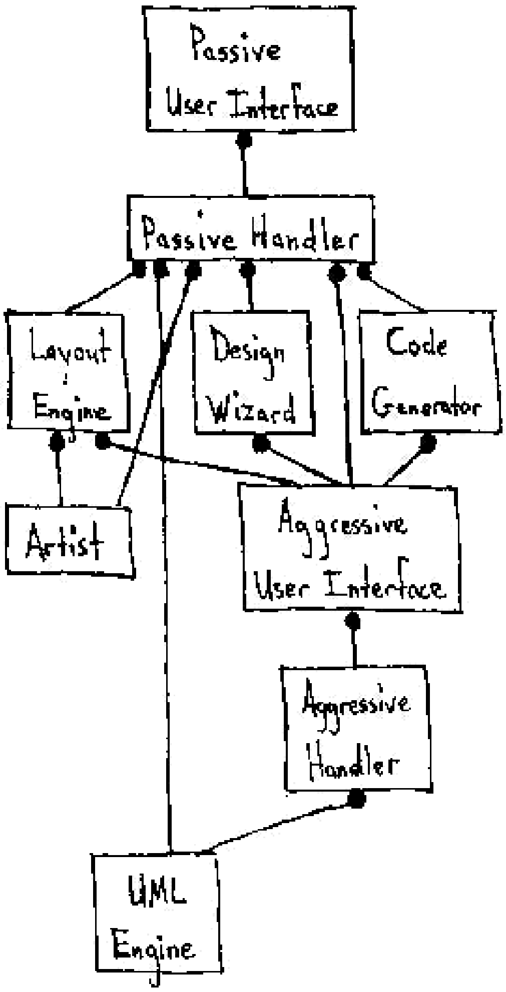
Levelized High-Level Component Diagram



Description of Components

- **Passive User Interface**

  the Passive User Interface is the application shell with which the user is presented upon startup and with which the user interacts for the duration of the helloUML session. this is the component in which standard UI features such as buttons and menus, as well as the playground for the UML engine, are defined.

- **Passive Handler**

  the Passive Handler is the event interpreter for the Passive User Interface. here all user actions are handled and delegated to the next relevant component in the hierarchy. the Passive Handler can communicate with the Artist component to directly update onscreen activity.

- **Layout Engine**

  the Layout Engine is responsible for basically performing onscreen pretty-print functionality for the playground. this communicates with the Artist component in order to visualize its calculated best fit for the UML model.

- **Design Wizard**

  the Design Wizard contains artificial intelligence designed to assist the user with well-codified design decisions. this is where the template repository will reside, for instance.

- **Code Generator**

  the Code Generator contains the knowledge of the programming languages supported by helloUML. it will need to communicate with the file system and may contain an inline editor/browser for the source code it generates.

- **Artist**

  the Artist is the graphics engine for the program. this is where the significant drawing actions of the playground are generated. the functionality here is different from that of merely popping up menus and such. this is more the customized graphics package for the actual modeling actions supported.

- **Aggressive User Interface**

  each of the Passive Handler, the Layout Engine, the Design Wizard, and the Code Generator talk to the Aggressive User Interface. the Aggressive User Interface is the secondary portion of the user interface responsible for true interface interactivity. this is the source from which all dialog boxes and wizard features should spring. the Passive User Interface has all of its data encoded and merely requires activation by the user. the Aggressive User Interface asks for information from the user.

- **Aggressive Handler**

  correspondingly, there is a handler for the secondary user interface. it is placed lower in the hierarchy because it should need to communicate with fewer of the components.

- **UML Engine**

  the UML Engine is where the guts of the application reside. this is where all knowledge of UML is kept, where all state is stored, and on which the core functionality of the application depends. each of the handlers talk to this component in order to determine the flow of control.

External Dependencies

the external dependencies i foresee are a successful (windowed) operating and file system (presumably Solaris and the associated file system and departmental resources such as disk space), a network (if we reach that point in our implementation), and possibly the existence of text editors/browsers for integration with code generation.

Task Breakdown

- project management (which really encompasses managing everything below).
- refinement and continual assessment of:
  - schedule.
  - design.
  - task assignment (i.e. personnel assessment in terms of personal and project-wide productivity and satisfaction)
  - documentation.
- establishment of development environment (this includes configuration management, source control, and coding conventions).
- tool building.
- coding.
- testing and debugging.

Group Organization

- **Project Leader**

  provides management skills and keeps the entire project on track with the original specifications and project goals in mind.

- **Project Architect**

  provides all important design decisions and interface choices.

- **Project Tester**

  in charge of developing the testing methodolgies and implementing them for the project.

- **Project Support**

  in charge of development environment decisions as well as providing all necessary utilities (in-house and third-party) for successful implementation of the project.

- **Project Journalist**

  responsible for maintaining all documentation as well as ensuring that all project tracking details are well-maintained.

- **Project Developer**

  based on the decisions of the Project Manager and Project Architect, takes a chunk of the project specification and begins implementing.

Schedule

1. 10 March, 2000

   the group organization should be well-established by now.

   also, by this point, as according to the course schedule, much more of the project design should be worked out. each of the components specified above should be further broken down (into classes, preferably).

2. 15 March, 2000

   the initial phases of documentation should be underway. right now, the Project Leader, Project Architect, and Project Journalist will probably be the busiest members of the team. the Project Leader and Project Architect should be working strenuosly to complete the interface design while listening to feedback from the entire team. the Project Journalist should be doing his/her best to record all activity worth recording (especially documents produced during this period).

3. 20 March, 2000

   by now, interfaces should be in place. the Project Architect should be working with Project Developers to establish component interactions.

   the Project Support should be looking for design tools (perhaps something like helloUML :).

4. 24 March, 2000

   as soon as the interfaces are complete, the Project Architect should be working closely with Project Developers to complete the design. this will probably be the point at which the intuitive divisible areas of the project become formalized.

5. 5 April, 2000

   as soon as the design is complete, the Project Support should establish exactly how the development environment is going to work. s/he should begin thinking about what tools will best aid in further project development.

   the design should now be divided among the Project Developers for implementation. serious development should be underway at this point. it is the responsibility of the Project Leader and Project Architect to have divided the project such that the development effort can be as highly parallelized as possible.

as soon as serious development is underway, the Project Tester should be establishing how and when components can be tested. as the development process continues, the Project Tester will need to work with Project Developers in order to test their components actively and revise whatever is necessary in order to keep inter-component dependencies at a minimum and productivity at a maximum. the Project Leader will probably have a hand in all this.

during this period of development, the UML Engine should be worked on at the same time as the various user interfaces. if all goes well, the handlers should go hand in hand with the interfaces. it will probably be a good idea to go ahead and integrate these components before work on the extra components such as the Layout Engine, Design Wizard, and Code Generator are touched. once the handlers and interfaces are integrated, we should probably go ahead and tie the handlers to the UML Engine. work on the bells and whistles should commence with work on the Artist.

6. 17 April, 2000

   core elements of the functionality should be in working order in the individual components. now comes the trial of seeing if there is any interoperability. everyone other than the Project Journalist should be working hard to make the integration go smoothly. the Project Tester will be a stressball during this period as all the bugs unite and revolt.

   optimally, the basic interfaces and handlers will be up and running, and tying them to the UML Engine will be the biggest chore here. development effort on any of the other components should be redirected to this effort.

   the Project Journalist should be able to begin work on the user manual in a serious and fastidious fashion. the Project Journalist should also be hounding people (with the aid of the Project Leader) to comment their code.

7. 28 April, 2000

   code. code. code.

   now the Artist should be coming into place based on the functionality of the handlers. this development should be pretty natural to integrate. the bells and whistles should be under development, now, too.

   i think it is unrealistic to think that we can shoot for integration of the entire project at the previous checkpoint. i think integration of the core is a more reasonable goal, and this checkpoint should be used for a sort of hybrid development/integration as extra features fall into place as the additional components are developed.

   and the rest shall be history...

8. 12 May, 2000

   the Project Journalist should put the finishing editorial touches on the user manual. a final version of all of the work of the past few months as it can be told in documents should be available for the world to see.

our story will have been told...

Assumptions

i assumed that nothing was set in stone and that the low priority associated with the networking aspect of this project meant that i did not need to take such functionality into account for my top-level design (although cursory thought on the matter makes me think that adding such functionality would not be too difficult based on my component hierarchy).