



# MC-Messenger

SE - Final Presentation

Marcel Fischer, Erik Günther, Tim Nau, Erik Schneider



# Agenda

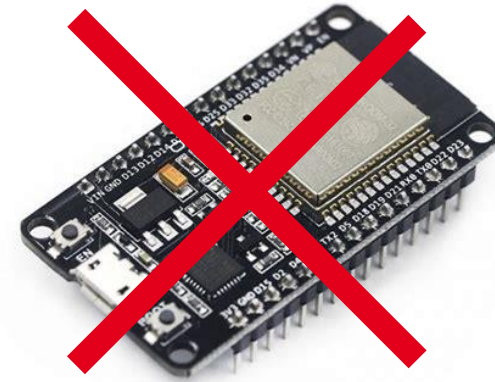
- Project goal + vision
- Design patterns/Development platform
- Architecture - decisions and reasoning
- Quality assurance
- CI/CD setup
- Project management - Facts and lessons learned
- Live Demo

# Project goal/vision

- MCM- Microcontroller Messenger
- Need to Communicate with each other
  - Make communication possible (in difficult Situations)
  - Independent and Secure Messaging Service
- Available from Everywhere
  - Web-Hosted
  - Fast & Easy

# Tech stack

- Hardware
  - Raspberry Pi 2b+
  - Client-Devices



# Tech stack

- Hardware
  - Raspberry Pi 2b+
  - Client-Devices
- Software
  - Clientside code: HTML, CSS, Javascript
  - Backend code: Javascript, Nodejs



# Design patterns

- KISS
- Clean Code
- Modifiability



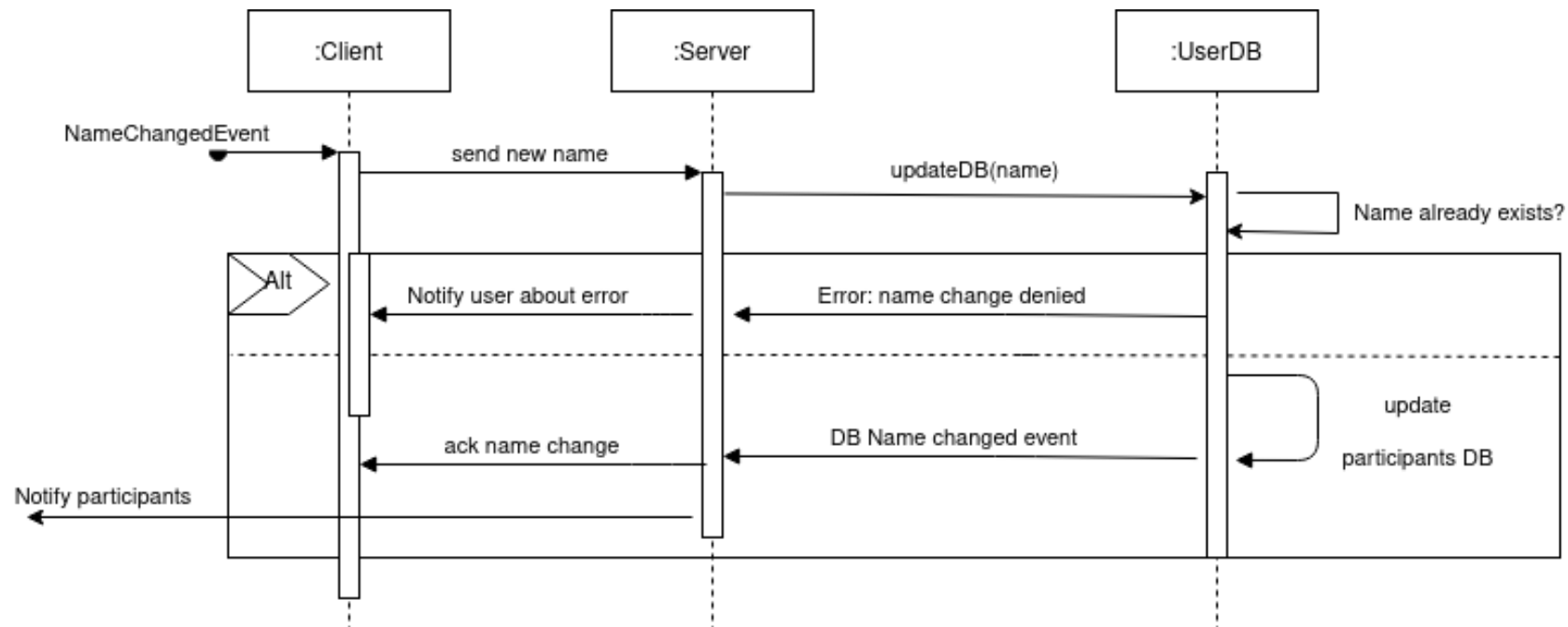
# Architecture - Design

Layered (open layers):

- Frontend
- Client-Backend: (scripts, js)
- Backend

Event-driven:

- Event: user input - message, namechange, ...)
- Server manages: flow, encapsulation of the header, check the database, forward to receivers





# Architecture - Design

Open Database design:

- Each client maintains its own database
- Mapping of the id of the user as the identifier (js mapping)
- The user object contains: id, name, color, public key
- No-persistency approach: no need for a more complex database.

Why:

Suits us very well: dependended on user interactions -> events

Event driven: perfect to handle those events (messages, name and color changes, votes etc).

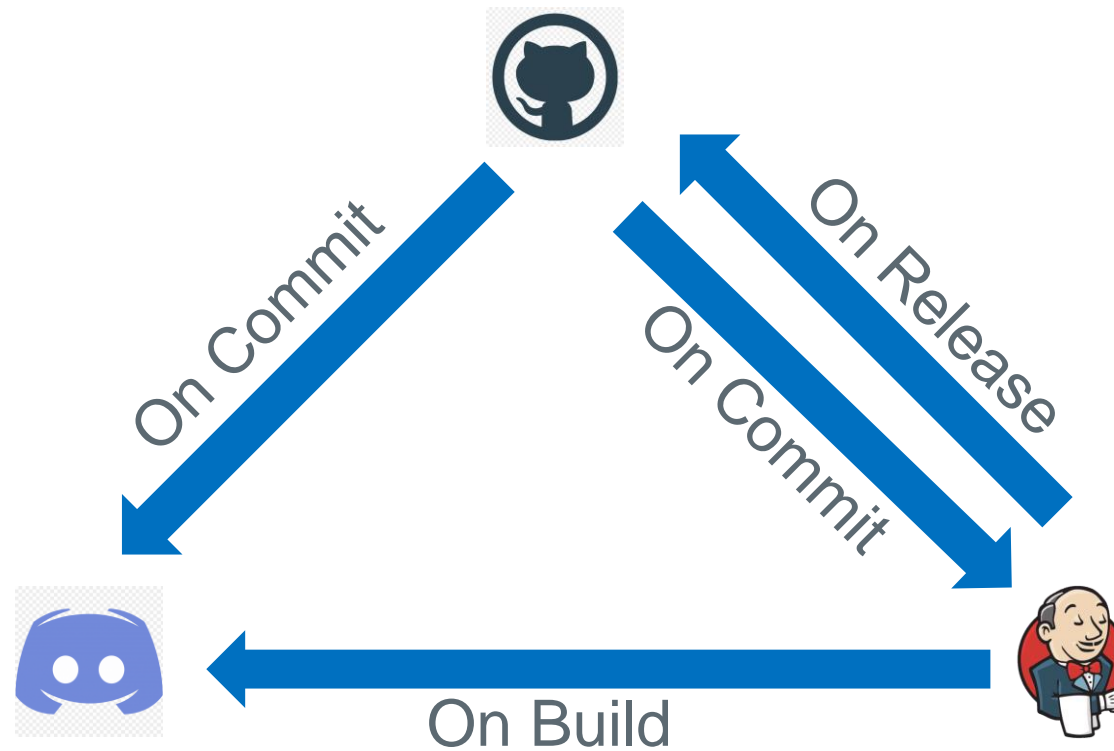
Open structure: distribute the resources and reduce the load



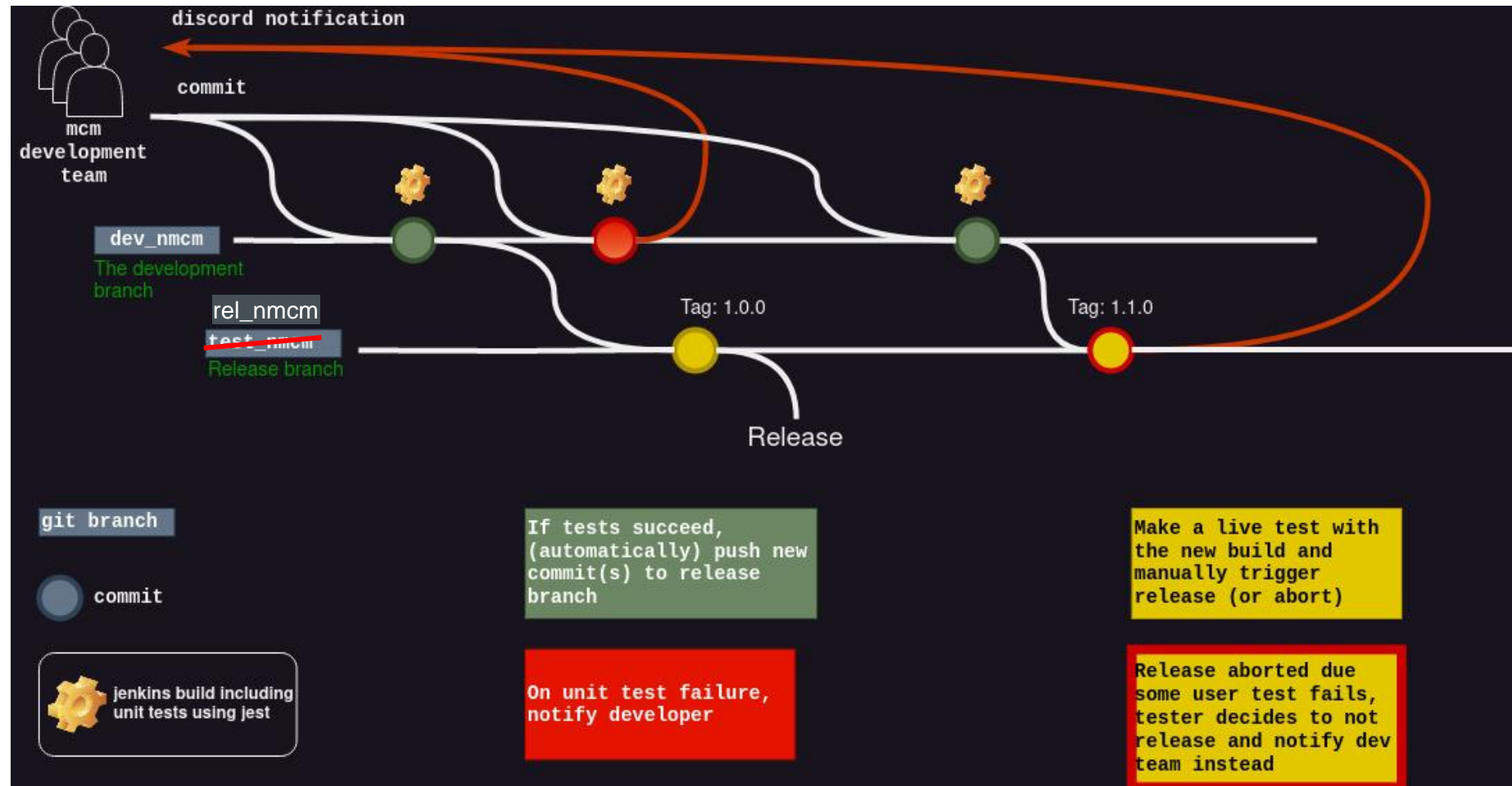
# Quality assurance

- Unittests (no release possible, if one of them fails)
- Strict separation of productive and developing code
- Continuous delivery instead of continuous deployment, everything is (or at least should be) reviewed by a human before releasing new version
- Following clean code principle
- Testing, testing and again...

# CI/CD setup



# CI/CD setup



# Project management - Facts and lessons learned

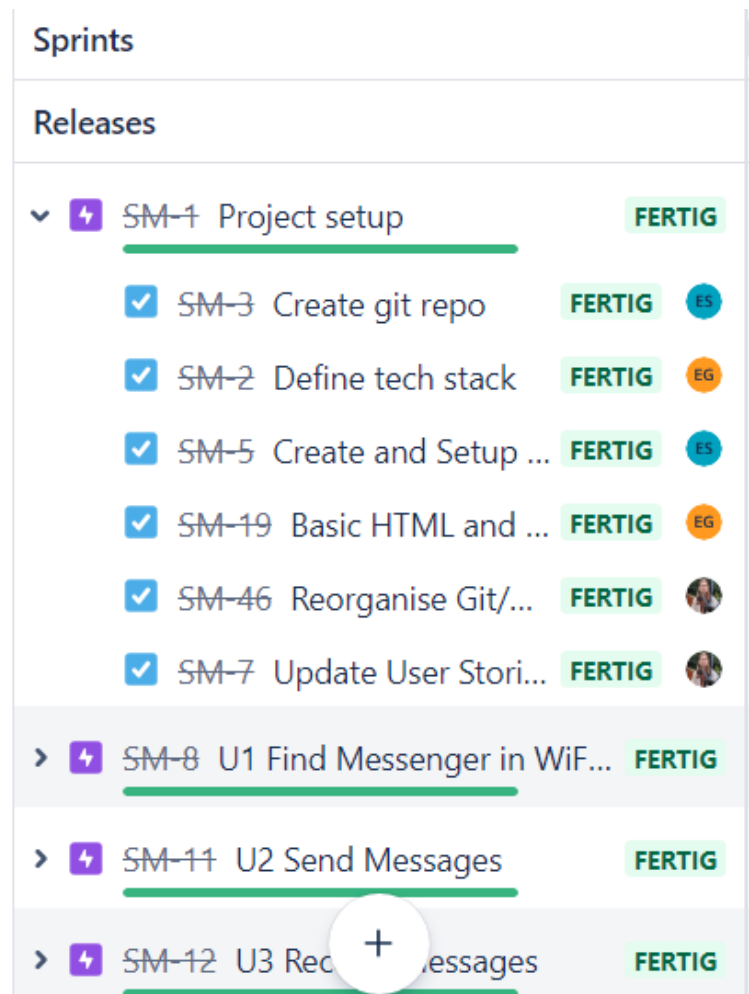
## Project Management:

- Jira
- GitHub
- Discord
- Blog: WordPress

## Scrum Setup:

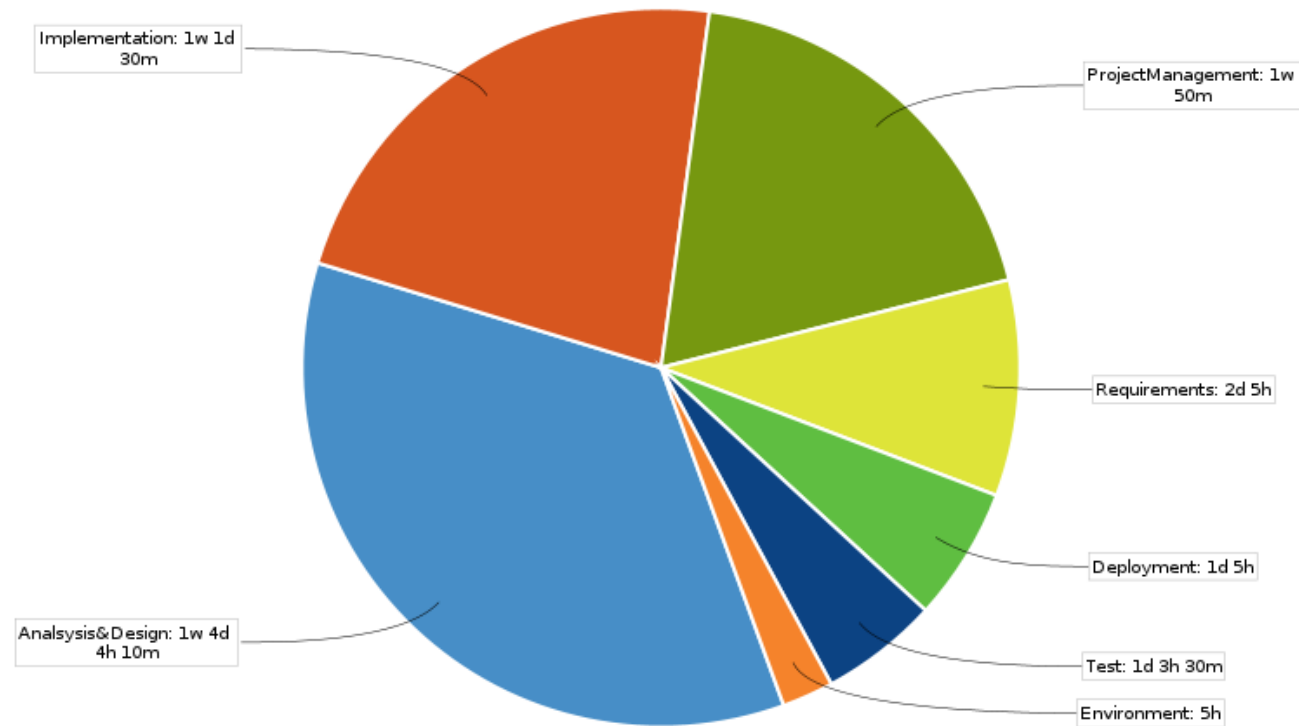
- Scrum-Plattform: Jira
- Sprint length: 1 week
- Sprint change meetings: Thursday, 03:00 pm, 1 hour
- Dailys: Monday + Wednesday, Lunch Break, 10 minutes

# Project management - Facts and lessons learned

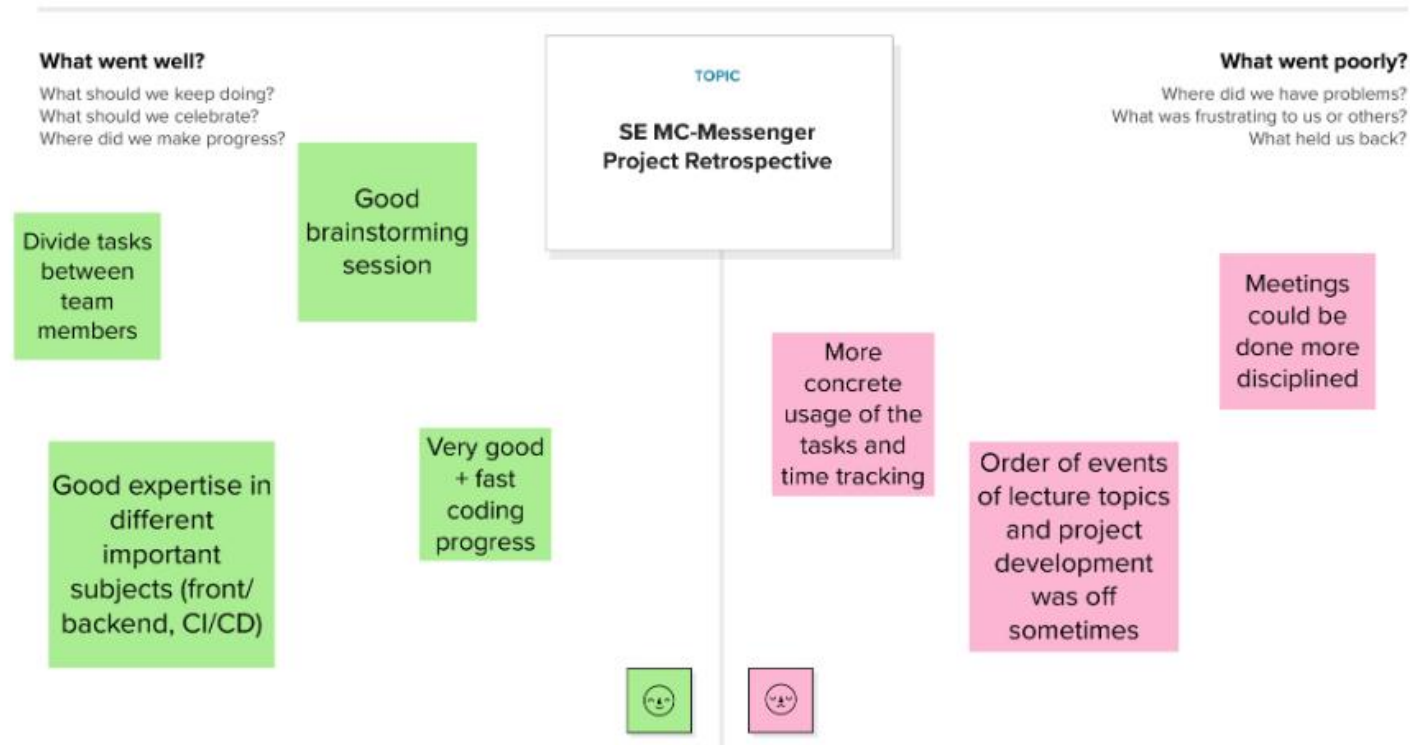
A screenshot of a project management application interface. The interface is divided into sections: 'Sprints' and 'Releases'. Under the 'Releases' section, there is a list of tasks. Each task is represented by a row with a checkbox, a task ID and description, a status label 'FERTIG' in a green box, and a small circular icon. The tasks are: SM-1 Project setup (with a lightning bolt icon), SM-3 Create git repo (with a blue checkmark icon), SM-2 Define tech stack (with a blue checkmark icon), SM-5 Create and Setup ... (with a blue checkmark icon), SM-19 Basic HTML and ... (with a blue checkmark icon), SM-46 Reorganise Git/... (with a blue checkmark icon), SM-7 Update User Stori... (with a blue checkmark icon), SM-8 U1 Find Messenger in WiF... (with a lightning bolt icon), SM-11 U2 Send Messages (with a lightning bolt icon), and SM-12 U3 Rec... Messages (with a lightning bolt icon). A large white plus sign is overlaid on the bottom right of the task list. The tasks SM-1, SM-8, SM-11, and SM-12 have green progress bars below their descriptions.

38 Epic-Task in total

# Project management - Facts and lessons learned

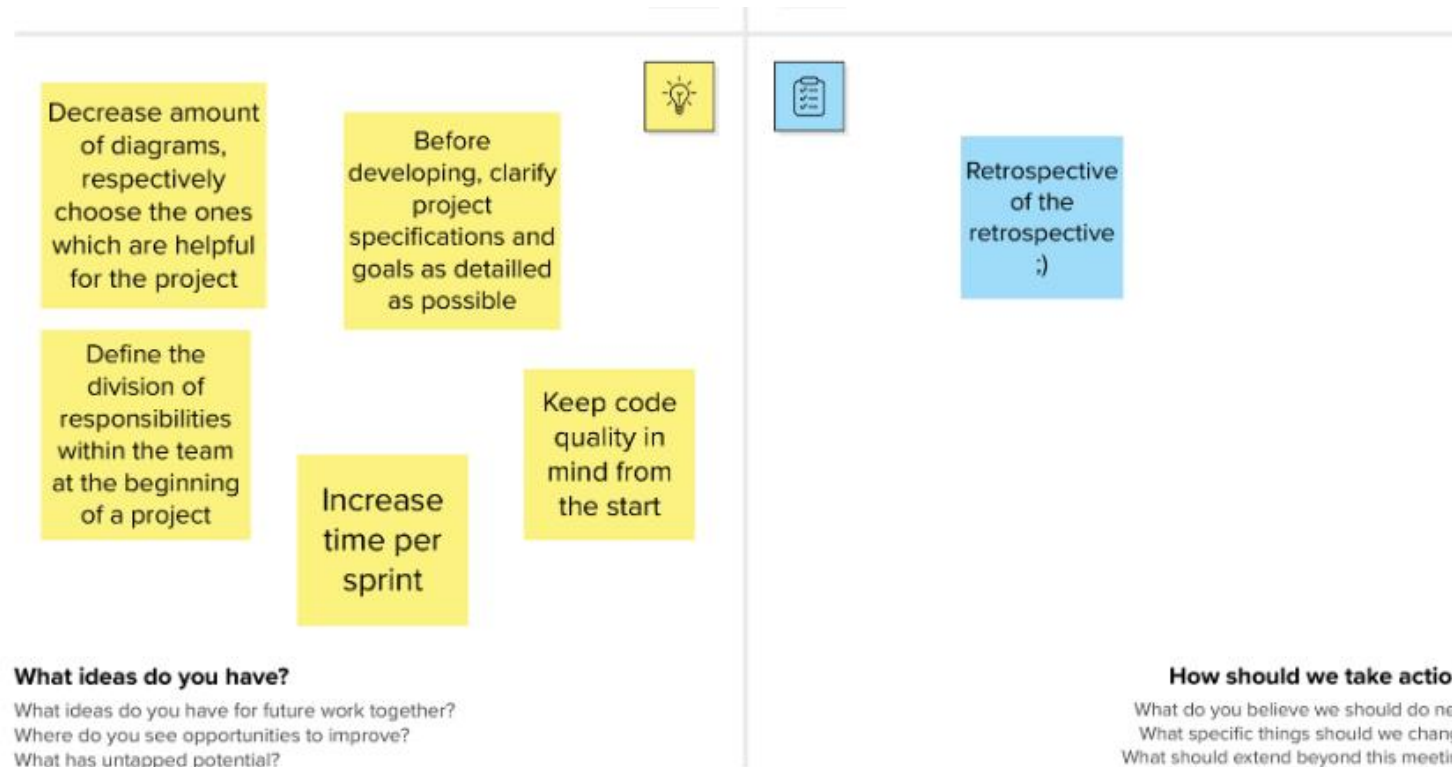


# Project management - Facts and lessons learned





# Project management - Facts and lessons learned



# Live Demo

Try it yourself!



<https://mcm.servebeer.com/live/>

Thank you for listening!  
Questions?

