TURNO:	**MATUTINO**	VERSÃO:	1	ANO / SEMESTRE:	2013.2	Nº	

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO — BACHARELADO
COORDENAÇÃO DE TRABALHO DE CONCLUSÃO DE CURSO

PROPOSTA PARA O TRABALHO DE CONCLUSÃO DE CURSO

TÍTULO: GERADOR DE CÓDIGO A PARTIR DE ESPECIFICAÇÕES COM PADRÕES DE REQUISITOS

ÁREA: Engenharia de Software

Palavras-chave: Processamento de linguagem natural. Análise de requisitos. Geração automática de código.

1 IDENTIFICAÇÃO

1.1 ALUNO

Nome: **Leandro Vilson Battisti**			Código/matrícula: 102993	
Endereço residencial:				
Rua: Arthur Weise			n°: 121	Complemento:
Bairro: Agua Verde	CEP: 89032290	Cidade: Blumenau		UF: SC
Telefone fixo: (47) 3330-5180		Celular: (47) 9175-1805		
Endereço comercial:				
Empresa: Senior Sistemas S/A				
Rua:			n°:	Bairro:
CEP:	Cidade:		UF: SC	Telefone: 3322-3300
E-Mail FURB: lbattisti@furb.br		E-Mail alternativo: asengardeons@hotmail.com		

1.2 ORIENTADOR

Nome: Joyce Martins	
E-Mail FURB: joyce@furb.br	E-Mail alternativo:

2 DECLARAÇÕES

2.1 DECLARAÇÃO DO ALUNO

Declaro que estou ciente do Regulamento do Trabalho de Conclusão de Curso de Ciência da Computação e que a proposta em anexo, a qual concordo, foi por mim rubricada em todas as páginas. Ainda me comprometo pela obtenção de quaisquer recursos necessários para o desenvolvimento do trabalho, caso esses recursos não sejam disponibilizados pela Universidade Regional de Blumenau (FURB).

Assinatura: _____ Local/data: _____

2.2 DECLARAÇÃO DO ORIENTADOR

Declaro que estou ciente do Regulamento do Trabalho de Conclusão do Curso de Ciência da Computação e que a proposta em anexo, a qual concordo, foi por mim rubricada em todas as páginas. Ainda me comprometo a orientar o aluno da melhor forma possível de acordo com o plano de trabalho explícito nessa proposta.

Assinatura: _____ Local/data: _____

3 AVALIAÇÃO DA PROPOSTA

3.1 AVALIAÇÃO DO(A) **ORIENTADOR(A)**

Acadêmico(a): Leandro Vilson Battisti

Orientador(a): Joyce Martins

	ASPECTOS AVALIADOS	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO 1.1. O tema de pesquisa está devidamente contextualizado/delimitado?			
	1.2. O problema está claramente formulado?			
	2. OBJETIVOS 2.1. O objetivo geral está claramente definido e é passível de ser alcançado?			
	2.2. São apresentados objetivos específicos (opcionais) coerentes com o objetivo geral? Caso não sejam apresentados objetivos específicos, deixe esse item em branco.			
	3. RELEVÂNCIA 3.1. A proposta apresenta um grau de relevância em computação que justifique o desenvolvimento do TCC?			
	4. METODOLOGIA 4.1. Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	4.2. Os métodos e recursos estão devidamente descritos e são compatíveis com a metodologia proposta?			
	4.3. A proposta apresenta um cronograma físico (período de realização das etapas) de maneira a permitir a execução do TCC no prazo disponível?			
	5. REVISÃO BIBLIOGRÁFICA 5.1. As informações apresentadas são suficientes e têm relação com o tema do TCC?			
	5.2. São apresentados trabalhos correlatos, bem como comentadas as principais características dos mesmos?			
	6. REQUISITOS DO SISTEMA A SER DESENVOLVIDO 6.1. Os requisitos funcionais e não funcionais do sistema a ser desenvolvido foram claramente descritos?			
	7. CONSIDERAÇÕES FINAIS 7.1. As considerações finais relacionam os assuntos apresentados na revisão bibliográfica com a realização do TCC?			
ASPECTOS METODOLÓGICOS	8. REFERÊNCIAS BIBLIOGRÁFICAS 8.1. As referências bibliográficas obedecem às normas da ABNT?			
	8.2. As referências bibliográficas contemplam adequadamente os assuntos abordados na proposta (são usadas obras atualizadas e/ou as mais importantes da área)?			
	9. CITAÇÕES 9.1. As citações obedecem às normas da ABNT?			
	9.2. As informações retiradas de outros autores estão devidamente citadas?			
	10. AVALIAÇÃO GERAL (organização e apresentação gráfica, linguagem usada) 10.1. O texto obedece ao formato estabelecido?			
	10.2. A exposição do assunto é ordenada (as idéias estão bem encadeadas e a linguagem utilizada é clara)?			

A proposta de TCC deverá ser revisada, isto é, necessita de complementação, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou
- pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADA ( ) NECESSITA DE COMPLEMENTAÇÃO

Assinatura do(a) avaliador(a): _____ Local/data: _____

CONSIDERAÇÕES DO(A) ORIENTADOR(A):

Caso o(a) orientador(a) tenha assinalado em sua avaliação algum item como “atende parcialmente”, devem ser relatos os problemas/melhorias a serem efetuadas.

Na segunda versão, caso as alterações sugeridas pelos avaliadores não sejam efetuadas, deve-se incluir uma justificativa.

Assinatura do(a) avaliador(a): _____ Local/data: _____

3.2 AVALIAÇÃO DO(A) **COORDENADOR DE TCC**

Acadêmico(a): Leandro Vilson Battisti

Avaliador(a): José Roque Voltolini da Silva

	ASPECTOS AVALIADOS	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO 1.1. O tema de pesquisa está devidamente contextualizado/delimitado?			
	1.2. O problema está claramente formulado?			
	2. OBJETIVOS 2.1. O objetivo geral está claramente definido e é passível de ser alcançado?			
	2.2. São apresentados objetivos específicos (opcionais) coerentes com o objetivo geral? Caso não sejam apresentados objetivos específicos, deixe esse item em branco.			
	3. RELEVÂNCIA 3.1. A proposta apresenta um grau de relevância em computação que justifique o desenvolvimento do TCC?			
	4. METODOLOGIA 4.1. Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	4.2. Os métodos e recursos estão devidamente descritos e são compatíveis com a metodologia proposta?			
	4.3. A proposta apresenta um cronograma físico (período de realização das etapas) de maneira a permitir a execução do TCC no prazo disponível?			
	5. REVISÃO BIBLIOGRÁFICA 5.1. As informações apresentadas são suficientes e têm relação com o tema do TCC?			
	5.2. São apresentados trabalhos correlatos, bem como comentadas as principais características dos mesmos?			
	6. REQUISITOS DO SISTEMA A SER DESENVOLVIDO 6.1. Os requisitos funcionais e não funcionais do sistema a ser desenvolvido foram claramente descritos?			
	7. CONSIDERAÇÕES FINAIS 7.1. As considerações finais relacionam os assuntos apresentados na revisão bibliográfica com a realização do TCC?			
ASPECTOS METODOLÓGICOS	8. REFERÊNCIAS BIBLIOGRÁFICAS 8.1. As referências bibliográficas obedecem às normas da ABNT?			
	8.2. As referências bibliográficas contemplam adequadamente os assuntos abordados na proposta (são usadas obras atualizadas e/ou as mais importantes da área)?			
	9. CITAÇÕES 9.1. As citações obedecem às normas da ABNT?			
	9.2. As informações retiradas de outros autores estão devidamente citadas?			
	10. AVALIAÇÃO GERAL (organização e apresentação gráfica, linguagem usada) 10.1. O texto obedece ao formato estabelecido?			
	10.2. A exposição do assunto é ordenada (as idéias estão bem encadeadas e a linguagem utilizada é clara)?			

A proposta de TCC deverá ser revisada, isto é, necessita de complementação, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou
- pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADA ( ) NECESSITA DE COMPLEMENTAÇÃO

OBSERVAÇÕES:

Assinatura do(a) avaliador(a): _____ Local/data: _____

3.3 AVALIAÇÃO DO(A) **PROFESSOR(A) DA DISCIPLINA DE TCCI**

Acadêmico(a): Leandro Vilson Battisti

Avaliador(a): Roberto Heinzle

	ASPECTOS AVALIADOS	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO 1.1. O tema de pesquisa está devidamente contextualizado/delimitado?			
	1.2. O problema está claramente formulado?			
	2. OBJETIVOS 2.1. O objetivo geral está claramente definido e é passível de ser alcançado?			
	2.2. São apresentados objetivos específicos (opcionais) coerentes com o objetivo geral? Caso não sejam apresentados objetivos específicos, deixe esse item em branco.			
	3. RELEVÂNCIA 3.1. A proposta apresenta um grau de relevância em computação que justifique o desenvolvimento do TCC?			
	4. METODOLOGIA 4.1. Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	4.2. Os métodos e recursos estão devidamente descritos e são compatíveis com a metodologia proposta?			
	4.3. A proposta apresenta um cronograma físico (período de realização das etapas) de maneira a permitir a execução do TCC no prazo disponível?			
	5. REVISÃO BIBLIOGRÁFICA 5.1. As informações apresentadas são suficientes e têm relação com o tema do TCC?			
	5.2. São apresentados trabalhos correlatos, bem como comentadas as principais características dos mesmos?			
	6. REQUISITOS DO SISTEMA A SER DESENVOLVIDO 6.1. Os requisitos funcionais e não funcionais do sistema a ser desenvolvido foram claramente descritos?			
	7. CONSIDERAÇÕES FINAIS 7.1. As considerações finais relacionam os assuntos apresentados na revisão bibliográfica com a realização do TCC?			
ASPECTOS METODOLÓGICOS	8. REFERÊNCIAS BIBLIOGRÁFICAS 8.1. As referências bibliográficas obedecem às normas da ABNT?			
	8.2. As referências bibliográficas contemplam adequadamente os assuntos abordados na proposta (são usadas obras atualizadas e/ou as mais importantes da área)?			
	9. CITAÇÕES 9.1. As citações obedecem às normas da ABNT?			
	9.2. As informações retiradas de outros autores estão devidamente citadas?			
	10. AVALIAÇÃO GERAL (organização e apresentação gráfica, linguagem usada) 10.1. O texto obedece ao formato estabelecido?			
	10.2. A exposição do assunto é ordenada (as idéias estão bem encadeadas e a linguagem utilizada é clara)?			
	PONTUALIDADE NA ENTREGA		atraso de _____ dias	

A proposta de TCC deverá ser revisada, isto é, necessita de complementação, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou
- pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADA ( ) NECESSITA DE COMPLEMENTAÇÃO

OBSERVAÇÕES:

Assinatura do(a) avaliador(a): _____ Local/data: _____

3.4 AVALIAÇÃO DO(A) **PROFESSOR(A) ESPECIALISTA NA ÁREA**

Acadêmico(a): Leandro Vilson Battisti

Avaliador(a): Everaldo Arthur Grahl

	ASPECTOS AVALIADOS	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO 1.1. O tema de pesquisa está devidamente contextualizado/delimitado?			
	1.2. O problema está claramente formulado?			
	2. OBJETIVOS 2.1. O objetivo geral está claramente definido e é passível de ser alcançado?			
	2.2. São apresentados objetivos específicos (opcionais) coerentes com o objetivo geral? Caso não sejam apresentados objetivos específicos, deixe esse item em branco.			
	3. RELEVÂNCIA 3.1. A proposta apresenta um grau de relevância em computação que justifique o desenvolvimento do TCC?			
	4. METODOLOGIA 4.1. Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	4.2. Os métodos e recursos estão devidamente descritos e são compatíveis com a metodologia proposta?			
	4.3. A proposta apresenta um cronograma físico (período de realização das etapas) de maneira a permitir a execução do TCC no prazo disponível?			
	5. REVISÃO BIBLIOGRÁFICA 5.1. As informações apresentadas são suficientes e têm relação com o tema do TCC?			
	5.2. São apresentados trabalhos correlatos, bem como comentadas as principais características dos mesmos?			
	6. REQUISITOS DO SISTEMA A SER DESENVOLVIDO 6.1. Os requisitos funcionais e não funcionais do sistema a ser desenvolvido foram claramente descritos?			
	7. CONSIDERAÇÕES FINAIS 7.1. As considerações finais relacionam os assuntos apresentados na revisão bibliográfica com a realização do TCC?			
ASPECTOS METODOLÓGICOS	8. REFERÊNCIAS BIBLIOGRÁFICAS 8.1. As referências bibliográficas obedecem às normas da ABNT?			
	8.2. As referências bibliográficas contemplam adequadamente os assuntos abordados na proposta (são usadas obras atualizadas e/ou as mais importantes da área)?			
	9. CITAÇÕES 9.1. As citações obedecem às normas da ABNT?			
	9.2. As informações retiradas de outros autores estão devidamente citadas?			
	10. AVALIAÇÃO GERAL (organização e apresentação gráfica, linguagem usada) 10.1. O texto obedece ao formato estabelecido?			
	10.2. A exposição do assunto é ordenada (as idéias estão bem encadeadas e a linguagem utilizada é clara)?			

A proposta de TCC deverá ser revisada, isto é, necessita de complementação, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou
- pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE.

PARECER: ( ) APROVADA ( ) NECESSITA DE COMPLEMENTAÇÃO

OBSERVAÇÕES:

Assinatura do(a) avaliador(a): _____ Local/data: _____

UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

GERADOR DE CÓDIGO A PARTIR DE ESPECIFICAÇÕES COM PADRÕES DE REQUISITOS

LEANDRO VILSON BATTISTI

BLUMENAU
2013

LEANDRO VILSON BATTISTI

GERADOR DE CÓDIGO A PARTIR DE ESPECIFICAÇÕES COM PADRÕES DE REQUISITOS

Proposta de Trabalho de Conclusão de Curso submetida à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso I do curso de Ciência da Computação — Bacharelado.

Profa. Joyce Martins - Orientadora

BLUMENAU
2013

1 INTRODUÇÃO

Segundo Ambler (2003, p. 21), o objetivo principal do desenvolvimento de software é construir sistemas de maneira mais eficaz e eficiente possível e que satisfaça a necessidade do cliente. Para atender este objetivo, podem ser usados vários modelos de processo de software, entre eles a prototipação. Esta abordagem permite evidenciar de forma mais rápida como ficará o software antes de entregá-lo ao usuário. No entanto, este e outros processos de software necessitam que requisitos sejam levantados junto ao usuário através da engenharia de requisitos.

Sommerville (2007, p. 97) diz que a principal preocupação na engenharia de requisitos "é criar e manter documentos de requisitos de [um] sistema". Dessa forma, uma vez identificados e negociados, os requisitos devem ser documentados em um nível apropriado de detalhes para que possam servir de base para o restante do processo de desenvolvimento. Em geral, é produzido um documento de especificação de requisitos, normalmente utilizando linguagem natural, de forma que todos os *stakeholders*[1] possam entendê-lo.

Após o levantamento e a documentação dos requisitos, cria-se um protótipo do sistema que possa ser apresentado aos *stakeholders* a fim de validar se o entendimento do engenheiro de software está de acordo com suas expectativas. Como o requisito é escrito em linguagem natural e o responsável por desenvolver o protótipo não necessariamente é o engenheiro de software, podem ocorrer pequenas diferenças entre o que está escrito e o protótipo apresentado. Tendo este cenário em mente, se o protótipo não atender às necessidades do usuário, o processo terá que recomeçar no levantamento ou na documentação dos requisitos para que os ajustes necessários, tanto nos requisitos quanto no protótipo, sejam feitos.

Para resolver o problema podem ser usados padrões de requisitos[2] que, segundo Marques (2008, p. 20), "vem sendo uma solução a ser adotada para agregar maior confiabilidade a [sic] etapa do levantamento de requisitos". Acredita-se também que se partes do protótipo, como as interfaces gráficas, tiverem o código correspondente gerado automaticamente durante a fase de levantamento de requisitos junto aos *stakeholders*, haverá

---

[1] "O termo *stakeholder* é usado para se referir a qualquer pessoa ou grupo afetado pelo sistema direta ou indiretamente. Os *stakeholders* incluem usuários finais que interagem com o sistema e todo o pessoal na organização que possa ser afetado por sua instalação". (SOMMERVILLE, 2007, p. 98).

[2] "Um padrão de requisito funciona como um guia para escrever um tipo particular de requisito. O padrão explica como descrever determinado tipo de requisito, como expressá-lo e revela possíveis requisitos adicionais implícitos" (MARQUES, 2008, p. 20).

uma proximidade maior entre o que o cliente espera e o que o software atende. Segundo Herrington (2003, p. xvii), geradores de código não apenas eliminam o trabalho pesado, mas proveem benefícios para o ciclo de vida da engenharia de software baseando-se em produtividade, qualidade, consistência e abstração.

Com base nestes fatos, propõe-se o desenvolvimento de um gerador de código para interfaces gráficas, baseado nas especificações de requisitos feitas utilizando padrões de requisitos de informação, pois, segundo Marques (2008, p. 21), focam "aspectos como a definição de [como] um item de informação deve ser apresentado ou representado". A ferramenta proposta terá como entrada requisitos escritos em língua portuguesa e como saída código para a interface correspondente.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é gerar código para interfaces gráficas de softwares a partir de especificações escritas utilizando padrões de requisitos.

Os objetivos específicos do trabalho são:

a) processar requisitos baseados em padrões de requisitos de informação;
b) processar requisitos escritos em língua portuguesa com vocabulário irrestrito;
c) processar componentes visuais mais comumente usados na construção de interfaces gráficas;
d) gerar metadados em JSON[3] correspondente à interface gráfica descrita;
e) gerar código fonte a partir dos metadados JSON em pelo menos uma linguagem de programação para validar a ferramenta.

1.2 RELEVÂNCIA DO TRABALHO

Considera-se que a complexidade do trabalho proposto está em como efetuar a

[3] JSON (JavaScript *Object Notation*) é um formato de intercâmbio de dados, baseado em um subconjunto de JavaScript, que possui similaridades com C, C++, C#, Java, JavaScript, entre outras (CROCKFORD, 2006, p. 1).

extração de características de interfaces gráficas a partir de requisitos escritos em língua portuguesa, sem restrição de vocabulário e que utilizam padrões de requisitos de informações. A partir da identificação de palavras que descrevam possíveis componentes da interface de um software, será gerada uma estrutura de metadados em formato aberto que possa ser utilizada para gerar interfaces gráficas em qualquer linguagem de programação.

Ainda, sabendo que, segundo Herrington (2003, p. 15), a geração automática de código traz benefícios para engenheiros de software e gerentes de projeto pois aumenta a qualidade e a consistência do código, além de proporcionar um desenvolvimento ágil de software, espera-se que a ferramenta proposta proporcione qualidade, produtividade e flexibilidade a equipes de desenvolvimento de software. A qualidade será obtida à medida que o engenheiro de software poderá avaliar a interface junto ao cliente enquanto efetua o levantamento de requisitos, sem ser necessário um segundo momento de validação da mesma. Além disso, códigos gerados tendem a serem padronizados. Uma vez que a interface gráfica será gerada de forma automatizada, o programador terá sua produtividade aumentada. Por fim, a flexibilidade está relacionada à utilização de um formato de saída aberto, sem gerar diretamente a interface para uma linguagem específica, permitindo que sejam geradas interfaces para diferentes linguagens.

1.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: realizar levantamento bibliográfico sobre geração automática de código, engenharia de requisitos, processamento de linguagem natural, JSON e trabalhos correlatos;
b) definição da linguagem de saída: definir para qual linguagem de programação será gerado o código das interfaces gráficas a partir do metadados JSON;
c) definição dos componentes suportados: definir quais componentes serão exportados para metadados a partir dos termos identificados nas descrições escritas seguindo padrões de requisitos de informações como sendo descritivos de interfaces gráficas;
d) especificação da ferramenta: especificar a ferramenta com análise orientada a objetos utilizando a *Unified Modeling Language* (UML). Será utilizada a

ferramenta Enterprise Architect para o desenvolvimento dos diagramas de casos de uso, de classes e de sequência (do analisador de requisitos, do gerador de metadados e do gerador de código);

e) implementação: implementar o analisador de requisitos, o gerador de metadados e o gerador de código conforme especificado na etapa anterior. Na implementação do analisador de requisitos, módulo responsável por processar o texto dos requisitos e identificar os componentes que compõem a interface, será utilizada a *Application Programming Interface* (API) JJSPEL[4]. Para implementar o gerador de metadados JSON, módulo que traduzirá os componentes identificados, será utilizado o ambiente de programação Eclipse 3.7 e a linguagem de programação Java na versão 1.6. Por fim, o gerador de código, responsável por transformar o arquivo JSON em código para uma interface gráfica em uma linguagem de programação, também será implementado utilizado o ambiente de programação Eclipse 3.7 e linguagem de programação Java na versão 1.6;

f) testes com especificações prontas: utilizar especificações exemplos já existentes como base para testar a ferramenta desenvolvida;

g) testes com voluntários: testar a ferramenta proposta com usuários voluntários. Os mesmos deverão descrever especificações de requisitos seguindo os padrões de requisitos de informações para, a partir desses, gerar código para interfaces gráficas. A eficiência da ferramenta será avaliada através de questionário.

As etapas serão realizadas nos períodos relacionados no Quadro 1.

Quadro 1 - Cronograma

	2014											
	jan.		fev.		mar.		abr.		maio		jun.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico												
definição da linguagem de saída												
definição dos componentes suportados												
especificação da ferramenta												
implementação												
testes com especificações prontas												
testes com voluntários												

4 JJSpell fornece uma API Java para JSpell. Assim os usuários podem utilizar o recurso JSpell em sistemas Java. JSpell é um analisador morfológico de código aberto (JANUÁRIO; LEITE; KOGA, 2010, p. 22).

2 REVISÃO BIBLIOGRÁFICA

A seção 2.1 fala sobre geração automática de código. A seção seguinte apresenta a engenharia de requisitos. A seção 2.3 trata do processamento de linguagem natural. A seção 2.4 descreve brevemente o JSON. Por último, na seção 2.5 são descritos os trabalhos correlatos.

2.1 GERAÇÃO AUTOMÁTICA DE CÓDIGO

O termo geração automática de código refere-se a escrever programas que escrevem programas (HERRINGTON, 2003, p. 3). Segundo Silveira (2006, p. 16), "a geração de código é uma técnica de construção de código que utiliza determinadas ferramentas para gerar programas". Estas ferramentas podem variar de *scripts* de ajuda a ferramentas que transformam modelos abstratos de lógica de negócio em aplicações completas.

Para Herrington (2003, p. xvii), geradores automáticos de código podem ser usados com o objetivo de gerar a infraestrutura do código, deixando para os engenheiros de software desafios de programação na solução dos problemas. Ainda para ele, a geração automática de código além de eliminar o trabalho pesado, provê benefícios para a engenharia de software, tais como, qualidade, consistência do código, desenvolvimento ágil de software e flexibilidade.

Na criação de um gerador de código deve-se seguir as seguintes etapas (HERRINGTON, 2003, p. 18): identificar a saída desejada, definir a entrada e como a mesma será analisada, interpretar e recuperar as informações da entrada necessárias para gerar a saída e gerar os arquivos de saída a partir da entrada.

2.2 ENGENHARIA DE REQUISITOS

"A Engenharia de Requisitos é o processo pelo qual os requisitos de um produto de software são coletados, analisados, documentados e gerenciados ao longo de todo o ciclo de

vida do software" (AURUM; WOHLIN, 2005 apud FALBO, 2012, p. 1).

Sommerville (2007, p.79) descreve que requisitos de um sistema são descrições, em linguagem natural, dos serviços e das restrições fornecidos pelo sistema e devem refletir a necessidade do cliente. Os requisitos podem ser divididos em duas categorias: funcionais e não funcionais. Os requisitos funcionais descrevem o que o sistema deve fazer, enquanto os não funcionais são restrições sobre funções oferecidas pelo sistema. Sabe-se, porém, que, segundo Decarle e Grahl (2008, p.1), um dos maiores problemas do desenvolvimento está no levantamento de requisitos por problemas como falha de comunicação e documentação, sendo que para dar maior confiabilidade aos requisitos pode-se adotar padrões de requisitos.

Um padrão de requisito é um guia para escrever um tipo de requisito, sendo que seu objetivo é que requisitos sejam escritos com mais qualidade, de forma mais rápida e com menos esforço. Um padrão é utilizado para descrever apenas um único requisito, pois contém informações detalhadas e específicas para esse requisito (WITHALL, 2008, p. 3). Um padrão descreve soluções para problemas recorrentes. Assim sendo, não há necessidade de estudar o mesmo problema novamente para identificar uma solução, produzindo assim um catálogo com soluções bem estruturadas, extensíveis e reutilizáveis de padrões de requisitos (TAGLIATI; JOHNSON; ROUSSOS, 2007, p. 4).

Withall (2008, p.5) apresenta a classificação dos padrões de requisitos em oito domínios: comercial, controle de acesso, entidade de dados, flexibilidade, função de usuário, fundamental, informação e performance. Estes grupos agregam 37 tipos de padrões de requisitos.

Um destes domínios é conhecido como domínio de informação.

> O domínio informação foca aspectos como a definição de um item de informação deve ser apresentado ou representado, como será a identificação única das entidades de dados, como será composto um determinado item de informação, como calcular determinado tipo de valor ou determiná-lo através de alguns passos lógicos, especificar a movimentação ou cópia de dados de um local para outro e por quanto tempo um certo tipo de dado deve ser mantido ou por quanto tempo deve estar disponível. (MARQUES, 2008, p. 21).

O Quadro 2 traz um exemplo de definição do padrão de requisito de tipo de dados, que pertence ao domínio de informação.

Quadro 2 – Exemplo de padrão de requisito

resumo	definição
nome do cliente	O nome do cliente deve ter no máximo 50 caracteres.
data de nascimento	A data de nascimento deve ser maior que 31/12/1950.
sexo	O sexo será representado por uma letra: “F” ou “M”.

2.3 PROCESSAMENTO DE LINGUAGEM NATURAL

O Processamento de Linguagem Natural (PLN) visa tratar a linguagem natural com o objetivo de desenvolver teorias computacionais, usando noções de algoritmos e estruturas de dados (ABRAHÃO, 1997, p. 10). Segundo Bhagat et al. (2012, p. 125), sistemas de PLN usam diferentes níveis de análise linguística, sendo elas: fonética, referente à análise dos sons; léxica, diz respeito à formação das palavras; sintática, analisa a constituição das frases; semântica, relacionada ao significado não só de cada palavra, mas também do conjunto resultante delas; pragmática, referente à interpretação das frases e morfológica.

> O analisador morfológico identifica palavras ou expressões isoladas em uma sentença, sendo este processo auxiliado por delimitadores (pontuação e espaços em branco). As palavras identificadas são classificadas de acordo com seu tipo de uso ou, em linguagem natural, categoria gramatical. (OLIVEIRA, 2002 apud OLIVEIRA NETO; TONIN; PIETRICH, 2010).

Ainda segundo os autores, o emprego do analisador morfológico é necessário, pois ele é essencial para identificar a formação de uma frase.

2.4 JSON

JSON é baseado em um subconjunto de JavaScript, sendo completamente independente de linguagem de programação, mas possuindo similaridades com C, C++, C#, Java, JavaScript, entre outras. Define regras de formatação para representação de dados estruturados (CROCKFORD, 2006, p. 1). Os dados são representados com informações de chave e valor, podendo ser aninhados. No Quadro 3 é possível visualizar componentes de descritos em JSON, a partir dos requisitos especificados no Quadro 2.

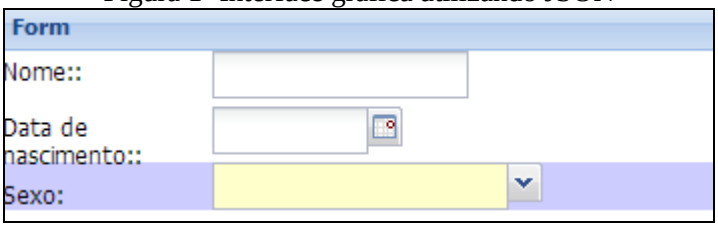
Quadro 3 – Exemplo de interface em JSON

```
{
   xtype:"form",
   title:"Form",
   items:[ {
                xtype:"textfield",
                fieldLabel:"Nome:",
                name:"textvalue"
          },{
                xtype:"datefield",
                fieldLabel:"Data de nascimento:",
                name:"datevalue"
          },{
                xtype:"combo",
                fieldLabel:"Sexo",
                name:"combovalue",
                hiddenName:"combovalue",

          } ]
}
```

A partir dos metadados do quadro anterior, pode-se, por exemplo, gerar a interface da Figura 1.

Figura 1- Interface gráfica utilizando JSON



2.5 TRABALHOS CORRELATOS

Não foram encontrados ferramentas ou trabalhos científicos que tratam da geração automática de código a partir de requisitos. Porém, existem ferramentas que geram diagramas da UML baseadas em linguagem natural, assim como existem diversos trabalhos envolvendo geração automática de código. São descritos os seguintes trabalhos: Paradigma (SILVA; MARTINS, 2008), a ferramenta para geração de classes a partir do PLN (BHAGAT et al., 2012) e Delphi2Java-II (SILVEIRA, 2006).

2.5.1 Paradigma

Silva e Martins (2008) apresentam uma ferramenta para geração de classes em UML,

baseada em processamento de linguagem natural.

Segundo Silva e Martins (2008, p. 142), a ferramenta etiqueta o texto de entrada (especificação de requisitos ou cenários de casos de uso) com marcações que identificam palavras em categorias morfossintáticas. Em seguida são aplicados padrões linguísticos, classificando cada palavra como classe, atributo ou operação. Também são identificadas as associações entre as classes e suas multiplicidades. Por último, as classes são geradas de forma automática. O usuário deve "interagir no processo automatizado para vincular os atributos e operações às respectivas classes" (SILVA; MARTINS, 2008, p. 142).

2.5.2 Ferramenta para geração de classes a partir de PLN

No estudo de Bhagat et al. (2012) são apresentadas as etapas para a extração de informações escritas em linguagem natural para a geração de diagramas de classe. A regra básica para a extração é analisar o texto em linguagem natural e extrair elementos de modelagem orientada a objetos. Os passos são (BHAGAT et al., 2012, p. 126):

a) processar os requisitos do sistema, extraindo os *tokens*;
b) fazer a análise morfológica das palavras;
c) efetuar a análise sintática validando frases de acordo com as regra gramaticais da língua inglesa, para identificar objetos, sujeitos, ações, atributos, entre outros;
d) realizar a análise semântica para identificar as associações, os métodos e os atributos de cada objeto, sendo substantivos e sujeitos identificados como objetos, verbos como métodos e adjetivos como atributos dos objetos;
e) criar associações e relações entre as classes e os objetos extraídos a partir das preposições existentes;
f) gerar um modelo lógico do diagrama de classes com base nas informações extraídas nos passos anteriores;
g) desenhar o diagrama de classes a partir do modelo lógico;
h) gerar código em VB.NET ou Java a partir do modelo lógico.

2.5.3 Delphi2Java-II

Silveira (2006) apresenta uma ferramenta para realizar a conversão de formulários elaborados na linguagem de programação Delphi para aplicações na linguagem de programação Java.

Segundo Silveira (2006, p. 19), foram seguidas as seguintes etapas para desenvolver a ferramenta:

a) identificar a saída desejada: interfaces gráficas Java desenvolvidas em *Swing*;
b) definir a entrada: formulários Delphi, um arquivo com extensão `dfm`;
c) interpretar e recuperar as informações da entrada, definindo a formatação e a geração de saída;
d) gerar a saída a partir da informações extraídas do arquivo de entrada.

No processamento dos formulários Delphi são utilizados analisadores léxico, sintático e semântico. Na geração da saída das classes Java é utilizado o motor de *templates* Velocity.

3 REQUISITOS DO SISTEMA A SER DESENVOLVIDO

A ferramenta deverá:

a) permitir como entrada textos escritos em língua portuguesa descrevendo um requisito funcional, sendo processado um requisito por vez (Requisito Funcional – RF);
b) identificar no texto de entrada palavras que descrevam possíveis componentes da interface gráfica de um software (RF);
c) identificar no texto de entrada palavras que descrevam possíveis validações dos componentes da interface gráfica (RF);
d) gerar metadados com os componentes de interface identificados (RF);
e) gerar código fonte de interfaces gráficas a partir dos metadados gerados (RF);
f) ser desenvolvida em Java 1.6 (Requisito Não Funcional - RNF);
g) utilizar a biblioteca JJSpell (RNF);
h) utilizar JSON para exportar os metadados (RNF).

4 CONSIDERAÇÕES FINAIS

Pode-se afirmar que existem problemas relacionados ao desenvolvimento de interfaces gráficas de sistemas, citando-se: inconsistências entre as descrições textuais dos requisitos e as interfaces apresentadas e falta de produtividade e de qualidade das interfaces projetadas manualmente baseadas nestas descrições textuais. Assim sendo, foi proposto o desenvolvimento de uma ferramenta voltada para a automatização da geração de código a partir de requisitos, escritos em português com vocabulário irrestrito utilizando padrões de requisitos de informação . Espera-se, com isso, atender às expectativas dos *stakeholders* quanto às interfaces gráficas, bem como permitir que desenvolvedores possam direcionar o foco para a solução de problemas mais complexos.

Tendo como entrada requisitos textuais, com o auxílio do processamento de linguagem natural serão identificadas e extraídas palavras que representem componentes de interface. Como saída será gerado um arquivo JSON, descrevendo a interface especificada, bem como o código fonte correspondente em alguma linguagem de programação.

Com relação aos trabalhos correlatos pode-se afirmar que a ferramenta diferencia-se em alguns aspectos. Pretende-se que a ferramenta proposta trate vocabulário irrestrito da língua portuguesa, sem ser necessário que o usuário interaja no processo como ocorre com a ferramenta Paradigma. Também diferente da ferramenta Paradigma, que gera classes UML, a ferramenta proposta irá gerar metadados JSON e código em linguagem de programação para a interface desejada. Quanto ao trabalho de Bhagat et al. (2012), as diferenças estão relacionadas ao tratamento dado à entrada e a saída gerada. Bhagat et al. (2012) identifica classes, objetos e suas relações e gera diagramas de classes e código em VB.NET e Java. O presente trabalho identificará componentes de interface para gerar metadados JSON e código em linguagem de programação. Quanto ao terceiro trabalho analisado, embora não faça processamento de linguagem natural, já que a entrada são arquivos `.dfm`, é um gerador de código de interfaces, como o trabalho proposto.

REFERÊNCIAS BIBLIOGRÁFICAS

ABRAHÃO, Paulo R. C. **Modelagem e implementação de um léxico semântico para o português**. 1997. 118 f. Dissertação (Mestrado em Informática) - Instituto de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre. Disponível em: <http://www3.pucrs.br/pucrs/files/uni/poa/facin/pos/dissertacoesdef/paulo>. Acesso em: 14 set. 2013.

AMBLER, Scott W. **Modelagem ágil**: práticas eficazes para a programação eXtrema e o processo unificado. Porto Alegre: Bookman, 2003.

BHAGAT, Sujata et al. Class diagram extraction using NLP. In: INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ENGINEERING & TECHNOLOGY - SPECIAL ISSUE OF INTERNATIONAL JOURNAL OF ELECTRONICS, COMMUNICATION & SOFT COMPUTING SCIENCE & ENGINEERING, 1st, 2012, Meerut. **Proceedings…** Meerut: Subharti Institute of Technology & Engineering, 2012. p. 125-128. Disponível em: <http://www.ijecscse.org/papers/SpecialIssue/comp2/190.pdf>. Acesso em: 14 set. 2013.

CROCKFORD, Douglas. **RFC 4627**: the application / json media type for JavaScript Object Notation (JSON). [S.l.], 2006. 10 p. Disponível em: <http://www.ietf.org/rfc/rfc4627.txt?number=4627>. Acesso em: 11 set. 2013.

DECARLE, Luiz S.; GRAHL, Everaldo A. **Experiência prática de aplicação de padrões de requisitos de software**. 2008. 9 f. Artigo de Conclusão de Curso (Especialização em Gestão de Desenvolvimento de Software), Instituto Catarinense de Pós-Graduação, Blumenau.

FALBO, Ricardo A. **Engenharia de requisitos**. Espírito Santo, 2012. Disponível em: <http://www.inf.ufes.br/~falbo/files/Notas_Aula_Engenharia_Requisitos.pdf>. Acesso em: 14 set. 2013.

HERRINGTON, Jack. **Code generation in action**. Greenwich: Manning, 2003.

JANUÁRIO, Guilherme C.; LEITE, Leonardo A. F.; KOGA, Marcelo L. **Poli-Libras:** um tradutor de Português para Libras. 2010. 93 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) – Escola Politécnica, Universidade de São Paulo, São Paulo. Disponível em: <http://www.polilibras.com.br/documentos/monografia.pdf?attredirects=0&d=1>. Acesso em: 08 set. 2013.

MARQUES, Tiago W. **Gerência de requisitos com adoção de padrões**. 2008. 124 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://campeche.inf.furb.br/tccs/2008-II/2008-2-26-vf-tiagowmarques.pdf>. Acesso em: 25 out. 2013.

OLIVEIRA NETO, João M.; TONIN, Sávio D.; PIETRICH, Soraia S. Processamento de linguagem natural e suas aplicações computacionais. In: ESCOLA REGIONAL DE INFORMÁTICA, 2., 2010, Manaus. **Anais eletrônicos...** Manaus: INPA, 2010. Não paginado. Disponível em: <http://www.inpa.gov.br/erin2010/Artigo/Artigo9.pdf>. Acesso em: 01 set. 2013.

SILVA,Wilson C., MARTINS, Luis E. G. Paradigma: uma ferramenta de apoio à elicitação e modelagem de requisitos baseada em processamento de linguagem natural. In: WORKSHOP ON REQUIRIMENTS ENGINEERING, 11th, 2008, Barcelona. **Proceedings...** Barcelona: [S.n.], 2008. p. 140-151. Disponível em <http://www.inf.puc-rio.br/wer/WERpapers/artigos/artigos_WER08/silva.pdf>. Acesso em: 14 set. 2013.

SILVEIRA, Janira. **Extensão da ferramenta Delphi2Java-II para suportar componentes de banco de dados**. 2006. 81 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://campeche.inf.furb.br/tccs/2006-I/2006-1janirasilveiravf.pdf>. Acesso em: 14 set. 2013.

SOMMERVILLE, Ian. **Engenharia de software**. 8. ed. Tradução Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, Edílson de Andrade Barbosa. São Paulo: Pearson Addilson Wesley, 2007.

TAGLIATI, Luca V.; JOHNSON, Roger; ROUSSOS, George. **Requirements analysis evolution through patterns**. Londres, 2007. Disponível em: <http://www.dcs.bbk.ac.uk/~gr/pdf/seke07_requirement_patterns_2.pdf>. Acesso em: 25 out. 2013.

WHITALL, Stephen. **Software requirements patterns**. [S.l.], 2008, Disponível em: <http://www.withallyourequire.com/software_requirements_patterns.pdf>. Acesso em: 25 out. 2013.