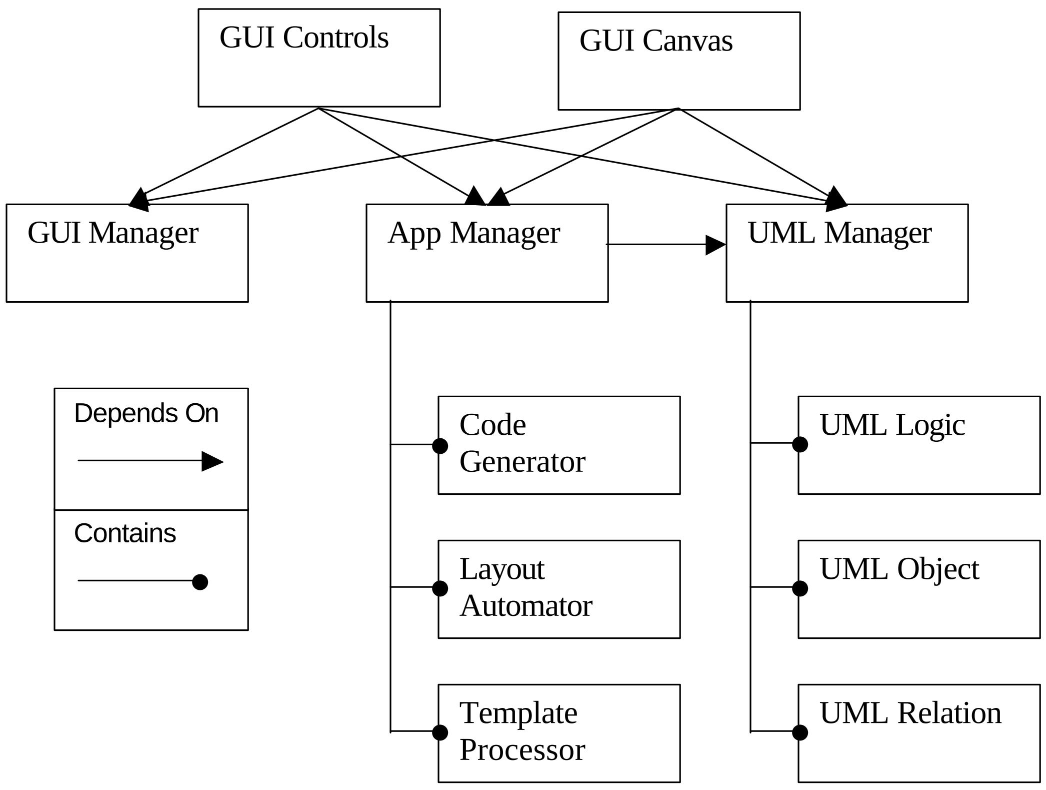
Di Wang
CS 190
Top Level Design

HelloUML

Component Diagram and Descriptions:



GUI Controls:

This component contains the various widgets that makes up the GUI:

- Pull-Down Menu: this is the standard menu that appears on the top of most applications. The exact items in the menu needs more specification, but it will include basic items for file operations (save, load, print), edit operations (undo, redo, copy, cut, paste, select all), user options (color preference, zoom in/out, font selection), and application tools (code generation, auto layout, template load)

- Toolbar: this is a row of icons that is similar in functionality to the menu, but with only the most commonly used operations included (save, load, copy, cut, paste, undo, redo, zoom, etc.) The toolbar also has toggle buttons for operations like drawing links between objects (inheritance, containment, etc.)

- Objects Display: this shows a list of objects in the diagram, clicking on the objects in this diagram will highlight the same objects on the canvas.

- Dialog Boxes: this will be the various dialog boxes required by operations such as save, load, print, and color/font preference.

The GUI Controls component will communicate with the three managers for all the functionalities that the GUI must trigger. For example, if the user clicks on save and then picks a file from the dialog box, this information will be send to the App Manager to actually do the saving. GUI Controls need to communicate with the GUI Manager in order to communicate with the GUI Canvas without introducing a cycle. Operations such as copy and paste need to trigger functionalities in both the GUI Canvas(through the GUI Manager) and the UML manager.

GUI Canvas:

This component contains the drawing area where the UML diagram is represented graphically and where the user can directly manipulate the UML objects. The functionalities include select objects, add/delete objects, context sensitive menus, modify/move/resize objects, and line drawing (to represent relationships). This component has similar relationships as the GUI Controls component.

GUI Manager:

This component serves as a middleman between the Canvas and the Controls. It saves the variables that are shared by both objects, and it has a way of relying messages between the Controls and the Canvas and vise versa. This can be accomplished with the signal/slot paradigm of the Qt toolkit.

App Manager:

This component controls all the application related functionalities, it contains three main modules deserving of their own attention:

- **Code Generator**: This module takes the UML model, do the necessary integrity tests, and then generates C++ or Java code.

- **Layout Automator**: This takes the UML model and rearranges the layout of the objects.

- **Template Processor**: This loads pre-defined templates into the program and processes them.

The App Manager takes input from the GUI modules and depends on the UML Manager to provide the UML model for functionalities such as code generation.

UML Manager:

This component controls the internal representation of the UML model, it does not have any functionalities except for the storage of data. (Basically, this is the main data structure of the program). It contains three major subcomponents:

- **UML Logic**: This module holds the rules for what can and cannot be done in UML, other components, such as the code generator, need to access the rules to see if certain operations are legal.

- **UML Object**: This module is a data structure that represents the object(class or interface). It includes information such as name, type, and any user-defined fields.

- **UML Relation**: This module is a data structure that represents a relationship between two objects. It includes information such as which two objects and the type of their relationship.

External dependencies:

There are no external programs (such as databases) that this project will need to depend on during runtime, but there are several libraries that will be needed, including Standard Template Library and the Qt Toolkit.

Task Breakdown:

Project Manager:
The project manager is the ultimate authority in the project and is also the person responsible for the entire project. Her main responsibilities are:
- Communication Coordination: Make sure people are on the same page, schedule meetings and setting agendas.
- Dispute Resolution: Any dispute that cannot be resolved below will be decided by her.
- Progress Monitor: Make sure progress is made and that everyone is doing their job.

Communication skills are absolutely essential for this role.

Program Architect:
The program architect is responsible for the integrity of the entire design, she must understand each and every single part of the design, make sure each part works together, and answer any questions if others have misunderstandings. If a design flaw is discovered, it is her responsibility to find a fix. The program architect is also expected to code in a supporting role.

GUI Architect:
The GUI architect specifies the look and feel of the GUI. (how the widgets are laid out, which dialog boxes are needed and what they look like, etc). It is also her responsibility to design and implement this GUI. This probably requires heavy amount of work and requires a deep understanding of the Qt toolkit.

Lead Coder:
The lead coder is responsible for the implementation of all the non-GUI functionalities of the program.

Support Coders(2):
The support coders helps the lead coder and the GUI architect in whatever coding capacity that's needed.

Librarian:
The librarian is responsible for several non-organizational and non-coding tasks:
- Internal Documentation (Specification): Make sure that the spec doc and design docs are updated with any the changes along the way.
- External Documentation (Manuals): Design and write a user manual.
- Source Control: Make sure that the source control software is setup correctly and that everyone is using it correctly. Should also be able to educate other group members on how to use the source control software.
- Makefile: Write and maintain the makefile(s).

Testing Coordinator:
Design and carry out stress tests on the various integrated product. Coordinate outside users to do bug and functionality tests and report any comments or requests to the project manager.

Schedule:

This is a schedule based on the assumption that we have 7 weeks. (3/6/2000 to 5/3/2000)

Week 1:
- Finalize specification
- Assign jobs
- Finalize top-level design

Week 2:
- Component designs (architect and coders)
- Rough sketch of the GUI (GUI architect)
- Source control and makefile is setup (librarian)

Week 3:
- Finalize Component designs (architect)
- Finalize the GUI look and feel (GUI architect)
- Learn as much as possible the necessary algorithms and libraries needed for this project, including writing small programs to test out ideas. (coders)
- Maintain Specification. (Librarian)

Week 4:
- Code the interface between modules (architect and coders)
- The GUI should start to take form, as least the very basic functions should be there for the coders to use.
- First Integration. Does not have functionality, but the program should run as expected and the flow of control is clear (through print statements in the code)
- Adjustment of schedule if necessary. (manager)

Week 5:
- Code functionalities (coders)
- GUI should be mostly complete. (GUI architect)
- 2nd Integration, the basic functionalities should now be there.
- Outside user tests need to be conducted. (tester)
- Progress review, final readjustment of the specification and required functionality. (manager)

Week 6:
- Code all functionalities (coders)
- GUI should be completed. (GUI architect)
- User manual should be completed. (librarian)
- 3rd Integration, most of the functionalities should be there.
- Outside user tests need to be conducted. (tester)

Week 7:
- All the bugs need to be fixed. (coders, GUI architect)
- Final integration and code freeze.
- Presentation prepared. (librarian, manager)

Week 8: (presentation week)
- fix remaining bugs
- Cleanup
- Package product
- Presentation

Other:

This top-level design is NOT complete, since there are two major components missing: the networking and the design wizard. The networking, as said in the specs, is probably not going to be part of this project. As for the design wizard, I am simply confused as to what it does. I didn't make too many assumptions about the specifications, simply because they don't really affect the design at this stage. However, as we do the more detailed components design later, these assumptions must be dealt with and put in writing. Lastly, I did not take the liberty to assign people to tasks yet, since I feel that is much more appropriate during a group meeting when everyone can voice their opinions/desires/concerns/etc.