Adicionando Variáveis a um Tipo Derivado: Refatoração no Photran

Roberto Lopes do Nascimento Filho[1], **Gustavo Rissetti**[1], **Andrea S. Charão**[1]

[1]Laboratório de Sistemas de Computação
Universidade Federal de Santa Maria – UFSM

{robertol, rissetti, andrea}@inf.ufsm.br

Resumo. *Photran é um plugin para desenvolvimento em linguagem Fortran no IDE Eclipse. Dentre os recursos dessa ferramenta, destacam-se as refatorações, que permitem melhorar o código de programas Fortran legados, sendo que algumas dessas refatorações foram desenvolvidas no Laboratório de Sistemas de Computação (LSC) da Universidade Federal de Santa Maria (UFSM). No presente trabalho, descreve-se o aprimoramento da refatoração Add Variable To Derived Data Type, que foi originalmente desenvolvida no LSC como parte de um trabalho de mestrado.*

1. Introdução

A linguagem de programação Fortran, embora sendo considerada antiga, ainda possui larga utilização em inúmeros programas, sobretudo naqueles ligados a programação de alto desempenho. Tais programas, devido ao tempo de vida da linguagem, muitas vezes apresentam trechos de código desenvolvidos por diferentes programadores, e codificados em diferentes versões da linguagem (principalmente Fortran 77 e Fortran 90).

Para desenvolver programas em Fortran, uma opção é utilizar a ferramenta Photran [eclipse.org 2011, Eipe 2004], um plugin para o IDE Eclipse voltado para códigos desta linguagem e que conta com recursos que facilitam a programação. Um dos recursos em destaque nessa ferramenta é a **refatoração**, que é um processo semi-automático executado com a intervenção do programador que permite alterar trechos do código. Esse processo visa melhorar atributos do código tais como legibilidade e organização, além de otimizar algumas ações desempenhadas pelo programa, sem alterar seu resultado final de execução [Opdyke 1992, Fowler et al. 1999, Tourwé and Mens 2004].

Embora já conte com diversas refatorações, inclusive algumas produzidas em trabalhos anteriores desenvolvidos no LSC (UFSM), ainda há recursos importantes que podem ser implementados no Photran. Uma refatoração que está em desenvolvimento é a *Add Variable To Derived Data Type*. Esta refatoração permite acrescentar variáveis existentes a um tipo derivado existente, permitindo que o programador melhor organize o código e gerencie de modo seguro as variáveis que precisam ser mantidas juntas durante a execução do programas.

Neste trabalho, descreve-se o aprimoramento da refatoração *Add Variable To Derived Data Type*, buscando suprir algumas limitações no seu desenvolvimento. No restante deste texto, discute-se inicialmente a refatoração de programas em Fortran e o plugin Photran (seção 2), para depois apresentar-se o aprimoramento da refatoração alvo deste trabalho (seção 3).

2. Fortran e Refatoração

A linguagem de programação Fortran constitui um marco importante na história da computação. Mesmo sendo uma linguagem bastante antiga, ainda continua sendo utilizada, sobretudo em projetos científicos. Todavia, apesar das vantagens oferecidas pela linguagem (p.ex. tratamento nativo de dados multidimensionais), trabalhar com códigos legados oferece um desafio aos programadores.

Durante sua existência, o Fortran passou por diferentes versões, e grande parte dos códigos ainda em uso podem contar com mesclas destes padrões, que mantém compatibilidade entre si. Além disso, algumas práticas que eram usadas em programas antigos hoje são consideradas inadequadas, tornando alguns códigos passíveis de erro, e dificultando seu entendimento e manutenção.

Uma forma de tornar códigos de antigos programas Fortram compreensíveis, seja para corrigir eventuais falhas ou adicionar novos recursos, é através da refatoração de código-fonte. Refatoração pode ser definida como uma alteração feita na estrutura interna de um sistema de software para torná-lo mais fácil de ser entendido e menos custoso de ser modificado, sem alterar o seu funcionamento aparente [Opdyke 1992, Fowler et al. 1999].

A refatoração, embora possa ser efetuada manualmente, pode ser realizada por ferramentas automáticas. Um programa que habilita refatorar códigos Fortran é o Eclipse através de seu plugin Photran, usado para desenvolver a refatoração abordada neste trabalho.

2.1. Photran

O Photran oferece suporte à programação em linguagem Fortran 77, 90, 95, 2003 e 2008, com ênfase na refatoração automatizada de códigos Fortran [Chen and Overbey 2010]. Essa ferramenta possui um analisador sintático completo e uma representação de programa, com o Photran VPG (*Virtual Program Graph*), que gera ASTs (*Abstract Syntax Tree*) do código-fonte, permitindo a sua manipulação e edição [Overbey 2007]. Com isso, o Photran oferece uma infraestrutura para refatoração de código-fonte que consiste de representações abstratas do programa, mecanismos para visualizar e comparar diferenças antes e depois da refatoração e, ainda, meios de cancelar ou desfazer uma refatoração efetuada.

Para automatizar uma refatoração no Photran são necessários três passos: fazer uma pré-validação da refatoração; fazer a manipulação da AST (introduzindo as mudanças no código-fonte); e fazer uma pós-validação da refatoração, para garantir a integridade da AST. O Photran possibilita atuar sobre o código Fortran por meio da manipulação de ASTs e da utilização da base de informações existentes nos VPGs.

Para implementar e integrar uma refatoração no Photran, é necessário estender o comportamento de algumas classes presentes em seu *framework*, tais como *FortranEditorRefactoring* e *FortranResourceRefactoring*. Nessas classes devem ser codificadas as ações e verificações de cada refatoração, através dos métodos *getName()*, *doCheckInitialConditions()*, *doCheckFinalConditions()* e *doCreateChange()*. O principal método a ser codificado é o método *doCreateChange*, que aplica a refatoração propriamente dita, fazendo as modificações necessárias no código-fonte.

3. Aprimoramento da Refatoração *Add Variable To Derived Data Type*

Um dos recursos introduzidos no Fortran 90 é a possibilidade de o programador definir seus próprios tipos derivados de dados (estruturas de dados). O uso desse recurso facilita a programação, e possibilita uma abstração das variáveis, permitindo o agrupamento de alguns atributos em uma entidade (estrutura), facilitando o seu acesso e sua modificação. Um exemplo de um tipo derivado de dados pode ser observado na Figura 1, onde é definido o tipo Ponto, que representa um ponto em um espaço bidimensional.

```
type Ponto
   real :: x, y
end type Ponto
```

Figura 1. Exemplo de um tipo derivado de dados.

Em um trabalho anterior [Rissetti 2011], foram propostas duas refatorações voltadas para tipos derivados de dados: *Transform to Derived Data Type* e *Add Variable to Derived Data Type*. A primeira serve para gerar um novo tipo derivado a partir da seleção de declarações de variáveis no código-fonte da aplicação. Já a segunda, serve para adicionar variáveis a um tipo derivado de dados já existente no código-fonte.

Em sua implementação original, a refatoração *Add Variable to Derived Data Type* possuía algumas limitações. Uma delas é a falta de verificação da existência de uma variável no interior do tipo derivado de dados com o mesmo nome da variável selecionada para inserção no mesmo. A verificação da possibilidade de inserção da variável no tipo derivado de dados é o principal aprimoramento da refatoração *Add Variable to Derived Data Type* almejado neste trabalho.

Na refatoração *Add Variable to Derived Data Type*, o método *doCheckInitialConditions()* verifica a seleção de declarações de variáveis no código-fonte, prosseguindo sua execução apenas se declarações de variáveis foram selecionadas.

Após selecionar as variáveis que deseja adicionar ao tipo derivado de dados, o usuário deve informar o nome do tipo derivado no qual as variáveis devem se adicionadas, e uma instância do mesmo para ser usada nos locais onde as variáveis selecionadas eram usadas no código-fonte. O método *doCheckFinalConditions()* verifica se estas informações passadas pelo usuário estão corretas, para dar prosseguimento à execução da refatoração.

O aprimoramento efetivado nesta refatoração consiste na codificação do método *doCreateChange()*. Esse método é responsável por fazer as devidas transformações no código-fonte. Para que isso seja realizado, a AST é pesquisada e os nomes das variáveis do tipo derivado são comparados com os nomes das variáveis a serem inseridas no tipo derivado. Sempre que houver coincidência entre os nomes previamente existentes e os novos nomes a serem inseridos, as variáveis que estão sendo inseridas e todas as suas ocorrências têm seu nome alterado a fim de que não ocorram erros devido a duplicidade de declaração de variáveis dentro de um mesmo tipo derivado. A seguir, o método altera o código Fortran, copiando para dentro da declaração do tipo derivado a declaração das variáveis selecionadas e alterando todas as ocorrências destas no código-fonte. A Figura 2 mostra um exemplo de como um código Fortran pode ser alterado pela refatoração.

```
!Código original
program exemploponto
implicit none
type :: ponto
  real :: x, y
end type ponto
type (ponto) :: p
real :: x
p%x= 3
p%y= 4
x= 5
print *, p
end program exemploponto
```

```
!Após após a refatoração
program exemploponto
implicit none
type :: ponto
  real :: x, y, variavelx
end type ponto
type (ponto) :: p
p%x= 3
p%y= 4
p%variavelx= 5
print *, p
end program exemploponto
```

```
!Após refatoração Rename
program exemploponto
implicit none
type :: ponto
  real :: x, y, z
end type ponto
type (ponto) :: p
p%x= 3
p%y= 4
p%z= 5
print *, p
end program exemploponto
```

Figura 2. Exemplo de aplicação da *Add Variable to Derived Data Type*.

4. Considerações Finais

Neste trabalho, utilizou-se os recursos da ferramenta Photran para continuar o desenvolvimento de uma refatoração de código Fortran. Com o aprimoramento implementado, a refatoração *Add Variable to Derived Data Type* passou a tratar as ocorrências de duplicidade de nomes em um tipo derivado de dados, eliminando essa limitação que estava presente em sua codificação original.

Além de produzir este resultado, o presente trabalho permitiu aprofundar o conhecimento sobre os recursos da ferramenta Photran. A partir deste ponto, pretende-se dar continuidade ao desenvolvimento de refatorações voltadas para programas Fortran de alto desempenho.

Referências

Chen, N. and Overbey, J. (2010). *Photran 7.0 Developer's Guide*.

eclipse.org (2011). Photran - An Integrated Development Environment for Fortran. Disponível em: `http://www.eclipse.org/photran/`. Acesso em: dezembro de 2011.

Eipe, R. M. (2004). Extending eclipse to create an ide plugin for a new language with fortran as a case study. Master's thesis, University of Illinois, Urbana-Champaign, EUA.

Fowler, M., Beck, K., Brant, J., Opdyke, W., and Roberts, D. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, Massachusetts.

Opdyke, W. (1992). *Refactoring Object-Oriented Frameworks*. Tese de doutorado, University of Illinois, Urbana-Champaign, EUA.

Overbey, J. L. (2007). Virtual program graph. Disponível em: `http://jeff.over.bz/software/vpg/doc/`. Acesso em: dezembro de 2011.

Rissetti, G. (2011). Catálogo de refatorações para a evolução de programas em linguagem fortran. Master's thesis, Universidade Federal de Santa Maria, Santa Maria, RS.

Tourwé, T. and Mens, T. (2004). A survey of software refactoring. *IEEE Transactions on Software Engineering*, 30(2):126–139.