Interactive Analysis of ROS Bag Files

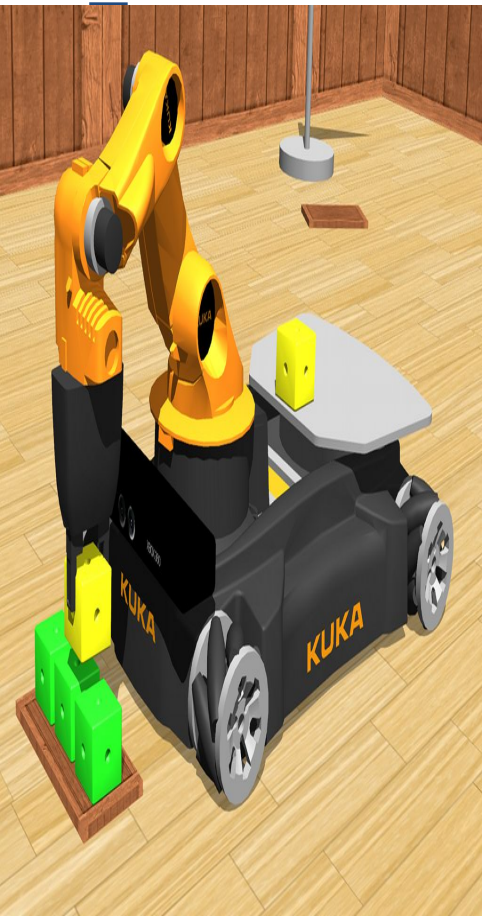
Alan Preciado Grijalva

Devaiah Ulliyada Arun

Ragith Ayyappan Kutty

Shravanthi Arvind Patil

23 January 2020, Software Development Project



Motivation

Use case from the **RoboCup@Work**

Visualization is necessary to analyze robot operation (Normal operation/ error situation)

Present workaround: Record bag file, playback rosbag to analyse problem after competition completes

Problem: No online tools available for timeline visualization of ROS bag.

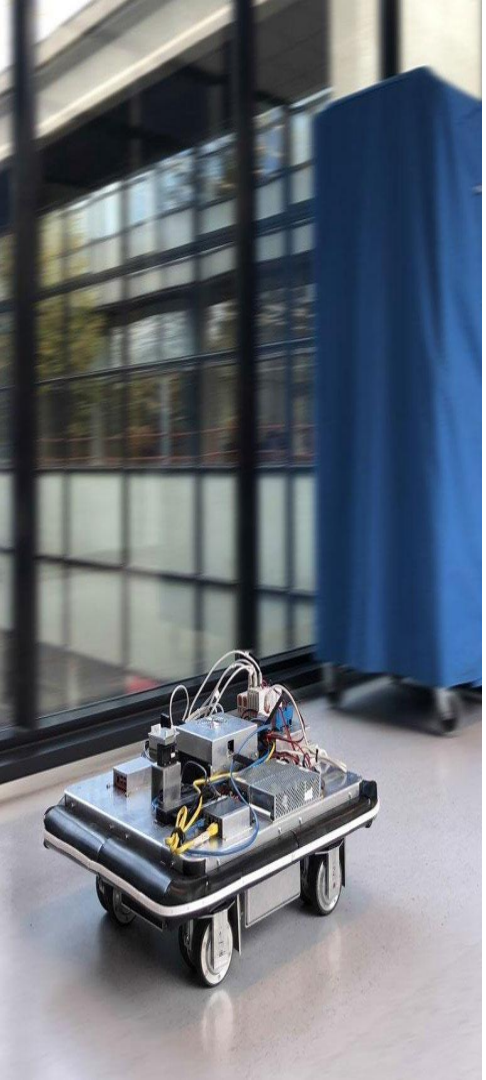
Problem Analysis

Bag files can be used to perform long-term hardware / software diagnostics logging for robots.

ROS bags are recordings of robot operations and are created by subscribing to one or more ROS topics.

Info: start and end times, topics with their types, message counts.

Contents: Timestamps, Topic names, Message data.



Solution

Developed an interactive graphical interface web based solution.

Visualized the relevant information extracted from the ROS bag for efficient analysis of the interactions between the ROS topics.

Extended the visualizer to visualize live ros topics

Technologies

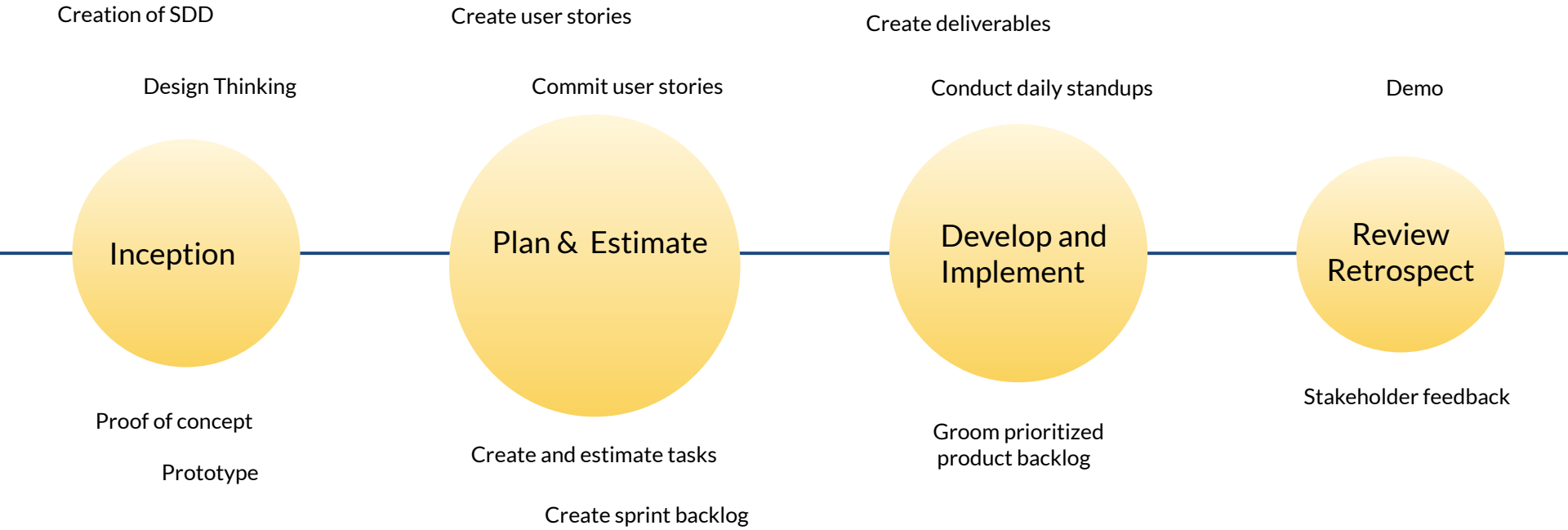
Front-end: HTML, CSS, JavaScript and Ajax

Visualization: Vis.js

Back-end: Flask
Since ROS bag are python based

Testing: Python Unit Testing

Agile Development Process



Project Team

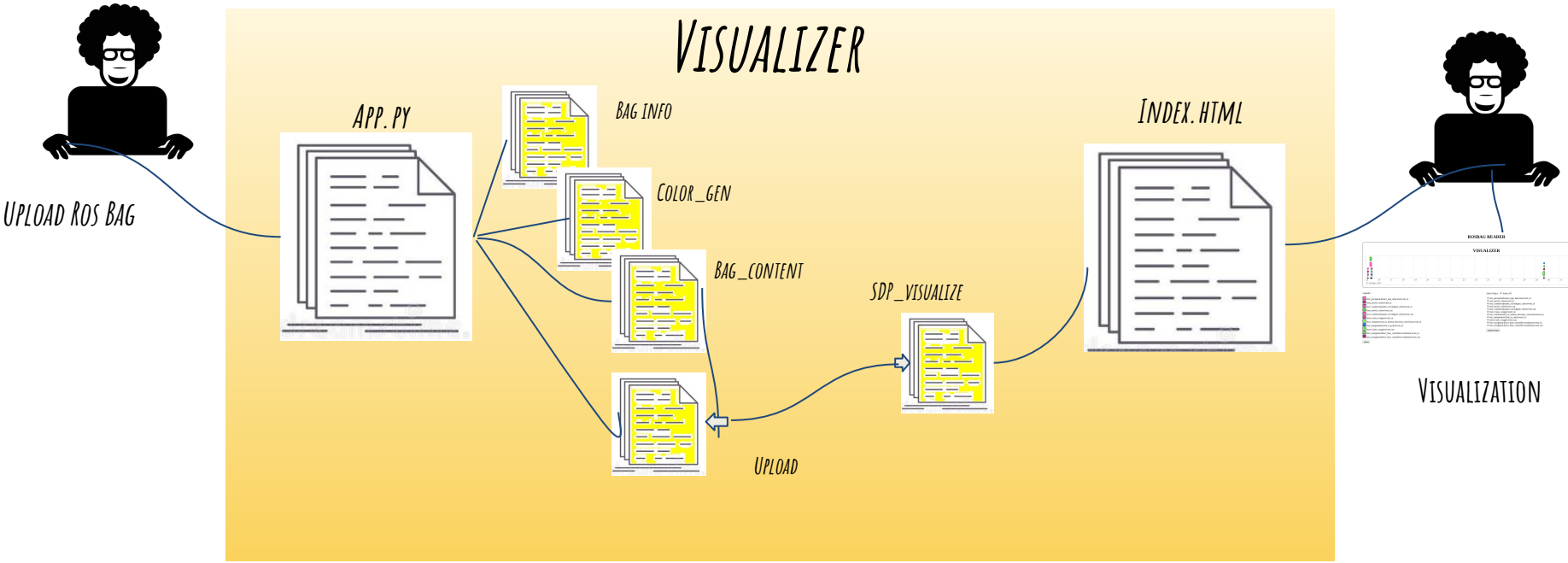
Frontend Developers

Ragith Ayyappan Kutty and Shravanthi Arvind Patil

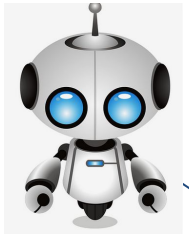
Backend Developers

Alan Preciado Grijalva and Devaiah Ulliyada Arun

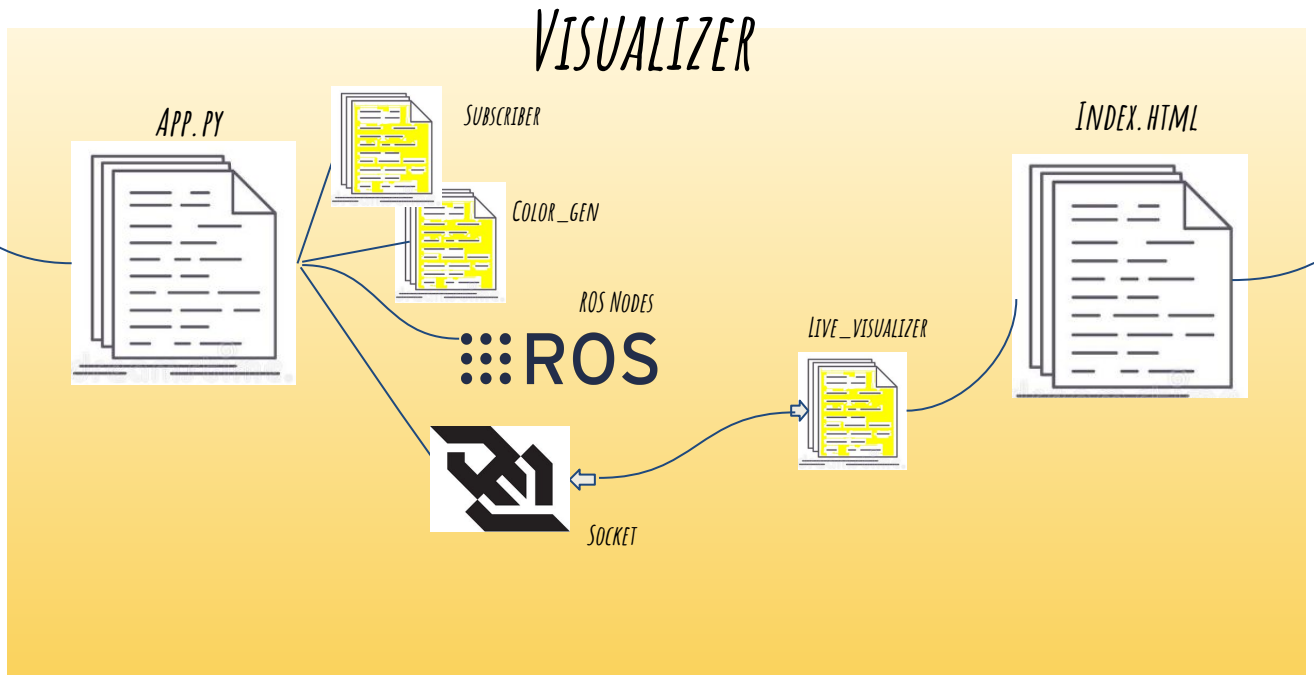
Process flow diagram: Upload

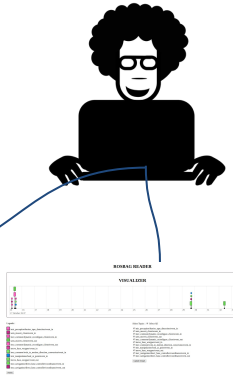


Process flow diagram: Live



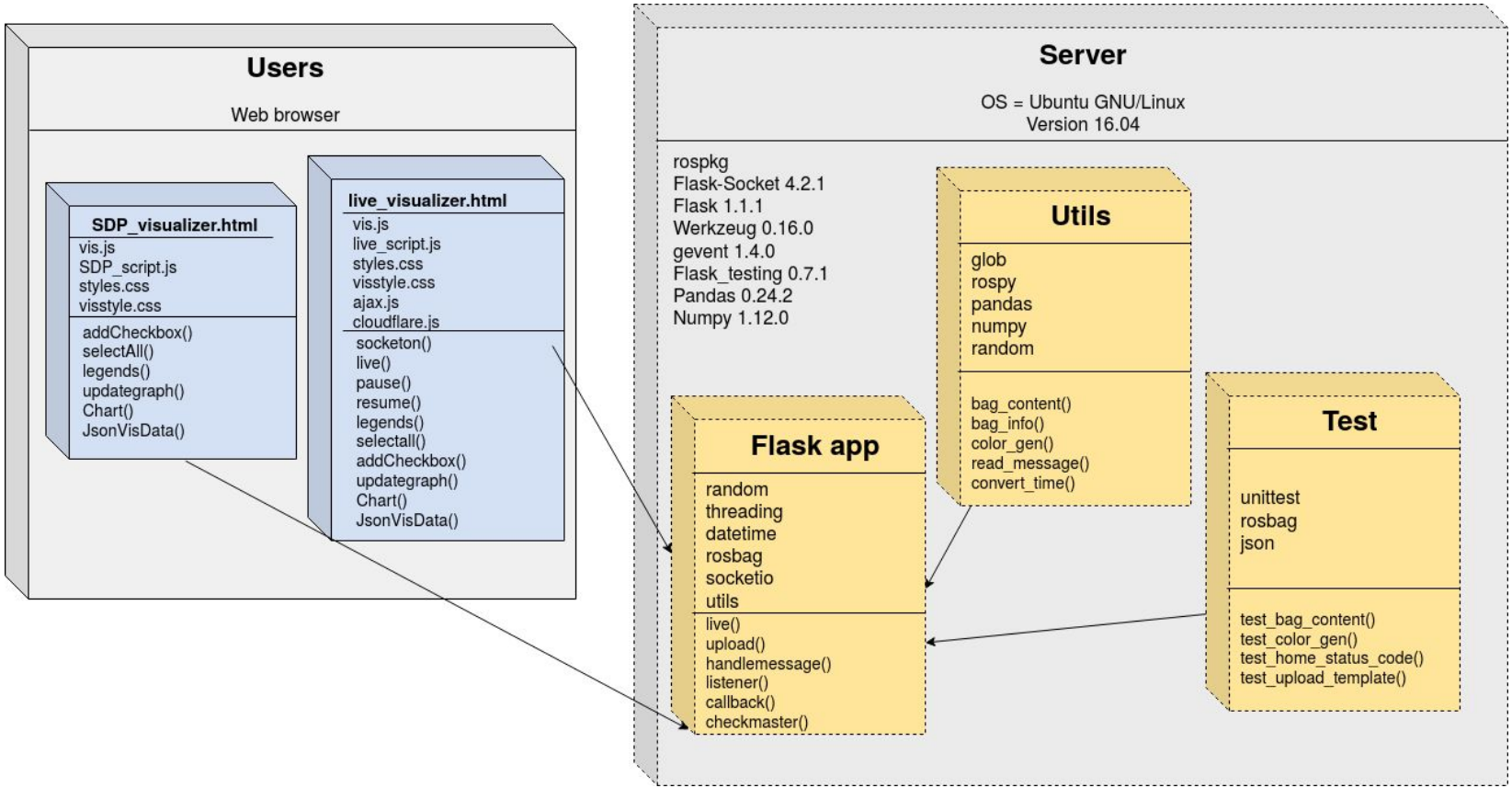
Live connection
Ros Bag





Visualization

Technical details



Challenges

- Distinguished representation of nodes based on message type
  Solution: Modified the open-source vis.js script

- Uncleared Json file during consecutive visualizations.
  Solution: Refresh both flask app and home page simultaneously

- Couldn't achieve Interactive visualization during live update
  Solution: Incorporated pause functionality.

- Could not use sockets in live visualizer using java scripts.
  Solution: Used AJAX programming instead.

- Continuous back and forth communication in Flask
  Solution: Sockets within Flask updates the page with new data without refreshing the page.

- Consequent action of launching the launch file and subscribing to ROS topics
  Solution: Achieved via multi-threading

- Granularity of time from IO.sys is not accurate to nanoseconds.
  Solution: Rounded up to microseconds and visualized.

- Information retrieval from rendered templates
  Solution: Use of self.app.test_client().get("/") to get status code

– Demo

Enhancements

- Publish the visualizer on central server
- Generalize the parser by incorporating a configuration file
- Visualize actions and services
- Handle huge bag files faster
- Visualize multiple live feeds and multiple bag files

Thank You
For Your Attention