Usando CORBA para Integrar um compilador desenvolvido em Java com ferramentas em outras linguagens de programação

Luiz Fernando Noschang[1], André Luís Alice Raabe[1,2]

[1] Bacharelado em Ciência da Computação
[2] Mestrado em Computação Aplicada
Universidade do Vale do Itajaí (UNIVALI) – Itajaí, SC - Brazil

{noschang,raabe}@univali.br

Abstract. *This article presents a solution for integrating the core of a compiler built using the Java language with other programming languages. Were analyzed different alternatives of integration and it was decided to use CORBA for it best meet of the desired requirements. All the core features were provided to other applications using CORBA. It was performed a case study of integration with an IDE built in C # which allowed the validation of the implementation and pointed performance problems that deserve further investigation.*

Resumo. *Este artigo apresenta uma solução para integração do núcleo de um compilador construído usando a Linguagem Java com outras linguagens de programação. Foram analisadas diferentes alternativas de integração e optou-se pela utilização de CORBA por melhor atender aos requisitos pretendidos. Todas as funcionalidades do núcleo foram disponibilizadas a outras aplicações usando CORBA. Foi realizado um estudo de caso de integração com uma IDE construída na linguagem C# a qual permitiu validar a implementação e apontou problemas de performance que merecem uma investigação posterior.*

1. Introdução

Durante o primeiro semestre de 2009, foi desenvolvida na UNIVALI (Universidade do Vale do Itajaí) uma linguagem de programação denominada Portugol 2.0. A linguagem surgiu de um projeto dos grupos de pesquisa na área de Informática na Educação, cuja proposta era alterar a sintaxe da linguagem Portugol (pseudocódigo escrito em português estruturado) para torná-la similar às linguagens C e PHP, e assim, facilitar a transição dos alunos para estas linguagens.

Buscando viabilizar o uso da nova linguagem pelos alunos, foi criado também um conjunto de ferramentas de desenvolvimento, para permitir a construção e a execução de programas escritos em Portugol 2.0. Foi criado um compilador denominado PortugolCore e uma IDE (Integrated Development Environment) denominada PortugolStudio. Ambas as ferramentas foram feitas na linguagem Java.

O PortugolCore compreende o núcleo da linguagem Portugol 2.0 e disponibiliza três principais serviços: análise de erros (léxica, sintática e semântica), geração de código intermediário (ASA – Árvore Sintática Abstrata) e execução de programas.

O PortugolStudio por sua vez, constitui o ambiente de desenvolvimento da linguagem e possui todos os recursos básicos de uma IDE: manipulação de arquivos de código-fonte (abrir, salvar, desfazer, etc.), execução e interrupção de programas, console para entrada e saída de dados, console para exibição dos erros de compilação e de execução, entre outros. Para compilar e executar os programas desenvolvidos, o PortugolStudio utiliza os serviços disponibilizados pelo PortugolCore.

O PortugolCore possui a vantagem de ser totalmente independente do PortugolStudio, permitindo que ele seja integrado com outras ferramentas desenvolvidas em Java. No entanto ainda era impossível integrá-lo a ferramentas desenvolvidas em outras linguagens de programação.

Esta restrição limitava a utilização da linguagem Portugol 2.0 em outros projetos de pesquisa, pois sem os serviços disponibilizados pelo PortugolCore, as aplicações que desejam utilizá-la precisam implementar estes serviços. Este é o caso, por exemplo, da ferramenta BIPIDE (Viera, 2009 e Vieira, Raabe e Zeferino, 2010), desenvolvida na UNIVALI utilizando a linguagem C# e na qual existe o interesse do grupo de pesquisa em permitir que a ferramenta reconheça a sintaxe do Portugol 2.0.

Assim, propôs-se neste trabalho desenvolver um mecanismo que permitisse integrar o PortugolCore com ferramentas escritas em outras linguagens de programação. O mecanismo de integração foi desenvolvido utilizando a arquitetura CORBA, a qual foi escolhida dentre várias outras tecnologias por melhor atender os requisitos desejados: Interoperabilidade, Reusabilidade, Licença e Documentação.

O artigo apresenta na seção 2 o PortugolCore detalhando suas funcionalidades. Na seção 3 apresenta o estudo comparativo das tecnologias de integração que levou a escolha do CORBA. Na seção 4 apresenta o estudo de caso desenvolvido e na seção 5 as conclusões deste trabalho.

2. O PortugolCore

O PortugolCore constitui o núcleo do Portugol 2.0 e foi construído utilizando técnicas tradicionais de construção de linguagens de programação que incluem especificação de expressões regulares, gramáticas livres de contexto e ações semânticas. Para facilitar seu desenvolvimento foi utilizada uma ferramenta para geração de compiladores chamada ANTLR (Another Tool for Language Recognition).

No momento de sua concepção, decidiu-se que o PortugolCore deveria ser desenvolvido como um projeto independente mas de modo que pudesse ser integrado a diferentes ambientes de desenvolvimento (IDEs). Tal decisão foi tomada tendo em vista os diferentes projetos em desenvolvimento na UNIVALI e que poderiam se aproveitar desta integração.

Assim sendo, o PortugolCore disponibiliza três serviços básicos: análise e tratamento de erros, geração de código intermediário e execução de programas, os quais são expostos através de um conjunto de classes que abstraem os detalhes de implementação.

Para integrar o PortugolCore a uma aplicação Java, primeiramente deve-se incluir o código compilado (arquivo JAR) no Classpath da aplicação. Uma vez que isto tenha sido feito as funcionalidades poderão ser acessadas a partir de uma classe de fachada (Facade). A seguir será descrita brevemente cada uma das funcionalidades.

2.1. Análise e tratamento de erros

O PortugolCore provê um mecanismo de análise e tratamento de erros para garantir a corretude dos algoritmos escritos em Portugol. Esta análise está internamente dividida em duas partes distintas: análise sintática e análise semântica e, ocorre nesta ordem. A análise sintática é responsável por detectar e tratar os erros sintáticos, isto é, os erros relacionados à má formação das construções gramaticais, como por exemplo, o não fechamento de um parêntese em uma expressão. A análise semântica por sua vez, identifica os erros semânticos, ou seja, as construções gramaticais sintaticamente corretas, mas que não possuem um significado válido dentro do contexto da linguagem, como por exemplo, a soma entre uma variável do tipo lógico (booleana) e uma variável do tipo inteiro.

2.2. Geração de código intermediário

O PortugolCore permite gerar o código intermediário de qualquer algoritmo escrito em Portugol 2.0. Este código intermediário é uma estrutura em árvore chamada ASA (Árvore Sintática Abstrata) onde cada nó da árvore é um objeto que representa uma instrução do código-fonte. Ao utilizar esta abordagem o PortugolCore abstrai a gramática, tornando a linguagem mais dinâmica. Não importa, por exemplo, se na gramática uma multiplicação é representada pelo operador '*' ou 'x', pois esta operação sempre será representada na ASA por um objeto "NoOperacao".

Internamente o PortugolCore utiliza a ASA para realizar a análise semântica e a execução dos programas escritos em Portugol 2.0. Entretanto, ela pode ser manipulada para obter outros resultados, como por exemplo: otimização de código, conversão do código-fonte em Portugol para outras linguagens de programação, análise de fluxo (ex.: detectar chamadas recursivas infinitas), detecção de código morto (ex.: atribuir uma variável a si mesma) entre outros. Para facilitar a criação de novas funcionalidades, a ASA do PortugolCore dá suporte ao padrão de projeto Visitor.

2.3. Execução de programas

O PortugolCore executa os programas em Portugol 2.0 percorrendo a ASA e interpretando cada nó no momento em que é encontrado, sendo classificado portanto, como um sistema de implementação híbrido. Para garantir sua independência de uma IDE, o PortugolCore provê um conjunto de interfaces para monitoramento e controle da execução do programa e para a entrada e saída de dados, que deverão ser implementadas pela aplicação utilizadora.

3. Tecnologias de integração

Para desenvolver o mecanismo de integração, foi realizada uma pesquisa bibliográfica das tecnologias de integração existentes, com o objetivo de identificar a mais adequada ao projeto.

As tecnologias foram avaliadas mediante uma série de requisitos desejáveis: Interoperabilidade, Reusabilidade, Licença e Documentação. Para cada um dos requisitos, foi estabelecido um peso indicando sua relevância para o trabalho e uma classificação utilizada para determinar seu valor. A soma do peso de todos os requisitos totalizava 100 pontos e o cálculo final da relevância foi feito utilizando uma média ponderada entre os requisitos, conforme mostra a Equação 1.

$$R = \frac{(VI * PI) + (VR * PR) + (VL * PL) + (VD * PD)}{PI + PR + PL + PD}$$ Equação 1

Onde: R = Relevância; VI = Valor Interoperabilidade; PI = Peso Interoperabilidade; VR = Valor Reusabilidade; PR = Peso Reusabilidade; VL = Valor Licença; PL = Peso Licença; VD = Valor Documentação; PD = Peso Documentação.

3.1. Interoperabilidade

Tendo em vista o cumprimento do objetivo geral do trabalho, a solução a ser adotada deveria permitir a integração com o maior número possível de linguagens de programação. A interoperabilidade foi classificada em [Alta = 100, Média = 50, Baixa = 25] e tinha peso de 25 pontos.

3.2. Reusabilidade

A principal motivação em possibilitar a integração do PortugolCore com outras ferramentas estava em permitir com que elas reutilizassem as funcionalidades oferecidas pelo PortugolCore ao invés de implementá-las. A solução a ser adotada deveria garantir a reutilização das implementações existentes. A reusabilidade foi classificada em [Contempla = 100, Não Contempla = 0] e tinha peso de 30 pontos.

3.3. Licença

Um dos objetivos específicos do trabalho era disponibilizar o PortugolCore e sua documentação em um repositório de projetos OpenSource para que ele viesse contribuir com a comunidade científica. Assim sendo, a solução a ser adotada deveria possuir licença livre ou OpenSource. A licença foi classificada em [OpenSource/Livre/Não Possui = 100, Proprietária/Indisponível/Outra = 0] e tinha peso de 30 pontos.

3.4. Documentação

A solução a ser adotada deveria possuir uma documentação ampla, que auxiliasse na elaboração e implementação do projeto do sistema. A documentação deveria conter descrições claras que possibilitassem sua compreensão, exemplos de utilização, códigos-fonte, diagramas, figuras, etc. A documentação foi classificada em [Rica = 100, Pobre = 50, Não Possui = 0] e tinha peso de 15 pontos.

Uma vez estabelecidos os requisitos, foi feito um estudo das várias tecnologias de integração pesquisadas, levantando as informações necessárias para avaliá-las em relação a estes requisitos. Logo após foi realizada uma análise comparativa entre todas

elas para determinar qual melhor atendia os requisitos especificados, conforme mostra o Quadro 1.

Quadro 1. Análise comparativa das tecnologias de integração

Tecnologia	Interoperabilidade	Reusabilidade	Licença	Documentação	Relevância
Arquivos textuais	Alta	Não contempla	Não possui	Não possui	55,00
CORBA	Alta	Contempla	Livre *	Rica	100,00
COM/ DCOM/ .NET	Média	Contempla	Livre	Rica	87,50
JNI	Média	Contempla	Livre	Rica	87,50
Java RMI	Baixa	Contempla	Livre	Rica	81,25
RPC	Alta	Não contempla	Não possui	Rica	70,00
Sockets	Alta	Não contempla	Não possui	Rica	70,00
XML	Alta	Não contempla	Não possui	Rica	70,00
WebService	Alta	Contempla	Não possui	Rica	100,00

* Depende da implementação CORBA

A partir desta análise comparativa foi possível identificar as arquiteturas CORBA e WebServices como as tecnologias mais relevantes para o trabalho. Embora estas duas tecnologias tenham obtido a mesma pontuação, a arquitetura CORBA foi a escolhida para a implementação do projeto pelos seguintes motivos: (i) o foco dos WebServices está na integração de aplicativos rodando em diferentes máquinas através da Internet, enquanto que, o foco do PortugolCore está na integração de duas aplicações (o PortugolCore e a aplicação que o utiliza) em ambiente Desktop localizadas na mesma máquina; (ii) os WebServices dependem de um aplicação Web Server para funcionar; e (iii) mesmo que o mecanismo de integração fosse implementado em CORBA, ainda seria possível transformá-lo em um WebService caso fosse necessário.

O CORBA (Common Object Request Broker) é uma arquitetura para desenvolvimento de sistemas de objetos distribuídos criada em 1991 pela OMG (Object Management Group), uma organização internacional que reúne diversas empresas. A arquitetura CORBA permite a comunicação entre objetos em linguagens de programação diferentes de forma transparente. Para permitir esta comunicação, CORBA utiliza uma linguagem chamada IDL (Interface Definition Language) que define as interfaces dos objetos distribuídos.

Para trabalhar com CORBA é necessário obter uma implementação da arquitetura em cada linguagem que se deseja integrar. Essas implementações são chamadas de ORB (Object Request Brokers) e são elas, por exemplo, que permitem o mapeamento das interfaces em linguagem IDL para as linguagens de programação comuns. A arquitetura CORBA possui uma documentação completa da especificação no

site da OMG. Além disso, cada implementação de CORBA tem sua própria licença e fornece sua própria documentação.

4. Implementação e estudo de caso

O mecanismo de integração proposto foi implementado utilizando a arquitetura CORBA e observando os requisitos não funcionais especificados na fase de projeto: (i) o sistema deverá ser desenvolvido utilizando a arquitetura CORBA; (ii) o sistema deverá suportar a integração com pelo menos uma linguagem de programação diferente da linguagem Java; e (iii) o sistema deverá ser implementado de forma que o PortugolCore não se torne dependente dele para funcionar. Para garantir este último requisito, o mecanismo foi dividido em dois módulos: módulo Java e módulo C#. A Figura 1 demonstra a estrutura e o funcionamento do mecanismo de integração.

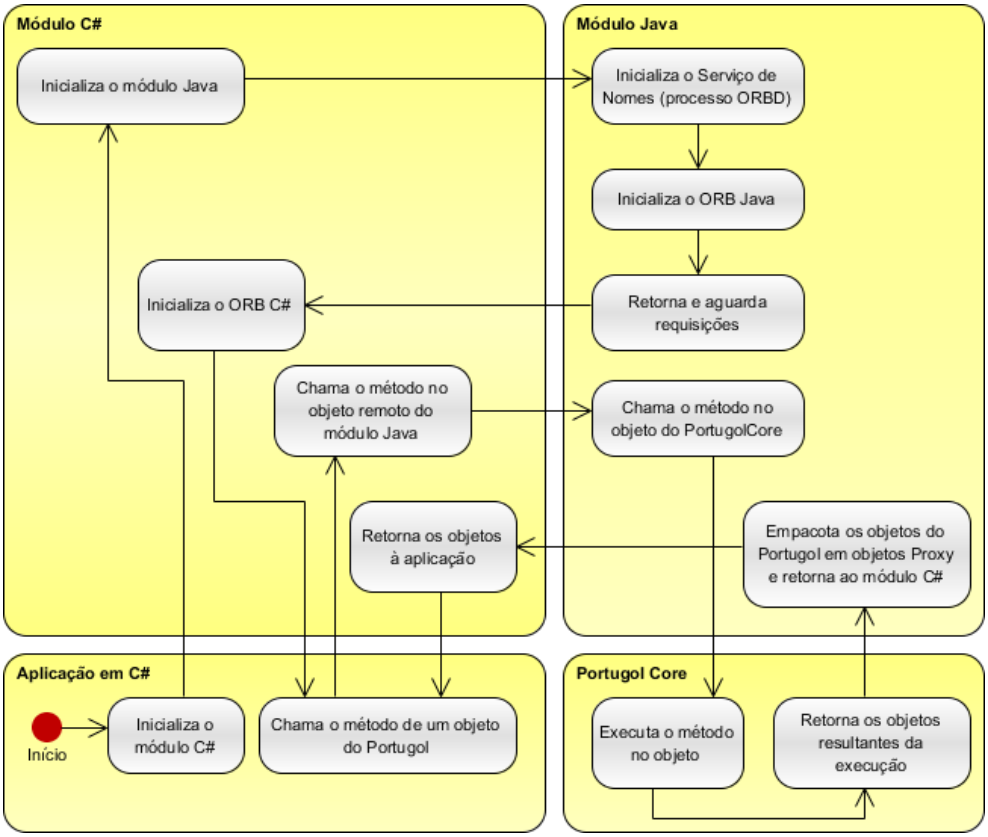

Figura 1. Funcionamento do mecanismo de integração

O módulo Java é a parte do mecanismo responsável por interagir com o PortugolCore e possui três funções principais: (i) gerenciar o serviço de nomes (Naming Service) necessário para o funcionamento da arquitetura CORBA; (ii) gerenciar o ciclo de vida do ORB Java; e (iii) empacotar os objetos Java vindos do PortugolCore dentro dos objetos CORBA utilizando o padrão de projeto Proxy. Durante o desenvolvimento optou-se por utilizar o serviço de nomes e a implementação ORB nativos do Java, de forma a não criar dependências externas.

O módulo C# é a parte do mecanismo responsável por interagir com as aplicações escritas em C# e possui duas funções principais: (i) inicializar e finalizar o

módulo Java; e (ii) gerenciar o ciclo de vida do ORB C#. Para este módulo optou-se por utilizar a biblioteca IIOP.NET como implementação ORB C#, a qual realiza a comunicação com o módulo Java através do protocolo IIOP e fica responsável por empacotar e desempacotar os objetos CORBA.

Para validar a implementação do mecanismo de integração desenvolvido, foi realizado um estudo de caso com a ferramenta BIPIDE desenvolvida na UNIVALI utilizando a linguagem C#. O estudo de caso foi realizado com o intuito de identificar e corrigir bugs no mecanismo de integração, de forma a tornar possível sua utilização. Maiores detalhes sobre o BIPIDE podem ser obtidos em (Viera, 2009 e Vieira, Raabe e Zeferino, 2010).

Foi estabelecido como critério de validação do mecanismo de integração, que todas as funcionalidades do PortugolCore pudessem ser utilizadas sem erros dentro da ferramenta. Se qualquer uma das funcionalidades apresentasse erros, a implementação seria invalidada. Com base neste critério, cada uma das funcionalidades do PortugolCore foi submetida a testes dentro do BIPIDE.

Para testar a análise e tratamento de erros do PortugolCore, foram escritos algoritmos em Portugol 2.0 contendo vários tipos de erros sintáticos e semânticos. Esta funcionalidade seria considerada validada caso fosse possível submeter estes algoritmos à análise do Portugol, capturar a lista de erros sintáticos e semânticos gerados durante a análise e exibi-los na interface da ferramenta. A funcionalidade passou em todos os testes realizados e, portanto, foi validada.

Para testar a geração de código intermediário (ASA) do PortugolCore, foram escritos em C# dois algoritmos de caminhamento na ASA. O primeiro algoritmo percorria a ASA do programa escrito em Portugol 2.0 e gerava o código assembly equivalente para o processador BIP, conforme mostra a Figura 2.

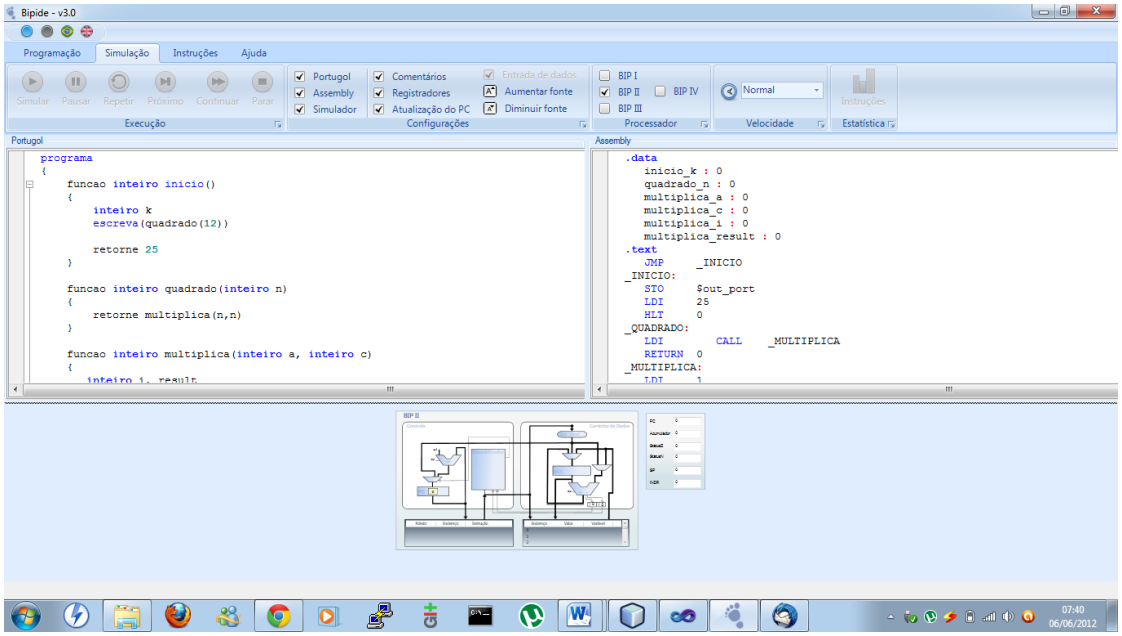

Figura 2. Geração de código assembly a partir da ASA do PortugolCore

O Segundo algoritmo funcionava como uma extensão do analisador semântico do Portugol, procurando os nós da ASA que representavam instruções não suportadas pelo processador BIP e disparando erros de análise ao encontrá-los.

Esta funcionalidade seria considerada validada se ambos os algoritmos conseguissem percorrer todos os nós da árvore necessários para realizar seu processamento, sem que fossem geradas exceções do CORBA. Os dois algoritmos foram executados com vários programas de teste escritos em Portugol e conseguiram percorrer corretamente a ASA validando, portanto, esta funcionalidade.

Para testar a execução de programas do PortugolCore, foram escritos algoritmos em Portugol livres de erros e contendo instruções de entrada e saída de dados. Esta funcionalidade seria considerada validada caso atendesse a dois requisitos: (i) se fosse possível inicializar e finalizar a execução de um programa através de botões na interface da ferramenta; e (ii) se fosse possível ler dados de entrada do usuário e escrever os dados de saída na interface da ferramenta.

Nos primeiros testes, o programa iniciava corretamente, mas na hora de realizar a entrada e saída dos dados ocorriam exceções no CORBA. Depois de exaustivos testes e várias investigações, descobriu-se que, o problema estava em implementar as interfaces de entrada, saída e monitoramento em uma única classe. A solução adotada foi implementar cada uma destas interfaces em uma classe diferente. O motivo deste comportamento ainda não foi descoberto, mas após fazer esta alteração a funcionalidade passou em todos os testes, sendo, portanto validada.

Outro problema encontrado na execução dos programas foi o desempenho durante a saída de dados. O BIPIDE levou cerca de 20 segundos para executar um laço de repetição com 1000 iterações escrevendo na saída de dados. O mesmo algoritmo levou cerca de 150 milissegundos para executar no PortugolStudio. Será necessário realizar um estudo mais aprofundado para determinar as causas deste problema, no entanto, isto não invalidou a implementação do mecanismo, já que o foco do trabalho estava na interoperabilidade e não no desempenho.

Também foram identificadas algumas limitações do módulo C# ocasionadas pela utilização da biblioteca IIOP.NET. A primeira limitação está relacionada ao fato de a biblioteca ter sido construída em cima da versão 2.0 da plataforma .NET. Isto dificulta ou impossibilita a utilização do módulo de integração com aplicações desenvolvidas em versões anteriores da plataforma, pois obriga que estas sejam migradas para a versão 2.0.

Outra limitação está relacionada ao compilador IDL da biblioteca, que ao ser executado gera uma DLL (Dynamic Link Library) contendo as interfaces para os objetos do PortugolCore. O problema é que a DLL é gerada na versão mais atual da plataforma .NET que estiver em uso na máquina. Durante o desenvolvimento do módulo C#, por exemplo, a versão da plataforma instalada na máquina era a versão 4.0. Isto fez com que o código fonte do BIPIDE (que usava a versão 3.5) tivesse que ser migrado para a versão 4.0 para possibilitar a utilização da biblioteca. Aparentemente não há como alterar este comportamento a não ser modificando o código fonte da biblioteca.

Ao final do estudo de caso, e depois de haver consertado os bugs encontrados, todas as funcionalidades do PortugolCore puderam ser utilizadas dentro do BIPIDE,

validando o mecanismo de integração desenvolvido e possibilitando sua utilização com outras ferramentas em C#. A Figura 3 demonstra a execução de um programa no BIPIDE utilizando o PortugolCore através do módulo de integração desenvolvido.

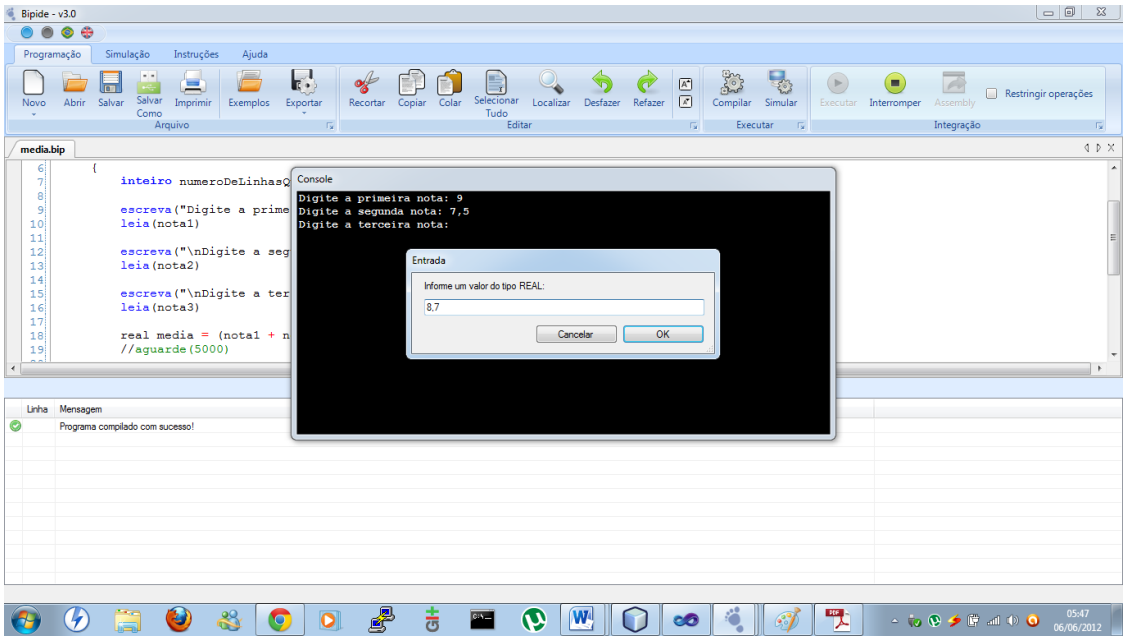

Figura 3. Execução de programas no BIPIDE utilizando o PortugolCore

5. Conclusões

Neste artigo foi apresentado um mecanismo de integração construído utilizando a arquitetura CORBA para permitir a integração entre o compilador PortugolCore, desenvolvido em Java na UNIVALI e aplicações desenvolvidas na linguagem C#.

O PortugolCore provê um conjunto de funcionalidades que podem ser acessadas como serviços por outras aplicações: análise e tratamento de erros, geração de código intermediário (ASA) e execução de programas. Dentre estas, a principal é a geração da ASA, que abre a possibilidade para a criação de muitas outras ferramentas e recursos, conforme foi apresentado na seção 2.2 e demonstrado na prática no estudo de caso realizado (Figura 2).

Espera-se que, ao permitir a integração do PortugolCore com outras linguagens de programação, este trabalho venha contribuir na criação de novas ferramentas de auxílio ao aprendizado de algoritmos. O objetivo principal é encorajar, facilitar e acelerar o desenvolvimento destas ferramentas fazendo com que, ao utilizarem os serviços disponibilizados pelo PortugolCore, preocupem-se em implementar apenas as funcionalidades específicas do domínio da ferramenta.

Espera-se também que, à medida que novas ferramentas sejam criadas, as funcionalidades mais relevantes destas ferramentas, possam ser incorporadas como novos serviços no PortugolCore, contribuindo assim para a evolução do mesmo.

Referências

AHO, A. V.; LAM, M. S.; SETHI, R.; ULLMAN, J. D. Compiladores: princípios, técnicas e ferramentas. 3.ed. Editora Pearson, 2007.

BISHOP, J.; HORSPOOL, R. N.; WORRAL, B. Experience in integrating Java with C# and .NET. Concurrency Computat.: Pract. Exper. 2003; 00:1-18.

BORSOI, T. B.; SCHULTZ, R. E. O. Estudo comparativo entre CORBA e Java RMI. In: Congresso Anual de Tecnologia de Informação. 2004, São Paulo. SP, 2004.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. Sistemas distribuídos: conceitos e projeto. 4.ed. Porto Alegre: Bookman, 2007.

CUMMINS, F. A. Integração de sistemas: EAI – Enterprise Application Integration: Arquiteturas para integração de sistemas e aplicações corporativas. 1.ed. Rio de Janeiro: Campus, 2002.

HOSTINS, H.; RAABE, A. L. A. Auxiliando a aprendizagem de algoritmos com a ferramenta Webportugol. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO - CONGRESSO DA SBC, 27., 2007, Rio de Janeiro. Anais... Rio de Janeiro: SBC, 2007. p. 96-105.

MIRANDA, E. M. Uma ferramenta de apoio ao processo de aprendizagem de algoritmos. 2004. Dissertação de Mestrado (Programa de Pós-Graduação em Ciência da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2004.

NARDI, A. R. Componentes CORBA. 2003. Dissertação de Mestrado (Ciência da Computação) – Universidade de São Paulo, São Paulo, 2003.

RAABE, A. L. A.; SILVA, J. M. C. Um ambiente para atendimento às dificuldades de aprendizagem de algoritmos. In: XIII WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 2005, São Leopoldo. RS, 2005.

SANTOS JUNIOR, A. L. Integração de sistemas com Java. 1.ed. Rio de Janeiro: Brasport, 2007.

SEBESTA, R. W. Conceitos de linguagens de programação. 5.ed. Porto Alegre: Bookman, 2003.

SIQUEIRA, L. L. Estudo comparativo entre plataformas de suporte a ambientes virtuais distribuídos. 2005. Dissertação de Mestrado (Ciência da Computação) – Universidade Federal de Uberlândia, Uberlândia, 2005.

TOGNI, J. D.; Uma Proposta para Auxiliar a Interoperabilidade entre Ambientes e Ferramentas de CAD para Microeletrônica. 2005. Dissertação de Mestrado (Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2005.

VIEIRA, P. V.; Bipide: Ambiente de Desenvolvimento Integrado para Utilização dos Processadores BIP no Ensino de Programação. 2009. Trabalho de Conclusão de Curso. (Graduação em Ciência da Computação) - Universidade do Vale do Itajaí.

VIEIRA, P. V.; RAABE, André Luís Alice ; ZEFERINO, Cesar Albenes. Bipide Ambiente de Desenvolvimento Integrado para a Arquitetura dos Processadores BIP. Revista Brasileira de Informática na Educação, v. 18, p. 32-43, 2010.