CS190 FORCE Top-level Design Document

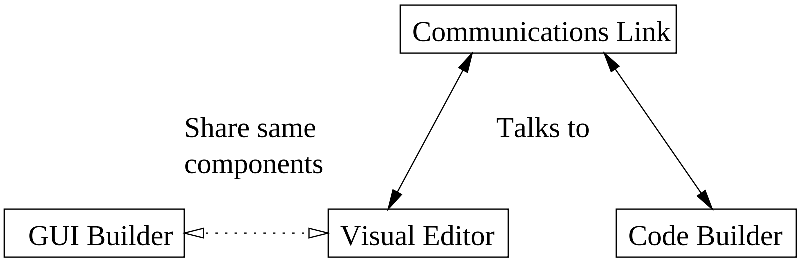
Junichi Kimura (jun)

February 24, 1999

1 Document Overview

This document is a draft of the top-level design for FORCE, a graphical programming environment designed to aid beginners and teachers.

2 Design Overview



The FORCE application consists of four main components: the GUI Builder, Visual Editor, Communications Link, and the Code Builder.

2.1 Visual Editor

The visual Editor is the most visible component to the user. This is where the user will interact with the program in creating a C code visually. The GUI of the program will live in the Visual Editor.

2.2 Code Builder

The Code Builder interacts with the Visual Editor through the Communications Link. This component is responsible for checking the syntax of the code that the user is trying to draw in the Visual Editor. The Code Builder maintains all the C code information of the user's program such as the parameter list of functions, return types, etc. and is responsible for generating the final code.

2.3 Communications Link

This component is used as a communications aid between the Visual Editor and the Code Builder. The other important role of this component is to insulate the two components from each other, keeping one's implementation hidden from the other.

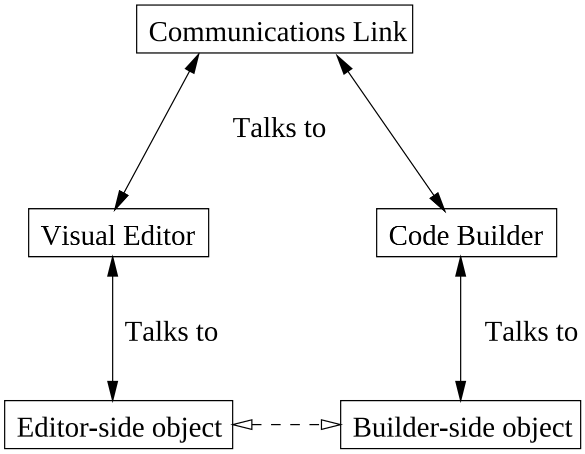
2.4 GUI Builder

This component allows the user to interact with the FORCE in creating a GUI-based program visually. It allows the user to design a GUI using a WYSIWYG GUI, and generates the code that can be edited within the Visual Editor.

3 Intercomponent Interactions

This section outlines the interactions between the four main components outlined above.

3.1 Visual Editor vs. Code Builder



These objects don't talk to each other directly. Communication is done through the Communications Link

For every node object created in the Visual Editor, its counterpart will be created in the Code Builder. The reason for this redundancy is to achieve encapsulation between the two components, and to allow for separate testing of each component without the other. This design has the following advantages:

- The Visual Editor does not have to perform checks to see if the connections that the user is trying to make is syntactically correct. The check will be done in the Code Builder's side of the object. When the user tries to make a connection between two nodes, the Visual Editor will only need to supply the nodes in question (probably done through some sort of id that the two components will share) to the Code Builder, which will then check to see if the connection is valid.
- The Code Builder side object will not be cluttered with GUI details. Likewise, the Visual Editor side object will not be cluttered with syntax and code generation details.