

# Real-time Visualization of Analyzed Industrial Communication Network Traffic

Xiaoru Li, Klevia Ulqinaku, Mario Alberto Gonzalez Ordiano, Philipp Mergenthaler

SOFTWARE DESIGN AND QUALITY GROUP  
INSTITUTE FOR PROGRAM STRUCTURES AND DATA ORGANIZATION, FACULTY OF INFORMATICS



# Background

- Industrial Network Security – want to understand the traffic
- Analysis of the traffic
- Real-time visualization to help the user understand
- Incidents can be detected visually

# Requirements

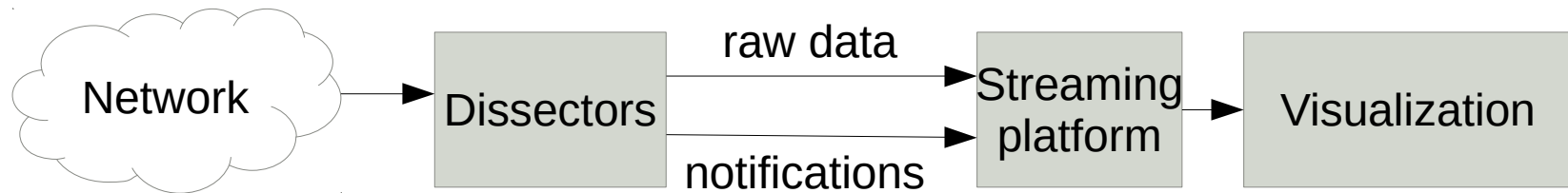
## ■ 24 Functional Requirements

- User access control, security roles
- Three different diagram types
- Brushing
- Data filter

## ■ 12 Non-Functional Requirements

# The Workflow

- Network traffic is recorded
- Traffic data is analyzed (dissected)
- Data is fed to a streaming platform (Kafka)
- A visualization tool displays data and analysis results



# Architecture and Design

## ■ Client-Server Architecture

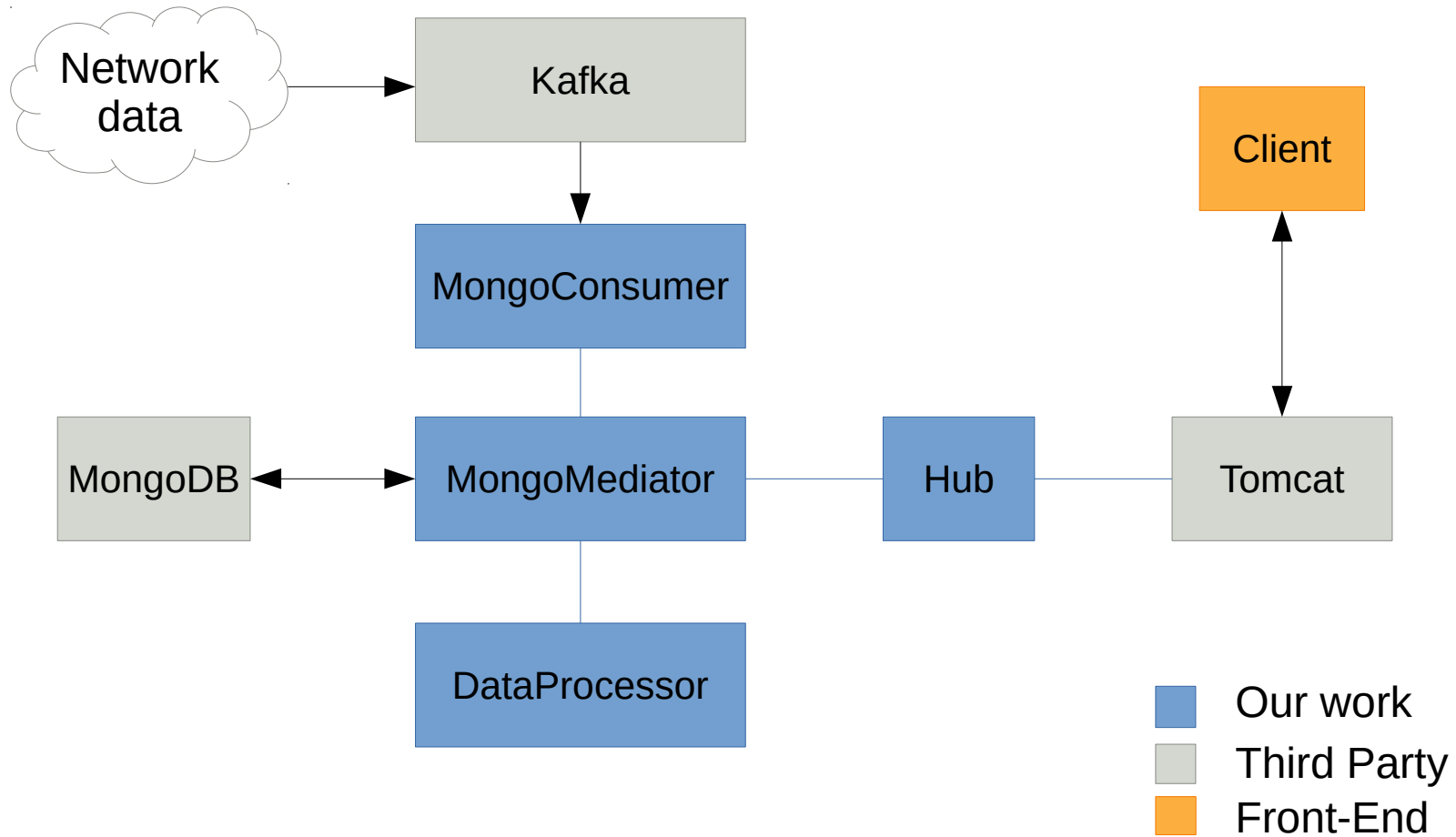
### ■ Back-End:

- Mediator pattern
- Strategy pattern

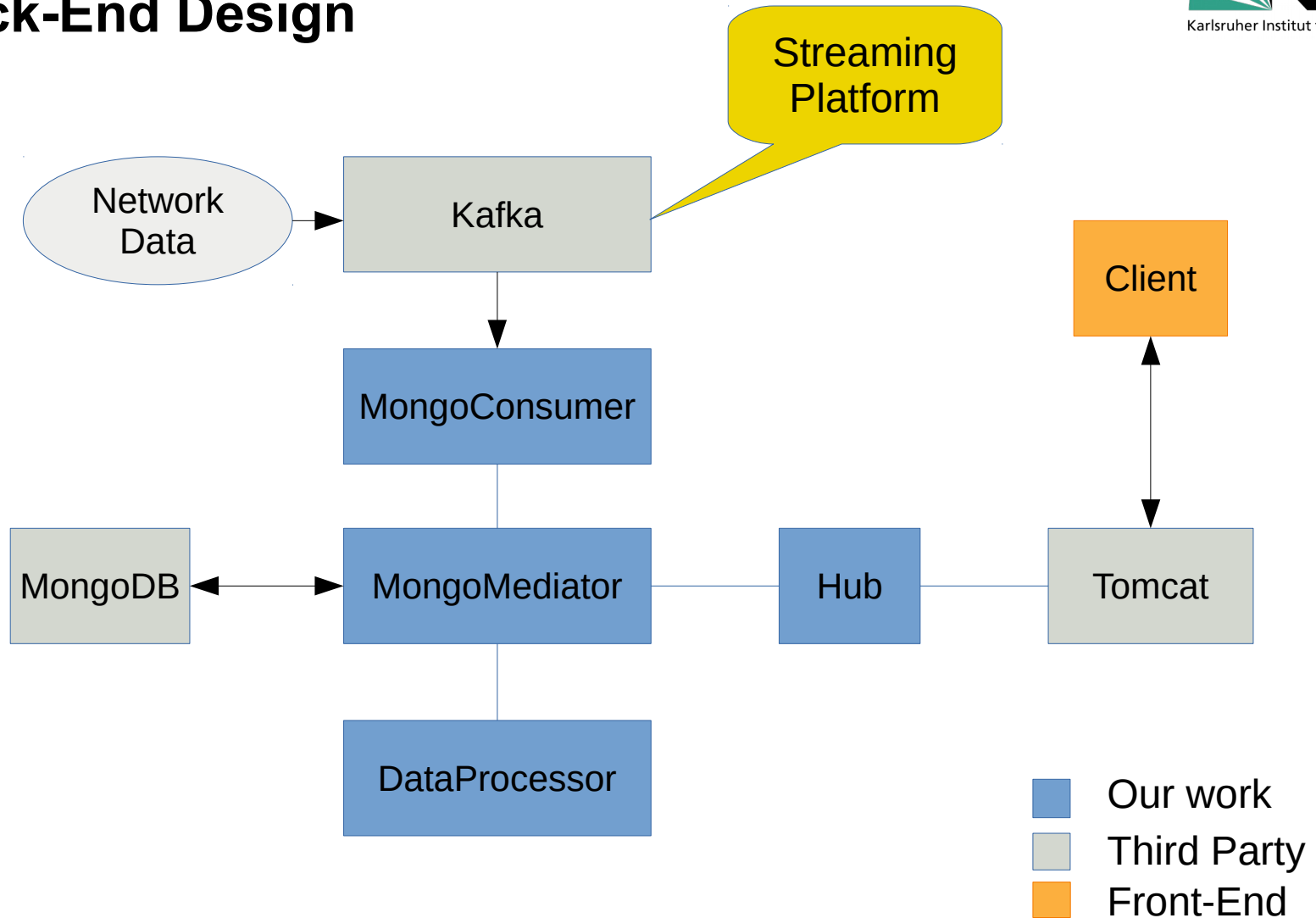
### ■ Front-End:

- Model-View-Controller
- Observer

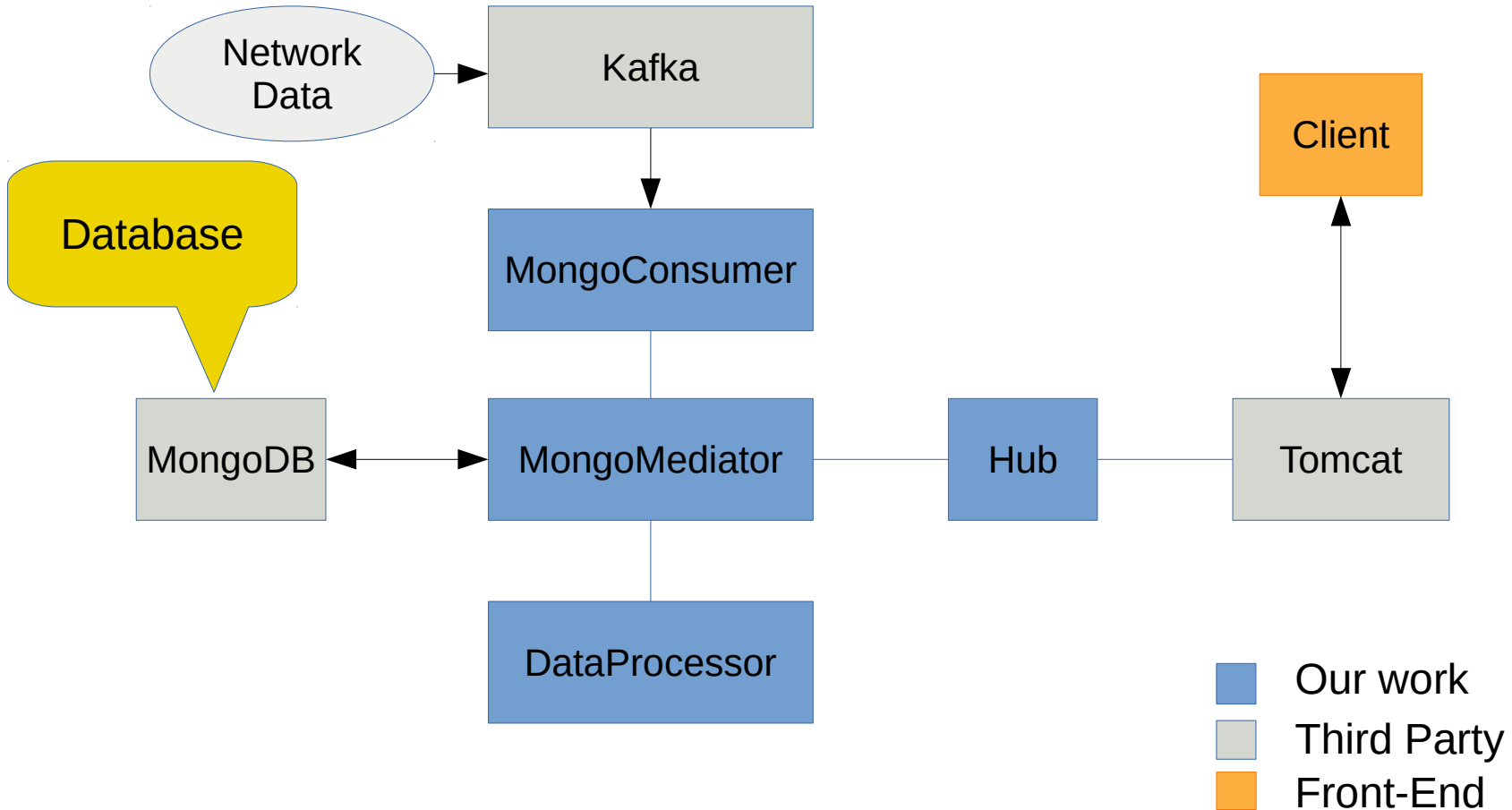
# Back-End Design



# Back-End Design

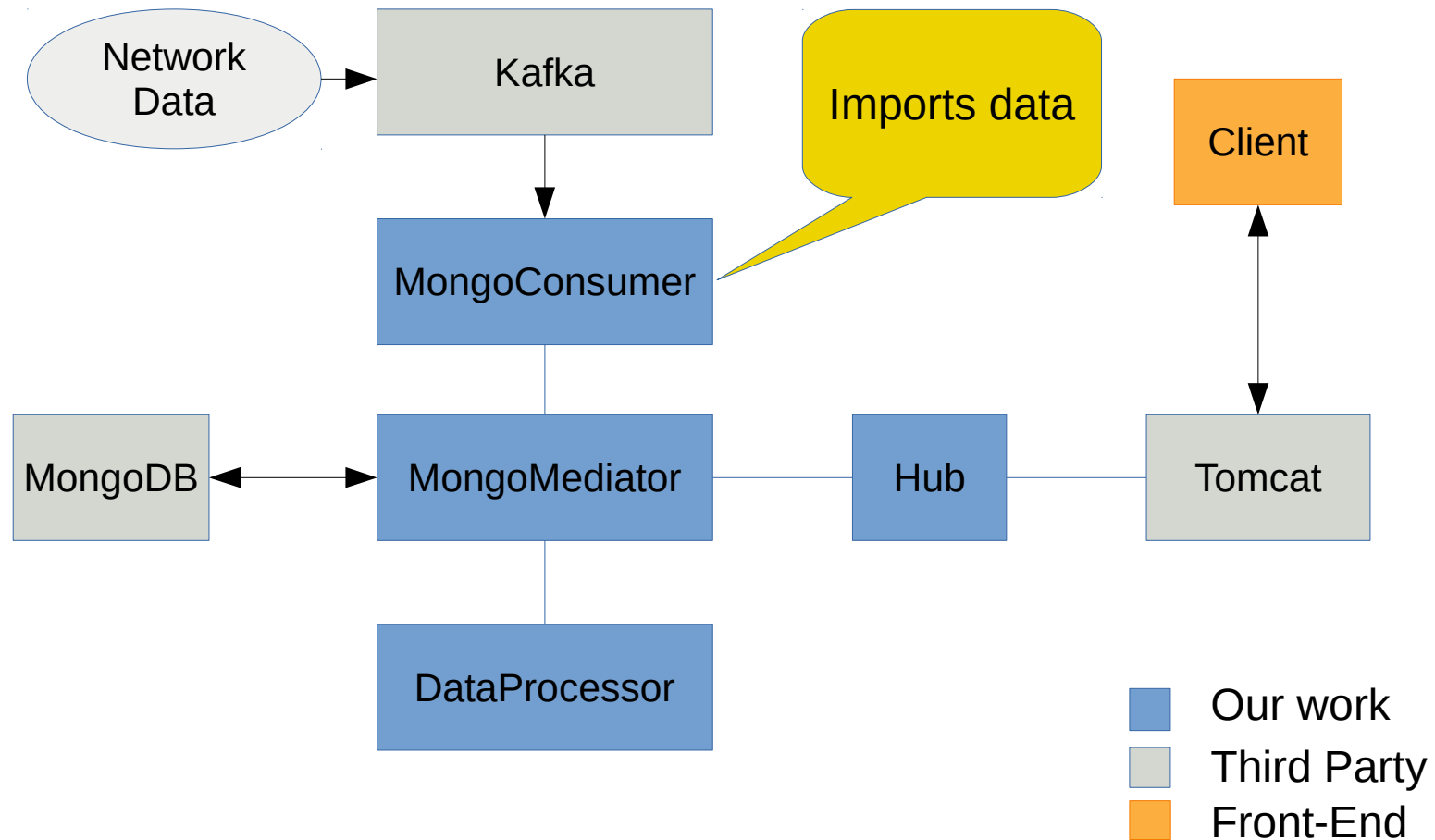


# Back-End Design

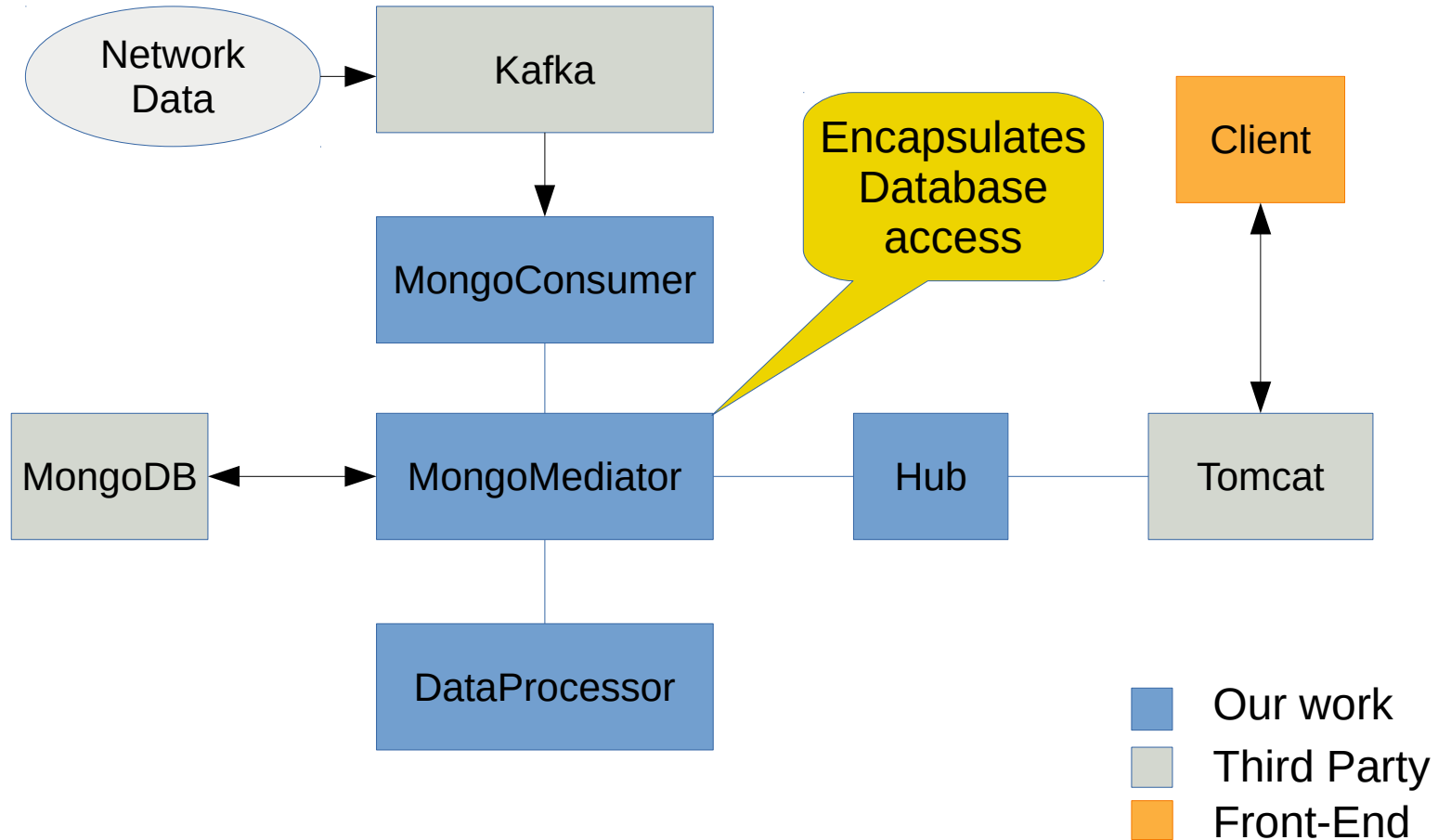




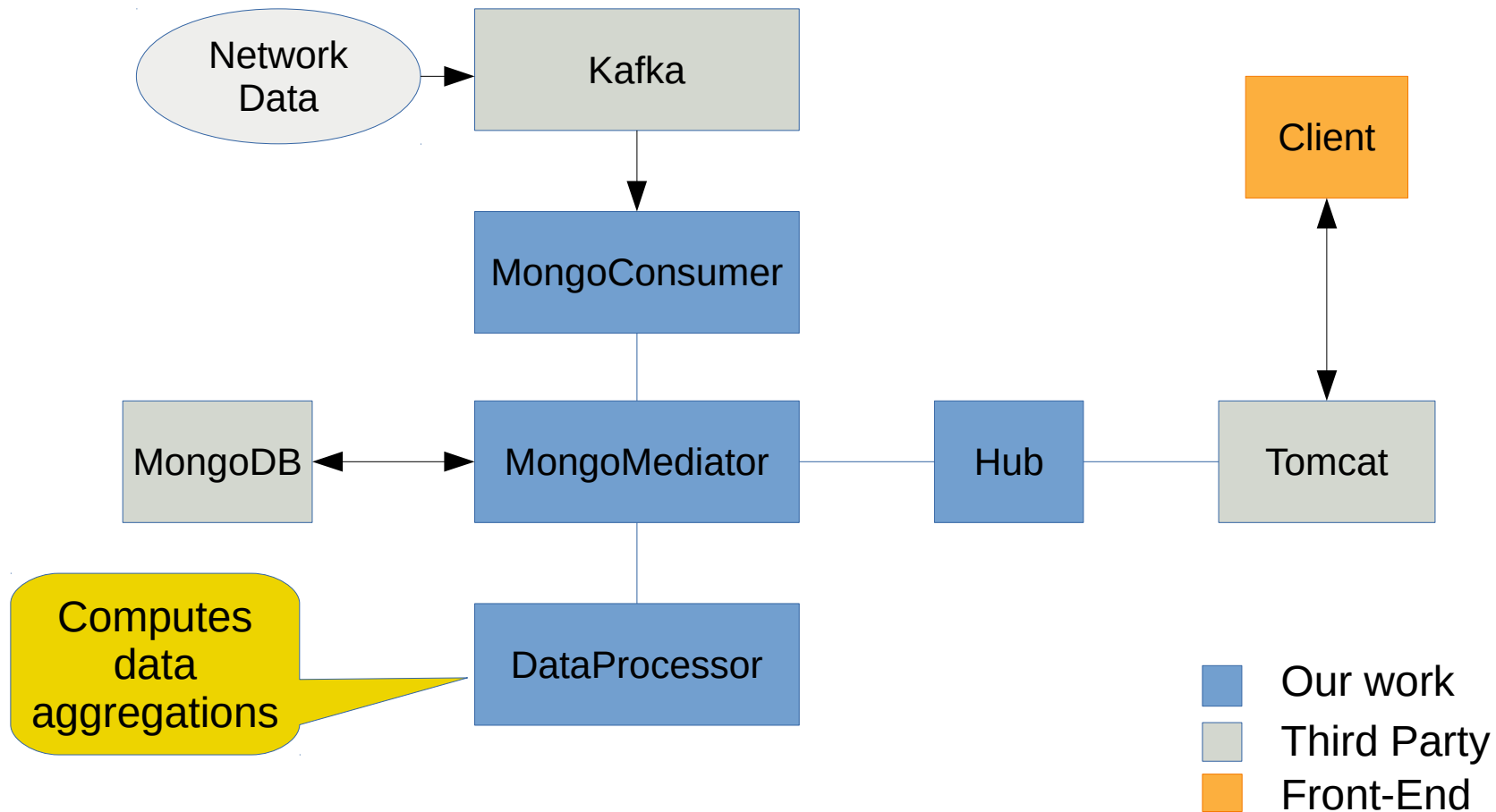
# Back-End Design



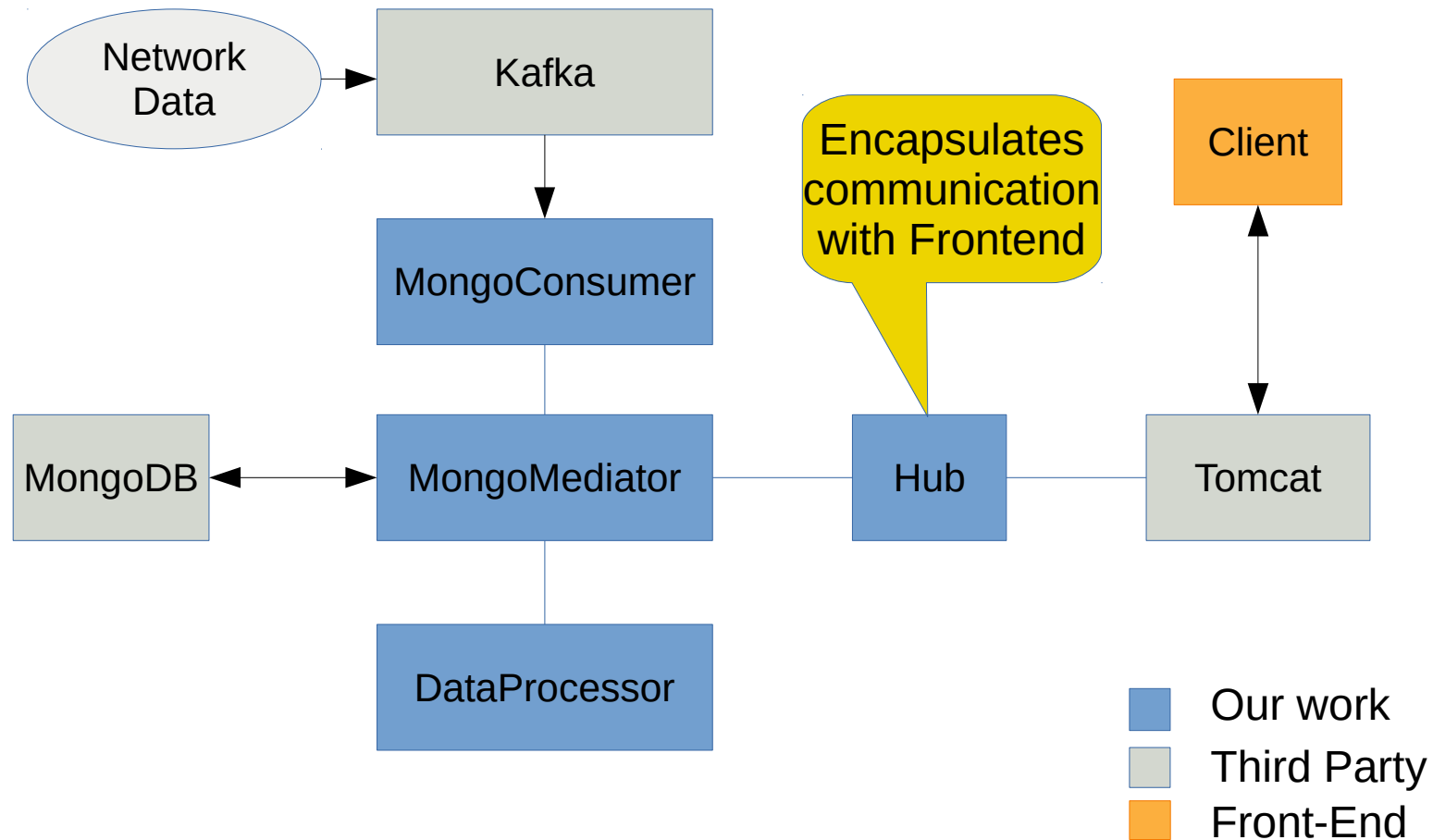
# Back-End Design



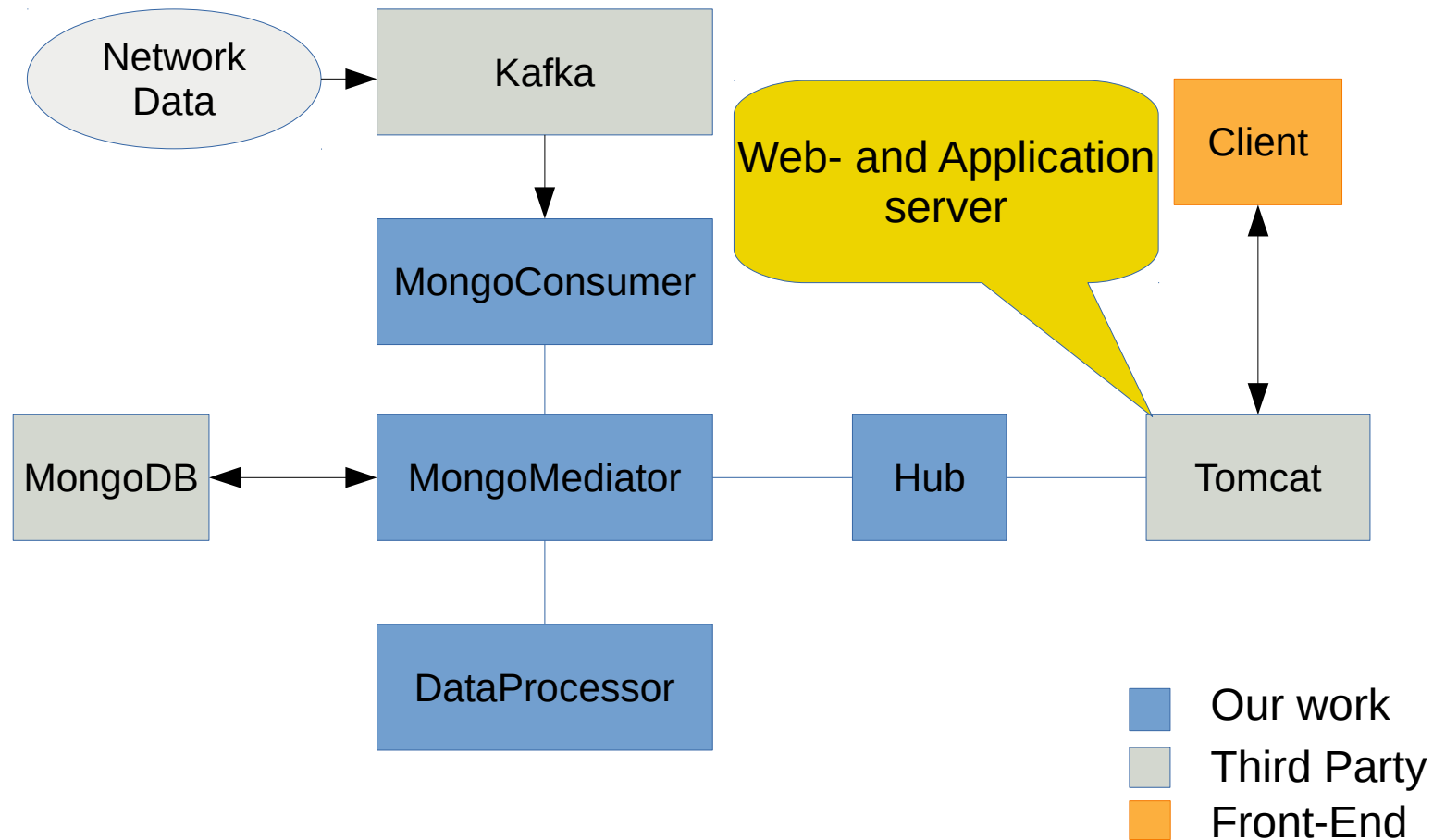
# Back-End Design



# Back-End Design



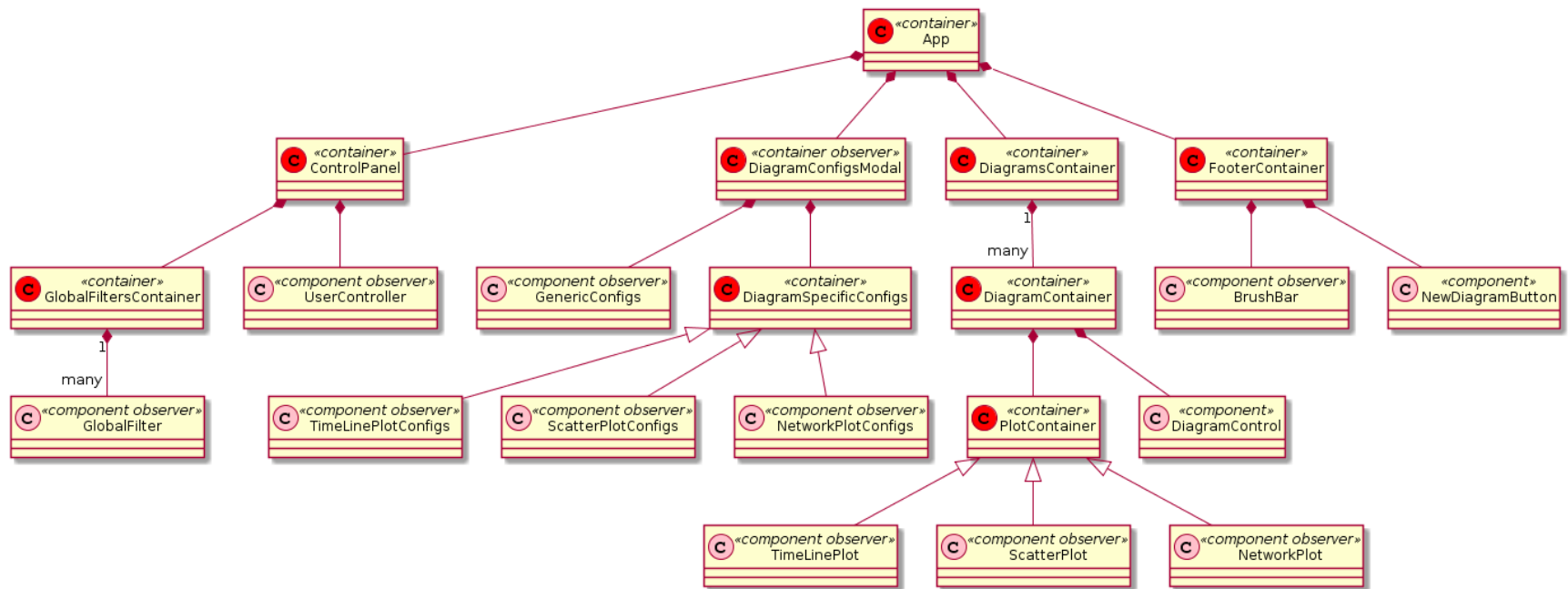
# Back-End Design



# Back-End Components

- Written in Java and using open source components

Overview of GUI Elements



# Front-End Components

- Written in Javascript
- Additional third party components (open source):

 React library

 D3 graphics library

 nivo diagram components

 MobX state management



# Development Tools Used

## ■ Tools for Back-end development:

Eclipse

 Maven

JUnit  JUnit

## ■ Tools for Front-end development:

 Visual Studio Code

 Parcel.js

 Netlify (CD)

 Git, Github 

 Slack

 Latex

# Implementation

- User access control
- Data source selection
- Multiple diagram types
- Brushing
- Modular structure
- 19 of 24 functional requirements
- ✗ Node-link diagram (partial)
- ✗ Filtering (partial)
- ✗ Data selection

# Unexpected Difficulties and Challenges

- Only four team members
- Larger Scope than expected
- Many different technologies
  - Javascript and the libraries make use of multiple programming paradigms
  - Complexity of D3
  - Nivo components have inconsistent features
  - MongoDB idiosyncracies

# Lessons Learned

- Design more thoroughly
  - Especially data structures
- Plan and schedule more strictly
- Evaluate third party components more thoroughly
- Waterfall model didn't work.

# Best Practices

- Overall design was viable
- Good commit practices
- Frequent team communication
- Flexibility
- Learning from each other

# Conclusion

- We produced a working system
- Usable as a good and extensible base for future work
- Underestimated the amount of work required
- Gained experience with teamwork
- Gained understanding of technologies