Estudo da usabilidade para ferramenta de busca de conteúdo matemático

Nobre, Renata V
Bacharelado em Ciência da Computação
Universidade Federal de Alfenas – UNIFAL-MG
Alfenas – MG, Brasil
a08031@bcc.unifal-mg.edu.br

Gonzaga, Flavio B.
Bacharelado em Ciência da Computação
Universidade Federal de Alfenas – UNIFAL-MG
Alfenas – MG, Brasil
fbgonzaga@unifal-mg.edu.br

***Resumo*—Com o constante crescimento da Internet, o usuário que antes pouco contribuía com conteúdos, passou a produzí-los ativamente, seja através de redes sociais, bibliotecas online colaborativas, blogs, dentre outros. Com base nesse crescimento do volume de informações disponível, recuperar páginas de interesse de cada usuário em um determinado contexto torna-se um desafio crescente e tema recorrente de pesquisa. Dentre os tipos de busca que começam a ser explorados, pode-se citar as ferramentas de busca por fórmulas matemáticas. A interface para o usuário desse tipo de ferramenta é também um desafio, dada a variedade de símbolos possíveis, e as diversas linguagens existentes para representação de conteúdo matemático. O trabalho apresenta uma análise de uma interface produzida por Cordeiro, que é composta só de botões. Com base na mesma, é verificada na biblioteca DLMF (*Digital Library of Mathematical Functions*) se existem ainda símbolos matemáticos que não estejam presentes na interface de Cordeiro. Dessa forma, obteve-se um conjunto completo de símbolos matemáticos que devem estar presentes em uma interface para esse tipo de ferramenta. Outra contribuição é o estudo da quantidade de cliques que é dispendida para a montagem das fórmulas, também com base na mesma interface.**

***Abstract*— Due the constant growth of the Internet, the user that was a small contributor of content before, started to produce them actively, whether through social networks, collaborative online libraries, blogs, among others. Based on this growing volume of information available, recover pages of interest to each user in a given context becomes a big challenge and so a recurring theme of research. Among the types of search engines that starts to be developed, we can mention the search by mathematical formulas. The user interface for this type of tool is also a challenge, given the range of possible symbols, and the various languages for representing mathematical content. This paper presents an analysis of an interface produced by Cordeiro, which is composed only of buttons. Based on the same, is verified in the DLMF (Digital Library of Mathematical Functions) if there are mathematical symbols that are not present in the Cordeiro's interface. Thus, there was obtained a complete list of mathematical symbols that should be present in an interface for this type of tool. Another contribution is the study of the number of clicks that is spent for mounting the formulas, also based on the same interface.**

Palavras-chave— Interface com usuário; Busca matemática; Usabilidade; SearchOnMath

I. INTRODUÇÃO

A cada dia, novas páginas são publicadas na Internet aumentando a quantidade de informação disponível na rede. Estima-se que a cada minuto são registrados 70 novos domínios em todo o mundo [1]. Considerando-se apenas a quantidade de páginas indexadas pelo *Google*[1] na atualidade, a quantidade é próxima de 40 bilhões [2]. Para facilitar o acesso a esta enorme quantidade de informação, as ferramentas de busca são de extrema importância. É inimaginável, no cotidiano de um usuário da Internet hoje, a ação de encontrar uma página de um conteúdo especifico sem uma ferramenta eficiente dessa natureza.

Com a eficiência já alcançada pelas ferramentas de busca tradicionais, baseadas em texto, começam a surgir, nos últimos anos, buscas mais inteligentes e de conteúdo específico. Como por exemplo, pode-se citar a busca de artigos científicos do *Scholar*[2] do *Google* e os relatórios oferecidos pelo *Wolfram Alpha*[3], dentre outras. Ainda nesse aspecto de recuperar informação específica, outra possibilidade são as bibliotecas digitais como a *Wikipedia*[4], *MathWorld*[5] e a DLMF (*Digital Library of Mathematical Functions*)[6].

Apesar da existência de bibliotecas digitais de conteúdo específico, a busca por fórmulas matemáticas ainda é um desafio, de modo que poucas ferramentas nesse contexto foram desenvolvidas. São vários os aspectos que tornam difícil o desenvolvimento desse tipo de ferramenta. A interpretação de fórmulas diferentes do ponto de vista sintático, mas equivalentes do ponto de vista matemático, como 1/x e x^(-1); a diversidade de linguagens e a falta de um padrão efetivo na representação desse tipo de conteúdo; e uma interface que forneça facilidade e eficácia, não só com pesquisas de textos, mas também para fórmulas e construções matemáticas [3] são

[1] http://google.com
[2] http://scholar.google.com
[3] http://wolframalpha.com
[4] http://wikipedia.org
[5] http://mathworld.wolfram.com
[6] http://dlmf.nist.gov

alguns exemplos. O presente trabalho ataca então a questão da usabilidade da interface para esse tipo de ferramenta.

Em 2010, Cordeiro [4] fez um estudo das ferramentas de busca matemática existentes e propôs uma interface nova para ser usada no trabalho desenvolvido por Gonzaga [5]. Nesta interface, a fórmula é construída através de botões organizados em seções. A representação correspondente da fórmula em MathML[7] (uma linguagem matemática descrita na Seção 2), é então obtida e enviada para o sistema de busca. Nessa época, as duas principais referências desse tipo de ferramenta de busca eram a *Math Web Search* [6] e a *Whelp* [7], ferramentas que não estão mais disponíveis. Atualmente, existe a *Symbolab*[8], lançada em outubro de 2012. Sobre essa última, até a presente data não encontramos na literatura referências à respeito.

A proposta deste trabalho é avaliar a interface apresentada por Cordeiro, onde através da quantidade de cliques necessária para representar as fórmulas, pode-se verificar se a interface proposta é eficaz e eficiente. Eficaz no sentido de constatar se existem símbolos que ainda não estão presentes na interface, não sendo assim possível a representação de fórmulas que usem tais símbolos; e eficiente ao se medir a quantidade de cliques necessária para se produzir as diversas fórmulas. Os resultados desse estudo serão depois usados na interface da ferramenta de busca *SearchOnMath*[9], atualmente em desenvolvimento no LaReS (Laboratório de Redes de Computadores e Sistemas Distribuídos)[10]. O estudo da eficácia da interface em representar as fórmulas foi com base nas expressões extraídas da biblioteca DLMF. A escolha pela biblioteca em questão será melhor explicada na Seção 2. Os objetivos do trabalho são:

O artigo segue organizado da seguinte forma: Na Seção II, é feita uma fundamentação teórica sobre bibliotecas matemáticas, ferramentas de buscas e suas interfaces, e sobre usabilidade. Na Seção III, é apresentada a metodologia utilizada para a realização deste trabalho, seguida pela Seção IV que apresenta os resultados encontrados. Na Seção V, são dadas as conlusões do trabalho e algumas ideias de trabalhos futuros são apresentadas.

II. FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta, na subseção A, a contextualização das bibliotecas matemáticas digitais, falando mais especificamente da DLMF. Na subseção B, são mostradas algumas ferramentas de busca matemática com suas interfaces. As linguagens usadas para representação matemática são mostradas na subseção C, com mais destaque para a MathML, que é a linguagem adotada na interface proposta em Cordeiro. Por fim, a subseção D apresenta o conceito de usabilidade e o método de análise que foi utilizado neste trabalho.

A. Bibliotecas matemáticas digitais

Existem diversas bibliotecas de conteúdo matemático online. No momento as suportadas pela ferramenta *SearchOnMath* são: *Wikipedia* (parte matemática)[11], *MathWorld* e a DLMF (já referenciadas na Seção 1). Tanto a *Wikipedia* quanto o *MathWorld* possuem representação das fórmulas em formato de imagem (GIF e PNG respectivamente) e textual (embutido no código HTML - *Hypertext Markup Language*), sendo que na *Wikipedia* a linguagem textual adotada é a TeX [8], enquanto que no *MathWorld* a representação é feita em linguagem própria. A DLMF de forma semelhante também possui representação gráfica e textual. Porém, a representação textual está disponível nas linguagens TeX e MathML. A DLMF foi escolhida para ter suas fórmulas analisadas no trabalho por ser a única com representação em MathML das fórmulas, que é a mesma linguagem utilizada na interface produzida por Cordeiro.

- DLMF

A DLMF é um projeto do NIST (*National Institute of Standards and Technology*) [12] onde foi realizada a revisão do *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* [9] e publicada uma versão web do mesmo em 2010.

Este livro é uma publicação famosa por organizar em um só local, as informações mais importantes e necessárias para usar as funções matemáticas em aplicações práticas. Ele padronizou e normalizou as funções especiais da matemática aplicada, facilitando a comunicação de resultados cientificos [10]. É o trabalho mais citado em publicações matemáticas. Este número vem crescendo ano-a-ano, desde a publicação em 1964, com estimadas 40.000 citações até 2011 [11].

A versão web utilizada no trabalho foi obtida em janeiro de 2013 e possui 908 páginas de conteúdo com um total de 9.834 fórmulas.

B. Ferramentas de Busca e Suas Interfaces

As ferramentas de busca que serão apresentadas nesta seção são a *Whelp*, a *Math Web Search* e a *Symbolab*. Logo em seguida, será apresentada a interface proposta por Cordeiro a qual será analisada neste trabalho.

- Whelp

A *Whelp* é um protótipo de uma ferramenta desenvolvida para buscar expressões na biblioteca de conteúdo matemático do *Coq*[13], que é um gerenciador formal de provas. O *Coq* possui uma linguagem formal própria, semelhante ao TeX. Esta linguaguem permite a escrita de definições matemáticas, algoritmos executáveis e teoremas.

Ela não possui uma interface gráfica para construção de expressões, o que dificulta o uso, visto que o usuário pode não conhecer a linguagem utilizada.

Essa ferramenta não se encontra mais disponível online, impossibilitando um estudo mais detalhado da sua eficiência e eficácia. A tela inicial que era usada pela *Whelp* é a apresentada na *Fig. 1*.

[7] http://www.w3.org/Math
[8] http://symbolab.com
[9] http://searchonmath.com
[10] http://www.bcc.unifal-mg.edu.br/lares
[11] http://en.wikipedia.org/wiki/Portal:Mathematics
[12] http://www.nist.gov
[13] http://coq.inria.fr

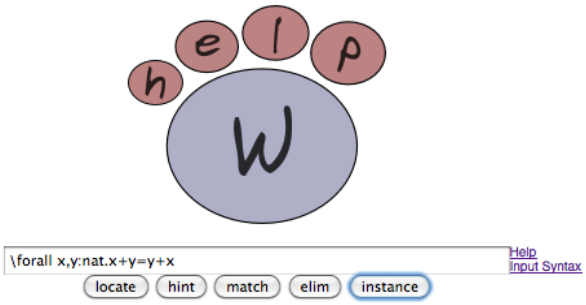

Figura 1: Tela inicial da Whelp (acesso em 03/2010).

- Math Web Search

A *Math Web Search* é uma ferramenta de busca de conteúdo matemático que tem como base de dados o repositório *CONNEXIONS Project*[14]. Este é um projeto que consiste em um ambiente colaborativo na Internet e disponibiliza diversos tipos de conteúdo como cursos e livros. Possui um índice contendo 87.000 fórmulas em formato MathML, obtidas na página *functions* do *Wolfram*[15].

A sua interface possui um menu com botões no qual o usuário escolhe as estruturas para montar a sua expressão. Percebeu-se que não existia uma boa organização dos símbolos, sendo possível encontrar o mesmo em várias seções diferentes do menu. Após montada a expressão, o usuário poderia fazer modificações de forma textual. Em seguida, o usuário seleciona a linguagem de entrada e então executa a busca.

Assim como a *Whelp*, a *Math Web Search* não se encontra mais disponível online. A tela inicial que era utilizada pode ser visualizada na *Fig. 2*.

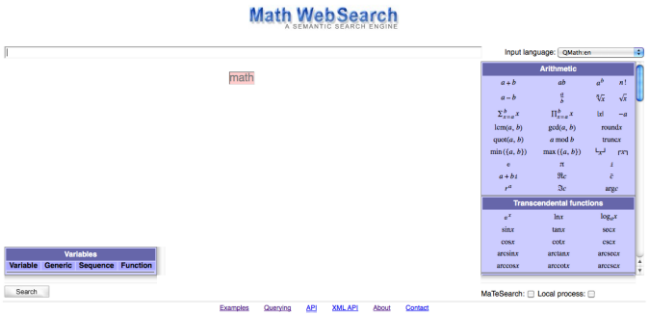

Figura 2: Tela inicial da *Math Web Search* (acesso em 03/2010).

- Symbolab

Conforme mencionado anteriormente, a única ferramenta funcional encontrada no momento da escrita do presente trabalho é a *Symbolab* que foi lançada em outubro de 2012.

Sua interface é bastante clara e intuitiva, como pode ser notada na *Fig. 3*. Os símbolos são separados nas seguintes abas: letras gregas mínusculas, letras gregas maiúsculas, funções trigonométricas, operadores, acentos, grandes operadores, derivadas, além das abas de física e química, esta última ainda em construção. Nota-se que alguns símbolos ficam 'perdidos' em abas não tão relacionadas com seu uso. Por exemplo, as funções exponencial e logarítmica estão na aba destinada a funções trigonométricas. Um destaque dessa interface é a primeira aba, chamada de básica, que possui símbolos com grande utilização nas fórmulas. Além disto, a barra de busca possui a característica de auto-completar. Ou seja, se o usuário insere o símbolo Δ a ferramenta já oferece várias opções de busca como, por exemplo, Δ ABC e Δ = b² - 4ac.

Figura 3: Tela inicial da *Symbolab* (acesso em 08/2013).

Apesar de ser bastante intuitiva, o fato de não ter nenhum trabalho cientifico sobre esta ferramenta ainda deixa em aberto algumas questões relacionadas à interface. Por exemplo, não se sabe qual a sua abrangência ao se considerar a variedade de símbolos usados na matemática (eficácia). Outra pergunta é se a aba básica foi construída apenas baseada na intuição dos autores no sentido de definir quais símbolos devem estar presentes, ou se de fato foi feito um estudo para descobrir quais são os elementos mais comuns em fórmulas dado um determinado domínio por exemplo.

- Interface proposta por Cordeiro

Em 2010, Cordeiro propôs uma interface para ferramenta de busca de conteúdo matemático. Ele usou a linguagem de marcação MathML para renderizar os símbolos e montar a fórmula que deve ser buscada pela ferramenta.

Nessa interface, como pode-se verificar na *Fig. 4*, os símbolos estão separados em painéis de acordo com as áreas de aplicação e agrupados nas seguintes abas: Símbolos, Funções e Letras Gregas. Há também um botão para a inserção de variáveis e outro para a inserção de números.

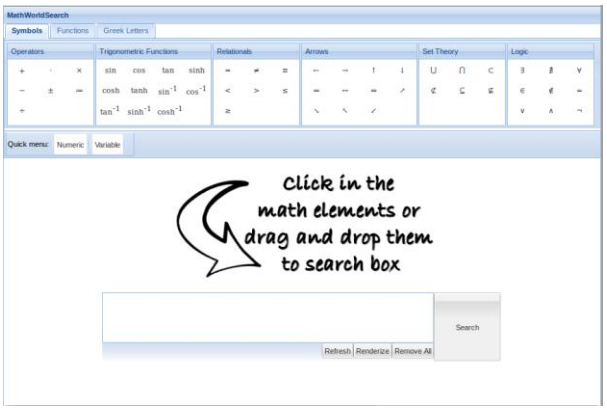

Figura 4: Interface proposta por Cordeiro

[14] http://cnx.org
[15] http://functions.wolfram.com

Um aspecto importante a ser ressaltado nessa interface é que cada símbolo clicado é acrescentado no campo da busca na forma de um botão. Isso dá uma possibilidade interessante que é o efeito de poder 'arrastar' e trocar de ordem os símbolos inseridos, como pode ser visualizado na *Fig. 5*. No entanto, um aspecto negativo é que não é possível digitar diretamente no campo de busca. Assim, um símbolo que não esteja presente nos botões não poderia ser digitado no campo (impossibilitando a busca), mesmo que o usuário soubesse a sintaxe correta do mesmo em MathML. Contudo, ficará claro na próxima seção que a sintaxe de MathML não é intuitiva e enxuta (aspectos que de certa forma podem ser considerados como presentes em TeX). Permitir que o usuário inserisse código MathML poderia ser uma alternativa também desanimadora.

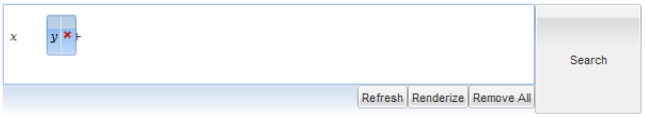

Figura 5: Interface proposta por Cordeiro

C. *Linguagens Matemáticas*

Com o enorme crescimento do conteúdo da Web, tornou-se necessária a criação de padrões para a comunicação de dados. Com isso, foram criadas linguagens de marcação para serem utilizadas com a internet. Entre elas: a XML[16], sigla *de eXtensible Markup Language*, que tornou possível a estruturação de documentos que fornecem um significado semântico [12]; e a MathML (melhor explicada na subseção seguinte) que tem como meta disponibilizar matemática para ser processada através da *World Wide Web*, da mesma forma que a HTML tem disponibilizado esta funcionalidade para texto. A linguagem TeX por sua vez foi criada fora desse escopo da Web, mas em função da sua popularidade entre pesquisadores na produção de documentos (especialmente na área das Ciências Exatas), aparece a cada dia mais em diversos sites. Em função dessa crescente, surgem a cada dia pacotes cujo objetivo é renderizar código TeX em navegadores. Dois exemplos bastante usados na atualidade são o *MathJax*[17] e o *MathQuill*[18]. No entanto, como o trabalho é focado na linguagem MathML, será explicado mais sobre a mesma a seguir.

- MathML

MathML é uma linguagem de marcação, baseada na linguagem XML, que descreve notações matemáticas e captura suas estruturas e conteúdos. Sua especificação foi liberada em 1999, e em 2010, a W3C lançou sua terceira versão, o MathML 3.0 [13].

Ela possui dois conjuntos de elementos de marcação, sendo um para apresentação e outro para conteúdo. As *Figs. 6* e *7* são a representação da fórmula b^2 -4ac em modo apresentação e conteúdo respectivamente. No conjunto de aprentação temos 28 elementos para codificar a notação matemática, com aproximadamente 50 atributos. Para conteúdo, existem aproximadamente 75 elementos com cerca de 12 atributos. A codificação da apresentação se preocupa com a maneira como as expressões serão exibidas e impressas, enquanto a codificação de conteúdo se preocupa com questões semânticas.

```
<mrow>
        <msup>
                <mi>b</mi>
                <mn>2</mn>
        </msup>
        <mo>-</mo>
        <mrow>
                <mn>4</mn>
                <mo>InvisibleTimes</mo>
                <mi>a</mi>
                <mo>InvisibleTimes</mo>
                <mi>c</mi>
        </mrow>
</mrow>
```

Figura 6: Código MathML com elementos de apresentação

```
<apply>
        <minus/>
        <apply>
                <power/>
                <ci>b</ci>
                <cn>2</cn>
        </apply>
        <apply>
                <times/>
                <cn>4</cn>
                <ci>a</ci>
                <ci>c</ci>
        </apply>
</apply>
```

Figura 7: Código MathML com elementos de conteúdo

D. *Usabilidade*

De acordo com o ISO 9241-11 [14], usablidade é a medida pela qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos, com eficácia, eficiência e satisfação em um contexto de uso específico.

Existem vários métodos de avaliação de usabilidade. Dentre estes métodos, temos os que usam a modelagem cognitiva, envolvendo a criação de um modelo computacional para estimar quanto tempo leva para as pessoas executarem uma determinada tarefa. Alguns exemplos de modelos cognitivos são: *Parallel Design* [15], onde em um grupo, cada pessoa cria um projeto inicial, e depois de várias reuniões de compartilhamento chegam a uma versão final; *Human Processor Model* [16], que quebra cada tarefa para analisar seus aspectos, sendo necessário entender como o cérebro humano processa informações separadamente; e o *GOMS(Goals, Operators, Methods and Selection Rules)* [17], que abrange uma família de técnicas que analisam a complexidade do uso de sistemas interativos.

Na família do GOMS temos, o CNM-GOMS que foi a primeira proposta de Card, Moran and Newell, e as variações

[16] http://www.w3.org/xml
[17] http://www.mathjax.org
[18] http://mathquill.com

do método que vieram depois como o KLM (*Keystroke Level Modeling)*, que faz várias hipóteses simplificadoras, sendo apenas uma versão restrita do GOMS; o NGOMSL (*Natural GOMS Language*) [18] que proporciona uma linguagem muito rigorosa, mas natural para a construção de modelos GOMS; e o CPM-GOMS que baseia-se no modelo de processador humano, permitindo a modelação do processamento de informação paralelo pelo usuário, sendo a técnica mais difícil de se implementar.

Neste trabalho, por ser a primeira análise da interface, optou-se por utilizar a técnica do CMN-GOMS para medir a eficácia e a eficiência, sendo assim um ponto inicial que pode ser estendido depois para trabalhos futuros no LaReS.

- CMN-GOMS

Este método reduz a interação do usuário com um computador para suas ações elementares. Nele definem-se quatro ações:

a) Objetivo: o que o usuário deseja;

b) Operadores: ações que devem ser executadas para alcançar o objetivo;

c) Método: sequência em que estas ações devem ser executadas;

d) Regras de Seleção: caso seja possível alcançar um objetivo através de mais de um método, as regras de seleção definem quando um usuário vai usar um ou outro método.

Após definir as ações, deve-se realizar uma investigação cautelosa de quanto tempo é necessário para que o usuário conclua cada ação detalhada. Através de um cálculo simples, somando os tempos das ações de um método, consegue-se o tempo necessário para que o usuário atinja o objetvo usando a interface.

As vantagens deste método são cálculos com pouco esforço e em um curto espaço de tempo. As desvantagens são a imprevisibilidade do usuário, como por exemplo, a fadiga, a suposição que usuário sempre vai saber o que fazer em qualquer ponto, e as personalidades dos usuários, que podem ter alguma restrição fisíca [19].

Ao invés de medir o tempo que seria gasto montando fórmulas, usou-se como medida a quantidade de cliques. A vantagem é que de certa forma também é uma medida relacionada a tempo, uma vez que uma fórmula que precise de mais cliques provavelmente gastaria mais tempo para ser feita (considerando um mesmo usuário), e possibilita o desenvolvimento de um algoritmo automatizado para essa finalidade, dispensando assim a necessidade de usuários reais, e atingindo o objetivo que é descobrir símbolos não contemplados. Dessa forma, anulamos a imprevisibilidade do usuário, a suposição do conhecimento, e as diferenças de personalidade.

III. METODOLOGIA E DESENVOLVIMENTO

Iniciamos o projeto com uma análise comparativa da forma como os símbolos matemáticos são representados pela DLMF e pela interface proposta por Cordeiro. Nesta análise, constatamos diferentes representações para um mesmo símbolo na própria DLMF, mesmo todos usando o conjunto de aprentação da MathML para as tags. Por exemplo, encontramos o símbolo ‘a’ com mais de 50 representações diferentes, desde o simples <mi>a</mi> até um <mi mathvariant = "bold-italic" href = "DLMF:/21.1#p2">a</mi>. Com isto, percebemos que a biblioteca utiliza diversos atributos nas tags para acrescentar funcionalidades como estilo e referência de capítulos. Entretanto, os atributos de capítulo nem sempre se referem ao capítulo onde a fórmula está sendo utilizada. Percebemos também diferenças de codificação para um mesmo símbolo. Por exemplo, para o símbolo *, encontramos <mo>*</mo>, <mo>\u002a</mo> e <mo>\u2217</mo>. Notamos que a forma como Cordeiro representou alguns símbolos na interface também diferia da forma como a DLMF representava. Além disto, encontramos símbolos na biblioteca que ainda não estavam disponíveis na interface. Desta forma, foi necessário realizar uma padronização na forma de representar cada símbolo.

Com a padronização da representação dos símbolos, tanto na biblioteca, como na interface, foi possível realizarmos a contagem de quantas fórmulas da DLMF poderiam ser representadas na interface proposta, bem como, quantos cliques eram necessários para representá-las.

De uma forma geral, o processo do desenvolvimento deste trabalho pode ser representado pelas etapas demonstradas na *Fig 8*. Após uma padronização prévia dos símbolos da interface

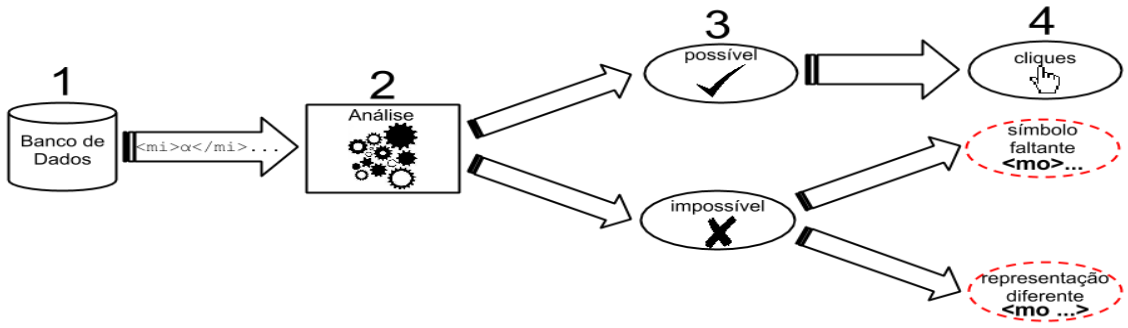

*Figura 8:*Etapas do Desenvolvimento

e de alguns símbolos da biblioteca, na etapa 1, selecionamos uma fórmula da base de dados da DLMF. Na etapa 2, o sistema analisa esta fórmula e responde se ela pode ou não ser representada na interface. Se for possível (etapa 3), o sistema realiza a contagem do número de cliques necessários para representá-la (etapa 4). Se o sistema detectou que a fórmula não pode ser representada (etapa 3), analisamos visualmente o símbolo que impossibilita a representação (etapa 4). Neste ponto, encontramos duas possibilidades, ou o símbolo estava sendo representado de uma maneira diferente, ou realmente este era um símbolo que não existia na interface. Se o símbolo somente estivesse com uma representação diferente, adicionavámos uma nova regra ao código do sistema para sempre que encontrar aquele símbolo, o padronizasse para a representação escolhida. Repetimos este processo até que todos os símbolos estivessem padronizados e a detecção de fórmulas impossíveis fosse somente devida a símbolos ausentes na interface.

Com a padronização da representação dos símbolos, tanto na biblioteca como na interface, foi possível realizarmos a contagem de quantas fórmulas da DLMF poderiam ser representadas na interface proposta, bem como, quantos cliques eram necessários para representá-las.

Nas subseções seguintes apresentaremos, na subseção A, os processos desenvolvidos para a padronização do banco de dados e para encontrar os símbolos ausentes. Na subseção B, explicaremos como realizamos a contagem dos cliques na interface proposta por Cordeiro e em uma proposta por nós.

A. Padroniza Banco e Encontra Símbolos Ausentes

Após uma análise visual das diferentes maneiras como a DLMF e a interface proposta representam os símbolos nas suas fórmulas, criamos o nosso padrão para cada um dos símbolos. Este padrão foi criado pensando em representar cada símbolo da maneira mais simples e clara, com o propósito de facilitar a comparação dos símbolos da fórmula da DLMF com os símbolos existentes na interface proposta. Esta padronização também possibilitará, no futuro, fazer esta análise com outras bibliotecas matemáticas.

Criamos então uma base de dados composta por duas tabelas: uma para armazenar todas as fórmulas extraidas da DLMF e outra para armazenar os símbolos existentes na interface proposta.

Após estes passos, desenvolvemos a primeira parte do sistema, a qual realiza a padronização das fórmulas da DLMF segundo o padrão definido por nós para cada símbolo matemático.

Para facilitar o entendimento da padronização desenvolvida nas duas seguintes subsubseções, considere a *Eq. 1*, e o seu respectivo código em MathML extraído da DLMF e exibido na *Fig. 9*.

$$x = 0 \tag{1}$$

```
1 <m:math xmlns:m="http://www.w3.org/1998/Math/MathML" alttext="x=0."
  display="block">
2         <m:mrow>
3                 <m:mi href="DLMF:/36.1#p2.t1.r4">x</m:mi>
4                 <m:mo>=</m:mo>
5                 <m:mn>0.</m:mn>
6         </m:mrow>
7 </m:math>
```

Figura 9: Representação da *Eq. 1* pela DLMF

- Padronização das tags

Tomando como base então a *Fig. 9*, na padronização os passos consistem em remover a tag inicial que apresenta a representação da fórmula em TeX (linha 1) e os atributos de estilo e referências das tags (href da tag na linha 3). Removemos também o prefixo 'm:' que está presente em todo nome de tag. Padronizamos o sufixo de 'tipo' da tag para 'm' (linhas 3, 4 e 5). Este sufixo poderia ser 'mi' ou 'mo', porém encontramos casos em que um mesmo símbolo era representado em tags com tipos diferentes. Por exemplo, encontrou-se <mo>∆</mo> e <mimathvariant=\"normal\">∆</mi>. Removemos também todos os fechamentos de tag e algumas tags que são indiferentes na contagem dos cliques, como por exemplo <mrow> (linha 6). Essa é uma tag típica do modo de apresentação do MathML, cujo objetivo é informar ao navegador que os dados contidos dentro da mesma devem ser renderizados em uma mesma linha.

- Padronização dos símbolos e identificação de símbolos ausentes

Na segunda etapa da padronização, o sistema analisa cada um dos símbolos que encontra na fórmula e compara com os símbolos existentes na interface proposta. O sistema então faz uma cópia da fórmula e retira todos os símbolos que existem na interface. Assim, restam somente os símbolos que o sistema não encontrou referência. Se o símbolo realmente não existir na interface, concluimos que esta é uma fórmula que a interface proposta não conseguiria reproduzir. Entretanto, se o símbolo encontrado for na verdade um símbolo disponivel na interface, porém com uma forma de representação diferente, adicionamos esta variação de representação na padronização. Assim, o sistema agora deve padronizar este símbolo da maneira que decidimos inicialmente.

Todas as variáveis foram padronizadas para indicar que ali existia uma variável, independente de qual letra ou código a representasse (linha 3). Como encontramos variáveis em tags de texto, também padronizamos que todo texto seria uma variável, visto que para renderizar qualquer texto na interface, o mesmo deveria ser digitado em uma variável. O mesmo processo foi desenvolvido com os número (linha 5). Todos os símbolos, exceto parênteses, colchetes, chaves e variaveis, foram padronizados para suas representações unicodes (linha 4). Padronizamos também as tags de espaços em branco. Apenas como exemplo, encontramos 6 tipos diferentes de representação do espaço em branco. Caso não fosse feita a padronização, sempre que um espaço presente na fórmula fosse de unicode diferente do usado na interface, o resultado seria uma interpretação errada de que é um símbolo que a interface não oferece suporte.

Nesta etapa, também padronizamos as representações dos simbolos das seguintes funções: logaritmo, logaritmo neperiano, limite, máximo, mínimo, traço, exponencial,

diferencial, integral, fatorial, somatório, produtório e as funções trigonométricas. Por exemplo, a representação de um log que antes era <msub><m>log</m> passou a ser <mlog>. Padronizamos a representação do símbolo de apóstrofo, do sinal de menos e de colchetes duplos. As tags que indicavam matriz ou lista também foram padronizadas. Alguns símbolos HTML (comumente representados por mais de um caractere), como o >, representado por >, precisaram ser tratados de forma separada na padronização.

Outra classe de símbolos que recebeu tratamento especial são os quatro elementos não renderizáveis da linguagem MathML: (i) ApplyFuction, (ii) InvisibleTimes, (iii) InvisibleComma e (iv) InvisiblePlus. Por exemplo, o InvisibleTimes é usado entre variáveis representando justamente a multiplicação entre as mesmas, mas ao se abrir em um editor de textos, essa tag não aparece renderizada. Para melhor ilustrar, ao visualizarmos a fórmula ab+c não conseguimos identificar se o ab é uma única variável, ou se são duas variáveis em uma operação de multiplicação. Quando analisamos o código da fórmula se existir a multiplicação, vamos nos deparar com o InvisibleTimes entre a representação das duas letras.

Todo este processo foi desenvolvido de uma maneira interativa, incremental e suportado por testes. Ou seja, a cada símbolo apontado como 'ausente', analisávamos visualmente. Caso fosse apenas uma representação diferente, criávamos um teste para garantir que a padronização atenderia a este caso, e depois criávamos o método que executava a padronização em si. Então, executávamos o sistema novamente e avaliavámos o próximo símbolo apontado como 'ausente'.

Repetimos o processo até termos certeza de que todos os símbolos ausentes realmente não estavam disponíveis na interface proposta.

Depois destas padronizações, o código resultante da *Eq. 1* ficou de acordo com o exibido na *Fig. 10*.

```
1 <mvariable>
2 <m>\\u003d</m>
3 <mnumeric>
```

Figura 10: Representação da *Eq. 1* após a Padronização

B. *Quantidades de cliques por fórmula*

Para obter uma métrica da usabilidade da interface proposta, usamos o método do CPM-GOMS definindo as ações da seguinte maneira:

a) Objetivo: Representar uma determinada fórmula na interface proposta.

b) Operadores: Cliques nos botões da interface para representar a fórmula.

c) Método: Sequência em que os botões devem ser selecionados respeitando a ordem que aparecem na representação em MathML da equação.

d) Regras de Seleção: Sem regras de seleção.

Para representar uma fórmula na interface proposta, se o usuário se orientar somente pela representação visual da fórmula, várias ordens diferentes de cliques poderiam ser executadas de modo a se obter a mesma fórmula. No entanto, ao se orientar pelo código, fixamos como ordem dos cliques de fato a ordem em que os elementos aparecem no MathML. Dessa forma, define-se somente um método como possível, não sendo aplicável assim o conceito de regras de seleção.

Por exemplo, para a *Eq. 2*, temos as seguintes maneiras de representá-la na interface, apresentadas na *Tab. 1*. Na Maneira 1, observa-se toda a estrutura da fórmula, e clica-se primeiro no botão que já produz a forma X(y), e depois no botão de variavél, 2 vezes, para escrever a função 'Hi' e a variável 'z'. Na Maneira 2, observa-se e monta-se a fórmula à medida em que os símbolos são 'descobertos'. Primeiro, clica-se no botão de váriável para escrever a função, depois no botão que produz a forma (x) e por último, no botão de variável, novamente, para escrever a variável 'z'. Se observarmos como o código MathML, da *Eq. 2*, apresentado na *Fig. 11*, percebemos que existe um tag com o ApplyFunction, que identifica que na fórmula existe uma função. Ou seja, somente a Maneira 1 deve ser considerada.

$$Hi(z) \tag{2}$$

Tabela 1: Maneiras de representar a fórmula 2

Maneiras de representar a fórmula 2			
	Passo 1	***Passo 2***	***Passo3***
Maneira1	X(y)	Variable 'Hi'	Variable 'z'
Maneira 2	Variable 'Hi'	(x)	Variable 'z'

```
1 <mrow>
2         <mi>Hi</mi>
3         <mo>ApplyFunction</mo>
4         <mrow>
5                 <mo>(</mo>
6                 <mi>z</mi>
7                 <mo>)</mo>
8         </mrow>
9 </mrow>
```

Figura 11: Representação MathML da *Eq.2*

Para cada fórmula da DLMF então, realizamos a contagem dos cliques. Foram propostas duas abordagens: na primeira considerou-se a interface proposta por Cordeiro, enquanto que na segunda uma possível interface futura, cujos detalhes serão descritos a seguir.

- Interface proposta por Cordeiro

Com a padronização das bases de dados da DLMF e da interface, foi possível definirmos quantas e quais fórmulas poderiam ser representadas na interface proposta.

Uma fórmula consegue ser representada, se e somente se, todos os seus símbolos existirem na interface. Para identificarmos se esta afirmação era válida para uma determinada fórmula, verificamos símbolo por símbolo, se todos estavam presentes na interface. Então, agrupamos todas as fórmulas identificadas como possíveis, e com isto

descobrimos quantas poderiam ser representadas na interface proposta.

Das fórmulas possíveis, fizemos a contagem de quantos cliques eram necessários para representá-las. Nesta contagem, consideramos a quantidade de cliques necessária para representar cada símbolo separadamente, e a posição deste símbolo nas abas da interface. Isso porque se dois símbolos estiverem em uma mesma aba, basta contar os cliques para inserir ambos. No entanto, se estiverem em abas diferentes, então é necessário acrescentar mais um clique considerando a troca de abas. Para os botões de inserção de variável e número não ocorre a troca de abas, uma vez que esses dois elementos são acessíveis diretamente independente da aba (é possível vê-los na *Fig. 4* na parte inferior das abas). Ao final da contagem então somamos os cliques de todos os símbolos da fórmula e os cliques das mudanças de abas necessária, respeitando a ordem que estes aparecem na representação MathML da fórmula. Por exemplo, para a *Eq. 3*, são necessários 18 cliques para representá-la na interface proposta, como pode ser observado na *Tab. 2*.

$$z \in R_1 \tag{3}$$

Tabela 2: Contagem de cliques da Fórmula 3

Contagem de cliques da Fórmula 3						
N. cliques	3	1	1	7	3	3
Símbolo	Z	ϵ			R	1
Representação	Variable	\u2208	aba	sub	Variable	Numeric

- Possível Interface Futura

Para o estudo de uma interface futura optamos por medir o impacto que a troca de abas causa na quantidade de cliques. Assim assumimos uma interface hipotética onde os símbolos fossem representados em uma única tela (ou seja, sem abas). Além disso, consideramos a inclusão de todos os símbolos detectados como ausentes na interface de Cordeiro. Para os novos símbolos, assumimos que os respectivos 'botões' necessitassem de apenas um clique para a representação do símbolo.

Então, refizemos a contagem considerando esta nova representação, o que possibilitou que fosse encontrado um número de cliques para a representação de cada uma das fórmulas da DLMF.

IV. RESULTADOS

Nesta seção apresentaremos os resultados encontrados sobre a eficácia, subseção A; e sobre a eficiência, subseção B, da interface proposta por Cordeiro e do estudo da interface futura.

A. Eficácia

Após a execução do sistema, obtivemos o resultado que das 9.834 fórmulas disponíveis na DLMF, 2.891 podem ser representadas na interface proposta por Cordeiro. Isto representa apenas 29% das fórmulas da DLMF. Este número baixo se deve ao fato da interface não possuir todos os símbolos utilizados nas fórmulas da DLMF. Foram encontrados 70 símbolos que a DLMF usa, mas a interface não disponibiliza. Como por exemplo, temos alguns destes símbolos apresentados na *Tab. 3*. A lista completa será disponibilizada em breve no endereço: searchonmath.com/symbols.txt .

Tabela 3: Símbolos ausentes na interface do Cordeiro.

Símbolos Ausentes		
Representação Unicode	***Símbolo***	***Nome***
<m>\\u002a</m>	*	Asterisk
<m>\\u002c</m>	,	Comma
<m>\\u002e</m>	.	Full Stop
<m>\\u002f</m>	/	Solidus
<m>\\u003a</m>	:	Colon
<m>\\u003b</m>	;	SemiColon
<m>\\u005c</m>	\	Reverse Solidus
<m>\\u005e</m>	^	Circunflex Accent
<m>\\u007e</m>	~	Tilde

A interface futura, com a inclusão de todos os símbolos ausentes encontrados, permite a representação de todas as 9.834 fórmulas.

B. Eficiência

- Interface proposta por Cordeiro

Nas 2.891 fórmulas possíveis de se representar na interface, encontramos uma grande variedade de números de cliques necessários para representar as fórmulas. Temos desde fórmulas que necessitam de apenas 6 cliques, como a *Eq. 4*, até fórmulas que necessitam de 662 cliques, como a *Eq. 5*.

$$\lambda = \alpha \tag{4}$$

$$\begin{Bmatrix} j_1 & j_2 & j_3 \\ l_1 & l_2 & l_3 \end{Bmatrix} = \sum_s \frac{(-1)^s (s+1)!}{(s-j_1-j_2-j_3)!(s-j_1-l_2-l_3)!(s-l_1-j_2-l_3)!(s-l_1-l_2-j_3)!} \times \frac{1}{(j_1+j_2+l_1+l_2-s)!(j_2+j_3+l_2+l_3-s)!(j_3+j_1+l_3+l_1-s)!} \tag{5}$$

Para um melhor entendimento da necessidade de cliques como um todo, temos a sua distribuição, apresentada na *Fig. 12*. Percebe-se que a maior probabilidade se encontra na faixa de até 100 cliques.

Figura 12: Distribuição de cliques na interface do Cordeiro

Analisando visualmente as fórmulas e suas respectivas quantidades de cliques, percebemos que o simples fato de trocar de abas eleva o número de cliques necessários. A *Fig. 13* mostra exatamente esse impacto. Considerou-se a quantidade de cliques para produzir as fórmulas na interface original, e depois com a retirada das abas. A porcentagem de fórmulas que necessitam de menos de 50 cliques aumentou de 25% para 31%.

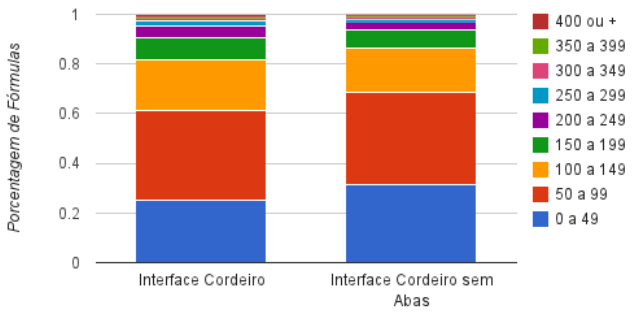

Figura 13: Comparação da distribuição de cliques na Interface proposta por Cordeiro, e esta mesma interface sem as abas

Mesmo a única diferença, neste caso, sendo a separação por abas presente na interface de Cordeiro, podemos perceber uma diferença significante entre as duas. Interessante notar que a representação da *Eq. 6* difere 82 cliques entre uma interface e outra. Enquanto na interface de Cordeiro são necessários 510 cliques para representá-la, na interface sem abas são necessários 428.

$$\int_0^\infty \frac{J_\mu(at)J_\nu(at)}{t^\lambda}dt = \frac{\left(\frac{1}{2}a\right)^{\lambda-1}\Gamma\left(\frac{1}{2}\mu+\frac{1}{2}\nu+\frac{1}{2}\lambda+\frac{1}{2}\right)\Gamma(\lambda)}{2\Gamma\left(\frac{1}{2}\lambda+\frac{1}{2}\nu-\frac{1}{2}\mu+\frac{1}{2}\right)\Gamma\left(\frac{1}{2}\lambda+\frac{1}{2}\mu-\frac{1}{2}\nu+\frac{1}{2}\right)\Gamma\left(\frac{1}{2}\lambda+\frac{1}{2}\mu+\frac{1}{2}\nu+\frac{1}{2}\right)} \quad (6)$$

- Possível Interface Futura

No estudo de uma interface futura, com todos os símbolos e sem nenhuma aba, onde todas as 9.834 fórmulas podem ser representadas, também encontramos uma variedade de números de cliques necessários para representar as fórmulas. A distribuição do número de cliques é apresentada na *Fig. 14.*

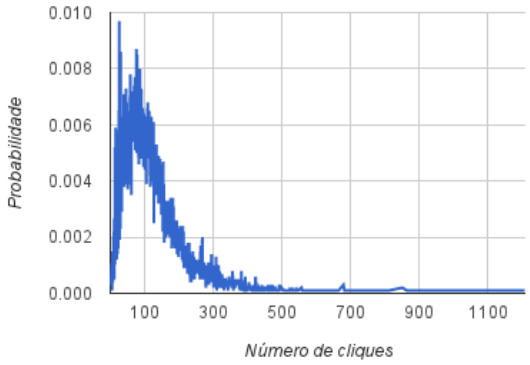

Figura 14: Distribuição de cliques na possível interface futura.

Apesar da ausência de abas indicar essa considerável redução nos cliques, fica claro que essa interface poderia aumentar consideravelmente o tempo de produzir uma fórmula, uma vez que o usuário tende a ficar mais tempo procurando pelo símbolo (por estarem todos dispostos na tela sem uma classificação, que é justamente o benefício de se usar abas). Essa questão ilustra a importância de se ter uma aba básica contendo os principais símbolos, questão que já é adotada na *Symbolab*, mostrada na *Fig. 3*. Assim as principais contribuições desse trabalho foram mostrar os símbolos faltantes (e que estarão presentes futuramente na *SearchOnMath*), e a necessidade de ter uma aba básica bem projetada de modo a minimizar a troca excessiva de abas. Pode-se perceber então que uma possível continuação desse trabalho seria agora elencar quais são os símbolos que mais aparecem. Isso daria um maior embasamento na produção futura da aba básica.

V. CONCLUSÕES E TRABALHOS FUTUROS

Pelos resultados obtidos, concluimos que a interface proposta por Cordeiro não possui boa eficácia, pois não consegue representar todas as fórmulas, nem uma boa eficiência, necessitando de um grande número de cliques para representar as que consegue. A questão do campo de busca não poder ser digitado pelo usuário impede que símbolos não cobertos sejam inseridos. E como mostrado, dada a complexidade da linguagem MathML, esperar que um usuário insira trechos desejados da fórmula usando essa linguagem pode ser bastante dispendioso. O ideal é cobrir a maior quantidade de símbolos de modo que o usuário não precise estar familiarizado com linguagens de cunho matemático. No entanto, caso seja necessária essa familiarização, a sintaxe da TeX é mais amigável e intuitiva, e deve se firmar como padrão adotado em páginas na Internet.

O estudo de interface que propusemos já se mostra eficaz, representando todas as fórmulas, e um pouco mais eficiente, necessitando de um número ainda alto, mas menor de cliques.

A retirada das abas se mostrou uma boa alternativa para diminuir a quantidade de cliques necessária para representar as

fórmulas. Entretanto, ainda pouco amigável, e o usuário poderia levar um maior tempo para encontrar os símbolos que deseja.

Assim, deixamos como proposta de trabalhos futuros, um estudo dos símbolos mais utilizados nas fórmulas, para que estes fiquem em uma aba básica, uma barra de busca com a função de autocompletar, que permita a inserção de letras do teclado e com a possibidade de importar uma fórmula no formato TeX. Todas estas características podem fornecer uma interface mais eficiente.

REFERÊNCIAS

[1] Qmee. http://blog.qmee.com/qmee-online-in-60-seconds, acessado em 19 de agosto de 2013.

[2] World Wide Web Size. http://www.worldwidewebsize.com, acessado em 19 de agosto de 2013.

[3] Youssef, A. “Roles of math search in mathematics”. 5th International Conference on Mathematical Knowledge Management, pp. 2-16., 2006.

[4] G. L. Cordeiro, “Interface para ferramenta de busca de conteúdo matemático.”, Alfenas: UNIFAL-MG, 2010.

[5] F. B. Gonzaga, “Recuperação de informação orientada ao domínio da matemática” , Rio de Janeiro: UFRJ/COPPE, 2013.

[6] M. Kohlhase, I. Sucan, “A search engine for mathematical formulae”, Lecture Notes in Computer Science - Artificial Intelligence and Symbolic Computation, vol. 4120, pp. 241–253, 2006.

[7] A. Asperti, F. Guidi C. S. Coen, T. Enrico, S. Zacchiroli “A content based mathematical search engine: Whelp”, Lecture Notes in Computer Science - Types for Proofs and Programs, vol. 3839, pp. 17–32, 2006.

[8] D. E. Knuth, The TeXbook. 1st ed. Cambridge, Massachusetts, USA, Addison-Wesley Professional, 1984.

[9] M. Abramowitz, I. A. Stegun. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. New York, NY, Dover Publications, 1965.

[10] R. F. Boisvert, D. W. Lozier, “Handbook of Mathematical Functions”, in A Century of Excellence in Measurements, Standards, and Technology - A Chronicle of Selected NBS/NIST Publications, 1901-2000 (D. Lide, ed.), NIST Special Publication 958, pp. 135–139, 2000.

[11] R. F. Boisvert, C. W. Clark, D. W. Lozier, F. Olver, “A special functions handbook for the digital age”, Notices of the AMS, pp. 905-911, Ago 2011.

[12] XML – Extensible Markup Language. http://www.w3.org/xml/, acessado em 19 de agosto de 2013

[13] MathML3 - http://www.w3.org/TR/MathML3, acessado em 19 de agosto de 2013.

[14] ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs). Part 11 - Guidelines for specifying and measuring usability. Gènève: International Organisation for Standardisation, 1997.

[15] J. Nielsen, Usability engineering, Academic Press, New York (1993), pp. 85- 87.

[16] S. K. Card, T. P. Moran, A. Newell, The Model Human Processor: An Engineering Model of Human Performance. In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), Handbook of Perception and Human Performance**.** vol. 2: Cognitive Processes and Performance, 1986, pages 1–35.

[17] S. K. Card, T. P. Moran, A. Newell, The Psychology of Human Computer Interaction. Lawrence Eribaum Associates. 1983.

[18] D. Kieras, "Towards a practical GOMS model methodology for user interface design". In Martin Helander. Handbook of Human-Computer Interaction. Amsterdam, The Netherlands: Elsevier Science Publishers. pp.135–157. 1988.

[19] Advantages and weaknesses of GOMS Overview, http://en.wikipedia.org/wiki/GOMS, acessado em 19 de agosto de 2013.