Survey of software architecture V0.0

See book: http://www.aosabook.org/en/index.html

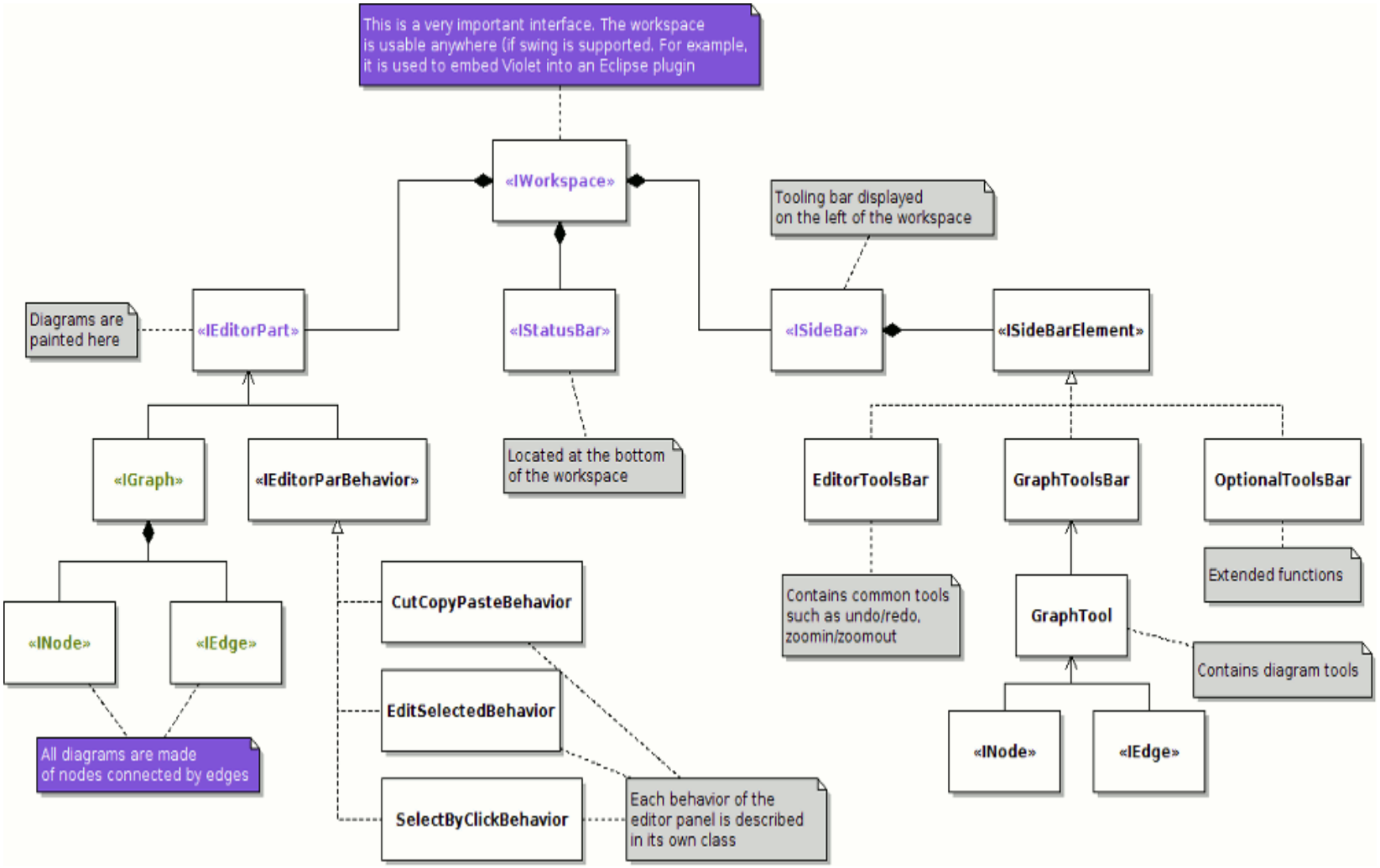
Name of system: Violet
Reviewer: David Gage
Date: 10/22/11
Author of software: Cay S. Horstmann
Author of book chapter: Cay Horstmann
Five star rating of book chapter: ***
Purpose of system: Violet is a simple, cross platform, UML editor. It's used to produce simple UML diagrams quickly and easily. It's also designed to have an easy learning curve, so that a new user can just pick it up and go.

Basic metrics

KLOC: 30
Project start-up: Around september 2006
Number of major releases: 176 revisions
Number of developers: 48
Size of user community or number of installations: 273,512 downloads over 4 years
Major stakeholders: None
Use of concurrency: None that I saw
Implementation language: Java
Supporting software: JavaBeans, Java Web Start, Java 2D, Swing

High level architecture

Diagram of software architecture



http://alexdp.free.fr/violetumleditor/WikiFiles/Image/contribute-workspace-uml-class-diagram.gif

High level scenarios

New diagram:

A user runs the Violet jar file on their computer (the difference between doing this and running it through Web Start or Eclipse is minimal). A selection screen is presented by the VioletProduct.swing component. Use case diagram is then selected by the User to start a new Workspace (VioletFramework workspace). The user can then select shapes to draw on the workspace from the sidebar. The shapes are defined in VioletFramework's product component and by plugins that use the interfaces defined there. The user can then continue to add, edit, and undo/redo these actions in the workspace. Finally the workspace can be saved by the user. This serializes the current workspace by recording the java commands necessary to create all the objects that currently exist in the workspace, handled by VioletFramework workspace.

Load diagram:

The difference from above starts with the user selecting a file to load at Violets start screen. The file is then essentially run, remember that it was saved as a series of commands to create a certain diagram, and then built by the VioletFramework workspace and displayed by the VioletProduct.swing component.

Data structures or algorithms Save and load functions make use of JavaBeans object serialization.

Control flow and/or data key to the architecture if any

Architectural style: Monolithic, it's self contained. Sort of event-driven, as far as user events are what create and change objects.

Major evolutionary changes: Added an Eclipse plugin version, but this didn't change the architecture significantly.

Performance bottlenecks: Possibly loading saved XML files and drawing general shapes on the graph, although neither of these have proven to be a problem so far.

Real time: No.

Notation for architecture: They used class diagrams to show the object inheritance architecture.

Methodology: not clear

Appendix:

Kruchten's eight context attributes applied to Brown/Wilson systems

Kruchten attributes described in his slides 17-24 of
https://files.me.com/philippe.kruchten/sbz0ma
Also in: https://files.me.com/philippe.kruchten/1q00nw

1. **Size:** S
2. **Criticality:** Lo
3. **Age of system:** M
4. **Rate of change:** Hi
5. **Business model:** open source
6. **Stable architecture:** Lo
7. **Team distribution:** Med
8. **Governance:** Med