Karlsruher Institut für Technologie

Real-time Visualization of Analyzed Industrial Communication Network Traffic

FAKULTÄT FÜR INFORMATIK

The 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

The System

- 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

The System

- 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

Requirements

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

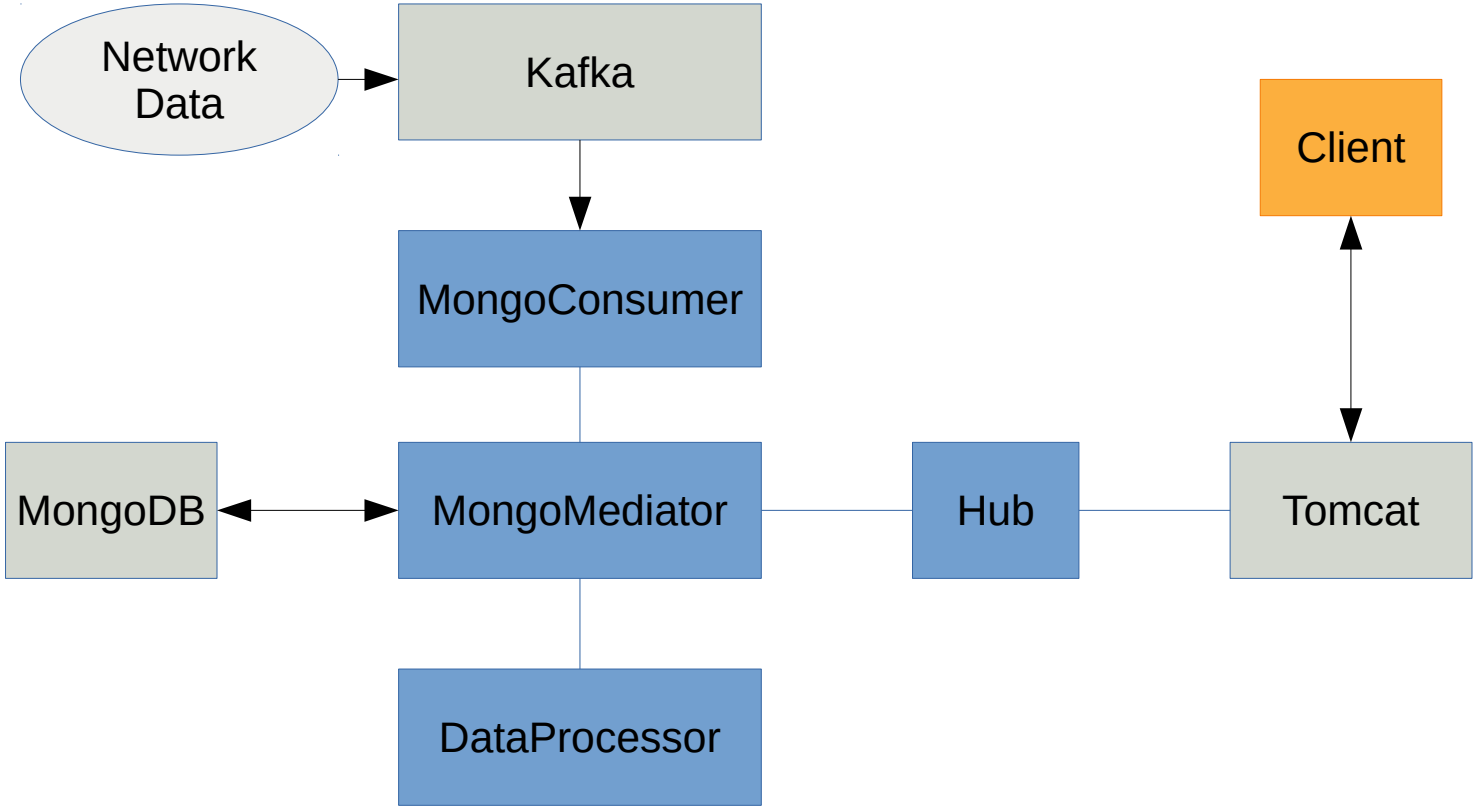
  - 12 Non-Functional Requirements

  - 15 Testcases

Architecture & Design

- Client-Server Architecture
- Back-End:
  - Mediator pattern
  - Strategy pattern
- Front-End:
  - Model-View-Controller
  - Observer

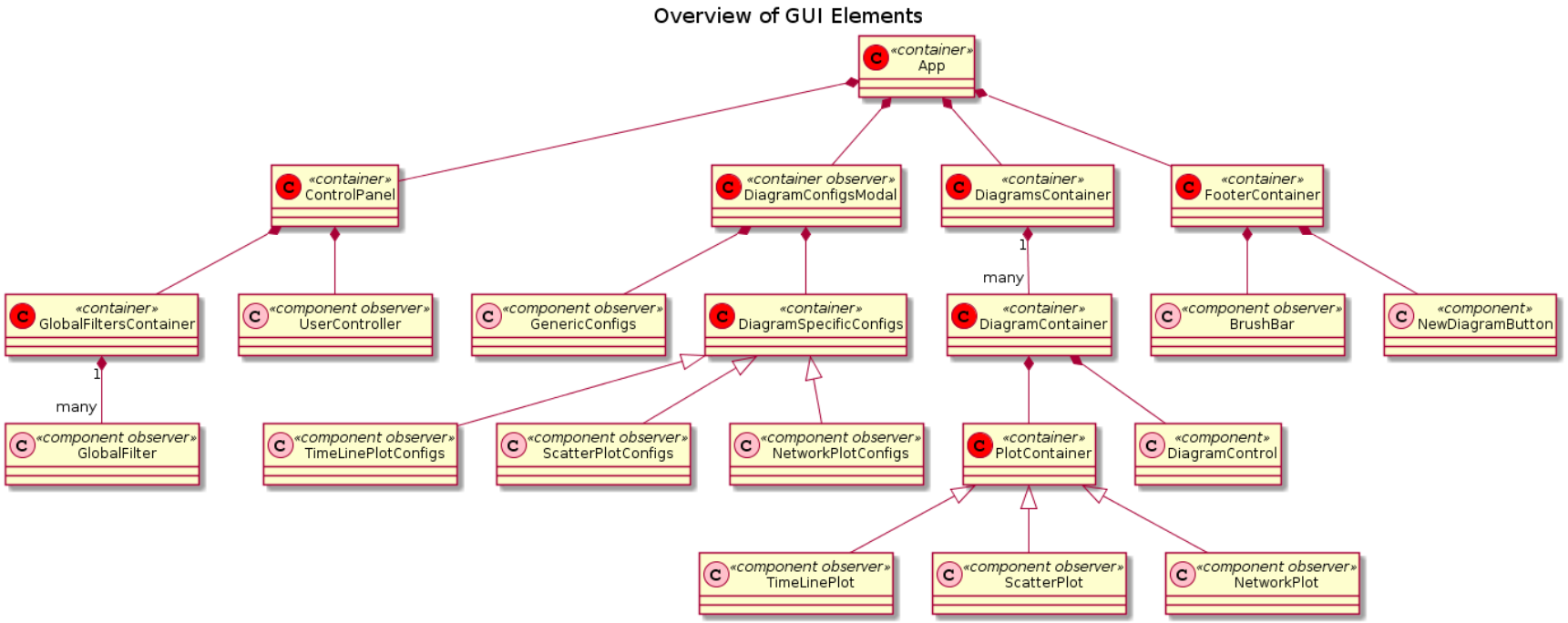
Back-End Design



Back-End Components

- Written in Java and using open source components
- Kafka streaming server
- MongoDB for both archived and real-time data
- A Kafka-Consumer move data to MongoDB
- Network hub realized as servlet running in Apache Tomcat
- A Mediator component for the database accesses

Front-End Design



Front-End Components

- Written in Javascript
- React library
- D3 graphics library
- nivo diagram components
- MobX state management
- All of the above are open source

Implementation

- Tools for Back-end development:
  - Eclipse
  - Maven
  - Junit
- Tools for Front-end development:
  - Visual Studio Code
  - Netlify (CD)
- Git, Github
- Slack
- Latex

Implementation Summary

Implemented:

- User access control
- Accessing archived and live data
- Different diagram types
- Brushing
- Modularity
- 19 of 24 functional requirements

Implementation Summary

Failed to implement:

- Node-link diagram (partially)
- Filtering (partially working)
- Live data display
- Selecting data points
- Keeping lag under 2 sec

Unexpected Difficulties

- Javascript and its libraries are more functional oriented than OO
- Complexity of D3
- Nivo components have inconsistent features
- MongoDB idiosyncracies

Challenges

- Only four team members
- Many different technologies:
  - Java, Javascript
  - Kafka
  - MongoDB
  - React framework
  - D3 graphics library
  - Nivo graphics library

Lessons Learned

- Design more thoroughly
  - Especially data structures
- Plan and schedule more strictly
- Evaluate third party components more thoroughly
- Waterfall model didn‘t work well

Conclusion

- We produced a partially working system
- Usable as base for future work and extensions
- Gained experience with teamwork