

Exercise 11

Richard Möhn (201311231)

Mathias Dannesbo (201206106)

February 20, 2014

1 Introduction

We used pairprogramming for all the code and “pairreporting” for the report, so we share the workload at 50% each.

In this report we present an editor, the instances of which are able to connect to each other and to capture editing actions in one editor and replay them in another *in the same textarea*. We build upon the editor from exercise 09. Which were able to connect and send edit event to each other, but not in the same textarea. Our contribution was to make two editors merge their edit in the same window.

The report describes how we change the architecture of the event flow based on operational transformation. It discusses some of the decisions we made in developing the editor. The conclusion contains a list of issues with the editor that still need to be addressed.

2 Operational transformation

3 Code Overview

Our main addition of code was in and around the `JupiterClient`-class. We changed the flow of the event to go through it. It is most easily shown on figure 2.

FiXme: Richard writes something about the techniques.

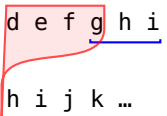
FiXme: Richard writes something.

3.1 JupiterClient

3.2 DocumentEventCapturer

3.3 Transformer

... a b c d e f g h i j k ...
... a b c h i j k ...



FiXme: Richard writes something.

FiXme: Mathias writes something.

FiXme: Richard writes something.

4 Conclusion

FiXme: Write a conclusion.

A Finding the Code and Running the Editor

The file Code1864-ex11.zip contains a Maven repository with the source code and a JAR file being the executable editor. From the root directory it can be run with ./run.sh.

Figure 1: The state space in the Jupiter algorithm.

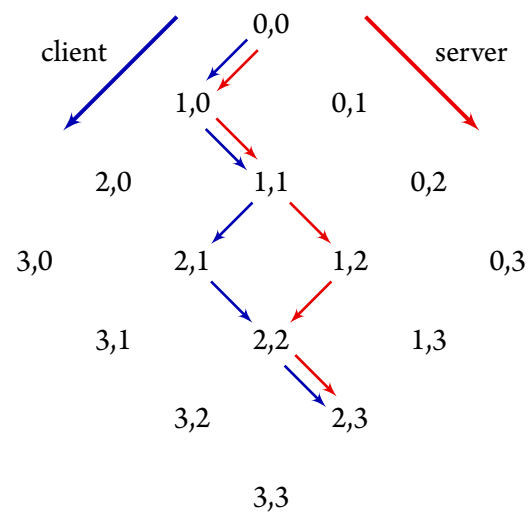


Figure 2: Event's path through the system in the editor as provided and in our version. → denote TextEditEvents and → denote JupiterEvents.

