# Edith

## Visual Editor

Graham Baker, Walker Bohannan, Steve Marx, Vikram Nilakantan,
Jessica Penick and Eli Spiegel

University of Puget Sound

# 1 Architecture

## 1.1 Model-View Controller

The architecture best for the visual editor module of Edith is the model-view-controller. This architecture, by splitting up the view controller and model into separate modules, allows us to separately tackle the issues of the draggable interface and the core structure of the editor. For example, a typical editor structure is a textbox where the programmer might directly edit code, but given that code has special syntax, we could cut the middle man so to speak and provide the structures for the language with editable parts. This model also allows us to later adjust the interface to meet non-functional requirements like easy to use.

## 1.2 Alternatives

Alternative architectures considered were the layered model and the pipe and filter model. The layered model also puts space between the interface and the core of the editor (the infrastructure layer) but the fact that each layer relies only on the layer beneath it is a kind of rigidity that will most likely hinder us later. Modifying the core editor module at later points in development more complicated than if each module was independent. The second alternative considered was the pipe and filter model, simply because it was a good structure for work flow. The model was rejected, however, because the scope of our module at this point is currently much narrower than document workflow. Our goal is to make a simple code editor and work our way into higher complexity from there. It is possible that we might later incorporate elements from these other models.

# 2 Reflections and Status Update

## 2.1 Reflections on Development Process

Developing a clear understanding of the Visual Editors role in the overall Edith project has been the major challenge. Of course, it is easy to say that understanding the VE modules role and its interaction with other modules should have been an easy first step. However, even with somewhat detailed requirements specifications, some of the language can be interpreted in different ways, resulting in miscommunication.

On October 16 (two days before the submission deadline for this Intermediate Report), we learned that the VE team and another team understood their modules roles to be nearly identical - almost as if one of the teams thought it was the other team. Most of the communication during development of the Requirements Specification was within the VE team or between the VE team and the professor. Although the VE team communicated with other teams during work periods, the interactions were limited to a few questions and answers

while teams figured out what their modules would do. In retrospect, a more thorough review between teams whose modules interact would have been beneficial earlier in the design process. Communication through the professor, as a third party and project client/designer, was important. Some roles (e.g., method creation) for the final project remain unclear, but we are now more confident in the direction of VE development.

It seems that, early in the project, the team can most effectively tackle theoretical issues and make decisions on the development direction when meeting as a whole group. As the project progresses, it may be beneficial to go through one segment of the work together as a team, particularly in cases where it could be used as a template. As a small example, we created one or two use cases together before dividing the use cases among team members. Coordination takes an investment of time up front, but can save later re-work and promote consistency. Of course, as programming work increases, it will be divided among team members equitably (and considering interests and strengths).

In programming, as with anything, things take longer than expected. With a group of six people, it will be important to allow time for team member review and interaction as we approach milestones and deadlines. This is some team members first experience writing software as part of a group (and being part of such a large software project). Thus, some learning through trial and error is to be expected.

### 2.2 Current Status Of Implementation

A thorough UML class diagram (included) has been developed. Also, a mock-up of the Visual Editors user interface has been created, and the team will begin building toward that design. Given the confusion on the VE modules role and its interaction with other modules (as described above), progress toward concrete build steps on the editor has been limited. Now that the team has additional clarity, the build pace will need to increase to provide the first demonstration on November 1.

### 2.3 Team Member Responsibilities and Contributions

We have done a good job of distributing work equitably and working together so far. At this point, team members do not have assigned roles (e.g. GUI design, JSON specialist, etc.), and our work has consisted mainly of concept development and design. Qualitatively, everyone has contributed and collaboratively worked toward the milestones (Requirements Specification and this Intermediate Report). Quantitatively, call it 1/6 each. Realistically, it is expected that not everyone will contribute exactly the same amount on each aspect of the program. However, we do expect to contribute equally in total by the end of the project. More specific roles will likely be assigned (at least for certain phases) as development pace increases.

## References

[RE1] Author: Article/Book: Other info: (date) page numbers.