Requirements Document: A Code in Sheep's Clothing

CS 472

Fall 2019

Johannes Freischuetz, Jackson Bizjak, Benjamin Warschauer, Fern Bostelman-Rinaldi, Thomas Croll

Team: Tom and Friends

1 Introduction

1.1 Purpose

The main goal of this project is to develop a DSL for middle schoolers to be able to define board games. Beginning programming is very difficult to relate to for many students. This DSL should be able to give students something that they can relate to as well as enjoy using. A DSL describing board games should be very relatable for most students and transfer well to another programming language. Programming is also a very difficult concept for new students to learn because it is very abstract. This DSL must be easy to use for middle schoolers and not require any advanced understanding of programming constructs.

1.2 Scope

This DSL is a very limited programming language. It will only be able to implement a very small subset of board games. The defined set is board games that can be described on a grid with no hidden information. This is because the main purpose is to develop a simple language and constructs to allow for more complicated board games would make it more complicated. There will also be limitations to the types of operations that are possible on the board. The goal of this is to make the DSL simpler at the cost of feature sets. Another goal for this project is to make an editor for the project. This would be an extension upon the language that would enhance the experience of using the language.

1.3 Definitions

1.3.1 DSL

A DSL is a domain specific language. This is a type of programming language that is only meant to do one thing and nothing more. The goal is often simplicity of use.

1.3.2 Board Games

A type of game typically played on a tabletop using pieces and boards.

2 Overall Description

2.1 Product Perspective

This DSL is very distinct from other programming languages. There are many restrictions unlike languages such as C or Python. The goal of this language is for middle school students to be able to program simple board games and nothing more. Unlike lower level languages like C++ or Python, users will not be encountering programming concepts such as memory management, classes, or pointers. This means users will not be encountering lower level errors but will restrict their ability to program things outside of board games.

This DSL is most similar to Scratch, another high-level language. Like Scratch this DSL is very restricted with the intention that the user does not encounter higher-level problems. Unlike Scratch this DSL will allow users to make errors. This is by design as it will allow the user to gain experience in debugging syntax errors. Another difference is that this language uses a text editor instead of a visual editor. Typing this DSL out will be most similar to a lower level language such as Python or C, as syntax and formatting will be important and there can be syntax errors. One main differentiator will be that the syntax errors that happen must provide very distinct errors that are easily distinguishable by the user, unlike in python or C for example where the error messages can be cryptic.

2.2 Product Functions

Our DSL will describe a set of tic-tac-toe related board games. It might be able to describe other games, but that is the only required game. It will provide users with a program structure and build in constructs that are similar to the structure and words that could be used to describe a game in english. A digital programming interface is a possible addition to our product, but this is an extensions rather than a requirement. The general idea is that it would guide students through the programming process and incorporate error handling, but we haven't really defined the functionality of the interface as it's not a part of the project yet.

2.3 User Characteristics

The board game DSL is targeted directly at middle school students who are trying to learn computer science topics. These students will be between 6th and 8th grade, around 12 or 13 years old. These students will likely have no prior knowledge of computer science or programming. They also will probably not have spent much time learning math concepts that related to programming. Additionally, students will be expected to understand general board games concepts to be able to implement them in the language. This should extend to knowledge of specific board games, such as TicTacToe or Connect4.

2.4 Constraints

Due to the nature of a DSL, heavy constraints must be put on the implementation. The language must be concise enough that it is understandable to young students. Thus, the implementation will be heavily constrained to defining board games. The language should not be useful for creating arbitrary programs, as this would make it overly complex. Additionally, constructs need to be well thought out for usability and clarity.

3 Specific Requirements

3.1 External Interfaces

If we do decide to develop an interface, it should be visually appealing, or at least not distracting, it should layout the program structure and usable constructs in a way that helps students use and understand them, and it should provide helpful error handling. These are estimates of what the requirements would be, as the interface is not currently part of the project plan. The main goal of the user interface would be increasing usability by preventing syntax errors from happening altogether. The main external interface for now will be the command line using command line arguments to pass files into the project.

3.2 Usability Requirements

Students using our language should be able to create a number of board games, including ones that already exist and ones they think of themselves. Once our language supports the basic elements of a board game, the amount and complexity of the board games supported will be unlimited, but the goal is to make complex games easy able to be specified. Additionally, different levels of board game will be supported. Very high level constructs to support early learning will be added, as well as more complex constructs to create more advanced games later.

4 Verification

4.1 External Interfaces

If we do decide to create an interface and then set requirements for it, we would verify that it's functional, nothing the user can do will break it or cause incorrect outputs. We would also test it's "usability", but unless we tested the product with middle school students, that would be mostly subjective.

4.2 Usability Requirements

One goal is to have games like tic tac toe and variations be only a few lines in our language with a proper implementation. In higher levels of difficulty, games like Go, Minesweeper, and Sudoku should be able to be implemented, albeit with a more involved process for the user. A variety of constructs including specifying board size, specifying turn order, pieces available, and moves available on a turn should be do-able for a user in just a few lines each. Additionally, many constructs should exist for aiding in creating the board game.

At a minimum, our language should support checking for a certain number of pieces in a row. Allow for different sized pieces, as well as pieces that can move differently. The language should also have built in features to check for valid moves, checking if a move is out of bounds, or if the move they specified is not valid according to the

5 Appendices

5.1 Gantt Chart

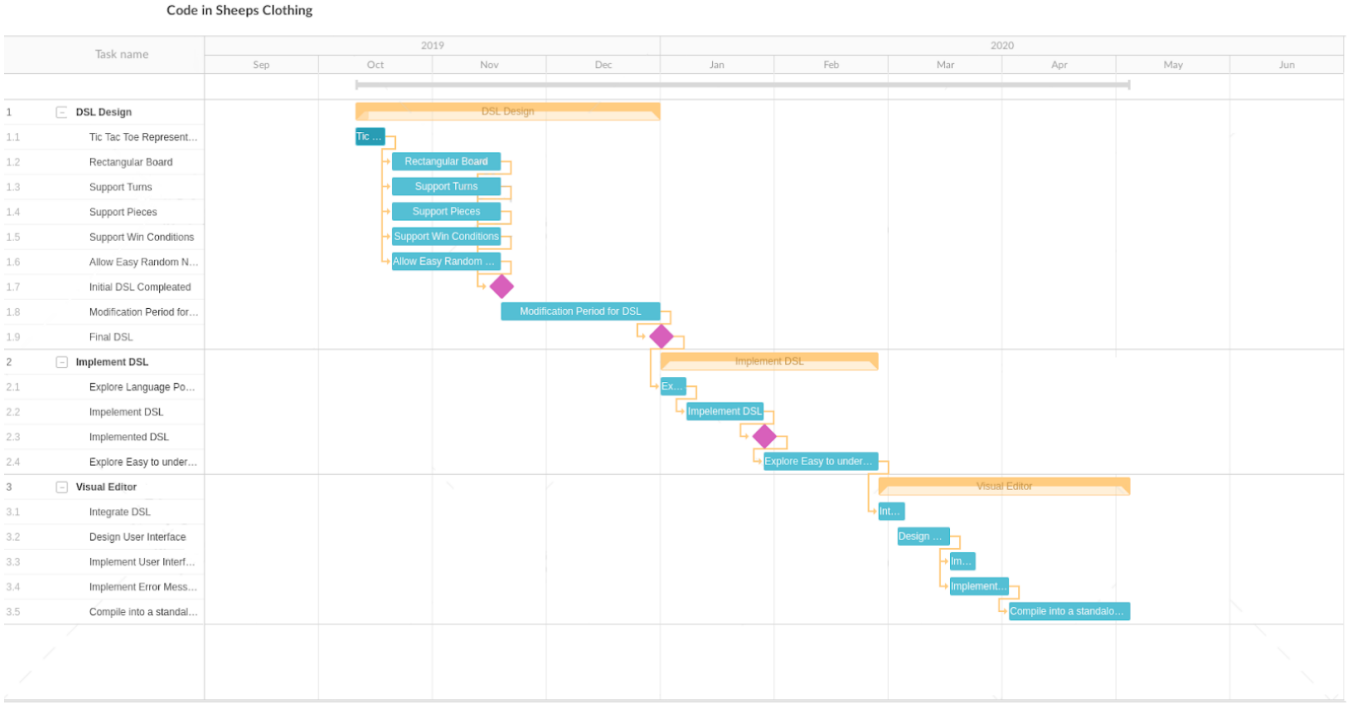

Fig. 1. Gantt Chart for the development of the DSL