Instituto Superior de Engenharia de Lisboa

Área Departamental de Engenharia de Eletrónica e Telecomunicações e de Computadores

Debugger BrightScript

Celso de Almeida Fernandes

(Licenciado Engenharia Informática e de Computadores)

Dissertação para obtenção do Grau de Mestre em Engenharia Informática e de Computadores

Orientadores: Eng. Paulo Pereia

Eng. Pedro Pereira

Júri:

Presidente: Eng. Manuel Barata

fevereiro de 2016

1. Resumo

O BrigthScript é uma linguagem de programação baseada em java script e visual basic que foi criada pela Roku. A Roku é uma empresa que desenvolve e comercializa box's para visualização de conteúdos na televisão. O BrigthScript é a linguagem para desenvolvimento de aplicações para as boxs.

Após análise contata-se que as ferramentas disponíveis para o desenvolvimento de aplicações são escassas e pouco funcionais. Essas ferramentas são um plugin básico para o Eclipse (validação sintática) e as boxes disponibilizam um porto telnet para output e comandos básicos de debug.

Este projeto consiste em implementar uma ferramenta integrada que permita facilitar o desenvolvimento das aplicações, nomeadamente através das funcionalidades de validação sintática, compilação, intellisense e interação gráfica de debug.

A implementação baseia-se no desenvolvimento dum plugin de BrightScript para o Visual Studio. O Visual Studio é um IDE desenvolvido pela Microsoft e largamente utilizado para desenvolver aplicações para Windows. Esse plugin assenta nas APIs (Language Services) disponibilizadas pelo SDK do Visual Studio.

Esta ferramenta pode ser complementada com a implementação dum simulador de execução das aplicações, esta implementação é considerada opcional, podendo ser uma mais-valia para o desenvolvimento de aplicações.

O projeto enquadra-se no desenvolvimento da aplicação Sky Store pertencente à empresa Sky UK Limited.

2. Índice

3. Introdução

O projeto divide-se em três fazes. A primeira é uma faze de investigação e aquisição de conhecimentos sobre a temática dos compiladores. A segunda faze consiste na análise das ferramentas disponíveis, para compreender o seu funcionamento e avaliar o benefício da sua utilização. A terceira faze corresponde à implementação do pluggin.

3.1. Compiladores

Na primeira fase foram utilizados um conjunto de vídeos (Aiken, s.d.) disponíveis no site youtube, realizados pelo professor Alex Aiken, da universidade de Standford e o livro Modern Compiler Implementation (Appel, 2002) in Java, recomendado pelo orientador. As duas fontes abordam a problemática de formas semelhantes, apresentando as problemáticas da implementação dos compiladores, sugerindo uma implementação modular.

O BrightScript é uma linguagem interpretada, não sendo necessário implementar todos os paços do processo de compilação. Para a ferramenta de debug basta-nos implementar o Lexer e o Parser. Para implementar simulador será necessário implementar os restantes paços do processo, podendo optar pela compilação para a linguagem intermedia MSIL (Microsoft Intermediate Language).

3.2. Ferramentas

Na segunda faze analisa-se a utilização de ferramentas para gerar o Lexer e o Parser.

O GPlex (GPlex, s.d.) é um gerador de código C#, que gera um analisador léxico com base num ficheiro de especificação semelhante à linguagem de especificação Lex. O analisador é baseado no algoritmo “finite state autómata”.

O Gppg (Gppg, s.d.) é um gerador de código C#, que gera um Parser com a abordagem bottom-up, que reconhece linguagens LALR(1) (1 Look-Ahead token, Left-to-Right - rightmost derivation), com as desambiguações yacc tradicionais. A especificação é feita numa linguagem semelhante ao YACC.

Os geradores de código foram desenhados para funcionar em conjunto, podendo ainda assim ser utilizados em separado. Foram também desenhados para integrar com o Visual Studio, existindo opções para gerar classes para a integração.

Para além dos geradores de código, também se analisou a interface disponibilizada pelo Visual Studio para a implementação de extensões, nomeadamente extensões de suporte a linguagens e de debug.

3.3. Implementação

Na terceira faze será implementa uma extensão baseada na implementação do Python Tools (Python Tools, s.d.). O Python Tools é uma extensão do Visual Studio para a linguagem Python, desenvolvido pela Microsoft.

4. Compilador

O compilador será composto por três componentes, o analisador léxico (scanner), o Parser e a componente de integração com o Visual Studio. O objetivo das duas primeiras componentes é processar os ficheiros de código de forma a detetar erros de compilação e disponibilizar dados para o intellisense.

4.1. Scanner

O analisador léxico tem como responsabilidade gerar tokens para serem usados no Parser e validar se o código obedece ao léxico definido para a linguagem.

O léxico define o formato dos tokens que compõem a linguagem, o formato é definido através dum conjunto de regras e as regras são definidas através de expressões regulares.

Um analisador léxico é uma máquina de estados que tenta encontrar os tokens com as maiores dimensões. O processamento é feito carater a carater, enquanto existir possibilidade de encontrar um token maior, ao encontrar o token a string é retirada do input e é emitido o token.

O Scanner será gerado utilizando o Gplex, que gera uma classe em C# com a implementação. O código gerado é obtido de três fontes, da estrutura base da classe, motor de reconhecimento das patterns e os decoders/buffer de leitura. O motor de reconhecimento é composto por tabelas que definem a máquina de estados (FSA - finite state automaton). Estas tabelas são geradas através do ficheiro de especificação (*.lex).

4.2. Parser

O Parser tem como objetivo analisar a estrutura gramatical da linguagem, as frases. Valida se a ordem pela qual se apresentam os tokens é valida. Essa análise permite estruturar o código numa arvore de tokens. O output do Parser é uma árvore abstrata da estrutura do código (AST - Abstract Syntax Tree).

A gramatica é constituía por um conjunto de regras, que determinam a ordem dos símbolos nas sequências validas.

O Parser será gerado utilizando o Gppg, que gera uma classe em C# com a implementação. O código gerado implementa o algoritmo Shift-Reduce para gerar a AST (Abstract Syntax Tree).

4.3. Visual Studio

O Visual Studio utiliza o Lexer para atribuir diferentes cores ao tokens e o Parser para carregar os dados para o Intellisense. São ainda usados os dois componentes para validar o código, evitando os erros de compilação na box.

Para além destas funcionalidades será criado um novo tipo de projeto para gerir as configurações e associar os templates para criação de novos ficheiros de código e outros tipos de ficheiros.

5. Debugger

O debuger será uma tool que permite fazer deploy da aplicação para as boxs e fazer o interface entre o Visual Studio e o porto Tenet da box. Para além do porto Telnet a box disponibiliza um porto http, que permite simular ações do comando e iniciar aplicações.

5.1. Deploy

O deploy consiste em gerar um zip com todos os ficheiros da aplicação e fazer o upload para a box, utilizando o porto http.

5.2. Telnet

O porto Telnet tem duas funcionalidades, mostrar o output da execução da aplicação e servir de terminal de debug.

O output é feito através da utilização da função “print” no código.

Para utilizar o terminal de debug utiliza-se a função “stop” no código. Quando a função for executada a aplicação para e é possível interagir com o debuger utilizando os seguintes comandos:

Comando	Descrição
bsc	Mostra as instancias correntes
bscs	Mostra o sumario das instancias correntes
brkd	Para execução com mensagens não fatais
bt	Mostra o callstack
classes	Mostra as classes
cont or **c**	Continua execução do Script
down or **d**	Mover execução para baixo
exit	Sair
gc	Correr o garbage collector
help	Mostra a lista de comandos do sistema
last	Mostra a ultima linha executada

list	Mostra a função corrente
next	Mostra a proxima linha
print, p, or **?**	Imprime para o output
step, s, or t	Executa uma instruçã
up or **u**	Mover execução para cima
var	Mostra variaveis locais e os seus tipos
Any Brightscript statement	Executar BrightScript

Utilizando o debugger é possível visualizar o valor corrente das variáveis, verificar em que ficheiro e em que linha está a execução e controlar a execução do código.

Para facilitar a utilização do porto Telnet será gerado um Scanner e um Paser para interpretar o output do porto Telnet.

5.3. Http

Utilizando o porto http será implementado um comando para controlar a box através do PC.

5.4. Visual Studio

A componente de integração do debugger com o Visual Studio, vai permitir utilizar a interface visual de debug de forma semelhante às aplicações .Net. Permitindo definir breakpoint's, controlar a execução do código e visualizar o valor corrente das variáveis. O ponto de execução vai ser visualizado nos ficheiros de código através da funcionalidade de consultar o ponto de execução.

6. Bibliografia

Aiken, A. (s.d.). *Compilers Theory*. Obtido de https://www.youtube.com/playlist?list=PLLH73N9cB21VSVEX1aSRlNTufaLK1dTAI

Appel, A. W. (2002). *Modern Compiler Implementation in Java.* Cambrige University Press.

GPlex. (s.d.). Obtido de http://gplex.codeplex.com/

Gppg. (s.d.). Obtido de https://gppg.codeplex.com/

Python Tools. (s.d.). Obtido de https://github.com/Microsoft/PTVS