GERAÇÃO AUTOMÁTICA DE CÓDIGO EFICIENTE PARA DISPOSITIVOS ANDROID A PARTIR DE MODELOS UML

ALINE RODRIGUES TONINI; ABILIO GAMBIM PARADA; LISANE BRISOLARA DE BRISOLARA

Universidade Federal de Pelotas – {artonini,agparada,lisane}@inf.ufpel.edu.br

1. INTRODUÇÃO

Atualmente, dispositivos móveis como *smartphones* e *tablets* são considerados uma alternativa à computação pessoal, impulsionando o mercado destes eletrônicos, assim como o mercado de aplicativos para estes dispositivos. Entretanto o desenvolvimento de aplicações móveis difere do desenvolvimento tradicional, seja pelo uso da programação orientada a eventos e de um ciclo de vida mais rígido para as aplicações, ou ainda pelas limitações relacionadas ao dispositivo móvel. Estas limitações impõem restrições como consumo de energia, desempenho e memória, que exigem otimizações no código. Além disso, o sucesso das aplicações móveis está fortemente ligado ao tempo de colocação no mercado, impondo prazos curtos para o desenvolvimento.

O emprego de modelos facilita o projeto, enquanto que abordagens dirigidas por modelos (*Model Driven Development*) (SELIC, 2003) tradicionalmente aceleram o processo de desenvolvimento. Na MDD, o modelo é o principal artefato de projeto e estes são transformados até a obtenção da implementação para uma determinada plataforma alvo. Este paradigma tem por base a utilização da linguagem UML (OMG, 2013).

Muitas plataformas de desenvolvimento são presentes no mercado. Dentre estas, a plataforma Android (ANDROID, 2013a) se destaca como uma solução livre e de código abertoe por serconsiderada a mais utilizada no mercado. Visando melhor eficiência dos códigos, a Google (ANDROID, 2013b) propõe uma série de boas práticas de codificação com foco no desempenho.

Com base nos princípios da MDD e visando gerar automaticamente código para aplicativos móveis, nosso grupo de pesquisa vem definindo uma abordagem MDD que suporte a geração de código a partir de modelos UML. Uma ferramenta para captura de modelos e geração de código Android foi proposta em (PARADA, 2012). Neste trabalho, esta ferramenta é estendida para incluir a aplicação da boa prática do uso apropriado do for, sugerida pela Google, durante a geração de código e desta forma, gerar código mais eficiente para a plataforma Android. Através de experimentos realizados com um estudo de caso, é analisado e discutido o potencial da abordagem proposta e as melhorias relativas ao desempenho do código gerado.

2. METODOLOGIA

Neste trabalho, a ferramenta GenCode foi estendida para suportar o desenvolvimento Android e para a inclusão de otimizações durante o processo de geração de código. Esta ferramenta suporta a geração de código estrutural através de diagramas de classes e também comportamental através de diagramas de sequência, os quais representam o comportamento de um determinado método, incluindo condicionais, iterações e trocas de mensagens.

A Figura 1 ilustra a utilização desta ferramenta, onde tem por base a entrada de diagramas de classes e de sequência, e como saída o código gerado para a plataforma Android. Na geração de código, o projetista pode configurar a ferramenta para que esta apliquea boa prática do uso do for apropriado e tenha como resultado, um código mais eficiente.

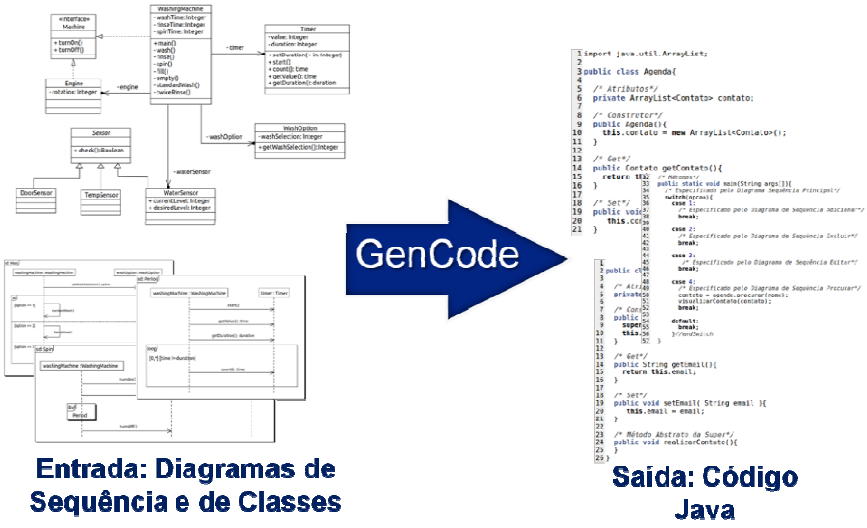

Figura 1. Representação do funcionamento da ferramenta GenCode

Existem três sintaxes diferentes que podem ser usadas para iterar um arranjo em códigos Java destinados a plataforma Android. A aplicação da boa prática do uso do *for* apropriado indica que esta escolha deve se basear principalmente no tipo de estrutura de dados e impacta no desempenho e na legibilidade do código. De forma resumida, no caso do uso de uma estrutura de ArrayList, o *for-each* (solução mais legível) deve ser evitada. A GenCode foi então alterada para gerar diferentes versões do código com e sem a aplicação da referida boa prática. Esta alteração afetou a geração de código comportamental.

Para avaliar os ganhos do emprego da geração automática com as otimizações de código, foi utilizada a aplicação móvel Snake como estudo de caso. Foi construído um modelo UML para esta aplicação e códigos foram gerados a partir deste modelo. Para a comparação e análise dos códigos gerados é utilizada a ferramenta DDMS (*Dalvik Debug Monitor Server*) (ANDROID, 2013c), que provê o tempo de execução do código Android (parte dele ou todo o código do aplicativo). No estudo de caso, a boa prática foi aplicada em um dos métodos da aplicação e utilizou-se a DDMS para obter o tempo de execução deste método, incluindo sub-chamadas a outros métodos, bem como para obter o tempo de execução total da aplicação. Todos os experimentos realizados foram repetidos trinta vezes, e a partir dos resultados obtidos foi calculada uma média utilizada nas comparações realizadas. Para analisar a significância das diferenças entre as médias foi utilizado o teste estatístico “t de Student”.

3. RESULTADOS E DISCUSSÃO

A partir do estudo de casode geração de código para o aplicativo Snake, foram realizadas análises de desempenho para os códigos gerados com a otimização do *for* e sem a referida otimização. Os resultados obtidos pela análise de desempenho são ilustrados no gráfico da Fig. 2, onde o eixo y representa o tempo da CPU em milissegundos (ms). Na Fig. 2 são apresentados os resultados da boa prática de escolha do *for* apropriado, observando o tempo de execução do

método otimizado (*enqueueDirection*) em relação ao código de toda a aplicação. Três implementações do *for* foram avaliadas. Na Tabela I estes resultados são detalhados incluindo o tempo médio de execução de cada versão do código e os desvios padrão, considerando as trinta execuções analisadas.

A Fig. 2 ilustra a proporção do tempo médio de execução do método enqueueDirection() em relação ao tempo médio de toda a aplicação, considerando cinquenta invocações do método que simulam um total de cinquenta jogadas. O gráfico permite observar uma variação no desempenho do método em relação a aplicação de acordo com o tipo de *for* implementado.

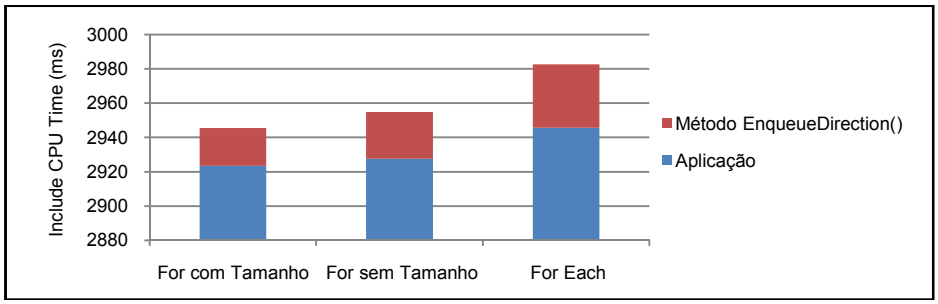

Figura 1. Desempenho: Método Avaliado em Relação à Aplicação.

Avaliando os resultados através do teste estatístico, foi concluído que as três maneiras de implementar o *for* apresentam resultados significativamente diferentes, portanto, a escolha do *for* apropriado tem impacto no desempenho do método avaliado. Ao implementar um *for* com maior abstração (*For-each*) em métodos frequentemente invocados, há um impacto negativo no desempenho. Isso se deve ao fato do *For-each* realizar chamadas internas aos métodos *hasNext()* e *next()*. Na implementação onde o tamanho do *array* é obtido a cada iteração do *loop* (*For* sem Tamanho) observa-se um desempenho intermediário. O melhor resultado foi obtido quando a iteração foi implementada através de um *for* onde o tamanho do *array* é obtido antes das iterações e armazenado em uma variável local.No entanto, o método modificado, através da aplicação da boa prática representa uma pequena porção do tempo de execução da aplicação e portanto, o que faz com que a melhoria não seja significativa quando considerado o tempo total da aplicação.

Tabela I. Resultados Experimentais: Tempo de Execução (ms)

	Aplicação		**Método enqueueDirection()**	
Otimização	Média	σ	Média	σ
For sem Tamanho	2927,4970	136,6422	27,4041	0,4528
For com Tamanho	2923,4090	114,3503	22,1056	1,0447
For Each	2945,7800	136,6422	36,7921	2,8601

4. CONCLUSÕES

Com a constante necessidade de aplicativos para dispositivos móveis torna-se necessário o uso de ferramentas que facilitem o desenvolvimento e ao mesmo tempo se preocupem com a eficiência do código gerado. Este artigo apresenta uma ferramenta que gera código Android eficiente a partir de modelos UML, considerando o paradigma MDD. Para obter código mais eficiente, a ferramenta aplica a boa prática de codificação que indica que o for apropriado deve ser adotado para iterar um arranjo de dados. Um estudo de caso demonstra

o emprego da ferramenta e uma análise do desempenho do código gerado é usada para demostrar o potencial da abordagem. Como trabalho futuro, será investigada a possibilidade de incorporar outras boas práticas na ferramenta de geração de código, bem como a realização de experimentos com outras aplicações.

5. REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID. Anatomia de uma aplicação Android. Disponível em: <http://developer.android.com/guide/components/fundamentals.html>. Acesso em: 19 set. 2013a.

ANDROID. Boas práticas para o desenvolvimento de aplicações Android. Disponível em: <http://developer.android.com/guide/practices/index.html>. Acesso em: 19 set. 2013b.

ANDROID. DDMS - Dalvik Debug Monitor Server . Disponível em: <http://developer.android.com/tools/debugging/ddms.html>. Acesso em: 19 set. 2013c.

OMG. Object Management Group. Disponível em: <http://www.omg.org/>. Acesso em: 19 set. 2013.

PARADA, A. G. ; BRISOLARA, L..A model driven approach for Android application development. SBESC, Brazilian Symposium on Computing System Engineering, Natal, 2012.

SELIC, B.Models, software models, and UML.In: LAVAGNO, L.; MARTIN, G.; SELIC, B. (Ed.) UML for real. Boston: Kluwer Academic Publishers, 2003. p.1-16.