Geração Automática de Código Java a partir do Modelo de Dados

Jansser Ribeiro, Renato Lima Novais, Thiago Souto Mendes

Grupo de Informática Aplicada - Instituto Federal da Bahia – Campus Santo Amaro
CEP 44.200-000 – Santo Amaro – BA – Brasil

jansser_costa@hotmail.com, renatonovais@gmail.com, tsoutom@gmail.com

Abstract. *In software development one has to often realize activities such as insert, update, select and delete of basic entities of the system. Such activities are repetitive and impact on cost and construction time of the software. This paper presents a tool for automating these activities from the data model. A study of function points was conducted to evaluate the impact of using the tool.*

Resumo. *No desenvolvimento de software é recorrente a realização de tarefas de cadastros básicos das entidades do sistema. Tais tarefas são repetitivas e impactam nos custos e prazos. O trabalho apresenta uma ferramenta de automatização dessas tarefas a partir do modelo de dados. Um estudo de pontos por função foi realizado para ver o impacto do uso da ferramenta.*

1. Introdução

No contexto de desenvolvimento de software de Sistema de Informação (SI), é recorrente a necessidade de implementação de cadastros básicos associados aos requisitos funcionais ou regras de negócio [Sommerville 2007, p. 18]. Esses cadastros são pré-requisitos para a utilização das principais funcionalidades do sistema. Como por exemplo, um software de venda, que tem como um de seus principais requisitos funcionais a venda de produto, necessita previamente que haja o cadastro de diversas entidades básicas do sistema, como: cliente, fornecedor, produto, bairro, cidade, etc. Para cada uma dessas entidades, é necessário fazer as quatro operações básicas que são: inclusão, atualização, listagem e exclusão.

Considerando a abordagem de orientação a objetos para a construção de SI, devem ser criadas, para cada uma das entidades básicas do sistema, tanto as classes de persistência que manipulam o banco de dados, quanto às classes que implementam as interfaces com o usuário.

Essas tarefas básicas consomem um tempo considerável do processo de desenvolvimento de software [Sommerville 2007, p. 281]. Por outro lado, seria interessante se este tempo pudesse ser dedicado principalmente às tarefas de implementação das funcionalidades mais importantes dos sistemas. É possível observar que existe uma repetição no desenvolvimento dessas tarefas. Da mesma forma, existe um padrão de codificação das mesmas, permitindo que elas possam ser automatizadas. Isto pode ser feito, baseando-se nas informações dos metadados das tabelas, os quais definirão como serão montados os SQLs e as telas das tabelas básicas.

Tendo em vista esse contexto, o objetivo deste trabalho é apresentar a ferramenta GEA (Geração de Código Automático) que proporciona uma alternativa para

automatizar a realização das tarefas de codificação citadas anteriormente. Essa ferramenta comunicasse com dois bancos de dados bastante utilizados na atualidade, *MySQL* [MySQL 2011] e *PostgreSQL* [PostgreSQL 2011], e recupera os metadados das tabelas. Com posse dos metadados ela gera, de maneira eficiente, eficaz e simples, toda estrutura básica do software na linguagem JAVA. A GEA foi desenvolvida como *plug-in* do Eclipse, uma das IDEs (*Integrated Development Environment*) mais utilizadas atualmente [Cardoso 2009].

Para avaliar a eficiência da ferramenta foi realizado um estudo de caso através da realização da APF (Análise dos Pontos de Função) do sistema e a comparação dos resultados obtidos no desenvolvimento do SIGE (Sistema de Gerenciamentos de Eventos) na versão para Desktop. O SIGE é um sistema desenvolvido pelo GIA (Grupo de Informática Aplicada). Atualmente existe uma versão web do sistema no endereço (http://www.gia.ifba.edu.br/eventos).

Neste artigo, a Seção 2 apresenta a ferramenta desenvolvida, descrevendo suas funcionalidades gerais, a Seção 3 apresenta o processo de extração de metadados do *MySQL* e do *PostgreSQL*. A Seção 4 apresenta a análise de um estudo de caso e a Contagem de Pontos de Função. A Seção 5 as vantagens e desvantagens da GEA. A Seção 6 traz a relação dos trabalhos relacionados. Finalmente, conclui-se o artigo com a Seção 7, apresentando as considerações finais e os trabalhos futuros.

2. A Ferramenta GEA

A ferramenta GEA foi desenvolvida utilizando a linguagem Java e integrada ao IDE Eclipse sob a forma de *plug-in*. Na Figura 1 são apresentados os principais passos do funcionamento da ferramenta: primeiramente a ferramenta conecta com o banco de dados escolhido (i), extrai os metadados (ii) e por fim, gera as classes do programa (iii).

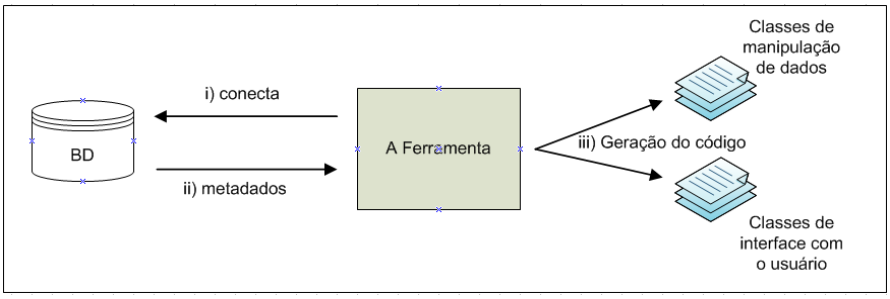

Figura 1. Funcionamento geral da ferramenta.

Tendo em vista as particularidades estruturais de cada SGBD (Sistema Gerenciador de Banco de Dados) utilizado, procurou-se desenvolver a ferramenta de uma maneira que pudesse ser estendida para outros SGBDs. Para tal, foi utilizado o padrão de projeto *Strategy*. A partir disso, foi desenhada primeiramente uma solução para os sistemas, *MySQL* e *PostgreSQL*, que será detalhada a seguir.

2.1. Extração de metadados do *MySQL* e do *PostgreSQL*

Para que um sistema seja desenvolvido a partir de um modelo de dados faz-se necessário o conhecimento dos metadados desse modelo. Através da extração dos metadados é possível extrair as informações necessárias para a geração do código fonte.

Nesta ferramenta foi realizada a extração dos metadados através da utilização de

APIs (*Application Programming Interface*) do JDBC (*Java Database Connectivity*). As principais informações a serem extraídas pela ferramenta são: a) os nomes das tabelas contidas em um banco de dados; b) as colunas de uma determinada tabela e os seus tipos.

A extração dos metadados foi feita utilizando as classes *DatabaseMetaData* e a *ResultSetMetaData* da API JDBC. Tais classes possuem métodos que facilitam a captura de metadados, como por exemplo, o método "*getSchemas*();" que retorna todos os *schemas* de um servidor de banco de dados.

Cada SGBD possui especificidades que os diferenciam. Como exemplo pode ser citado o *MySQL*, que identifica um banco de dados como um "*catalog*". Por sua vez, o *PostgreSQL* utiliza o termo "*schemas*". Além disso, existem nomenclaturas diferentes para tipos de dados nesses dois SGBDs, as quais devem ser tratadas pela ferramenta. Tais diferenças vão influenciar diretamente na implementação das APIs do JDBC.

2.2. Tipo de dados

Uma questão fundamental para se desenvolver sistemas de geração automática a partir do modelo de dados é o mapeamento dos tipos de dados do SGBD para a linguagem escolhida. A Tabela 1 apresenta um mapeamento dos tipos de dados do *MySQL* para tipos de dados Java, componentes *Swing* e ainda componentes *HTML*.

Tabela 1: Mapeamento dos tipos dos dados do MySQL para os tipos do Java.

Tipo do Banco de dados	Tipo Java	Componente Swing	Componente Web (HTML)
VARCHAR	String	Text Field	Input Text
TEXT	String	Text Area	Input Text
BIT	Boolean	Radio Button ou CheckBox	Radio Button ou CheckBox
SmallInt	Short	Text Field	Input Text
Decimal	Java.math.BigDecimal	Text Field	Input Text
Int	Int	Text Field	Input Text
Integer	Int	Text Field	Input Text
Float	Double	Text Field	Input Text
Double	Double	Text Field	Input Text
Real	Float	Text Field	Input Text
Binary	Byte[]	Text Field	Input Text
Date	Java.sql.Date	Text Field	Input Text
Time	Java.sql.Time	Text Field	Input Text

2.3. Geração de código

Após a extração dos metadados, é feita a transformação das informações extraídas em código-fonte. Para a geração automática de código foram utilizadas APIs fornecidas pelo JDT (*Java Development Tools*), que permitem acessar e manipular código-fonte Java [Vogella 2010]. Uma classe no JDT é chamada de "*ICompilationUnit*". A Figura 2 mostra a criação de uma classe de maneira programática.

```
String sourceCode = "public Class BD {}";
ICompilationUnit ClassDB = pacote.createCompilationUnit("BD.java", sourceCode, true, null);
```

Código fonte da classe.

Pacote

Nome do arquivo

Código fonte

Figura 2. Gerando uma classe com JDT.

3. Exemplo de uso

Nesta seção é apresentado um exemplo de uso da ferramenta. Na Figura 3 são apresentadas as telas de configuração utilizadas para geração do código fonte. Na (Figura 3a) tem-se a tela na qual o usuário deve informar os parâmetros de configuração para acesso ao sistema gerenciador banco de dados. Após essa etapa, o usuário deve selecionar o banco de dados (*schema*) com o qual deseja trabalhar (Figura 3b). Por fim, todas as tabelas do banco de dados selecionado são apresentadas (Figura 3c). O usuário então seleciona as tabelas de interesse e clica em "OK". Passado todas essas etapas, o código fonte será gerado para as tabelas selecionadas.

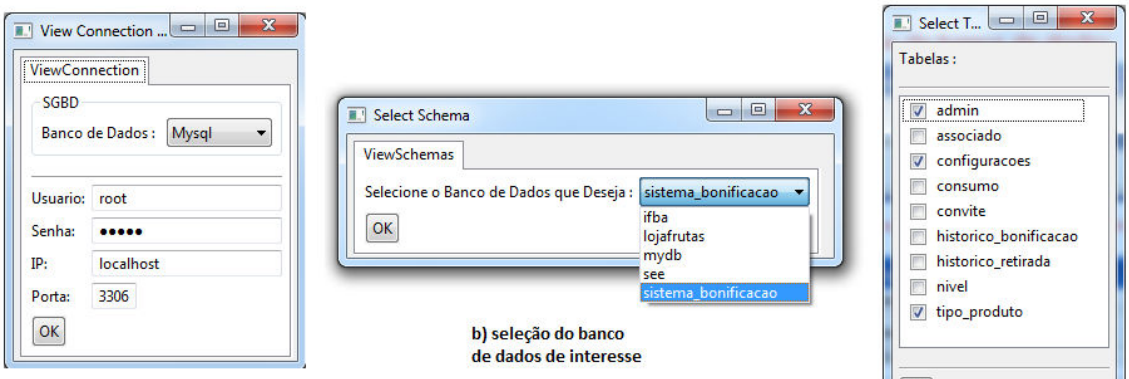

Figura 3. Telas de configuração do sistema GEA.

Na Figura 4 é apresentada da estrutura do projeto Java gerado no Eclipse, mostrando cada uma das classes geradas para a tabela selecionada. São geradas classes *Beans* que representam cada tabela selecionada do banco de dados. No pacote 'bd', as classes de manipulação e acesso ao banco de dados são criadas. As classes que implementam as interface com o usuário ficam localizadas no pacote *view*. É importante notar que o sistema gerado utiliza o padrão arquitetural MVC (*Model View Controller*), e padrões de projeto, dentre eles, DAO (*Data Access Object*), *Singleton*, *Facade* [Gamma et al. 1995].

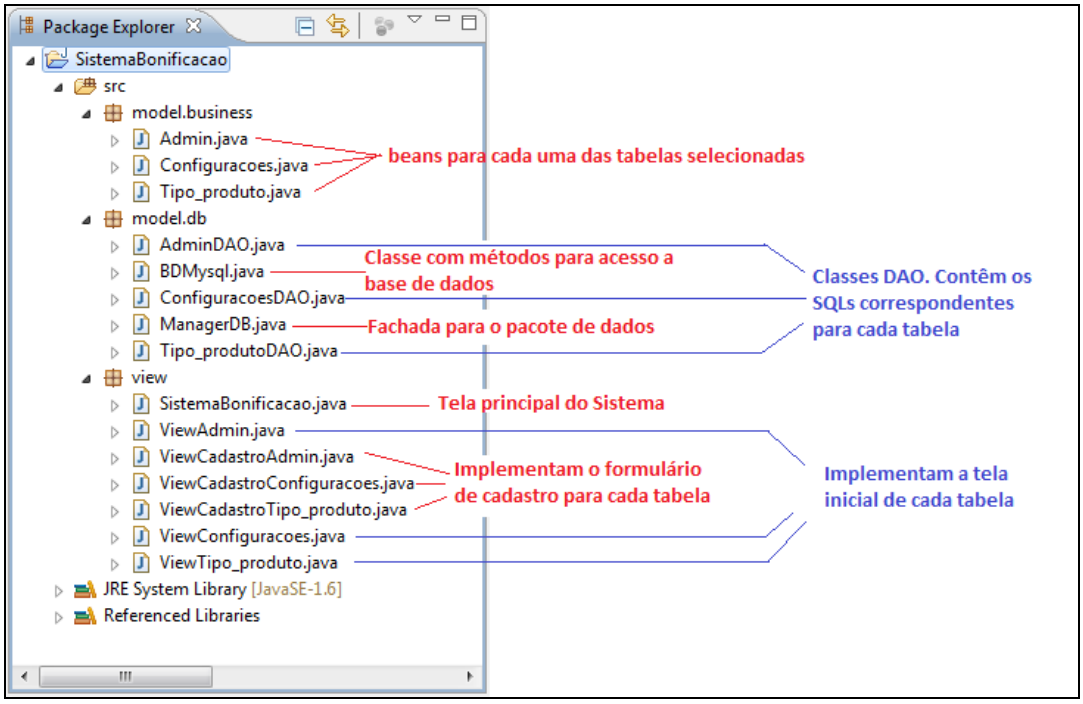

Figura 4. Classes do programa gerado.

Na Figura 5 é possível observar algumas das telas do programa gerado pela ferramenta já em execução. A primeira tela – (Figura 5a) – representa a tela principal do sistema. Observe que no menu Iniciar é criado uma opção para cada tabela gerada. Ao clicar em uma dessas opções desse menu ('Configuracoes', por exemplo), é aberta a tela Inicial de tabela, onde têm-se a listagem dos dados, e as opções de Inserir, Atualizar, Excluir e Sair (Figura 5b). A tela apresentada na (Figura 5c) representa o formulário utilizado para Inserir/Atualizar os dados da tabela Configuracoes.

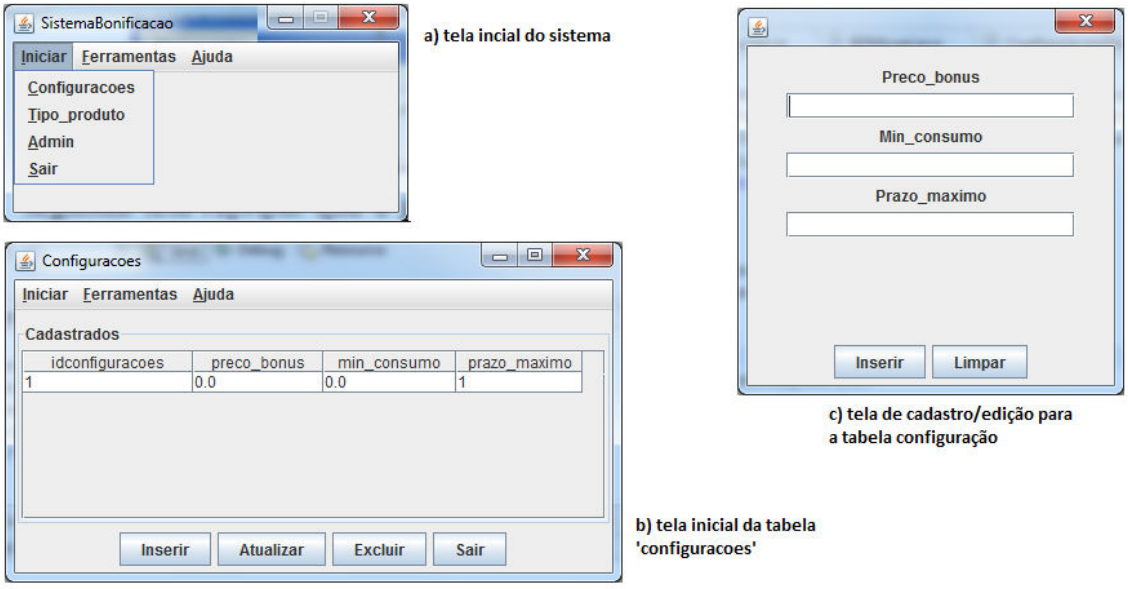

Figura 5. Algumas telas do sistema gerado.

4. Estudo de Caso

4.1. Escopo do Estudo de Caso

Na intenção de validar a ferramenta GEA foi feito um estudo de caso para realizar a medição e análise do SIGE. O objetivo do SIGE é apoiar as atividades de gerenciamento dos eventos e acompanhamento das atividades de capacitação dos servidores de instituições de ensino.

A Tabela 2 apresenta as principais necessidades e funcionalidades do SIGE, retiradas do Documento de Requisitos do projeto elaborado pelo o analista do grupo.

Tabela 2. Funcionalidades do Sistema de Gerenciamento de Eventos (SIGE).

Requisitos do SIGE		
Requisitos	**Descrição das Funcionalidades**	**Atores**
REQ 1.1	Cadastrar Usuários do Sistema	Admin. do Evento /Admin. Geral
REQ 1.2	Cadastrar evento	Admin. Geral
REQ 1.3	Gerenciar evento	Admin. do Evento / Admin. Geral
REQ 1.4	Gerenciar cronograma do Evento	Admin. do Evento / Admin. Geral
REQ 1.5	Consultar cronograma do Evento	Admin. do Evento / Admin. Geral
REQ 1.6	Consultar eventos por data e local	Admin. do Evento / Admin. Geral
REQ 1.7	Consultar detalhes de um evento	Admin. do Evento / Admin. Geral
REQ 1.8	Emitir Certificado de Participantes	Admin. do Evento / Admin. Geral
REQ 1.9	Consultar acompanhamento do pessoal capacitado	Admin. do Evento / Admin. Geral
REQ 1.10	Gerar Consultas Gerenciais (Gráficos e Relatórios) de Capacitação	Admin. do Evento / Admin. Geral

É importante ressaltar que o SIGE foi desenvolvido através da metodologia de processo de desenvolvimento RUP (*Rational Unified Process*) [RUP 2011]. Segundo [Silva et al. 2006], o RUP é um processo de engenharia de software bem definido e bem estruturado que define claramente quem é responsável pelo que, como as coisas devem ser feitas e quando fazê-las. O RUP também provê uma estrutura bem definida para o ciclo de vida de um projeto RUP, articulando claramente os marcos essenciais e pontos de decisão; tornando-se uma maneira de desenvolvimento de software que é iterativa, centrada à arquitetura e guiada por casos de uso. Por fim, o RUP utiliza a UML (Linguagem Unificada de Modelagem) para especificar, modelar e documentar artefatos.

Para realizar a medição do SIGE elegeu-se a APF que é uma técnica bastante utilizada para medir o tamanho do software, que visa estabelecer uma medida de tamanho do software em Pontos por Função [Vazquez et al. 2005]. Também porque é sugerida para as empresas que pretendem se certificar com MPS.BR (Melhoria de Processos do Software Brasileiro) [Softex 2011].

4.2. A Medição do SIGE sem a Utilização da GEA

Segue abaixo na Tabela 3, o detalhamento da contagem estimativa de Pontos de Função do Sistema de Gerenciamento de Eventos SIGE sem considerar a utilização da ferramenta GEA.

Tabela 3. Estimativa de Tamanho do Sistema de Gerenciamento de Eventos (SIGE).

Descrição da Função	Tipo	Complexidade	Tamanho (PF)
Usuários	ALI	Simples	7 PF
Incluir usuário do sistema	EE	Simples	3 PF
Alterar usuário do sistema	EE	Simples	3 PF
Excluir usuário do sistema	EE	Simples	3 PF
Lista de usuários do sistema	CE	Simples	3 PF
Consultar usuários do sistema	CE	Simples	3 PF
Controle de acesso da aplicação	SE	Simples	4 PF
Alterar senha	EE	Simples	3 PF
Esqueceu senha	SE	Simples	4 PF
Evento	ALI	Média	10 PF
Incluir evento	EE	Média	4 PF
Alterar evento	EE	Média	4 PF
Gerenciar evento	EE	Média	4 PF
Consultar plano do evento	CE	Média	4 PF
Definir cronograma do evento	EE	Média	4 PF
Consultar cronograma do evento	SE	Média	5 PF
Consultar eventos por data e local	SE	Média	5 PF
Consultar detalhes do evento	CE	Complexa	6 PF
Incluir participantes para evento	EE	Média	4 PF
Alterar participantes para evento	EE	Média	4 PF
Excluir participantes para evento	EE	Simples	3 PF
Consultar participantes cadastrados no evento	CE	Média	4 PF
Enviar para e-mail para participação do evento	SE	Média	5 PF
Emitir certificado para o participante	SE	Média	5 PF
Lista de participantes com emissão de certificado pendente	CE	Simples	3 PF
Relatórios gerenciais (1 gráficos e 2 relatórios com dados calculados)	3 SE	Média	15 PF
Total			**122 PF**

O total da medição do tamanho do SIGE descritas na Tabela 3 resulta em 122 pontos de função não ajustados. Após a aplicação do fator de ajuste no valor de 1,1, tem-se uma estimativa de 134,2 horas a serem gastas no desenvolvimento do SIGE.

Com base na produtividade histórica dos projetos já realizados pelo grupo de pesquisa GIA será desenvolvido em JAVA por uma equipe com experiência intermediária na plataforma. Assim, o índice de produtividade utilizado é o de 10 Horas/PF. Então, a estimativa de esforço é de: 134,2 × 10 = 1342 HH (Homens\Hora).

Levando-se em conta que a equipe de desenvolvedores é formada por 3 estagiários que trabalham 4 Horas/Dia, mas que o trabalham efetivamente 3 horas/dia. E ainda, em média um mês possui 22 dias úteis. O prazo deste projeto será medido por

prazo = (Esforço em HH) / (Tamanho Equipe*Horas Trabalhadas Dia*Dias Úteis Mês), ou seja, prazo=6,8 meses.

Portanto, é possível verificar que foi um sistema que exigiu um tempo de desenvolvimento bem razoável e grande parte do esforço foi gasto no desenvolvimento das funções de inclusão, alteração e exclusão que poderiam ser facilmente eliminado caso fossem geradas através de uma ferramenta de geração automática de código fonte como a GEA.

4.3. A Medição do SIGE com a Utilização da GEA

Com o objetivo de validar a utilização da GEA o grupo de pesquisadores utilizou-se o modelo de dados do sistema SIGE para gerar automaticamente as telas de Inclusão, Alteração e Exclusão das tabelas Usuário e Evento, assim as principais telas do sistema foram geradas com pouco esforço como se pode observar através da Tabela 4, onde estão destacadas as funcionalidades que foram eliminadas da contagem de pontos de função.

Tabela 4. Estimativa de Tamanho do Sistema de Gerenciamento de Eventos (SIGE) sem as funcionalidades eliminadas após a utilização da GEA.

Descrição da Função	Tipo	Complexidade	Tamanho (PF)	Excluído
Usuários	ALI	Simples	7 PF	Não
Incluir usuário	EE	Simples	3 PF	Sim
Alterar usuário	EE	Simples	3 PF	Sim
Excluir usuário	EE	Simples	3 PF	Sim
Lista de usuários	CE	Simples	3 PF	Sim
Consultar usuários	CE	Simples	3 PF	Sim
Controle de acesso da aplicação	SE	Simples	4 PF	Não
Alterar senha	EE	Simples	3 PF	Não
Esqueceu senha	SE	Simples	4 PF	Não
Evento	ALI	Média	10 PF	Não
Incluir evento	EE	Média	4 PF	Sim
Alterar evento	EE	Média	4 PF	Sim
Gerenciar evento	EE	Média	4 PF	Sim
Consultar plano do evento	CE	Média	4 PF	Sim
Definir cronograma do evento	EE	Média	4 PF	Sim
Consultar cronograma do evento	SE	Média	5 PF	Sim
Consultar eventos por data e local	SE	Média	5 PF	Sim
Consultar detalhes do evento	CE	Complexa	6 PF	Sim
Incluir participantes para evento	EE	Média	4 PF	Não
Alterar participantes para evento	EE	Média	4 PF	Não
Excluir participantes para evento	EE	Simples	3 PF	Não
Consultar participantes cadastrados no evento	CE	Média	4 PF	Não
Enviar para e-mail para participação do evento	SE	Média	5 PF	Não
Emitir certificado para o participante	SE	Média	5 PF	Não

Lista de participantes com emissão de certificado pendente	CE	Simples	3 PF	Não
Relatórios gerenciais (1 gráficos e 2 relatórios com dados calculados)	3 SE	Média	15 PF	Não
Total			**76 PF**	

A totalização do tamanho do SIGE descritas na Tabela 4 resultou em 76 pontos de função não ajustados. Após a aplicação de um fator de ajuste no valor de 1,1, tem-se uma estimativa de 83,6, horas a serem gastas no desenvolvimento do SIGE de acordo com as estimativas descritas na seção 4.3. Então, a estimativa de esforço foi de: 83,6×10 = 836 HH (Homens\Hora). Já o prazo do projeto medido foi de 3,2 meses. Porém, o prazo gasto foi maior do que o previsto, no total de 4,0 meses, pois mesmo com a utilização da ferramenta GEA foi necessária à realização de ajustes nas telas geradas automaticamente, além da integração com as demais telas construídas manualmente.

4.3. Resultados Obtidos

Através da utilização da GEA foi possível gerar as principais telas do sistema rapidamente e eliminar grande parte do esforço gasto no desenvolvimento apenas utilizando o modelo de dados. Mesmo sendo necessária a realização de ajustes a ferramenta se mostrou eficiente e de fácil utilização, pois como é possível observar na Tabela 5 a diferença do prazo foi de 2,8 meses, por exemplo. Principalmente, no desenvolvimento de funcionalidades básicas de cadastro presentes em praticamente todos os sistemas desenvolvidos e que poderiam ser facilmente geradas.

Tabela 5. Estimativa de Tamanho do Sistema de Gerenciamento de Eventos (SIGE) sem as funcionalidades eliminadas após a utilização da GEA.

Parâmetros	SIGE (Sem a utilização da GEA)	SIGE (Com a utilização da GEA)	Diferença
PF não ajustados	122	76	46
PF ajustados (Fator=1,10)	134,2	83,6	50,6
Esforço (Homens\Hora)		836	506
Produtividade (Horas por PF)	10	10	0
Tamanho da Equipe	3	3	0
Horas trabalhadas por dia	4	4	0
Dias Úteis no Mês	22	22	0
Prazo (real)	6,8	4,0	2,8

5. Vantagens e Desvantagens

Após o estudo realizado é possível afirmar que a ferramenta traz diversas vantagens e desvantagens, dentre os quais, podem ser citadas: (i) Aumento da produtividade da equipe de desenvolvimento; (ii) Ganho de qualidade interna do projeto, uma vez parte do código fonte é gerado automaticamente; (iii) Garantia de eficácia, pois existe uma redução dos erros, pois, por mais que as atividades são básicas, desenvolvimento de software sempre está sujeito a inclusão de defeitos, etc.; (iv) Código com qualidade, uma vez que são implementados pela ferramenta e podem ser gerados seguindo as boas práticas de desenvolvimento da empresa;

As principais desvantagens e limitações encontradas foram: (i) A ferramenta não gera sistemas web; (ii) É necessária a criação de interfaces de comunicação com outros SGDBs; (iii) As telas seguem sempre o mesmo formato, fazendo com que os sistemas fiquem sempre com o mesmo *template*. Servindo assim para família de sistemas. Uma solução para isso seria a utilização de diferentes *templates* de tela. A geração seria feita baseado no *template* escolhido pelo usuário.

6. Trabalhos Relacionados

Os trabalhos de pesquisa envolvendo geração de código com acesso livre a comunidade acadêmica são escassos. A pesquisa revelou que a maioria dos *framewoks* é desenvolvida pela indústria para atender suas necessidades especificas. O que dificulta a evolução dos trabalhos nessa área. Porém, foram encontradas abordagens interessantes sobre o tema que serão descritas a seguir e comparadas com a ferramenta GEA.

A ferramenta IZCode [Izcode 2010] é um gerador de softwares comerciais de propósito especifico. Isto significa que ele é muito eficiente em gerar sistemas comerciais e se limita a isto. A geração de software através IZCode, parte do princípio que todo software comercial segue um padrão pré-determinado de codificação, design de tela e funcionalidades. Através dessa ferramenta é possível gerar códigos em JAVA e .NET. Mas, para ter acesso a todas as funcionalidades do sistema é necessário comprar a ferramenta diferentemente da GEA.

O e-Gen [Egen 2010] é uma ferramenta disponibiliza um conjunto de recursos e facilidades que auxiliam a equipe de desenvolvimento e enriquecem a aplicação gerada. Essa ferramenta disponibiliza um IDE *(Integrated Development Environment*) que proporciona rapidez no desenvolvimento e garante a robustez das aplicações. Porém, o suporte a ferramenta é pago e a sua conexão com o banco de dados só funciona em SGDB comerciais.

O FUMIGANT [Adamatti 2006] permite a geração de código fonte para a linguagem de programação Java. Essa ferramenta permite ganhar rapidez e padrão no desenvolvimento de atividades repetitivas e produtividade na construção de sistemas. O grande problema desta ferramenta é que utiliza diversos frameworks para permitir a geração de código o que dificulta a sua implementação.

Em resumo a ferramenta desenvolvida apresenta três características que a diferenciam dos outros trabalhos encontrados. A ferramenta e os SGDBs são abertos a comunidade cientifica (i). O código é gerado a partir de um dicionário de dados (ii). A ferramenta gera código para softwares *desktop* (iii).

7. Conclusão

Este trabalho teve como resultado o desenvolvimento de uma ferramenta para geração automática de código a partir de um modelo de dados, chamada GEA. Tal resultado foi alcançado com êxito, de modo que a ferramenta construída atendeu adequadamente as expectativas.

Um ponto a se destacar do trabalho é a integração da ferramenta com o IDE Eclipse, um dos mais populares ambientes de desenvolvimento da atualidade, possibilitando a geração de código inserida no mesmo ambiente de trabalho, sem a necessidade de utilizar programas externos. A facilidade na adequação de outros

SGBDS por colaboradores, devido à estrutura na qual a ferramenta foi codificada é outra característica que deve ser enfatizada como um ponto positivo.

Através do estudo de caso realizado e a utilização da métrica de Análise de Pontos de Função foi possível verificar as vantagens e desvantagens da utilização da ferramenta.

A ferramenta possui algumas limitações, dentre elas, não possuir suporte a chaves estrangeira e tratamento de dados. Como trabalhos futuros pretendem-se estendê-la para suporte as limitações apresentadas, além de suporte a outros SGBDs. Além disso, realizar novos estudos de caso com sistemas de diferentes domínios de aplicação.

Referências

Adamatti, Marcelo, (2006) FUMIGANT: Gerador De Código Java A Partir De Base De Dados, Faculdade Cenecista Nossa Senhora Dos Anjos, Gravataí.

Cardoso, Luis, (2009) Desenvolvimento De Um Plug-In Para Extração De Estruturas De Código Fonte - Regras de Negócio. UFBA.

Eclipse, (2010) Disponível em <http://www.eclipse.org/>. Acesso em: 20 outubro de 2010.

Egen, (2010) E-GEN Developer . Disponível em <http:// www.egen.com.br />. Acesso em: 11 dezembro de 2010.

Erich Gamma, Richard Helm , Ralph Johnson , John Vlissides, (1995) Design patterns: elements of reusable object-oriented software, Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Izcode, (2010) IZCode - Gerador de Programas .Net e Java. Disponível em <http://www.izcode.com/>. Acesso em: 11 dezembro de 2010.

MySQL, (2011) White Paper, MySQL Enterprise Edition: Database. Management. Support., < http://www.mysql.com/why-mysql/white-papers/mysql_wp_enterprise_ready.php>. Acesso em: 21 de maio de 2011.

PostgreSQL, (2011) PostgreSQL Global Development Group, Disponível em <http://www.postgresql.org/>. Acesso em: 21 de maio de 2011.RUP, (2011) White Paper, Rational Unified Process, Best practices for Software Development Teams, Disponível em < http://www.ibm.com/developerworks/rational/library/253.html>. Acesso em: 21 de maio de 2011.

Santos, Anderson B. (2005), Desenvolvimento De Ferramenta E De Processos Para A Geração Automática De Código Java A Partir De Um Dicionário De Dados, Universidade Do Vale Do Rio Dos Sinos Ciência Da Computação, São Leopoldo.

Sommerville, I. (2007) Engenharia de Software, 8a. ed, p.18, p.281, São Paulo: Pearson Addison Wesley.

Silva, C. C.; Sousa K. F.; Dantas, S. D., (2006) METODES - Metodologia De Desenvolvimento De Software. Faculdade Cenecista de Brasília.

Softex, (2011) Associação Para Promoção Da Excelência Do Software Brasileiro – SOFTEX. MPS.BR . Disponível em <www.softex.br> Acesso em 14 de janeiro 2011.

Vazquez, C. E.; Simões, G. S; Albert, R. M., (2005) Análise De Pontos De Função – Medição, Estimativas E Gerenciamento De Projetos De Software. Editora Érica, São Paulo, 3.ed..

Vogella, (2010) Eclipse JDT - Abstract Syntax Tree (AST) and the Java Model – Tutorial. Disponível em: <http://www.vogella.de/articles/ EclipseJDT/article.html#jdt_javamodel> Acesso em: 10 de outubro 2010.