helloUML
Top-Level Design

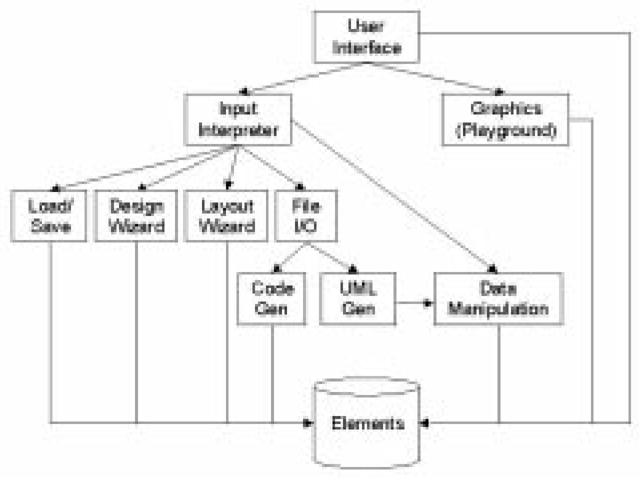
CS 190, Spring 2000
George Quievryn

Overview

HelloUML, as outlined in the requirement and specifications documents, is designed to be a design and organizational tool geared towards novice software developers. The project is designed to be implemented in a time frame of just under three months (February through May 2000) by a group of eight students. The top-level design should be constructed so as to reflect the number of programmers, the time frame, and the important theme of maintainability. Care was taken to avoid excessive dependencies and maintain modularity.

Component Diagram

The following is a depiction of the overall system modules and their interactions. Arrow depict dependencies. Individual components will be discussed in the section that follows.



Component Descriptions

- *User Interface*

  This module includes all aspects of the graphical user interface, including menus, floating windows, buttons, object lists, and object fields. This module *does not* include graphical elements and operations involved in the UML playground. This module needs to query the elements database in order to create object lists and fields. All user input is passed directly to the input processor.

- *Input Interpreter*

  This module relays all messages from the user input module to the proper action modules. The reason for keeping this module separate is that it facilitates easy GUI modification. This module has no dependencies on the user interface module – the UI module will make all calls and any information can be communicated through return values.

- *Load/Save*

  This module is responsible for constructing and destructing a persistent form of the elements database.

- *Design & Layout Wizards*

  The design wizard is responsible for design template management and loading, while the layout wizard is responsible for automatic layout reconstruction. I will leave the details of these modules to those more qualified.

- *File I/O*

  This module contains two separate divisions – code generation and UML generation (parsing). Code generation involves accessing the elements database. UML generation creates elements though the data manipulation module.

- *Data Manipulation*

  This is the module responsible for adding, removing, and changing elements of the UML playground. Information is fed from the input interpreter. This module must also check whether the intended modification is legal.

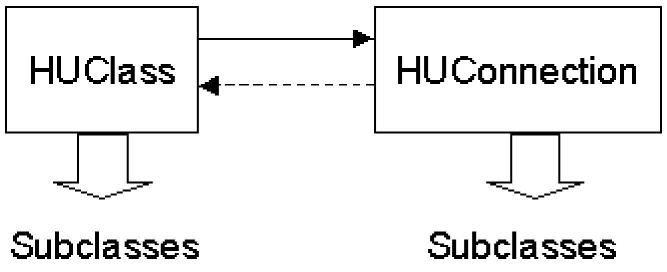
- *Graphics (UML Playground)*

  This module represents the UML canvas. It is the job of this module to extract information from the elements database and draw each visible element of the screen.

Elements Database – Data Repository

The elements database is the centralized data repository of the application. This repository is just that – a repository. The only methods of this module are constructors, destructors, and accessors. Since most other modules depend on this module, this module should be void of much functionality. This will ease testing and maintenance.

The database consists of two basic components – classes and connections. These components have the following interactions:



HUClass contains information on how to draw itself (i.e. layout position, filters, etc.), a name, a list of fields, tags, and a list of all the connections *from* the class. Connections contain a link to both classes that it connects. To avoid dependency cycles, the link from connections to classes is what Lakos calls a opaque pointer. In other words, connections store links to dumb data that can be retrieved and utilized by other modules.

Project Division – Assigning Developers

The following is my suggestions for division of coding duties:

1. User Interface & Input Interpreter
2. Elements Database
3. Code Generation
4. UML Generation
5. Layout Wizard & Graphics
6. Design Wizard
7. Load/Save
8. Data Manipulation

As far as additional project responsibilities (such as project leader, documentation, etc.), these roles can be assigned independent of coding duties. Some of the above coding segments may involve more/less coding, allowing time for other responsibilities. For example, the elements database does not seem to involve much coding and needs to be completed fairly quickly. It seems that this individual might be a candidate for project leader. The design wizard, on the other hand, might not allow much extra time.

Project Timeline

The actual dates need to be settled upon after fine-tuning of design. Instead of giving dates, I will put stages in a chronological order.

STEP 1: Finalize top-level design, project roles.
STEP 2: Complete interfaces & stubs.
STEP 3: Complete & test elements database.
STEP 4: Finalize user interface & input interpreter.
STEP 5: Finalize and test all additional components.
STEP 6: Initial integration and testing.
STEP 7: User testing
STEP 8: Finalize documentation