JAVA ON ROAD: Um Framework para Desenvolvimento WEB

Igor M. Vanderlei
Professor da Faculdade Sete de Setembro (FASETE), Doutorando em Ciência da Computação pela Universidade Federal de Pernambuco (UFPE)
Rodrigo G. C. Rocha
Professor da Universidade Federal da Paraíba (UFPB) e da Faculdade Sete de Setembro (FASETE), Doutorando em Ciência da Computação pela Universidade Federal de Pernambuco (UFPE)
Edson A. Silva
Bacharelando do curso de Sistemas de Informação da Faculdade Sete de Setembro- FASETE

RESUMO

Com a globalização, o desenvolvimento de Software evoluiu muito nas últimas décadas. Dessa forma, as empresas necessitam desenvolver produtos e serviços com maior qualidade e menor prazo para se tornarem mais competitivas no mercado. Com isso, as organizações que desenvolvem software necessitam de meios capazes de promover maior rapidez no desenvolvimento de software, visando diminuir esforço e aumentar a padronização dos seus projetos. Este artigo apresenta o Java on Road, um framework que tem por objetivo o desenvolver de forma rápida aplicações para plataforma WEB na linguagem Java, que utiliza uma arquitetura pré-estabelecida e entidades de negócio da aplicação como ponto de partida. O artigo também descreve a utilização da ferramenta com a técnica prototipação. Além do desenvolvimento rápido, as aplicações geradas pelo Java on Road possuem características de qualidade de software como a padronização do código fonte e a reutilização de componentes.

Palavras-Chave: Java, Framework, Desenvolvimento Web, Desenvolvimento rápido de aplicações.

ABSTRACT

Globalization has increased software development for the past decades. This fact brings to the companies the need of having a better quality in their service as well as in their products, in order to become more competitive into the business market. With this, the software development organization needs faster development in software in order to decrease stress and increase the standardization of their projects. This article presents JAVA Road, which is a framework that aims to develop in a fast way the application on the Web platform into JAVA language, which uses the pre-established architecture and application business entities as the start point. This article also describes the use of tools with the prototyping technique. Besides the fast development of applications done by JAVA Road have quality characteristics of software as source code standardization and reuse of components.

Key-words : JAVA, Framework, Web development, fast application development

INTRODUÇÃO

Nos últimos anos, com o crescimento da área de desenvolvimento de software, o mercado se tornou muito competitivo, exigindo das empresas maior qualidade e menor prazo para a realização de seus projetos. Segundo Crnkovic (2003), os desenvolvedores de software procuram maneiras para diminuir o esforço e aumentar a padronização dos seus projetos.

Em função do deferido problema, algumas empresas vêm buscando a utilização de meios capazes de promover maior rapidez no desenvolvimento de software. Para isso, a engenharia de software, compreende alguns paradigmas, envolvendo métodos, ferramentas e alguns procedimentos. O objetivo deste trabalho é apresentar o Java on Road (JoR), um *framework* capaz de desenvolver rapidamente aplicações WEB na linguagem Java, conservando características da qualidade de software, como a padronização do código e da reutilização de componentes, além de servir como ferramenta de prototipação, utilizando-se dos benefícios inerentes a este paradigma.
O trabalho está dividido da seguinte forma: na seção 2, é apresentado o framework em si, em que seus conceitos, arquitetura, vantagens, limitações e sua utilização com prototipação são descritos; e na última seção, a 3, os trabalhos relacionados são citados e suas diferenças apresentadas, e por fim, as considerações finais a respeito do desenvolvimento de aplicações WEB com o JoR são relatadas.

1 JAVA ON ROAD

O Java On Road (JoR) é um framework que tem por objetivo o desenvolver de forma rápida uma aplicação para WEB na linguagem Java, tomando como base uma arquitetura pré-estabelecia e as entidades de negócio da aplicação.

O JoR surgiu, a partir da observação de que o código fonte de uma aplicação típica é composta de uma grande parte repetitiva, algumas vezes com pequenas modificações. Desta forma, esta parte do código fonte pode ser feito de forma automática, o que reduz a quantidade de erros inseridos pelos programadores e permite que os mesmos se concentrem em tarefas que exigem um maior esforço intelectual e que possuem um impacto maior na qualidade final do software desenvolvido, como por exemplo, na implementação das regras e restrições do negócio.

Segue na Figura 1, uma tela de listagem de usuários gerada pelo Java on Road no exemplo de uma aplicação teste. Na imagem, são listados os usuários e, para cada registro, é possível visualizar,

detalhadamente, todos os dados, como também editar e remover tais registros. Além de permitir também que novos usuários sejam adicionados.

Figura 1 – Tela de listagem de Usuários

1.1. DESENVOLVIMENTO BASEADO EM ARQUITETURAS

O desenvolvimento de qualquer aplicação no JoR é baseado em uma arquitetura, que é uma generalização para uma família de aplicativos, que podem possuir finalidades distintas, mas são construídos, a partir de componentes com características comuns. Por exemplo, um aplicativo para gerenciamento de cemitério e um aplicativo para uma agência de empregos podem possuir o componente de interface gráfica utilizando JSP[2] (Java Server Pages) e o componente de persistência de dados utilizando o JDBC[3] (Java Database Connectivity), bem como outros.

Uma arquitetura possui sua descrição, além de um conjunto de componentes e um conjunto de variáveis. Os **componentes** são os elementos centrais que compõem a arquitetura, neles, está embutido o modelo dos arquivos de código fonte (*template*) necessário para a criação do projeto. As **variáveis** armazenam as configurações necessárias para a criação de um projeto.

A criação de uma arquitetura é um processo relativamente complexo. Inicialmente, deve ser feito um pequeno projeto, seguindo o modelo que se deseja criar. Em seguida, este projeto modelo é avaliado, a fim de produzir as generalizações de código (*templates*). Por fim, os *templates* são testados utilizando as classes de entidade do projeto modelo para verificar, se o projeto gerado pelo JoR com aquela arquitetura estão de acordo com o projeto modelo. Uma vez que a arquitetura foi verificada, a mesma pode ser utilizada para a geração de vários outros aplicativos maiores.

2Tecnologia utilizada no desenvolvimento de aplicações para Web.
3Tecnologia que permite a comunicação entre as instruções de banco de dados escritas no código fonte e diversos bancos de dados.

1.2. DESENVOLVIMENTO COM O JoR

O processo de criação de um projeto utilizando o JoR é definido em seis etapas. E cada etapa é responsável por realizar um papel importante para a obtenção da aplicação. As etapas que compõem o processo de desenvolvimento são indicadas na Figura 2.

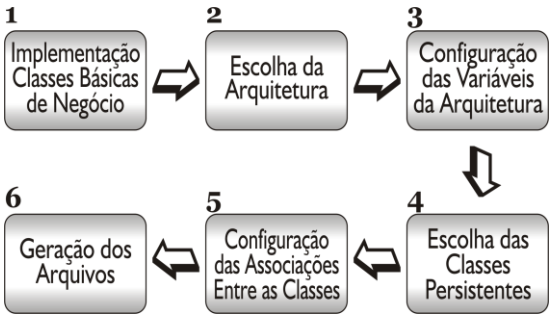

Figura 2 – Processo de desenvolvimento com JoR

Primeiramente, é necessário definir o modelo de dados da aplicação, através da obtenção dos requisitos e desenvolvimento dos diagramas, posteriormente, geram-se classes básicas Java, de acordo com o diagrama de classe (Etapa 1), seguindo o padrão POJO (*Plain Old Java Objects*).

Após a definição das classes, é necessário escolher com qual template de arquitetura será desenvolvida a aplicação (Etapa 2), nessa etapa o desenvolvedor possui três opções: usar um dos *templates* disponibilizado pela equipe do JoR, criar um novo *template* e editar um *template* existente.

Em seguida, o desenvolvedor poderá configurar as variáveis do *template* (Etapa 3), e escolher quais serão as classes que deverão ser persistentes ou que serão gerenciadas através de um DAO (Data Access Object) (Etapa 4).

Em alguns casos, os relacionamentos entre as classes podem ocorrer de forma bidirecional, como somente o desenvolvedor pode informar de que forma deve ser tratada a essa navegação, o JoR possibilita que seja possível essa configuração, através da configuração da associação entre as classes (Etapa 5). Por fim, tendo que o JoR está totalmente configurado, pode-se gerar os códigos dos arquivos que compõem a aplicação (Etapa 6)

Para facilitar o entendimento do funcionamento do *framework* é apresentado o Diagrama de Casos de Uso na Figura 3, sendo possível visualizar as principais funcionalidades realizadas na

ferramenta. Existem dois atores: o Desenvolvedor e o Arquiteto de software. O primeiro é responsável pela interação com o JoR, permitindo a escolha de um *template* que atenda às necessidades do projeto e, em seguida, criando o protótipo funcional. O arquiteto de software, diferente do Desenvolvedor, não se preocupa somente com a criação do projeto, mas sim com os requisitos da aplicação, podendo, a depender de sua necessidade, criar um novo *template*, ou editar um existente para atender as suas necessidades.

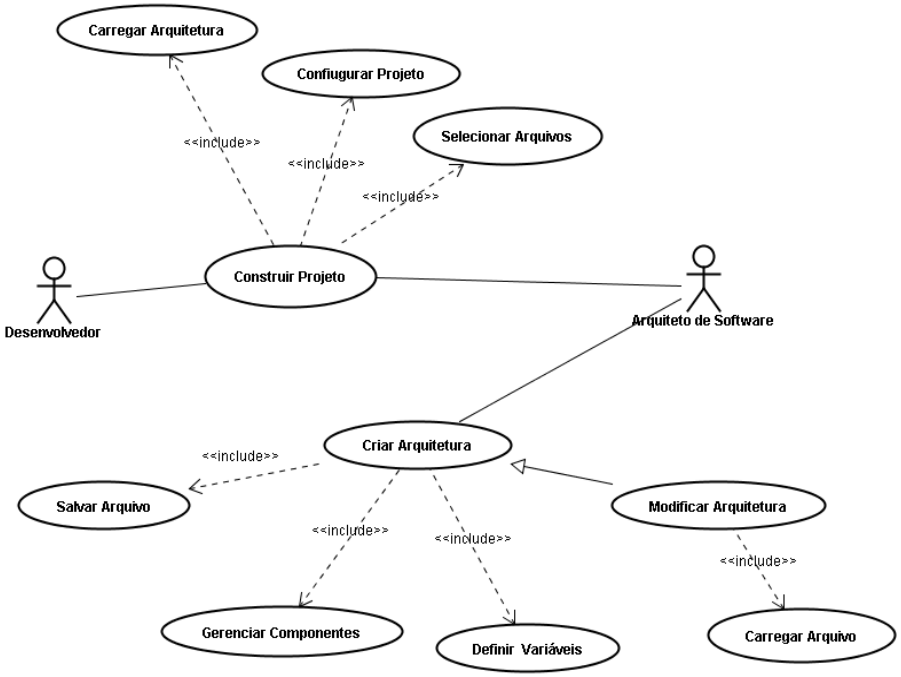

Figura 3 – Diagrama de Caso de Uso do JoR

1.3. ARQUITETURA

Os principais componentes da arquitetura do Java on Road são: *Plugin*, *Parser* e *CodeGenerator*, como observados na imagem abaixo.

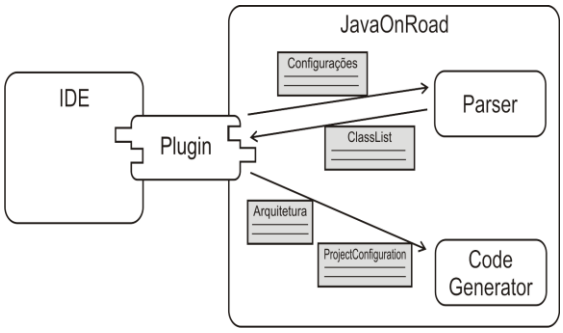

Figura 4 – Processo de desenvolvimento com JoR

O Componente *Plugin* é responsável pela integração entre o JoR, a IDE utilizada e pela interface com o usuário. O *Plugin* ao ser inicializado detecta o diretório da raiz do projeto e guarda suas configurações. Em seguida, o usuário carrega uma arquitetura e configura o pacote base do projeto. Em posse dessas configurações, o *plugin* passa essas informações ao *parser* que irá realizar uma leitura das classes para representá-las no modelo de classes do JoR. Após construir o modelo de classes, o *parser* retorna o modelo para o *plugin*, em que o usuário ajusta algumas configurações e seleciona quais componentes devem ser criados.

1.4. LIMITAÇÕES DA FERRAMENTA

O JoR não é capaz de criar o produto final da aplicação, pois as regras de negócio e tratamento de exceções não podem ser generalizados para qualquer tipo de projeto. Além disso, quando a equipe de desenvolvimento não possui um padrão arquitetural em suas aplicações não compensa investir tempo no processo custoso de criação de uma arquitetura para cada nova aplicação a ser desenvolvida.

1.5. VANTAGENS

Através do JoR é possível desenvolver aplicações, rapidamente, somente, a partir de classes básicas de negócio. Em um estudo de caso realizado por Varjão (2008), o JoR levou 53 segundos para a geração de um protótipo com as funcionalidades básicas de uma aplicação, contendo 154 itens (arquivos e diretórios), tomando como entrada apenas 20 itens construídos manualmente.

Além do rápido tempo de desenvolvimento, as aplicações geradas pelo JoR possuem características de qualidade de software como a padronização do código fonte, visto que o código é criado, a partir de um *template* e a reutilização de componentes.

Embora o desenvolvimento de uma arquitetura não seja uma tarefa fácil, a reutilização da mesma em vários projetos contribui para melhorar a qualidade da arquitetura, pois, quando um erro é descoberto em um projeto, a solução do erro deve ser incluída na arquitetura e, desta forma, os projetos gerados, a partir de arquiteturas, exaustivamente, testadas serão livres de erro.

1.6. UTILIZAÇÃO DE PROTOTIPAGEM COM JoR

Construir protótipos é uma maneira de explorar ideias de projeto de um determinado produto antes de investir tempo e recursos na sua implementação. Existem diversas abordagens para a utilização

de protótipos, como a prototipagem exploratória, a prototipagem experimental e a prototipagem evolutiva (Berkun, 2000). Segundo Aguiar et al (2003):

- **Prototipagem exploratória:** é quando vários protótipos são criados e avaliados, para que a equipe de projeto possa expressar, claramente, como o futuro software deve funcionar ou parecer, objetivando esclarecer os requisitos funcionais e os requisitos do usuário.
- **Prototipagem experimental:** avalia as várias opções de projeto sob o ponto de vista técnico de implementação. O objetivo é experimentar alternativas de projeto que utilizam diferentes tecnologias e avaliar a viabilidade de implementar o futuro sistema utilizando uma ou outra tecnologia ao analisar variáveis como desempenho e outros aspectos técnicos.
- **Prototipagem evolutiva:** baseia-se na evolução incremental do protótipo e os projetistas assumem que o protótipo está em constante evolução e, portanto, pode ser modificado de acordo com o surgimento de novos requisitos ou alterações nos requisitos já existentes (Lichter, Schineider-Hufsmidt, Züllighoven, 1993).

A utilização da prototipação pode trazer diversos benefícios para um projeto, como por exemplo, identificar requisitos incompletos e facilitar o processo de desenvolvimento, já que um protótipo pode ser usado como sistema final, entre outros. Abaixo seguem algumas vantagens de se criar protótipos com o framework Java on Road.

A utilização do JoR para a prototipação é suportada pelas abordagens exploratória, experimental e evolutiva. No caso da exploratória é possível pois o JoR possibilita a criação rápida de diversos protótipos, permitindo a exploração dos requisitos de usuários e funcionais por parte da equipe do projeto. Já na prototipação experimental, que tem o foco no ponto de vista técnico de implementação, o *framework* permite a utilização de várias arquiteturas diferentes, favorecendo a ideia de experimentar alternativas de projeto antes de definir a que será utilizada. E na abordagem evolutiva onde o objetivo central é a evolução incremental do protótipo, o JoR possibilita modificações em seus protótipos facilmente.Desta forma, o protótipo pode evoluir constantemente sem maiores problemas.

Todas as três abordagens são utilizadas através de prototipagem de alta-fidelidade, já que o Java on Road gera protótipos que, se for o caso, já podem ser usados como sistemas funcionais.

Uma das vantagens de se utilizar prototipação de baixa e média-fidelidade é o pouco tempo necessário para criar os protótipos, porém os mesmos só servirão para identificar os requisitos. Já a prototipação de alta-fidelidade é uma de suas desvantagens é que alguns protótipos podem demandar muito tempo para serem desenvolvidos. Entretanto a utilização do Java on Road possibilita que os protótipos de alta-fidelidade possam ser rapidamente construídos, servem para a fase de requisitos e se aprovados, já podem entrar em operação como sistema final.

Sommerville (2006) afirma que os protótipos que evoluem para o sistema final devem ser desenvolvidos com os mesmo padrões de qualidade organizacional de qualquer outro software. Eles devem ter sólida estrutura, de modo que possam receber manutenção por muitos anos. Eles devem ser confiáveis e eficientes e se adequar a padrões organizacionais relevantes. Dessa maneira, o JoR se apresenta como uma alternativa interessante, já que o mesmo mantém conceitos chaves para o desenvolvimento de software, como exemplo, a padronização de código e a reutilização de componentes.

Uma das possibilidades de utilizar a técnica de prototipação com o Java on Road é unir as características dos modelos exploratório e evolutivo.Inicialmente seria escolhida a arquitetura a ser utilizada, após isso, inicia-se um ciclo baseado na prototipação exploratória com as seguintes atividades: fazer a implementação das classes básicas de negócio, em sequência gerar o protótipo e logo, em seguida, fazer a validação do modelo de negócio. Este ciclo só seria finalizado se o modelo de negócio for validado. A partir desta validação, inicia-se outro ciclo, referente à prototipação evolutiva, teriam duas atividades: refinar o protótipo e realizar os testes unitários; estas atividades se repetiriam até que as mesmas fossem validades. A Figura 5 representa melhor essa ideia de unir as características dos dois modelos de prototipação.

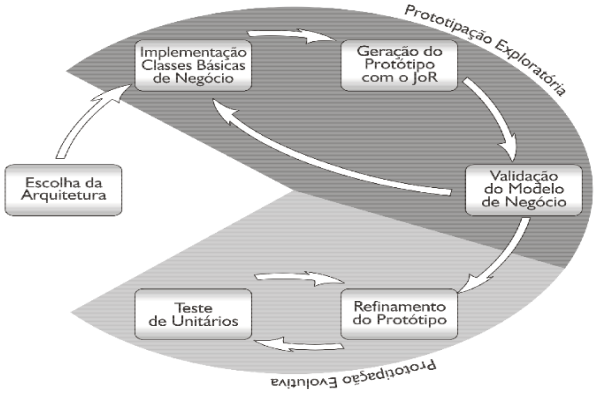

Figura 5 – Prototipação evolucionária e evolutiva com JoR

2 TRABALHOS RELACIONADOS

Existem diversas ferramentas e metodologias de prototipação disponíveis. Ozaki et al (2003) apresentaram uma técnica para desenvolvimento de protótipos evolutivos na linguagem Java, baseado em objetos *proxy* que possibilita a execução de teste do software, parcialmente, construído. Apesar da técnica apresentada oferecer uma solução para desenvolver protótipos em etapas intermediárias do processo de desenvolvimento, a metodologia apresentada não é amparada por uma ferramenta de automação de desenvolvimento.

O Ruby on Rails é um *framework* para desenvolvimento de aplicativos para internet, com a linguagem Ruby, que vem se destacando por sua elevada produtividade (HIBBS e TATE, 2006). O desenvolvimento de um aplicativo no Ruby on Rails inicia com a criação do modelo de dados. Em seguida, através da execução de alguns comandos, o programa é gerado, contendo, entre outras funcionalidades, código de acesso a banco de dados e validação dos formulários. Devido à grande produtividade, o Ruby on Rails tem o potencial de ser utilizado para criação de protótipos. Entre as principais críticas relacionadas ao framework Ruby on Rails está o fato que o mesmo se baseia no princípio de "convenção sobre configuração", o que o torna uma ferramenta limitada, no sentido que retira a liberdade do desenvolvedor em escolher a arquitetura do sistema.

IBM Model Transfer *Framework* (MTF) (MTF, 2009) e Xdoclet (XDOCLET, 2009) são duas tecnologias que permitem criar rapidamente partes significativas do código fonte automaticamente. O XDoclet através da utilização de meta-informação colocada nos comentários javadoc (padrão para comentar código) para poder gerar automaticamente classes JAVA, arquivos XML, entre outros. Enquanto o MTF através da modelagem UML, a partir da criação do diagrama de classes do sistema são geradas as classes java referentes ao diagrama. Novamente, essas duas ferramentas citadas não permitem a liberdade ao engenheiro de software no sentido de escolher e/ou conceber a arquitetura do sistema.

CONSIDERAÇÕES FINAIS

Com a evolução do mercado de desenvolvimento de software, torna-se visível a necessidade que as empresas tem de se aperfeiçoar, usando técnicas, ferramentas ou metodologias que possam contribuir, para que o produto seja entregue com qualidade, dentro do prazo e custo previsto.

Como foi descrito, o Java on Road permite a criação rápida de um sistema na plataforma WEB, sendo apenas necessária a escolha da arquitetura e a concepção das classes básicas. Através do JoR,

também, é possível desenvolver rapidamente protótipos de alta-fidelidade, nas três abordagens (exploratória, experimental e evolutiva), que serão utilizados como parte do sistema assim que os mesmos forem validados.

Como trabalhos futuros, seria interessante: estudar a integração da ferramenta JoR com processos de software, como RUP e XP; Realizar estudos de caso da ferramenta com equipes maiores e projetos maiores, a fim de verificar seu desempenho e possa permitir que o JoR gere os protótipos através de diagrama de classes, diferente do que acontece hoje, já que a geração de código é feita a partir da criação das classes básicas.

REFERÊNCIAS

AGUIAR, Y.; LULA JUNIOR, B.; LIMA, C.; LIMA, G.; GOUVEIA, R. A. **Uso de Protótipos no Processo de Concepção de Interfaces do Usuário**. In: Congresso de Pesquisa e Inovação Tecnológica da Rede Norte e Nordeste de Educação Tecnológica da Paraíba – CONNEPI, 2.,2007, v. 1. p. 1-10.

BERKUN, S. 2000. **The Art of UI Prototyping**. Disponível em: <http://www.scottberkun.com/essays/essay12.htm>. Acesso em: mai. 2009.

CRNKOVIC, IVICA. 2003. **Component-Based Software Engineering–New Challenges in Software Development.** 25th International Conference on Information Technology Interfaces (ITI'03), p. 127-133, Cavtat, Croatia.Hibbs, C.; Tate, B. A. Ruby on Rails - Executando. Rio de Janeiro: Alta Books, 2006.

FRAMEWORK. Disponível em: <http://www.ibm.com/mrf>. Acesso em: Ago. 2009.

LICHTER, H., SCHINEIDER-HUFSMIDT, M., ZÜLLIGHOVEN, H. 1993. **Prototyping in Industrial Software Projects** – Bridging the Gap Between Theory and Pratice. IEEE.

OZAKI, H.; BAN, S.; GONDOW, K.; KATAYAMA, T. 2003. **An Environment for Evolutionary Prototyping Java Programs based on Abstract Interpretation**. In: Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference (Dec. 10 - 12, 2003). APSEC. IEEE Computer Society, Washington, DC, 362.

SOMMERVILLE I. **Software Engineering**. 8. Edition, Addison Wesley-Pearson Education Limited. 2006.

VARJÃO, Thiago F. G. **JavaOnRoad:** Uma Ferramenta de Desenvolvimento Ágil em Java. Monografia (Graduação) - Faculdade Sete de Setembro – FASETE. Paulo Afonso, BA. 2008.

XDoclet (2009). **Attribute-Oriented Programming**. Disponível em: <http://xdoclet.sourceforge.net>. Acesso em: Ago. 2009.