Comparativo do Uso de Linguagens de Programação e Geradores de Código no Desenvolvimento de Sistemas

Jean Felipe Diel[1], Luiz Cláudio Dalmolin[2], Nilson Ribeiro Modro[3]

[1]Universidade do Estado de Santa Catarina (UDESC)
Centro de Educação Superior do Alto Vale do Itajaí (CEAVI)

[2]Universidade do Estado de Santa Catarina (UDESC)
Centro de Educação do Planalto Norte (CEPLAN)

[3]Universidade do Estado de Santa Catarina (UDESC)
Centro de Educação do Planalto Norte (CEPLAN)

jfdiel@gmail.com, luiz.dalmolin@udesc.br, nilson.modro@udesc.br

Abstract. *Choose the best platform or technology for use in developing a system requires a number of factors are analyzed. A wrong choice can compromise the delivery and future maintenance of the final product. No documents or templates to follow and help when choosing which adopt technology to build a system. Thus, this article aims to highlight advantages and disadvantages as well as positives and negatives of using a code generator or a programming language in the software development stage. For comparison, PHP and Java programming languages were adopted, those used in the Upper Itajaí Valley region. The used generators, GeneXus and Scriptcase were chosen because they generate their code in Java and PHP, respectively, and for use by companies in the region.*

Resumo. Escolher a melhor plataforma ou tecnologia para se utilizar no desenvolvimento de um sistema requer que uma série de fatores sejam analisados. Uma escolha errada pode comprometer a entrega e futuras manutenções no produto final. Não há documentos ou modelos para seguir e ajudar no momento de escolher qual tecnologia adotar para construir um sistema. Logo, este artigo tem como objetivo destacar vantagens e desvantagens bem como pontos positivos e negativos de se utilizar um gerador de código ou uma linguagem de programação na etapa de desenvolvimento de *software*. Para o comparativo, foram adotados as linguagens de programação PHP e Java, estas utilizadas na região do Alto Vale do Itajaí. Os geradores utilizados, GeneXus e Scriptcase, foram escolhidos por gerarem seus códigos em Java e PHP respectivamente e por serem utilizados por empresas da região.

1. Introdução

Com o passar dos anos é possível observar que o mercado da informática vem se tornando cada vez mais competitivo e exigente. Os usuários e clientes buscam sistemas mais dinâmicos e flexíveis, mas que também sejam desenvolvidos em curto prazo e principalmente, que necessitem de pouco investimento.

Na hora de desenvolver uma aplicação, vários fatores devem ser levados em consideração pela equipe. Ao levantar os requisitos, tanto cliente, quanto membros do projeto, imaginam como deverá ficar o sistema ao final do processo de construção.

Porém, se o desenvolvimento falhar ou construir algo distinto do planejado, pode comprometer todo o projeto e relação junto ao cliente [O'keeffe 2012].

Escolher a ferramenta e linguagem a ser adotada, se esta ainda não estiver definida, deve ser considerada como uma etapa importante dentro do processo de desenvolvimento. Errar nessa escolha, pode acarretar não só em problemas durante a construção da aplicação mas em futuras melhorias e manutenções. Todos os envolvidos na etapa de desenvolvimento tem de conhecer ou ter o mínimo de conhecimento da ferramenta escolhida [Varejão 2004].

Embora leva-se quase sempre em consideração o conhecimento dos membros do projeto, é preciso que a equipe procure em outros ambientes possíveis problemas que possam vir a ocorrer, tais como: falta de mão de obra, carência de suporte da linguagem ou da ferramenta e custos elevados.

Este trabalho tem por objetivo apontar aspectos positivos e negativos, bem como vantagens e desvantagens entre o uso de ferramentas geradoras de código fonte e linguagens de programação no ambiente de desenvolvimento. Foram adotadas as linguagens de programação Java e PHP, levando-se em consideração que ambas são ensinadas e utilizadas na região do Alto Vale do Itajaí, onde encontra-se o campus da Udesc. Para as ferramentas geradoras de código, optou-se para comparação, as ferramentas GeneXus e Scriptcase, também utilizadas na região e que geram seus códigos em Java e PHP respectivamente.

2. Trabalhos correlatos

A busca por trabalhos correlatos concentrou-se na pesquisa de artigos científicos que abordassem comparativos entre linguagens de programação e ferramentas geradoras de código fonte, bem como o uso de métricas e critérios para avaliação de cada um dos artefatos. Durante a pesquisa, não foram encontrados artigos relacionados a geradores de código e sua relação com linguagens de programação ou seu uso no ambiente de desenvolvimento. Ao todo foram abordados dois trabalhos correlatos que estão listados a seguir.

O primeiro trabalho correlato intitulado: "*Propriedades Desejáveis a uma Linguagem de Programação: Uma Análise Comparativa entre as Linguagens C, C++ e Java*", [VIRTUOSO et. al. 2005], é um comparativo entre três linguagens de programação baseados em critérios preestabelecidos, apontando vantagens e desvantagens e os aspectos negativos e positivos considerando cada um dos critérios.

Já o segundo trabalho correlato "*Comparative Studies of 10 Programming Languages within 10 Diverse Criteria*", [Naim; et. al. 2010] trata-se de um comparativo entre dez linguagens de programação definidas, seguindo critérios estabelecidos pelos autores. As linguagens foram elencadas conforme o número de critérios atendidos descritos em uma tabela.

Os trabalhos correlatos utilizados limitam-se na comparação de sintaxes, pontos fortes e falhas das linguagens de programação elencando as melhores baseadas nos critérios definidos pelos autores e suas fontes. No primeiro trabalho os critérios utilizados e os resultados são apresentados de forma textual. Já no segundo trabalho, são usados cálculos matemáticos para avaliar o desempenho das linguagens e com base nos resultados destes cálculos, os autores elencaram as linguagens. Nenhum dos trabalhos citados, apresentou resultados levando em consideração um ambiente de

desenvolvimento, levando como métricas para avaliação, parâmetros como custos, mão de obra e tempo.

Com a contextualização dos trabalhos correlatos, este trabalho tem como proposta e diferencial comparar e discutir o uso de linguagens de programação e geradores de código no desenvolvimento de *software,* descrevendo aspectos positivos e negativos de cada uma das tecnologias.

3. Linguagens de Programação

Uma linguagem de programação pode ser definida segundo [Tucker 2007], como uma forma de comunicação de ideias entre humanos e computadores, porém com um domínio de expressão mais reduzido do que as linguagens naturais. Assim, a linguagem de programação traduz as ideias humanas em comandos compreendidos pelos computadores.

Muitas linguagens de programação foram criadas ao longo dos anos. Para [Aguillar 2011], as linguagens de programação servem de base para a escrita de algoritmos, cujo objetivo é solucionar problemas por meios computacionais. Essa escrita por meio de uma linguagem, quase sempre baseada em palavras no idioma inglês, facilita a comunicação humano-computador.

4. Geradores de Código Fonte

Um gerador de código pode ser definido como uma ferramenta que ao receber uma entrada de dados de forma estruturada, retorna como saída, o código fonte em uma linguagem definida, dispensando trabalho manual de um programador. Um gerador de código pode ser traduzido como um modelo a ser seguido pelos programadores, que uma vez escrito, irá gerar o código fonte sempre da mesma forma [Hunt; Thomas 1999].

Para [Martins 2007], gerar o código de uma aplicação manualmente, pode acarretar em diversos problemas para a equipe de desenvolvimento. Quando escrita sem um gerador, o código pode ter sua qualidade aumentada pelo programador a cada manutenção, porém, com o uso de um gerador, o código é gerado de forma automática e a qualidade será consistente.

Os maiores benefícios do uso de geradores de código são a qualidade: todo o código gerado seguirá um *template* bem escrito e testado; consistência: há um padrão entre as classes, já que se segue o mesmo *template* e estes por sua vez, recebem todas as decisões de arquitetura; único ponto de conhecimento: para mudanças na aplicação, basta alterar o *template* ou elementos de entrada do gerador; e por fim, foco no *design*: todo código repetido é gerado de forma automática, o programador tem de focar apenas no conhecimento e análise [Martins 2007].

4.1 GeneXus

Criada pela empresa uruguaia Artech, o GeneXus, lançado no ano de 1988, é uma ferramenta direcionada à criação de aplicações Web. Desenvolvida para o sistema operacional Windows e permite a geração de código para as linguagens C#, Java e Ruby [Genexus 2016].

Segundo [Genexus 2016], a partir da modelagem do sistema desejado, o GeneXus cria automaticamente o banco de dados, o código fonte do aplicativo, a

interface do usuário para o cliente e os serviços necessários para o servidor. A ferramenta permite ainda gerar e utilizar *Web Services*, gerar documentos, requisições HTTP e funções para *e-mail* (SMTP, IMAP, POP3, entre outros) sem exigir muito conhecimento e esforço por parte do desenvolvedor.

A figura 1 ilustra a criação de um relatório utilizando GeneXus. É possível observar que a ferramenta disponibiliza caixas de ferramentas e menus que auxiliam ao programador na criação de aplicações na forma visual sem que precise ter contato com o código fonte. A ferramenta abstrai toda a comunicação com o banco de dados, propriedades e sintaxe gerada, requisitando apenas ao usuário entradas visuais.

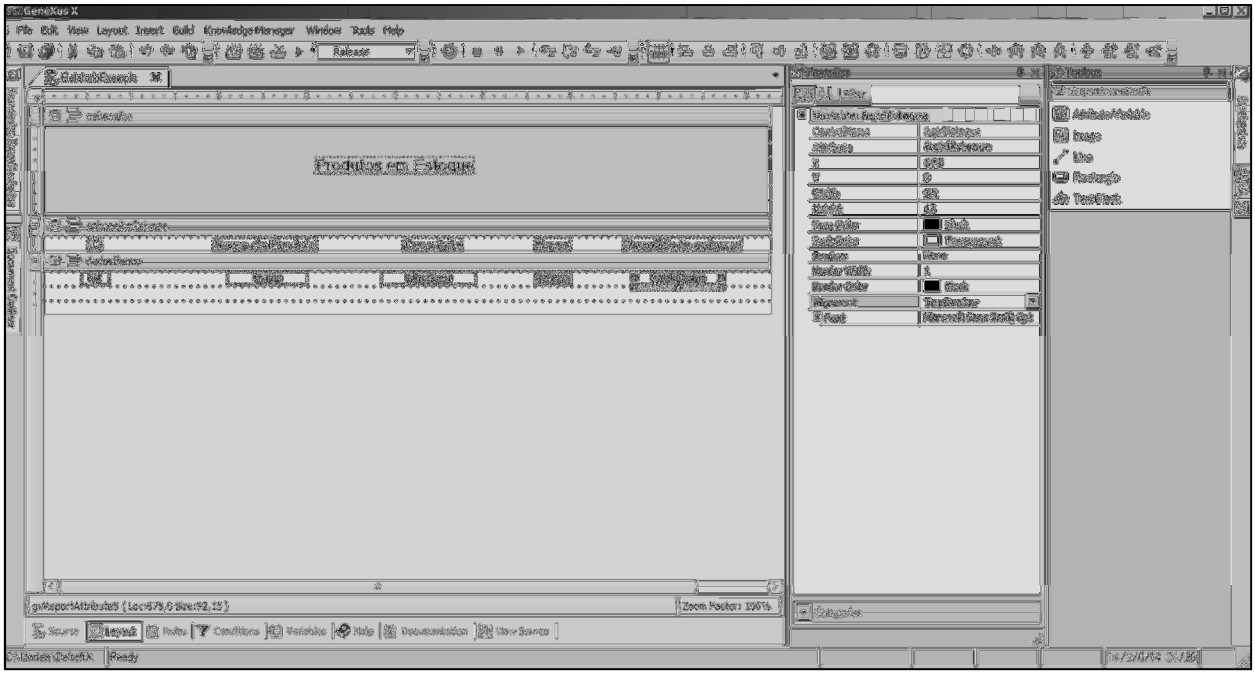

Figura 1. Criação de um relatório na ferramenta GeneXus.

4.2 Scriptcase

O Scriptcase é um gerador de código PHP, criado em 2000, pela empresa NetMake. Atualmente na versão 8.1, é uma ferramenta multiplataforma para a criação de aplicações Web, que se ajustam a qualquer dispositivo móvel [Scriptcase 2016].

Segundo [Scriptcase 2016], o gerador Scriptcase, pode ter múltiplas conexões com diferentes bancos de dados disponíveis no mercado. Possibilita a importação e exportação de documentos, disponibiliza ferramentas e gráficos para análise de dados, temas para as aplicações, filtros dinâmicos e outras inúmeras funcionalidades.

Na figura 2, pode-se visualizar uma das telas de execução do Scriptcase. Nesta tela de exemplo o usuário tem a opção de escolher o banco de dados e tem como obrigação, informar os parâmetros de conexão. Todos os comandos de conexão e iteração da aplicação do usuário, será posteriormente gerada pelo Scriptcase.

Assim como na tela da figura 2, o Scriptcase procura utilizar em todas as *interfaces,* menus e barras de ferramentas com imagens e textos, que permitam a qualquer usuário a utilização desta ferramenta, mesmo que este usuário tenha pouco conhecimento em programação de sistemas.

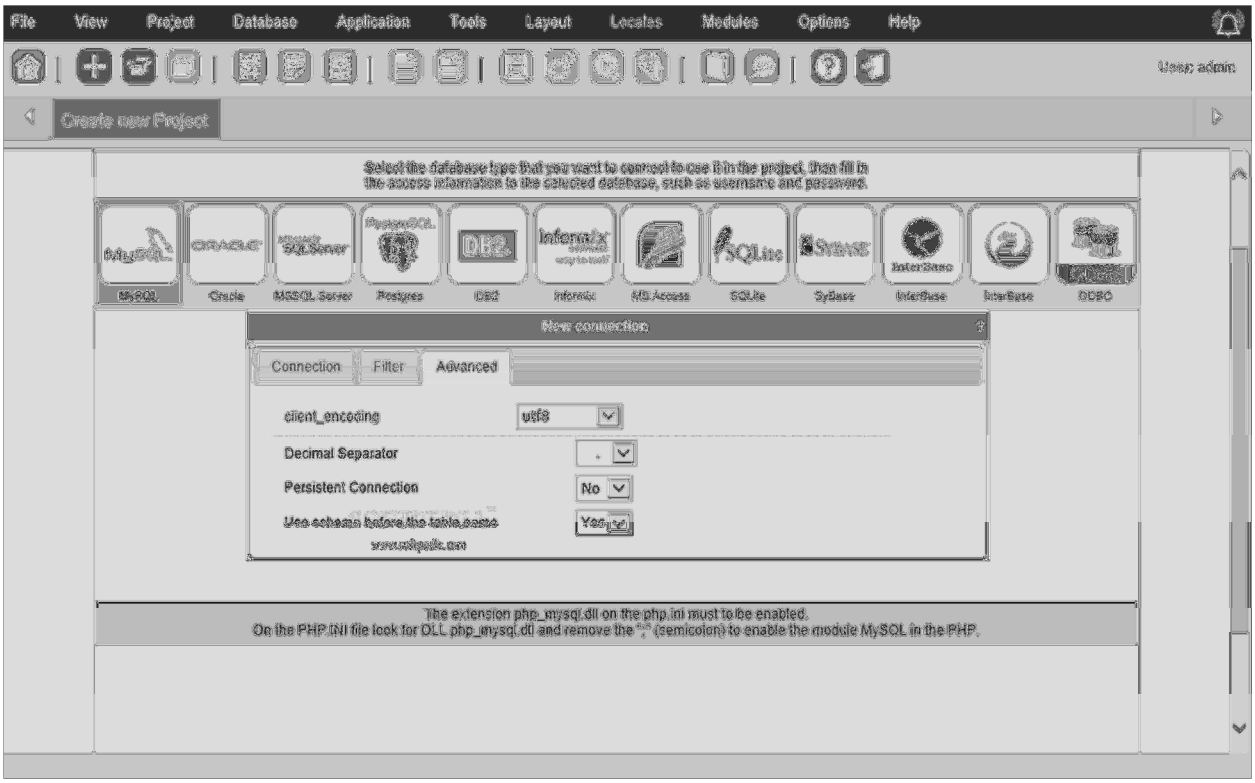

Figura 2. Tela execução do Scriptcase.

5. Metodologia de comparação

Ao iniciar um novo artefato de software, a equipe de desenvolvimento precisa planejar como conduzirá seu projeto e como o fará. Esse planejamento ocorre principalmente após a etapa de levantamento das regras de negócio e os requisitos funcionais e não funcionais. É com base neles que serão definidos a linguagem a adotar, o editor ou ambiente de desenvolvimento integrado a utilizar, os recursos e as funcionalidades [Gerhardt 2003].

O planejamento para escolher uma nova linguagem ou ambiente, pode ocorrer por diversas razões. De acordo com [Rezende 2005], todo sistema sofrerá mudanças, sejam elas ajustes após sua implantação ou por substanciais melhorias. Porém, [Rezende 2015] afirma que a medida que o sistema passa a sofrer diversas manutenções, o mesmo passa a ficar obsoleto, causando um desgaste do produto para com o usuário final e que a melhor alternativa é construir um novo sistema. Outros fatores podem influenciar a troca de tecnologias ou reconstrução, são mudanças na legislação, incompatibilidade com novos sistemas operacionais, diferentes dispositivos e hardware, documentação ou trocas de membros da equipe.

A exigência para alterar o produto faz com que a equipe precise estudar qual tecnologia irá adotar. Não por acaso, existem inúmeras situações no mercado de informática em que erros nas estimativas levam a erros na condução do projeto, causando a perda de clientes e de dinheiro [O'keeffe, 2012]. Por outra vez, muitos processos de migração de linguagem ou de versão de tecnologias chegam ao seu final, mas levam um tempo muito além do esperado.

Muitos fatores devem ser levados em consideração na hora de migrar para uma nova versão do software ou alterar a tecnologia usada. Os profissionais envolvidos no projeto, devem buscar informações além das dependências da empresa. Buscas na Web, entrevistas com outros profissionais que passaram por migrações, construção de protótipos, são técnicas que podem auxiliar para evitar possíveis problemas [Gerhardt 2003].

Quando a necessidade se dá em relação à linguagem de programação ou gerador de código a serem adotados, a experiência da equipe é sempre determinante. Ao surgir uma nova versão da linguagem ou do gerador de código em que a equipe tenha domínio, os membros precisam apenas se atualizar em relação às novas funcionalidades e correções adotadas. Entretanto, ao se adotar uma tecnologia ainda desconhecida, exige-se que todos aprendam a usá-la, o que toma tempo e exige planejamento para não prejudicar o andamento do projeto [Gerhardt 2003].

Independente do domínio da tecnologia utilizada pela empresa, os colaboradores tendem a mudar ao decorrer do processo de desenvolvimento. Principalmente na área de informática, onde a troca de membros é comum devido à falta de profissionais qualificados no mercado. A dificuldade em encontrar novos colaboradores aumenta quando a tecnologia usada pela empresa não é a adotada em escolas e universidades e, principalmente, quando há custo significativo para treinamentos no seu uso [Paquet; Mokhov 2010].

Diferentemente das linguagens de programação, uma dificuldade encontrada no uso dos geradores de código é a documentação disponível para auxílio de seus usuários. Assim a documentação fica restrita ao da empresa criadora e de alguns sites de usuários com maior conhecimento. Já as linguagens, dispõem de inúmeros sites oficiais ou não, *blogs*, fóruns e milhares de livros.

Em sua maioria, um gerador de código trata-se de um produto vendido por uma empresa. Assim qualquer problema ou dificuldade em seu uso, o usuário pode contar com o apoio da empresa para auxiliá-lo. O mesmo não ocorre em relação às linguagens de programação, onde há a disponibilidade de documentações, sites oficiais das linguagens e relatos de outros programadores com domínio da linguagem, porém, os membros do projeto precisam dominá-la ou pagar consultorias para alguém que domine o conhecimento sobre essas.

Ao se utilizar uma linguagem de programação para o desenvolvimento de um sistema, precisa-se optar por duas opções, implementar todo o código de forma manual ou utilizar um *framework.* Existe para cada linguagem, diversos *frameworks.* Os *frameworks* disponibilizam diversas funcionalidades implementadas, que podem se necessário serem utilizadas, como interação com banco de dados, serviços de e-mail, *web services*, entre outros, porém exigem domínio da linguagem e conhecimento de alguns paradigmas de programação [Alomari et. al. 2015].

O uso da linguagem de programação seja ela escrita manualmente ou optando por um framework, faz com que as classes sejam criadas pelos membros do projeto, ou seja, ao menos sabe-se como foi escrito e como manter a aplicação. No caso dos geradores de código, o código fonte gerado seguirá um *template* e manutenções só poderão ser feitas utilizando-se o próprio gerador. Sendo assim, o desenvolvedor fica refém das limitações da ferramenta, de forma que qualquer situação além do código gerado que venha a ser solicitada e que não possa ser uma entrada no gerador, ficará pendente por parte da equipe.

Na figura 3, pode-se visualizar o trecho de um código de uma aplicação criada através do gerador de código Scriptcase. O exemplo de aplicação, tem como resultado esperado apresentar uma frase na tela. Mesmo se tratando de uma única linha, o exemplo não deixa de ser uma aplicação, sendo assim, o Scriptcase gera toda a estrutura necessária para caso a aplicação seja continuada.

Se o mesmo código fosse escrito com a linguagem PHP, a frase poderia ser escrita utilizando-se menos de cinco linhas e utilizando apenas um arquivo. O trecho exibido na figura 3, é parte de um arquivo e este por sua vez, faz parte do conjunto de dez arquivos gerados para exibição da frase de exemplo.

```
$_SESSION['scriptcase']['charset_entities']['ISO-8859-1'] = 'ISO-8859-1';
$_SESSION['scriptcase']['charset_entities']['ISO-8859-5'] = 'ISO-8859-5';
$_SESSION['scriptcase']['charset_entities']['ISO-8859-15'] = 'ISO-8859-15';
$_SESSION['scriptcase']['charset_entities']['WINDOWS-1251'] = 'cp1251';
$_SESSION['scriptcase']['charset_entities']['WINDOWS-1252'] = 'cp1252';
$_SESSION['scriptcase']['charset_entities']['BIG-5'] = 'BIG5';
$_SESSION['scriptcase']['charset_entities']['EUC-CN'] = 'GB2312';
$_SESSION['scriptcase']['charset_entities']['GB2312'] = 'GB2312';
$_SESSION['scriptcase']['charset_entities']['SJIS'] = 'Shift_JIS';
$_SESSION['scriptcase']['charset_entities']['EUC-JP'] = 'EUC-JP';
$_SESSION['scriptcase']['charset_entities']['KOI8-R'] = 'KOI8-R';
$_SESSION['scriptcase']['trial_version'] = 'N';
$_SESSION['sc_session'][$this->sc_page]['Exemplo']['decimal_db'] = ".";

$this->nm_cod_apl      = "Exemplo";
$this->nm_nome_apl     = "";
$this->nm_seguranca    = "";
$this->nm_grupo        = "exemplo";
$this->nm_grupo_versao = "1";
$this->nm_autor        = "admin";
$this->nm_versao_sc    = "v8";
$this->nm_tp_lic_sc    = "demo";
$this->nm_dt_criacao   = "20160531";
$this->nm_hr_criacao   = "234303";
$this->nm_autor_alt    = "admin";
$this->nm_dt_ult_alt   = "20160531";
$this->nm_hr_ult_alt   = "234604";
$temp_bug_list         = explode(" ", microtime());
list($NM_usec, $NM_sec) = $temp_bug_list;
$this->nm_timestamp    = (float) $NM_sec;
$this->nm_app_version  = "1.0.0";

$NM_dir_atual = getcwd();
if (empty($NM_dir_atual))
{
    $str_path_sys          = (isset($_SERVER['SCRIPT_FILENAME'])) ? $_SERVER['SCRIPT_FILENAME'] : $_SERVER['ORIG_PATH_TRANSLATED'];
    $str_path_sys          = str_replace("\\", '/', $str_path_sys);
}
else
{
```

Figura 3. Exemplo de código gerado pelo Scriptcase.

Ao optar por um gerador de código, as empresas e seus colaboradores podem acabar ficando, ao decorrer dos anos, reféns destas ferramentas automáticas. Tomando como exemplo o Scriptcase, o desenvolvedor pode criar programas e mantê-los por anos sem ver uma linha de código durante este tempo. Ao receber um desafio em uma linguagem ou projeto que não exija o uso de um gerador, o mesmo pode encontrar dificuldades.

O uso de uma ferramenta para gerar código, em ambos os casos GeneXus e Scriptcase, ao decorrer do desenvolvimento de aplicações mais robustas, faz com que a geração venha a ficar mais lenta, uma vez que ao implementar uma funcionalidade, ambos geram vários arquivos e assim exigem que o usuário tenha um computador com mais recursos.

Para [Varejão 2004] e [Sebesta 2011], existe certa dificuldade em se estabelecer critérios para avaliação de uma linguagem e geradores de código, pois muitas vezes

esses critérios são subjetivos e até mesmo pessoais. Os critérios podem estar em diferentes níveis de granularidade, podendo ser considerados mais ou menos importantes de acordo com a perspectiva de quem os analisa.

Com base nos critérios de [Varejão 2004] e [Sebesta 2011], foram elaborados aspectos para o comparativo entre o uso de ferramentas geradoras de código (GeneXus e Scriptcase) e linguagens de programação (PHP e Java), comparando as principais características de cada critério e tecnologia na construção de sistemas. Tal comparativo é demonstrado na tabela 1.

Tabela 1. Comparativo entre linguagens de programação e geradores de código.

Critérios	Linguagem de Programação (PHP e Java)	Geradores de código (GeneXus e Scriptcase)
Custo	Não possuem custos. Treinamentos e suportes de terceiros tem custos e qualidade variáveis.	Possuem custo de licença. O treinamento e o suporte estão inclusos na aquisição.
Facilidade de aprendizado	Exige treinamento e prática.	Exige prática, mas em grande maioria, são intuitivos e de fácil aprendizagem.
Fácil manutenção do código	Todo o código é escrito pela equipe.	A ferramenta gera o código na linguagem definida, exige alto conhecimento para manutenções no código gerado pela ferramenta.
Integração com banco de dados	Se não utilizar frameworks, necessita ser implementada toda a estrutura, conexão e comandos SQL.	A ferramenta cria o banco, tabelas, conexões e comandos SQL automaticamente.
Documentação	Dispõe de livros, inúmeros sites, milhares de usuários e páginas oficiais.	Em sua maioria, é fornecida pelo fabricante.
Suporte	Apenas em livros, sites oficiais, sites da Web e programadores com conhecimento.	A fabricante dispõe de canais para auxiliar seus clientes.
Mão de obra	Profissionais são ensinados em cursos, universidades, livros ou sites da Web.	Mão de obra escassa. Profissionais precisam ser treinados.
Portabilidade	Rodam em qualquer plataforma.	O GeneXus está disponível apenas para o sistema operacional Windows. O Scriptcase possui versões para todos os sistemas operacionais disponíveis.

6. Análise dos resultados

As linguagens de programação destacam-se como vantagens por sua portabilidade, dando ao programador a liberdade de trabalhar com qualquer plataforma do mercado. Destaca-se também que as linguagens não possuem custo e dispõe de vários profissionais e treinamentos disponíveis.

Todo o código escrito da aplicação é feita por membros da equipe, o que facilita futuras manutenções. Entretanto, como ponto negativo, os programadores precisam ter conhecimento da linguagem, buscar ajudas em meios alternativos e em grande maioria, desenvolver todas as funcionalidades do sistema.

Os geradores de código tem como desvantagens seu custo, a falta de mão de obra e dificuldade em manter o código gerado pela própria ferramenta. Porém, essas desvantagens são supridas pela facilidade em se aprender a utilizar a ferramenta.

Os geradores tendem a gerar todos os comandos de iteração com o banco de dados e outros serviços, assim ganha-se tempo no desenvolvimento do sistema. Quando necessário, a equipe tem a sua disposição o suporte necessário para resolução de problemas ou dúvidas.

7. Considerações Finais

Cada empresa tem suas regras e padrões de trabalho, onde seus profissionais tendem a direcionar o desenvolvimento para tecnologias que tenham conhecimento e domínio. Uma mudança ou adoção de tecnologia diferente do habitual, em grande maioria dos casos, desagrada os profissionais. Porém, apontar qual é a melhor opção adotar não é uma tarefa simples, sendo ideal toda a equipe ser envolvida no processo de decisão.

Ao comparar geradores de código com linguagens de programação não é possível apontar de maneira superficial qual é a melhor opção a ser escolhida. Pode-se citar que um gerador tem um custo elevado, mas este é absorvido quando o ponto em questão é a oferta de suporte e treinamento, inclusos na adoção da ferramenta. O mesmo não ocorre com as linguagens, estas em sua maioria, sem custos de aquisição, mas que não dispõem de uma empresa de suporte e treinamento à disposição.

Possivelmente a forma mais correta de definir a melhor tecnologia, seria colocar a mesma equipe para desenvolver um software utilizando uma linguagem de programação e um gerador de código, de forma comparativa, utilizando a mesma linguagem de programação. Poderiam ser definidas métricas para mensurar tempo e qualidade, mas há outros aspectos que não podem ser avaliados, como máquinas, momento emocional da equipe, entre outros.

Conclui-se que cada equipe deve definir qual tecnologia adotar segundo suas observações e experiências. Entretanto, sempre que possível, é interessante que a equipe discuta o desempenho do projeto com a ferramenta utilizada, levantando aspectos positivos e negativos, buscando sempre melhorar o andamento dos processos, seja com a própria tecnologia ou na busca de uma nova opção.

Embora o presente estudo seja um comparativo entre linguagens de programação e ferramentas geradoras de código, como limitações da pesquisa, recomenda-se cautela na generalização dos resultados pois foram utilizadas apenas duas linguagens dentre milhares existentes e apenas duas ferramentas disponíveis no mercado, pois foram

utilizados como base para o comparativo, o uso destas em empresas e universidades da região.

Como trabalhos futuros, recomenda-se o uso de métricas qualitativas e quantitativas para mensurar a criação de um artefato, tanto na linguagem quanto no gerador de código, utilizando os mesmos recursos computacionais e humanos para ambos de forma que se possa apontar qual a melhor alternativa considerados valores numéricos como resultante.

Referências

Genexus. Disponível em: <http://www.genexus.com>. Acesso em: 02 de Maio de 2016.

Gerhardt, Frank. **Integration of Programming Environments for Platform Migration.** 2003. 128 f. Dissertação (Mestrado) - Eberhard Karls Universität Tübingen, Tübingen, 2003.

Hunt, A.; Thomas, D. **The pragamatic programmer**: from journeyman to master. Estados Unidos: Artmed, 1999.

Martins, J. C. C. **Técnicas para gerenciamento de projetos de software**. Rio de Janeiro: Brasport, 2007.

Naim, R.; et.al. **Comparative Studies of 10 Programming Languages within 10 Diverse Criteria**. Artigo publicado no COMP, Montreal-QC, Canadá. Montreal: 2010. Disponível em: https://arxiv.org/pdf/1008.3561.pdf. Acesso em: 05 mai. 2016.

O'keeffe, Juan. **Análise de fatores de impacto no erro de estimativa de esforço e de duração em projetos de software.** 2012. 94 f. Dissertação (Mestrado) - Curso De Faculdade De Administração, Contabilidade e Economia Mestrado em Administração e Negócios, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2012.

Paquet, J.; Mokhov, S. A. **Comparative Studies of Programming Languages**. Course Lecture Notes, Montreal, v. 6, p. 7-61, 2010.

Rezende, D. A. **Engenharia de software e sistemas de informação**. 3. ed. Rio de Janeiro: Brasport, 2005.

Scriptcase. Disponível em: <http://www.scriptcase.com.br>. Acesso em: 02 de Maio de 2016.

Sebesta, R. W. **Conceitos de linguagens de programação**. 9. ed. Tradução de Eduardo Kessler Piveta. Porto Alegre: Bookman, 2011.

Tucker, A. B.; Noonan, R. **Programming languages**: principles and paradigms. 2.ed. Estados Unidos: McGraw-Hill: 2007.

Varejão, Flavio Miguel. **Linguagem de programação**: conceitos e técnicas. Rio de Janeiro: Elsevier: 2004.

Virtuoso, G.; Couto Junior, M.; Martins P. **Propriedades Desejáveis a uma Linguagem de Programação: Uma Análise Comparativa entre as Linguagens C, C++ e Java**. In: I Congresso Sul Catarinense de Computação, Criciúma-SC. Criciúma: 2005. Disponível em: http://periodicos.unesc.net/sulcomp/article/viewFile/796/747. Acesso em: 05 mai. 2016