perfurados) e os meios ópticos (CD-ROM).

Quanto à volatilidade dos dados, existem memórias que só permitem a leitura dos dados que armazena, não sendo possível apagar os dados nela contidos, nem podendo introduzir-se novos dados e memórias que permitem não só a leitura dos dados, como também que nela se insiram novos dados e que os apague. As primeiras recebem o nome de ROM (Read Only Memory) e as segundas, RAM (Random Access Memory).

O processador é um complexo circuito eletrônico contido em um chip, capaz de processar dados. É o cérebro do computador.

É composto por, no mínimo, dois circuitos: a Unidade Lógico-Aritmética (ULA) e a Unidade de Controle (UC). A ULA é um circuito lógico cuja função é realizar operações lógicas e aritméticas. Consiste na realização de operações aritméticas fundamentais, nas operações da lógica formal, tais como a negação (NOT), a disjunção (OR), e a conjunção (AND), podendo realizar outras operações, de acordo com a evolução do processador em questão.

A UC controla todas as operações do computador. Ordena à ULA que realize as operações lógico-aritméticas e gerencia a utilização e o fluxo de dados entre a memória, os dispositivos de entrada e saída do computador e o próprio processador.

4.3. O software

O software, também chamado de programa de computador, controla o hardware. Dessa maneira, um único hardware pode ser utilizado em tantas aplicações, quantos forem os softwares utilizados. Pode-se compará-los ao automóvel e seu motorista. O hardware seria o automóvel, que pode ser utilizado em diversas atividades, como por exemplo, o transporte de pessoas ou de coisas. O software seria o motorista do automóvel, que; além de dirigir o automóvel, decide como este será usado.

Existem dois tipos de software:

a) Software Básico ou Sistema Operacional: é o conjunto de programas imprescindíveis ao funcionamento do computador. É um software que controla o funcionamento dos softwares aplicativos.

b) Software Aplicativo: é o conjunto de programas a serem utilizados pelo usuário.

Os softwares aplicativos realizam as mais variadas tarefas, tais como a edição de textos, elaboração de planilhas, construção de bancos de dados, controle de sistemas, etc. O software básico, também chamado de sistema operacional, tem por função controlar o funcionamento dos componentes do hardware e escalonar as tarefas para que o processador trabalhe com apenas uma de cada vez, embora pareça ao usuário estar executando mais de um programa ao mesmo tempo. Por exemplo, é o sistema operacional que cria a ilusão no usuário de que se está utilizando, ao mesmo tempo, o editor de textos, a Internet e ainda ouvir música de um CD, embora o processador esteja utilizando um software por vez.

4.4. O funcionamento dos computadores modernos[20]

Os computadores modernos funcionam com energia elétrica. Em linhas gerais, a técnica utilizada para seu funcionamento é valer-se de dois estados físicos: ligado ou desligado. Isso é semelhante aos cartões perfurados utilizados nos computadores mais rudimentares. Ao invés de furarem cartões, ligam-se ou desligam-se circuitos elétricos. Na informática, cada um desses estados físicos recebe o nome de bit.[21]

Um bit apenas é capaz de transmitir duas únicas informações: cada uma correspondente a um estado elétrico. Tendo em vista existirem só esses dois estados, convencionou-se que a ausência de tensão elétrica em um circuito corresponde ao número "0" e a presença de tensão, ao número "1".

Figura 1:
Tabela de correlação de estados elétricos e estados lógicos

Estado Elétrico	Estado Lógico
Ligado	1
Desligado	0

No entanto, há uma grande limitação de veiculação de informações, já que as duas informações possíveis de serem representadas são os números "0" e "1". Por essa razão, convencionou-se trabalhar com 8 bits agrupados, isto é, o computador, ao invés de ler um bit por vez, lê oito bits de uma vez como se fosse um único número. A esses deu-se o nome de byte.

20. Aqui são descritos modelos de computadores extremamente simplificados.

21. O Código Morse utiliza o mesmo princípio para a transmissão de informações.

Figura 2
Exemplo de byte, equivalente ao número 7D em base hexadecimal

0	1	1	0	1	1	0	1

Aqui já é possível representar mais informações. Considerando que cada casa pode assumir duas combinações – 0 ou 1 – basta calcular todas as combinações e verificar ser possível representar 256 números.

Porém, as pessoas não trabalham com números binários no dia a dia. Trabalham com números decimais. Desse modo, a vantagem de trabalhar-se com um byte é ser possível convertê-lo com mais facilidade para números decimais, como segue a figura abaixo:

Figura 3
Quadro representativo dos números em base binária e hexadecimal

Binário	0 0 1 1	0 1 1 0
Hexadecimal	3	6
Binário	0 0 0 1	1 1 0 0
Hexadecimal	1	C

A fim de ilustrar de que forma é realizado o processamento de dados pelo computador, será utilizado um processador "ideal", extremamente simples, que apenas seja capaz de somar dois números contidos em um dispositivo de memória.

Todo processador possui um manual, no qual estão contidos os códigos de operação do mesmo. Supondo-se que, toda vez que for fornecida ao processador uma determinada seqüência de números, deverá ser realizada uma determinada operação matemática:

Assim, se for fornecido ao processador a seqüência de três bytes, correspondente aos números 37, 51 e 30, tal como representado abaixo, o processador verificará quais são os dois números contidos nas posições de memória de número 60 e 61 e os somará, gravando o resultado na posição número 62.

Figura 4
Tabela de comandos e operações do processador ideal usado neste texto

Seqüência de Comandos	Operação
37 - 71 - 50	Soma de dois números gravados nas posições 60 e 61 e grava o resultado na posição 62.

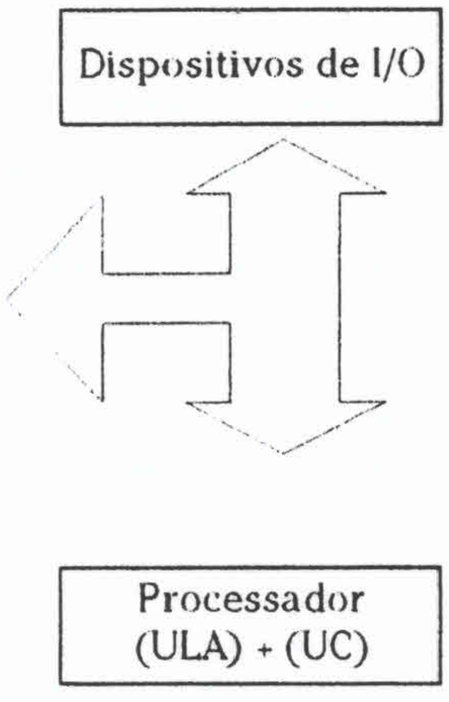
Figura 5
Representação em bytes da seqüência de comandos

0	0	1	1	0	1	1	1	0	1	1	1	0	0	0	1	0	1	0	1	0	0	0	0
3				7				7				1				5				0			
37								71								50							

Então, o processador verificará a instrução contida na posição número 0 da memória. Em seguida, verificará a instrução contida na posição 1; depois, na posição 2 e assim por diante.

Figura 6
Esquema do computador idealizado neste texto

Memória	
Posição	Dados
00	
01	
02	
(...)	
10	37
11	71
12	50
(...)	
60	12
61	15
62	27
(...)	



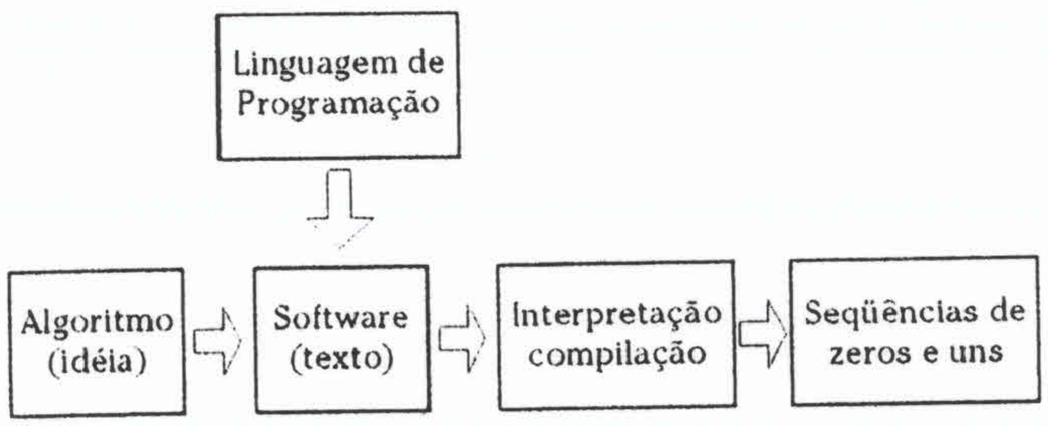
Av. Paulista nº 1294, 16º - Cerqueira César
São Paulo - SP - Brasil - CEP.: 01310.915
fone.:(55 11) 3372.3766 - fax.:(55 11) 3372.3767 / 68 / 69
mail@daviddonascimento.com.br
www.daviddonascimento.com.br

Para ilustrar, o leitor deve olhar na figura acima os dados contidos na memória deste computador "ideal". O processador lê na memória o código contido na posição 10 (37); depois lê o código da posição 11 (71); depois lê o código da posição 12 (50). Ao constatar essa seqüência, o processador soma os números contidos nas posições 60 e 61 (no caso, os números 12 + 15) e grava o resultado na posição 62 (=27).

4.5. A Criação do Software

Não é nada fácil operar um computador da maneira como foi feita acima, utilizando-se diretamente os comandos do processador através da seqüência de números, ainda mais tendo em vista que um software realiza milhões de operações para produzir os resultados desejados pelo seu criador. Por essa mesma razão, foram desenvolvidas as linguagens de programação. O criador de software escreve um programa em uma determinada linguagem e, através dos processos de interpretação e compilação, geram-se todos os bytes que realizam a operação desejada. Graficamente o procedimento é o seguinte:

Figura 7
Fluxograma de criação do software



A seguir, será descrito sucintamente o processo de desenvolvimento de um software. No caso, o problema a ser resolvido pelo computador é permitir ao usuário calcular as raízes reais de uma equação de segundo grau.

Uma equação do segundo grau possui a seguinte estrutura: $ax^2 + bx + c = 0$

O programador do software terá de analisar esta equação e criar uma seqüência de instruções para serem executadas pelo computador. Necessariamente serão formuladas as seguintes perguntas, que delimitarão o problema a ser resolvido:

a) Serão calculadas apenas as raízes reais? E em caso de não existirem raízes reais?

b) Será fornecida ao usuário a coordenada do vértice da parábola?

c) Será fornecida ao usuário a concavidade da parábola?

d) Haverá alguma mensagem se "a" for igual a zero?

Respondidas estas perguntas, será necessário estabelecer uma seqüência correta de instruções para a operação. Neste exemplo, o software calculará as raízes reais de uma dada equação do segundo grau, mediante a digitação pelo usuário dos valores de "a", "b" e "c", indicando a este, se houver, quais as raízes, a concavidade da parábola e as coordenadas de seu vértice. Na hipótese de erro, avisará essa ocorrência ao usuário.

Definida esta primeira parte, o programador passará a elaborar o modo de operação, trabalhando com as várias possibilidades de se chegar ao resultado. Na solução deste problema, utilizará as fórmulas:

Figura 8
Fórmulas para a obtenção das raízes reais de uma equação do segundo grau

$$\Delta = b^2 - 4ac$$
$$X_v = -b/2a$$
$$Y_v = \Delta/4a$$
$$X_1 = b + \Delta/2a$$
$$X_2 = b - \Delta/2a$$

E as seguintes condições:

Se $\Delta > 0$, a equação tem raízes reais;

Se $\Delta < 0$, a equação não tem raízes reais;

Se $a > 0$, a concavidade da parábola é para cima;

Se $a < 0$, a concavidade da parábola é para baixo;

Se $a = 0$, não há equação do segundo grau.

A obtenção da curvatura da parábola independe do cálculo das raízes, porque esta é obtida com a utilização dos coeficientes da equação. Mesmo assim, é imprescindível verificar se "a" é igual a zero, pois, se for igual, não se inicia o processo e se evita a divisão por zero, já que "a" é denominador das fórmulas das raízes e das coordenadas do vértice.

A etapa seguinte consistirá em encontrar uma forma de representar estas informações. Em geral, é uma tarefa difícil, não determinável pelos recursos disponíveis, mas também depende das operações a serem realizadas sobre os dados. A partir desta análise, chega-se a uma idéia do modo pelo qual o computador irá funcionar. Esta esquematização recebe o nome de algoritmo. Um algoritmo é a descrição de um padrão de comportamento, expressado em termos de um repertório bem definido e finito de ações primitivas, que provavelmente podem ser executadas[22]. É um texto estático, capaz de cobrir toda uma classe de acontecimentos.[23]

e) O algoritmo para o problema é o seguinte:

1. Apresentação do programa na tela do computador;
2. Solicitar ao usuário o valor de "a";
3. Verificar se "a" é igual a zero. Se for igual a zero, ir para a instrução 4; se não for igual a zero, ir para a instrução 5.
4. Avisar na tela que não é possível o cálculo, porque "a" é igual a zero.
5. Solicitar ao usuário o valor de "b";
6. Solicitar ao usuário o valor de "c";

22. Wirth, Niklaus. *Algoritmos e Estruturas de Dados*. Trad. de Cheng Mei Lee, Revisão de João José Neto. Rio de Janeiro: Ed. Prentice-Hall, 1989, pp. 1-3

23. Guimarães, Ângelo de Moura; Lages, Newton Alberto de Castilho. *Algoritmos e Estruturas de Dados*. Rio de Janeiro: LTC, 1993, p. 7

7. Cálculo do Δ;
8. Cálculo do Xv;
9. Cálculo do Yv;
10. Verificação se Δ < 0; se for menor que zero, ir para a instrução 11; se não for, ir para a instrução 12;
11. Avisar na tela que a equação não possui raízes reais e indicar a concavidade da parábola e as coordenadas do vértice;
12. Calcular as raízes reais;
13. Avisar na tela que a equação possui raízes reais e indicar a concavidade da parábola e as coordenadas do vértice;
14. Fim.

O algoritmo possui duas características. A primeira delas é a linearidade: não é possível realizar todas as instruções ao mesmo tempo. A segunda delas é a dependência: há instruções que não podem ser executadas sem outras. Por exemplo, a ausência da instrução 3 poderia causar problemas na obtenção dos resultados; não é possível que a instrução 12 venha antes das instruções 2, 5 e 6.

f) As linguagens de programação

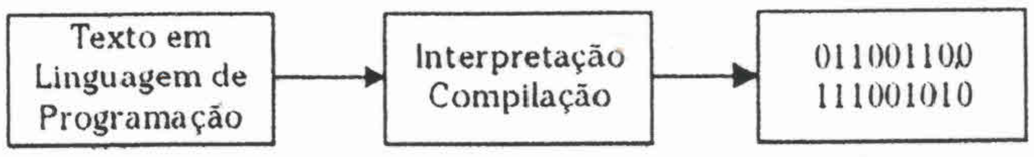
Elaborado o algoritmo, o programador irá escrever o software em uma linguagem de programação.

Há dois níveis de linguagem de programação: as linguagens de baixo nível e as linguagens de alto nível. A linguagem de baixo nível foi a primeira tentativa de substituição da linguagem de máquina por uma outra, mais próxima do entendimento do ser humano. Aqui, cada instrução equivale a uma instrução em linguagem de máquina, utilizando-se mnemônicos. Já as linguagens de alto nível permitem uma maior independência em relação ao processador, pois o texto escrito em linguagem de programação é convertido para operar qualquer processador. Não só isso, é muito mais próximo das diversas línguas, pois utilizam palavras de uso corrente.

No caso, será escolhida a linguagem C para a elaboração deste software que calcula as raízes reais de uma equação de segundo grau. Trata-se de uma linguagem de alto nível, que substitui as milhares de operações matemáticas, tais como a que foi simulada acima.

A título de ilustração, o texto escrito em Linguagem C, capaz de calcular as raízes da equação de segundo grau, é o seguinte:

Figura 9
Software, escrito em Linguagem C, que calcula as raízes reais de uma equação do segundo grau



Escrito o software em uma linguagem de programação, será submetido ao processo de compilação, que consiste na sua análise do ponto de vista léxico (se há palavras escritas de forma errada), semântico (se foi incluída alguma função ou palavra que não pertença à linguagem) e sintático (se o texto está escrito na seqüência correta). Se não houver problema algum, este texto é transformado em uma seqüência de bytes que serão executados pelo processador do computador, tornando-se um software executável (arquivos com extensão "exe").

Figura 10
Fluxograma de compilação do software em seqüência de bytes

```
main( )
{
#include <math.h>
int a,b,c,d,x,y,p,n,z,j;
printf("Programa de Cálculo de Raízes de Equação do Segundo Grau.");
printf("\n Digite o valor de a: ");
scanf("%d",&a);
while(a = = 0)
{printf("\n Não existe equação do segundo grau com este coeficiente. Tente novamente.");
printf("\n Digite o valor de a: ");
scanf("%d",&a);
}
printf("\n Digite o valor de b: ");
scanf("%d",&b);
printf("\n Digite o valor de c: ");
scanf("%d",&c);
d = (b * b) - (4 * a * c);
x = - ( b ) / (2 * a);
y=-( d ) / (4 * a);
if(d < 0)
{
printf("Esta equação não possui raízes reais.");
printf("\n As coordenadas do vértice são %d e %d.",x,y);
if(a>0)
printf("\n A concavidade da parábola é para cima.");
if(a < 0)
printf("\n A concavidade da parábola é para baixo.");          }
else
{
printf("Esta equação possui raízes reais!");
z = sqrt ( d );
p =( b + z) / (2 * a);
n = ( b - z ) / (2 * a);
printf("\n As raízes são %d e %d.",p,n);
printf("\n As coordenadas do vértice são %d e %d.",x,y);
if(a > 0)
printf("\n A concavidade da parábola é para cima.");
if(a < 0)
printf("\n A concavidade da parábola é para baixo.");
}
}
```

5. Análise Lingüística do Software

5.1. Introdução

Para entender qual é a natureza jurídica do software, alguns conceitos de lingüística são necessários. Deve-se atentar para o fato de que os modelos a serem utilizados nesse estudo foram desenvolvidos para a linguagem humana, mas são igualmente válidos para o entendimento do software.

A lingüística é a ciência que tem por objeto a linguagem. Esta foi analisada por vários estudiosos, que formularam diversas teorias sobre ela.

Na análise do software, serão utilizados os modelos de linguagem de Ferdinand de Saussure e Louis Hjelmslev.

5.2. O modelo de Ferdinand de Saussure

Ferdinand de Saussure (1857-1913) foi um lingüista suíço e professor da Universidade de Genebra. Fez da lingüística uma ciência autônoma. Seu modelo de linguagem constituiu a base dos estudos modernos, uma vez que os demais teóricos construíram seus modelos de linguagem com base em seu pensamento.

Seu modelo de linguagem tem dois aspectos. O primeiro deles refere-se à sua estrutura e normatividade da linguagem, a que ele denominou língua. O segundo aspecto refere-se à operação da linguagem pelo indivíduo, a que ele denominou fala. Embora a língua esteja sempre em evolução, é um sistema preexistente, constituído pelo conjunto dos hábitos que permitem um indivíduo compreender e fazer-se compreender.[24] Por esta razão, a língua tem um aspecto social. Por outro lado, a fala, por ser a concretização do uso da língua, teria um aspecto individual. Saussure utiliza uma analogia com a matemática, ao afirmar que a linguagem seria a soma da língua com a fala.

Para Saussure, as relações e as diferenças entre termos lingüísticos se desenvolvem em dois eixos que se cruzam transversalmente, gerando cada um certa ordem de valores. O primeiro eixo é formado pelo sintagma. Um sintagma é uma seqüência de signos, linear e irreversível.[25] Significa que não se pode produzir todos os signos de uma mensagem de uma única vez, existindo relações seqüenciais entre os termos decorrentes do encadeamento dos signos. O segundo eixo é formado pelo paradigma. Significa que, ao criar uma mensagem, escolhe-se um signo entre um conjunto de signos de mesmo significado. Esses fenômenos foram denominados de relações sintagmáticas e relações associativas, respectivamente.

Graficamente, pode-se dar o seguinte exemplo:[26]

Figura 11
Quadro demonstrativo das relações entre paradigma e sintagma na produção de um texto

O A	bola	entrou	no na	véu da noiva	>paradigma>
	balão	adentrou		rede	
	pelota	penetrou		gol	
(1)	(2)	(3)	(4)	(5)	
>sintagma>					

Para se formar uma mensagem a partir do quadro acima, é necessário obedecer a uma seqüência linear; no caso, (1) + (2) + (3) + (4) + (5). Mas também se deve escolher qual termo será utilizado para ocupar cada posição. Logo, pela seqüência de termos se forma o sintagma e pela escolha de um entre vários termos se forma o paradigma. Pode-se perceber, ainda, que a escolha de um dos termos no eixo paradigmático implicará uma restrição do número de termos a serem utilizados no momento seguinte à elaboração do sintagma. Por exemplo, no quadro acima, se o sintagma se iniciar com o artigo definido "O", só é possível usar o termo "balão". Se se iniciar com o artigo definido "A", pode-se utilizar tanto o termo "bola", quanto o termo "pelota".

24. Saussure, Ferdinand de. *Curso de Lingüística Geral*. Trad. de Antonio Chelini, José Paulo Paes e Izidoro Blikstein. 22a ed. São Paulo: Cultrix, 2000, p. 92
25. Saussure, Ferdinand de. *op.cit.* p. 142
26. Coelho Neto, J. Teixeira. *Semiótica, Informação e Comunicação*. São Paulo: Perspectiva, 1980, p. 27

5.3. O modelo de Louis Hjelmslev

Louis Hjelmslev (1899-1965), lingüista dinamarquês, foi fundador do Círculo Lingüístico de Copenhague.

Ele propôs uma teoria da linguagem capaz de descrever de maneira não contraditória todo e qualquer texto em qualquer língua, conhecida ou não, existente ou que venha a ser criada, já escrito ou que venha a ser escrito,[27] cuja vantagem em relação à concepção de Saussure está no fato de aplicar-se a qualquer linguagem. Por essa razão, seu modelo pode ser utilizado inclusive para uma análise do software sob o ponto de vista lingüístico.

Hjelmslev apontou que a lingüística utilizava o método indutivo para a descrição do fenômeno da linguagem. A hierarquia dos conceitos lingüísticos em seu tempo partia dos sons aos fonemas, dos fonemas às sílabas, das sílabas às palavras e das palavras às frases. Ele, ao contrário, adotou orientação diversa, utilizando o método dedutivo. Isso implicou considerar o texto como ponto de partida, decompondo-o em unidades lingüísticas menores. Considerando o texto como objeto a ser estudado, a análise do texto seria sua descrição através de suas dependências homogêneas, de seus elementos em relação ao texto, e das dependências entre os elementos, reciprocamente.[28]

Segundo este modelo, a linguagem consiste numa relação entre um processo e um sistema. O processo é um número limitado de elementos que constantemente reaparecem em novas combinações. Por exemplo, o texto é um processo, porque é formado por um número limitado de elementos – as palavras – que constantemente aparecem em novas combinações, cada qual produzindo um texto diferente. O sistema é a estrutura que permite analisar o processo lingüístico. A gramática é um exemplo de sistema, pois esta é que permite o entendimento do texto segundo suas regras.

A teoria da linguagem deveria ser capaz de sustentar a existência de um sistema subjacente ao processo, bem como sustentar a tese de uma constância em meio às variações. Neste caso, Hjelmslev baseou-se em Saussure, ao vir a constância na língua e a variação, na fala. Assim, "o processo só existe em virtude do sistema subjacente que o governa e que determina sua função possível. Não seria possível imaginar um processo sem um sistema por trás dele, porque, neste caso, tal processo seria inexplicável, no sentido absoluto da palavra. Um sistema, pelo contrário, não é inconcebível sem um processo"[29].

Isso pode ser demonstrado através do Paradoxo de Bolzano. De acordo com a maneira pela qual o sistema analisa o processo, obtém-se um resultado diferente. Nos exemplos abaixo, o processo consiste na soma e subtração infinitas do mesmo número. Ao variar a análise feita sobre essa soma, obtêm-se resultados diversos.

Seja o seguinte processo:

S = a - a + a - a + a - a + a - a + a - a + a - a + a - a + a - a + (...)

A primeira possibilidade de sistematização desse processo é a seguinte:

S = (a - a) + (a - a) + (a - a) + (a - a) + (a - a) + (a - a) + (a - a) + (a - a) +...

S = 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 +...

S = 0

A segunda possibilidade é agrupar todos os subtraendos:

S = a - a - a - a - a - a - a - a - a - a - a - a - a - a - a -...

S = a - (a - a) - (a - a) - (a - a) - (a - a) - (a - a) - (a - a) - (a - a) -...

S = a - 0 - 0 - 0 - 0 - 0 - 0 - 0 -...

S = a

A terceira possibilidade é a seguinte:

S = a - a + a - a + a - a + a - a + a - a + a - a + a - a + a - a +...

S = a - (a - a + a - a + a - a + a - a + a - a + a - a + a - a +...) a própria série está entre parêntesis

S = a - S

S = a / 2

27. Hjelmslev, Louis. *Prolegômenos a uma Teoria da Linguagem*. São Paulo: Perspectiva, 1975, p. 20

28. Hjelmslev, Louis. *op.cit.* p. 34

29. Hjelmslev, Louis. *op.cit.* p. 41

Uma outra dicotomia na teoria da linguagem de Hjelmslev é a existente entre a conjunção e a disjunção. É sobre esta distinção que se baseia a distinção entre paradigma e sintagma. No processo encontra-se disjunção. No sistema, há conjunção. A representação lógica da disjunção é feita pelo sinal de soma, e a representação da conjunção, pelo sinal de multiplicação.

Figura 12
Tabela de correlação entre operações lógicas e sua representação

Operação Lógica	Representação
Disjunção (ou A ou B)	A + B
Conjunção (ou A ou B)	A * B

Assim, o esquema de Saussure acima mencionado é representado da seguinte maneira, segundo a concepção de Hjelmslev: (A + O) * (bola + redonda + balão) * (adentrou + entrou + penetrou) * (na + no) * (rede + véu da noiva + gol)

5.4. Finalidade da Linguagem: economia relativa entre signos e não-signos

Outra característica da linguagem é que, embora sejam ilimitadas as possibilidades de criação de um texto, as grandezas que o compõe são limitadas. Não há uma infinidade de sílabas e os fonemas não passam de duas dezenas. Hjelmslev diz que se não houvesse inventários limitados, a teoria da linguagem não poderia alcançar seu objetivo de tornar possível uma descrição simples e exaustiva que está por trás do processo textual. Por exemplo, não se pode imaginar um idioma que tenha uma palavra unívoca. Além disso, em todas as línguas conhecidas, há um momento da análise da expressão em que as grandezas não mais veiculam significação e, portanto, não são expressões de signos. As sílabas e os fonemas não são expressões de signos, mas apenas partes de expressões de signos.[30]

Em suma, Hjelmslev explica que:

"a economia relativa entre os inventários de signos e de não-signos responde inteiramente àquilo que é provavelmente a finalidade da linguagem. Segundo sua finalidade, uma linguagem é, antes de qualquer coisa, um sistema de signos; a fim de preencher plenamente esta finalidade, ela deve ser sempre capaz de produzir novos signos, novas palavras e novas raízes. (...) a exigência de uma quantidade ilimitada de signos só é realizável se todos os signos forem formados com a ajuda de não-signos, cujo número é limitado e, mesmo, extremamente reduzido. Tais não-signos que entram como partes de signos num sistema de signos serão denominados figuras".[31]

As figuras distinguem-se umas das outras por oposição. Isto é, um fonema ou uma letra só existe porque outros fonemas ou outras letras existem.

5.5. Plano da Expressão e Plano do Conteúdo

A linguagem, além dos dois eixos em que se desenvolve, também possui dois planos: o plano da expressão e o plano do conteúdo. Umberto Eco explica que "em todos os processos sígnicos temos um elemento de expressão (chamamos-lhe, também, o significante) que veicula um elemento de conteúdo (o significado).[32]

Hjelmslev diz que:

"é impossível existir – a menos que sejam isolados artificialmente – um conteúdo sem expressão e uma expressão sem conteúdo. Se se pensa sem falar, o pensamento não é um conteúdo lingüístico (...). Se se fala sem pensar, produzindo séries de sons sem que aquele que os ouve possa atribuir-lhes um conteúdo, isso será um abracadabra e não uma expressão lingüística (...).[33]

Tanto o significante – expressão – quanto o significado – conteúdo – têm uma forma e uma substância. A forma é o conjunto dos aspectos dos fenômenos lingüísticos que podem ser descritos sem se recorrer a premissas de fora da linguagem – sistema. A substância é o conjunto dos aspectos dos fenômenos lingüísticos que não podem ser descritos sem se recorrer a premissas fora da linguagem. Isso pode ser representado pelo seguinte esquema:

Figura 13
Quadro indicativo da forma e substância do significante e significado de um signo

Significante (Expressão)	Forma
	Substância
Significante (Conteúdo)	Forma
	Substância

Porém, tanto o significante – expressão – quanto o significado – conteúdo – têm forma e substância.

A forma da expressão, constituída pelas regras paradigmáticas e sintagmáticas, é o modo pelo qual se expressa o signo. A substância da expressão é a matéria que suporta esta forma. Por exemplo, a substância fônica ou as letras.

A forma do conteúdo é o modo pelo qual se constrói o sentido. Isso é verificável ao se comparar uma mesma frase em várias línguas:[34]

30. Hjelmslev, Louis. *op.cit.* p. 51
31. Hjelmslev, Louis. op.cit. p. 52
32. Eco, Umberto. O Signo. Trad. de Maria de Fátima Marinho. 3ª ed. Lisboa: Editorial [illegible] p. 77
33. Hjelmslev, Louis. *op.cit.* p. 54.
34. Hjelmslev, Louis. *op.cit* p. 56.

Figura 14
Tabela indicativa da forma do conteúdo da frase "Eu não sei" em outras línguas

Eu não sei	Português
Jeg véd det ikke	Dinamarquês
I do not know	Inglês
Je ne sais pás	Francês
En tiedä	Finlandês
Naluvara	Esquimó

Embora ao ler cada frase se obtenha o mesmo sentido, cada uma das línguas possui um modo especial de criá-lo. Se for comparado com o português, o sentido da frase "Eu não sei" é obtido da seguinte maneira nos idiomas abaixo assinalados:[35]

Figura 15
Sentido da frase "Eu não sei" em outras línguas

Língua	Tradução ao Pé da Letra em Português
Dinamarquês	Eu sei aquilo não
Inglês	Eu (sem sentido) não sei
Francês	Eu (negação) não sei
Finlandês	Não-eu sei
Esquimó	Não sabendo sou eu isso

Por último, a substância do conteúdo é o sentido da frase, que vêm à mente quando esta é lida.

Em resumo, estes quatro conceitos podem ser colocados no seguinte quadro:

Figura 16
Quadro das relações entre os conceitos de forma e substância da expressão e do conteúdo

Texto (forma de expressão)	Gramática/Vocabulário (forma do conteúdo)
Papel/Fonema (substância da expressão)	Idéia (Significado) (substância do conteúdo)

35. Hjelmslev, Louis. *op.cit* p. 56.

6. O Software na Perspectiva da Lingüística

Após os esclarecimentos técnicos sobre o funcionamento dos computadores e do software, bem como os conceitos básicos de lingüística, segue agora a demonstração de que este último é um texto escrito segundo uma linguagem de programação. Dessa forma, conclui-se que sua natureza é a mesma que a das obras artísticas, literárias e musicais.

Na concepção de Saussure sobre a linguagem (= língua + fala), pode-se afirmar analogicamente que o software é equivalente à fala, enquanto a linguagem de programação é equivalente à língua.

Já na concepção de linguagem de Hjelmslev, em que a linguagem consiste num processo criado segundo a estrutura decorrente de um sistema, o software pode ser considerado um processo elaborado segundo as regras do sistema, que é a linguagem de programação.[36]

Encarada como sistema, a linguagem de programação estabelece as regras de criação do software. Ela constituirá a seqüência de signos a ser utilizada pelo programador, estabelecerá quais comandos (=palavras) desta linguagem serão utilizados e em que ordem os mesmos devem ser utilizados. Além disso, o software é um processo, porque só é possível compreendê-lo de uma forma geral, isto é, não é inteligível se for considerado apenas em diversas partes simples. Trata-se do que Hjelmslev disse, ao afirmar que o todo não é a soma das partes, mas as relações existentes entre seus elementos. Este processo é composto por um número limitado de comandos que se combinam de ilimitadas formas, para formarem novos textos.

Os comandos, por sua vez, são formados por figuras. E o fenômeno também se repete quando o software é compilado em seqüências de bits. Neste caso, cada bit é uma figura, e a união desses bits forma os comandos e dados, que serão processados pelo computador.

Segundo a lingüística, um texto só é inteligível, quando interpretado segundo a língua em que foi escrito. Isto ocorre quando uma pessoa lê um texto em um idioma desconhecido dela. Pode-se ler o quanto quiser, e nada fará sentido. O software, do mesmo modo, só é inteligível segundo uma linguagem de programação em que foi escrito. Um

36. Rigorosamente, dever-se-ia falar em língua de programação, e não, linguagem de programação.

programador que somente conheça a linguagem C, não consegue entender um software escrito em linguagem Pascal, e assim por diante.

O software também pode ser traduzido em outras linguagens e viceversa. Por exemplo, quando se escreve um software em português, esse texto recebe o nome de algoritmo. Quando o algoritmo é traduzido para uma linguagem de programação, daí se obtém o software.

Figura 17
Quadro de correspondência entre algoritmo e software

Texto escrito em linguagem humana (Português, Inglês, etc.)	⇔	Texto escrito em linguagem de programação (C, Pascal, etc.)
Algoritmo		Software

No exemplo de elaboração do programa que calcula as raízes reais de uma equação de segundo grau, pode-se notar que o mesmo foi escrito em língua portuguesa, ou seja, foi escrito o algoritmo para o cálculo das raízes reais de uma equação do segundo grau. Em seguida, há o mesmo texto (=processo) escrito em linguagem C. Os comandos utilizados devem ser aqueles próprios da linguagem utilizada no caso. Se fosse utilizada uma outra linguagem de programação, logo, os comandos desta outra linguagem devem ser usados.

Em um segundo plano, o software apresenta coerência tanto nas relações sintagmáticas – disjuntivas –, quanto nas relações paradigmáticas – conjuntivas –, que são inerentes à sua elaboração pelo programador.

No software elaborado para o cálculo das raízes reais de uma equação de segundo grau, as relações paradigmáticas existem à medida que se podem utilizar diversas formas, funções e variáveis, para representar o algoritmo. Em outras palavras, isso significa que o software pode ser escrito de forma simples ou complexa, conforme a intenção do programador. Já as relações sintagmáticas ocorrem à medida que não se pode inverter a ordem do texto, sob pena de não produzir sentido, isto é, de não funcionar como deveria. Por exemplo, não é possível calcular as raízes reais sem antes fazer o teste para saber se a equação tem ou não raízes reais, ou mesmo que se imprimam os resultados da equação de segundo grau, sem antes ter solicitado ao usuário que os forneça.

As linguagens de programação e os softwares também podem ser analisados segundo o plano de expressão e o plano de conteúdo.

Como visto acima, o plano da expressão corresponde, em semiótica, ao significante, e o plano do conteúdo se relaciona com o significado. Segundo Hjelmslev, tanto o significante quanto o significado têm forma e conteúdo.

Em relação ao plano da expressão, pode-se afirmar que o software corresponde à forma da expressão. Já o *corpus mechanicus*, que pode ser o meio magnético – quando em forma de bits – ou o papel que o suporta – texto do software escrito –, corresponde à substancia da expressão.

A forma do conteúdo corresponde aos comandos e suas relações paradigmáticas e sintagmáticas, determinadas pela linguagem de programação. A substância do conteúdo é a maneira pela qual a linguagem de programação produz os efeitos desejados pelo criador do software. Esquematicamente é o seguinte:

Figura 18
Quadro correspondente aos conceitos de Hjelmslev aplicados ao software

Software (forma de expressão)	Comandos Relações Paradigmáticas e Sintagmáticas (forma do conteúdo)
Meio Magnético/Papel (substância da expressão)	Resultado produzido pelos elementos da Linguagem de Programação (substância do conteúdo)

7. Conclusão

O software é, portanto, um texto escrito em uma linguagem de programação. Pelo fato de ser um texto que apresenta as mesmas características dos demais textos escritos, bem como as linguagens de programação terem a mesma estrutura das demais línguas, a sua proteção jurídica adequada é o direito de autor.

Por isso, é possível assegurar ao criador do software não somente os direitos patrimoniais, mas também determinados direitos morais, como o direito à paternidade da obra e direito à integridade.

Logo, o direito de autor é o sistema mais adequado para a proteção do software, porque se destina, em última análise, à proteção de obras lingüísticas, isto é, criações intelectuais que comunicam uma informação. Não porque este ramo protege melhor que os demais sistemas de proteção, mas porque são obras do mesmo gênero e, assim, devem ser protegidas pelo mesmo sistema.

8. Referências Bibliográficas

Alcalde, E.; Garcia, M.; Penuelas, S. *Informática Básica*. São Paulo: Makron Books, 1991.

Ascensão, José de Oliveira. "Software e Direito Autoral". in Gomes, Orlando (org). *A proteção jurídica do software*. Forense. Rio de Janeiro, 1985.

Baptista, Luiz Olavo. "A Proteção dos Softwares em Direito Comparado e Internacional". in *Revista de Direito Mercantil*, São Paulo, vol. 22, nº 50, p. 26-41, abr/jun 1983.

_____. "Proteção Jurídica do Software - Novos Desenvolvimentos", in *Revista Forense*. Rio de Janeiro. v. 84. n. 301. p. 47-52. jan./mar. 1988.

Coelho Neto, J. Teixeira. *Semiótica, Informação e Comunicação*. São Paulo: Perspectiva, 1980.

Eco, Umberto. *O Signo*. trad. de Maria de Fátima Marinho. 3ª ed. Lisboa. Editorial Presença.

Hammes, Bruno Jorge. "O Programa de Computador segundo a Lei nº 7.646, de 18.12.1987". in *Estudos Jurídicos*. São Leopoldo, v. 21, nº 52, p. 17-26, mai/ago 1988.

Gomes, Orlando. "A proteção dos softwares". in Gomes, Orlando (org). *A proteção jurídica do software*. Rio de Janeiro. Forense, 1985.

Guimarães, Ângelo de Moura; Lages, Newton Alberto de Castilho. *Algoritmos e Estruturas de Dados*. Rio de Janeiro. LTC, 1993.

Hjelmslev, Louis. *Prolegômenos a uma Teoria da Linguagem*. São Paulo. Perspectiva, 1975.

Kindermann, Manfred. "O direito do autor internacional e a proteção do software. Histórico, situação atual e fatos novos", tradução de Bruno Jorge Hammes. in *Estudos Jurídicos*, São Leopoldo, v. 22. n. 54. p. 65-126. jan./abr. 1989.

Neto Lobo, Paulo Luiz. "Direito de Autor de Software", in *Revista Forense*, Rio de Janeiro, v. 85. n. 305. p. 85-93. jan./mar. 1989.

Santos, Manoel Joaquim Pereira dos. "A Proteção Adequada ao 'Software'". in *Revista de Direito Civil, Imobiliário, Agrário e Empresarial*. São Paulo, v. 11. n. 40. p. 131-43. abr./jun. 1987.

Saussure, Ferdinand de. *Curso de Lingüística Geral*. trad. de Antonio Chelini, José Paulo Paes e Izidoro Blikstein. 22ª ed. São Paulo. Cultrix, 2000.

Wirth, Niklaus. *Algoritmos e Estruturas de Dados*. Trad. de Cheng Mei Lee, revisão de João José Neto. Rio de Janeiro. Ed. Prentice-Hall, 1989.