UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OTIMIZANDO COMPILADORES DE ASPECTJ PARA JAVA ME

PROPOSTA DE TRABALHO DE GRADUAÇÃO

2007.1

Aluno: Fernando Henrique Calheiros Lopes (fhcl@cin.ufpe.br)
Orientador: Paulo Henrique Monteiro Borba (phmb@cin.ufpe.br)

Recife, 18 de Maio de 2007

Sumário

1. Contexto

O desenvolvimento de jogos móveis é um dos campos mais promissores da indústria de entretenimento digital. Projeções mostram que, até 2011, espera-se que sejam movimentados, em todo mundo, aproximadamente US$ 7 bilhões com jogos móveis [Tercek, 2007].

Um dos principais problemas do desenvolvimento de jogos móveis é o *porting*: para uma empresa atuar nesse mercado é necessário que os jogos desenvolvidos pela mesma estejam disponíveis para uma grande variedade de aparelhos que possuem características, recursos e API's distintas. Cada produto gerado para atender uma família de aparelhos de um mesmo jogo é chamado uma instância desse jogo.

O processo de *porting* de jogos consiste na adaptação de um jogo a fim de que ele possa ser executado em diferentes aparelhos, que envolve a identificação dos pontos de variação e o tratamento dos mesmos.

De acordo com [Tercek, 2007], em 2006, o processo de *porting* representou um terço do custo total do desenvolvimento de jogos móveis.

As duas técnicas mais usadas pela indústria no processo de *porting* para tratar as variações no código dos jogos são: o uso de compilação condicional [Câmara et al, 2006] que deixa o código entrelaçado com os trechos de diferentes produtos no mesmo arquivo, e alteração de *bytecodes* [Tira, 2007]. Compilação condicional tem a vantagem de não adicionar overhead ao tamanho dos jogos.

Outras técnicas estão sendo analisadas para substituir compilação condicional. Uma das técnicas mais promissoras para tratamento de variações é o uso de Programação Orientada a Aspectos (AOP) [Kiczales et al., 1997][Alves et al., 2006],, através de AspectJ, uma extensão à linguagem Java.

Um empecilho para o uso de AspectJ, como mecanismo de inserção de variações no código de jogos desenvolvidos em Java ME [Java ME, 2007], é o overhead no código gerado pelos compiladores AspectJ. No contexto de jogos móveis esse overhead é proibitivo, pois boa parte dos aparelhos usados no mercado (por exemplo, no Brasil) tem limitações de memória. Mesmo que a tendência do mercado seja os aparelhos evoluirem para ter mais memória, as aplicações também evoluem em complexidade e sempre ficam no limite, portanto uma diminuição deste overhead pode ser um diferencial para a criação de aplicações mais ricas.

Um exemplo simples desse overhead é que cada aspecto gera um .class próprio. No contexto de jogos móveis, onde o espaço disponível em memória de armazenamento e de execução é consideravelmente restrito, a adição de um .class pode restringir o número de features do jogo devido ao menor espaço disponível após a introdução dos mesmos.

Otimizadores de código [ProGuard, 2007][Retroguard, 2007] possuem passos de redução, onde o tamanho do bytecode de jogos em J2ME é diminuído

consideravemente, tais como remover classes e atributos não usados, diminuir o nome de classes, atributos e métodos, entre outras.

Mesmo com o uso de otimizadores de código, o uso de AspectJ ainda deixa um overhead proibitivo. Se os pontos onde os compiladores de AspectJ são ineficientes, no sentido de gerar código menor, forem identificados e modificações forem realizadas para melhorar a geração de código, o uso de AspectJ como mecanismo de inserção de variações de jogos em J2ME se tornaria uma opção viável.

2. Objetivos

A pesquisa terá por finalidade *melhorar a geração de código AspectJ para torná-la viável para o uso do mesmo como mecanismo de inserção de variações em jogos Java ME*.

As contribuições deste trabalho serão:

- Identificação de gargalos na geração de código dos compiladores de AspectJ
- Implementação de melhorias em um dos compiladores
- Identificação de boas práticas no uso de AspectJ como mecanismo de inserção de variações em jogos Java ME

3. Metodologia

Para atingir os objetivos dessa pesquisa serão executadas cinco etapas: análise comparativa, discussão, implementação, coleta de resultados e conclusão.

Na ***análise comparativa*** será feita comparação entre código gerado pelos compiladores ajc [ajc, 2007] e ABC [Avgustinov et al, 2005] em um jogo real desenvolvido por um parceiro industrial [Meantime, 2007] e consistirá de dois passos.

O primeiro passo da análise será a comparação do tamanho dos *jar's* das instâncias, da versão original do jogo, ou seja, apenas com compilação condicional, com os de uma versão do jogo com variações extraídas para aspectos e compilada com o ajc e com o ABC.

O segundo passo consistirá numa inspeção individual no código gerado por cada tipo de construção utilizada para tratar as variações existentes.

Na ***discussão***, os resultados levantados na análise serão utilizados para propor melhorias na geração de código de AspectJ e para selecionar qual compilador será utilizado na etapa de ***implementação***. Nesta fase, inicialmente, será feito um estudo da arquitetura do mesmo realizado para indicar a melhor forma de implementá-las. Nesta etapa também será feita a prova de que as melhorias propostas preservam o comportamento do código gerado.

A ***coleta de resultados*** consistirá numa repetição da análise apenas com o compilador otimizado, para comparação de tamanho dos jar's gerados antes e depois da implementação das melhorias e quantificação da diminuição no tamanho do código.

Na ***conclusão*** será feita uma apreciação dos resultados obtidos, serão levantadas as lições aprendidas e boas práticas identificadas durante a realização desta pesquisa e serão apresentadas as oportunidades de trabalhos futuros.

4. Cronograma

O cronograma abaixo mostra as etapas, com suas respectivas datas, em que será desenvolvido o trabalho de graduação descrito acima.

	Mês															
Atividade	Maio/07				Junho/07				Julho/07				Agosto/07			
Análise – 1º passo			■	■												
Análise – 2º passo				■	■	■										
Discussão						■	■	■								
Implementação								■	■	■	■					
Coleta de Resultados											■	■	■			
Escrever o relatório final					■	■	■	■	■	■	■	■	■	■	■	
Elaborar apresentação final													■	■	■	

5. Referências

TERCEK, R. **The First Decade of mobile Games**. 2007. Presentation at GDC Mobile 2007, San Francisco

CÂMARA, T.; LIMA, R.; GUIMARÃES, R.; DAMASCENO, A.; ALVES, V.; MACEDO, P.; RAMALHO G. **Massive Mobile Games Porting: Meantime Study Case**. In Brazilian Symposium on Computer Games and Digital Entertainment - Computing track, Recife, Brazil, 2006.

ALVES, V.; COSTA NETO, A.; SOARES, S.; SANTOS, G.; CALHEIROS, F.; NEPOMUCENO, V.; PIRES, D.; LEAL, J.; BORBA, P. **From Conditional Compilation to Aspects: A Case Study in Software Product Lines Migration**. In 1st Workshop on Aspect-Oriented Product Line Engineering, in conjunction with 5th ACM International Conference on Generative Programming and Component Engineering (GPCE'06), Portland, USA, October 2006

KICKZALES, G.; IRWIN, J.; LAMPING, J. ; MENDHEKAR, A. ; MAEDA, C.; LOPES, C. V. ; LOINGTIER, J-M. **Aspect-Oriented Programming**, In: ECOOP'97, Finlândia – 1997

Java ME, **Sun MicroSystems Java Platform Mobile Edition**. http://java.sun.com/j2me, 2007

Tira. **Tira Wireless**, http://www.tirawireless.com, 2007

ajc. **The AspectJ Compiler**. http://www.eclise.org/aspectj, 2007

AVGUSTINOV, P.; CHRISTENSEN, A. S.; HENDREN, L.; KUZINS, S.; LHOT´AK, J.; LHOT´AK, O.; SERENI, D; SITTAMPALAM, G.; Tibble, J. **abc: An extensible AspectJ compiler**. In AOSD 2005, pages 87–98, March 2005

MEANTIME. **Meantime Mobile Creations**. http://www.meantime.com.br, 2007

ProGuard. **ProGuard, java class file shrinker, optimizer and obfuscator.** http://proguard.sourceforge.net, 2007

RetroGuard. **RetroGuard for Java Bytecode Obfuscation.** http://www.retrologic.com/retroguard-main.html, 2007

6. Data e Assinaturas

Recife, 18 de maio de 2007

_____

Paulo Henrique Monteiro Borba
(Orientador)

_____

Fernando Henrique Calheiros Lopes
(Aluno)