Quill – browser based editor

Design goals

Quill – browser-based editor

- Goal – to provide a browser-based editor implementing Serenoa concepts
- HTML5 browser platform
  - Modular JavaScript for easier development
  - Graphical models with <canvas> element
  - Live communication with server via Web Sockets
- Server holds UI models persistently
  - As well as the adaptation rule engine

Architecture

- Clean separation between each abstraction layer in the Cameleon Reference Framework
- User views/updates only one layer at a time
- Quill dynamically sends to the server the changes made by the user to the visible layer
- Server-side rule engine propagates changes to other abstraction layers
- Changes sent to client to update local versions

Models, rules and visualisation

- Models held as graph of nodes and links
  - Graph mutation protocol
  - Rule conditions and actions specify mutations
    - Infer changes to neighbouring abstraction layers
    - Design agenda for tasks users have to deal with
- Automatic layout for visualization of models
  - Visualization adapts to browser window size, to changes made by user, and to changes made by server-side adaptation rules

Rule engine

- Forward chaining inference
  - Rete algorithm + further optimizations
- Existing rule engine e.g. JESS
- We define the predicates and actions the rules operate over, including context models (CARFO)
- Objects rather than strings as a basis for enabling rules at different levels of abstraction
- Mapping of rules between user editable format, internal format, and tool interchange format (RIF)

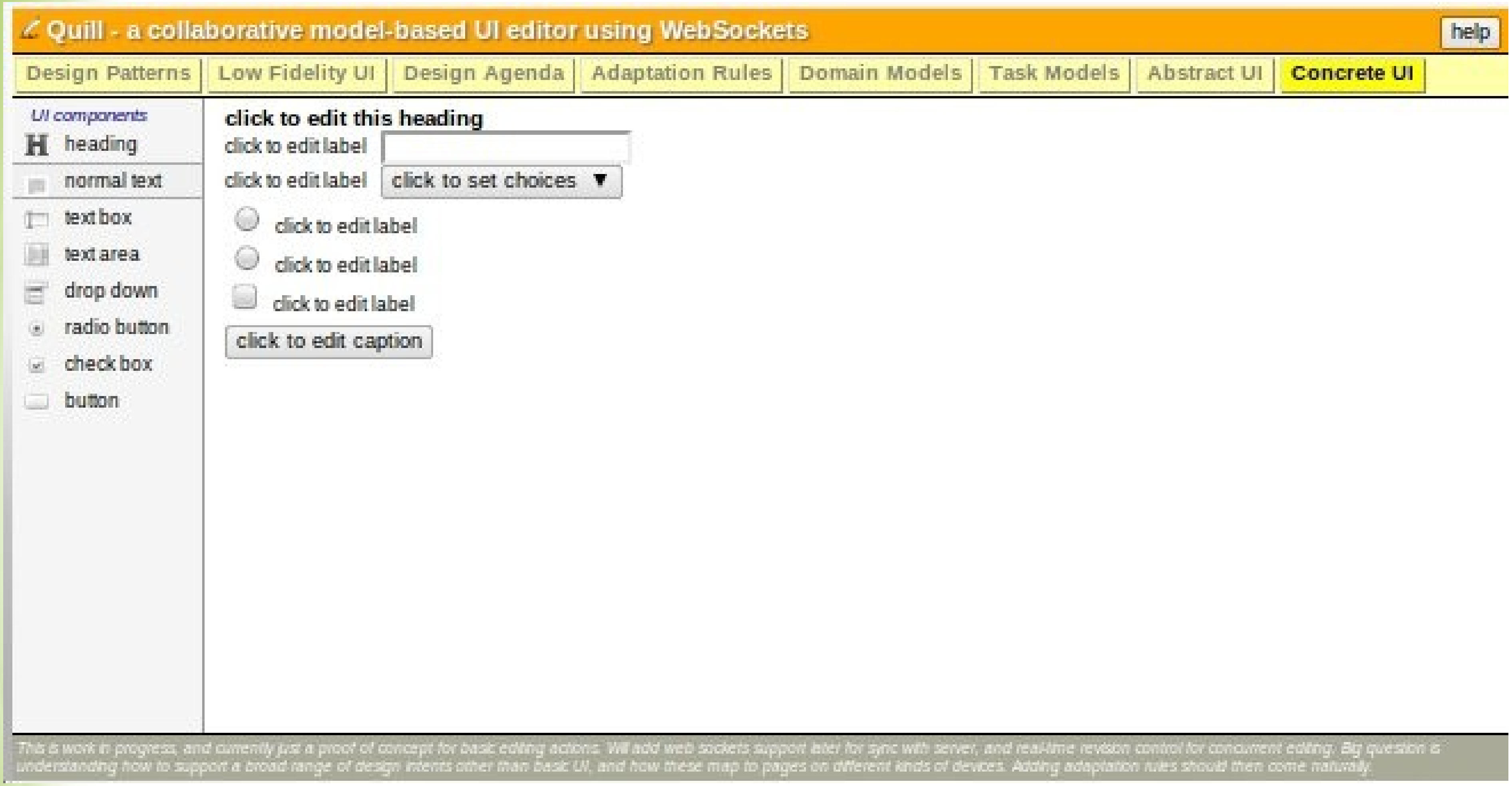
Multi-user editing sessions

- Support for distributed authoring teams
  - Team members typically playing different roles
- Dynamic version control enables multiple people to view/edit models at same time
  - “Junior” editors propose changes
  - “Senior” editor reviews changes
  - Based upon tree comparison algorithms

Quill – run-time framework

- Client dynamically coupled to server-side adaptation engine via web sockets
- Events signalling changes in context are sent to the server to trigger adaptation rules
- Changes are sent back to clients to update the user interface
- Expressed as changes to concrete UI layer
- Client-side script then works out the changes needed to the final UI

Quill – screen shot – January 2012



This is work in progress, and currently just a proof of concept for basic editing actions. Will add web sockets support later for sync with server, and real-time revision control for concurrent editing. Big question is understanding how to support a broad range of design intents other than basic UI, and how these map to pages on different kinds of devices. Adding adaptation rules should then come naturally.

See http://www.w3.org/2012/quill/

Quill – html markup

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US" lang="en-US">
<head>
<title>Quill - a collaborative model-based UI editor using WebSockets</title>
<link href="quill.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="websocket.js"></script>
<script type="text/javascript" src="quill.js"></script>
<script type="text/javascript" src="abstract.js"></script>
<script type="text/javascript" src="concrete.js"></script>
<!-- public domain quill icon by "ocal" -->
</head>
<body>
<div id="banner">
<h1><img src="quill-low.png" alt="quill icon"/> Quill -
a collaborative model-based UI editor using WebSockets
<button title="Link to documentation on how to use Quill">help</button></h1>
</div>
```

Quill – current status – your help needed for next steps

- Proof of concept for client-side UI and modularization of scripts
- Previous work on tree algorithms for distributed editing with Web Sockets and JSON encoding of mutations
- Next step is work on visualization for domain models, task models, abstract UI and improvements to concrete UI
- And work on rule engine and mutation protocol

Questions?