UNIVERSIDADE DO VALE DO RIO DOS SINOS

CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

Vinícius Costa de Souza

SWService: uma biblioteca para a escrita da Língua Brasileira de Sinais baseada em Web Services

São Leopoldo, janeiro de 2005

UNIVERSIDADE DO VALE DO RIO DOS SINOS

CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

Vinícius Costa de Souza

SWService: uma biblioteca para a escrita da Língua Brasileira de Sinais baseada em Web Services

Dissertação a ser avaliada como requisito parcial para a obtenção do título de Mestre em Computação Aplicada.

Orientador
Prof. Dr. Sérgio Crespo Coelho da Silva Pinto

São Leopoldo, janeiro de 2005

Ficha catalográfica elaborada pela Biblioteca da
Universidade do Vale do Rio dos Sinos

S829s Souza, Vinícius Costa de
SWService: uma biblioteca para a escrita da Língua Brasileira de Sinais baseada em Web Services / por Vinícius Costa de Souza – 2005.
128 f.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, 2005.
"Orientação: Prof. Dr. Sérgio Crespo Coelho da Silva Pinto, Ciências Exatas e Tecnológicas".

1.Engenharia de software. 2. Língua Brasileira de Sinais. 3. Sign Writing . 4. SWService I. Título.

CDU 004:81'221.24

Catalogação na Publicação:
Bibliotecária Eliete Mari Doncato Brasil - CRB 10/1184

"A voz dos surdos são as mãos e os corpos que pensam, sonham e expressam. As línguas de sinais envolvem movimentos que podem parecer sem sentido para muitos, mas que significam a possibilidade de organizar as idéias, estruturar o pensamento e manifestar o significado da vida para os surdos. Pensar sobre a surdez requer penetrar no mundo dos surdos e ouvir as mãos que, com alguns movimentos, nos dizem que para tornar possível o contato entre os mundos envolvidos se faz necessário conhecer a língua de sinais."

Ronice Müller de Quadros

Dedico este trabalho às pessoas mais importantes da minha vida: minha esposa Ana Rita Breier, minha razão de viver; meus pais Orlando e Nilsa, que são a base do meu caráter e formação e meus irmãos, Eduardo, Fernanda e Cristian, meus companheiros de sempre.
A vocês pelo amor, apoio, confiança, por tudo.
Agradeço a Deus por tê-los em minha vida.

Agradecimentos

Gostaria de agradecer a algumas pessoas que me auxiliaram ao longo destes dois anos.

- A Deus, pelo dom da vida e por me dar forças para lutar sempre.
- A minha esposa, Ana Rita Breier, pela pessoa especial que é, por todo seu amor, carinho, apoio e compreensão.
- Aos meus amigos e colegas Leonardo Lemes Fagundes, Letícia Rafaela Rheinheimer e Daniela Rossi pelas trocas de idéias, ajuda e amizade.
- A minha família pelo apoio e amor incondicional.
- Ao Professor Dr. Sérgio Crespo Coelho da Silva Pinto, pela orientação e auxílio durante o desenvolvimento deste trabalho.
- A Professora Marianne Rossi Stumpf, surda especialista em SignWriting, pelos nossos encontros onde, entre sinais e palavras, muitas contribuições surgiram.
- As pessoas que participaram da pesquisa de opinião, por acreditarem no trabalho e por disporem do seu tempo.
- A Capes e MEC pelo incentivo financeiro concedido pelo Programa de Apoio à Pesquisa em Educação a Distância.
- O trabalho foi parcialmente financiado pela Fundação de Ampara a Pesquisa do Estado do Rio Grande do Sul - FAPERGS - por meio do projeto "Libras_F: Um Framework para a Geração de Feramentas de Comunicação para Ambientes de EAD Utilizando a Linguagem Libras". Edital Pro-Livre, Processo: 02/0537.6
- A todos que de alguma forma colaboraram para a realização deste trabalho.

Resumo

Grandes avanços na ciência têm ocorrido nos últimos anos, provocando um processo de mudanças sócio-culturais, no qual o acesso a novas tecnologias se torna extremamente importante para todos. Entretanto, a grande maioria das ferramentas computacionais não está preparada para uso por pessoas portadoras de necessidades especiais. Mesmo assim, algumas iniciativas estão sendo desenvolvidas para difundir as línguas de sinais como forma de apoiar a inclusão digital e social dos surdos. Porém, a utilização dessas línguas em software ainda é bastante complexa e dispendiosa. Assim, este trabalho teve como objetivo o desenvolvimento da SWService, uma biblioteca que visa fornecer os recursos necessários para que softwares baseados na web possam utilizar a Língua Brasileira de Sinais. A solução apresenta como principal vantagem a utilização da tecnologia de Web Services, o que permite seu uso sem necessidade de desenvolvimento ou instalação local. Além disso, foram realizados um estudo de caso e uma pesquisa de opinião sobre o Sign WebForum, um fórum de discussão que utiliza a SWService para permitir a escrita e leitura de mensagens em Português e Libras.

Palavras-chave: Língua Brasileira de Sinais, Sign Writing, Web Services, SWService

Abstract

Great advances in science have been made in the last years promoting a social-cultural changing process, in which the access to the new technologies becomes too important for all. However, most of computational tools are not prepared to people who has special necessities. Some initiatives have been developed to spread out the sign languages as a way to make possible the deaf digital and social inclusion. But, the use of these languages in software is still complex and expensive. Thus, the aim of this work was to develop a library, SWService, which provides the wanted features to enable web-based software to use the Brazilian Sign Language. This solution presents as main advantage the use of Web Services technology witch allows the SWService use without local development or installation necessity. Besides, a case study and an opinion research applied to Sign WebForum, a discussion forum which uses the SWService to permits the writing and reading of messages both in Portuguese and Libras, were realized.

Keywords: Brazilian Sign Language, Sign Writing, Web Services, SWService

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

API	:*Application Program Interface*
Capes	:Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CSS	:*Cascading Style Sheet*
DAC	:*Deaf Action Committee*
DTD	:*Document Type Definition*
DOM	:*Document Object Model*
EAD	:Ensino a Distância
FTP	:*File Transfer Protocol*
HTML	:*HiperText Markup Language*
HTTP	:*HiperText Transfer Protocol*
Libras	:Língua Brasileira de Sinais
LSKB	:Língua Brasileira de Sinais Kaapor
L1	:Língua materna
L2	:Segunda língua
MEC	:Ministério da Educação
MIME	:*Multipurpose Internet Mail Extension*
OMT	:*Object Modeling Technique*
PAPED	:Programa de Apoio à Pesquisa em Educação à Distância
PEAR	:*PHP Extension and Application Repository*
REST	:*Representational State Transfer*
PHP	: *Hypertext Preprocessor*
RPC	:*Remote Procedure Call*
SAX	:*Simple API for XML*
SMTP	:*Simple Mail Transfer Protocol*
SOAP	:*Simple Object Access Protocol*
SSS	:*Sing Symbol Sequence*
SWML	:*SignWriting Markup Language*
SWService	:*SignWriting Web Service*
UDDI	:*Universal Distribution Discovery and Interoperability*
UML	:*Unified Modeling Language*
URL	:*Uniform Resource Locator*
XSLT	:*eXtensible Stylesheet Language Transformations*
XML	:*Extensible Markup Language*
WSA	:*Web Services Architecture*
WSDL	:*Web Services Description Language*

Sumário

1 Introdução

A popularização do computador e sua utilização em diversas áreas é fato inquestionável, assim como o uso da informática na educação e o crescente desenvolvimento de softwares educacionais. Entretanto, muitas pessoas, pelos mais variados motivos, não têm acesso às novas tecnologias da informação. Um dos problemas é o fato de a maioria dos softwares serem desenvolvidos sem levar em consideração a grande diversidade de usuários que os mesmos possam ter, o que vem a limitar a sua utilização por portadores de necessidades especiais [CAM 98].

Vivemos em um processo de grandes mudanças sócio-culturais onde a difusão de novas ferramentas que possibilitam transferência de conhecimento se torna extremamente importante, a fim de preparar as pessoas para o exercício da cidadania e para a qualificação profissional. Segundo o último censo demográfico realizado pelo IBGE em 2002, cerca de 14,5% da população brasileira apresenta algum tipo de deficiência, sendo que 3,38% dos brasileiros possuem algum nível de deficiência auditiva [SOU 03b].

A audição é um sentido importante, e sua ausência pode provocar sérias dificuldades no desenvolvimento individual-social. Porém, se as peculiaridades dos surdos forem respeitadas e, se lhes for oferecida educação que respeite sua condição, podem se desenvolver perfeitamente [ROC 00].

Um dos grandes problemas enfrentados pelos surdos é não poder se expressar através da escrita de sua própria língua (língua de sinais). Por isso, precisam fazer uso da língua oral para escrever, o que é muito difícil, pois o código escrito de uma língua oral está fundamentado em um foneticismo, grafia baseada nos sons, o que dificulta seu aprendizado [SKL 01a]. Em função disso, sua comunicação se estabelece quase que exclusivamente de forma presencial, entre interlocutores um diante do outro, de forma que os surdos não podem usufruir totalmente das novas tecnologias como, por exemplo, o correio eletrônico e a Internet [NOV 98].

Nos últimos anos, o SignWriting, um sistema para escrita de línguas de sinais, desenvolvido por Valerie Sutton e difundido pelo *Deaf Action Committee for SignWriting* - DAC em La Jolla na Califórnia, vem despertando interesse de lingüistas, pesquisadores da língua de sinais, professores e surdos de vários países. O DAC vem oferecendo suporte a um projeto de alfabetização em SignWriting e várias escolas para surdos em todo mundo vêm desenvolvendo educação bilíngüe [MAC 99b].

As experiências no Brasil de uso da informática com surdos ressaltam diversos pontos positivos, como mudança na dimensão cognitiva, afetiva e social [SAN 97] e [SOU 03a]. Apesar disso, a classe surda brasileira ainda é tratada como deficiente, e pouco se tem a lhe oferecer no que se refere aos novos avanços tecnológicos [PEL 98].

Dentro desse contexto, foi desenvolvido na Unisinos o Sign WebMessage (http://www.inf.unisinos.br/swm) um ambiente para comunicação assíncrona na web, através do qual pode-se interagir tanto através da escrita da língua portuguesa quanto através da escrita da Língua Brasileira de Sinais - Libras. Nas mensagens, os sinais podem ser visualizados em SignWriting e, opcionalmente, seus significados em português, o que proporciona uma forma de aprendizagem de ambas as línguas. Essa ferramenta tem como

objetivo principal minimizar as dificuldades de comunicação escrita e à distância entre os surdos e entre surdos e ouvintes [SOU 03a] e [SOU 03b].

Através desse trabalho de mestrado, pretende-se dar continuidade ao projeto Sign WebMessage, otimizando seus recursos específicos para utilização da língua de sinais de forma a modelá-los, implementá-los e disponibilizá-los através da tecnologia de Web Services proporcionando, assim, uma maior divulgação e uso das línguas de sinais em ambientes computacionais.

1.1 Motivação

Góes [GOE 96], Quadros [QUA 97] e Skliar [SKL 99] salientam que em função das dificuldades com as línguas orais a maioria das pessoas surdas não pode aproveitar plenamente os cursos a distância e a própria Internet. Um dos fatores, senão o principal, é que inexistem ambientes na web com suporte à comunicação de surdos baseados na Língua de Sinais. Além disso, inexistem ferramentas para a educação a distância que permitam a comunicação escrita entre surdos e surdos e ouvintes que estejam baseadas tanto na escrita da língua de sinais quanto na escrita da língua oral de forma integrada. Inclusive, devido ao fato de a própria escrita da língua de sinais ser recente, existem poucas ferramentas computacionais que a utilizem nesta modalidade.

Os esforços para a utilização do sistema SignWriting em softwares ainda são muito imaturos e se dão isoladamente sendo que cada ferramenta implementa os recursos para criação, edição, consulta e escrita dos sinais de diferentes formas, utilizando diferentes linguagens e técnicas de programação. A cada nova ferramenta o esforço necessário para que a aplicação possibilite a utilização das línguas de sinais se repete e conseqüentemente esta fase do desenvolvimento acaba consumindo muito tempo e recursos nos projetos.

Assim, um dos primeiros ambientes web, no Brasil, a trabalhar com a manipulação de língua de sinais escrita através do SignWriting, foi o Sign WebMessage. Esse fato se dá justamente pela dificuldade de se disponibilizar nos softwares, principalmente em aplicações web, o uso da língua de sinais na forma escrita.

Por isso, pretende-se neste trabalho utilizar técnicas computacionais e de engenharia de software, Web Services *e Design Patterns*, a fim de criar uma camada de software que implemente as funcionalidades básicas para manipulação dos sinais (criação, edição e consulta) e disponibilizá-la através da Internet, via interfaces bem definidas, de forma a reduzir a complexidade, o tempo e os custos envolvidos no desenvolvimento de novos ambientes e ferramentas para Internet que utilizem as línguas de sinais na forma escrita.

1.2 Problema

Diversas iniciativas estão sendo desenvolvidas para difundir as línguas de sinais, inclusive através do desenvolvimento de recursos de tecnologia da informação, como forma de apoiar não apenas a inclusão digital, mas também a inclusão social das pessoas surdas. Entretanto, uma das grandes dificuldades ocorre no momento de se registrar as línguas de sinais na forma escrita. Por isso, existem poucos ambientes que as utilizam como idioma principal em suas funcionalidades e interfaces.

Alguns softwares já fazem uso do sistema para escrita das línguas de sinais SignWriting. Porém, a tarefa para disponibilizá-lo ainda é bastante complexa, dispendiosa e demanda muito tempo e recursos. Isso ocorre, porque ainda não existe uma solução padrão confiável e documentada que possa ser facilmente reutilizada de forma a diminuir o tempo e a complexidade para o desenvolvimento de novos softwares que utilizem a escrita de línguas de sinais.

1.3 Questão de pesquisa

A questão central deste trabalho é: definir os recursos necessários para utilização das línguas de sinais em ambientes web e implementá-los em uma camada de software afim de disponibilizá-los amplamente através da Internet, para que outras aplicações possam fazer uso de forma rápida e eficiente, sem a necessidade de implementar ou instalar localmente e com garantia de qualidade e confiabilidade?

1.4 Áreas de pesquisa envolvidas

Este trabalho utilizará técnicas de engenharia de software a fim de minimizar as dificuldades encontradas pela área de informática para surdos, unindo estudos em Computação, Educação e Lingüística. Para o desenvolvimento do trabalho são necessários estudos sobre *Design Patterns*, XML, desenvolvimento de software para Internet e Web Services. São importantes, também, estudos na área de educação e comunicação de surdos, além da Língua Brasileira de Sinais e do sistema SignWriting. A figura 1.1 apresenta as áreas relacionadas ao trabalho.

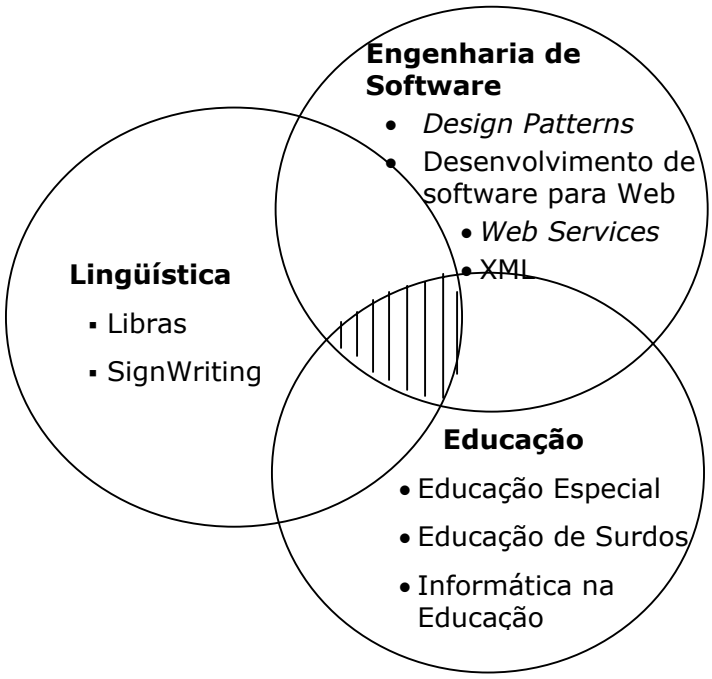


Figura 1.1 - Áreas relacionadas ao trabalho

É importante salientar que este trabalho assume uma proposta bilíngüe, a qual reconhece a língua de sinais como a língua materna (L1) dos surdos ao mesmo tempo em que reconhece a língua da sociedade ouvinte em que os surdos vivem como sua segunda língua (L2). Assume-se, também, uma visão bicultural que encara os surdos como membros de uma comunidade minoritária com língua e cultura próprias.

1.5 Objetivos

Pretende-se consolidar todo o conhecimento e experiência adquiridos no desenvolvimento do Sign WebMessage através da modelagem das funcionalidades fundamentais da aplicação que tratam da criação, edição, consulta e escrita de sinais. Além disso, como forma de socializar este conhecimento e contribuir para o surgimento de novas

aplicações na Internet que façam uso do sistema SignWriting, pretende-se implementar e disponibilizar essas funcionalidades através da tecnologia de Web Services.

Assim, o objetivo geral deste trabalho será modelar e desenvolver uma biblioteca chamada SWService - SignWriting Web Service - que utilizará a tecnologia de Web Services de modo a fornecer os recursos necessários para que softwares baseados na web possam utilizar o sistema para escrita das línguas de sinais - SignWriting. Este serviço proporcionará a difusão do SignWriting e da própria língua de sinais, pois permitirá que qualquer aplicação possa utilizar o SignWriting sem que seja necessário ter esta camada de software instalada ou construída localmente.

Visando alcançar este objetivo, definiu-se os seguintes objetivos específicos:

- realizar um estudo aprofundado sobre as tecnologias necessárias para o desenvolvimento de Web Services e sobre *Design Patterns*;
- modelar os recursos a serem desenvolvidos;
- implementar a biblioteca de serviços *SWService*, utilizando tecnologias baseadas em Web Services*;*
- formalizar a arquitetura e interoperabilidade dos serviços através do uso e da extensão da linguagem *SignWriting Markup Language* – SWML;
- avaliar o potencial da biblioteca desenvolvida em um estudo de caso, onde a mesma será validada através de sua utilização em uma aplicação;
- tornar os serviços públicos e disponíveis para quaisquer outras aplicações.

1.6 Organização do volume

Este volume está organizado em 7 capítulos. O capítulo 2 apresenta a revisão bibliográfica realizada para o desenvolvimento do trabalho. Foram realizados estudos sobre educação de surdos, Língua de Sinais, sistema SignWriting, *SignWriting Markup Language* – SWML, *Design Patterns,* Web Services e sobre as possibilidades de desenvolvimento e uso de Web Services com PHP.

No capítulo 3 são apresentados os trabalhos relacionados, ou seja, os softwares já desenvolvidos que utilizam a língua de sinais escrita, além de um estudo comparativo.

No capítulo 4 é apresentado o aprimoramento realizado no Sign WebMessage como base para o desenvolvimento da SWService.

Já no capítulo 5 a biblioteca SWService é descrita detalhadamente. São apresentados a sua arquitetura, casos de uso, modelo entidade-relacionamento e diagrama de classes, além dos serviços implementados: *getSign, dicPort, dicSW* e *createSign*. São apresentados, também, exemplos de uso de cada serviço e um estudo de caso aplicado a SWService, no qual seus serviços foram utilizados pelo Sign WebForum, um fórum de discussão que permite escrita e leitura de sinais em Libras.

Finalmente, no capítulo 6 são apresentadas as conclusões e trabalhos futuros referentes a este trabalho e no capítulo 7 as referências bibliográficas.

2 Contexto do trabalho

2.1 Educação de Surdos

A preocupação com questões educacionais e procedimentos de instrução para pessoas surdas aparece em documentos apenas a partir do século XVI, quando começam a ser relatados casos de preceptores que se propunham a educar e desenvolver a fala de surdos da nobreza como condição necessária para preservar seu lugar social ou seus direitos de herança [PER 01]. No século XVIII, a primeira escola pública para surdos foi fundada em Paris, pelo Abade de L'Epée. Seu trabalho educacional era baseado no uso de sinais, num sistema que incorporava elementos da língua falada. Outro tipo de iniciativa se desenvolvia ao mesmo tempo, na Alemanha, com a proposta de uma educação exclusivamente oralista, defendida por Heinicke. Essas alternativas educacionais se tornaram alvo de disputas. O uso dos sinais ainda continuava aceito no atendimento educacional, assim como a participação de professores surdos, mas o oralismo divulgou-se, foi ganhando adesões e veio a alterar o cenário [CAP 01].

No segundo congresso internacional sobre educação de surdos, realizado em 1880, em Milão (o primeiro havia sido realizado em Paris, dois anos antes), a visão oralista se impôs, com as teses de que só a fala permite integração do surdo à vida social e de que os sinais prejudicam o desenvolvimento da linguagem, bem como a precisão das idéias. Como essas metas de integração e desenvolvimento não foram atingidas, os debates continuaram, gerando a busca de caminhos alternativos [GOE 96].

A orientação oralista consolidou-se no final do século passado, predominou por um longo período e se faz presente ainda hoje. Nessa proposta, os esforços educacionais são apoiados, de forma exclusiva, no uso da língua majoritária. Entretanto, o oralismo passou a ser amplamente criticado pelo fracasso em oferecer condições efetivas para a educação e o desenvolvimento do surdo. Entre muitas críticas, aponta-se o fato de que dificulta ganhos nas esferas lingüística e cognitiva por exigir do surdo a incorporação da linguagem exclusivamente numa modalidade a qual este não pode ter acesso natural [SKL 01b], [GOE 96] e [CAP 01].

As indicações dos fracassos educacionais do oralismo conduziram, então, a propostas de ampliação dos recursos comunicativos. A corrente da comunicação total, que passa a se expandir a partir de meados deste século, defende o uso de múltiplos meios de comunicação, buscando trazer para a sala de aula os sinais utilizados pelas comunidades de pessoas surdas. Essa corrente teve aceitação crescente, mas foi incorporada em versões muito variadas. Basicamente, o que a caracteriza é o conjunto de recursos comunicativos, na busca de ensinar a língua majoritária e de dar acesso a outras áreas curriculares. Essa proposta resultou na criação de variados métodos e sistemas de comunicação, que podem constituir-se das seguintes possibilidades: língua falada sinalizada (codificada em sinais), língua falada sinalizada exata (variante da anterior, em que se busca a reprodução precisa da estrutura da língua), associação de códigos manuais para auxiliar na discriminação e articulação dos sons e combinações diversas de sinais, fala, alfabeto digital, gesto e pantomima [GOE 96].

Os debates em torno da comunicação total começaram a surgir desde sua proposta e as oposições intensificaram-se, ou porque os esforços para concretizar as diretrizes resultaram numa multiplicidade de soluções, como o uso de sistemas que não são línguas; ou porque acabaram orientando-se, implícita ou explicitamente, apenas à aprendizagem da língua

majoritária. As críticas apontam, ainda, para o fato de que as práticas de comunicação total servem mais aos pais e professores ouvintes do que aos alunos surdos [CAP 01].

Ao longo desses debates, emergiu uma orientação educacional comprometida com a efetiva formação bilíngüe da pessoa surda. A corrente do bilingüismo assume a língua de sinais como primeira língua da criança surda, que deve ser aprendida o mais cedo possível, e como segunda língua está aquela utilizada pelo grupo majoritário. A implementação dessa abordagem envolve problemas complexos, já que implica mudanças de concepção e reorganização de modos de atendimento em várias esferas institucionais, além da escola e da família. Experiências de educação bilíngüe vêm sendo desenvolvidas mais sistematicamente, e com caráter oficial, em alguns países como Uruguai, Venezuela e Suécia. Também há tentativas, de caráter mais localizado, de transição para o bilingüismo em outros países, como parece ser o caso do Brasil [GOE 96] e [CAP 01].

Segundo Góes [GOE 96], a deficiência não torna a criança um ser que tem possibilidades a menos. Ela tem possibilidades diferentes, por isso o planejamento educacional deve orientar-se para os pontos fortes da criança, e não para a falta. Por isso, a incorporação da língua de sinais mostra-se necessária para que sejam configuradas condições mais propícias à expansão das relações interpessoais, que constituem o funcionamento nas esferas cognitiva e afetiva e fundam a construção da objetividade. Portanto, os problemas tradicionalmente apontados como característicos da pessoa surda são produzidos por condições sociais. Não há limitações cognitivas ou afetivas inerentes à surdez, tudo depende das possibilidades oferecidas pelo grupo social para seu desenvolvimento, em especial para a consolidação da linguagem.

2.2 Informática na educação

O ambiente sociocultural do indivíduo, atualmente, está rodeado pela informática nas mais diversas situações do seu cotidiano. A educação e a escola, que visam à formação integral deste indivíduo, precisam explorar, da melhor forma possível, esta ferramenta. Não há mais possibilidade de se ignorar a presença do computador na sociedade moderna. É necessário que se prepare o aluno, seja especial ou não, para cada vez mais conviver com a informática [MAC 99a].

Segundo Valente [VAL 91], a implantação da informática na educação consiste basicamente em quatro componentes: o computador, o software educativo, o professor capacitado a usar o computador como ferramenta educacional e o aluno. O software educativo tem tanta importância quanto os outros, pois sem ele o computador jamais poderia ser utilizado na educação. Valente salienta que uma das questões fundamentais no desenvolvimento de software educativo é o aspecto pedagógico, o que o software se propõe a ensinar e como isso é realizado. Quanto ao conteúdo, o computador tem sido utilizado para ensinar informática e para ensinar praticamente qualquer assunto (ensino pela informática). Quanto à maneira como o ensino da informática ocorre, o software pode ser classificado em três grandes categorias: instrução auxiliada por computador, aprendizagem por descoberta e ferramentas educacionais tanto para o aluno como para o professor.

Através das novas tecnologias é possível resgatar uma nova forma de aprender e, com o computador como aliado no processo educativo, torna-se possível que os professores atuem como mediadores, cujo papel fundamental é facilitar a aprendizagem, atuando como orientador e estimulador do processo de ensino-aprendizagem. No caso da educação, sobretudo, o recurso é fundamental porque permite à criança superar suas limitações, comunicando e construindo seu conhecimento de forma criativa [CUN 00].

Vygotsky (1984) e Papert (1960), citados em [MAC 99a], consideram que a aprendizagem tem um comportamento dialético podendo orientar e estimular processos evolutivos internos desde que o indivíduo seja capaz de interagir com o seu meio ambiente sócio-cultural. Considerando que o ambiente sócio-cultural do indivíduo, atualmente, é rodeado pela informática nas mais diversas situações do cotidiano, a educação e a escola que visam à formação integral deste indivíduo precisam explorar o melhor possível esta ferramenta.

2.2.1 Informática na educação especial

Segundo a Lei de Diretrizes e Bases Brasileira - LDBEN, a educação especial tem os mesmos objetivos que a geral, sendo dever da família e do Estado, inspirada nos princípios de liberdade e nos ideais de solidariedade humana. Tem por finalidade o pleno desenvolvimento do educando, seu preparo para o exercício da cidadania e sua qualificação para o trabalho. Ainda segundo a LDB, entende-se por educação especial a modalidade de educação escolar para educandos portadores de necessidades especiais.

A diferença básica entre a educação geral e a especial é dada em termos de local de atendimento, tipo de material pedagógico, currículo trabalhado, profissionais envolvidos e individualização no atendimento. Educação especial é definida como a modalidade de ensino que se caracteriza por um conjunto de recursos e serviços educacionais especiais organizados para apoiar, suplementar e, em alguns casos, substituir os serviços educacionais comuns, de modo a garantir a educação formal dos educandos que apresentam necessidades educacionais muito diferentes das da maioria de crianças e jovens [CAM 98].

A educação especial encontra na informática uma área cuja principal característica é a de manipulação da informação voltada ao usuário. Utiliza-se o computador como ferramenta auxiliar no processo de ensino-aprendizagem, onde o aluno vivencia situações que possibilitem o desenvolvimento de suas potencialidades de maneira lúdica. Assim, torna-se possível que o aprendiz passe de objeto a ser educado a sujeito de sua própria aprendizagem, tornando-se pensador ativo e crítico, refletindo o seu conhecimento sobre determinado assunto e seu estilo de pensar [PAI 03].

A inserção de um trabalho pedagógico apoiado no computador pode despertar na criança, seja ela portadora de necessidades especiais ou não, o interesse e motivação pela descoberta do conhecimento, a partir do mecanismo do aprender fazendo. Assim, o portador de necessidades especiais pode utilizar o computador como uma ferramenta a mais em sua vida escolar, encontrando neste um maior leque de opções do que as oferecidas pela escola [MAC 99a].

Além disso, com uso do computador, as pessoas ganham um espaço onde podem romper as barreiras e reduzir os problemas de comunicação, pois além de permitir a exposição de suas idéias, o computador melhora a capacidade de expressar seus pensamentos, as torna mais descontraídas e participativas, facilitando o processo de sociabilidade e, conseqüentemente, a sua inclusão na sociedade [SAN 01].

2.2.1.1 Informática na educação de surdos

Stumpf [STU 00] afirma que a comunidade surda é uma nação sem localização geográfica, pois noventa por cento dos surdos pertencem a famílias ouvintes e a nacionalidade que eles passam a integrar consiste em sua forma própria de comunicar-se. Assim, uma comunidade com estas características pode, desde que com ferramentas adequadas, beneficiar-se, mais do que qualquer outra, de um ambiente em rede que ultrapasse os limites geográficos e possibilite a integração entre seus membros em torno de objetivos comuns [JOK 99].

O que se pode observar no uso da informática na educação de surdos é uma abordagem de ensino oral, bimodal e bilíngüe. No caso dos oralistas, existe uma ênfase na língua oral, não se atribuindo valor real às línguas de sinais. Enquadram-se nesta situação todos os softwares que têm por meta o treinamento de voz e a leitura labial, ou seja, todos aqueles que utilizam o computador fortemente como uma ferramenta em auxílio a tratamentos fonológicos. Nos programas bimodais a língua de sinais passa a ser utilizada, mas basicamente como um recurso para o ensino da língua oral. Já os programas baseados na abordagem bilíngüe consideram a língua de sinais como língua materna dos surdos e a língua oral como segunda língua, estando de acordo com a identidade e cultura surda [CAM 00].

Segundo Stumpf [STU 00], a ferramenta computador interessa muito aos surdos quando aliados a este encontram-se softwares, educacionais ou não, que são específicos para a comunidade de surdos e respeitam sua diferença. O problema é que ainda são raros os ambientes computacionais que trabalham com a língua de sinais. Porém, cada vez mais este quadro vem sendo alterado [ROC 00]. A seguir, são citados alguns trabalhos já desenvolvidos com e sobre língua de sinais no Brasil e no exterior.

Inicialmente podem ser considerados os trabalhos do grupo do professor Fernando Capovilla [CAP 96], nos quais foram desenvolvidos sistemas para comunicação de surdos utilizando sinais, textos e símbolos através de computadores em rede. Destacam-se, também, muitos trabalhos desenvolvidos em vários países que fazem uso do SignWriting (sistema para escrita de sinais) tais como o SignWriter[SUT 01], SignDic [MAC 99a], SignHTML [MAZ 01], SignEd [CAM 01] e SWEdit [TOR 02], entre outros. Quanto a ambientes computacionais disponíveis na Internet, podem ser encontrados vários dicionários como por exemplo, o HandSpeak (http://www.handspeak.com) e Dicionário Libras *on-line* (http://www.dicionariolibras.com.br), além do Sign WebMessage [SOU 03a] (http://www.inf.unisinos.br/swm).

2.3 Língua de Sinais

> *"Nas mãos de seus mestres, a Língua de Sinais é extraordinariamente bela e expressiva, um veículo para atingir a mente dos Surdos com facilidade e rapidez, e para permitir-lhes comunicar-se; um veículo para o qual nem a ciência nem a arte produziu um substituto à altura. Aqueles que não a entendem falham ao perceber suas potencialidades para os Surdos, sua poderosa influência sobre o moral e a felicidade social daqueles que são privados da audição, e seu admirável poder de conduzir o pensamento a mentes que, de outro modo, estariam em perpétua escuridão. Tampouco podem avaliar o poder que ela tem sobre os Surdos. Enquanto houver dois surdos sobre a face da Terra e eles se encontrarem, haverá sinais."*
>
> J. Schuyler Long (1910). *The Sign Language*

A comunicação é uma necessidade humana, e as linguagens oral e escrita são as formas mais comuns de comunicação. Por isso, pode-se dizer que: (a) a linguagem é natural do ser humano; (b) através da linguagem, o ser humano estrutura seu pensamento, traduz o que sente, registra o que conhece, se comunica com os outros, produz significação e sentido; (c) o ser humano cria novas linguagens para expressar o que pensa, sente, deseja e para comunicar-se com seus semelhantes [SAN 00].

Segundo Fernandes [FER 03a], linguagem é um sistema de comunicação natural ou artificial, já o conceito de língua é mais restrito. Língua é um tipo de linguagem e defini-se como um sistema abstrato de regras gramaticais. Além disso, ressalta-se o conceito de língua não só como meio de comunicação, mas, também, como um dos principais instrumentos de

desenvolvimento dos processos cognitivos do ser humano e, evidentemente, de seu pensamento. Por isso, a presença de uma língua é considerada fator indispensável ao desenvolvimento dos processos mentais. As línguas são denominadas oral-auditivas quando a forma de recepção não-grafada é a audição e a forma de reprodução é a oralização, já as línguas espaço-visuais são naturalmente reproduzidas por sinais manuais e sua recepção é visual.

A língua utilizada por um indivíduo para comunicação depende do grupo em que está inserido. Para os ouvintes, a comunicação se estabelece em termos oral-auditivos. No entanto, para os surdos pode se estabelecer em termos gestual-visuais, em que gestual significa o conjunto de elementos lingüísticos manuais, corporais e faciais necessários para a articulação e a significação visual-cultural do sinal [GOE 96]. Nas línguas de sinais, enquanto o emissor constrói uma sentença a partir desses elementos, o receptor utiliza os olhos para entender o que está sendo comunicado. Desta forma, já que a informação lingüística é percebida pelos olhos, os sinais são construídos de acordo com as possibilidades perceptíveis do sistema visual humano [MAC 99a].

A língua de sinais foi desvalorizada durante muito tempo, devido à intolerância da época (1820-70) com as minorias e com a preocupação dos pais e professores de surdos em ensiná-los a falar. Somente no final da década de 1950, se começou a dar a importância que a língua merece. Mesmo assim, somente a partir de 24 de abril de 2002 a Libras foi reconhecida como meio legal de comunicação e expressão das comunidades surdas do Brasil, de acordo com a lei Nº. 10.436, decretada pelo Congresso Nacional e sancionada pelo presidente da república, Fernando Henrique Cardoso.

Santarosa, em [SAN 00] afirma que "língua" designa um específico sistema de signos que é utilizado por uma comunidade para comunicação. Portanto, a Libras é uma língua natural surgida entre os surdos brasileiros com o propósito de atender às necessidades comunicativas de sua comunidade. Brito [BRI 95] e Fernandes [FER 03a] afirmam que são línguas naturais porque, como as línguas orais, surgiram espontaneamente da interação entre os surdos, além de, através de sua estrutura, poderem expressar qualquer conceito desde o descritivo/concreto ao emocional/abstrato.

Vygotsky [VYG 98], destaca que a linguagem não depende da natureza do meio material que utiliza, o que importa é o uso efetivo dos signos, de quaisquer formas de realização, que possam assumir papel correspondente ao da fala. A linguagem não está necessariamente ligada ao som, pois não é encontrada só nas formas vocais. Por isso, os surdos não são deficientes na esfera lingüística-comunicativa ou na construção da identidade social, mas podem tornar-se pelas condições em que se constituem como pessoas. Assim, a incorporação de uma língua de sinais mostra-se necessária para que sejam configuradas condições mais propícias à expansão das relações interpessoais, que constituem o funcionamento nas esferas cognitiva e afetiva e fundam a construção da subjetividade. [GOE 96].

As línguas de sinais são utilizadas pela maioria das pessoas surdas. No Brasil, existem duas línguas de sinais: a Língua Brasileira de Sinais Kaapor – LSKB, utilizada pelos índios da tribo Kaapor, cuja maioria são surdos, e a Língua Brasileira de Sinais - Libras, que é utilizada nos centros urbanos. A língua portuguesa, no caso dos surdos brasileiros, é considerada uma segunda língua [CAM 00].

As línguas de sinais são dotadas de toda a complexidade e utilidade encontradas nas línguas orais e, assim como elas, possuem gramáticas próprias, com regras específicas em seus níveis lingüísticos, fonológico, morfológico e sintático. Um fator que as diferencia é a estrutura seqüencial no tempo, onde as línguas orais são caracterizadas pela linearidade, pois

os fonemas se sucedem seqüencialmente em contraste com simultaneidade das línguas de sinais, em que estes possuem uma estrutura paralela, podendo emitir sinais envolvendo simultaneamente diversas partes do corpo do sinalizador [BRI 95] e [QUA 97].

Cabe salientar que a língua de sinais não é uma língua universal e, da mesma forma que a língua oral, é diferente em vários países, podendo até mesmo apresentar sinais que variam entre regiões e entre comunidades de surdos [MAR 00].

Além disso, os sinais são considerados, por muitas pessoas, como mímicas pelo fato de alguns possuírem representações icônicas. Entretanto, esse não é o aspecto mais significante da estrutura e uso da língua. Os sinais podem ser icônicos ou arbitrários. Os icônicos reproduzem a forma ou o movimento do que se quer referir. Porém, o estudo de Karnopp [KAR 94] mostra que pesquisadores concluem que iconicidade não é relevante na determinação da forma do sinal. Aliás, diversos processos lingüísticos e sociolingüísticos tendem a inibir a natureza icônica dos sinais, tornando-os mais arbitrários. O mesmo pode-se observar nas línguas orais, em que, por exemplo, um relógio não é chamado de *Tic tac* e nem um cachorro de *Au au*. Eles possuem vocábulos próprios para designá-los.

Com estudos cada vez mais aprofundados sobre línguas de sinais, sugere-se que esta muito se assemelha com as línguas orais, visto que ambas originaram-se naturalmente pela necessidade que o ser humano tem de se comunicar, fator este essencial para seu desenvolvimento. O que caracteriza a distinção entre as línguas é a diferença existente entre os sistemas fonológico, morfológico, sintático e semântico-pragmático [FER 03a]. Por isso, será apresentado a seguir a estrutura da Língua Brasileira de Sinais com o objetivo de informar a estrutura própria da língua, suas riquezas e especificidades.

2.3.1 Empréstimos lingüísticos

Para os empréstimos lexicais, a Libras desenvolveu um alfabeto manual (figura 2.1) que é constituído de configurações de mão que representam as letras do alfabeto da língua portuguesa. Através da datilologia, o alfabeto manual é utilizado para traduzir nomes próprios ou palavras para as quais ainda não há um sinal correspondente, para soletrar uma palavra com o intuito de mostrar como esta é escrita em português ou para explicar o significado de um sinal [QUA 03].

O alfabeto manual, apesar de configurar-se como empréstimo lingüístico, é um instrumento de grande valia para o processo de aquisição do português como L2, sendo utilizado como um meio para verificação, questionamento ou veiculação da ortografia da língua oral [CAM 01].

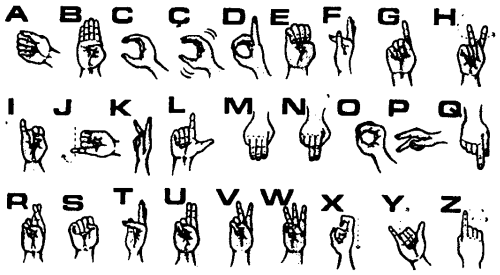


Figura 2.1 - Alfabeto manual em Libras [CAM 01]

Além da soletração digital das letras que constituem a palavra, há sinais que apresentam empréstimo parcial. Este é o caso do sinal para a palavra *azul*, que utiliza a soletração da primeira e da última letra com um movimento entre elas, como é ilustrado na figura 2.2 (a) [QUA 03].

Outro empréstimo lingüístico, chamado de Inicialização, recorre à utilização da configuração de mão que corresponde, no alfabeto manual, à primeira letra da palavra equivalente em português. Como exemplo, cita-se *Goiás* em que o sinal é feito a partir da configuração de mão correspondente à letra *g*, figura 2.2 (b) [MAC 99a].

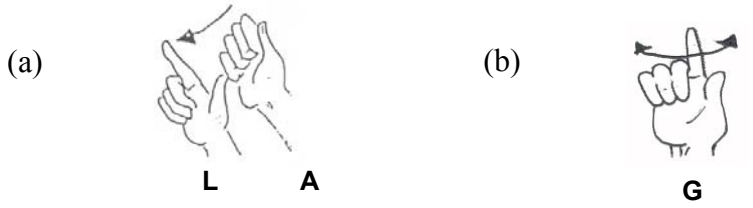


Figura 2.2 - Sinais referentes às palavras *azul* e *Goiás* [CON 02]

Segundo Brito [BRI 95], além dos empréstimos do tipo lexical e de inicialização, há o empréstimo de itens lexicais de outras línguas de sinais. Também há empréstimos de domínio semântico, identificado na maioria dos sinais referentes a cores e os empréstimos de ordem fonética, que são obtidos pela tentativa de representação visual do som que constitui a palavra em português, tal como são percebidas pelo surdo.

2.3.2 Sistema fonológico

O sistema fonológico define as unidades mínimas da língua. Nas línguas orais o plano fonológico caracteriza-se pela organização dos fonemas [FER 03]. A fonologia das línguas de sinais estuda as configurações das mãos, o ponto de articulação e os movimentos, que são tidos como parâmetros primários. A região de contato, orientação e disposição das mãos são considerados parâmetros secundários. Contudo, de acordo com Brito [BRI 95], existem outras classificações para analisar a fonologia de uma língua de sinais.

Apesar das muitas variações dos diferentes parâmetros, cada língua de sinais organiza-se a partir de um número limitado de configurações de mãos, pontos de articulação e movimentos.

A configuração das mãos refere-se às diversas formas que as mãos podem tomar na realização do sinal. Karnopp [KAR 94] salienta que essas configurações podem se diferenciar pela extensão, que é o lugar e o número de dedos estendidos; pela contração, que se refere a mãos fechadas ou abertas; e pelo contato e/ou divergência dos dedos. As configurações ainda podem variar apresentando somente uma mão configurada, uma mão configurada sobre a outra que serve como apoio ou as duas mãos configuradas de forma espelhada. Alguns exemplos são apresentados na figura 2.3.

Figura 2.3 - Variações quanto ao uso das mãos [CAP 01]

A configuração de mão pode permanecer a mesma durante a articulação de um sinal ou pode ser alterada, passando de uma configuração para outra, como é o caso dos sinais compostos, formados pelo processo de composição, pela adjunção de dois sinais simples em

formas compostas [QUA 03]. Como exemplo, pode-se citar o sinal para *igreja*, formado pelos sinais para *casa* e *cruz* (figura 2.4).

Figura 2.4 - Exemplo de sinal composto [CAP 01]

Quanto ao movimento do sinal, para que este seja realizado é preciso haver um objeto e um espaço. Nas línguas de sinais, as mãos do enunciador representam o objeto, enquanto o espaço em que o movimento se realiza é a área em torno do corpo do enunciador. O movimento pode ser analisado levando-se em consideração o tipo, a direção, a maneira e a freqüência do sinal. O tipo de sinal refere-se às variações do movimento das mãos, dedos, pulsos e antebraços. Quanto à direção, o sinal pode ser unidirecional, bidirecional ou multidirecional, uma vez que a conjugação verbal da língua oral é representada, nas línguas de sinais, pelo movimento, assim como o sujeito e o objeto são marcados com verbos direcionais. A maneira do sinal, descreve a qualidade, a tensão e a velocidade, podendo haver movimentos mais rápidos, mais tensos ou mais tranqüilos, o que diferencia, por exemplo, o sinal para *feliz* do sinal para *muito feliz*. Já a freqüência, indica se os movimentos são simples - feitos uma só vez - ou repetidos [MAC 99a].

O ponto de articulação, que é outro parâmetro do sistema fonológico, refere-se ao local do corpo do enunciador em que o sinal é realizado. Em Libras, este espaço é limitado e vai do topo da cabeça até a cintura, como é mostrado na figura 2.5 (a). De acordo com Brito [BRI 95], os sinais realizados próximos a determinadas partes do corpo pertencem a um campo semântico específico. Assim, em geral, o que se refere à visão é realizado próximo dos olhos, o que se refere à alimentação perto da boca, o que se refere a sentimentos próximo do peito e assim por diante. Além de espaços bem delimitados, o ponto de articulação pode ocorrer em espaço neutro quando o local da produção deste não é relevante.

A localização no espaço onde os sinais são realizados é tão importante que, conforme for, o significado do sinal pode ser alterado. Os sinais para *aprender* e para *laranja* (fruta) se diferenciam apenas pelo ponto de articulação, como mostra a figura 2.5 (b).

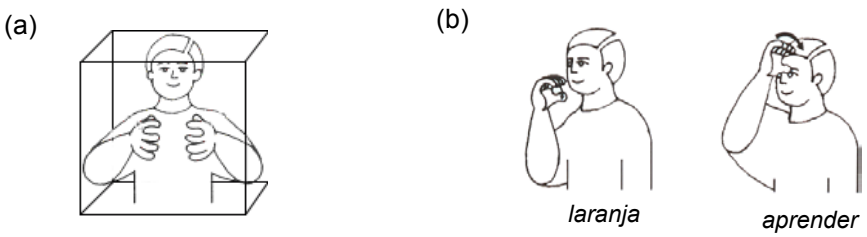


Figura 2.5 – Ponto de articulação dos sinais [CAP 01]

2.3.3 Sistema morfológico

O plano morfológico se caracteriza pelo estudo da forma. Dividi-se em dois subplanos: o das classe de palavras propriamente dito e aquele que estuda a estrutura e a formação das palavras. Assim como as línguas orais possuem um sistema para formação de palavras, as línguas de sinais também o possuem. O que as diferencia de algumas línguas orais é que as línguas de sinais são sintéticas. O sinteticismo é uma característica de línguas como o grego e o latim, por exemplo. Por esta razão, as línguas de sinais não têm artigo, como ocorreu no latim clássico. Ademais, o seu sinteticismo permite que não haja uma lista ampla como a do português, no que se refere às classes das preposições e conjunções [FER 03]. A seguir serão apresentadas algumas características quanto ao gênero, grau, tempo e negação em Libras.

Quanto ao **gênero**, segundo Brito [BRI 95] e Fernandes [FER 99], ocorre uma ausência de marcação de gênero em Libras, a menos que esta informação seja relevante. Para este caso, a indicação é feita colocando-se o sinal referente à *mulher* ou *homem* após o sinal que deseja-se flexionar, independente de serem pessoas ou animais. Em alguns casos, a flexão é feita com sinais próprios, como é o caso dos sinais para *mãe* e *pai* e para *galo* e *galinha*. Alguns exemplos são apresentados na figura 2.6.

Figura 2.6 - Sinais com flexão de gênero [CAP 01]

Quanto ao **grau**, o sinal pode diferenciar-se pela intensidade, movimento, velocidade, expressão facial ou ser um sinal próprio. Segundo Brito [BRI 95], o grau dos adjetivos pode ser representado de diversas formas e o dos substantivos são expressos através dos sinais *muito, pouco, grande* ou *pequeno*, geralmente após o sinal.

Figura 2.7 - Exemplos de sinais com indicação de grau [CAP 01]

Conforme Fernandes [FER 99], na Libras o **tempo** é manifesto por relações espaciais em que o passado é indicado por um movimento de mão para trás, o futuro por um movimento de mão para frente e o presente, no espaço imediatamente à frente do corpo do sinalizador. Além destes sinais para marcação do tempo, há sinais próprios para *ontem, hoje, amanhã* e *futuro* conforme ilustrado na figura 2.8.

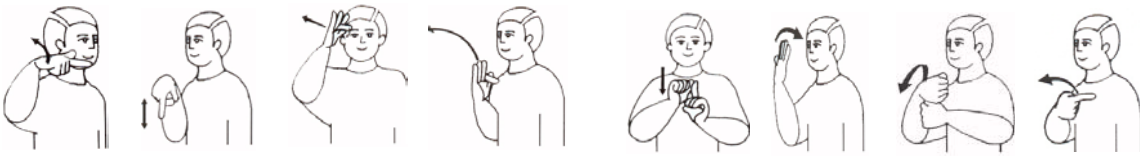


Figura 2.8 - Exemplos de sinais com indicação de tempo [CAP 01]

A **negação** pode ser feita através do movimento de negação com a cabeça, juntamente com o sinal que se deseja negar, ou do movimento do dedo seguido do sinal, ou ainda do sinal que já incorpora a sua negação [QUA 03]. Alguns exemplos são ilustrados na figura 2.9.

Figura 2.9 - Exemplos de sinais com negação [CAP 01]

2.3.4 Sistema sintático

A sintaxe é o estudo das inter-relações dos elementos estruturais das frases e das regras que regem a combinação das sentenças. Estudos que têm sido realizados na descrição da sintaxe das línguas de sinais atestam como característica principal o sinteticismo, já assinalado anteriormente. Das inter-relações dos elementos estruturais e das regras que regem a combinação das sentenças, podemos afirmar que esta combinação de sinais apresenta regras próprias e básicas que a caracterizam como língua [FER 03].

De acordo com Brito [BRI 95], a organização sintática básica dos sinais segue a estrutura (sujeito-verbo-objeto) SVO. Entretanto, a organização também pode ser OSV ou SOV, que são permitidas nos casos de um dos elementos da frase ser topicalizado. A topicalização refere-se à informação que é deslocada para o começo da sentença e fica de forma destacada introduzindo o assunto a ser tratado na frase.

Brito [BRI 95], Fernandes [FER 99] e Karnopp [KAR 94] afirmam que é raro o uso de artigos, preposições e conjunções em Libras bem como, de um modo geral, inexistem os verbos de ligação.

Basicamente os verbos na Libras se apresentam em três classes:

- verbos simples – são verbos que não se flexionam em pessoa e número e não tomam afixos locativos. Alguns desses verbos se flexionam em aspecto. Exemplos desta categoria são: *conhecer, amar, aprender, saber e gostar;*
- verbos com concordância – são verbos que se flexionam em pessoa, número e aspecto, mas não tomam afixos locativos. Exemplos desta categoria são: *dar, enviar, responder, perguntar, dizer e provocar;*
- verbos espaciais – são verbos que tem afixos locativos. Exemplos desta classe são: *viajar, ir e chegar.*

2.3.5 Sistemas de classificação

Os sistemas de classificação são conjuntos de elementos visuais, entre os quais se pode encontrar ou definir relações para a visualização da imagem mental. Sendo o único recurso

dos surdos, a visualização da imagem do todo, uma vez que não podem usufruir do sentido da audição, os sistemas de classificação definem relações para determinar a comunicação. Os sistemas de classificação nos permitem explicar com clareza frases, palavras e objetos que não possuem sinal próprio. A divisão apresentada a seguir é meramente didática, pois os sistemas se interligam e se inter-relacionam intrinsecamente [BRI 95].

- Sistema descritivo – utiliza figuras geométricas para expor minuciosamente os elementos visuais, além de forma, tamanho, textura e cor.
- Sistema específico – este sistema retrata características especiais, com explicações minuciosas, como por exemplo, as particularidades do corpo.
- Sistema funcional – reproduz a imagem da ação, a maneira como um corpo ou parte do corpo age e atua.
- Sistema de locação – reproduz a imagem de como um corpo se relaciona num determinado lugar, definindo posições e localizações.
- Sistema instrumental – demonstra a imagem de como se serve, se utiliza alguma coisa.
- Sistema de pluralização – classifica números determinados ou indeterminados de alguma coisa, pessoa ou animal.
- Sistema de elementos na natureza – reproduz a imagem de elementos que não são sólidos, como por exemplo, o ar, a fumaça, a água, o fogo e a luz.

2.4 Sistema de escrita visual direta de sinais - SignWriting

> *"A escrita está entre as maiores invenções da história humana, talvez a maior, pois ela tornou a história possível."*
>
> Andrew Robinson 1995. *The story of writing*.

A língua de sinais é considerada por muitos lingüistas, professores e pelos próprios surdos como ágrafa, sendo a única forma de registro por meio de vídeos, desenhos ou fotos. Assim, sempre que os surdos precisam fazer uso de uma comunicação não presencial têm de recorrer à escrita da língua oral [QUA 00].

Costa [COS 98] afirma que, do ponto de vista da cultura surda, isso não só significa que as relações pessoais entre surdos precisam ser mediadas por elementos cultural-comunicativos que não lhes são próprios, como também implica necessariamente um processo de tradução entre a língua de sinais e a falada. Desta forma, mesmo que se produza em língua de sinais, esta produção só pode ser escrita se convertida para a escrita da língua falada.

Nenhuma forma de escrita de língua de sinais foi até agora amplamente divulgada e estabelecida. Porém, nos últimos anos, o SignWriting, um sistema de representação gráfica das línguas de sinais, vem sendo difundido e pesquisado por lingüistas, professores e surdos de vários países. O sistema de Escrita Visual Direta de Sinais SignWriting foi desenvolvido por Valerie Sutton e faz parte de um sistema maior, o Sistema de Escrita e Notação de Movimentos Sutton (*Sutton Movement Writing & Shorthand*). Trata-se de um completo sistema de notação de movimentos capaz de registrar todo e qualquer movimento, não apenas humano, mas também de animais e insetos. O sistema compreende cinco divisões:

1. DanceWriting para registrar a coreografia de danças;
2. SignWriting para registrar as línguas de sinais;

3. MimeWriting para registrar a mímica e a pantomima clássicas;
4. SportsWriting para registrar a ginástica, a patinação e o caratê;
5. ScienceWriting para registrar a fisioterapia, a linguagem corporal e os movimentos de animais e insetos.

O DanceWriting foi desenvolvido e ensinado pela primeira vez no Royal Danish Ballet em 1974. Naquele mesmo ano, o SignWriting começou na Dinamarca, na Universidade de Copenhague. O propósito do SignWriting é permitir que surdos possam ler e escrever os sinais e empregar esta escrita como ferramenta para o registro e aperfeiçoamento de sua língua, para seu desenvolvimento cognitivo na idade de alfabetização e para o início da edificação da sua história, cultura e literatura em sua língua materna [SUT 03a].

Conforme Sutton [SUT 03a] e Capovilla [CAP 01], o SignWriting é um sistema de escrita visual direta de sinais capaz de transcrever as propriedades sublexicais das línguas de sinais, os quiremas (do grego *quiros*, mão) ou configurações de mãos, sua orientação e movimentos no espaço e as expressões faciais associadas, do mesmo modo como o alfabeto fonético internacional é capaz de transcrever as propriedades sublexicais das línguas orais, os fonemas. Assim como o alfabeto fonético permite uma descrição detalhada dos fonemas de uma língua falada e um registro preciso das palavras que resultam de sua combinação, SignWriting permite uma descrição detalhada dos quiremas de uma língua de sinais e um registro preciso dos sinais que resultam de sua combinação.

Contudo, o *SignWriting* objetiva ser mais que um mero sistema de notação científica para a descrição detalhada de sinais em estudos lingüísticos. Ele objetiva ser um sistema prático para a escrita de sinais que possibilite a comunicação rápida e inequívoca dos surdos no dia a dia.

O sistema *SignWriting* é definido por três estruturas básicas: posição de mão, movimentos e contato. Além destas, existem símbolos para expressões faciais, pontos de articulação, pontuação, entre outros [SUT 03a].

2.4.1 Posição de mão

As configurações básicas são mão fechada, circular e aberta (figura 2.10). Os outros símbolos de mão são variações destes símbolos básicos [SUT 03a].

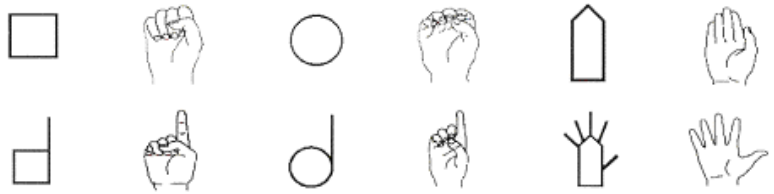


Figura 2.10 - Configurações básicas de mão [CAP 01]

A orientação da palma da mão é representada com as cores preto e branco. A palma da mão é representada pela cor branca (clara ou vazia). O dorso da mão é representado pela cor preta (escuro ou cheio). O lado da mão é representado com uma metade em branco e a outra em preto, sendo que a metade branca sempre mostra a direção da palma. A figura 2.11 ilustra como escrever a mão plana dependendo de sua posição no espaço. Na figura, as mãos da primeira linha estão na vertical (os dedos apontam para cima) e as da segunda linha na horizontal, escritas com uma quebra ou falha na horizontal como ilustrado na figura [SUT 03a].

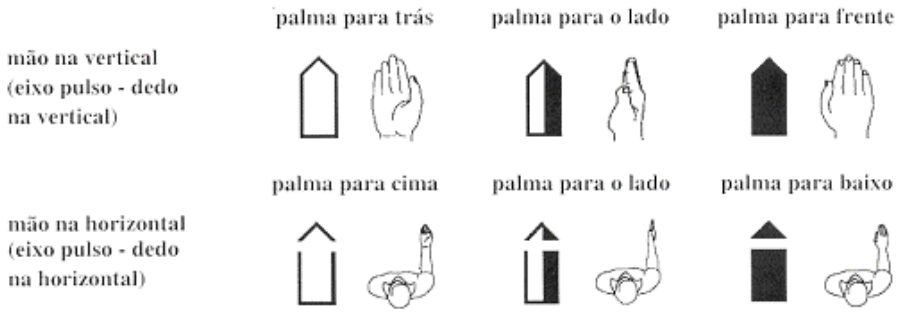


Figura 2.11 - Escrevendo a palma da mão [CAP 01]

Quanto às direções, os símbolos para as mãos podem apontar para oito posições diferentes, estando a mão no plano vertical ou horizontal [SUT 03a].

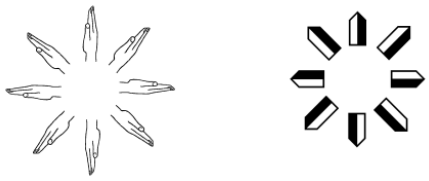


Figura 2.12 - Direções para as mãos [SUT 03a]

As configurações de mãos são organizadas em grupos de acordo com os dedos utilizados. Existem dez grupos de símbolos para as mãos, como mostra a figura 2.13, e as configurações de mãos de todas as línguas de sinais estão incluídas nesses grupos [SUT 03a]. A tabela 2.1 apresenta exemplos de sinais de cada grupo.

Figura 2.13 - Grupos de mãos [MAC 99a]

Tabela 2.1 - Exemplos de sinais de cada grupo de mão [CAP 01] e [SUT 03]

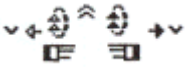
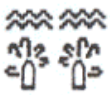
Grupo	Símbolo	Descrição	Exemplo	SignWriting
1		dedo indicador e suas variações	 computador	
2		dedos indicador e médio e suas variações	 usar	

3		dedos indicador, médio e polegar e suas variações	 enganar	
4		neste grupo utilizam-se os quatro dedos, exceto o polegar	 Brasil	
5		no grupo 5 os cinco dedos são utilizados	 árvore	
6		no grupo 6 são utilizados os três dedos centrais, ou seja, não utilizam-se o polegar e o mínimo	 terça-feira	
7		neste grupo os dedos anelar e o polegar não são utilizados	 noivo	
8		neste grupo os dedos médio e polegar não são utilizados	 só	
9		no grupo 9 grupo o dedo indicador e o polegar não são utilizados	 justiça	
10		neste grupo utiliza-se apenas o dedo polegar	 esporte	

2.4.2 Movimentos

Os movimentos podem ser classificados em movimentos de mãos e de dedos. Além disso, todo o movimento no plano horizontal é representado por uma linha simples enquanto que, movimentos no plano vertical, são representados por duas linhas [SUT 03a]. A tabela 2.2 apresenta alguns exemplos de movimento para dedos e mãos.

Tabela 2.2 - Tipos de movimento para dedos e mãos [CAP 01] e [SUT 03a]

Tipo de movimento	Símbolo	Exemplo	SignWriting
Fechar a articulação do meio do dedo 	●	 atirar	
Abrir a articulação do meio do dedo 	○	 rasgar	
Fechar a articulação do nó dos dedos 	˅	 presente	
Abrir a articulação do nó dos dedos 	^	 acordar	
Abrir e fechar a articulação do nó dos dedos 	^^	 prefeitura	
Abrir e fechar a articulação do nó dos dedos alternadamente 	^^^	 queimar-se	
Mão direita para frente (na horizontal)	↑	 andar	
Mão direita para cima (na vertical)	⇑	 convencido	

2.4.3 Contato

Existem seis formas de representar o contato dos elementos que compõem o sinal, seja mão com mão, mão com corpo ou mão com cabeça [SUT 03a]. A tabela 2.3 apresenta todos os símbolos para contato.

Tabela 2.3 - Símbolos para contato [CAP 01]

Tipo de contato	Símbolo	Exemplo	SignWriting
Tocar em outra parte do corpo	*	 surdo	
Pegar em alguma parte do corpo ou roupa	+	 camiseta	
Tocar entre duas partes do corpo, geralmente entre dois dedos	⋈	 plantar	
Bater em alguma parte do corpo	#	 bater	
Raspar em alguma parte do corpo saindo da superfície	⊙	 mês	
Esfregar em alguma parte do corpo	⊚	 conversar	

De maneira a visualizar melhor a representação escrita da língua de sinais através do SignWriting é transcrito em anexo um parágrafo escrito em sinais por Ronice Quadros em [QUA 97], com a correspondente tradução para o português.

2.5 SignWriting Markup Language – SWML

A SWML é uma linguagem padrão *eXtinsible Markup Language* (XML), pois provê um formato de troca de dados flexível e independente de software que pode ser facilmente analisado sintaticamente. Foi desenvolvida por Costa e Dimuro [COS 01a], com aprovação oficial do DAC, com o objetivo de tornar possível e viável a realização de operações de troca, armazenamento e processamento de textos escritos em língua de sinais por diversas aplicações. A interoperabilidade proposta através do uso da SWML é ilustrada na figura 2.14, a qual exemplifica a troca de textos em SignWriting entre diferentes softwares. Conforme a figura, a aplicação remetente conferte o texto a ser compartilhado para o padrão SWML e a aplicação destino recebe esse arquivo SWML convertendoo para seu padrão próprio.

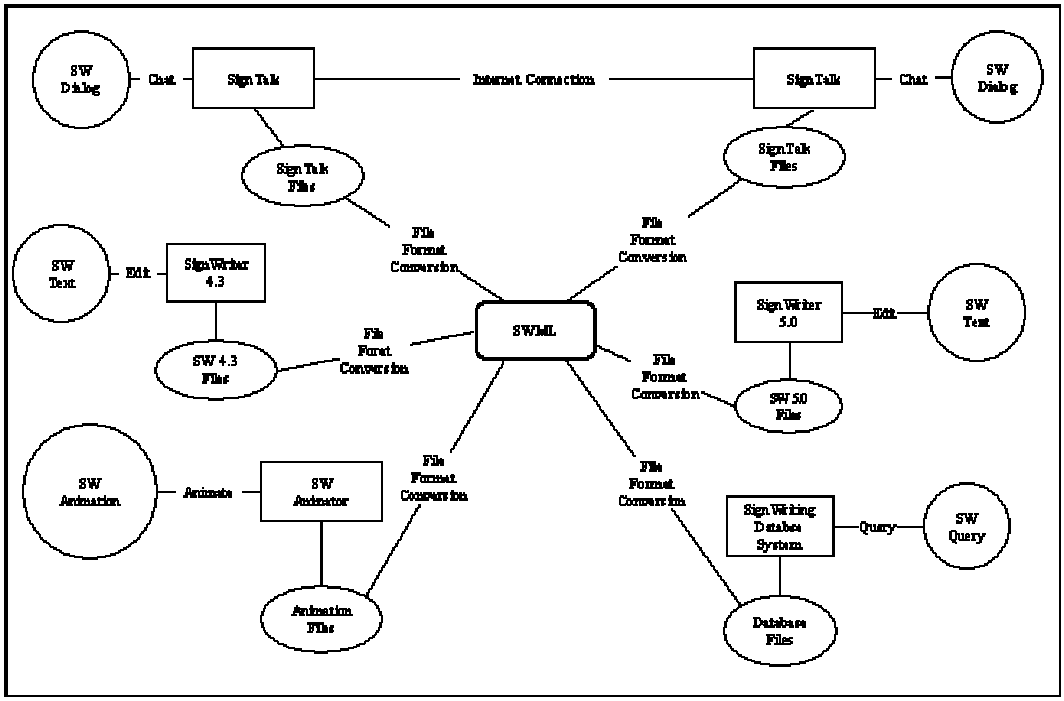


Figura 2.14 - Interoperabilidade através da SWML [COS 03]

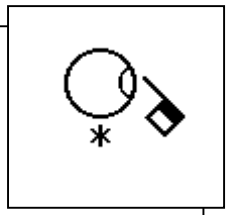
A estrutura dos arquivos SWML descreve, por intermédio de suas *tags* e respectivos atributos, todas as características necessárias para que qualquer software, apto a processar o padrão XML, reproduza com fidelidade os sinais criados com base nos símbolos do sistema SignWriting (*Sign-Symbol-Sequence*) [COS 01a]. Sua atual versão é a 1.0-draft2, definida pela DTD (*Document Type Definition)* disponível em http://swml.ucpel.tche.br/dtd-version1.0-draft2.htm.

O padrão SWML define, inicialmente, se um documento contém texto <sw_text> ou se é um dicionário <sw_dic>. Um <sw_text> é constituído por elementos <signbox> (definem os sinais) e <textbox> (contém caracteres alfanuméricos para possibilitar a inclusão de textos em línguas orais). Um <sw_dic> é constituído de elementos <entry> os quais contém elementos <signbox> e <gloss> (glosa na língua oral referente ao sinal) [COS 01b].

Basicamente, um arquivo SWML descreve quais símbolos compõem cada sinal, o posicionamento de cada símbolo no espaço e as transformações as quais os símbolos devem ser submetidos. Para cada sinal no texto ou dicionário, existe um elemento denominado <signbox> o qual concentra um grupo de símbolos <symb> que juntos compõe um sinal. Para cada símbolo <symb> em um sinal <signbox> existe um conjunto de informações definidas pelos elementos *category*, *group*, *symbnum*, *variation*, *fill* e *rotation* que identificam o símbolo base e a quais transformações ele deve ser submetido para compor determinado sinal [COS 01a].

Além disso, existe um grupo de atributos *x*, *y*, *x-flop*, *y-flop* e *colour* identificando as coordenadas de cada símbolo, se o símbolo possui rotação em x ou em y e sua cor, respectivamente. Desta forma, as características de qualquer sinal podem ser extraídas de um arquivo SWML e a imagem do sinal pode ser reconstruída sempre que necessário [COS 01b].

A figura 2.15 apresenta o sinal em Libras para a palavra *Surdo,* representado graficamente em SignWriting, e o código SWML referente ao mesmo.



```
<swml version="1.0" file-type="dic"
      xmlns="http://swml.ucpel.tche.br/swml/1.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
    <generator>
       <name>Sign WebMessage</name>
       <version>1.0</version>
    </generator>
    <doc symbol-set="sss-2002">
    <signbox>
            <symb x="35" y="15" x-flop="0" y-flop="0" colour="0,0,0">
               <category>01</category>
               <group>01</group>
               <symbnum>001</symbnum>
               <variation>01</variation>
               <fill>02</fill>
               <rotation>02</rotation>
            </symb>
            <symb x="75" y="25" x-flop="0" y-flop="0" colour="0,0,0">
                <category>03</category>
                <group>04</group>
                <symbnum>003</symbnum>
                <variation>01</variation>
                <fill>02</fill>
                <rotation>01</rotation>
            </symb>
            <symb x="50" y="55" x-flop="0" y-flop="0" colour="0,0,0">
                <category>02</category>
                <group>01</group>
                <symbnum>001</symbnum>
                <variation>01</variation>
                <fill>01</fill>
                <rotation>01</rotation>
            </symb>
    </signbox>
</swml>
```

Figura 2.15 - Código SWML que descreve o sinal referente à palavra *Surdo*.

2.6 Trabalhos relacionados

Como já mencionado na introdução desta proposta, diversas iniciativas estão sendo desenvolvidas para difundir as línguas de sinais, inclusive através do desenvolvimento de software, como forma de apoiar a inclusão digital e social das pessoas surdas. Porém, uma das dificuldades encontradas para o cumprimento deste objetivo ocorre no momento de se registrar as línguas de sinais na forma escrita e, por isso, existem poucos ambientes que as utilizam como idioma principal em suas funcionalidades e interfaces.

Alguns softwares já fazem uso do sistema para escrita das línguas de sinais SignWriting, porém a tarefa para disponibilizá-lo ainda é bastante complexa e demanda muito tempo e recursos. Não existe uma solução padrão confiável e documentada que possa ser facilmente reutilizada de forma a diminuir o tempo e a complexidade para o desenvolvimento de novos softwares que utilizem a escrita de línguas de sinais.

Atualmente cerca de 30 países utilizam o sistema SignWriting (África do Sul, Alemanha, Bélgica, Brasil, Canadá, Dinamarca, Espanha, Estados Unidos, França, entre outros) [SUT 03a]. Como subsídio para o desenvolvimento da dissertação proposta, foi realizado um estudo de algumas ferramentas computacionais direcionadas ao público surdo e que utilizam o sistema de escrita SignWriting.

2.6.1 SignWriter

O SignWriter é um editor de textos para escrita em língua de sinais, atualmente na versão 4.4, projetado pelo *Deaf Action Committe for SignWriting – DAC* e desenvolvido por

Richard Gleaves. O SignWriter não é indicado para o aprendizado da língua, visto que foi desenvolvido para pessoas que já tenham conhecimento e habilidade para se comunicar através da língua de sinais. A edição dos sinais é feita através do próprio teclado, onde cada tecla representa um grupo de variações de um símbolo. Uma outra maneira de edição é através do dicionário do sistema, em que o usuário informa a palavra e o editor retorna o sinal correspondente. Além disso, é possível imprimir, criar e abrir um novo arquivo; além de selecionar, copiar, procurar e excluir sinais já editados [SUT 01].

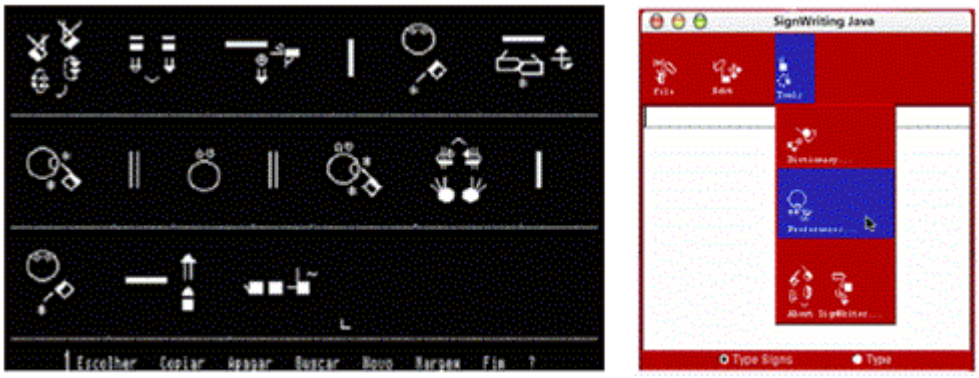


Figura 2.16 - SignWriter 4.4 e SignWriter Java [SUT 01] [SUT 03b]

Atualmente o DAC está trabalhando no desenvolvimento de uma nova versão do SignWriter, em JAVA (figura 2.16 lado direto), cujo objetivo maior é disponibilizar a mesma ferramenta para edição de textos em línguas de sinais com uma interface gráfica mais amigável do que a versão para DOS (figura 2.16 lado esquerdo). Além disso, pretende-se com esta nova versão tornar o software multi-plataforma abrangendo assim, um número maior de usuários [SUT 03b].

2.6.2 SignEd, SignSim e SignTalk

Os três softwares [CAM 01] possibilitam a utilização das línguas de sinais escritas em SignWriting. O ***SignEd*** é também um editor para escrita das línguas de sinais, desenvolvido em Delphi, no qual o usuário necessita informar as características do sinal, tais como posicionamento de mãos, dedos, movimento, expressão facial, entre outros.

O ***SignSim*** é uma ferramenta para tradução entre a Língua Brasileira de Sinais e a língua portuguesa, e vice-versa, possuindo módulos que se diferenciam pela língua em questão: o módulo de escrita de sinais, que utiliza a mesma base de dados do *SignEd*, e o módulo de escrita da língua oral, português, que possui algumas diferenças significativas de configuração para tradução. O sistema possui uma área para entrada do texto a ser traduzido e possibilita ao usuário traduzir, imprimir, salvar ou abrir um texto salvo anteriormente.

Já ***SignTalk*** (figura 2.17) é uma ferramenta de *chat* que permite comunicação síncrona entre seus usuários, tanto através da Libras quanto através do Português. Para tanto, é composto por duas aplicações: *SignTalk* servidor e cliente. O servidor é responsável por disponibilizar as salas *on-line* para o bate-papo enquanto, através do cliente, os usuários participam do *chat* escolhendo uma das dez salas de bate-papo que desejam participar, se desejam receber as mensagens em Libras, português ou ambas, bem como qual língua irão utilizar para escrever suas mensagens.

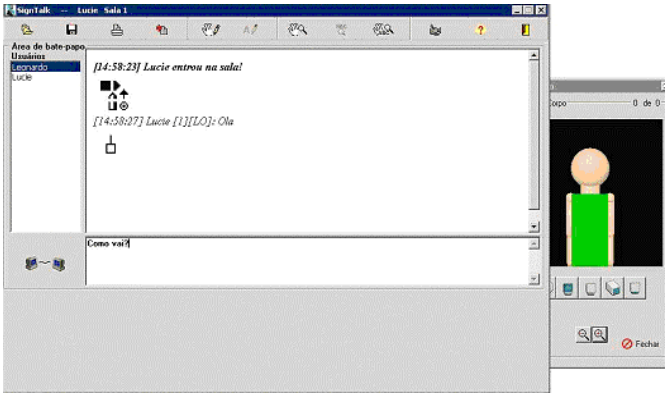


Figura 2.17 - SignTalk [CAM 01]

Os três softwares compartilham o mesmo dicionário bilíngüe de sinais e possuem interface bastante semelhante. Além disso, nestas ferramentas os usuários podem visualizar os sinais com o auxílio de um personagem 3D, além de poder gerenciar uma base particular de sinais, incluindo, alternando ou apagando-os.

2.6.3 SWEdit

O *SWEdit* é mais um editor de textos em línguas de sinais baseado no sistema SignWriting. Além disso, este software permite inclusão de textos em língua oral, figuras, *drag & drop* entre diferentes programas e salvar e carregar arquivos no formato SWML, além de disponibilizar uma base de sinais expansível [TOR 02].

A figura 2.18 apresenta a interface do editor, na qual as seguintes características podem ser observadas: (1) *tabs* contendo os conjuntos de símbolos, (2) conjunto de símbolos da *tab* atual, (3) área de edição de sinal, (4) área de edição do documento, (5) uma célula, (6) exemplo de inserção de texto da língua oral, (7) exemplo de inserção de uma figura, (8) menu sensível ao contexto e (9) *comboBox* contento os dicionário disponíveis [TOR 02].

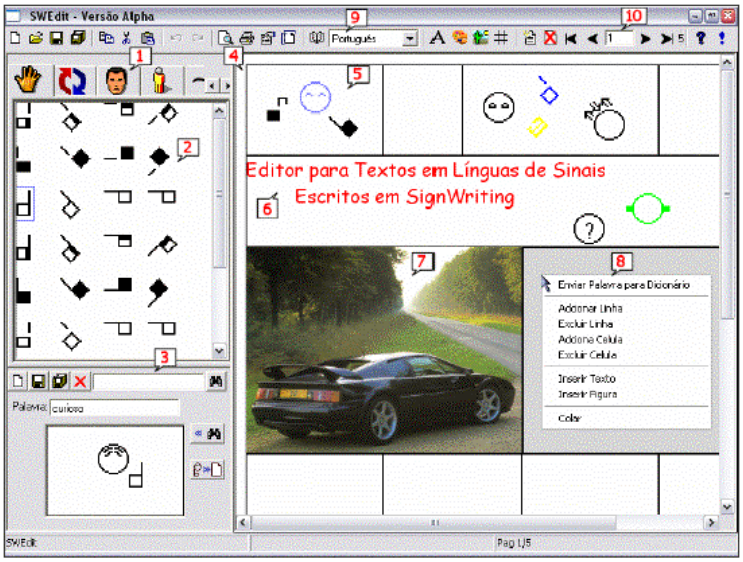


Figura 2.18 - SWEdit [TOR 02]

2.6.4 SignBank

O SignBank é um banco de dados relacional, desenvolvido em FileMaker Pro, para criação de dicionários em SignWriting ordenados pelo *Sign-Symbol-Sequence* – SSS (conjunto de símbolos disponíveis para composição dos sinais). O SignBank possui três partes [SUT 02]:

O **SignBank Edior** (figura 2.19, imagem mais à esquerda) é utilizado para criar dicionários ou modificar dicionários existentes, pois possibilita a edição de sinais, palavras, figuras, animações e vídeos.

O **SymbolBank** (figura 2.19, imagem central) armazena todos os símbolos disponíveis para criação dos sinais, os quais são organizados segundo o SSS.

Já **SignBank Portal** (figura 2.19, imagem mais à direita) é uma biblioteca de sinais através da qual os sinais podem ser consultados e impressos, mas não podem ser editados. Existem oito possibilidades de consulta ao dicionário através do portal:

1. **palavras** - o usuário digita ou seleciona uma palavra da lista e o sistema mostra o sinal correspondente;
2. **sinais –** seleciona-se um sinal e o sistema mostra a palavra correspondente em língua oral e os símbolos utilizados em sua composição;
3. **figuras -** além de visualizar a palavra em língua oral e o sinal correspondente o usuário visualiza uma figura representativa do seu significado;
4. **vídeos –** possibilita a visualização de vídeos com a execução dos sinais;
5. **animação –** apresenta animações dos sinais em SignWriting;
6. **multilíngüe –** permite aos usuários conhecerem os sinais em outras línguas
7. ***Sign-Symbol-Sequence* –** apresenta os sinais segundo a ordenação do *Sign-Symbol-Sequence;*
8. **símbolos** - o usuário escolhe um símbolo e o sistema apresenta todos os sinais que possuem tal símbolo em sua composição.

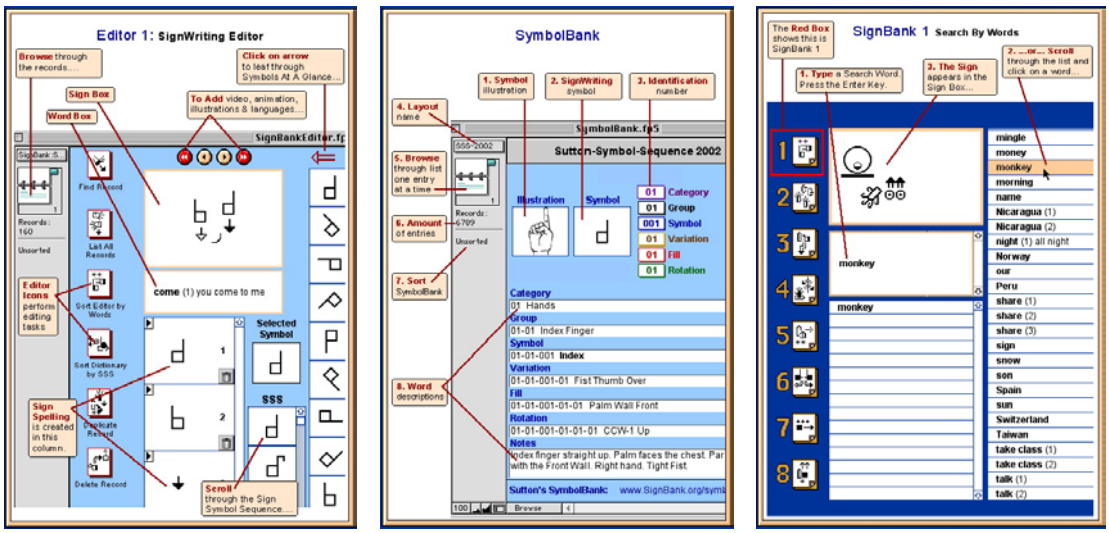


Figura 2.19 - SignBank 2002 [SUT 02]

2.6.5 Comparativo

É possível perceber pontos em comum nos sistemas descritos, bem como características que não estão presentes em todos eles, ou que se apresentam em alguns de formas diferentes. A tabela 2.4 apresenta uma análise comparativa entre os softwares estudados.

Tabela 2.4 - Estudo comparativo entre softwares que utilizam o SignWriting

Software/ Recurso	**SignWriter**	**SignEd SignSim SignTalk**	**SWEdit**	**SignBank**
Cria sinais	x	x	x	x
Edita sinais	x	x	x	x
Possui dicionário de sinais	x	x	x	x
Consulta o dicionário a partir da língua de sinais		x		x
Consulta o dicionário a partir da língua oral	x	x	x	x
Importa e exporta sinais em SWML			x	

3 Tecnologias envolvidas

3.1 Design Patterns

O constante avanço da tecnologia da informação exige um constante repensar sobre a forma de se desenvolver software. Assim, diversas metodologias são propostas a fim de aprimorar esta atividade e, neste processo, três aspectos chamam mais atenção: a tecnologia e a metodologia a ser utilizada para o desenvolvimento e a possibilidade de reutilização de softwares ou parte de softwares já desenvolvidos [RHE 02].

Pesquisas têm mostrado que altos níveis de reutilização de software podem ser alcançados através do paradigma de programação orientados a objetos (OO) e de *design patterns* (padrões de projeto) [GAM 94]. Entretanto, no desenvolvimento de software há uma grande escassez na documentação dos problemas e nas soluções encontradas para solucioná-los. Com isso, problemas que muitas vezes se repetem, geram esforços adicionais para a implementação de suas soluções [RHE 03].

O uso dos *design patterns* vem preencher esta lacuna, tornando-se um mecanismo eficiente no compartilhamento de conhecimento entre os desenvolvedores. A meta é a criação de uma linguagem comum, que permita uma comunicação efetiva no que se refere à troca de experiências sobre problemas e suas soluções. Desta forma, soluções que se aplicaram a situações particulares, podem ser novamente aplicadas em situações semelhantes. O foco principal é criar uma cultura de catalogação de experiências e soluções para apoiar o desenvolvimento de software [FRY 03].

Os *design patterns* originam-se no final dos anos 80 quando Ward Cunningham e Kent Beck desenvolveram um conjunto de padrões para serem aplicados no desenvolvimento de interfaces em Smalltalk. No mesmo período, Jim Coplien estava desenvolvendo um catálogo de padrões C++ chamados idiomas. Enquanto isso, Erich Gamma estava trabalhando em sua tese de doutorado sobre desenvolvimento de software orientado a objeto, e reconheceu a importância de acumular explicitamente as estruturas de projetos que se repetiam com freqüência [RIE 03].

Todas essas atividades foram influenciadas pelos trabalhos de Christopher Alexander, um arquiteto que associou o termo "*pattern*" às repetições de formas em arquitetura. Ele argumentava que os métodos de arquiteturas não atendiam às reais necessidades dos indivíduos e da sociedade. Por isso, *design patterns* têm sido usados nos mais diferentes domínios, abrangendo desde o desenvolvimento de software até o uso em Arquitetura e Educação [DOU 03].

Assim, quanto ao desenvolvimento de software, pode-se dizer que *design patterns* são soluções genéricas para problemas recorrentes em Engenharia de Software nas quais cada padrão identifica classes e instâncias participantes com seus papéis, colaborações e distribuição de responsabilidades, sendo que estes elementos podem ser customizados para resolver um problema num contexto particular [GAM 00].

Os *design patterns* possuem quatro elementos essenciais [GAM94]:

- **Nome** – identifica o problema e a solução adotada em uma ou duas palavras.
- **Problema** – informa em que ocasiões o padrão pode ser utilizado.

- **Solução** – descreve os elementos que compõem o projeto, seus relacionamentos, responsabilidades e colaboração. A solução não descreve uma implementação particular. Ao invés disso, fornece uma descrição abstrata de um problema e como uma combinação de classes e objetos o resolve.
- **Conseqüências -** informa os resultados e desafios na utilização do padrão a fim de esclarecer os custos e benefícios de sua aplicação.

3.1.1 Classificação

A comunidade de *design patterns* está crescendo tanto em membros quanto em abrangência e, com isso, a literatura sobre o assunto descreve novos padrões que resolvem problemas emergentes relacionados a avanços técnicos. Visto que existem muitos padrões de projeto, se faz necessário uma forma de organizá-los e eles têm sido classificados de diferentes formas por diferentes autores. Em geral, os padrões relacionados são classificados em famílias, o que auxilia o aprendizado bem como direciona esforços na descoberta de novos padrões.

Segundo Gamma [GAM 00], os padrões de projeto são classificados por dois critérios: **finalidade** (reflete o que o padrão faz) e **escopo** (especifica se o padrão se aplica primariamente a classes ou a objetos). Em relação ao primeiro critério, os padrões podem ter finalidade de criação, estrutural ou comportamental. Os padrões de criação se preocupam com o processo de criação de objetos, os estruturais lidam com a composição de classes ou de objetos e os comportamentais caracterizam as maneiras pelas quais classes ou objetos interagem e distribuem responsabilidades. O segundo critério especifica se o padrão se aplica primariamente a classes ou a objetos. Os padrões para classes lidam com os relacionamentos estáticos entre classes e subclasses, já os padrões para objetos lidam com relacionamentos entre objetos que podem ser mudados em tempo de execução e são mais dinâmicos.

Os padrões de criação voltados para classe deixam alguma parte da criação de objetos para subclasses, enquanto que os voltados para objetos postergam esse processo para outro objeto. Os padrões estruturais voltados para classes utilizam a herança para compor classes e os voltados para objetos descrevem maneiras de montar objetos. Já os padrões comportamentais voltados para classes usam herança para descrever algorítmos e fluxo de controle e os voltados para objetos descrevem como um grupo de objetos coopera para executar uma tarefa que um único objeto não pode executar.

A tabela 3.1 apresenta a classificação dos *design patterns*.

Tabela 3.1 - Classificação dos *design patterns*

		Finalidade		
		Criação	**Estrutural**	**Comportamental**
Escopo	**Classe**	*Factory Method*	*Adapter (class)*	*Interpreter* *Template Method*
	Objeto	*Abstract Factory* *Builder* *Prototype* *Singleton*	*Adapter (object)* *Bridge* *Composite* *Decorator* *Façade* *Flyweight* *Proxy*	*Chain of Responsibility* *Command* *Iterator* *Mediator* *Memento* *Observer* *State* *Strategy* *Visitor*

3.1.2 Catálogo

O catálogo de padrões de projetos em [GAM 00] possui 23 padrões, conforme a tabela 3.1, os quais são apresentados a seguir:

Padrões de criação

Classe:

- ***Factory Method* -** define uma interface para criar um objeto, mas deixa as subclasses decidirem qual classe a ser instanciada.

Objeto:

- ***Abstract Factory* -** fornece uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
- ***Builder*** – separa a construção de um objeto complexo de sua representação para que o mesmo processo de construção possa criar diferentes representações.
- ***Prototype* –** especifica os tipos de objetos a serem criados usando uma instância prototípica e permite criar novos objetos copiando este protótipo.
- ***Singleton* -** garante que uma classe tenha uma única instância e provê um ponto global de acesso a ela.

Padrões estruturais

Classe:

- ***Adapter (class)* –** converte a interface de uma classe em outra interface esperada.

Objeto:

- ***Adapter (object)*–** converte a interface de um objeto em outra interface esperada.
- ***Bridge* –** separa uma abstração da sua implementação de modo que as duas possam variar independentemente.
- ***Composite* –** compõe objetos em estrutura de árvore para representar hierarquias do tipo parte-todo.
- ***Decorator* –** atribui responsabilidades adicionais a um objeto dinamicamente.
- ***Façade* –** fornece uma interface unificada para um conjunto de interfaces.
- ***Flyweight* –** usa compartilhamento para suportar grandes quantidades de objetos.
- ***Proxy* -** provê um objeto procurador (*surrogate*), ou um marcador de outro objeto, para controlar o acesso a ele.

Padrões de comportamento

Classe:

- ***Interpreter* –** dada uma linguagem, define uma representação para a sua gramática juntamente com um interpretador que usa a representação para interpretar sentenças nesta linguagem.

- ***Template Method -*** define o esqueleto de um algorítmo numa operação, deixando que subclasses completem algumas das etapas. Esse padrão permite que subclasses redefinam determinadas etapas de um algorítmo sem alterar a estrutura do mesmo.

Objeto:

- ***Chain of Responsibility*** – evita o acoplamento do remetente de uma solicitação ao seu destinatário, dando a mais de um objeto a chance de tratar a solicitação.
- ***Command* –** encapsula uma solicitação como um objeto, desta forma permitindo que se parametrize clientes com diferentes solicitações.
- ***Iterator* –** fornece uma maneira de acessar seqüencialmente os elementos de um objeto agregado sem expor sua representação subjacente.
- ***Mediator -*** define um objeto que encapsule a forma com a qual um conjunto de objetos interage. O padrão *Mediator* promove o acoplamento fraco evitando que objetos referenciem uns aos outros explicitamente.
- ***Memento -*** sem violar o princípio de encapsulamento, captura e externaliza o estado interno de um objeto de forma a poder restaurar o objeto mais tarde.
- ***Observer -*** define uma dependência um-para-muitos entre objetos de forma a avisar e atualizar vários objetos quando o estado de um objeto muda.
- ***State -*** permite que um objeto altere seu comportamento quando seu estado interno muda. O objeto estará aparentemente mudando de classe com a mudança de estado.
- ***Strategy -*** define uma família de algorítmos, encapsula cada um, e deixe-os intercambiáveis. O padrão *Strategy* permite que o algorítmo varie independentemente dos clientes que o usam.
- ***Visitor*** - representa uma operação a ser realizada nos elementos de uma estrutura de objetos. O padrão *Visitor* permite que se defina uma nova operação sem alterar as classes dos elementos nos quais a operação age.

3.1.3 Descrição

As notações gráficas existentes para modelagem de projetos, embora importantes e úteis, não são suficientes para descrever os *design patterns*, pois apenas capturam o produto final do projeto como relacionamentos entre classes e objetos. Para reutilizar o projeto se faz necessário registrar também as decisões, alternativas e análises de custos e benefícios que resultaram do mesmo. Além disso, exemplos concretos também são importantes, pois auxiliam a visualizar o padrão sendo utilizado efetivamente. Por isso, sugere-se um formato coerente para a descrição dos padrões em que cada padrão é descrito segundo um modelo, o que resulta em uma uniformidade na estrutura das informações tornando-os mais fáceis de aprender, comparar e utilizar.

Não existe um consenso sobre como descrever um *template* de *patterns*. Alguns autores preferem ser mais expressivos e menos estruturados, enquanto outros preferem que seus *templates* sejam mais precisos e altamente estruturados. O modelo apresentando por Gamma [GAM 94] possui a seguinte estrutura:

- **Nome e classificação –** descreve sucintamente a essência do padrão.
- **Objetivo** – pequena descrição sobre o que o padrão faz, quais são seus princípios e qual problema particular do projeto ele trata.
- **Também conhecido como** – outros nomes que possam identificar o padrão.

- **Motivação** – descrição de um cenário que ilustre um problema de projeto e como as estruturas de classes e objetos do padrão solucionam tal problema.
- **Aplicabilidade –** indica as situações nas quais o padrão pode ser aplicado.
- **Estrutura** - representação gráfica das classes do padrão baseada na *Object Modeling Technique* (OMT) e diagramas de interação para ilustrar seqüências de solicitações e colaborações entre objetos.
- **Participantes –** classes e objetos participantes e suas responsabilidades.
- **Colaborações -** descreve a colaboração necessária entre os participantes para que executem suas responsabilidades.
- **Conseqüências –** custos, benefícios e resultados da utilização do padrão.
- **Implementação –** dicas, técnicas e sugestões para a implementação do padrão.
- **Exemplo de código –** trechos de código que ilustram como o padrão pode ser implementado.
- **Usos conhecidos –** exemplos da utilização do padrão em sistemas reais.
- **Padrões relacionados –** quais outros padrões estão relacionados com este que está sendo descrito, quais as diferenças importantes entre eles e com quais outros padrões este deveria ser utilizado.

A tabela 3.2 apresenta um exemplo da descrição parcial do padrão estrutural de objetos *Composite.*

Tabela 3.2 - Exemplo da descrição parcial do padrão *Composite* [GAM 00]

COMPOSITE	**Estrutural de objetos**

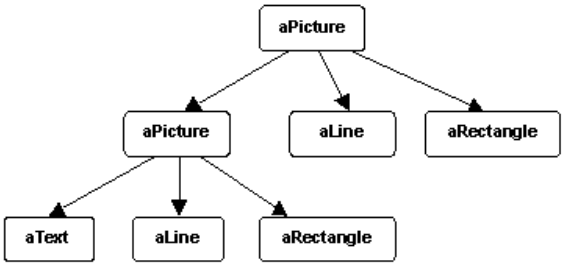
Intenção

Compor objetos em estruturas de árvore para representarem hierarquias partes-todo, permitindo aos clientes tratarem de maneira uniforme objetos individualmente e composições de objetos.

Motivação

Aplicações gráficas, pois permite construir diagramas complexos a partir de componentes simples. Porém, há um problema com esta abordagem: o código deve tratar objetos primitivos e objetos-recipientes de modo diferente, mesmo se na maior parte do tempo o usuário os trata de forma idêntica. O padrão *Composite* descreve como utilizar a composição recursiva de maneira que os clientes não tenham que fazer esta distinção.

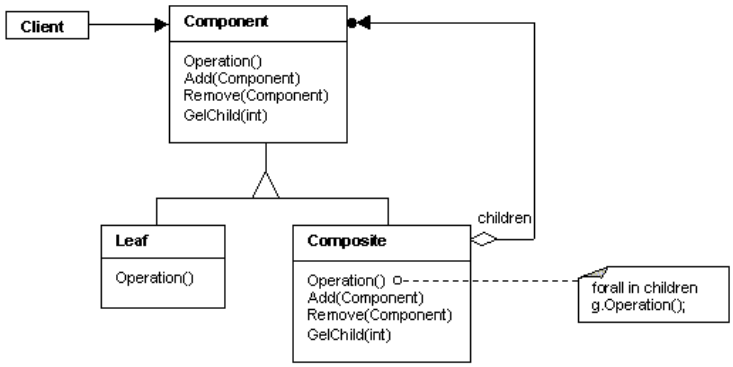
A chave para este padrão é uma classe abstrata que representa tanto as primitivas como os seus recipientes. O diagrama a seguir mostra uma típica estrutura de objeto composto, composta recursivamente por objetos gráficos:



Aplicabilidade

Use o padrão *Composite* quando quiser representar hierarquias partes-todo de objetos ou quando quiser que os clientes sejam capazes de ignorar a diferença entre composições e objetos individuais.

Estrutura



Colaborações

Os clientes usam a interface da classe *Component* para interagir com os objetos na estrutura composta. Se o receptor é uma *leaf* (folha), então a solicitação é tratada diretamente. Caso o receptor seja um *Composite* ele repassa normalmente as solicitações para os seus componetes-filhos.

Padrões relacionados

Chain of Responsability, *Decorator*, *Flyweight*, *Iterator* e *Visitor.*

3.2 Web Services

Com o crescimento da Internet surgem novas tecnologias para disponibilização de serviços e, com isso, a necessidade de tornar estes serviços interoperáveis e reutilizáveis. Aplicações baseadas na web precisam estar aptas a encontrar, acessar e interagir automaticamente com outras aplicações e isso se tornou possível através dos Web Services,

pois eles apresentam uma estrutura arquitetural que permite a comunicação efetiva entre aplicações [HAN 03b].

Os Web Services têm gerado grande excitação na indústria global da computação e com isso aproximadamente 3.300 projetos baseados em *Web Services* foram desenvolvidos na América do Norte em 2002. Assim, estima-se que os gastos com hardware, software e serviços relacionados a *Web Services* irão chegar a 15,2 milhões de dólares até 2007 [CHU 03].

Web Services são componentes de software que independem de implementação ou de plataforma e podem ser descritos, publicados e invocados sobre uma rede, geralmente a web, através de mensagens padrão XML [PIN 02]. Com o uso de Web Services é possível se estabelecer integração entre aplicações baseadas na Internet, de forma padronizada, utilizando-se para isso padrões abertos incluindo XML, *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL) e *Universal Description, Discovery e Integration specification* (UDDI). As mensagens são estruturadas com XML, o SOAP encarrega-se de transferir as mensagens, o WSDL descreve os serviços disponíveis e o UDDI lista os mesmos [CHU 03].

Em suma, Web Services são interfaces que descrevem uma coleção de operações que são acessíveis pela rede através de mensagens em formato XML padronizadas, que permitem uma integração de serviços de maneira rápida e eficiente. São descritos através de uma linguagem de descrição de serviços, publicados em um registro e descobertos através de um mecanismo padrão [HAN 03b].

3.2.1 Arquitetura

A arquitetura Web Services, apresentada na figura 3.1, é baseada em um Provedor de serviços, um Solicitante de serviços e um Registro de serviços. O Provedor é responsável por disponibilizar os serviços e armazenar sua descrição, através da WSDL, a qual contêm detalhes de interface, operações e mensagens de entrada e saída. Após o recebimento da descrição de um serviço, o provedor publica a descrição do mesmo, em WSDL, em um Registro de serviços. O Solicitante é uma aplicação que invoca uma interação com o serviço, a qual pode ser um navegador web ou outra aplicação qualquer como um outro Web Service. O Registro é o local onde os Servidores publicam seus serviços e onde os Solicitantes procuram por serviços [FER 03b].

Na arquitetura Web Service, a descrição de um serviço cobre todos os detalhes necessários para garantir sua utilização, incluindo o formato das mensagens, protocolos de transporte e localização.

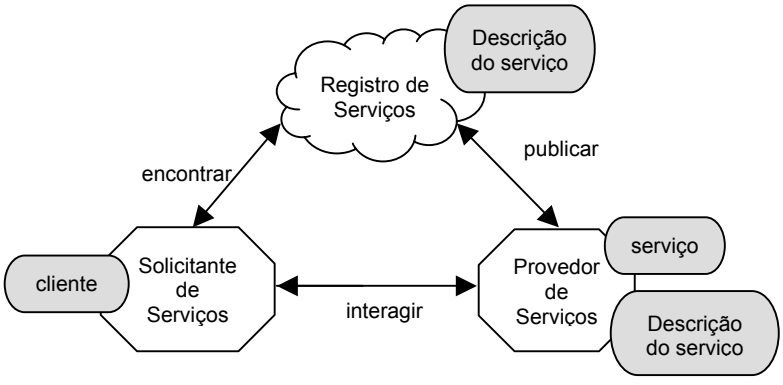


Figura 3.1 - Arquitetura dos Web Services [FER 03a]

3.2.2 Tecnologia

A tecnologia utilizada em Web Services permite que serviços possam ser disponibilizados na web de forma padronizada. Esta padronização permite a localização e acesso aos serviços de forma bastante eficiente, já que as tecnologias empregadas estão baseadas em XML. Além disso, a arquitetura de Web Services envolve muitas camadas de tecnologias que se inter-relacionam e, por isso, existem várias formas de se construir e utilizar Web Services [HAN 03b]. A figura 3.2 apresenta uma ilustração com algumas dessas tecnologias.

Figura 3.2 - Tecnologias de Web Services [BOO 03]

Primeiramente, cabe salientar que a XML é a base de toda a arquitetura, pois é a chave fundamental para a essência de Web Services que é a interoperabilidade. Isto conduz ao segundo conceito chave na *Web Service Architecture* (WSA): os serviços são invocados e fornecem resultados através das mensagens que devem ser trocadas sobre algum protocolo de comunicação [BOO 03].

A WSA suporta uma grande variedade de mecanismos para estabelecer comunicação tais como o HTTP e outros protocolos da Internet como o SMTP e FTP, entre outros. Por isso, não apresenta nenhuma definição sobre a camada de comunicação, permitindo que as mensagens dos Web Services sejam transmitidas por protocolos projetados para outras finalidades [NEW 02]. Já o SOAP é a tecnologia chave das mensagens na WSA, pois é a maneira padrão de empacotar as informações das mensagens que são transmitidas em formato XML [HAN 03b].

Para promover interoperabilidade entre sistemas heterogêneos é necessário um mecanismo que permita que a estrutura e o tipo de dados possam ser compreendidos pelos *Web Services*. A WSDL é utilizada com este objetivo, pois permite que mensagens com a descrição precisa dos serviços possam ser trocadas [BOO 03].

Para que um serviço possa ser utilizado, ele precisa ser publicado e posteriormente descoberto por quem desejar utilizá-lo. O registro UDDI é utilizado para publicação e descoberta de informações sobre Web Services [PIN 02].

Além das tecnologias específicas para a troca de mensagens e descrição, a arquitetura prevê também tecnologias para prover segurança e gerência [BOO 03].

Como já mencionado anteriormente, as tecnologias padrão utilizadas para estruturação de Web Services são XML, SOAP, WSDL e UDDI. Nas sessões seguintes essas tecnologias serão abordadas em maior profundidade.

3.2.2.1 *eXtensible Markup Language* – XML

No contexto de Web Services, a XML não é apenas utilizada como formato para troca de mensagens, mas também como a forma através da qual os serviços são definidos. Por isso, é importante conhecer XML especialmente no contexto de como ela é utilizada para definir e implementar Web Services [HAN 03a].

A XML foi desenvolvida para superar as limitações do HTML, especialmente para dar suporte à criação e ao gerenciamento de conteúdos dinâmicos. Através da XML pode-se criar qualquer número de elementos (*tags*) com significado associado às informações, ou seja, pode-se descrever as informações e o que fazer com as mesmas utilizando para isso um ou mais elementos criados para este propósito [NEW 02].

Além disso, é possível associar os documentos XML a XML esquemas a fim de validar as informações separadamente e descrever outros atributos e características dos dados, o que com HTML é impossível. Entretanto, é evidente que problemas podem surgir justamente devido a grande flexibilidade oferecida pela XML. Como a XML possibilita a criação de infinitos elementos, torna-se muito difícil que todos utilizem os mesmos elementos da mesma maneira, como é o caso do HTML.

Duas partes só podem trocar informações em XML e entender os elementos da mesma forma se compartilharem uma mesma definição sobre quais e como os elementos podem ser utilizados. Se duas partes compartilharem o mesmo documento XML e o mesmo XML esquema, pode-se ter certeza que ambas entenderão os elementos da mesma forma e é assim que os Web Services funcionam [NEW 02].

A XML é, na verdade, uma família de tecnologias: uma linguagem de marcação de dados, vários modelos de conteúdo, um modelo de ligação, um modelo *namespace* e vários mecanismos de transformação. A seguir são apresentados os membros significantes da família XML utilizados como base dos Web Services [NEW 02]:

- **XML v1.0:** as regras para definição de elementos, atributos e *tags* incluídas dentro do elemento raiz fornecem um modelo abstrato de dados;
- **XML esquema:** documentos XML que definem os tipos de dados, conteúdo, estrutura e elementos permitidos em um documento XML associado. Também são usados para descrever instruções de processamento semântico;
- **XML *namespace:*** nomes para documentos de elementos e aplicações XML;
- **XML *Information Set*:** representação abstrata das partes do documento XML;
- **XPointer:** um ponteiro para uma parte específica de um documento;
- **XPath:** expressões para pesquisa de documentos XML;
- **XLink:** para pesquisa de múltiplos documentos XML;
- ***eXtensible Stylesheet Language Transformations* (XSLT):** utilizado para transformar documentos XML em outros formatos de documentos XML ou não;
- ***Document Object Model* (DOM):** biblioteca de funções que permite analisar sintaticamente arquivos padrão XML através da criação, na memória, de uma árvore de objetos passível de ser percorrida;
- ***Simple API for XML* (SAX):** também é uma biblioteca para análise sintática de arquivos XML, porém a análise é realizada uma única vez no documento, de cima para baixo.

3.2.2.2 Web Service Description Language – WSDL

A WSDL é uma linguagem padrão XML utilizada para descrever interfaces de Web Services. É o coração da arquitetura Web Services e fornece um formato padrão para representar os tipos de dados passados nas mensagens, interface descrevendo as funções disponíveis, informações sobre o protocolo de transporte a ser utilizado e informações sobre endereços dos serviços na rede [NEW 02 e CER 02].

A WSDL foi projetada para ser analisada como qualquer outro documento XML e assim como as demais tecnologias XML, a WSDL é altamente flexível e extensível. Assim, se o remetente e o destinatário da mensagem puderem compartilhar e entender arquivos WSDL da mesma forma, então a interoperabilidade pode ser assegurada.

A WSDL é dividida em três elementos principais [NEW 02]: definições de tipo de dados, operações abstratas e protocolos de ligação. Cada um desses elementos pode ser especificado em documentos XML diferentes e importados em diferentes combinações, para criar a descrição final de um Web Service, ou definidos juntos em um único arquivo XML. A definição de tipo de dados determina a estrutura e o conteúdo das mensagens. As operações abstratas determinam as operações possíveis e o protocolo de ligação determina a forma de transmissão das mensagens pela rede até os destinatários [NEW 02].

O uso da SWDL permite a divisão da descrição dos serviços básicos em duas partes (interface e implementação do serviço) como mostra a figura 3.3. Esta divisão permite que estas partes possam ser reutilizadas separadamente [PIN 02].

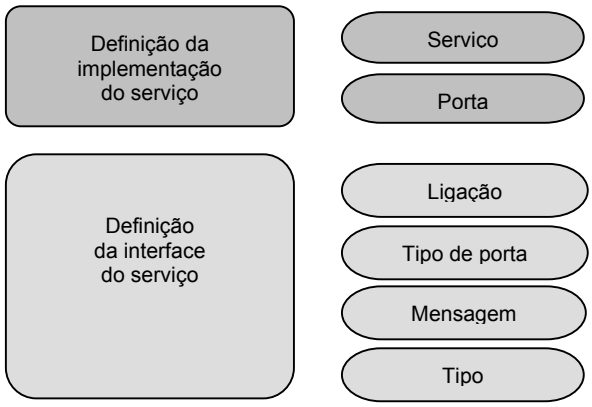


Figura 3.3 - Camada de descrição dos serviços [PIN 02].

A seguir é apresentada uma descrição de cada elemento da camada de descrição de serviços, conforme apresentado em Hansen [HAN 03a].

A **Camada de Definição da Interface do Serviço** contém a definição de serviço WSDL a qual permite que uma interface possa ser utilizada, instanciada e referenciada por múltiplas definições de implementação de serviços. Um arquivo de interface descreve o Web Service incluindo os métodos que podem ser chamados, os parâmetros que são passados e a codificação utilizada.

A **ligação** descreve o protocolo, o formato dos dados, a segurança e outros atributos para a interface de um serviço particular. No **tipo de porta** os elementos das operações do Web Service são definidos. A **mensagem** é utilizada para definir os parâmetros de entrada e saída de dados de uma operação. O **tipo** define o uso de tipos de dados complexos dentro de uma mensagem.

A **Camada de Definição da Implementação do Serviço** é definida através de um documento WSDL que descreve como uma interface de serviço é implementada por um

provedor de serviço. Um arquivo de implementação descreve onde o Web Service está instalado e como este é acessado. Além das definições de interface e implementação de serviço, a WSDL especifica extensões para ligação com protocolos e formatos de mensagem, como SOAP, HTTP e MIME.

Para apresentar o conceito da WSDL da forma mais clara possível, será apresentado um exemplo, definido em Cerami [CER 02], de um documento WSDL (HelloService.wsdl). O documento descreve um serviço com apenas uma função pública, chamada sayHello. A função recebe como parâmetro uma *string* e retorna, também, uma *string*. Por exemplo, se passarmos o parâmetro "world", o serviço retorna "Hello, world!".

HelloService.swdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
   targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>

   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name="Hello_Binding" type="tns:Hello_PortType">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="sayHello">
         <soap:operation soapAction="sayHello"/>
         <input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:helloservice"
               use="encoded"/>
         </input>
         <output>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:helloservice"
               use="encoded"/>
         </output>
      </operation>
   </binding>

   <service name="Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding="tns:Hello_Binding" name="Hello_Port">
         <soap:address
            location="http://localhost:8080/soap/servlet/rpcrouter"/>
      </port>
    </service>
</definitions>
```

3.2.2.3 Simple Object Acesses Protocol – SOAP

O SOAP é um protocolo para troca de informações em ambiente descentralizado e distribuído que permite comunicação entre aplicações de forma simples, fácil de ser desenvolvida e completamente independente de sistema operacional, linguagem de programação ou plataforma [NEW 02].

A comunicação é realizada através de trocas de mensagens, transmitidas em formato XML, incluindo parâmetros usados na chamada, bem como os dados de resultados. Isto significa que as mensagens podem ser entendidas por quase todas as plataformas de hardware, sistemas operacionais, linguagens de programação ou hardware de rede. Também, pode ser utilizado para invocar, publicar e localizar Web Services no registro UDDI [HAN 03a].

O SOAP pode ser utilizado em combinação com uma variedade de outros protocolos, como HTTP, SMTP, FTP, entre outros, e suporta *Remote Procedure Call* (RPC). Além disso, o modelo de dados SOAP oferece definições para tipos de dados como *string*, *integer*, *float*, *double* e *date*. O processo de traduzir os dados, parâmetros e resultados, em XML, é chamado de codificação [PIN 02].

Um pacote SOAP possui as seguintes partes [NEW 02]:

- **Envelope:** define o início e o fim das mensagens, quem poderá tratá-las e se o tratamento é obrigatório ou opcional.
- **Cabeçalho:** contém atributos opcionais das mensagens **.**
- **Corpo:** contém os dados em XML.
- **Anexo:** consiste de um ou mais documentos anexados a mensagem principal.
- **RPC:** define como o modelo RPC (*Remote Procedure call*) interage com o SOAP, com o objetivo de invocar procedimentos em um sistema remoto.
- **Codificação:** define como representar dados simples e complexos a serem transmitidos nas mensagens.

A figura 3.4 apresenta as três partes principais das mensagens SOAP: envelope, cabeçalho e corpo. O envelope é obrigatório e marca o início e o fim das mensagens. O cabeçalho é opcional e pode conter um ou mais blocos com atributos da mensagem. O corpo também é obrigatório e contém um ou mais blocos contento a mensagem propriamente dita [NEW 02].

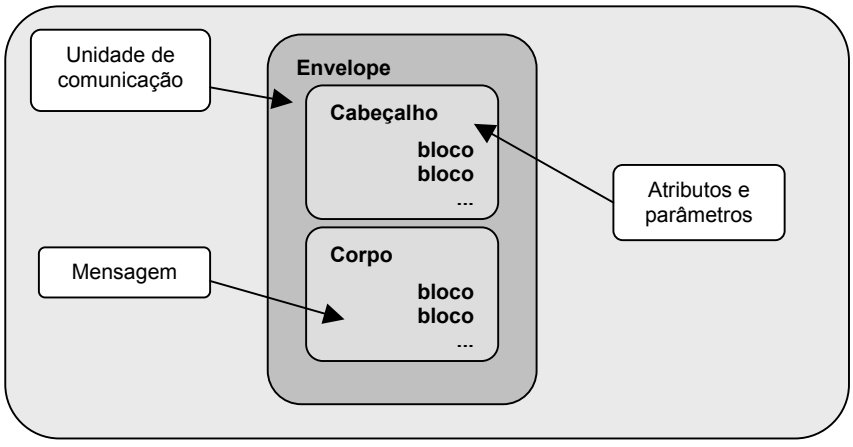


Figura 3.4 - As três partes principais das mensagens SOAP [NEW 02]

O SOAP não define o serviço propriamente, mas apenas o suficiente para que o processador SOAP possa reconhecê-lo. A implementação do Web Service é que precisa saber como interpretar os dados contidos nas mensagens SOAP. Um dos aspectos mais importantes para o entendimento do SOAP é que ele é definido com um nível de abstração suficiente para abranger tanto documentos como interações RPC [NEW 02].

A invocação do serviço ocorre conforme apresentado na figura 3.5.

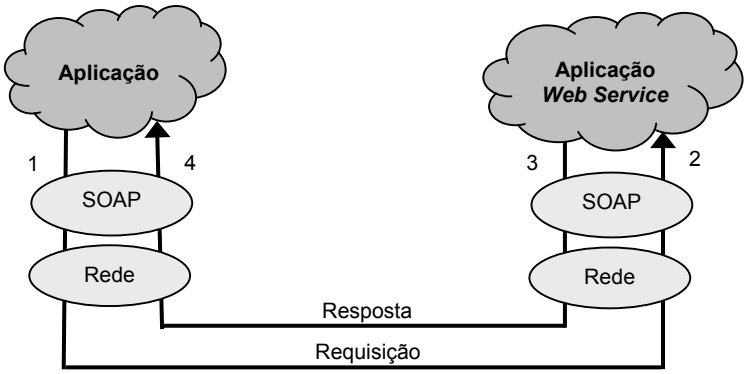


Figura 3.5 - Invocação do serviço utilizando SOAP [HAN 03a]

A aplicação (1) requisita uma mensagem SOAP e invoca a operação do serviço através de um provedor de Web Services. O solicitante de serviço apresenta a mensagem junto com o endereço de rede do provedor de Web Service. A infra-estrutura de rede (2) entrega a mensagem para um servidor SOAP. O servidor SOAP redireciona a mensagem requisitada para o Provedor de serviço Web Service. O servidor web (3) é responsável por processar uma mensagem de requisição e formular resposta. Quando a mensagem XML chega no nodo requisitante, é convertida para uma linguagem de programação, sendo então entregue para a aplicação (4) [HAN 03a].

A seguir, é apresentado um exemplo de uma requisição e de uma resposta SOAP. Neste exemplo, uma requisição de verificação de preço é enviada ao serviço de informação de preços de uma loja. O envelope SOAP é o elemento raiz do documento XML. *Namespaces* são utilizados para desambiguar identificadores SOAP de identificadores específicos da aplicação. O protocolo de rede utilizado para transmissão da mensagem SOAP, neste exemplo, é o HTTP [WAL 02].

Mensagem de requisição SOAP (HTTP)

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>DIS</symbol>
        </m:GetLastTradePrice>
    </SOAP-ENV:Body>
<SOAP-ENV:Envelope>
```

Mensagem de resposta SOAP (HTTP)

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:GetLastTradePriceResponse xmlns:m="Some-URI">
            <Price>34.5</Price>
        </m:GetLastTradePriceResponse>
    </SOAP-ENV:Body>
<SOAP-ENV:Envelope>
```

3.2.2.4 Universal Description, Discovery and Integration – UDDI

Para que um serviço seja utilizado é necessário que o cliente consiga localizá-lo, e esta localização pode ser feita através do UDDI, que é uma especificação técnica para descrever, descobrir e integrar Web Services [NEW 02] e [CER 02].

Seu componente central, chamado UDDI *project,* manipula um registro global público, chamado UDDI *business registry,* no qual toda a informação está disponível para consultas em geral. Entretanto, um registro privado pode adicionar controle de segurança para proteger os dados e prevenir acessos não autorizados [HAN 03a].

O UDDI consiste de duas partes: uma especificação técnica para construir e distribuir Web Services, através da qual as informações são armazenadas em um formato XML específico e a segunda parte, o UDDI *Business Registry* (também chamado de UDDI "*cloud services*") que é uma implementação operacional completa da especificação UDDI [NEW 02].

Os dados UDDI são divididos em três categorias principais [CER 02]:

- **Páginas brancas (*White pages*):** nesta categoria são incluídas informações gerais sobre uma empresa específica. Por exemplo: nome, descrição, contato, endereço e os números de telefone.
- **Páginas amarelas (*Yellow pages*):** inclui dados de classificação gerais da empresa ou serviço oferecido. Por exemplo, estes dados podem incluir o ramo da empresa, seus produtos.
- **Páginas verdes (*Green pages*):** esta categoria contém informações técnicas sobre Web Services. Geralmente, possui um ponteiro para uma especificação externa e um endereço para invocar o Web Service.

Um registro UDDI contém informação categorizada sobre negócios, serviços que eles oferecem e associações destes serviços com especificações dos Web Services. Essas especificações são feitas em WSDL através de um registro UDDI [HAN 03a].

Para registrar os serviços utiliza-se SOAP com UDDI e após os interessados utilizam consultas API para pesquisar informações registradas. Uma consulta inicial pode retornar muitos resultados dos quais apenas um é escolhido e uma API final é feita para obter informações e o contato de um serviço específico [NEW 02].

A figura 3.6 ilustra como pode-se registrar informações de um Web Service com o registro UDDI.

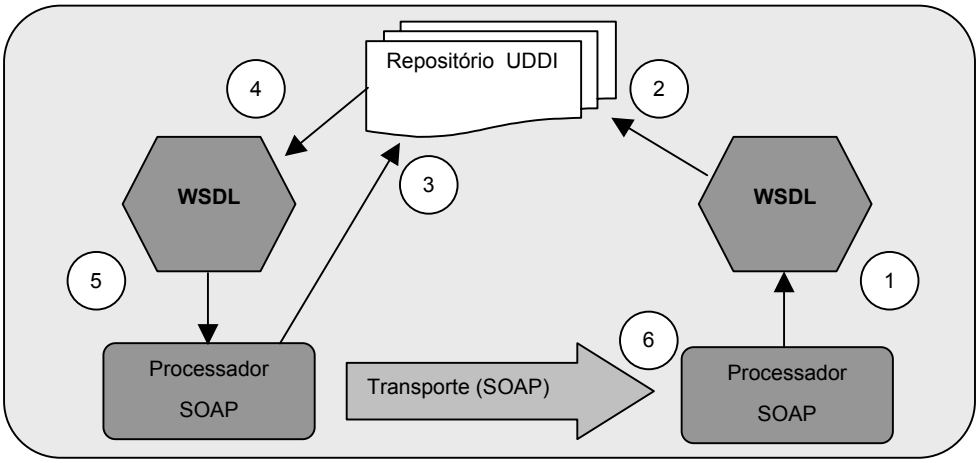


Figura 3.6 - UDDI utilizado para descobrir um Web Service [NEW 02]

Primeiramente, gera-se o arquivo WSDL para descrever o Web Service com suporte do Processador SOAP (1) e utiliza-se a API UDDI para registrar as informações no repositório (2). Os dados são transmitidos junto com informações sobre contato e o registro possui uma entrada com uma URL que aponta para o Servidor SOAP com a localização do WSDL ou outro documento com a descrição do Web Service. Um outro processador SOAP requisita o registro (3) para obter o WSDL (4). Após, o cliente gera a mensagem apropriada (5) para enviar a uma operação específica através de determinado protocolo (6). Cabe salientar que o cliente e o servidor devem estabelecer o mesmo protocolo, nesse exemplo SOAP sobre HTTP, e compartilhar a mesma semântica para a definição do serviço, a qual neste exemplo é definida através da WSDL [NEW 02].

A informação referente ao registro de um serviço consiste de quatro tipos de estrutura de dados: *businessEntity, businessService, bindingTemplate* e *tModel* [CER 02] e [WAL 02]. Esta divisão por tipos de informação fornece partições simples para auxiliar na rápida localização e compreensão das diferentes informações que compõem o registro. As quatro estruturas são apresentadas na figura 3.7.

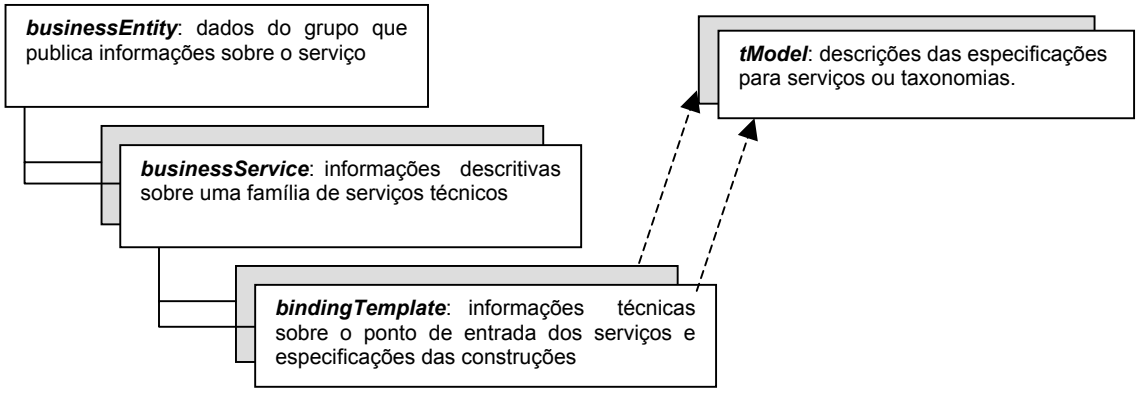


Figura 3.7 - Modelo de informações UDDI [WAL 02] e [CER 02]

O ***businessEntity*** fornece informação sobre um negócio (nome, descrição, endereço e informações para contato), podendo conter um ou mais *businessServices.* Representa toda a informação conhecida sobre um negócio ou informações descritivas sobre uma entidade, bem como os serviços que ela fornece.

O ***businessService*** define descrições técnicas e de negócios (nome, descrição e uma lista opcional de *bindingTemplates*) para um Web Service ou um para grupo de Web Services

relacionados. Representa uma classificação lógica do serviço. O nome do elemento inclui o termo "*business*" numa tentativa de descrever o propósito deste nível na hierarquia de descrição de serviços. Assim, cada estrutura *bussinessService* é filho lógico de uma única estrutura *businessEntity*

BindingTemplate são descrições técnicas de como e onde acessar um Web Service específico. Essas estruturas fornecem suporte para acessar remotamente os serviços. Também são definidos o suporte a tecnologias, parâmetros específicos da aplicação e os arquivos de configuração. Não é utilizado apenas com serviços baseados em HTTP, mas também com baseados em e-mail, fax, FTP, entre outros.

A estrutura ***tModel*** é representada por meio de metadados (dados sobre dados). O propósito de um *tModel* dentro de um registro UDDI é oferecer um sistema de referência para os documentos WSDL.

Caba salientar que muitas organizações não são efetivamente representados por um único *businessEntity* e como conseqüência diversas estruturas *businessEntity* podem ser publicadas. Mesmo assim, elas continuam representando um agrupamento e poderiam necessitar que alguns de seus relacionamentos fossem visíveis em seus registros UDDI [HAN 03a]. Assim, quando dois negócios estão relacionados utiliza-se mensagens ***publisherAssertion***, como forma de publicar declarações a respeito das relações existentes entre eles. Ambos devem publicar exatamente a mesma informação, para que o relacionamento torne-se visível [NE0W 02].

3.3 Desenvolvimento e uso de Web Services com PHP

Atualmente os Web Services são suportados pela maioria das plataformas tecnológicas incluindo IBM, Microsoft, Sun Microsystems, dentre outras e, assim, grandes investimentos financeiros têm sido aplicados no desenvolvimento desta tecnologia [ALS 04].

Existem muitas linguagens que suportam Web Services e PHP é uma delas, a qual possui uma variedade de funções e ferramentas livres para cumprir este objetivo. Quanto às possibilidades para desenvolvimento e uso de Web Services com PHP, pode-se citar pelo menos cinco formas diferentes: XML-RPC, NuSOAP, PEAR, REST e SOAP *Extension* [CAR 03].

O XML-RPC é uma implementação de RPC (*Remote Procedure Calls*) que possibilita o transporte de dados em XML entre dois servidores utilizando o protocolo http [FUE 03]. O NuSOAP (http://sourceforge.net/projects/nusoap) e o PEAR SOAP *Client/Server for* PHP (http://pear.php.net/package/SOAP), assim como o XML-RPC, são bibliotecas de classes que possibilitam a criação e uso de Web Services. Porém, são baseadas no uso do protocolo SOAP [NIC 03] [RUB 04] .

O REST (*Representational State Transfer*) fornece um processo mais simples do que via XML-RPC e SOAP, através do uso de métodos padrão HTTP como o GET, POST e PUT para enviar e recuperar dados XML. Entretanto, não existem classes pré-construídas para facilitar o seu uso [TRA 03].

Já o SOAP *Extension*, trata-se de uma nova extensão disponível a partir da versão 5 do PHP, lançada recentemente, que tem como objetivo possibilitar o uso de Web Services através de funções nativas do PHP [ZEN 04].

3.3.1 XML-RPC

XML-RPC (http://www.xmlrpc.com) é uma forma simples e eficiente de se transferir dados em XML, a qual tem sido implementada em projetos como o FreeNET, O'Reilly's Meerkat, Syndic8, the Google API e em inúmeras outras aplicações [LAU 01a].

Remote Procedure Calls – RPC são utilizadas para estabelecer e facilitar transações entre dois sistemas remotos. Assim, XML-RPC é uma implementação de RPC que possibilita o transporte de dados em XML entre dois servidores utilizando o protocolo http [FUE 03].

Para facilitar o desenvolvimento e uso de Web Services em PHP, através do XML-RPC, foram desenvolvidas e disponibilizadas algumas classes batizadas como XML-RPC *tollkit* (*xmlrpc_client, xmlrpc_server, xmlrpcmsg, xmlrpcval, xmlrpcresp*). Assim, para utilizar estas classes com o PHP é necessário baixar o *tollkit* que está disponível em: http://sourceforge.net/projects/phpxmlrpc [LAU 01n]

A figura 3.8 apresenta as classes, atributos e métodos do XML-RPC *toolkit*

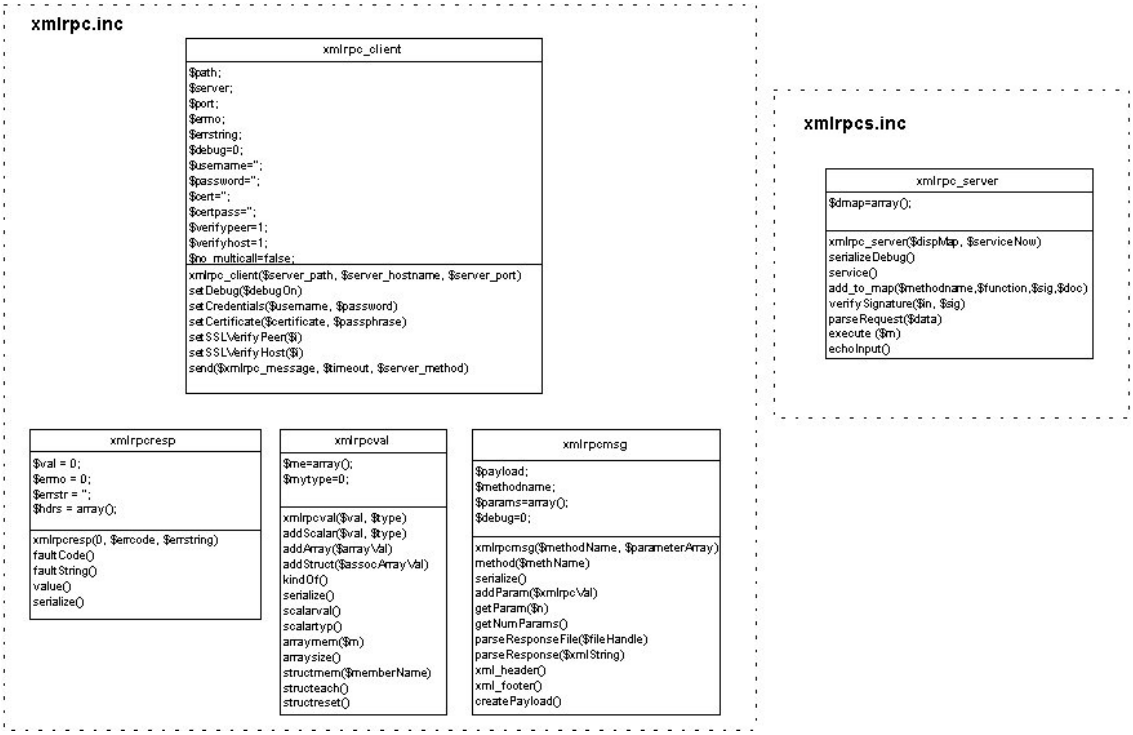


Figura 3.8 – Classes, atributos e métodos do XML-RPC *tollkit*

A seguir, é apresentada uma implementação de um Web Service simples, **xmlrpc_server.php**, utilizando XML-RPC.

Os arquivos com as classes necessárias para implementação são: xmlrpc.inc e xmlrpcs.inc. Após a inclusão das bibliotecas, definimos uma função chamada CalculaICMS. Essa função será um Web Service que tem como objetivo calcular o ICMS (Imposto sobre Circulação de Mercadorias e Prestação de Serviços).

A função recebe como parâmetro um valor em Reais, o qual é convertido para uma variável escalar. Após o cálculo do imposto, uma resposta é criada, utilizando a classe xmlrpcresp, para retornar o valor do ICMS. Então, deve-se instanciar o servidor e fornecer a função ICMS de volta ao cliente.

xmlrpc_server.php

```php
<?php
        // inclusão do arquivo de classes xmlrpc.inc
        include ("xmlrpc.inc");
        // inclusão do arquivo de classes xmlrpcs.inc
        include ("xmlrpcs.inc");
        // definição do método como uma função do PHP
        // retorno através da classe xmlrpcresp
        function CalculaICMS ($objeto){
                $valor = $objeto->GetParam(0);
                $valor_escalar = $valor->scalarval();
                $valorICMS = $valor_escalar * 1.5;
                return new xmlrpcresp (new xmlrpcval($valorICMS, "string"));
        }
        // criação da instância da classe xmlrpc_server
        $servidor = new xmlrpc_server
          (array("valorICMS.CalculaICMS"=>array("function"=>"CalculaICMS")));
 ?>
```

A seguir, é apresentando o programa que utilizará o Web Service criado anteriormente. Primeiramente, é necessário incluir a biblioteca de classes xmlrpc.inc. Após, deve-se instanciar o cliente XML-RPC o qual irá se conectar ao servidor. O valor da variável $valor será passado ao servidor através do objeto xmlrpcmsg.

Sendo a conexão estabelecida, a requisição é enviada ao servidor. A resposta, correspondente ao cálculo do ICMS sobre o valor, é passada para a variável $resp. Esta variável é então convertida para uma variável escalar e retornada para o usuário.

xmlrpc_client.php

```php
<?php
        // inclusão do arquivo de classes xmlrpc.inc
        include ("xmlrpc.inc");
        // definição de um valor para teste do web service
        $valor_nf = "15.000";
        // criação de uma instância da classe xmlrpcmsg
        // o objeto criado define a mensagem a ser passada ao Web Service
        // na mensagem é passado o nome da função a ser executada e os parâmetros
        $format = new xmlrpcmsg('valorICMS.CalculaICMS',
                        array(new xmlrpcval($valor_nf, "double")));
        // criação de uma instância do cliente
        // os parâmetros são o nome do programa, a URL e a porta para acesso
        $client = new xmlrpc_client("xmlrpc_server.php", "localhost", 80);
        // chamada do método para envio da mensagem de requisição ao Web Service
        $request = $client->send($format);
        // chamada do método para retorno da mensagem de resposta do Web Service
        $resp = $request->value();
        // exibição da resposta fornecida pelo Web Service
        echo $value->scalarval();
?>
```

3.3.2 NuSOAP

O NuSOAP é uma API desenvolvida em PHP, através da qual é possível construir clientes e servidores para Web Services. Um dos grandes recursos do NuSOAP é o seu suporte embutido a WSDL [MEL 02].

A instalação da API é muito simples. Tudo que você precisa é um servidor web com suporte a PHP e a biblioteca de classes que está disponível em um único arquivo chamado nusoap.php. Tanto o arquivo quanto a documentação podem ser baixados no site do projeto http://sourceforge.net/projects/nusoap [NIC 04a].

A última versão estável é a 0.6.7 de 4/5/2004, a qual suporta a especificação SOAP 1.1, WSDL 1.1 e HTTP 1.0/1.1. Entretanto, o NuSOAP ainda não fornece uma cobertura do SOAP 1.1 e WSDL 1.1 tão completa como em outras implementações como .NET e Apache Axis [KRA 04].

As classes disponíveis na API NuSOAP são: *nusoap_base, soap_fault, XMLSchema, soapval, soap_transport_http, soap_server, wsdl, soap_parser* e *soap_client*. A figura 3.9 apresenta as classes, atributos e métodos da API NuSOAP [AYA 04].

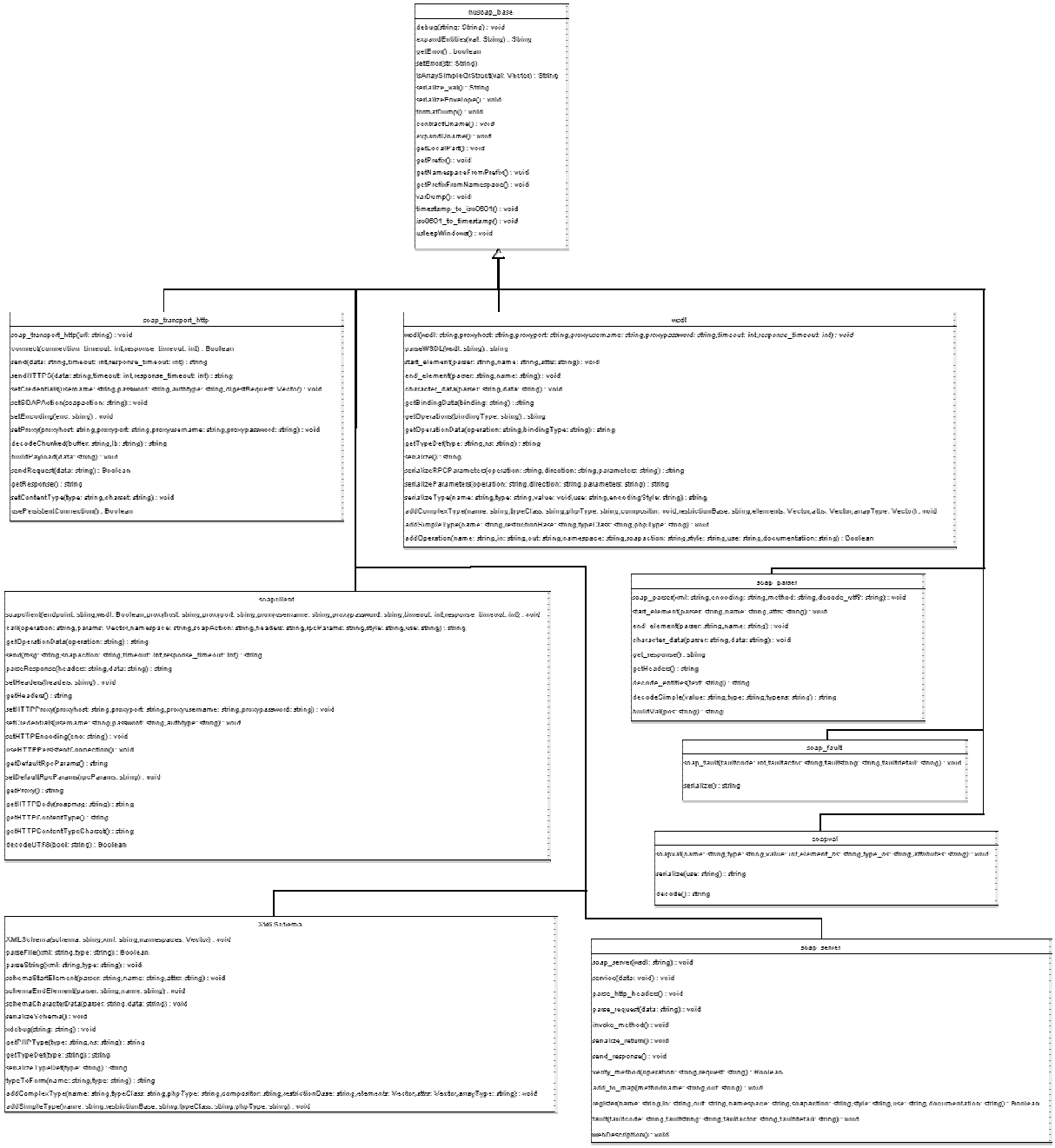


Figura 3.9 – Classes, atributos e métodos do NuSOAP *tollkit*

O NuSOAP fornece algumas facilidades para ajudar o *debug* dos programas. Com o SOAP um tipo de *debug* muito comum é visualizar a requisição e a resposta enviadas. Para isso, a classe *soapclient* possue atributos chamados *request* e *response.* Além disso, o NuSOAP possibilita visualizar informações para *debug* através de métodos das classes como *getError e fault.* Além disso, com o NuSOAP é possível gerar automaticamente informações, tanto em WSDL com em HTML, sobre os Web Services desenvolvidos [NIC 04b].

A seguir, um exemplo de desenvolvimento e uso de Web Services com PHP via NuSOAP.

nusoap-server.php

```php
<?php
      // inclusão do arquivo de classes NuSOAP
      require_once('nusoap.php');
      // criação de uma instância do servidor
      $server = new soap_server;
      // inicializa o suporte a WSDL
      $server->configureWSDL('server2','urn:server2');
      $server->wsdl->schemaTargetNamespace = 'urn:server2';
      // registra o método a ser oferecido
      $server->register('hello',                          //nome do método
            array('name' => 'xsd:string'),                //parâmetros de entrada
            array('return' => 'xsd:string'),              //parâmetros de saída
            'urn:server2',                                //namespace
            'urn:server2#hello',                          //soapaction
            'rpc',                                        //style
            'encoded',                                    //use
            'Says hello to the caller'                    //documentação do serviço
      );
      // definição do método como uma função do PHP
      function hello($name) {
         return 'Hello '.$name;
       }
       // requisição para uso do serviço
       $HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA : '';
       $server->service($HTTP_RAW_POST_DATA);
?>
```

Assim, podemos digitar no navegador a localização do servidor (http://localhost/server2.php) para termos acesso a um resumo do serviço **hello** (figura 3.10 à esquerda) e ao arquivo WSDL (figura 3.10 à direita) que descreve o mesmo, conforme apresentado a seguir.

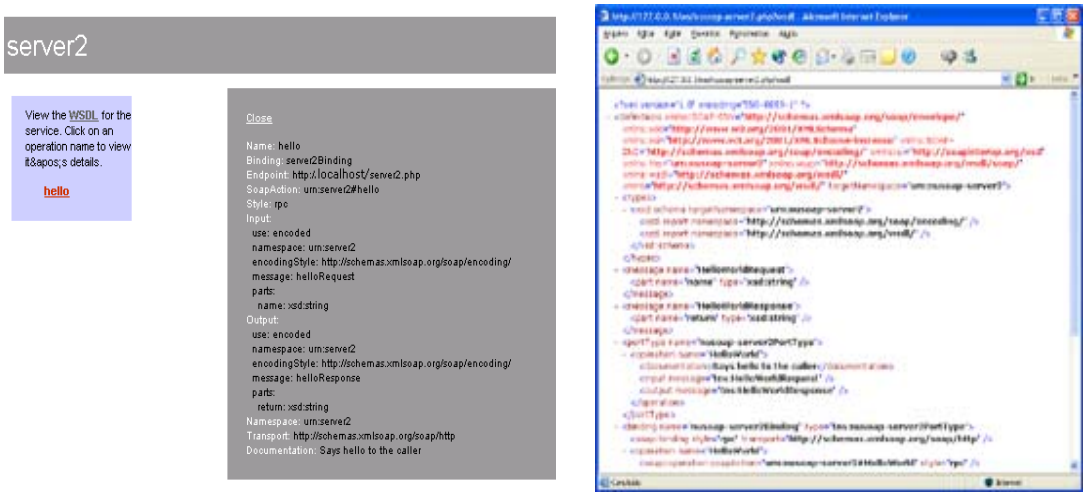


Figura 3.10 – Informações sobre o serviço geradas pelo NuSOAP

Após a criação do servidor, precisamos criar um cliente para acessar o serviço. O código para criação do cliente é apresentado a seguir.

nusoap-client.php

```php
<?php
        // inclusão do arquivo de classes NuSOAP
        require_once('nusoap.php');
        // Definição da localização do arquivo WSDL
        $wsdl = 'http://localhost/nusoap-server.php?wsdl';
        // criação de uma instância do cliente
        $client = new soapclient($wsdl,_true);
        // verifica se ocorreu erro na criação do objeto
        $err = $client->getError();
        if ($err){  echo "<h2>Erro no construtor</h2><pre>".$err."</pre>"; }
        // chamada do método SOAP
        $result = $client->call('hello',array('name' => 'World'));
        // verifica se ocorreu falha na chamada do método
        if ($client->fault){
          echo "<h2>Falha</h2><pre>";
          print_r($result);
          echo "</pre>";
        }else{
          // verifica se ocorreu erro
          $err = $client->getError();
          if ($err){
              echo "<h2>Erro</h2><pre>".$err."</pre>";
          }else{
              // exibe o resultado
              echo "<h2>Result</h2><pre>";
              print_r($result);
              echo "</pre>";
          }
        }
        // exibe a requisição e a resposta
        echo '<h2>Requisição</h2>';
        echo '<pre>'.htmlspecialchars($client->request).'</pre>';
        echo '<h2>Resposta</h2>';
        echo '<pre>'.htmlspecialchars($client->response).'</pre>';
        // Exibe mensagens para debug
        echo '<h2>Debug</h2>';
        echo '<pre>'.htmlspecialchars($client->debug_str).'</pre>';
?>
```

3.3.3 PEAR SOAP Client/Server for PHP

A PEAR (*PHP Extension and Application Repository*) trata-se de um repositório de extensões e aplicações em PHP. Essas são disponibilizadas através de pacotes de classes desenvolvidas em PHP orientado a objeto, sobre licença livre, os quais são desenvolvidos pela comunidade de desenvolvedores PHP [RUB 04].

O pacote SOAP da PEAR, trata-se da implementação do protocolo SOAP e serviços. Sua última versão é 0.8RC3 de 16/01/2003. A figura 3.11 apresenta as classes disponíveis na PEAR SOAP [RUB 04].

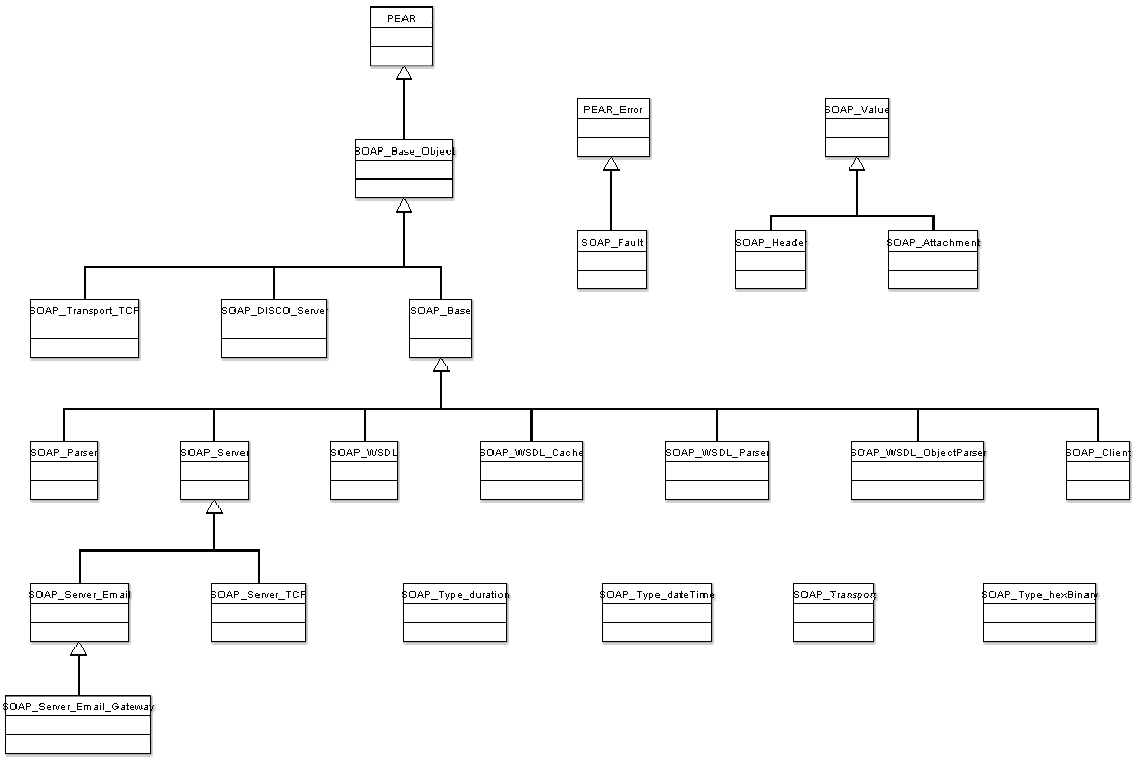


Figura 3.11 – Classes da PEAR SOAP

Assim como o NuSOAP, o pacote SOAP da PEAR oferece algumas facilidades para ajudar o *debug* dos programas. Para isso, a classe *Soap_Client* possue atributos chamados *lastrequest* e *lastesponse.* Além disso, os erros ocorridos podem ser reportados através de objetos da classe *SOAP_Fault* [RUB 04]

Os erros podem ser de quarto tipos:

VersionMismatch: namespace inválido para o envelope SOAP

MustUnderstand: o cabeçalho não foi entendido pelo servidor

Client: a mensagem enviada pelo cliente não está no formato correto ou não contém a informação necessária

Server: a mensagem não pode ser processada

O objeto também pode conter outras informações, como mensagens básicas com a descrição do erro e a URI da origem da falha.

No exemplo a seguir, o serviço recebe dois números inteiros como parâmetros de entrada e retorna uma string como resposta. Primeiramente, inclui-se as bibliotecas *Server.php* e *Disco.php,* a qual contém uma série de métodos para geração de arquivos WSDL. Após a declaração da classe do Web Service, chamada `Sales`, define-se um *array* chamado $`__dispatch_map` e o construtor da classe, necessários para a geração do arquivo WSDL. Assim, garante-se que sempre que a classe for instanciada o atributo `__dispatch_map` será populado com informações sobre os campos de entrada e saída do método `onsale,` o qual contém toda a funcionalidade do Web service [TRA 03].

Após a criação de objetos das classes *SOAP_Server* e *Sales,* cria-se um teste para verificar o tipo de requisição enviada pelo cliente. Se for detectado uma requisição explícita o método `service` é invocado o qual retorna o resultado do serviço. Caso contrário, uma

instância da classe *SOAP_DISCO_Server* é criada, a qual facilita o retorno de um arquivo WSDL ao requisitante. Nesta situação, cria-se mais um teste para verificar se a URL contém a extensão `wsdl`. Caso afirmativo, o arquivo WSDL correspondente é retornado; senão, uma URL absoluta é retornada indicando o local de onde o arquivo WSDL deve ser extraído

pear-server.php

```php
<?php
// inclui as bibliotecas das classes Server e Disco
require_once('SOAP/Server.php');
require_once('SOAP/Disco.php');
// definição do serviço a ser oferecido
class Sales{
     var $__dispatch_map = array();
     // definição da assinatura do método Web services
     function Sales() {
        // necessário para a criação do arquivo WSDL
        $this->__dispatch_map['onsale'] =
             array('in' => array('low' => 'integer', 'high' => 'integer'),
                   'out' => array('sales' => 'string'));
     }
     // definição do método
     function onsale($low, $high)  {
       if ($low < 1) {
           return "We have some bargains!";
       } else if ( $high < 100 ) {
           return "We have a few gadgets which fit your price range!";
       } else {
           return "Thats more than spare change!";
       }
   }
}
// criação de um objeto da classe SOAP_Server
$server = new SOAP_Server();
// criação de um objeto da classe Sales
$webservice = new Sales();
// registro do método
$server->addObjectMap($webservice,'http://schemas.xmlsoap.org/soap/envelope/');

if (isset($_SERVER['REQUEST_METHOD'])  &&
     $_SERVER['REQUEST_METHOD']=='POST') {
     $server->service($HTTP_RAW_POST_DATA);
} else {
     // Cria um servidor DISCO
     $disco = new SOAP_DISCO_Server($server,'Sales');
     header("Content-type: text/xml");
     if (isset($_SERVER['QUERY_STRING']) &&
         strcasecmp($_SERVER['QUERY_STRING'],'wsdl') == 0) {
         echo $disco->getWSDL();
     } else { echo $disco->getDISCO();  }
}
exit;
?>
```

Agora, criamos o cliente que fará acesso ao Web Service através de informações contidas no arquivo WSDL.

O código também inicia com a inclusão da biblioteca de classes Client.php. Logo após, cria-se uma instância da classe *SOAP_WSDL* a qual recebe uma URL indicando o local do arquivo WSDL. O serviço é chamado através do método *onsale* que passa como parâmetro as duas variáveis contendo inteiros `$int1` e `$int2`.

```
pear-client.php
<?php
   // inclui a biblioteca da classe Client
   require_once 'SOAP/Client.php';
   // cria uma instância da classe SOAP_WSDL
   $wsdl  = new SOAP_WSDL ('http://localhost/pear-service.php?wsdl');
   // recebe uma instância do objeto Sales
   $Sales = $wsdl->getProxy();
   // definição dos parâmetros
   $int1  = (integer) $argv[1];
   $int2  = (integer) $argv[2];
   // chamada do do serviço
   echo ( $Sales->onsale($int1,$int2) );
?>
```

3.3.4 SOAP *Extension*

Trata-se da implementação do protocolo SOAP e serviços, versão 0.8RC3 de 16/01/2003. A extensão SOAP do PHP 5 é a primeira implementação do protocolo SOAP para PHP em C. Esta implementação possui algumas vantagens em relação a outras implementações escritas em PHP, sendo a principal delas a velocidade [ZEN 04].

Essa extensão, ainda definida como experimental, pode ser utilizada para implementar servidores e clientes SOAP com suporte a SOAP 1.1, SOAP 1.2 e WSDL 1.1. Como requisito para o uso desta extensão, temos a biblioteca GNOME XML e, por isso, é necessário pelo menos a versão 2.5.4 da mesma. Além disso, esta extensão só estará disponível se o PHP for configurado com a diretiva para uso da mesma *–enable-soap* [ARG 04] [FUE 04].

As classes e métodos disponíveis através da extensão SOAP são apresentadas na figura a seguir.

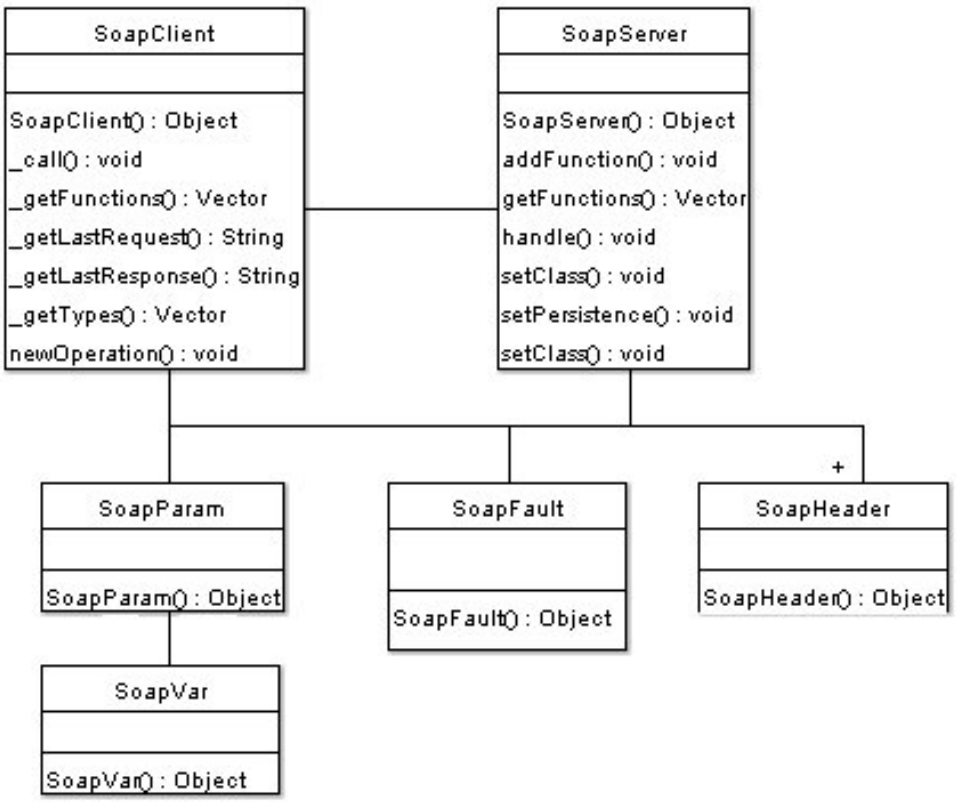


Figura 3.12 – Classes e métodos da SOAP *Extension*

A seguir um exemplo simples de criação de um serviço chamado *helloWorld*, que recebe um nome (*string*) e retorna um texto (*string*). O código pra criação do servidor é apresentado abaixo:

soap-server.php

```php
<?php
        //criação de uma instância do servidor
        $server = new SoapServer(null, array('uri' => "http://localhost/"));
        //definição do serviço
        function helloWorld($name)
        {
           return "Hello ".$name;
        }
        //registro do serviço
        $server->addFunction("helloWorld");
        //chamada do método para atender as requisição do serviço
        $server->handle();
?>
```

Após a criação do servidor, precisamos criar um cliente para acessar o serviço. O código para criação do cliente é apresentado a seguir.

soap-client1.php

```php
<?php
     // criação de uma instância do cliente
     $client = new SoapClient(null, array(
                     'location' => 'http://localhost/soap-server.php',
                     'uri'      =>  'http://localhost/
                     'trace'    =>  1);
     // chamada do serviço SOAP
     $result = $client->helloWorld('Hello');
     // verifica erros na execução do serviço e exibe o resultado
     if (is_soap_fault($result)){
          trigger_error("SOAP Fault: (faultcode: {$result->faultcode},
                     faultstring: {$result->faulstring})", E_ERROR);
     }else{
          print_r($result);
     }
     //Retorna a requisição SOAP
     echo "<br><br><b>Requisição: <br></b>";
     echo str_replace(" ","<br>",htmlspecialchars($client->__getLastRequest()));
     echo "<br><br>";
     //Retorna a resposta SOAP
     echo "<br><br><b>Resposta: <br></b>";
     echo str_replace(" ","<br>",htmlspecialchars($client->__getLastResponse()));
     echo "<br><br>";
?>
```

Ao chamarmos o cliente através de um navegador, http://localhost/soap-client.php, visualizaremos a seguinte mensagem: “Hello World”

3.3.5 REST

O *Representational State Transfer* - REST difere bastante do SOAP e do XML-RPC. Primeiramente, ele não é um padrão. Segundo, não existem classes pré-construídas para a criação de clientes e servidores em PHP. O ponto forte do REST é que ele não exige nenhuma extensão especial ou biblioteca de classes para o desenvolvimento de Web Services. O protocolo HTTP possui todas os recursos necessários para o envio e recebimento de mensagens padrão XML através de métodos padrão como o GET, POST e PUT. Dessa

forma, é necessário utilizar ferramentas como PHP DOM, SAX, ou também XSL para fazer o *parsing* [TRA 03].

Abaixo é apresentado um exemplo de desenvolvimento e uso de Web Services via REST através do uso da função file_get_contents() que lê todo o conteúdo de um arquivo para uma string. O programa obtém o valor da *query string*, executa o cálculo do imposto e gera um XML com o resultado.

```
            echo "</taxinfo>";
?>
```

A seguir, é apresentado o código do cliente REST.

```
?>
```

4 Metodologia

Este trabalho está baseado na experiência adquirida no desenvolvimento de um projeto prévio chamado Sign WebMessage. O Sign WebMessage apresentado na figura 4.1 (http://www.inf.unisinos.br/swm) trata-se de uma ferramenta para comunicação assíncrona na web, através da qual pode-se interagir tanto através da escrita da língua portuguesa quanto através da escrita da Libras. Esta ferramenta foi desenvolvida pelo autor, como parte do trabalho de conclusão do curso de graduação em Análise de Sistemas da Unisinos [SOU 02a].

Nas mensagens, os sinais podem ser visualizados em SignWriting e, opcionalmente, seus significados em português, o que proporciona apoio à aprendizagem de ambas as línguas. Essa ferramenta tem como objetivo principal minimizar as dificuldades de comunicação escrita entre os surdos e entre os surdos e ouvintes, pois permite a interação de seus usuários em ambas as línguas [SOU 02b].

A partir do momento em que o usuário acessa o sistema é exibida sua caixa de entrada com as mensagens recebidas. Para a leitura das mensagens, o usuário pode optar em visualizar a mensagem em sinais e português simultaneamente ou apenas em sinais. Já para enviar uma mensagem o usuário deve selecionar o destinatário e editar a mensagem procurando pelos sinais no dicionário do ambiente.

Existem duas formas para pesquisar os sinais do dicionário: pesquisa por sinais através da língua portuguesa, na qual o usuário poderá realizar consultas a partir da palavra na língua oral, obtendo o sinal correspondente; e pesquisa a partir da língua de sinais, de acordo com as características gestuais do sinal [SOU 03a].

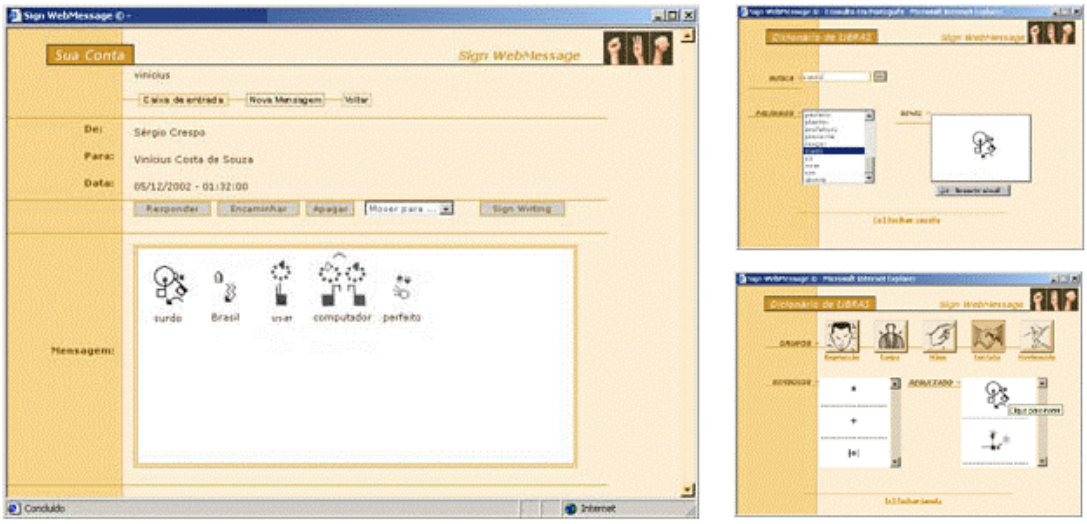


Figura 4.1 - Interface do Sign WebMessage [SOU 03a]

Sabe-se que ao desenvolver um produto de software, não se pode esperar o final do processo de desenvolvimento para avaliar se o produto tem a qualidade desejada. Dessa maneira, durante todo o processo de desenvolvimento do Sign WebMessage foram realizadas avaliações com a colaboração de surdos e ouvintes. Além disso, foi realizada uma pesquisa de opinião sobre as funcionalidades e potencialidades da ferramenta. Os resultados foram obtidos

através de observações durante o uso do software e de aplicação de questionário[3]. O questionário aplicado foi elaborado para abordar três aspectos fundamentais quanto ao Sign WebMessage: interface, utilização e contribuições.

Visto que o sistema pode ser utilizado por tipos de usuários diferentes, definiu-se dois grupos de avaliação. Um grupo de usuários surdos, conhecedores da Língua Brasileira de Sinais, e outro de usuários ouvintes desconhecedores da Libras. Em cada experimento, foi realizada uma explicação do funcionamento do sistema permitindo, posteriormente, sua utilização pelos participantes.

Quanto ao grupo de surdos, foi composto um grupo de referência com 12 pessoas que possuíam exclusivamente surdez, para participar de um curso de introdução à informática com o objetivo de conhecer e interagir com o público-alvo. O curso foi realizado de 06 de junho a 31 de outubro de 2002, totalizando 60h/aula, com 3h/aula semanais. O projeto contou com a participação de três voluntários: um instrutor, autor deste trabalho e duas monitoras, Márcia Rafaeli Aguiar, aluna do curso de pedagogia e Maria Helena Pilengi, aluna do curso de psicologia, os quais formaram uma equipe interdisciplinar para organizar, acompanhar e ministrar as aulas.

Os alunos foram selecionados com o apoio do Instituto Humanitas da Unisinos, o qual firmou parceria com entidades regionais que realizam trabalhos com surdos. As aulas foram ministradas no laboratório de informática deste instituto. O objetivo do curso foi capacitar os alunos na utilização dos recursos disponíveis no laboratório, como por exemplo: editor de imagens, editor de textos, correio eletrônico, Internet, entre outros. Dessa forma, foi possível realizar observações e tomar conhecimento sobre a língua e a cultura surda, além de constatar os benefícios que o contato com a informática pode trazer para essa comunidade.

A pesquisa com o grupo de surdos foi realizada durante uma aula do curso, o qual os alunos vinham freqüentando há mais de três meses. Estiveram presentes 11 alunos.

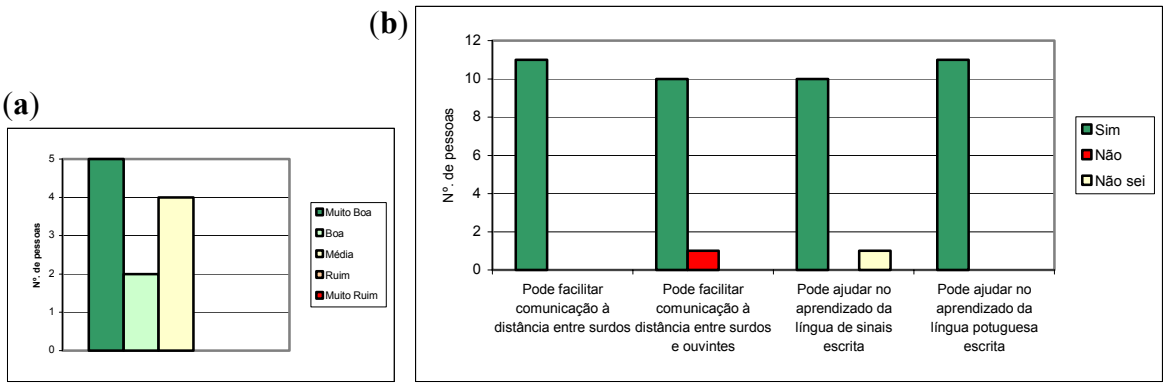


Figura 4.2 – Opinião dos surdos quanto a interface e contribuições do Sign WebMessage

Em geral, os resultados apurados foram muito satisfatórios nos três aspectos abordados, pois quanto à interface, as opiniões variaram entre média e muito boa (figura 4.2 a). Em relação à utilização da ferramenta, os resultados apontaram-na como sendo de média à fácil utilização e quanto às contribuições que o Sign WebMessage poderá proporcionar ao processo de aprendizagem das línguas envolvidas, a quase totalidade das opiniões afirmam que a ferramenta cumprirá com seus objetivos como pode ser observado no gráfico b apresentado na figura 4.2.

[3] Anexo E – Questionário de avaliação do Sign WebMessage

Tendo-se em vista a avaliação do sistema no que se refere à utilização e aprendizado das línguas de sinais por ouvintes, realizou-se uma pesquisa de opinião com 12 pessoas, alunos do curso de Informática e funcionários da Unisinos. Neste caso, todos os participantes já tinham experiência no uso de ferramentas de comunicação via Internet. Essa pesquisa foi realizada em momentos diferentes e em pequenos grupos, de até 4 pessoas.

Os resultados nos três aspectos: interface, utilização e contribuições foram muito satisfatórios, pois quanto à interface a maioria dos participantes qualificaram-na como muito boa (figura 4.3 a). Em relação à utilização da ferramenta, os resultados apontaram-na como de fácil ou muito fácil utilização e quanto às contribuições que o Sign WebMessage poderá proporcionar ao processo de aprendizagem das línguas envolvidas, novamente a quase totalidade das opiniões afirmam que a ferramenta cumprirá com seus objetivos, como pode ser observado no gráfico b apresentado na figura 4.3

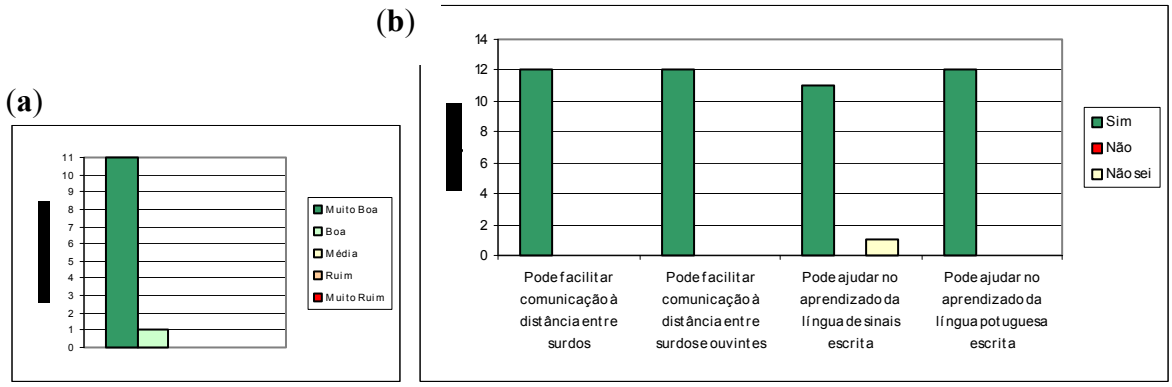


Figura 4.3 – Opinião dos ouvintes quanto a interface e contribuições do Sign WebMessage

Considerando os resultados obtidos na pesquisa, foi possível demonstrar a utilidade da ferramenta desenvolvida para o objetivo a que se destina, ou seja, sua validade no sentido de contribuir para uma maior divulgação da língua de sinais como importante meio de comunicação e acesso à informação foi comprovada. Por essa razão, tornou-se evidente a importância de aprimorar as funcionalidades e características da ferramenta, e disponibilizá-las, através de uma tecnologia apropriada, para que possam contribuir para o surgimento de novas ferramentas que utilizem a Libras.

5 Aprimoramento do Sign WebMessage

A partir da pesquisa realizada sobre trabalhos relacionados, verificou-se que existem alguns recursos, essenciais para a manipulação das línguas de sinais, que ainda não estavam disponíveis no Sign WebMessage, software no qual a biblioteca SWService está baseada. Os recursos ainda não implementados são os necessários para criação e edição de sinais e para importação e exportação de arquivos no formato SWML.

Implementou-se tais recursos no Sign WebMessage de forma que, para o desenvolvimento da SWService, já se tenha experiência sobre os mesmos garantindo, assim, maior qualidade da biblioteca. A seguir, é apresentada uma descrição de como tais recursos foram implementados.

A implementação foi realizada em duas etapas: primeiramente desenvolveu-se um módulo através do qual os usuários podem criar novos sinais e editar sinais já existentes, os quais são armazenados em arquivos SWML. A segunda etapa foi desenvolver um módulo que permita ao Sign WebMessage ler um arquivo qualquer padrão SWML e analisá-lo sintaticamente de forma a reproduzir fielmente os sinais especificados no mesmo. Além disso, realizou-se um estudo sobre o *Sing-Symbol-Sequence* – SSS a fim de minimizar o grande número de símbolos utilizados nas interfaces [SOU 04].

5.1 Adoção do formato SWML

Macedo, em [MAC 99], afirma que a principal dificuldade de se ter uma notação escrita para as línguas de sinais é o seu caráter não seqüencial. As línguas de sinais têm uma estrutura paralela, com o uso de gestos complexos envolvendo simultaneamente diversas partes do corpo do sinalizador (braços, mãos, dedos, cabeça, face, tronco etc.). Por isso, a representação das línguas de sinais costuma ser feita figurativamente, com o auxílio de desenhos, fotografias e filmes, que não apresentam a mesma facilidade de edição da língua escrita.

Inicialmente, o dicionário de sinais do Sign WebMessage foi composto através de imagens (cada sinal uma imagem GIF), o que dificulta a edição e a criação de novos sinais, pois para isso é necessário utilizar um software de edição de imagens que gere imagens GIF. Notoriamente, esta não é a forma mais adequada para compor o dicionário de sinais, pois não viabiliza facilidades de edição, dificulta a criação de novos sinais e ocupa grande espaço no servidor.

Como alternativa para este problema, um grupo de pesquisa da Universidade Católica de Pelotas – UCPEL ligados a um projeto chamado SignNet - Adaptando a tecnologia da Internet para as linguagens de sinais e a educação de surdos, o qual objetiva criar, desenvolver, pesquisar e implementar softwares, ferramentas, utilitários e sistemas computacionais dedicados ao uso e à educação de pessoas surdas sugere a utilização da linguagem SWML associada a uma biblioteca de imagens, SSS, a qual contém todos os símbolos do sistema SignWriting, que combinados podem dar origem aos mais diversos sinais.

A validação da SWML como alternativa para melhorar a forma de armazenamento e processamento dos sinais do Sign WebMessage foi realizada através da criação de um módulo

para criação de novos sinais e um que permite ao Sign WebMessage ler um arquivo SWML e reproduzir fielmente os sinais especificados no mesmo.

5.1.1 Criação e edição de sinais

A figura 5.1 apresenta o módulo para criação de sinais, onde o usuário deve primeiramente selecionar os símbolos que irão compor o sinal, os quais são exibidos na área de construção. Após, o usuário pode movê-los (à esquerda, à direita, para cima ou para baixo) ou até mesmo apagá-los, em caso de erro, até compor o sinal desejado. Para possibilitar a manipulação dos símbolos (imagens GIF do SSS) em um ambiente web, foram desenvolvidas funções em JavaScript que são chamadas cada vez que ocorre um evento *onClick* sobre os botões da interface. Essas funções manipulam as seguintes propriedades da página HTML: *left layer, top layer, visibility layer, border image* e *form field.*

Na implementação destes recursos foram utilizados, também, *layers* que possibilitam a criação de áreas dentro de páginas web com largura e altura pré-definidas e em uma posição exata, através de coordenadas dos eixos x e y. As *layers* podem conter textos, imagens, formulários ou qualquer outro objeto utilizado em HTML, até mesmo outras *layers*. Além disso, fornecem a possibilidade de um controle fino sobre o seu posicionamento em relação à página e possuem as seguintes propriedades: *left* e *top* (coordenadas x e y, respectivamente), *z-index* que permite definir a sobreposição das *layers* e *visibility* que define se uma *layer* é visível ou não.

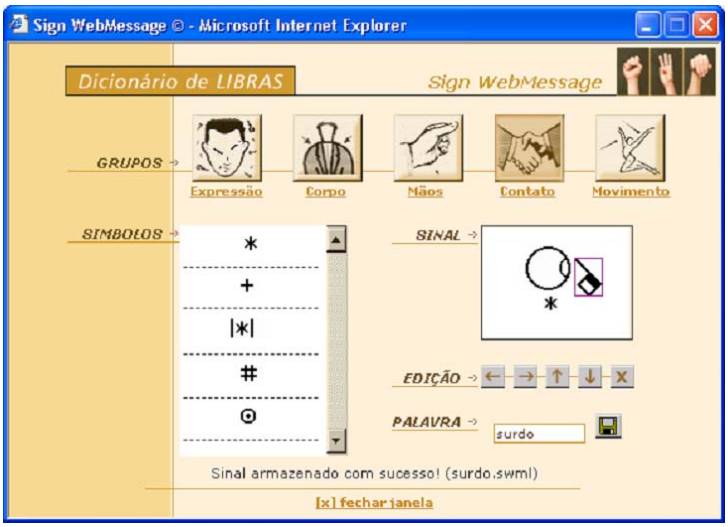


Figura 5.1 - Módulo para criação e edição de sinais no Sign WebMessage

5.1.2 Análise de arquivos padrão SWML

O arquivo SWML gerado no exemplo da figura 5.1 é apresentado a seguir.

```
<swml version="1.0"
       file-type="dic"
       xmlns="http://www.ucpel.tche.br/2002/05/swml"
       xmlns:xlink="http://www.w3.org/1999/xlink"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.ucpel.tche.br/2002/05/swml
       http://swml.ucpel.tche.br/schemas/swml/2002/05/swml.xsd">
  <generator>
       <name>Sign WebMessage</name>
       <version>1.0</version>
  </generator>
  <dic symbol-set="sss2002" oral-lang="BR" sign-lang="BR"/>
       <dic-entry>
         <signbox>
            <symb x="35" y="15" x-flop="0" y-flop="0" colour="0,0,0">
                <category>01</category>
                <group>01</group>
                <symbnum>001</symbnum>
                <variation>01</variation>
                <fill>02</fill>
                <rotation>02</rotation>
            </symb>
            <symb x="75" y="25" x-flop="0" y-flop="0" colour="0,0,0">
                <category>03</category>
                <group>04</group>
                <symbnum>003</symbnum>
                <variation>01</variation>
                <fill>02</fill>
                <rotation>01</rotation>
            </symb>
            <symb x="50" y="55" x-flop="0" y-flop="0" colour="0,0,0">
                <category>02</category>
                <group>01</group>
                <symbnum>001</symbnum>
                <variation>01</variation>
                <fill>01</fill>
                <rotation>01</rotation>
            </symb>
         </signbox>
         <gloss>surdo</gloss>
   </dic-entry>
</swml>
```

Para realizar a análise do arquivo SWML utilizou-se a *Simple API for XML* – SAX, uma API leve e de fácil utilização, disponível no PHP, que permite analisar sintaticamente arquivos padrão XML. Outro recurso utilizado foi a biblioteca gd, desenvolvida em C e atualmente com interface para PHP, a qual provê funções para criação e manipulação de imagens.

A seguir, é apresentado o algorítmo, implementado em PHP, para análise do arquivo SWML e exibição do sinal:

1. Criar o *parser*
2. Registrar as funções de início e fim de *tag*
3. Registrar a função de tratamento de caractere
4. Percorrer arquivo SWML
5. Para cada *tag* <SINGBOX> encontrada criar uma *Layer box*
6. Para cada *tag* <SYMB>
   - 6.1 Obter as seguintes informações:
     - ***Coordenada x***
     - ***Coordenada y***
     - ***Category***
     - ***Group***

- ***Symbnum***
- ***Variation***
- ***Fill***
- ***Rotation***

6.2 Montar o nome do símbolo (arquivo GIF) a ser exibido
- **Category-Group-Symbnum-Variation-Fill-Rotation.gif**

6.3 Abrir o arquivo com a imagem do símbolo
6.4 Utilizar a biblioteca gd para descobrir
- largura da GIF
- altura da GIF

6.5 Criar uma *layer símbolo* com a largura e altura da imagem e posicionar essa *layer* na posição (x,y) da *Layer Box*
6.6 Exibir o símbolo (imagem GIF) dentro da *Layer símbolo*

A figura 5.2 apresenta a forma de exibição do sinal, utilizado no exemplo anterior, através das *layers*.

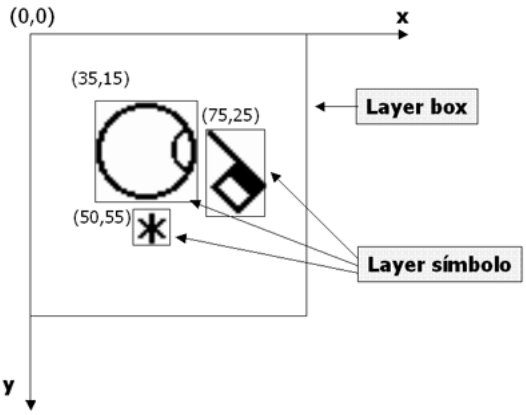


Figura 5.2 - Exibição de um sinal através de *layers*

Com essa experiência, pôde-se constatar que através da linguagem SWML foi possível melhorar a forma de armazenamento e processamento de sinais no Sign WebMessage, pois conseguiu-se criar um novo sinal, em ambiente exclusivamente web, e gerar um arquivo SWML com as características do mesmo. Posteriormente, conseguiu-se reproduzir este sinal com base apenas na análise do arquivo SWML.

5.2 *Sing-Symbol-Sequence* - SSS

Ainda como resultado do estudo dos softwares apresentados no capítulo 3, constatou-se que uma das dificuldades na utilização das ferramentas se dá pela enorme quantidade de símbolos existentes no sistema de escrita SignWriting, os símbolos do SSS que em 2002 foram contabilizados em 16.111 símbolos [SUT 04]. Com tantos símbolos, torna-se difícil disponibilizá-los nas interfaces dos softwares de modo que os usuários tenham acesso aos símbolos desejados de forma rápida e eficiente. Além disso, como existem muitos símbolos parecidos, as interfaces acabam ficando visualmente poluídas, como por exemplo no SWEdit apresentado na figura 5.3, o que causa confusão e dificuldade para os usuários.

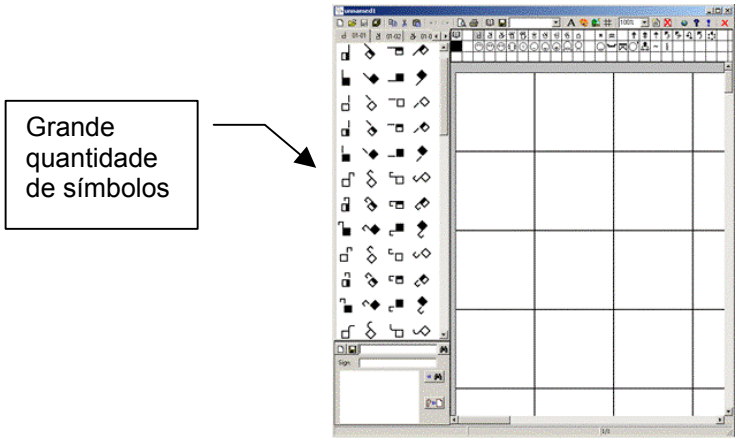


Figura 5.3 – Interface do SWEdit

Por isso, foi realizado um estudo sobre o SSS-2002 [SUT 02], no qual identificou-se aproximadamente 250 símbolos base dos quais todos os outros derivam. Através deste estudo, foi possível identificar a possibilidade de apresentar nas interfaces, no caso dos símbolos para mãos e para movimentos, apenas os símbolos base e, a partir destes, obtermos qualquer uma de suas variações. Para viabilizar as variações dos símbolos base, foram adicionadas novas opções de ações na edição dos símbolos: espelhamento, rotação horizontal/vertical, rotação em torno do eixo Z e rotação em torno do próprio eixo (palma, lado e dorso).

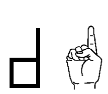
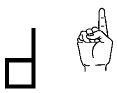
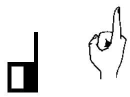
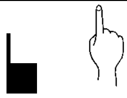
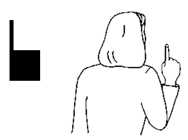
A seguir é apresentado um exemplo do padrão de variações para símbolos de mãos e um para símbolos de movimentos. Primeiramente, analisaremos os símbolos derivados do primeiro símbolo base do SSS-2002, os símbolos pertencentes ao grupo de mão número um. Neste grupo de mão, que utiliza apenas o dedo indicador esticado, assim como para todos os outros símbolos base de mão, existem 95 variações conforme apresentado na tabela 5.1.

Tabela 5.1 – Variações do primeiro símbolo base do *Sign-Symbol-Sequence*

Mão na vertical						**Mão na horizontal**					
Palma		**Lado**		**Dorso**		**Palma**		**Lado**		**dorso**	
rotação	espelho	rotação	espelho	rotação	espelho	rotação	espelho	Rotação	espelho	rotação	espelho

Como pode ser observado na tabela 5.1, as variações do símbolo base são decorrentes de rotações da mão do sinalizador em torno do próprio eixo (palma, lado e dorso), mão na horizontal ou na vertical, espelhamento e oito rotações com as possibilidades de direcionamento da mão. A tabela 5.2 apresenta as variações iniciais que combinadas dão origem às 95 variações existentes para este símbolo base.

Tabela 5.2 – Símbolo base do grupo 1 e suas variações iniciais

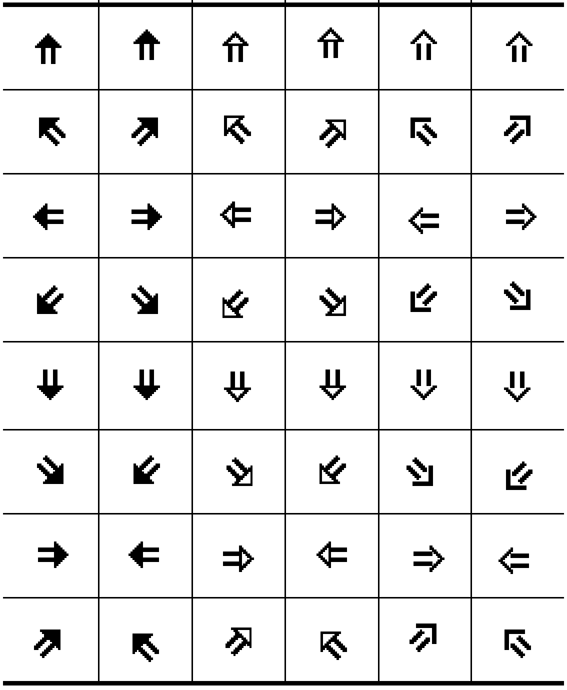
Símbolo base	Rotação em torno do próprio eixo			Orientação da mão	
	Palma	Lado	Dorso	Vertical	Horizontal
					

O anexo B apresenta os símbolos base de cada grupo de mão utilizados nas interfaces da SWService, bem como o número total de símbolos e o número de reduções obtidas através da solução adotada.

A seguir, é apresentado um exemplo do padrão de variações para símbolos de movimentos. Como exemplo, são apresentadas todas as variações existentes para o símbolo base de movimento reto e na vertical. Conforme apresentado na tabela 5.3, existem 48 variações para este símbolo, assim como para a maioria dos demais símbolos para movimento.

Tabela 5.3 – Variações do símbolo base de movimento reto e na vertical

Movimento reto e na vertical					
Mão direita		**Mão esquerda**		**Duas mãos**	
rotação	espelho	rotação	espelho	rotação	espelho



Como o estudo realizado com os símbolos para movimento, foi possível constatar que é possível obter qualquer uma das 47 variações do símbolo base a partir de três ações na edição do mesmo: rotação em torno do eixo Z, espelhamento e mão direita, esquerda ou ambas.

O anexo C apresenta os símbolos base de cada grupo de movimento utilizados na nas interfaces da SWService, bem como o número total de símbolos e o número de reduções obtidas através da solução adotada.

A tabela 5.4 apresenta um resumo dos resultados obtidos através da solução proposta, quanto a diminuição do número de símbolos nas interfaces dos softwares que utilizam o SignWriting.

Tabela 5.4 – Resultados obtidos

Grupo de símbolos	Subgrupo de símbolos	Símbolos existentes	Símbolos utilizados	Símbolos redução
Mão	Um	480	5	475
	Dois	576	6	570
	Três	768	8	760
	Quatro	384	4	380
	Cinco	1920	20	1900
	Seis	1056	11	1045
	Sete	192	2	190
	Oito	480	5	475
	Nove	1632	17	1615
	Dez	672	7	665
	Símbolos de mão não utilizados na Libras	2400	0	2400
	Totais	**10560**	**85**	**10475**
Grupo de símbolos	**Subgrupo de símbolos**	**Símbolos existentes**	**Símbolos utilizados**	**Símbolos redução**
Movimento	Parede	696	16	680
	Diagonal	216	12	204
	Chão	1080	25	1055
	Curvo parede	716	24	692
	Curvo bate parede	96	8	88
	Curvo bate chão	282	14	268
	Curvo chão	232	16	216
	Circular	426	16	410
	Totais	**3744**	**131**	**3613**
	Totais gerais	**14304**	**216**	**14088**

Como pode ser observado nos números apresentados na tabela anterior, conseguiu-se diminuir o número de símbolos utilizados na SWService de 16.111 para 2.023, o que significa uma redução de 87,44% (14.088 símbolos).

A nova interface para criação e edição dos sinais, após o estudo sobre o SSS, é apresentada na figura 5.4

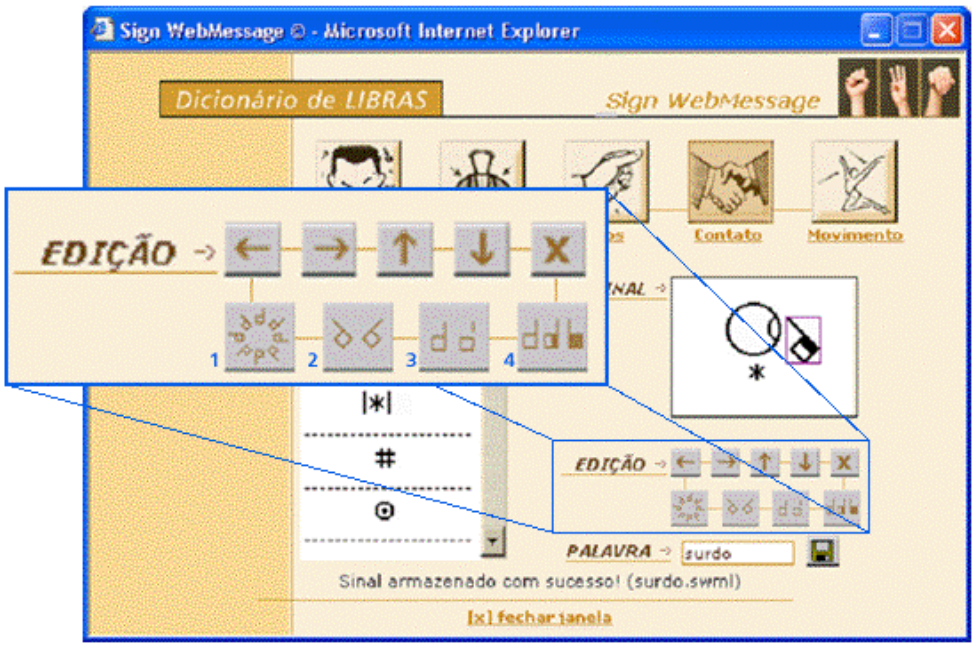


Figura 5.4 – Nova interface para criação e edição de sinais

Dessa forma, para se chegar no símbolo marcado na figura 5.4, o qual é uma variação do símbolo base do grupo de mão um, o usuário deverá inserir o símbolo base do grupo 1 de mão, clicar uma vez no botão para rotação em torno do próprio eixo (botão 4) e uma vez no botão para mudança de direção da mão (botão 1).

Assim, através da redução do número de símbolos será possível disponibilizar aos usuários interfaces mais amigáveis, de forma que os mesmos poderão obter os símbolos desejados de maneira mais rápida, fácil e eficiente.

6 SignWriting Web Service – SWService

Conforme já mencionado, o objetivo geral do trabalho foi modelar e desenvolver a biblioteca SWService que utilizou a tecnologia de Web Services de modo a fornecer os recursos necessários para que softwares baseados na web possam utilizar o sistema para escrita das línguas de sinais - SignWriting.

A seguir, é apresentada a arquitetura da biblioteca, seus casos de uso, o modelo entidade-realcionamento e diagrama de classes. Além disso, são apresentados os serviços implementados, exemplos de uso dos mesmos e um estudo de caso, no qual os serviços foram utilizados para possibilitar que uma ferramenta de fórum possibilite a escrita em Libras.

6.1 Arquitetura proposta

A figura 6.1 apresenta a arquitetura que possibilita a integração e interoperabilidade entre a SWService, as aplicações web e os usuários finais. Através da arquitetura proposta, os usuários podem utilizar aplicações web (chats, webmails, fóruns, dentre outros) as quais estarão fazendo uso dos serviços oferecidos pela biblioteca SWService (*getSign*, *dicPort*, *dicSW* e *createSign*) sem que seja necessário ter estes componentes de software instalados ou construídos no servidor que contém as aplicações.

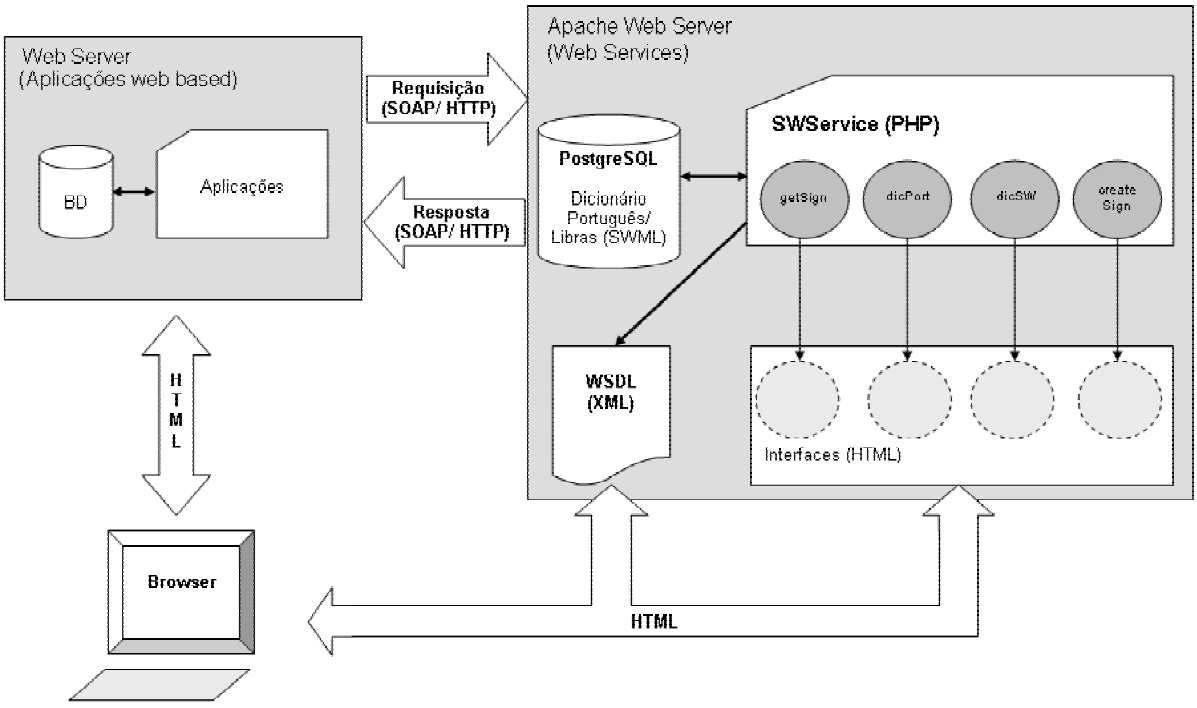


Figura 6.1 - Arquitetura de uso da biblioteca de serviços SWService

A interação entre o servidor web e o *browser* dos usuários se dá através de arquivos HTML que são transportados via protocolo HTTP. Já a interação entre o servidor de aplicação e o servidor que contém os serviços é estabelecida via protocolo SOAP sobre HTTP.

A interoperabilidade é garantida pelo uso de padrões como a linguagem SWML, através da qual as características dos sinais são transmitidas em formato padrão XML. Dessa forma, as aplicações web que utilizam os serviços da SWService podem ser desenvolvidas em qualquer linguagem e plataforma, independentemente da linguagem e plataforma utilizadas na implementação da SWService. A biblioteca foi desenvolvida sobre sistema operacional Red Hat Linux, banco de dados PostgreSQL, PHP para programação e servidor web Apache.

Além disso, a arquitetura prevê a disponibilização de um dicionário público bilíngüe (língua de sinais – língua oral) junto à SWService, o qual está disponível para consultas e será compartilhado por todas as aplicações que fizerem uso da biblioteca. Para os desenvolvedores que desejam fazer uso da biblioteca existe a possibilidade de consultas às interfaces dos serviços e ao arquivo WSDL, com a descrição dos mesmos.

Através da arquitetura proposta, foi possível implementar uma camada de software que fornece os recursos necessários para utilização das línguas de sinais em ambientes web e disponibilizá-la amplamente através da Internet, para que outras aplicações possam fazer uso de forma rápida, eficiente e independente da linguagem ou plataforma de desenvolvimento escolhida.

6.2 Diagramas de casos de uso

Foram definidos quatro diagramas de caso de uso, um para cada serviço implementado na biblioteca SWService: *getSign*, *dicPort*, *dicSW* e *createSign*.

A figura 6.2 apresenta o diagrama de caso de uso do serviço *getSign*. Este caso de uso apresenta as duas possibilidades de interações dos usuários com as funcionalidades disponíveis pelo serviço: enviar texto em português e receber texto em Libras e português.

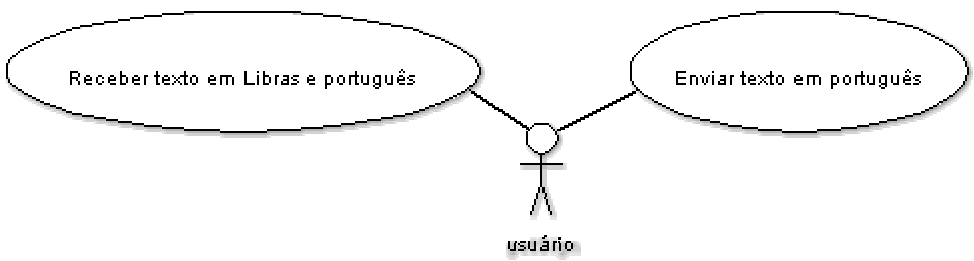


Figura 6.2 – Diagrama de caso de uso *getSign*

Já a figura 6.3 apresenta o diagrama de caso de uso do serviço *dicPort*, o qual possibilita consultas ao dicionário bilíngüe da SWService a partir da língua portuguesa. As ações possíveis na utilização deste serviço são: digitar um palavra em português para pesquisar o sinal correspondente, selecionar uma palavra na lista de entradas do dicionário para visualização do sinal em Libras, inserir o sinal na mensagem e fechar a janela de pesquisa.

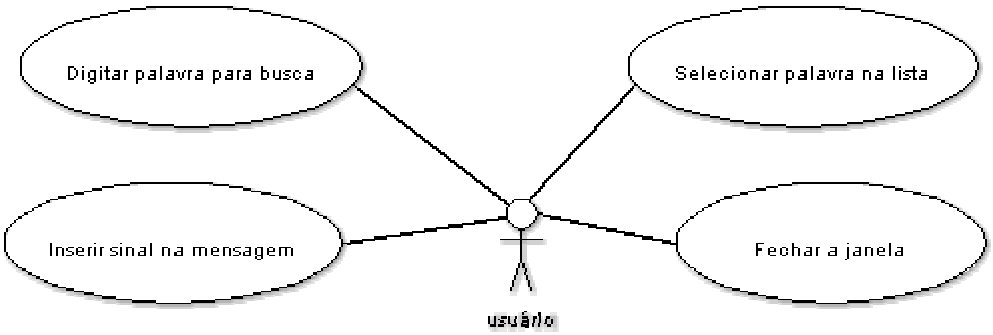


Figura 6.3 – Diagrama de caso de uso *dicPort*

O diagrama de caso de uso referente ao serviço *dicSW* é apresentado na figura 6.4. Este serviço possibilita consultas ao dicionário da SWService a partir da Libras. As ações possíveis para sua utilização são: selecionar um dos cinco grupos de símbolos, selecionar um dos subgrupos de símbolos referentes ao grupo escolhido, selecionar um dos símbolos pertencentes ao subgrupo escolhido, inserir um dos sinais exibidos como resultado da busca na mensagem (são exibidos todos os sinais que contenham em sua composição os símbolos selecionados), reiniciar a busca e fechar a janela de consulta ao dicionário.

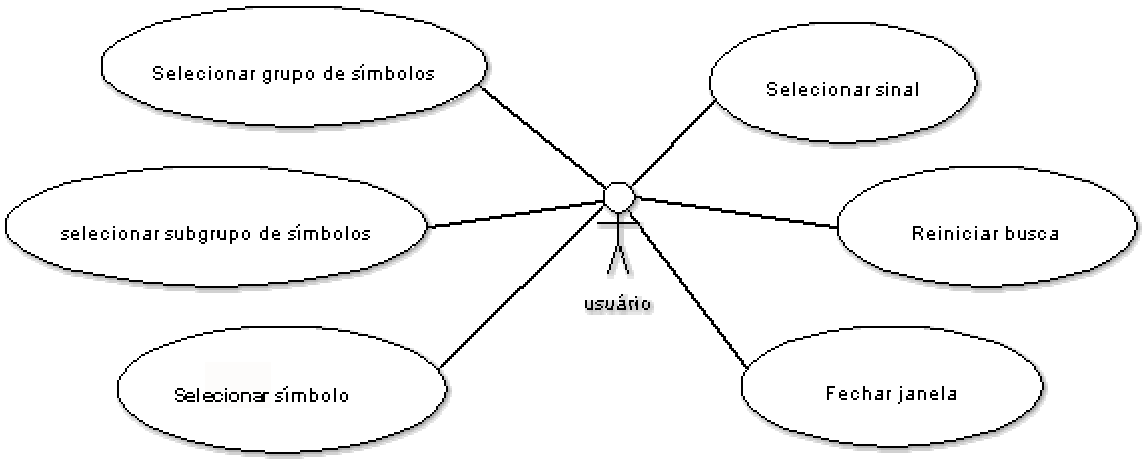


Figura 6.4 – Diagrama de caso de uso *dicSW*

A figura 6.5 apresenta o diagrama de caso de uso do serviço *createSign*, o qual possibilita a criação de novos sinais e a geração do arquivo SWML dos mesmos. As ações possíveis através do uso deste serviço podem ser divididas em dois grupos: ações para pesquisa dos símbolos e ações para edição do sinal na área de construção. As ações para pesquisa de símbolos são: selecionar um dos cinco grupos de símbolos, selecionar um dos subgrupos de símbolos referentes ao grupo escolhido e inserir um dos símbolos do subgrupo escolhido na área de construção do sinal. As ações para edição do sinal são: arrastar símbolos através de movimentos com o mouse, mover símbolos (para cima, para baixo, à esquerda e à direita) através de botões da interface, apagar símbolos, rotacionar os símbolos em torno do eixo Z, girar os símbolos em torno do próprio eixo, trocar a posição dos símbolos de horizontal para vertical e vice-versa, espelhar os símbolos e limpar sinal. Além disso, é possível associar uma palavra em português ao sinal criado, gravar o sinal ou fechar a janela para criação de sinais.

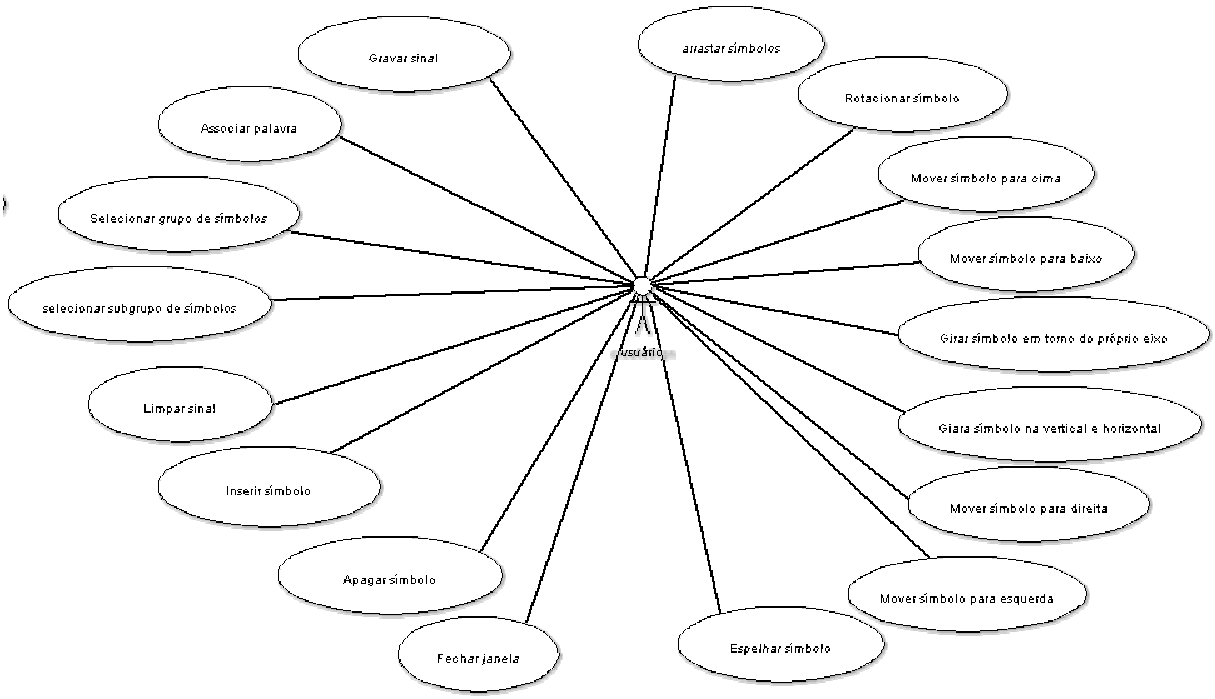


Figura 6.5 – Diagrama de caso de uso *createSign*

6.3 Modelo Entidade-Relacionamento

A figura 6.6 apresenta o modelo entidade-relacionamento referente ao banco de dados criado em PostgreSQL para uso pela biblioteca de serviços SWService. No modelo destaca-se a utilização de uma tabela chamada *Sign* para armazenamento de informações sobre os sinais disponibilizados pelo dicionário da SWService, tais como língua oral, grupo de símbolos, e a definição do sinal em SWML. Além dessa tabela principal, existem tabelas complementares para associação dos sinais com palavras em português e para estabelecimento do relacionamento entre os sinais e os símbolos utilizados em sua construção.

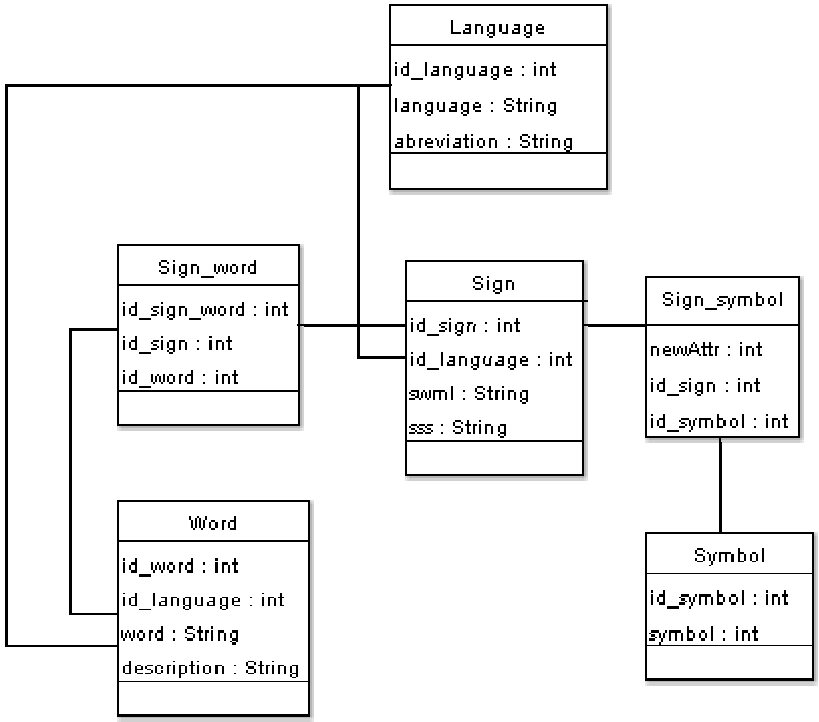


Figura 6.6 – Modelo Entidade-Relacionamento

6.4 Diagrama de classes

A figura 6.7 apresenta o modelo de classes da biblioteca SWservice, no qual destaca-se o uso do *Design Pattern Composite*. O *Composite* foi utilizado para a definição dos sinais, visto que é apropriado para compor objetos em estruturas de árvore para representarem hierarquias parte-todo. Assim, um sinal *Composite Sign*, pode ser composto por qualquer combinação de elementos que especializam *Component Image*.

Os serviços da biblioteca SWService consistem basicamente das classes *SWService*, *Dictionary* e *Language*. Os símbolos e sinais utilizados pela biblioteca são contemplados pelas classes *Image*, *Sign*, *Symbol* e as suas especializações, além das classes *Category* e *Group*. A definição dos sinais, a qual garante total interoperabilidade através do uso da linguagem SWML, é garantida pela classe *SWML* e as interações com banco de dados pela classe *DataBase*. A seguir, será apresentada uma breve descrição das principais classes do modelo.

A classe *SWService* corresponde à biblioteca propriamente dita e, assim, contém os métodos que representam os serviços oferecidos, além de métodos para geração dos retornos as chamadas dos serviços em formato HTML. Possui também métodos para obtenção da URL referente à localização da biblioteca e para obtenção do WSDL, o qual descreve a mesma.

A classe *Dictionary* representa o dicionário bilíngüe Português/Libras disponibilizado pela biblioteca. Esta classe possui vários métodos para realização de pesquisas no dicionário: palavras em português, sinais em Libras, palavras associadas a um sinal, sinais associados a uma palavra, dentre outras. Já a classe *Language* foi definida para representar as línguas envolvidas no dicionário, prevendo futura expansão do uso e desenvolvimento da SWService.

A classe *Image* contempla todas as imagens, sinais e símbolos, existentes na solução desenvolvida. Possui métodos para exibição da imagem, adição e remoção de imagens, obtenção de imagens filhas (das classes que a especializam) e obtenção de tamanho e cor da imagem. As classes *Symbol* e *Sign* especializam a classe *Image*, sendo a classe *Sign* responsável por representar os sinais criados a partir de composições da classe *Symbol*. Nesta parte da modelagem foi utilizado o *Design Pattern Composite* [GAM 00], através do qual um sinal *Composite Sign* pode ser composto por qualquer combinação de elementos que especializam *Component Image*.

A classe *Sign* representa a composição de símbolos utilizada para a criação dos sinais. Possui métodos para exibição, adição e remoção de imagens, obtenção de imagens filhas (das classes que especializam a classe *Component Image*) e obtenção do SSS, ou seja, a identificação do grupo de símbolos utilizados na composição do sinal. Para a SWService foi utilizado o grupo de símbolos chamado SSS2002, disponibilizado no ano de 2002.

Já a classe *Symbol* representa os símbolos utilizados para compor os sinais. Armazena informações sobre a classificação do símbolo: categoria, grupo, variação, rotação e posicionamento nos eixos X e Y. Possui métodos para exibição e exclusão do símbolo, além de métodos para variação: movimentação (para cima, para baixo, à direita e à esquerda), rotação e espelhamento. As classes que especializam a classe *Symbol* representam a forma como os símbolos estão classificados, ou seja, cinco grupos (representados pelas classes *Face*, *Body*, *Hand*, *Ponctuation* e *Movement*) e cada grupo com suas várias categorias.

Figura 6.7 - Diagrama de classes da SWService

6.5 Serviços

Foram desenvolvidos quatro serviços para a biblioteca SWService: *getSign* para tradução direta do português para Libras, *dicPort* para consultas ao dicionário em português, *dicSW* para consultas ao dicionário em Libras e *createSign* para criação de novos sinais. A seguir, cada um dos quatro serviços implementados será descrito detalhadamente.

6.5.1 getSign

Este serviço traduz, através da tradução direta, um texto do Português para Libras, em SignWriting. O formato da resposta, em Libras, pode ser tanto em HTML como em SWML dependendo do parâmetro passado para o serviço.

As informações principais para uso do serviço são:

- Nome: *getSign*
- URL do serviço: http://inf.unisinos.br/~swm/swservice.php
- Parâmetros de entrada:
  - texto em português
  - formato de saída desejado (HTML ou SWML)
  - número de colunas, ou seja, número de sinais a serem exibidos por linha
- Parâmetro de saída:
  - texto em Libras (HTML ou SWML)

A descrição do serviço, bem como os parâmetros de entrada e saída do mesmo, entre outras informações, são apresentados na tabela 6.1 e figura 6.8, que apresenta o HTML gerado automaticamente pela API NuSOAP.

Tabela 6.1 - Descrição do serviço *getSign*

Informação	**Valor**
Name	getSign
Binding	SWService-WSDLBinding
Endpoint	http://inf.unisinos.br/~swm/swservice.php
SoapAction	urn:SWService-WSDL#getSign
Style	Rpc
Input	use: encoded namespace: urn:SWService-WSDL encodingStyle: http://schemas.xmlsoap.org/soap/encoding/ message: getSignRequest parts: texto: xsd:string formato: xsd:string

	cols: xsd:string
Output	use: encoded namespace: urn:SWService-WSDL encodingStyle: http://schemas.xmlsoap.org/soap/encoding/ message: getSignResponse parts: libras: xsd:string
Namespace	urn:SWService-WSDL
Transport	http://schemas.xmlsoap.org/soap/http
Documentation	Este serviço traduz um texto do Português para Libras, em SignWriting. O formato da resposta, em Libras, pode ser HTML ou SWML dependendo do parâmetro passado para o serviço.

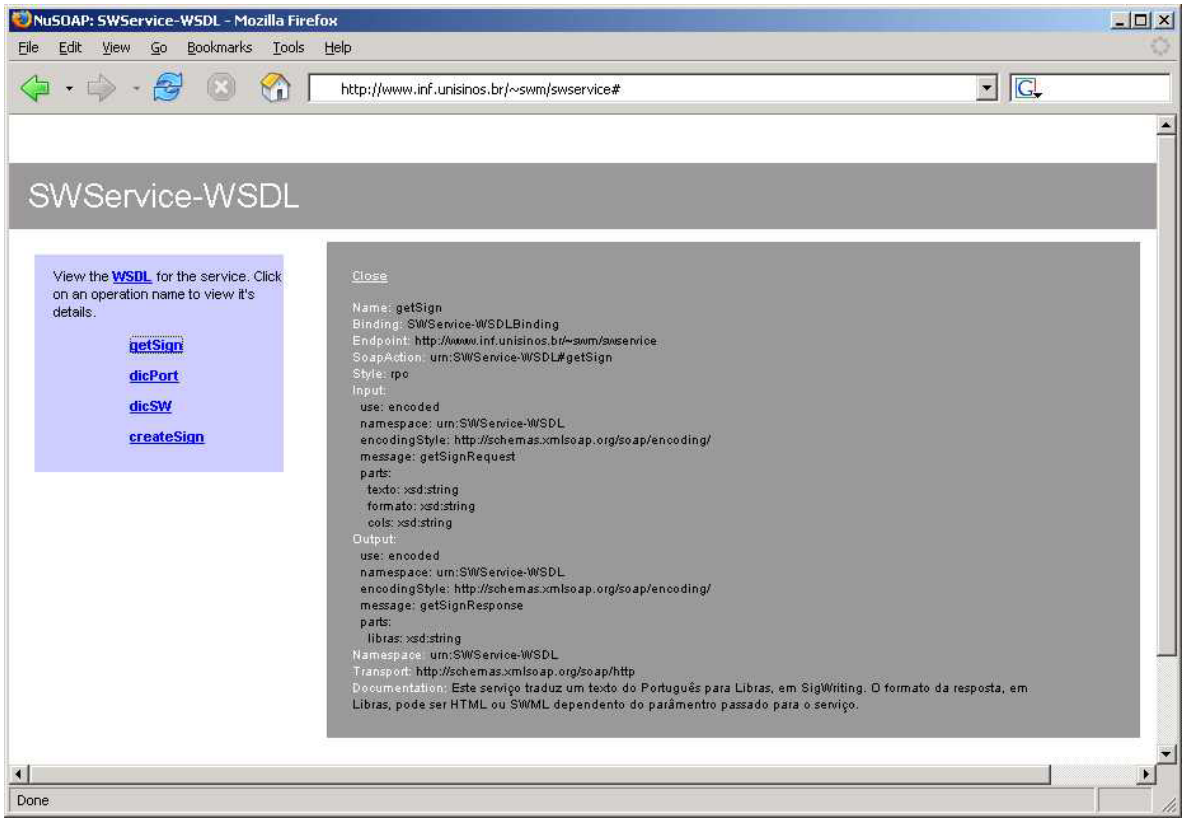


Figura 6.8 - Interface do serviço *getSign*

A seguir é apresentado o código-fonte, em PHP, com a definição do serviço *getSign*. A implementação está baseada no registro do serviço a ser oferecido, para geração do arquivo WSDL, e na implementação do serviço como uma função do PHP.

A implementação do serviço realiza, basicamente, uma pesquisa no dicionário de sinais da SWService para cada palavra do texto passado como parâmetro de entrada. Para isso, são utilizados os métodos *searchIdWord*, *searchSign*, e *datilology* da classe *dictionary* e o método *draw* da classe *Sign*. Caso o sinal seja encontrado no dicionário, o qual é armazenado em uma base de dados postgreSQL em formato padrão SWML, o método *draw* realiza o *parser* do SWML e retorna um código HTML que contém os símbolos utilizados na composição do sinal e o posicionamento de cada símbolo (coordenadas x e y) em uma *layer*.

Código-fonte para definição do seriço *getSign*

```
// registra o método a ser oferecido para geração do WSDL
$server->register('getSign',
   array('texto' => 'xsd:string', 'formato' => 'xsd:string', 'cols' => 'xsd:string'),
   array('libras' => 'xsd:string'),
     'urn:SWService-WSDL',
     'urn:SWService-WSDL#getSign',
     'rpc',
     'encoded',
     'Este serviço traduz um texto do Português para Libras, em SigWriting. O formato da
       resposta, em Libras, pode ser  HTML ou SWML dependendo do parâmetro passado para o
       serviço.'
);
// define o método a ser oferecido como uma função do PHP
function getSign($texto, $formato, $cols=3){
        // chama a biblioteca com as classes e métodos desenvolvidos para a SWService
        require_once ('swsevice_lib.php');
        $connpg = conectDB ();
        // define o Sign Symbols Sequence utilizado na SWService
        $GLOBALS['sss'] = "sss2002";
        // monta o HTML a ser retornado na variável $html_resp
        $html_resp = "<table border='0'>";
        // pesquisa cada palavra do texto passado no dicionário de sinais da SWService
        $palavras = split(" ",$texto);
        $dic = new dictionary();
        for ($i=0; $i<sizeof($palavras); $i++){
                $id_word = $dic->searchIdWord($connpg,$palavras[$i]);
                if ($i%$cols == 0) { $html_resp .= "</tr><tr>"; }
                $html_resp .= "<td align=center>";
                if ($id_word){
                        $id_sign = $dic->searchSign($connpg,$id_word);
                         $sign = new sign();
                        $html_resp .= $sign->draw($connpg,$id_sign,$formato);
                }else{
                        $datil = $dic->datilology($palavras[$i],$connpg);
                        $html_resp .= "<div id='Layer' style='position:relative; width:158px;
                        height:120px; z-index:1; background-color: #FFFFFF; layer-background-
                        color: #FFFFFF; border: 1px solid #000000'>".$datil."</div>";
                }
                $html_resp .= "".$palavras[$i]."</td>";
        }
        $html_resp .= "</table>";
        return $html_resp;
}
```

6.5.2 dicPort

Este serviço possibilita a consulta ao dicionário Libras/Português da SWService a partir da língua portuguesa. Através de alguns parâmetros passados ao serviço é possível obter uma página HTML para consultas ao dicionário de forma bastante simplificada e rápida.

As informações principais para uso do serviço são:

- Nome: *dicPort*

- URL do serviço: http://inf.unisinos.br/~swm/swservice.php
- Parâmetros de entrada:
  - URL do script que faz a chamada ao serviço
  - Cor de fundo a ser utilizada na página
  - URL do arquivo contendo a folha de estilos (CSS - *Cascading StyleSheet*) a ser utilizada
  - Imagem para utilização no cabeçalho da página
  - URL de destino para inserção das palavras referentes aos sinais consultados no dicionário
  - Parâmetro disponível para uso da aplicação que chama o serviço
  - Palavra referente ao sinal que se deseja inserir na mensagem
  - Parâmetros a serem definidos pelo o próprio serviço durante as consultas ao dicionário (*id_word*, *action* e *id_sign*).
- Parâmetro de saída:
  - HTML para montagem da página de consulta ao dicionário da SWService em português.

A descrição do serviço, bem como os parâmetros de entrada e saída do mesmo, entre outras informações, são apresentados pela tabela 6.2 e figura 6.9.

Tabela 6.2 - Descrição do serviço *dicPort*

Informação	**Valor**
Name	dicPort
Binding	SWService-WSDLBinding
Endpoint	http://inf.unisinos.br/~swm/swservice.php
SoapAction	urn:SWService-WSDL#dicPort
Style	rpc
Input	use: encoded namespace: urn:SWService-WSDL encodingStyle: http://schemas.xmlsoap.org/soap/encoding/ message: dicPortRequest parts: script: xsd:string background: xsd:string css: xsd:string action_form: xsd:string params: xsd:string image_head_software: xsd:string id_word: xsd:string action: xsd:string word: xsd:string id_sign: xsd:string

Output	use: encoded namespace: urn:SWService-WSDL encodingStyle: http://schemas.xmlsoap.org/soap/encoding/ message: dicPortResponse parts: html: xsd:string
Namespace	urn:SWService-WSDL
Transport	http://schemas.xmlsoap.org/soap/http
Documentation	Este serviço monta o HTML para consulta ao dicionário a partir do português.

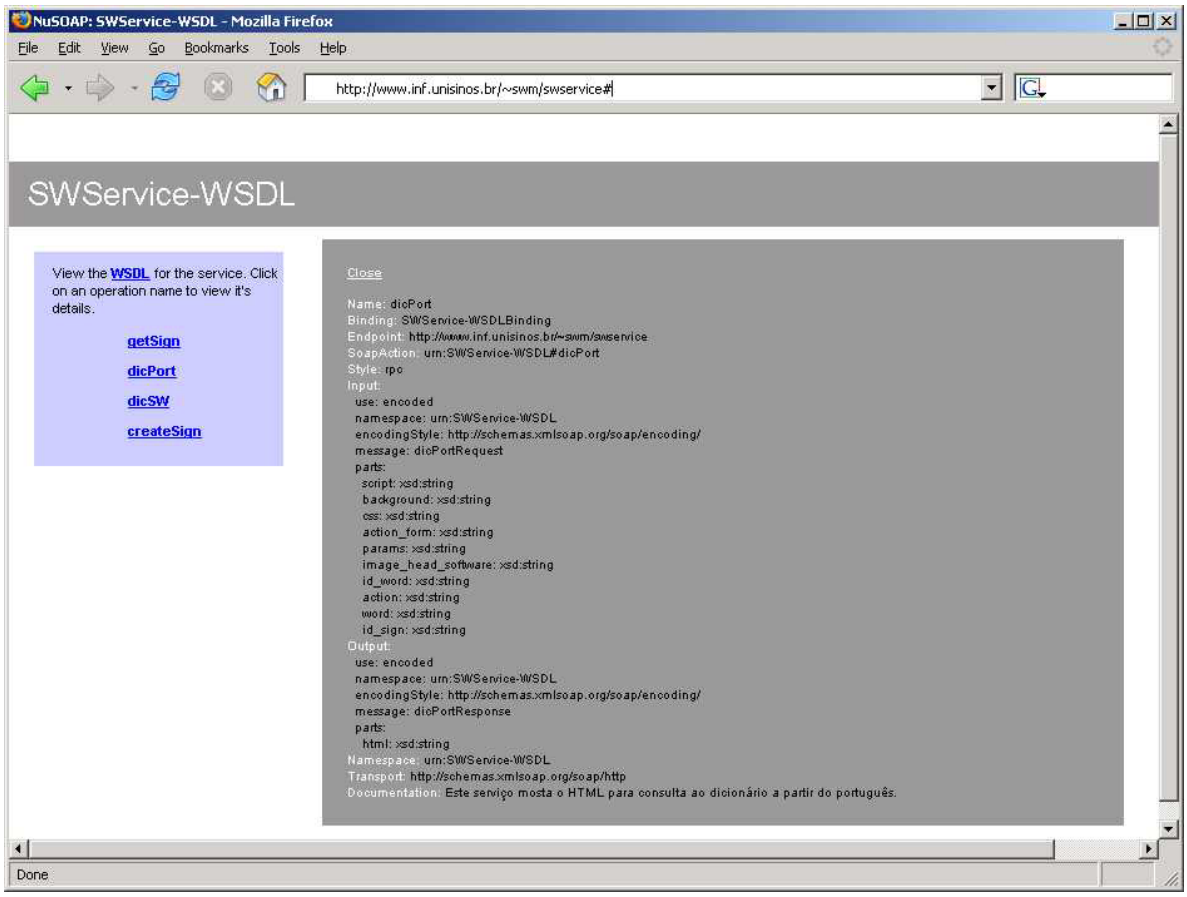


Figura 6.9 - Interface do serviço *dicPort*

A seguir é apresentado o código-fonte, em PHP, com a definição do serviço *dicPort*. A implementação tem como objetivo montar o código HTML para exibição da página de consulta ao dicionário bilíngüe da SWService. Para isso, são utilizados os métodos *searchIdWord, searchSign e searchWord* da classe *dictionay,* além do método *draw* da classe *Sign*.

Através da página gerada pelo serviço, é possível realizar consultas ao dicionário através de navegação da lista de entradas do dicionário ou através da digitação da palavra a ser pesquisada. Caso exista o sinal referente à palavra pesquisada, o mesmo é exibido na

página do próprio serviço. Após visualização do sinal, o usuário poderá incluí-lo na mensagem que está criando e continuar pesquisando outros sinais.

Código-fonte para definição do seriço *dicPort*

```
// registra o método a ser oferecido para geração do WSDL
$server->register('dicPort',
                  array(
                     'script'               => 'xsd:string',
                     'background'           => 'xsd:string',
                      'css'                 => 'xsd:string',
                      'action_form'         => 'xsd:string',
                      'params'              => 'xsd:string',
                      'image_head_software' => 'xsd:string',
                      'id_word'             => 'xsd:string',
                      'action'              => 'xsd:string',
                      'word'                => 'xsd:string',
                      'id_sign'             => 'xsd:string'

                   ), array(
                       'html'  => 'xsd:string'
                     ),
                     'urn:SWService-WSDL',
                     'urn:SWService-WSDL#dicPort',
                     'rpc',
                     'encoded',
                     'Este serviço monta o HTML para consulta ao dicionário a partir do
                     português.'
                  );

// define o método a ser oferecido como uma função do PHP
function dicPort($script,$background,$css,$action_form,$params,

                 $image_head_software,$id_word,$action,$word,$id_sign){

    // chama a biblioteca com as classes e métodos desenvolvidos para a SWService

    require_once ('swsevice_lib.php');

    $connpg = conectaDB();

    // define o Sign Symbols Sequence utilizado na SWService

    $GLOBALS['sss'] = "sss2002";

    // chama um método existente na swservice_lib que gera o HTML da página para consulta
    ao dicionário, o qual é retornado na variável $html_resp

    $html_resp = html_dic_port($connpg,$script,$background,$css,$action_form,$params,

                          $image_head_software,$id_word,$action,$word,$id_sign);

    return $html_resp;

}
```

6.5.3 dicSW

Ao contrário do *dicPort*, este serviço possibilita a consulta ao dicionário Libras/Português da SWService a partir da língua de sinais. Através dos parâmetros passados, o serviço é capaz de gerar uma página HTML para consultas ao dicionário a partir das características da escrita do sinal em Libras.

As informações principais para uso do serviço são:

- Nome: *dicSW*
- URL do serviço: http://inf.unisinos.br/~swm/swservice.php
- Parâmetros de entrada:
    - URL do script que faz a chamada ao serviço

- URL do arquivo contendo a folha de estilos (CSS - *Cascading StyleSheet*) a ser utilizada
- Imagem para utilização no cabeçalho da página
- URL de destino para inserção das palavras referentes aos sinais consultados no dicionário
- Parâmetro disponível para uso da aplicação que chama o serviço
- Id dos símbolos selecionados pelo usuário para pesquisas por sinais que contenham os mesmos
- Parâmetros a serem definidos pelo o próprio serviço durante as consultas ao dicionário (*symbols*, *action* e *btn*).

- Parâmetro de saída:
  - HTML para montagem da página de consulta ao dicionário da SWService em Libras

A descrição do serviço, bem como os parâmetros de entrada e saída do mesmo, entre outras informações, são apresentados na tabela 6.3 e na figura 6.10.

Tabela 6.3 - Descrição do serviço *dicSW*

Informação	**Valor**
Name	dicSW
Binding	SWService-WSDLBinding
Endpoint	http://inf.unisinos.br/~swm/swservice.php
SoapAction	urn:SWService-WSDL#dicSW
Style	rpc
Input	use: encoded namespace: urn:SWService-WSDL encodingStyle: http://schemas.xmlsoap.org/soap/encoding/ message: dicSWRequest parts: script: xsd:string css: xsd:string image_head_software: xsd:string action_form: xsd:string params: xsd:string action: xsd:string id_symbol: xsd:string symbols: xsd:string btn: xsd:string
Output	use: encoded namespace: urn:SWService-WSDL encodingStyle: http://schemas.xmlsoap.org/soap/encoding/ message: dicSWResponse

	parts: html: xsd:string
Namespace	urn:SWService-WSDL
Transport	http://schemas.xmlsoap.org/soap/http
Documentation	Este serviço monta o HTML para consulta ao dicionário a partir da Libras.

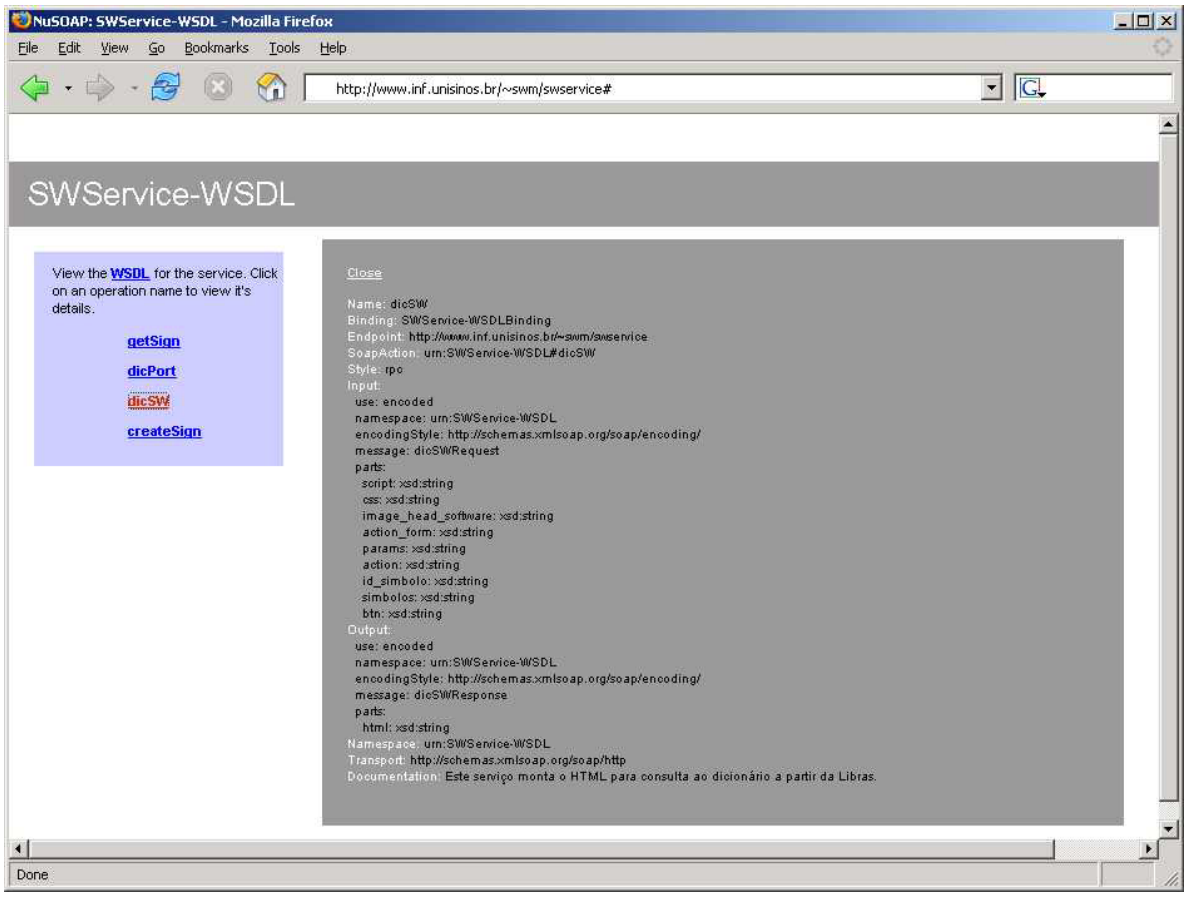


Figura 6.10 - Interface do serviço *dicSW*

A seguir é apresentado o código-fonte com a definição do serviço *dicSW*. A implementação tem como objetivo gerar o código HTML para exibição da página de consulta ao dicionário a partir da Libras. Para isso, são utilizados os métodos *wordsSign* da classe *dictionay e draw* da classe *Sign*.

Através da página gerada pelo serviço, é possível realizar consultas ao dicionário através da seleção dos símbolos utilizados na composição dos sinais. Os símbolos estão divididos em cinco grupos (face, corpo, mãos, pontuação e movimento) e, conforme o usuário vai selecionando os símbolos, são exibidos na lista de resultados da pesquisa todos os sinais que contém os símbolos selecionados.

Código-fonte para definição do seriço *dicSW*

```
// registra o método a ser oferecido para geração do WSDL
$server->register('dicSW',
                  array(
                      'script'               => 'xsd:string',
                      'css'                  => 'xsd:string',
                      'image_head_software'  => 'xsd:string',
                      'action_form'          => 'xsd:string',
                      'params'               => 'xsd:string',
                      'action'               => 'xsd:string',
                      'id_symbol'            => 'xsd:string',
                      'symbols'              => 'xsd:string',
                      'btn'                  => 'xsd:string'
                   ), array(
                       'html'  => 'xsd:string'
                     ),
                     'urn:SWService-WSDL',
                     'urn:SWService-WSDL#dicSW',
                     'rpc',
                     'encoded',
                     'Este serviço monta o HTML para consulta ao dicionário a partir da
                     Libras.'
                 );

// define o método a ser oferecido como uma função do PHP
function dicSW($script,$css,$image_head_software,$action_form,$params,
              $action,$id_symbol,$symbols,$btn){
    // chama a biblioteca com as classes e métodos desenvolvidos para a SWService
    require_once ('swsevice_lib.php');
    $connpg = conectaDB();
    // define o Sign Symbols Sequence utilizado na SWService
    $GLOBALS['sss'] = "sss2002";
    // chama um método existente na swservice_lib que gera o HTML da página para consulta
    ao dicionário, o qual é retornado na variável $html_resp
    $html_resp = html_dic_sw($connpg,$script,$css,$image_head_software,$action_form,
                             $params,$action,$id_symbol,$symbols,$btn);
    return $html_resp;
}
```

6.5.4 createSign

O serviço chamado *createSign* é um dos mais importantes e complexos da SWService. Tem como objetivo possibilitar a criação e edição de novos sinais através de interface web. Para tornar esta serviço possível, foram utilizadas diversas funções criadas na linguagem JavaScript, a qual possibilita uma séria de manipulações de eventos do *browser*.

As informações necessárias para uso do serviço são:

- Nome: *createSign*
- URL do serviço: http://inf.unisinos.br/~swm/swservice.php
- Parâmetros de entrada:
  - URL do script que faz a chamada ao serviço
  - URL do arquivo contendo a folha de estilos (CSS - *Cascading StyleSheet*) a ser utilizada
  - Imagem para utilização no cabeçalho da página
  - URL de destino para o código SWML gerado
  - Parâmetro disponível para uso da aplicação que chama o serviço
  - Palavras associadas ao sinal criado

- Parâmetros a serem definidos pelo o próprio serviço durante as consultas ao dicionário (*action, id_symbols, symbols* e *btn*).

- Parâmetros de saída:
  - HTML para montagem da página que possibilita a criação e edição de sinais
  - SWML gerado na criação ou edição do sinal
  - Palavras associadas ao sinal

A descrição do serviço, bem como os parâmetros de entrada e saída do mesmo, entre outras informações, são apresentados na tabela 6.4 e na figura 6.11.

Tabela 6.4 - Descrição do serviço *createSign*

Informação	**Valor**
Name	createSign
Binding	SWService-WSDLBinding
Endpoint	http://inf.unisinos.br/~swm/swservice.php
SoapAction	urn:SWService-WSDL#createSign
Style	rpc
Input	use: encoded namespace: urn:SWService-WSDL encodingStyle: http://schemas.xmlsoap.org/soap/encoding/ message: createSignRequest parts: script: xsd:string css: xsd:string image_head_software: xsd:string action_form: xsd:string params: xsd:string action: xsd:string id_symbol: xsd:string symbols: xsd:string btn: xsd:string words: xsd:string
Output	use: encoded namespace: urn:SWService-WSDL encodingStyle: http://schemas.xmlsoap.org/soap/encoding/ message: createSignResponse parts: html: xsd:string swml: xsd:string word: xsd:string

Namespace	urn:SWService-WSDL
Transport	http://schemas.xmlsoap.org/soap/http
Documentation	Este serviço monta o HTML para criação e edição de sinais em SignWriting.

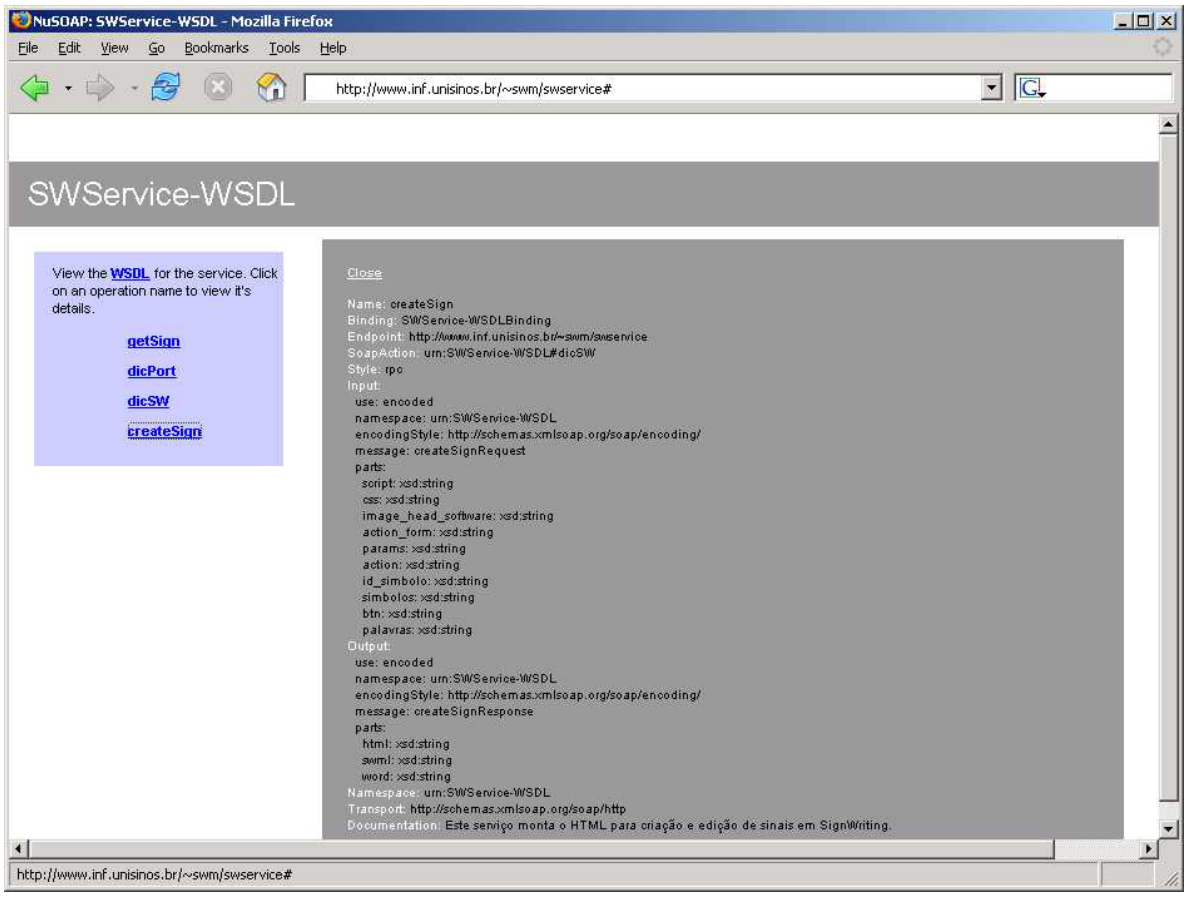


Figura 6.11 - Interface do serviço *createSign*

A seguir é apresentado o código-fonte para o serviço *createSign*. A implementação tem como objetivo gerar o código HTML para exibição da página com as funcionalidades para a criação e edição de sinais. Para isso, são utilizados os métodos *startElement, endElement e dataElement* da classe *SWML* e *saveSign* da classe *Dictionay*, além de várias funções em JavaScript.

Através da página gerada pelo serviço, é possível criar sinais através de uma interface muito semelhante a interface desenvolvida para a consulta ao dicionário em Libras. Os principais recursos disponibilizados para criação dos sinais são: diminuição considerável do número de símbolos na interface, possibilidade de obter qualquer variação de um símbolo base a partir de quatro botões de ação na interface, possibilidade de movimentação dos símbolos na área de construção dos sinais através de movimentos com o mouse e possibilidade de associação de mais de uma palavra da língua portuguesa com um sinal criado pelo serviço.

Código-fonte para definição do seriço *createSign*

```
// registra o método a ser oferecido para geração do WSDL
$server->register('createSign',
                  array(
                      'script'               => 'xsd:string',
                      'css'                  => 'xsd:string',
                      'image_head_software'  => 'xsd:string',
                      'action_form'          => 'xsd:string',
                      'params'               => 'xsd:string',
                      'action'               => 'xsd:string',
                      'id_symbol'            => 'xsd:string',
                      'symbols'              => 'xsd:string',
                      'btn'                  => 'xsd:string',
                      'words'                => 'xsd:string'
                    ), array(
                      'html'  => 'xsd:string',
                      'swml'  => 'xsd:string',
                      'word'  => 'xsd:string'
                     ),
                      'urn:SWService-WSDL',
                      'urn:SWService-WSDL#createSign',
                      'rpc',
                      'encoded',
                      'Este serviço monta o HTML para criação e edição de sinais em
                      SignWriting.'
                  );

// define o método a ser oferecido como uma função do PHP
function createSign($script,$css,$image_head_software,$action_form,$params,
              $action,$id_symbol,$symbols,$btn,$words){
    // chama a biblioteca com as classes e métodos desenvolvidos para a SWService
    require_once ('swsevice_lib.php');
    $connpg = conectaDB();
    // define o Sign Symbols Sequence utilizado na SWService
    $GLOBALS['sss'] = "sss2002";
    // chama um método existente na swservice_lib que gera o HTML da página para criação e
    edição de sinais, gera o SWML do sinal criado e retorna as palavras associadas ao sinal
    bem como os outros itens em um array ($array_resp)
    $array_resp = html_dic_sw($connpg,$script,$css,$image_head_software,$action_form,
                              $params,$action,$id_symbol,$symbols,$btn);
    return $array_resp;
}
```

6.6 Exemplos de uso

A seguir, são apresentadas quadro implementações de clientes acessando os serviços desenvolvidos para a biblioteca SWService. Os principais componentes das implementações são: inclusão da biblioteca de classes NuSOAP, definição da localização do arquivo WSDL com a descrição e definição dos serviços, criação de um objeto da classe *soapclient* (da API NuSOAP), definição dos parâmetros de entrada para o serviço, chamada do serviço e impressão dos resultados.

Além do código-fonte de cada uma das implementações, são apresentadas figuras com o resultado visual das chamadas aos serviços.

6.6.1 *getSign*

Código-fonte exemplo de chamada ao seriço *getSign*

```
<?
        // inclusão do arquivo de classes NuSOAP
        require_once('nusoap.php');
        // definição da localização do WSDL
        $wsdl = "http://www.inf.unisinos.br/~swm/swservice.php?wsdl";
        // criação da instância do cliente
        $client = new soapclient($wsdl, true);
        $return = $client->call('getSign',
                array('texto'  => 'VINÍCIUS IDÉIA DICIONÁRIO SURDO',
                    'formato'  => 'html',
                       'cols'  => 2));
        // verifica se ocorreu falha na chamada do método
        if ($client->fault){
                echo "<h2>Fault</h2>
                print_r($return);
        }else{
                // verifica se ocorreu erro na execução do método
                $err = $client->getError();
                if ($err){
                        echo "<h2>Error</h2>".$err;
                }else{
                        // exibe o resultado
                        echo "<h2>Result</h2>";
                        print_r($return);
                }
        }
        // exibe a requisição e a resposta
        echo '<h2>Requisição</h2>';
        echo '<pre>' . htmlspecialchars($client->request) . '</pre>';
        echo '<h2>Resposta</h2>';
        echo '<pre>' . $client->response . '</pre>';
?>
```

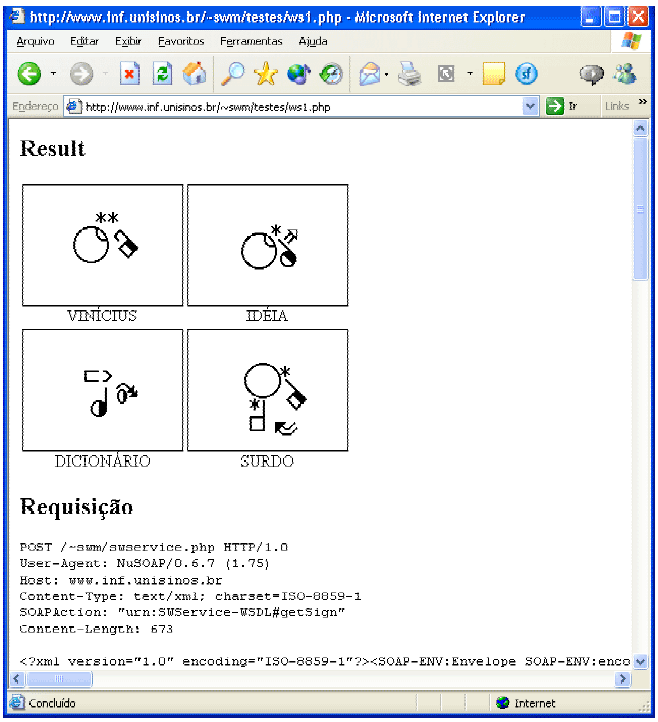


Figura 6.12 – Exemplo de uso do serviço *getSing*

6.6.2 *dicPort*

Código-fonte exemplo de chamada ao seriço *dicPort*

```
<?
        // inclusão do arquivo de classes NuSOAP
        require_once('nusoap.php');
        // definição da localização do WSDL
        $wsdl = "http://www.inf.unisinos.br/~swm/swservice.php?wsdl";
        // criação da instância do cliente
        $client = new soapclient($wsdl, true);
        $params = "tema_id=$tema_id";
        $return =  $client->call('dicPort',
                array(      'script' => 'http://www.inf.unisinos.br/~swm/testes/ws2.php',
                        'background' => '#FFEFD6',
                               'css' => 'http://www.inf.unisinos.br/~swm/_css/swm_ie6.css',
                       'action_form' => 'http://www.inf.unisinos.br/~swm/teste.php',
                            'params' => $params,
                'image_head_software' => 'dicPort',
                          'id_word' => $id_word,
                           'action' => $action,
                             'word' => $word,
                         'id_sign' => $id_sign));
        // verifica se ocorreu falha na chamada do método
        if ($client->fault){
                echo "Falha: ".$return;
        }else{
                // verifica se ocorreu erro
                $err = $client->getError();
                if ($err){
                        echo "Erro: ".$err;
                }else{
                        // exibe o resultado
                        echo $return;
                }
        }
?>
```

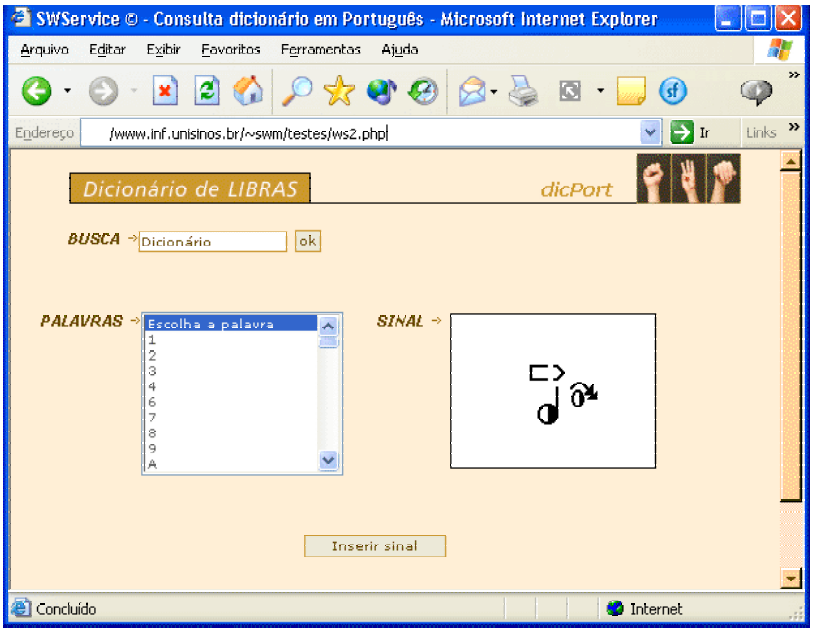


Figura 6.13 – Exemplo de uso do serviço *dicPort*

6.6.3 *dicSW*

Código-fonte exemplo de chamada ao seriço *dicSW*

```
<?
        session_start();
        if (!$action){
                $_SESSION['sbs'] = "";
        }elseif ($id_simbolo != ""){
                $_SESSION['sbs'] .= $id_simbolo."#";
        }
        $simbolos = $_SESSION['sbs'];
        // inclusão do arquivo de classes NuSOAP
        require_once('nusoap.php');
        // definição da localização do WSDL
        $wsdl = "http://www.inf.unisinos.br/~swm/swservice.php?wsdl";
        // criação da instância do cliente
        $client = new soapclient($wsdl, true);
        $return = $client->call('dicSW',
                array('script' => 'http://www.inf.unisinos.br/~swm/testes/ws3.php',
                        'css' => 'http://www.inf.unisinos.br/~swm/_css/swm_ie6.css',
        'image_head_software' => 'DicSW',
                 'action_form' => 'http://www.inf.unisinos.br/~swm/teste.php',
                      'params' => $params,
                      'action' => $action,
                   'id_simbolo' => $id_symbol,
                     'simbolos' => $symbols,
                         'btn' => $btn));
        // verifica se ocorreu falha na chamada do método
        if ($client->fault){
                echo "Falha: ".$return;
        }else{
                // verifica se ocorreu erro
                $err = $client->getError();
                if ($err){
                        echo "Erro: ".$err;
                }else{
                        echo $return;
                }
        }
?>
```

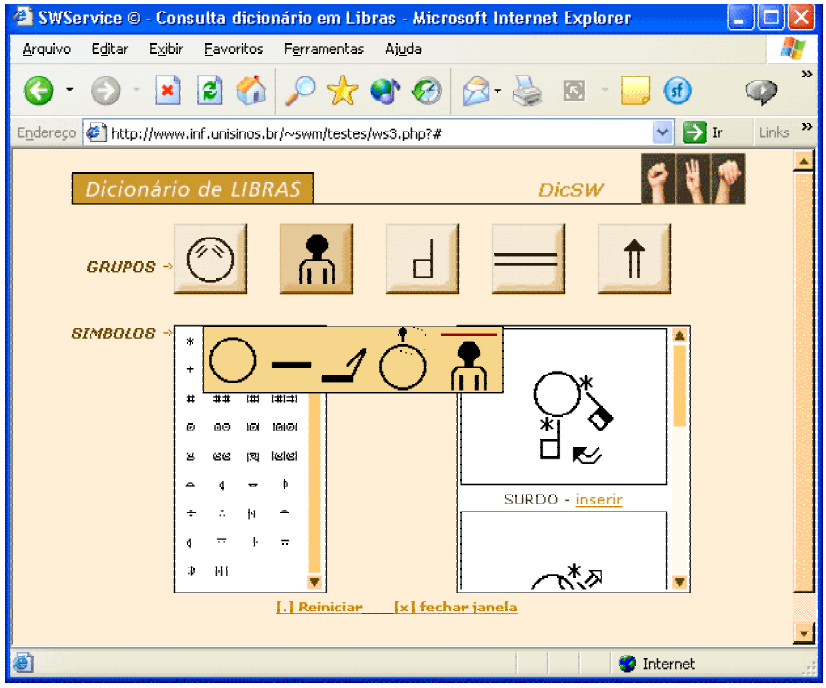


Figura 6.14 – Exemplo de uso do serviço *dicSW*

6.6.4 *createSign*

Código-fonte exemplo de chamada ao seriço *createSign*

```
<?
        // inclusão do arquivo de classes NuSOAP
        require_once('nusoap.php');
        // definição da localização do WSDL
        $wsdl = "http://www.inf.unisinos.br/~swm/swservice.php?wsdl";
        // criação da instância do cliente
        $client = new soapclient($wsdl, true);
        $return = $client->call('createSign',
                array('script' => 'http://www.inf.unisinos.br/~swm/testes/ws4.php',
                        'css' => 'http://www.inf.unisinos.br/~swm/_css/swm_ie6.css',
         'image_head_software' => 'createSign',
                  'action_form' => 'http://www.inf.unisinos.br/~swm/teste.php',
                       'params' => $params,
                       'action' => $action,
                   'id_simbolo' => $id_symbol,
                     'simbolos' => $symbols,
                          'btn' => $btn,
                    'palavras' => $palavras));
        // verifica se ocorreu falha na chamada do método
        if ($client->fault){
                echo "<h2>Fault</h2>";
                print_r($return);
        }else{
                // verifica se ocorreu erro
                $err = $client->getError();
                if ($err){
                        echo "<h2>Error</h2><pre>".$err."</pre>";
                }else{
                        if ($return['swml'] != ""){
                                echo "<b>PALAVRA:</b> ";
                                print_r($return['word']);
                                echo "<br><bR><b>SWML:</b><br><br>";
                                print_r($return['swml']);
                        }else{
                                print_r($return['html']);
                        }
                }
        }
?>
```

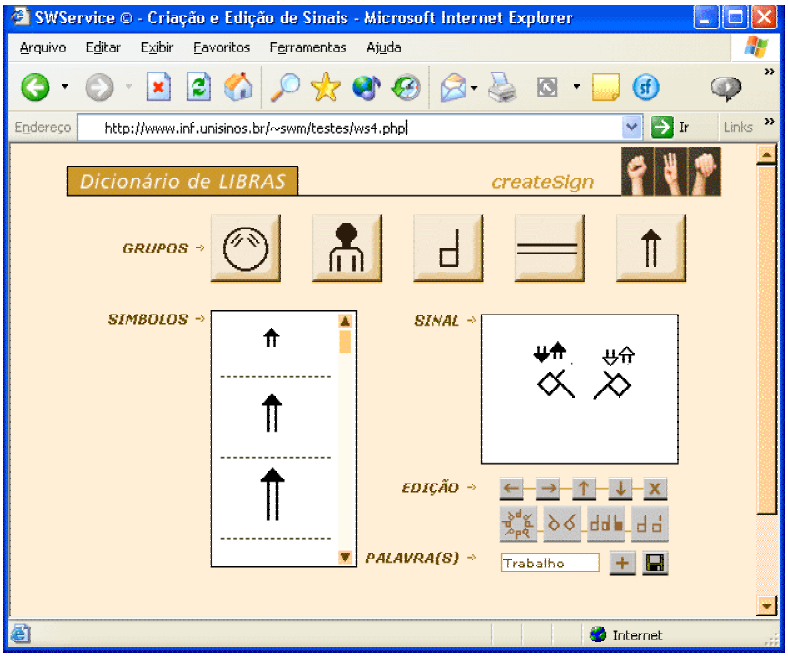


Figura 6.15 – Exemplo de uso do serviço *createSing*

7 Estudo de caso

Com o objetivo de testar os Web Services desenvolvidos, foi realizado um estudo de caso no qual os mesmos foram utilizados para tornar uma ferramenta web acessível aos surdos através da possibilidade leitura e escrita das mensagens em Libras. A ferramenta em questão trata-se de um fórum de discussão chamado Sign WebForum [ROS 04a], o qual foi desenvolvido em colaboração com a aluna de graduação em Análise de Sistemas da Unisinos, Daniela Rossi, como parte integrante de seu trabalho de conclusão de curso.

7.1 Metodologia

A metodologia utilizada no estudo de caso foi baseada no desenvolvimento de uma aplicação web, utilização dos serviços disponibilizados pela SWService na mesma e avaliação da ferramenta por um grupo de pessoas ouvintes e por um de surdos. A avaliação teve como objetivo averiguar as percepções quanto ao uso geral da aplicação e especificamente quanto as funcionalidades provenientes da biblioteca SWService.

A seguir, será apresentado o Sign WebForum bem como seu processo de desenvolvimento e os resultados da avaliação realizada com surdos e ouvintes.

7.1.1 Sign WebForum

O Sign WebForum tem como objetivo permitir a um grupo de pessoas, com interesses comuns, a troca de informações e o debate de idéias tanto em português como em Libras através do sistema SignWriting.

O software (http://www.inf.unisinos.br/swf) possui dois módulos: módulo principal, onde estão as funções comuns relacionadas à ferramenta de fórum (visualização de mensagens, seleção e leitura das mensagens, envio de novos temas e/ou comentários, entre outros) e o módulo de integração, correspondente a integração do mesmo às funcionalidades de leitura, escrita e consulta de sinais disponibilizadas através da biblioteca SWService.

Em seu desenvolvimento, foi utilizado PHP para programação, banco de dados PostgreSQL e servidor web Apache, pois tratam-se de ferramentas não proprietárias e de ampla utilização na Internet [ROS 04b].

Cabe salientar que durante todo o projeto e desenvolvimento do software os esforços foram concentrados unicamente na natureza do problema que se desejava tratar, ou seja, o desenvolvimento de um fórum de discussão. Em nenhum momento do desenvolvimento foi necessário abordar questões sobre as funcionalidades necessárias para o uso da Libras. Apenas depois de implementar e testar o funcionamento do fórum de discussão, exclusivamente em português, é que se fez a incorporação de chamadas aos Web Services da biblioteca SWService com o objetivo de tornar possível a leitura e escrita das mensagens do fórum também em Libras.

Os três serviços, da SWService, utilizados no Sign WebForum foram: *getSign*, *dicPort* e *dicSW*. A seguir, são apresentados os parâmetros passados para cada serviço e o código fonte para chamada dos mesmos. Logo após, são apresentadas as interfaces do Sign WebForum, nas quais é possível destacar as funcionalidades obtidas pela utilização da biblioteca SWService, bem como uma explicação sobre funcionamento da ferramenta.

A tabela 7.1 apresenta os parâmetros passados ao serviço *getSign*. Logo abaixo é apresentado o código-fonte referente à utilização do serviço pelo Sign WebForum.

Tabela 7.1 – Parâmetros passados para o serviço *getSign*

Parâmetro	valor
text	Texto passado dinamicamente através de consultas ao banco de dados
format	html
cols	3

Código-fonte para chamada ao seriço *getSign* pelo Sign WebForum

```php
<?php
        // inclusão do arquivo de classes NuSOAP
        require_once('nusoap.php');
        // definição da localização do WSDL
        $wsdl = "http://www.inf.unisinos.br/~swm/swservice.php?wsdl";
        // criação da instância do cliente
        $client = new soapclient($wsdl, true);
        //chamada do seriço
        echo $client->call('getSign',  array('text'   => $mensagem_texto,
                                             'format' => 'html',
                                             'cols'   => '3'));
?>
```

Os parâmetros passados ao serviço *dicPort* são apresentados na tabela 7.2. O código-fonte referente a chamada deste serviço é apresentado logo a seguir.

Tabela 7.2 – Parâmetros passados para o serviço *dicPort*

Parâmetro	Valor
script	http://www.inf.unisinos.br/~swm/signforum/dic_port.php
background	#dbe8e3
css	http://www.inf.unisinos.br/~swm/signforum/css/sf_ie6.css
action_form	http://www.inf.unisinos.br/~swm/signforum/swf_5cinema.php
params	$params
image_head_software	Sign WebForum
id_word	$id_word
action	$action
word	$word
id_sign	$id_sign

Código-fonte para chamada ao seriço *dicPort* pelo Sign WebForum

```php
<?php
        // inclusão do arquivo de classes NuSOAP
        require_once('nusoap.php');
        // definição da localização do WSDL
        $wsdl = "http://www.inf.unisinos.br/~swm/swservice.php?wsdl";
        // criação da instância do cliente
        $client = new soapclient($wsdl, true);
        $params = "tema_id=$tema_id";
        echo $client->call('dicPort', array(
                    'script' => 'http://www.inf.unisinos.br/~swm/signforum/dic_port.php',
                'background' => '#dbe8e3',
                       'css' => 'http://www.inf.unisinos.br/~swm/signforum/css/sf_ie6.css',
                'action_form' => 'http://www.inf.unisinos.br/~swm/signforum/swf_5cinema.php',
                     'params' => $params,
        'image_head_software' => 'Sign WebForum',
                   'id_word' => $id_word,
                    'action' => $action,
                      'word' => $word,
                   'id_sign' => $id_sign));
?>
```

A tabela 7.3 apresenta os parâmetros passados pelo Sign WebForum para chamada ao serviço *dicSW*. A seguir, é apresentado o código-fonte referente à chamada deste serviço.

Tabela 7.3 – Parâmetros passados para o serviço *dicSW*

Parâmetro	valor
script	http://www.inf.unisinos.br/~swm/signforum/dic_sw.php
css	http://www.inf.unisinos.br/~swm/signforum/css/sf_ie6.css
image_head_software	Sign WebForum
action_form	http://www.inf.unisinos.br/~swm/signforum/swf_5cinema.php
params	$params
action	$action
id_symbol	$id_symbol
symbols	$symbols
btn	$btn

Código-fonte para chamada ao seriço *dicSW* pelo Sign WebForum

```
<?php
  session_start();
  if (!$action){
      $_SESSION['sbs'] = "";
  }elseif ($id_simbolo != ""){
      $_SESSION['sbs'] .= $id_simbolo."#";
  }
  $simbolos = $_SESSION['sbs'];
  // inclusão do arquivo de classes NuSOAP
  require_once('nusoap.php');
  // definição da localização do WSDL
  $wsdl = "http://www.inf.unisinos.br/~swm/swservice.php?wsdl";
  // criação da instância do cliente
  $client = new soapclient($wsdl, true);
  $params = "tema_id=$tema_id";
  echo $client->call('dicSW',   array(
                    'script' => 'http://www.inf.unisinos.br/~swm/signforum/dic_sw.php',
                       'css' => 'http://www.inf.unisinos.br/~swm/signforum/css/sf_ie6.css',
        'image_head_software' => 'Sign WebForum',
               'action_form' => 'http://www.inf.unisinos.br/~swm/signforum/swf_5cinema.php',
                    'params' => $params,
                    'action' => $action,
                 'id_simbolo' => $id_symbol,
                   'simbolos' => $symbols,
                       'btn' => $btn));
?>
```

A partir do momento em que o usuário acessa o sistema, é exibida a tela principal (figura 7.1) do Sign WebForum.

À esquerda, constam as ações possíveis de serem realizadas no fórum (consultar *Ajuda*, *Definição*, *Uso*, *Contato* e as quatro opções de fórum ofertadas - *Cinema, Informática, Música e Variedades*).

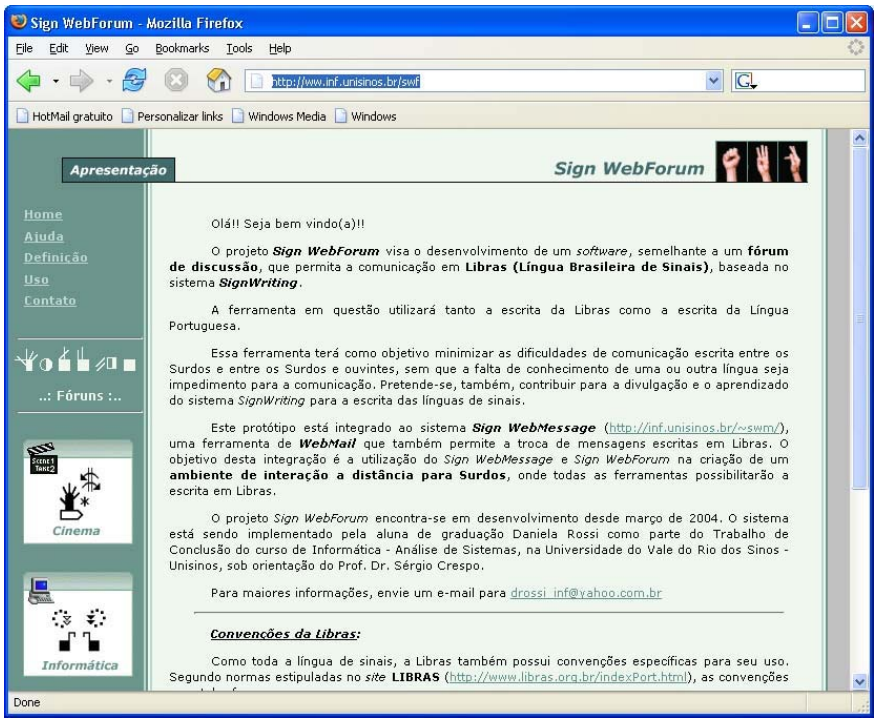


Figura 7.1 – Página inicial do Sign WebForum

Ao selecionar uma mensagem específica no fórum, é possível executar 4 ações distintas: visualizar a mensagem escrita em português, visualizar a mensagem escrita em Libras, escrever um novo comentário em português ou escrever um novo comentário em Libras (figura 7.2).

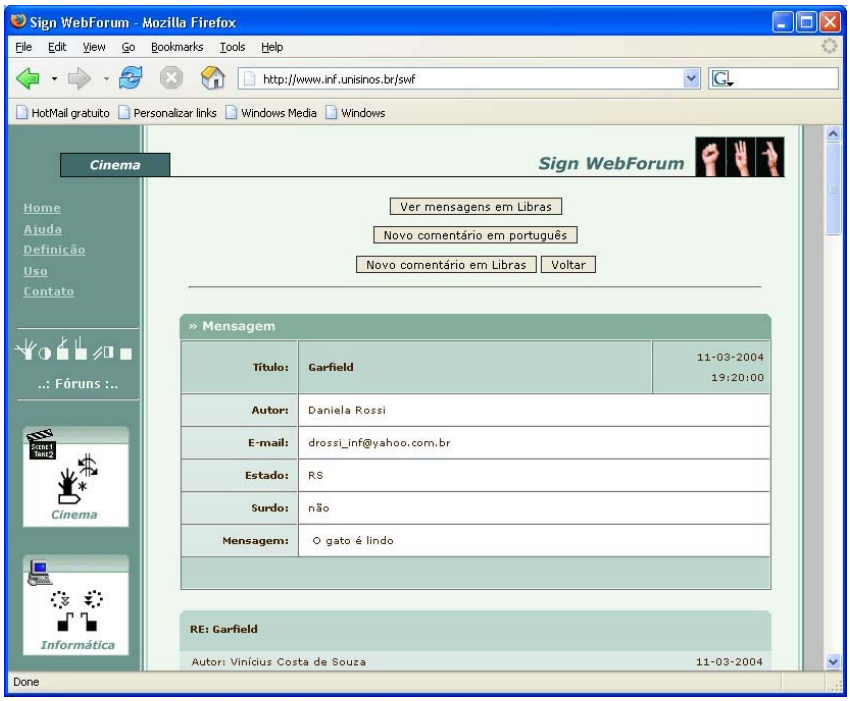


Figura 7.2 – Lendo uma mensagem em português

Se o usuário selecionar a opção para visualizar a mensagem em Libras, a mesma é exibida com o cabeçalho em português e seu texto propriamente dito (somente as palavras que possuem correspondência em Libras no dicionário do sistema) em sinais (figura 7.3). Essa funcionalidade foi obtida através do uso do serviço *getSign*.

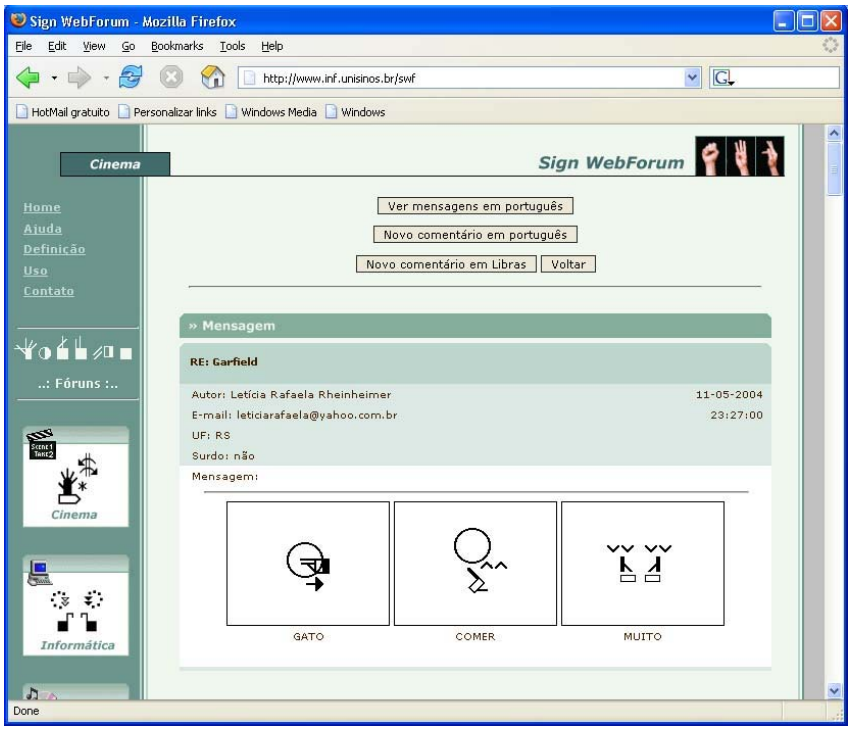


Figura 7.3 - Lendo uma mensagem em Libras.

Para enviar uma mensagem para o fórum, os usuários também poderão optar pela língua desejada: português (figura 7.4) ou Libras (figura 7.5). No caso da Libras, é possível consultar o dicionário de sinais tanto a partir da língua portuguesa, através do uso do serviço *dicPort*, quanto através da própria Libras, serviço *dicSW*.

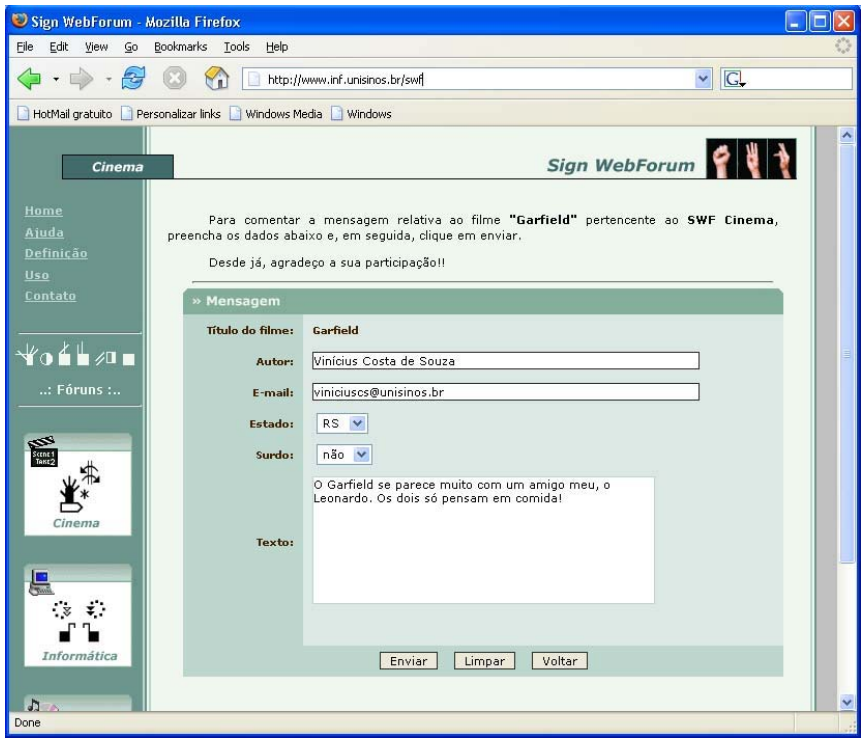


Figura 7.4 - Escrevendo uma mensagem em português

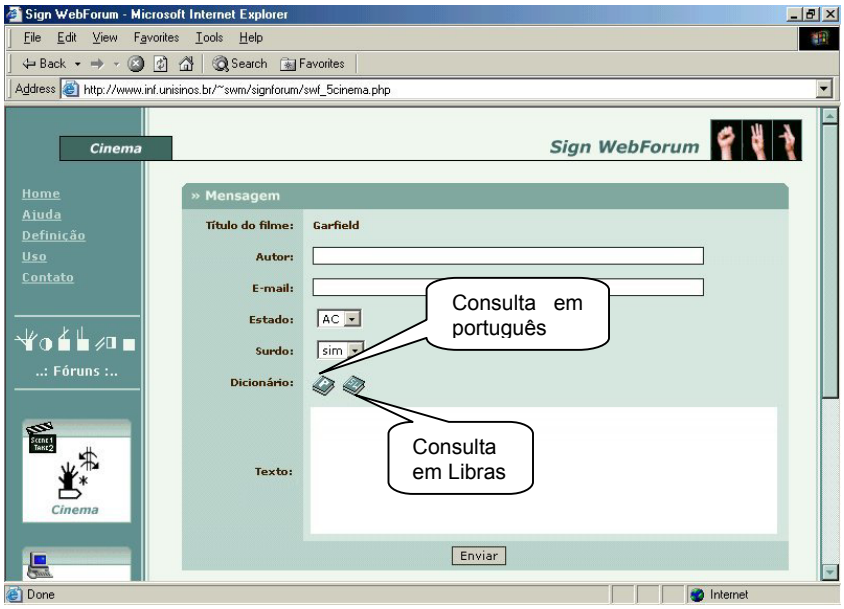


Figura 7.5 - Escrevendo uma mensagem em Libras

A figura 7.6 apresenta as das duas formas de busca aos sinais do dicionário, utilizando a integração e interoperabilidade garantidas pelo uso dos serviços da SWService.

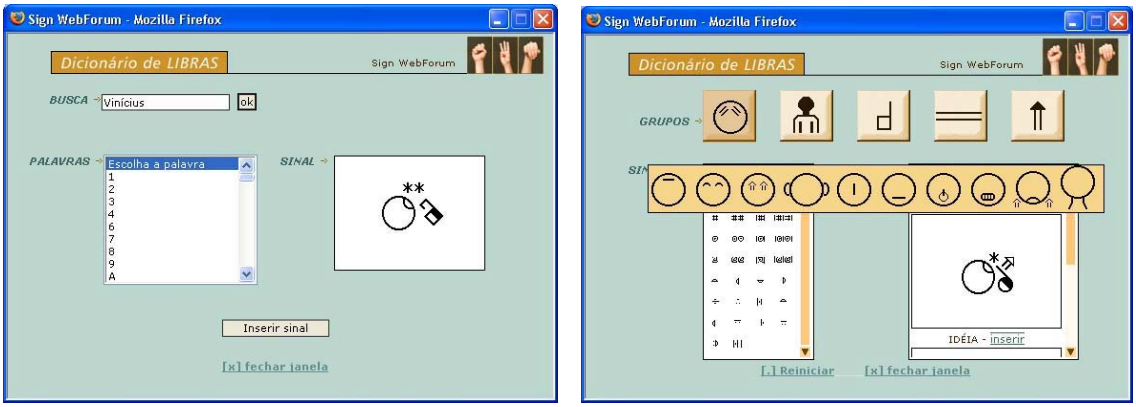


Figura 7.6 - Formas de consulta ao dicionário de sinais

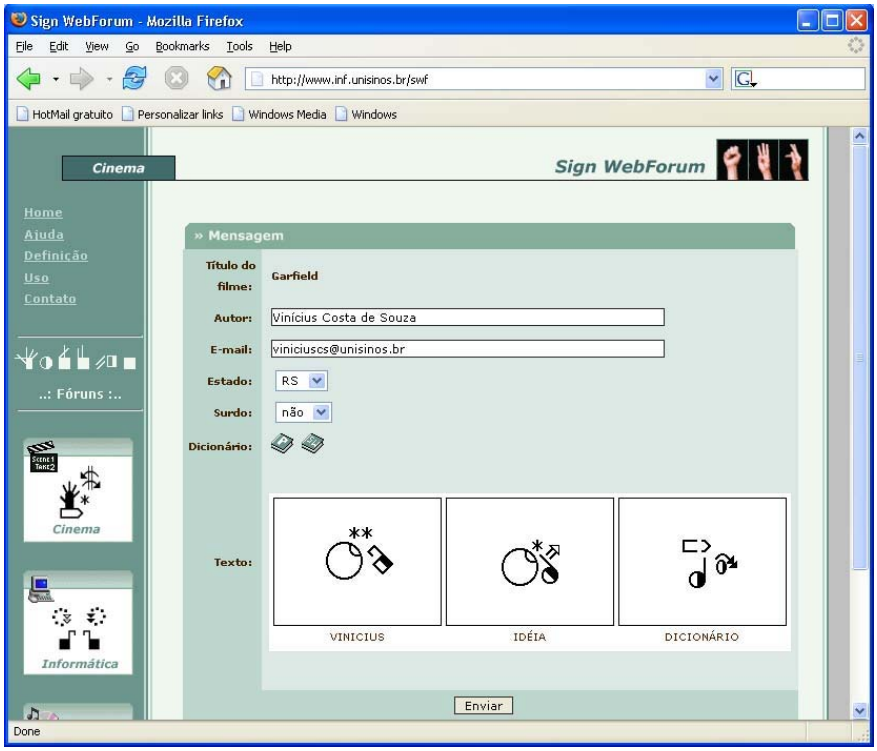


Figura 7.7 - Mensagem em Libras

7.1.2 Avaliação

Com o objetivo de validar as funcionalidades nativas do Sign WebForum e as obtidas através do uso dos serviços da biblioteca SWService, foi realizada uma breve pesquisa de opinião com um grupo de cinco surdos, conhecedores da Libras e do sistema SignWriting, e com um grupo de dez ouvintes os quais não conheciam Libras.

A avaliação foi baseada nos mesmos princípios da pesquisa realizada com o Sign WebMessage (capítulo 4), ou seja, primeiramente a ferramenta foi demonstrada e posteriormente solicitou-se aos participantes que a testassem e respondessem um questionário com perguntas objetivas (anexo E).

A figura 7.8 apresenta o resultado da pesquisa de opinião realizada com o grupo de ouvintes sobre a interface e utilização do Sign WebForum.

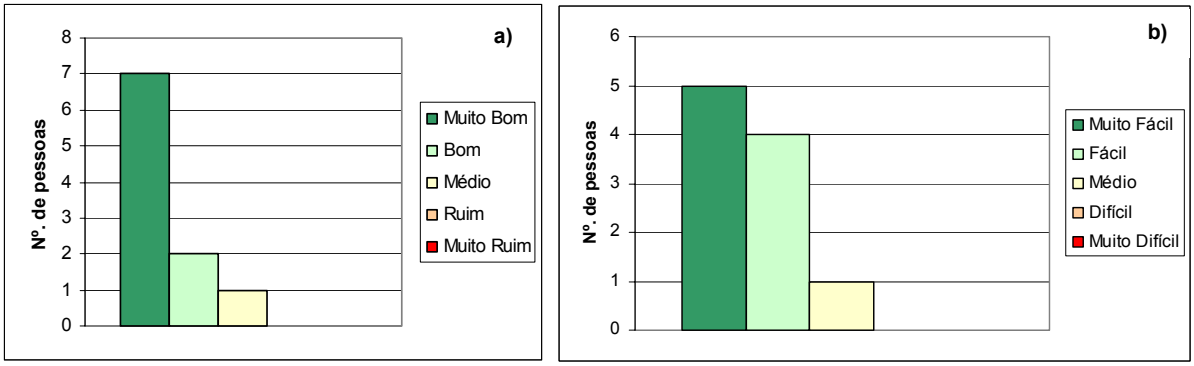


Figura 7.8 – Opinião de ouvintes sobre a interface (a) e utilização (b) do Sign WebForum

A opinião dos ouvintes quanto as possíveis contribuições do software é apresentada na figura 7.9.

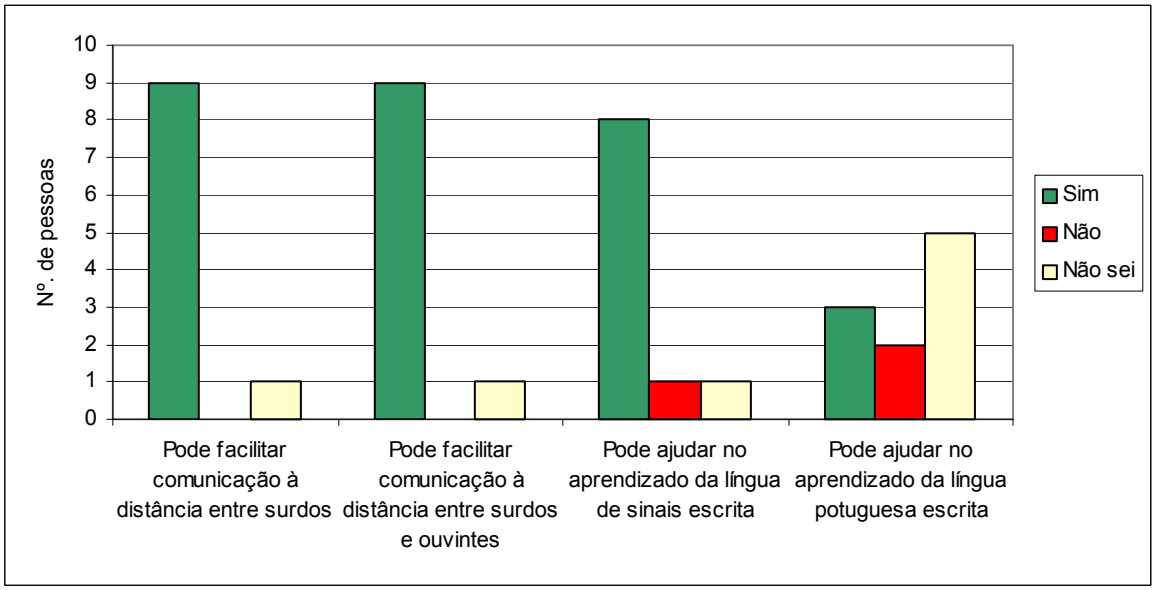


Figura 7.9 – Opinião de ouvintes sobre as contribuições do Sign WebForum

A figura 7.10 apresenta o resultado da pesquisa de opinião realizada com o grupo de surdos sobre a interface e utilização do Sign WebForum.

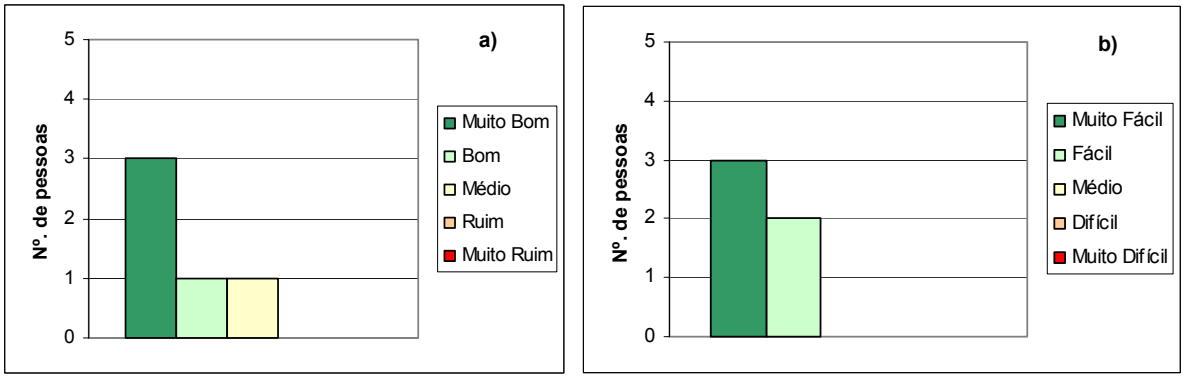


Figura 7.10 – Opinião de surdos sobre a interface (a) e utilização (b) do Sign WebForum

A opinião dos surdos quanto as possíveis contribuições do software é apresentada na figura 7.11.

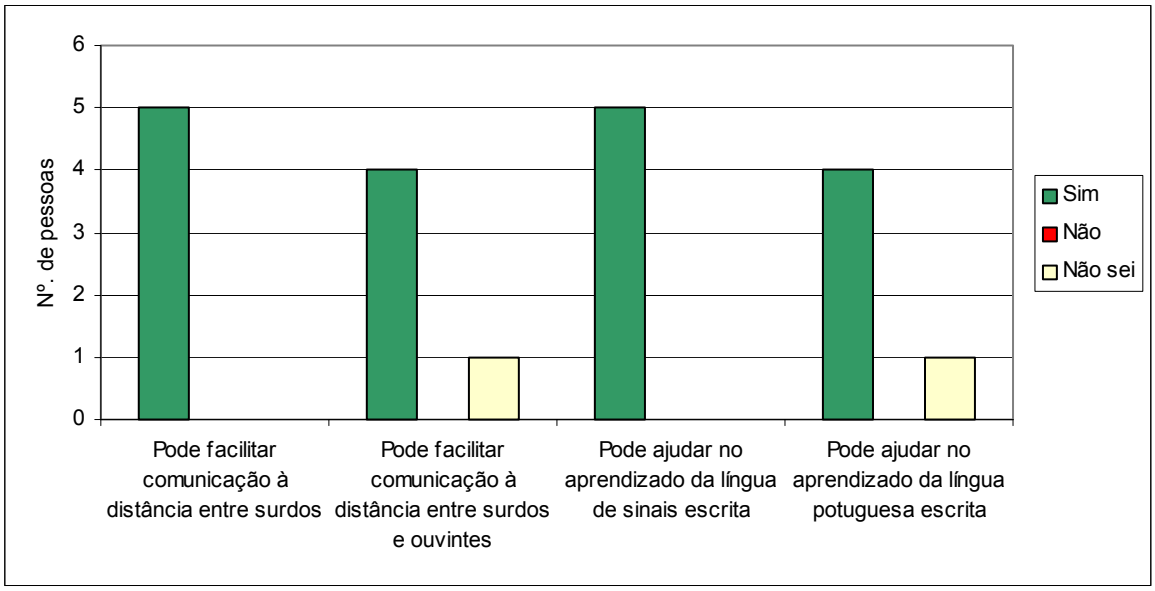


Figura 7.11 – Opinião de surdos sobre as contribuições do Sign WebForum

7.2 Resultados e discussão

Os resultados obtidos nas pesquisas de opinião demonstram que o software desenvolvido atende aos objetivos propostos de forma satisfatória, tanto no que se refere à interface, quanto em relação à utilização e às contribuições que poderá proporcionar ao processo de aprendizagem das línguas envolvidas e à comunicação à distância.

O acentuado número de respostas negativas, do grupo de ouvintes (figura 7.9), quanto à pergunta "Pode ajudar no aprendizado da língua portuguesa escrita" se deve, provavelmente, a um problema de interpretação da questão. A intenção da pergunta é saber se o software pode ajudar os surdos e não os ouvintes no aprendizado da língua portuguesa.

Embora a pesquisa tenha sido realizada sobre o Sign WebForum, ela permite avaliar também as funcionalidades disponibilizadas através dos serviços da biblioteca SWService uma vez que encontram-se integradas com a ferramenta de forma transparente para os usuários.

8 Considerações finais

Este trabalho apresentou a SWService, uma camada de software que fornece os recursos necessários para utilização das línguas de sinais via Web Services. A seguir, serão apresentadas algumas conclusões deste trabalho bem como alguns trabalhos relacionados que poderão ser desenvolvidos futuramente.

8.1 Conclusões

Considerando que a maioria das ferramentas computacionais não está preparada para utilização por pessoas portadoras de necessidades especiais e a dificuldade de disponibilizar as línguas de sinais em software, foi proposto o desenvolvimento deste trabalho.

Para seu desenvolvimento, realizou-se um estudo sobre a cultura surda, a Língua Brasileira de Sinais e o sistema de escrita SignWriting, o qual iniciou em 2002 e foi continuamente aprimorado. Com base neste estudo desenvolveu-se em 2002 a primeira ferramenta que possibilitou a troca de mensagens em Libras, o Sign WebMessage, a qual serviu como base para experimentos e testes das funcionalidades a serem disponibilizadas pela biblioteca SWService. Todos os recursos mapeados para serem disponibilizados na biblioteca, identificados através do estudo de trabalhos relacionados, foram primeiramente implementados no Sign WebMessage e após seus testes e validações foram incorporados na SWService.

Posteriormente, foram estudadas técnicas computacionais necessárias para a implementação e disponibilização dos recursos. Assim, foram realizados estudos sobre a linguagem SWML, *Design Patterns* e Web Services. Após o estudo teórico sobre Web Services, foi realizado um estudo sobre as possibilidades de implementação e uso dessa tecnologia através da linguagem PHP. Esta linguagem foi escolhida para a implementação da SWService por se tratar de uma linguagem *open source*, livre, poderosa e de ampla utilização para o desenvolvimento de soluções baseadas na web, além de ter sido utilizada no desenvolvimento e aprimoramento do Sign WebMessage.

Embasado no estudo sobre as possibilidades de implementação de Web Services com PHP, decidiu-se utilizar a API NuSOAP devido a seus diferenciais como suporte embutido a WSDL, instalação simples, facilidades para realizar o *debug* dos programas, geração automática de informações sobre os serviços desenvolvidos tanto em WSDL com em HTML e projeto documentado e continuamente atualizado.

Com base nos estudos e experimentos realizados, a biblioteca SWService foi modelada através da definição de sua arquitetura base, criação dos seus casos de uso, modelo entidade-relacionamento e de classes. Posteriormente, foram implementados os serviços a serem disponibilizados via Web Services: *getSign* para tradução de português para Libras, *dicPort* para consultas ao dicionário a partir do português, *dicSW* para consultas a partir da Libras e *createSign* para possibilitar a criação de novos sinais.

Como forma de testar a validar os serviços desenvolvidos, foi realizado um estudo de caso no qual três dos quatro serviços disponíveis foram utilizados no desenvolvimento do Sign WebForum, um fórum de discussão que tem como objetivo permitir a escrita e leitura das mensagens em português e Libras. Com o uso da SWService, foi possível concentrar todos os esforços unicamente na natureza do problema que se desejava tratar, ou seja, o

desenvolvimento de um fórum de discussão. Em nenhum momento do desenvolvimento foi necessário abordar questões sobre as funcionalidades necessárias para o uso da Libras. Apenas depois de implementar e testar o funcionamento do fórum de discussão, em português, é que se fez a incorporação de chamadas aos Web Services da biblioteca SWService de maneira rápida e simples.

Com objetivo de validar as funcionalidades nativas do Sign WebForum e as obtidas através do uso dos serviços da biblioteca SWService, foi realizada uma breve pesquisa de opinião com um grupo de surdos e de ouvintes. Os resultados obtidos demonstraram que o software desenvolvido atende aos objetivos propostos de forma bastante satisfatória. Cabe salientar que a pesquisa permitiu avaliar também as funcionalidades disponibilizadas através da SWService, uma vez que, encontram-se integradas ao fórum de discussão de forma transparente para os usuários.

Três grandes avanços foram obtidos através deste trabalho em relação à primeira versão do Sign WebMessage: utilização da linguagem SWML para armazenamento dos sinais do dicionário do ambiente, possibilidade de criação de novos sinais através de interface HTML e redução considerável do número de símbolos utilizados na interface, de 16.112 para 2.024 símbolos.

Desta forma, os objetivos deste trabalho foram plenamente alcançados na medida que, dada a sua conclusão, a comunidade de desenvolvedores tem à disposição os recursos necessários para disponibilizar a escrita em Libras, através do uso da biblioteca SWService. Além disso, destaca-se a total disponibilidade e interoperabilidade que os recursos implementados possuem para serem reutilizados através da tecnologia de Web Services.

A seguir, são apresentados alguns resultados obtidos durante o desenvolvimento deste trabalho:

- artigo completo publicado no XV Simpósio Brasileiro de Informática na Educação – SBIE, ocorrido em novembro de 2004 em Manaus, sobre o aprimoramento do Sign WebMessage como base para o desenvolvimento da SWService [SOU 04];

- artigo completo publicado, também no SBIE 2004 em parceira com a aluna de graduação Daniela Rossi, sobre desenvolvimento do Sign WebForum, software utilizado no estudo de caso desta dissertação [ROS 04b];

- prêmio PAPED 2004 do Programa de Apoio à Pesquisa em Educação à Distância da Secretaria de Educação à Distância do MEC em parceria com a Capes;

- minicurso de introdução a Web Services ministrado no III Seminário de Desenvolvimento de Software Livre – SDSL, realizado em julho de 2004 no Centro Universitário Univates, em Lajeado – RS;

- tutorial sobre Web Services com PHP apresentado no Workshop de Web Services do SBIE 2004;

- elaboração de um livro sobre Web Services com PHP a ser publicado pela editora Novatec de São Paulo em 2005.

8.2 Trabalhos futuros

A seguir, são apresentados alguns trabalhos que poderão ser realizados como continuidade desta dissertação.

- Manter e aprimorar os serviços desenvolvidos de forma que atendam o maior número de demandas possíveis com garantia de flexibilidade, interoperabilidade e qualidade.
- Aumentar o número de entradas no dicionário bilíngüe da SWService através da possibilidade de administração do mesmo por um usuário especialista em Libras e no sistema Sign Writing.
- Possibilitar que o serviço *getSign* retorne a datilologia, em Sign Writing, das palavras em português que não possuem correspondência em Libras.
- Utilizar o serviço *createSign* no Sign WebForum, a fim de possibilitar a criação de novos sinais que não existam do dicionário do ambiente.
- Adaptar o Sign WebMessage para que utilize os serviços disponibilizados pela SWService.
- Criar um ambiente de EAD no qual todas as ferramentas permitam escrita e leitura em Libras através do uso da SWService.
- Divulgar a biblioteca desenvolvida para a comunidade em geral através da publicação de artigos em eventos e revistas da área.

9 Referências Bibliográficas

[ALS 04] ALSHANETSKY, Ilia. **Web Services Programming PHP**. Disponível em: <http://talks.php.net/show/webservices_2004/0>. Acesso em: junho de 2004.

[ARG 04] ARGERICH, Luis. **Introduction to PHP5**. Disponível em: <http://www.phpbuilder.com/columns/argerich20030411.php3>. Acesso em: abril de 2004.

[AYA 04] AYALA, Dietrich e NICHOL, Scott. **NuSOAP - SOAP Toolkit for PHP**. Disponível em: <http://sourceforge.net/projects/nusoap/>. Acesso em: junho de 2004.

[BOO 03] BOOTH, David et. al. **Web Service Architecture**. W3C Working Draft. August 2003. Disponível em http://www.w3.org/TR/2003/WD-ws-arch-20030808

[BRI 95] BRITO, Lucinda Ferreira. **Por uma Gramática de Sinais**. Rio de Janeiro: Tempo Brasileiro. 1995.

[CAM 98] CAMPOS, Márcia de Borba e Silveira, Milene Selbach**. Tecnologias para a Educação Especial**. In: IV Congresso Ibero-Americano de Informática na Educação - RIBIE, Brasília 1998.

[CAM 00] CAMPOS, Márcia Borba et. al. **SIGNSIM: uma ferramenta para auxílio à aprendizagem da língua brasileira de sinais**. In: V Congresso Ibero-Americano de informática na Educação – RIBIE – Chile, 2000.

[CAM 01] CAMPOS, Márcia de Borba. **Ambiente Telemático de interação e comunicação para suporte à educação bilíngüe de surdos**. Tese de doutorado apresentada ao Programa de Pós-Graduação em Informática na Educação da UFRGS. Porto Alegre, 2001.

[CAP 96] CAPOVILLA, Fernando César et. al. **Sistema de Multimídia para comunicação surdo-surdo e surdo-ouvinte em línguas brasileira e americana de sinais via rede de computador**. Revista Informática em saúde. Ano 20, Vol. 20, Nº. 03. p. 110 –114. Abril 1996.

[CAP 01] CAPOVILLA, Fernando C. e Raphael, Walkiria Duarte. **Dicionário Enciclopédico Ilustrado Trilíngüe da Língua de Sinais Brasileira.** São Paulo: Editora da Universidade de São Paulo, 2001.

[CAR 03] CARAVEO, Shane. **Web Services for PHP**. Disponível em: <http://conf.php.net/soap>. Acesso em: junho de 2004.

[CER 02] CERAMI, Ethan. **Web Services Essentials - Distributed Applications with XML-RPC, SOAP, UDDI & WSDL.** O'Reilly, First Edition February 2002.

[CHU 03] CHUNG, Jen-Yao et. al. **Web Services Computing: Advancing Software Interoperability**. Publishe by the IEEE Computer Society. October 2003.

[CON 02] Revista **Contato Surdo.** Ano 2 - Nº 8 – Março de 2002.

[COS 98] COSTA, Antônio Carlos da Rocha. ***SignWriting*: um sistema de escrita para Línguas de Sinais**. EFETA – Publicação Pastoral do Surdo. Ano 5, n 32. Set/Out 1998.

[COS 01a] COSTA, Antônio Rocha e Dimuro, G. P. "**A SignWriting-Based Approach to Sign Language Processing**", In: GW2001 – Gesture Workshop 2001. City Unversity, Londres, 2001.

[COS 01b] COSTA, Antônio Rocha e Dimuro, G. P. "**Supporting Deaf Sign Languages in Written Form on the Web**", In: 10th World Wide Web Conference, Hong Kong, Home Page of Web and Society Track, http://www10.org/cdrom/posters/frame.html.Processing", Written Form on the Web. 2001

[COS 03] COSTA, Antônio C. Rocha. **SignWriting Markup Languege (SWML) site**. [on-line] Consultado em 23 de dezembro de 2003. Disponível em http://swml.ucpel.tche.br.

[CUN 00] CUNHA, Maria Tereza Alvarenga; OLIVEIRA, Marília Peixoto D'; OLIVEIRA, Roseli Duarte; SCHLÜNZEN, Elisa Tomoe Moriya. **O desenvolvimento de projetos e o uso do computador no ambiente de aprendizagem para crianças com necessidades especiais físicas**. in: V Congresso Ibero-Americano de Informática na Educação- RIBIE: Viñadelmar, 2000.

[DOU 03] Lea's, Doug ***Christopher Alexander: An Introduction for Object-Oriented Designers*** [*on-line*] Consultado em 5 de dezembro de 2003. Disponível em http://gee.cs.oswego.edu/dl/ca/ca/ca.html

[FER 99] FERNANDES, Sueli. **É possível ser surdo em Português? Língua de Sinais e escrita: em busca de uma aproximação**. In: SKLIAR (Org.) **A Atualidade da Educação Bilíngüe para Surdos**. Porto Alegre. Editora Mediação, 1999. p. 59-81.

[FER 03a] FERNANDES, Eulália. **Linguagem e Surdez**. Porto Alegre. Editora Artmed, 2003.

[FER 03b] FERRIS, Christopher and FARRELL, Joel. **What Are Web Services?** Communications of the ACM. Vol.46 nº. 6. June 2003.

[FRY 03] FRYE, Jason e Yoder, Joseph. **Patterns Home Page** [*on-line*] Consultado em 5 de dezembro de 2003. Disponível em http://hillside.net/patterns

[FUE 03] FUECKS, Harry. **XML-RPC Progress**. Disponível em: <http://www.phppatterns.com/index.php/article/articleview/83/1/2/>. Acesso em: maio de 2004.

[FUE 04] FUECKS, Harry. **PHP5 SOAP Extension**. Disponível em: <http://www.phppatterns.com/index.php/article/articleview/102/1/11/>. Acesso em: abril de 2004.

[GAM 94] GAMMA, Erich et. Al. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1994.

[GAM 00] GAMMA, Erich et. al. **Padrões de Projeto: soluções reutilizáveis de software** orientado a objetos. Bookman, Porto Alegre, 2000.

[GOE 96] GÓES, Maria Cecília Rafael de. **Linguagem, Surdez e Educação**. Editora Autores Associados. Campinas – São Paulo, 1996.

[HAN 03a] HANSEN, Roseli. **Glue Script: Uma Linguagem Específica de Domínio para Composição de *Web Services***. Dissertação de Mestrado. Programa Interdisciplinar de Pós-Graduação em Computação Aplicada - PIPCA. Universidade do Vale do Rio dos Sinos – Unisinos. São Leopoldo, fevereiro de 2003.

[HAN 03b] HANSEN, Roseli e PINTO, Sérgio Crespo C. S. **Construindo Ambientes de Educação Baseada na Web Através de Web Services Educacionais** In: XIV Simpósio Brasileiro de Informática na Educação – SBIE. Rio de Janeiro, novembro de 2003

[JOK 99] JOKINEN, Markku. **Alguns pontos de vista sobre a educação de surdos nos países nórdicos**. In: SKLIAR, Carlos (Org.). **A Atualidade da Educação Bilíngüe para surdos**. Porto Alegre. Editora Mediação. 1999. p. 105-127.

[KAR 94] KARNOPP, Lodenir Becker. **Aquisição do parâmetro de configuração de mão na língua brasileira de sinais (LIBRAS): estudo sobre quatro crianças surdas, filhas de pais surdos.** Dissertação de mestrado em letras – Instituto de Letras e Artes, Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS. Porto Alegre, 1994.

[KRA 04] KRAMBERGER, Mitja. **Building Web Services Using NuSOAP Toolkit**. Disponível em: http://www.phpbuilder.com/columns/kramberger20031226.php3?page=2&print_mode=1. Acesso em: maio de 2004.

[LAU 01a] LAURENT, Simon St. et. al. **Programming Web Services with XML-RPC**. 1ª Edição. O'Reilly, junho de 2001.

[LAU 01b] LAURENT, Simon St. et. al. **Integrating Web Applications: XML-RPC in PHP**. Disponível em: <http://www.xml.com/pub/a/2001/07/18/excerpt/xml-rpc.html>. Acesso em: abril de 2004.

[MAC 99a] MACEDO, Daniela Remião. ***Sign Dic*: Um ambiente multimídia para a criação e consulta de dicionários bilíngües de línguas de sinais e línguas orais**. Dissertação de mestrado - PUCRS. Porto Alegre, 1999

[MAC 99b] MACEDO, Daniela R. e COSTA, Antônio C. R. ***Sign Dic*: um ambiente para geração de dicionários bilíngües Língua de Sinais – Língua Oral Língua Oral – Língua de Sinais**. In: V Congresso de Educação Bilíngüe para Surdos. UFRGS, Porto Alegre, 1999.

[MAR 00] MARCATO, Simone A. et al. **Um Ambiente para a Aprendizagem da Língua de Sinais**. In: SBC 2000 – XX Congresso da Sociedade Brasileira de Computação, PUCPR - Curitiba, agosto de 2000.

[MAZ 01] MAZUTTI, Caroline et al**. SignHTML Editor HTML para escrita de Língua de Sinais**. Trabalho de conclusão - PUCRS. Porto Alegre, 2001

[MEL 02] MELONFIRE, Icarus. **Using Amazon Web Services with PHP and SOAP**. Disponível em: <http://www.devshed.com/c/a/PHP/Using-Amazon-Web-Services-With-PHP-And-SOAP-part-1/>. Acesso em: junho de 2004.

[NEW 02] NEWCOMER, Eirc. **Understanding Web Services**. Independent Technology Guides. Dadid Chappell, Series Editor. 2002.

[NIC 03] NICHOL, Scott. **Programming with NuSOAP**. Disponível em: <http://www.scottnichol.com/nusoapprog.htm>. Acesso em: maio de 2004.

[NIC 04a] NICHOL, Scott. **Introduction to NuSOAP**. Disponível em: <http://www.scottnichol.com/nusoapintro.htm>. Acesso em: maio de 2004.

[NIC 04b] NICHOL, Scott. **Programming with NuSOAP Using WSDL**. Disponível em: <http://www.scottnichol.com/nusoapprogwsdl.htm>. Acesso em: maio de 2004.

[NOV 98] NOVOA, Julia Iglesias. **Internet en la educación de sordos**. In: Primer Congresso Iberolatinoamericano de Informática Educativa Especial – CIIEE 98. Argentina, outubro de 1998.

[PAI 03] PAIXÃO, Regina Berta M. e BORGES, José Antônio S. **Computadores e alunos com necessidades especiais: o uso de softwares educativos nas Salas de Recursos**. In: XIV Simpósio Brasileiro de Informática na Educação – SBIE 2003. Rio de Janeiro, 2003.

[PEL 98] PELLEGRINO, Claudia Negrão. **Estudos em Imagens Falantes: estimulação do ensino e treinamento de Leitura Labial e Língua de Sinais – LIBRAS VIA CD-ROM em crianças surdas**. In: IV Congresso Ibero-Americano de Informática na Educação - RIBIE, Brasília 1998.

[PER 01] PERLIN, Gladis T. T. **Identidades Surdas** in: SKLIAR, Carlos. **A SURDEZ um Olhar Sobre as Diferenças**. Editora Mediação, 2ª Edição. Porto Alegre, 2001.

[PIN 02] PINTO, Sério Crespo C. S et. al. ***Web Services: An Architectural Overview***. In: First Internationl Seminar on Advanced Research in E-Business – ERB. PUC-RIO. November 2002.

[QUA 97] QUADROS, Ronice Müller. **Educação de Surdos – Aquisição da Linguagem**. 1ª. Edição. Porto Alegre, 1997: Artes Médicas.

[QUA 00] QUADROS, Ronice Müller. **Alfabetização e o ensino da Língua Brasileira de Sinais.** In: Congresso Nacional da ABRALIN. Florianópolis, Santa Catarina, 2000.

[QUA 03] QUADROS, Ronice Muller e KARNOPP, Lodenir Becker. **Língua de Sinais Brasileira – Estudos Lingüísticos.** 1ª. Edição. Editora Artmed, Porto Alegre, 2003.

[RHE 02] RHEINHEIMER, Letícia **JlearningServices: um Framework para Serviços Síncronos em Ambiente para EAD.** Trabalho de conclusão. Universidade do Vale do Rio dos Sinos – Unisinos. Junho de 2002

[RHE 03] RHEINHEIMER, Letícia e PINTO, Sério Crespo C. S. **Usando o Framework JLearningServices para Instanciar Serviços Síncronos para Ambientes de EAD** In:

XIV Simpósio Brasileiro de Informática na Educação – SBIE. Rio de Janeiro, novembro de 2003

[RIE 03] RIEDY, Jason. **The "History of Patterns"** on Ward Cunningham's WikiWiki Web [*on-line*] Consultado em 9 de dezembro de 2003. Disponível em http://c2.com/cgi-bin/wiki?HistoryOfPatterns

[ROC 00] ROCHA, Heloísa Vieira et al. **Um Ambiente para a Aprendizagem da Língua de Sinais**. In: SBC 2000 – XX Congresso da Sociedade Brasileira de Computação, PUCPR - Curitiba, agosto de 2000.

[ROS 04a] ROSSI, Daniela. **Sign WebForum: um Fórum de Discussão que Possibilita Troca de Mensagens em Libras**. Trabalho de conclusão. Graduação em Análise de Sistemas. Universidade do Vale do Rio dos Sinos - Unisinos. São Leopoldo, dezembro de 2004.

[ROS 04b] ROSSI, Daniela, SOUZA, Vinícius Costa e PINTO, Sérgio Crespo C. S. **Sign WebForum: um Fórum de Discussão que Possibilita Troca de Mensagens em Libras**. In: XV Simpósio Brasileiro de Informática na Educação – SBIE 2004. Manaus, novembro de 2004.

[RUB 04] RUBIO, Daniel. **Building PHP Web services with PEAR**. Disponível em: <http://webservices.devchannel.org/webserviceschannel/04/02/11/1432220.shtml?tid=47&tid=51&tid=54>. Acesso em: junho de 2004.

[SAN 97] SANTAROSA, Lucila M. C. **Telemática: um novo canal de comunicação para deficientes auditivos**. In: World Conference on Educational Multimedia and Hypermed and World Conference on Educational Telecommunications. Calgary, Canadá, 1997.

[SAN 00] SANTAROSA, Lucila Maria. **Telemática y la inclusión virtual y social de personas con necesidades especiales: un espacio posible en la Internet**. in: V Congresso Ibero-Americano de Informática na Educação- RIBIE: Viñadelmar, 2000.

[SAN 01] SANTAROSA, Lucila. M. C. **INTERNET: espaço possível para o desenvolvimento e inclusão digital pessoas com necessidades educativas especiais**. In: OLIVEIRA, M. L W. (org). Tempos Modernos: os desafios da atualidade. Rio de janeiro. 2001 – p. 70-77

[SKL 99] SKLIAR, Carlos. **A Localização Política da Educação Bilíngüe para Surdos**. In: SKLIAR, Carlos (Org.). **A Atualidade da Educação Bilíngüe para surdos**. Porto Alegre. Editora Mediação. 1999. p. 07-14.

[SKL 01a] SKLIAR, Carlos. **A SURDEZ, um olhar sobre as diferenças**. Editora Mediação, 2ª Edição. Porto Alegre, 2001.

[SKL 01b] SKLIAR, Carlos. **Educação e Exclusão – Abordagens Sócio-antropológicas em Educação Especial**. Editora Mediação, 3ª Edição. Porto Alegre, 2001.

[SOU 02a] SOUZA, Vinícius Costa. ***Sign WebMessage*: um ambiente para comunicação via *Web* baseado na escrita de LIBRAS**. Trabalho de conclusão. Graduação em Análise de Sistemas. Universidade do Vale do Rio dos Sinos - Unisinos. São Leopoldo, dezembro de 2002.

[SOU 02b] SOUZA, Vinícius Costa e PINTO, Sérgio Crespo C. da Silva. ***Sign WebMessage*: um ambiente para comunicação via *Web* baseado na escrita de LIBRAS**. In: III Congresso Ibero-americano de Informática na Educação Especial – CIIEE2002. Fortaleza, agosto de 2002.

[SOU 03a] SOUZA, Vinícius Costa e PINTO, Sérgio Crespo C. S. ***Sign WebMessage:* uma ferramenta para comunicação via web através da Língua Brasileira de Sinais – Libras**. In: XIV Simpósio Brasileiro de Informática na Educação – SBIE 2003. Rio de Janeiro, novembro de 2003.

[SOU 03b] SOUZA, Vinícius Costa, AGUIAR, Márcia Rafaeli e PINTO, Sérgio Crespo C. S. **Desafios e Resultados de uma experiência na Inclusão Digital de Surdos**. In: XIV Simpósio Brasileiro de Informática na Educação – SBIE 2003. Rio de Janeiro, novembro de 2003.

[SOU 04] SOUZA, Vinícius Costa e PINTO, Sérgio Crespo C. S. **O Aprimoramento do Sign WebMessage como Base para o Desenvolvimento da SWService: uma Biblioteca para a Escrita da Libras na Internet Baseada em Web Services**. In: XV Simpósio Brasileiro de Informática na Educação – SBIE 2004. Manaus, novembro de 2004.

[STU 00] STUMPF, Marianne Rossi. **Língua de Sinais: escrita dos surdos na Internet**. In: V Congresso Ibero-Americano de Informática na Educação – RIBIE – Chile, 2000.

[SUT 01] SUTTON, Valerie. **SignWriter Computer Program - Instruction Manual (version 4.4).** La Jolla, Deaf Action Committee for SignWriting, 2001.

[SUT 02] SUTTON, Valerie. **SignBank 2002 – The SignWriting Online Dictionary** Center for Sutton Movement Writing, Inc. A US nonprofit educational organization. La Jolla, 2002.

[SUT 03a] SUTTON, Valerie. **Lessons in SignWriting – Textbook and Workbook**, Third Edition, La Jolla, Deaf Action Committee for SignWriting, 2003.

[SUT 03b] SUTTON, Valerie. **SignWriter Java – Sign Language Processor.** La Jolla, Deaf Action Committee for SignWriting, 2003.

[SUT 04] SUTTON, Valerie**. Sutton's Sign-Symbol-Sequence.** La Jolla, Deaf Action Committee for SignWriting, 2004.

[TOR 02] TORCHELSEN, Rafael P. e COSTA, Antônio Carlos R. **Editor para Língua de Sinais Escritas em SignWriting**. In: XIII Simpósio Brasileiro de Informática na Educação – SBIE 2002. São Leopoldo, novembro de 2002.

[TRA 03a] TRACHTENBERG, Adam. A PHP Web Services Client. Disponível em: <http://www.onlamp.com/pub/a/php/2003/07/03/php_amazon_soap.html >. Acesso em: maio de 2004.

[TRA 03b] TRACHTENBERG, Adam. **PHP Web Services Without SOAP**. Disponível em: <http://www.onlamp.com/pub/a/php/2003/10/30/amazon_rest.html>. Acesso em: junho de 2004.

[VAL 91] VALENTE, José Armando. **Usos do Computador na Educação**. In: **Liberando a Mente: Computadores na Educação Especial.** Graf. Central da UNICAMP: Campinas, 1991. P. 16-31.

[VYG 98] VYGOTSKY, Lev S. **Pensamento e Linguagem**. 2ª edição. São Paulo, 1998.

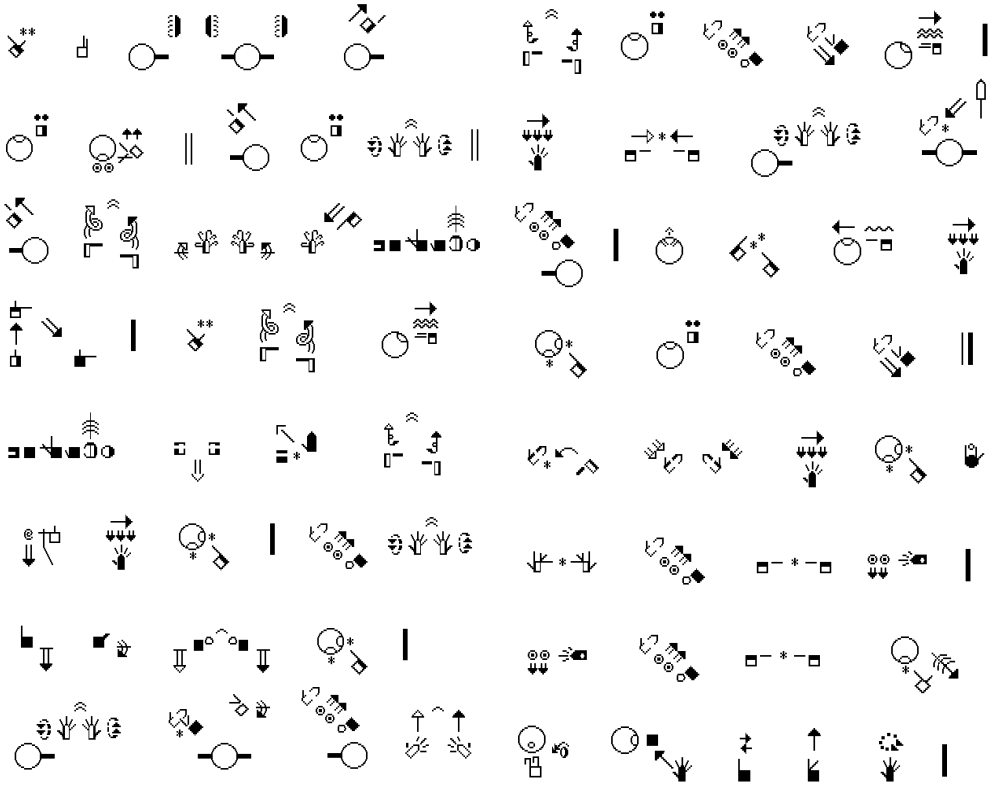
[WAL 02] WALSH, Aaron E. **UDDI, SOAP and WSDL The Web Services Specification Reference Book**. Prentice Hall Books. United States of America 2002.

[ZEN 04] ZEND Technologies, Inc. **PHP SOAP Extension**. Disponível em: <http://www.devshed.com/c/a/Zend/PHP-SOAP-Extension/>. Acesso em: junho de 2004.

Anexo A - Parágrafo em Português e Libras

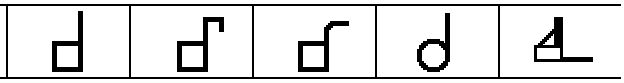
Parágrafo escrito por Ronice Quadros [QUA 97], com a correspondente tradução para língua de sinas, através da escrita em *SignWriting*.

"Há dois grupos, aqueles que aprendem a falar e aqueles que aprendem a língua de sinais. Esses últimos desenvolvem a habilidade espacial no cérebro de forma mais sofisticada do que o outro. A possibilidade de se ter um desenvolvimento mais natural do espaço pode favorecer o processo educacional da criança surda. A escrita da língua de sinais é uma forma de aproveitar o potencial dos surdos. A representação da língua de sinais através da escrita permite um processo de aprendizagem da leitura e escrita natural. As crianças estabelecem relações diretas das línguas de sinais para a escrita. Por que é tão complicada a alfabetização das crianças surdas? Até o presente, as crianças surdas só tiveram contato com a escrita do português. Esta forma escrita está relacionada com a língua oral auditiva e não com uma língua visual espacial."

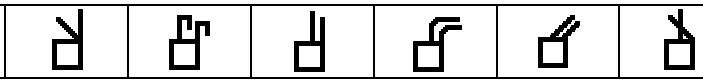


Anexo B – Símbolos de mão da SWService

Símbolos base do grupo de mão 1

Símbolos utilizados (5)	
Símbolos existentes: 480	**Redução:** 475

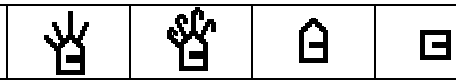
Símbolos base do grupo de mão 2

Símbolos utilizados (6)	
Símbolos existentes: 576	**Redução:** 570

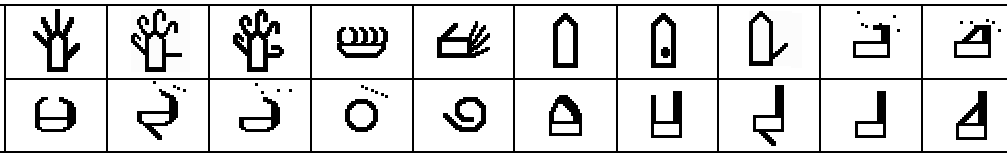
Símbolos base do grupo de mão 3

Símbolos utilizados (8)	
Símbolos existentes: 768	**Redução:** 760

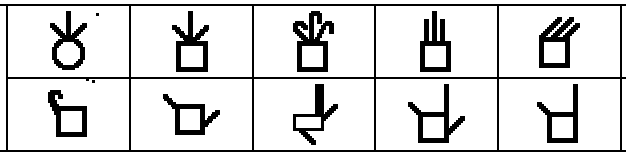
Símbolos base do grupo de mão 4

Símbolos utilizados (4)	
Símbolos existentes: 384	**Redução:** 380

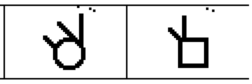
Símbolos base do grupo de mão 5

Símbolos utilizados (20)	
Símbolos existentes: 1920	**Redução:** 1900

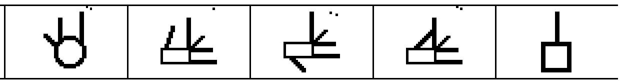
Símbolos base do grupo de mão 6

Símbolos utilizados (11)	
Símbolos existentes: 1056	**Redução:** 1045

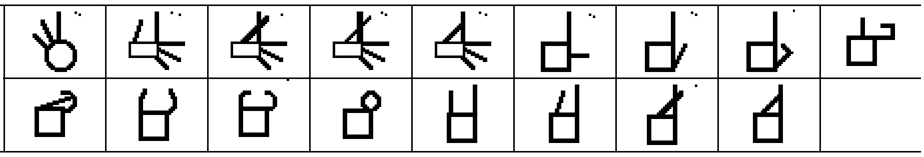
Símbolos base do grupo de mão 7

Símbolos utilizados (2)	
Símbolos existentes: 192	**Redução:** 190

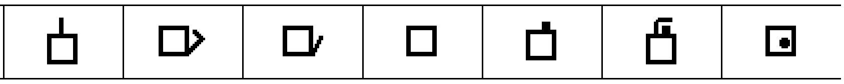
Símbolos base do grupo de mão 8

Símbolos Utilizados (5)	
Símbolos existentes: 480	**Redução:** 475

Símbolos base do grupo de mão 9

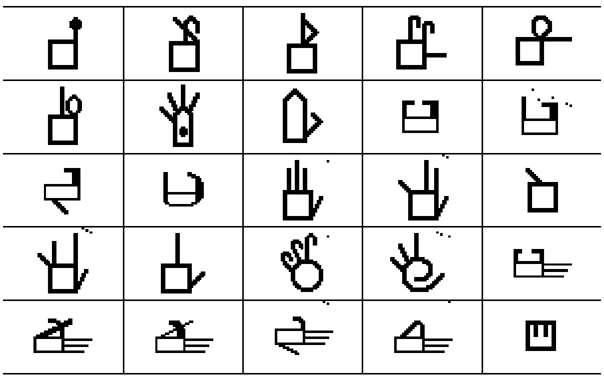
Símbolos utilizados (17)	
Símbolos existentes: 1632	**Redução:** 1615

Símbolos base do grupo de mão 10

Símbolos utilizados (7)	
Símbolos existentes: 672	**Redução:** 665

Conforme orientação da pesquisadora e professora de escrita de sinais em SignWriting, a surda Marianne Stumpf, existem 25 símbolos base do SSS, bem como suas variações, que não são utilizados na Libras, totalizando 2400 símbolos. A seguir são apresentados os 25 símbolos base não utilizados na Libras.

Símbolos base não utilizados na Libras



Anexo C – Símbolos de movimento da SWService

Símbolos base do grupo de movimento Parede

Símbolos utilizados (16) e Número de variações								
	47	47	47	47	47	47	23	47
								
	23	23	47	47	47	47	47	47
Símbolos existentes: 696							**Redução:** 680	

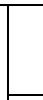
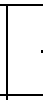
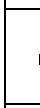
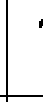
Símbolos base do grupo de movimento Diagonal

Símbolos utilizados (12) e Número de variações						
	17	17	17	17	17	17
						
	17	17	17	17	17	17
Símbolos existentes: 216					**Redução:** 204	

Símbolos base do grupo de movimento Chão

Símbolos utilizados (25) e Número de variações									
	47	47	47	23	23	23	23	23	47
									
	47	47	47	47	47	47	47	47	47
	47	47	47	47	47	47	47		
Símbolos existentes: 1080							**Redução:** 1055		

Símbolos base do grupo de movimento Curvo paralelo à parede

Símbolos base (24) e Número de variações								
	47	47	47	47	47	47	11	11
								
	47	47	47	47	47	47	11	47
								
	15	11	11	11	1	1	1	1
Número de símbolos: 716							**Redução:** 692	

Símbolos base do grupo de movimento Curvo bate parede

Símbolos Base (8) e Número de variações				
	11	11	11	11
				
	11	11	11	11
Número de símbolos: 96			**Redução:** 88	

Símbolos base do grupo de movimento Curvo bate chão

Símbolos Base (14) e Número de variações							
	35	35	17	11	11	11	11
							
	47	23	23	11	11	11	11
Número de símbolos: 282						**Redução:** 268	

Símbolos base do grupo de movimento Curvo paralelo ao chão

Símbolo base (16) e Número de variações								
	11	11	11	11	11	11	11	47
	15	11	11	11	11	11	11	11
Número de símbolos: 232							**Redução:** 216	

Símbolos base do grupo de movimento Circular

Símbolo base (16) e Número de variações								
	47	47	47	47	47	47	47	47
								
	17	17	1	1	1	1	1	1

Número de símbolos: 426 **Redução:** 410

Anexo D – SWService WSDL

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
  ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:si="http://soapinterop.org/xsd" xmlns:tns="urn:SWService-WSDL"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:SWService-WSDL">
- <types>
- <xsd:schema targetNamespace="urn:SWService-WSDL">
  <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
  <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
  </xsd:schema>
  </types>
- <message name="getSignRequest">
  <part name="texto" type="xsd:string" />
  <part name="formato" type="xsd:string" />
  <part name="cols" type="xsd:string" />
  </message>
- <message name="getSignResponse">
  <part name="libras" type="xsd:string" />
  </message>
- <message name="dicPortRequest">
  <part name="script" type="xsd:string" />
  <part name="background" type="xsd:string" />
  <part name="css" type="xsd:string" />
  <part name="action_form" type="xsd:string" />
  <part name="params" type="xsd:string" />
  <part name="image_head_software" type="xsd:string" />
  <part name="id_word" type="xsd:string" />
  <part name="action" type="xsd:string" />
  <part name="word" type="xsd:string" />
  <part name="id_sign" type="xsd:string" />
  </message>
- <message name="dicPortResponse">
  <part name="html" type="xsd:string" />
  </message>
- <message name="dicSWRequest">
  <part name="script" type="xsd:string" />
  <part name="css" type="xsd:string" />
  <part name="image_head_software" type="xsd:string" />
  <part name="action_form" type="xsd:string" />
  <part name="params" type="xsd:string" />
  <part name="action" type="xsd:string" />
  <part name="id_simbolo" type="xsd:string" />
  <part name="simbolos" type="xsd:string" />
  <part name="btn" type="xsd:string" />
  </message>
- <message name="dicSWResponse">
```

```
  <part name="html" type="xsd:string" />
    </message>
- <message name="createSignRequest">
  <part name="script" type="xsd:string" />
  <part name="css" type="xsd:string" />
  <part name="image_head_software" type="xsd:string" />
  <part name="action_form" type="xsd:string" />
  <part name="params" type="xsd:string" />
  <part name="action" type="xsd:string" />
  <part name="id_simbolo" type="xsd:string" />
  <part name="simbolos" type="xsd:string" />
  <part name="btn" type="xsd:string" />
  <part name="palavras" type="xsd:string" />
    </message>
- <message name="createSignResponse">
  <part name="html" type="xsd:string" />
  <part name="swml" type="xsd:string" />
  <part name="word" type="xsd:string" />
    </message>
- <portType name="SWService-WSDLPortType">
- <operation name="getSign">
  <documentation>Este serviço traduz um texto do Português para Libras, em SigWriting. O
    formato da resposta, em Libras, pode ser HTML ou SWML dependendo do parâmetro
    passado para o serviço.</documentation>
  <input message="tns:getSignRequest" />
  <output message="tns:getSignResponse" />
    </operation>
- <operation name="dicPort">
  <documentation>Este serviço monta o HTML para consulta ao dicionário a partir do
    português.</documentation>
  <input message="tns:dicPortRequest" />
  <output message="tns:dicPortResponse" />
    </operation>
- <operation name="dicSW">
  <documentation>Este serviço monta o HTML para consulta ao dicionário a partir da
    Libras.</documentation>
  <input message="tns:dicSWRequest" />
  <output message="tns:dicSWResponse" />
    </operation>
- <operation name="createSign">
  <documentation>Este serviço monta o HTML para criação e edição de sinais em
    SignWriting.</documentation>
  <input message="tns:createSignRequest" />
  <output message="tns:createSignResponse" />
    </operation>
    </portType>
- <binding name="SWService-WSDLBinding" type="tns:SWService-WSDLPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <operation name="getSign">
  <soap:operation soapAction="urn:SWService-WSDL#getSign" style="rpc" />
- <input>
  <soap:body use="encoded" namespace="urn:SWService-WSDL"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
- <output>
  <soap:body use="encoded" namespace="urn:SWService-WSDL"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
```

```
    </output>
    </operation>
- <operation name="dicPort">
  <soap:operation soapAction="urn:SWService-WSDL#dicPort" style="rpc" />
- <input>
  <soap:body use="encoded" namespace="urn:SWService-WSDL"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
- <output>
  <soap:body use="encoded" namespace="urn:SWService-WSDL"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
    </operation>
- <operation name="dicSW">
  <soap:operation soapAction="urn:SWService-WSDL#dicSW" style="rpc" />
- <input>
  <soap:body use="encoded" namespace="urn:SWService-WSDL"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
- <output>
  <soap:body use="encoded" namespace="urn:SWService-WSDL"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
    </operation>
- <operation name="createSign">
  <soap:operation soapAction="urn:SWService-WSDL#dicSW" style="rpc" />
- <input>
  <soap:body use="encoded" namespace="urn:SWService-WSDL"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
- <output>
  <soap:body use="encoded" namespace="urn:SWService-WSDL"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
    </operation>
    </binding>
- <service name="SWService-WSDL">
- <port name="SWService-WSDLPort" binding="tns:SWService-WSDLBinding">
  <soap:address location="http://www.inf.unisinos.br/~swm/swservice.php" />
    </port>
    </service>
    </definitions>
```

Anexo E – Questionário

QUESTIONÁRIO DE AVALIAÇÂO DO SIGN WEBFORUM

Dados pessoais

Nome: _____ **Data**: _____

Idade: _____ **Escolaridade**: _____ **Surdo**: _____

Como é o seu conhecimento/uso da Língua Brasileira de Sinais (Libras)?

( ) muito bom ( ) bom ( ) médio ( ) ruim ( ) muito ruim

Como é a sua capacidade de leitura e escrita em Português?

( ) muito boa ( ) boa ( ) média ( ) ruim ( ) muito ruim

Sobre o Sign WebForum

1) Quanto à Interface do sistema?

( ) muito boa ( ) boa ( ) média ( ) ruim ( ) muito ruim

2) Quanto à leitura das mensagens em Português ?

( ) muito fácil ( ) fácil ( ) médio ( ) difícil ( ) muito difícil

3) Quanto à escrita de uma mensagem em Português ?

( ) muito fácil ( ) fácil ( ) médio ( ) difícil ( ) muito difícil

4) Quanto à leitura das mensagens em Libras ?

( ) muito fácil ( ) fácil ( ) médio ( ) difícil ( ) muito difícil

5) Quanto à escrita de uma mensagem em Libras ?

( ) muito fácil ( ) fácil ( ) médio ( ) difícil ( ) muito difícil

6) Quanto à consulta ao dicionário em Português?

( ) muito fácil ( ) fácil ( ) médio ( ) difícil ( ) muito difícil

7) Quanto à consulta ao dicionário em Libras?

( ) muito fácil ( ) fácil ( ) médio ( ) difícil ( ) muito difícil

8) Quanto à utilização geral:

( ) muito fácil ( ) fácil ( ) médio ( ) difícil ( ) muito difícil

9) Você acha que o *Sign WebForum* pode facilitar a comunicação a distância entre surdos?

( ) sim ( ) não ( ) não sei

10) Você acha que o *Sign WebForum* pode facilitar a comunicação a distância entre surdos e ouvintes?

( ) sim ( ) não ( ) não sei

11) Você acha que o *Sign WebForum* pode ajudar no aprendizado da língua de sinais escrita?

( ) sim ( ) não ( ) não sei

12) Você acha que o *Sign WebForum* pode ajudar no aprendizado da língua portuguesa escrita?

( ) sim ( ) não ( ) não sei