knowit

knowit

Sustainable Coding Practices

How to build sustainable software?

Adapted from
Koodihuoneilmiö (podcast)
episode Kahvitaukokapina: tietokoneresurssit kuriin

https://podcasters.spotify.com/pod/show/koodihuoneilmio/episodes/Kahvitaukokapina-tietokoneresurssit-kuriin-e2li8rn

The process (simplified):

- Take some metric or estimate
  - The metric does not need to be anything fancy
  - Record your results
- Discuss changes with your team
  - You will break things less
  - Some ideas may be counterproductive
- Make *small* improvements
  - No need for a budget
  - No need for a massive project
- Make it a part of your daily work
  - Treat *sustainability* as a quality of software, like for example...
    - security, accessibility, maintainability, efficiency, testability
- Your skill and knowledge will improve over time
  - Learning by doing

Menti

- https://www.menti.com/alx56n4pqe3f

Take some metric or estimate

- The metric does not need to be perfect:
  - CI-job run length
  - Web request response time, transferred data size
  - CPU/Core/Memory utilisation, idle time
  - Cloud costs
- Record your results
  - Keep track of your measurements !
    - You need them later for comparing results
    - Keep the results in your internal document platform (Confluence, Jira)

Discuss changes with your team

- You will break things less
  - Maybe that ”unused” server had a purpose after all... 😰
- Some ideas may be counterproductive to sustainability
  - For example:
    - Pre-calculating (or caching) a financial report for all users during the night while using renewable energy
      - Wasteful if <1% of all users ever bother reading the report
      - In this case: calculate the report in real-time (on user interaction) instead

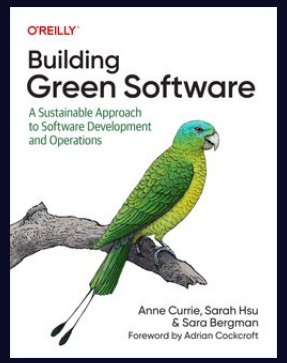
Make *small* improvements

- No need for a separate budget
- No need for that massive Jira ticket
  - “Improve system-wide energy efficiency, 21 Story Points”
    - What does that even mean?
- Again, you will break less things

Make it a part of your daily work

- Treat *sustainability* as just another quality of software, such as...
  - security (Is it secure?)
  - accessibility (What about vision-impaired users?)
  - maintainability (Is the code maintainable?)
  - efficiency (Requests per minute? Cloud costs, etc.)
  - testability (Can I test this function?)
  - ...
- When you pick up a new task, consider:
  - Is sustainability a concern?
    - How many users will use this feature?
    - How often is this function executed?
    - How long will this software be running? (1, 5 ,10, 20 years?)
  - What is the underlying infrastructure?
    - Are we using it efficiently?
    - Are we overprovisioning?
    - Autoscaling

Your skill and knowledge will improve over time



- *Learning by doing*
- Podcasts
  - Koodihuoneilmiö
    - koodihuoneilmio.fi
  - Environment variables
    - podcast.greensoftware.foundation
- Books
  - Building Green Software
    - O'Reilly: oreilly.com/library/view/building-green-software/9781098150617/
    - Free version: strategically.green/book
- Knowit Solutions Intra
  - Green Code Handbook
    - https://knowit.sharepoint.com/sites/Org-510-internal/SitePages/GreenCode.aspx
- Web Sustainability Guidelines (from W3C community group)
  - sustainablewebdesign.org/guidelines

Recap: How to build sustainable software?

The process:

- Take some metric or estimate
  - The metric does not need to be anything fancy
  - Record your results
- Discuss changes with your team
  - You will break things less
  - Some ideas may be counterproductive
- Make *small* improvements
  - No need for a budget
  - No need for a massive project
- Make it a part of your daily work
  - Treat *sustainability* as a quality of software, like for example...
    - security, accessibility, maintainability, efficiency, testability
- Your skill and knowledge will improve over time
  - Learning by doing

Workshop time!

- https://github.com/knowit-finland-javascript-guild/green-code-pipelines-workshop