Extensões para Refatoração de Código Fortran no Eclipse

Bruno B. Boniati[1], Gustavo Rissetti[2], Andrea S. Charão[2], Eduardo K. Piveta[2]

[1]Colégio Agrícola de Frederico Westphalen (CAFW)
Universidade Federal de Santa Maria (UFSM)
Caixa Postal 54 – 98.400-000 – Frederico Westphalen – RS – Brasil

[2]Departamento de Eletrônica e Computação – Universidade Federal de Santa Maria (UFSM) – Santa Maria – RS – Brasil.

brunoboniati@gmail.com, {rissetti,andrea,piveta}@inf.ufsm.br

Abstract. *Refactoring is a software engineering technique aiming to perform internal changes in application source code, without influencing its behaviour. In scientific computing, in which there is a lot of legacy code, refactoring is not enough explored, because such applications are written in non-object oriented languages, such as Fortran. In this context, we explore this field by the development and deploying of new refactorings for the Eclipse IDE, using the Photran framework (a plugin for Fortran programming integrated into Eclipse).*

Resumo. *Refatoração é uma técnica de engenharia de software que visa aplicar mudanças internas no código-fonte de aplicações, sem afetar seu comportamento observável. Na computação científica, onde existem muitos códigos legados, a refatoração é pouco explorada, pois tais códigos são escritos em linguagens não orientadas a objetos, como Fortran. Este trabalho explora tal lacuna através do desenvolvimento e disponibilização de extensões de refatoração para o ambiente integrado de desenvolvimento Eclipse, utilizando-se do framework Photran (um plugin para programação Fortran integrado ao Eclipse).*

1. INTRODUÇÃO

A evolução é uma propriedade natural em processos de desenvolvimento de software. Durante o ciclo de vida de um sistema, geralmente existe necessidade de evolução do software, seja para a adição de um novo requisito, ou para a alteração de funcionalidades existentes. O problema é que nem sempre o sistema está preparado para receber novos requisitos ou para suportar adaptações em suas funcionalidades. Dependendo de como essas alterações são feitas, elas podem acabar degradando a qualidade e enfraquecendo o projeto original do sistema.

Refatoração é uma técnica de evolução que consiste na aplicação de transformações na estrutura interna de aplicações, sem que isso afete seu comportamento observável [Fowler et al. 1999, Opdyke 1992]. Tal técnica é empregada com o intuito de melhorar a qualidade de software durante o ciclo de vida das aplicações.

As refatorações estão fortemente vinculadas ao paradigma da orientação a objetos e estão presentes de forma automatizada em diversas ferramentas alinhadas com este pa-

radigma. Na computação científica, a refatoração de código é uma técnica pouco explorada, principalmente em função de que boa parte do código legado de tais aplicações está escrita em linguagens procedurais [Johnson et al. 2006]. Esse é o caso do Fortran, uma linguagem estruturada cuja gramática original já possui mais de cinquenta anos e que, ainda hoje, é amplamente utilizada em aplicações científicas.

Este artigo descreve a automação de um conjunto de refatorações para código em Fortran, implementadas como extensões ao framework Photran do Eclipse, que é um popular ambiente integrado de desenvolvimento (Integrated Development Environment - IDE). Essa automação visa reduzir a lacuna existente entre a grande quantidade de código legado escrito em Fortran e o reduzido número de IDEs voltadas para essa linguagem, distribuídas como software livre.

O artigo está organizado como segue. A Seção 2 apresenta os recursos e as características do Eclipse e do Photran, bem como os requisitos para desenvolvimento e integração de ações de refatoração ao IDE. A Seção 3 descreve as extensões desenvolvidas, incluindo sua forma de atuação, questões de implementação e eventuais limitações. Por fim, a Seção 4 apresenta as considerações finais e as possibilidades de trabalhos futuros.

2. ECLIPSE E PHOTRAN

O Eclipse é um IDE com suporte para diversas linguagens, tais como Java, C e C++. Recentemente, por meio do plugin Photran, o Eclipse passou também a suportar a linguagem Fortran, em suas versões 77, 90, 95 e 2003. No texto Eclipse Plataform Technical Overview, o Eclipse é definido como “uma IDE para nada, e para nada em particular” [Beaton and des Rivieres 2006]. Isso caracteriza um pouco da filosofia de desenvolvimento do Eclipse, que é fortemente baseada na idéia de plugins, os quais são estendidos para prover em um só ambiente de desenvolvimento uma série de funcionalidades de suporte ao desenvolvimento de software.

Arquiteturalmente pode-se compreender o framework do Eclipse como sendo uma aplicação que disponibiliza uma interface gráfica padrão (com menus, barras de ferramentas, editores, etc.), na qual as funcionalidades são implementadas por meio de plugins. Um plugin é um artefato de software conectável por meio de uma interface padrão a outros artefatos. O núcleo do Eclipse (microkernel) implementa mecanismos para a descoberta dinâmica dos plugins. O gerenciamento das funções do ambiente é feito pelo núcleo e cada plugin responsabiliza-se por tarefas com escopo bem definido.

O Photran [Eclipse.org 2010] é um plugin para o Eclipse que estende as funcionalidades do IDE para suportar o desenvolvimento de código Fortran. Atualmente, seu desenvolvimento é mantido como um projeto oficial da Eclipse Foundation e faz parte de um macro projeto denominado PTP (Parallel Tools Platform). O objetivo do projeto PTP é explorar a deficiência de ferramentas para computação científica [Watson and DeBardeleben 2006] e de alto desempenho [Vanter et al. 2005], organizando em torno do IDE do Eclipse ferramentas para o desenvolvimento, a depuração, a distribuição e o monitoramento de tais aplicações.

Um dos objetivos do projeto Photran é disponibilizar um framework que possibilite o desenvolvimento rápido de ações de refatoração para código Fortran, reutilizando a infraestrutura fornecida pelo Eclipse [De 2004]. Para isso, são utilizadas reescritas de árvores sintáticas abstratas (ASTs), as quais são manipuladas através da adição, da modificação e da remoção de nós e informações desses nós em suas estruturas.

3. EXTENSÕES DESENVOLVIDAS

A automação de ações de refatoração normalmente requer a existência de um analisador sintático específico para a linguagem de programação com a qual se deseja trabalhar. A análise sintática utiliza algoritmos para a decomposição de sentenças de um código-fonte a partir de uma gramática formal e constrói uma árvore gramatical (AST, Abstract Syntax Tree) [Aho et al. 2006].

O plugin Photran disponibiliza, além de métodos para construção e manipulação da AST, uma estrutura denominada VPG (Virtual Program Graph). Um VPG agrega às ASTs algumas ligações adicionais entre seus nós. Essas ligações podem representar, por exemplo, um vínculo entre a utilização de determinada variável (em uma expressão) e sua respectiva declaração, ou ainda um comando e seu escopo (bloco lógico de atuação).

Através de uma VPG é possível obter um conjunto bastante rico de informações para auxiliar na manipulação dos nós de uma AST. A construção de ações de refatoração para a ferramenta Photran é feita a partir de um conjunto base de classes que são estendidas e da implementação de métodos de pré-validação, manipulação e pós-validação das ASTs. A utilização dessas ferramentas abstrai do programador a ação direta sobre o código-fonte, respeitando eventuais dependências e interligações entre suas entidades.

Neste trabalho, foram automatizadas algumas técnicas de refatoração aplicadas a características da linguagem Fortran, utilizando-se o framework Photran. As técnicas selecionadas objetivam a evolução da linguagem (refatoração de código escrito para determinada versão de Fortran, substituindo-o por construções de uma versão mais recente) e organização de código (refatorações que afetam apenas aspectos ligados ao projeto do código-fonte, facilitando a manutenção de código). A seguir, são detalhadas as técnicas de refatoração automatizadas.

3.1. Substituir Operadores Obsoletos

Dentre as várias evoluções sintáticas pelas quais Fortran tem passado ao longo do tempo, uma delas é a forma de representação dos operadores relacionais (por exemplo, o operador de igualdade .EQ. é equivalente ao sinal ==). Em geral, as construções mais recentes são mais facilmente entendidas, principalmente por programadores menos experientes em Fortran.

A substituição de operadores obsoletos consiste na alteração de símbolos obsoletos da linguagem, que nesse caso são operadores relacionais, por suas construções mais recentes. É uma técnica ligada à evolução e ao projeto do código-fonte. Sua mecânica de funcionamento consiste em percorrer a árvore sintática em busca de nós que representem operadores. Ao identificar uma construção obsoleta, o texto do token correspondente é alterado para o novo formato do operador relacional.

3.2. Extrair Subrotina

Extrair uma subrotina consiste em escolher um fragmento de código e transformá-lo no corpo de um novo subprograma substituindo o fragmento original por uma chamada ao novo subprograma. A principal problemática de longos trechos de código é a excessiva informação que fica oculta pela complexidade da lógica associada [Fowler et al. 1999].

Para extrair uma subrotina é preciso indicar um conjunto de linhas no editor de código e um nome para a nova subrotina. A ação de refatoração deve verificar dentro do conjunto de comandos selecionados para extração se existe a utilização de variáveis. Se não existirem variáveis, a refatoração deve simplesmente declarar um novo subprograma (SUBROUTINE) con-

tendo o código selecionado pelo programador e no lugar do código que foi selecionado deve ser incluído uma chamada ao subprograma (no caso do Fortran, através do comando CALL). Se houver utilização de variáveis no código a ser extraído é necessário transformá-las em parâmetros (caso sejam utilizadas além do código a ser extraído) ou então declará-las como variáveis locais dentro da nova subrotina (quando não são utilizadas externamente ao código extraído).

3.3. Introduzir Intenção

O atributo INTENT é uma construção relativamente recente (versão 90 do Fortran) e é utilizada para especificar a intenção de uso de determinado argumento em um subprograma. São três formas possíveis de indicar a intenção de uso de um argumento com o atributo INTENT: IN (quando o atributo somente é referenciado), OUT (se o atributo sofre somente alteração sem ser previamente referenciado) e INOUT (se o atributo é referenciado e alterado).

A automação da refatoração Introduce INTENT pressupõe a indicação por parte do programador de um subprograma no qual a refatoração atuará e também que o escopo do subprograma indicado seja implicit none, ou seja, exija a declaração explícita dos tipos de dados utilizados[1]. A declaração explícita pode ser conseguida através da refatoração Introduce Implicit None (disponível na ferramenta Photran).

A mecânica da refatoração detecta todas as declarações do subprograma que são argumentos do mesmo e verifica no corpo do subprograma que tipo de ação estes argumentos sofre. As ações podem ser basicamente duas: referência (em expressões, por exemplo) ou alteração (atribuição e leitura). Uma vez detectados os parâmetros e as ações que os mesmos sofrem, é possível inferir o tipo de INTENT.

3.4. Remover Variáveis Não Utilizadas

Normalmente, durante a codificação de um programa, podem sobrar variáveis inutilizadas, muitas vezes, por descuido do programador, que comenta algum bloco do código que não será mais utilizado sem remover as variáveis utilizadas exclusivamente no bloco. A eliminação de variáveis não utilizadas possibilita diminuir o tamanho do código-fonte e o tamanho do código executável gerado, além de melhorar sua legibilidade.

A mecânica de funcionamento desta refatoração consiste em percorrer as ASTs em busca de nós que representem a definição de variáveis. Quando uma definição de variável encontrada não possui nenhuma referência no código-fonte, significa que ela nunca foi usada depois de sua declaração. Assim essa definição de variável sem referências consiste em uma das variáveis esquecidas do código, e deve, portanto, ser eliminada da AST, para que o objetivo da refatoração seja alcançado.

3.5. Padronizar Sentenças

Na linguagem Fortran, variáveis podem ser declaradas de diferentes maneiras (uma variável por declaração ou múltiplas variáveis de mesmo tipo na mesma declaração). Essa mistura de gêneros de declarações (comum em grandes projetos de software, onde estão envolvidos diferentes desenvolvedores) pode dificultar a leitura de código, uma vez que uma grande lista de variáveis pode ser declarada, e quando o leitor do código chega ao final de tal lista, não lembra mais qual era o tipo de declaração das variáveis, por exemplo.

[1] *Fortran permite que as variáveis utilizadas não sejam previamente declaradas, neste caso considerando que variáveis que iniciem com as letras I..N sejam do tipo INTEGER e as demais do tipo REAL.*

A mecânica de funcionamento desta refatoração consiste em percorrer as ASTs em busca de nós que representem a declaração de variáveis. Quando alguma ocorrência de nós desse tipo for encontrada, deve ser verificada a existência de uma única declaração de variável presente no nó ou se o nó consiste em uma lista de declaração de variáveis. Caso seja uma lista de declaração de variáveis, devem ser criados novos nós de declaração, cada um contendo apenas uma variável que estava contida na lista de declaração do nó original. Assim, no final da refatoração todas as declarações de variáveis ficam padronizadas, com uma variável declarada em cada nó, sempre contendo o sinal de dois pontos utilizados na declaração, por exemplo, integer :: variável.

3.6. Dado para Parâmetro

No Fortran, uma variável pode ser inicializada de diversas formas e, usualmente, em programas existem variáveis que são constantes, isto é, tem o seu valor definido no início do código e nunca é alterado. No Fortran, existe um atributo que serve para esse fim, chamado parameter. Uma variável declarada como parameter tem um valor constante que o compilador propaga diretamente em todo o código-fonte. Porém, as vezes essa declaração é confundida com a declaração do tipo data. Uma variável do tipo data é uma variável estática. Quando se pensa em paralelizar um código-fonte, a fim de melhorar o desempenho, as variáveis estáticas podem ser acessadas concorrentemente por várias linhas de execução, dificultando a programação. Por isso é vantajoso usar parameter quando se quer apenas tornar uma variável constante em todo o código-fonte.

A mecânica de funcionamento desta refatoração consiste em percorrer as ASTs em busca de nós que representem declarações do tipo data. Quando um nó encontrado não possuir seu valor inicial modificado ao longo do código-fonte, nos deparamos com uma variável que deveria ser declarada com o tipo parameter, pois foi usada apenas para propagar um valor constante em todo o código. Portanto, tal nó deve ser substituído na AST por um novo nó com a declaração da variável do tipo parameter, preservando o nome e valor inicial que continha anteriormente.

4. CONSIDERAÇÕES FINAIS

As extensões foram desenvolvidas gradualmente e disponibilizadas aos desenvolvedores do Photran. Algumas delas já se encontram integradas à versão oficial da ferramenta.

Em trabalhos anteriores [Boniati et al. 2009, Rissetti et al. 2010] buscou-se avaliar o impacto do uso de técnicas de refatoração em aplicações escritas em Fortran, utilizando o código-fonte de uma aplicação cedida pelo Laboratório de Micrometeorologia, vinculado à Universidade Federal de Santa Maria. Essa aplicação foi escrita em Fortran 77, e é atualmente utilizada para analisar grandes conjuntos de dados captados por sensores em estações meteorológicas. As refatorações foram aplicadas de forma isolada, e combinadas. Em todos os casos os resultados produzidos pela compilação e execução do código refatorado e do código não refatorado foram os mesmos, sendo que o código refatorado apresenta melhor estruturação e legibilidade sem comprometer o desempenho da aplicação (requisito importante em aplicações científicas).

A utilização do Photran como ferramenta de apoio à codificação e à automatização de técnicas de refatoração merece destaque como contribuição do trabalho. Ferramentas como o Photran representam um importante avanço no sentido de se preencher a lacuna existente entre a grande quantidade de código Fortran e o limitado número de ferramentas livres com técnicas de refatoração integradas.

Em relação à ferramenta Photran, existem ainda diversas questões em aberto que vão desde otimizações nos mecanismos que analisam o código-fonte, até novos recursos para a ferramenta e principalmente a automatização de outras técnicas de refatoração. Desde o início de seu desenvolvimento, em meados de 2004, é possível observar um crescimento constante no número de usuários que utilizam a ferramenta, bem como de voluntários que implementam melhorias e lhe agregam funcionalidades.

REFERÊNCIAS

Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D. (2006). Compilers: Principles, Techniques, and Tools. Addison Wesley, 2 edition.

Beaton, W. and des Rivieres, J. (2006). Eclipse Platform Technical Overview. Technical report, The Eclipse Foundation.

Boniati, B. B., Charão, A. S., and Stein, B. O. (2009). Automação de Refatorações para Programas Fortran de Alto Desempenho. In X Simpósio em Sistemas Computacionais, pages 71–78, São Paulo, Brasil. SBC.

De, V. (2004). A Foundation for Refactoring Fortran 90 in Eclipse. Dissertação de mestrado, University of Illinois, Urbana-Champaign, EUA.

Eclipse.org (2010). Photran - An Integrated Development Environment for Fortran. Disponível em: http://www.eclipse.org/photran/. Acesso em junho de 2010.

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

Johnson, R., Foote, B., Overbey, J., and Xanthos, S. (2006). Changing the Face of High-Performance Fortran Code. A White Paper.

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

Rissetti, G., Charão, A. S., and Boniati, B. B. (2010). Incorporação de Novas Refatorações para Linguagem Fortran no IDE Eclipse. In X Escola Regional de Alto Desempenho, pages 233–236, Passo Fundo, Brasil. SBC.

Vanter, M. L. V. D., Post, D. E., and Zosel, M. E. (2005). HPC Needs a Tool Strategy. In Second International Workshop on Software Engineering for High Performance Computing System Applications, pages 55–59, St. Louis, EUA. ACM.

Watson, G. R. and DeBardeleben, N. (2006). Developing Scientific Applications Using Eclipse. Computing in Science and Engineering, 8(4):50–61.