Conceptual Modeling Language Specification

Version 1.0

Quenio Cesar Machado dos Santos

Universidade Federal de Santa Catarina[*]

July 2017

[*] Initially developed as part of the author's Bachelor Technical Report in Computer Sciences

Contents

List of Figures

List of Tables

One

Introduction

Two

Concepts

2.1 Properties

2.2 Inheritance

```
conceptDeclaration returns [Concept concept]:
    ABSTRACT? 'concept' NAME
    (':' ancestorList)?
    (';' | propertyList);

ancestorList:
    NAME (',' NAME)*;

ABSTRACT:
    'abstract';
```

Figure 2.1: Concept Declaration Syntax

```
propertyList:
    ’{’ (propertyDeclaration ’;’)* ’}’;

propertyDeclaration returns [Property property]:
    NAME (’:’ typeDeclaration)? (’=’ STRING)?;
```

Figure 2.2: Properties Declaration Syntax

Three

Associations

3.1 Unidirectional Associations

3.2 Bidirectional Associations

3.3 Collections

Four

Values

4.1 Literals

4.2 Primitive Types

```
STRING:
    ’"’ .*? ’"’;
```

Figure 4.1: Literals Lexical Structure

Five

Expressions

Six

Targets

Seven

Modules and Libraries

A

Concrete Syntax (Grammar)

A.1 ANTLR Grammar

```
// Compilation Units:


compilationUnit returns [Model model]:
    declarations*;

declarations:
    conceptDeclaration | targetDeclaration;


// Concept Declarations:


conceptDeclaration returns [Concept concept]:
    ABSTRACT? 'concept' NAME
    (':' ancestorList)?
    (';' | propertyList);

ancestorList:
    NAME (',' NAME)*;

ABSTRACT:
    'abstract';


// Property Declarations:


propertyList:
    '{' (propertyDeclaration ';')* '}';

propertyDeclaration returns [Property property]:
    NAME (':' typeDeclaration)? ('=' STRING)?;
```

```
// Type Declarations:


typeDeclaration returns [Type type]:
    NAME CARDINALITY?;

CARDINALITY:
    (’?’ | ’*’);



// Target Declarations:


targetDeclaration returns [Target target]:
    ’target’ NAME propertyList;



// Names:


NAME:
    (’A’..’Z’ | ’a’..’z’)
    ( ’A’..’Z’ | ’a’..’z’ | ’0’..’9’ | ’_’ )*;


// Literals:


STRING:
    ’"’ .*? ’"’;



// Ignoring Whitespace:

WS:
    ( ’ ’ | ’\t’ | ’\f’ | ’\n’ | ’\r’ )+ -> skip;
```

```
// Ignoring Comments:

COMMENT:
    (’//’ .*? ’\n’ | ’(*’ .*? ’*)’ ) -> skip;
```

B

Abstract Syntax (Metamodel)

C

Abstract Syntax Tree (Instantiation)