ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria e Scienze Informatiche

Utilizzo di Neverlang per la creazione di Domain Specific Languages

Tesi di laurea in:
Programmazione ad Oggetti

Relatore
Prof. Viroli Mirko

Candidato
Terenzi Mirco

Correlatore
Prof. Aguzzi Gianluca

II Sessione di Laurea
Anno Accademico 2023-2024

Sommario

Max 2000 characters, strict.

Indice

Capitolo 1

Introduzione

Write your intro here.

Struttura della Tesi

Capitolo 2

Background

2.1 Neverlang

Neverlang Language Workbench è un framework sviluppato presso l'Università di Milano dal professor Cazzola e dai suoi collaboratori, il cui scopo è favorire lo sviluppo di linguaggi di programmazione, in particolare secondo il paradigma di programmazione feature-oriented.

È basato sull'idea che i linguaggi di programmazione abbiano un'intrinseca divisione modulare in più caratteristiche, o *features*, ciascuna delle quali è implementata da un componente specifico. In accordo con tale visione, l'obiettivo del framework è definire i linguaggi tramite una divisione in frammenti, chiamati moduli, ognuno dei quali si occupa di implementare una specifica caratteristica e, infine, tramite la combinazione dei diversi moduli, ottenere un linguaggio di programmazione specifico per il contesto applicativo richiesto, ossia un Domain Specific Language (DSL) [Caz].

In particolare, all'interno di ogni modulo vengono definite due parti principali:

- la **sintassi**, utilizzando una grammatica formale;
- la **semantica**, in funzione della sintassi e sfruttando i vari elementi non-terminali e i loro attributi. Inoltre, il comportamento del componente può essere suddiviso in diverse fasi, ciascuna identificata da un ruolo specifico del componente.

Successivamente, i componenti del linguaggio vengono definiti combinando definizioni di sintassi e semantica provenienti da diversi moduli, all'interno di elementi detti *slice* [VC15].

Tra i vantaggi principali di Neverlang troviamo [Caz12]:

- **Modularità**: Ognuno dei moduli che compongono il linguaggio viene compilato separatamente, permettendo di utilizzarne uno o più di uno (in tal caso aggregandoli in uno *slice*) all'interno di altri linguaggi.

- **Riutilizzo**: Neverlang offre la possibilità di riutilizzare frammenti di linguaggio in più di un contesto. Ad esempio, un frammento può utilizzare la sintassi di un altro frammento definito in precedenza e ridefinire la semantica, o viceversa. Inoltre, è possibile ridefinire l'ordine dei simboli non-terminali utilizzati nella sintassi o nella semantica importata.

- **Estensibilità**: L'architettura modulare utilizzata all'interno di Neverlang facilita l'estensione di linguaggi esistenti. Per aggiungere nuove funzionalità non è necessario modificare il codice, ma è sufficiente integrare un nuovo *slice*.

2.2 Java

In aggiunta a Neverlang, per la realizzazione del progetto è stato utilizzato il linguaggio di programmazione Java. Java è un linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica, sviluppato da Sun Microsystems nel 1991. È molto diffuso e ben supportato, con una vasta comunità di sviluppatori e una grande quantità di librerie. Uno degli obiettivi principali di Java è quello di essere il più possibile indipendente dalla piattaforma di esecuzione, permettendo di scrivere una volta il codice e farlo eseguire su qualsiasi Java Virtual Machine (JVM), indipendentemente dall'architettura del computer [IBM].

Java è stato utilizzato per la realizzazione del progetto in quanto Neverlang è sviluppato per essere completamente integrato con esso. Il suo compilatore, nlgc, è stato sviluppato per poter convertire il codice scritto utilizzando il DSL di Neverlang, generando un nuovo codice supportato dalla JVM. Inoltre, Neverlang

permette di utilizzare Java (ma non solo; anche Scala, ad esempio, è supportato) come linguaggio per la definizione della semantica all'interno dei moduli del DSL. Ciò è possibile in quanto gli accessi a variabili non-terminali, definiti all'interno della sintassi, sono sostituiti dallo specifico plug-in con accessi alla reale rappresentazione interna del linguaggio. In particolare, l'accesso alle variabili viene effettuato tramite una chiamata all'n-esimo figlio dell'Abstract Syntax Tree (AST) [CV13].

Capitolo 3

Requisiti

Capitolo 4

Design e Implementazione

Capitolo 5

Validazione e Conclusioni

Bibliografia

[Caz] Walter Cazzola. Neverlang 2: Language workbench. Ultimo accesso: 23 Agosto 2024.

[Caz12] Walter Cazzola. Domain-specific languages in few steps. In Thomas Gschwind, Flavio De Paoli, Volker Gruhn, and Matthias Book, editors, *Software Composition*, pages 162–177, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[CV13] Walter Cazzola and Edoardo Vacchi. Neverlang 2 – componentised language development for the jvm. In Walter Binder, Eric Bodden, and Welf Löwe, editors, *Software Composition*, pages 17–32, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[IBM] IBM. What is java? Ultimo accesso: 23 Agosto 2024.

[VC15] Edoardo Vacchi and Walter Cazzola. Neverlang: A framework for feature-oriented language development. *Computer Languages Systems & Structures*, 43:1–40, 2015.

Acknowledgements

Optional. Max 1 page.