Constructed Language Generator

Version 0.0

Robert D. Schultz

# Abstract

This document describes the functions of a constructed language (conlang ) generator.

# Table of Contents

[1 Abstract 2](#_Toc439334283)

[2 Table of Contents 3](#_Toc439334284)

[3 Introduction 5](#_Toc439334285)

[3.1 Purpose 5](#_Toc439334286)

[3.2 Scope 5](#_Toc439334287)

[3.3 References 5](#_Toc439334288)

[3.4 Overview 5](#_Toc439334289)

[4 Glossary 5](#_Toc439334290)

[4.1 Definitions 5](#_Toc439334291)

[4.2 Abbreviations, Acronyms, and Initialisms 6](#_Toc439334292)

[5 Use Cases 6](#_Toc439334293)

[5.1 Actors 6](#_Toc439334294)

[5.1.1 User 6](#_Toc439334295)

[5.1.2 Various “generators” 6](#_Toc439334296)

[5.2 List of Use Cases 7](#_Toc439334297)

[5.2.1 User Use Cases 7](#_Toc439334298)

[5.3 Use Case Diagrams 7](#_Toc439334299)

[5.3.1 User Use Case Diagram 7](#_Toc439334300)

[5.4 7](#_Toc439334301)

[5.5 Use Cases 8](#_Toc439334302)

[5.5.1 Create New Constructed Language 8](#_Toc439334303)

[6 Design Overview 9](#_Toc439334304)

[6.1 Introduction 9](#_Toc439334305)

[6.2 System Architecture 9](#_Toc439334306)

[6.3 System Interfaces 9](#_Toc439334307)

[6.3.1 User Interface 9](#_Toc439334308)

[6.4 Constraints and Assumptions 9](#_Toc439334309)

[7 System Object Model 9](#_Toc439334310)

[7.1 Introduction 9](#_Toc439334311)

[7.2 Subsystems 9](#_Toc439334312)

[7.2.1 Phone Generator 9](#_Toc439334313)

[7.2.2 Phoneme Generator 9](#_Toc439334314)

[7.2.3 Grapheme Generator 9](#_Toc439334315)

[7.2.4 Phonotactics Generator 9](#_Toc439334316)

[7.2.5 Syllable Generator 9](#_Toc439334317)

[7.2.6 Morpheme Generator 9](#_Toc439334318)

[7.2.7 Grammar Generator 9](#_Toc439334319)

[7.2.8 Inflection Generator 9](#_Toc439334320)

[7.2.9 Word Generator 9](#_Toc439334321)

[7.2.10 Dictionary Output Generator 9](#_Toc439334322)

[7.2.11 Various Rule Set Output Generators 9](#_Toc439334323)

[7.3 Subsystems Interface 9](#_Toc439334324)

[8 Object Descriptions 9](#_Toc439334325)

[8.1 Objects 10](#_Toc439334326)

[8.1.1 Phone Generator 10](#_Toc439334327)

[9 Object Collaboration 10](#_Toc439334328)

[9.1 Object Collaboration Diagram 10](#_Toc439334329)

[10 Data Design 10](#_Toc439334330)

[10.1 Entity Relationship Design 10](#_Toc439334331)

[11 Dynamic Model 10](#_Toc439334332)

[11.1 Sequence Diagrams 10](#_Toc439334333)

[11.2 State Diagrams 11](#_Toc439334334)

[12 Non-Functional Requirements 11](#_Toc439334335)

[12.1 Performance Requirements 11](#_Toc439334336)

[12.2 Design Constraints 11](#_Toc439334337)

# Introduction

## Purpose

The purpose of this software is purely recreational.

## Scope

At the very least this program should generate a bilingual dictionary where the two languages are English and the constructed language.

Other things it may generate:

* A description of the phones used, as well as allophone rules
* Description and list of morphemes
* Character set in .ttf format
* Rules for writing out the language
* A language development tree
* Language families
* Song and poems

The interface and user interactivity will be very limited. There will be a settings file that can be tweaked, but besides that the only interaction will be the “generate language” button.

## References

## Overview

The program will pick phones (represented by IPA symbols) and assign them to phonemes. The program will then pick letters and assigned them to graphemes. Next, it will pair the phonemes and graphemes (the pairing will be referred to as a phoneme-graphemes).

In the syllable generating system, the program will first create phonotactic rules. Then it will follow those rules when assigning phoneme-graphemes to the onset, nucleus, and coda positions of a syllable.

In the morpheme generating system, the program will decide the number of syllables in the morpheme and the position of syllable stress(es).

The program will analyze an external network of concepts (referred to as “meanings”) and decide for which it will generate root words (morphemes). The remaining “meanings” will be based on these root words.

A grammatical system will be generated. If applicable, inflections will be generated to represent this system. Particles may also be generated to express the grammatical categories that the inflections do not.

A dictionary will then be compiled with the English “meaning”, the conlang word, and conlang IPA.

# Glossary

## Definitions

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Constructed Language | A language whose phonology, grammar, and vocabulary have been consciously devised for human communication, as opposed to having developed naturally. Also known as a “conlang”. |
| Phone |  |
| Consonant |  |
| Vowel |  |
| Place of Articulation |  |
| Manner of Articulation |  |
| Stricture |  |
| Sibilance |  |
| Phonation |  |
| Phoneme |  |
| Allophone |  |
| Letter |  |
| Grapheme |  |
| Phonotactics |  |
| Syllable |  |
| Onset |  |
| Nucleus |  |
| Coda |  |
| Word |  |
| Morpheme |  |
| Lexeme |  |
| Grammatical Category |  |

## Abbreviations, Acronyms, and Initialisms

|  |  |
| --- | --- |
|  |  |
| Conlang | Constructed language |
| CVC | Consonant-Vowel-Consonant [Syllable pattern] |
| VC | Vowel-Consonant [Syllable pattern] |
| CV | Consonant- Vowel [Syllable pattern] |
| V | Vowel (Syllable pattern) |
| IPA | International Phonetic Alphabet |
| ONC | Onset, Nucleus, and Coda |

# Use Cases

## Actors

### User

The user is any person that uses the program.

### Various “generators”

Various subsystems of the component can be described as making decisions. These “generators” can be considered actors in various use cases.

## List of Use Cases

### User Use Cases

#### Create Constructed Language

## Use Case Diagrams

### User Use Case Diagram

|  |  |  |
| --- | --- | --- |
|  |  | Command Prompt |
|  | CreateNewConLang |  |
| User | ➡➡➡➡➡➡➡➡➡➡➡➡➡➡➡ | Create New Constructed Language |

## 

## Use Cases

### Create New Constructed Language

|  |  |  |
| --- | --- | --- |
| Use Case Name:  Create New Constructed Language | ID:  CNCL | Priority:  High |
| Primary Actor:  User | Use Case Type:  Essential | Level:  Overview |
| Interested Stakeholders:  ConLangers, Fiction Writers, Computational Linguists, Linguists | | |
| Brief Description:  This use case describes the creation of a constructed language file which is a key function of the system. In this use case, the actor’s goal is to generate a constructed language file. | | |
| Goal:   * The successful completion of file generation. | | |
| Success Measurement:   * One or more outputs are generated from the constructed language file and are reviewed and found acceptable. | | |
| Precondition:   * All data required to generate a constructed language is available (list of phones, letters, etc.)   Trigger:   * User feels like generating another conlang | | |
| Relationships:  Include:   * Nothing   Exclude:   * Nothing   Depends on:   * Nothing | | |
| Typical Flow of Events:   1. User enters “CreateNewConstructedLanguage” command the command prompt. 2. The program generates the constructed language file and saves it into the program’s directory. | | |
| Assumptions:   1. It is assumed that a constructed language will be stored in a file to be accessed at a later time for other outputs (dictionaries, rule summaries, etc.). 2. It is assumed that this file will not be immediately verifiable by the user. | | |
| Implementation Constraints and Specifications: | | |

# Design Overview

## Introduction

## System Architecture

Diagram of various systems and arrows connecting them.

## System Interfaces

### User Interface

Command prompt.

Notepad to edit settings and other inputs.

## Constraints and Assumptions

Crap load of linguistic assumptions. Treating phones are discrete things is the biggest assumption.

# System Object Model

## Introduction

## Subsystems

### Phone Generator

### Phoneme Generator

### Grapheme Generator

### Phonotactics Generator

### Syllable Generator

### Morpheme Generator

### Grammar Generator

### Inflection Generator

### Word Generator

### Dictionary Output Generator

### Various Rule Set Output Generators

## Subsystems Interface

None.

# Object Descriptions

## Objects

### Phone Generator

|  |  |
| --- | --- |
| Class name: | |
| Brief Description: | |
| Attributes (fields) | Attribute Description |
|  |  |
| Program Description Language |
|  |
|  | Attribute Description |
|  |
| Program Description Language |
|  |
|  | Attribute Description |
|  |
| Program Description Language |
|  |
|  | Attribute Description |
|  |
| Program Description Language |
|  |
| Methods (operations) | Method Description |
|  |  |
| Program Description Language |
|  |
|  | Method Description |
|  |
| Program Description Language |
|  |

# Object Collaboration

## Object Collaboration Diagram

Depicts object relationships

# Data Design

## Entity Relationship Design

Shows relationship between different data types.

# Dynamic Model

## Sequence Diagrams

Step by step what each part of the program does.

## State Diagrams

Probably will be quite simple.

# Non-Functional Requirements

## Performance Requirements

* The program shouldn’t take too long when generating a language. Let’s say anything over a half hour is unacceptable.

## Design Constraints

* Try and use open source libraries.
  + The work will be licensed under \*some\* license.
* The software should adhere to locally or nationally recognized standards.
* The software’s outputs should adhere to academic linguistic conventions
  + Phonetic transcriptions should follow the International Phonetic Alphabet , <http://www.internationalphoneticalphabet.org/>.