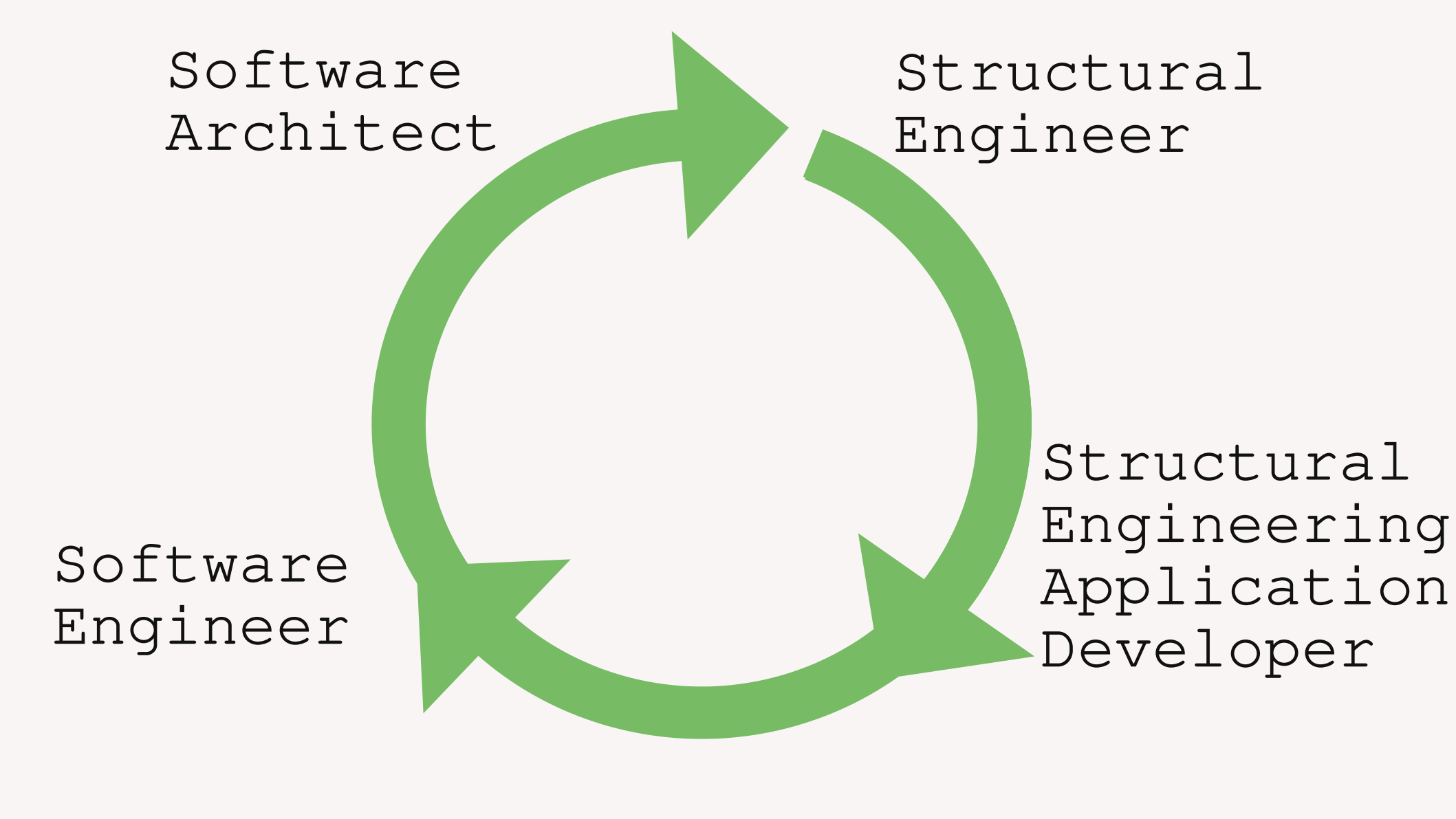
Building Bridges:
From Structural Engineering to Strategic Design

Katharina Damschen

Software Architect


Software
Architect
Structural
Engineer
Structural
Engineering
Application
Developer
Software
Engineer

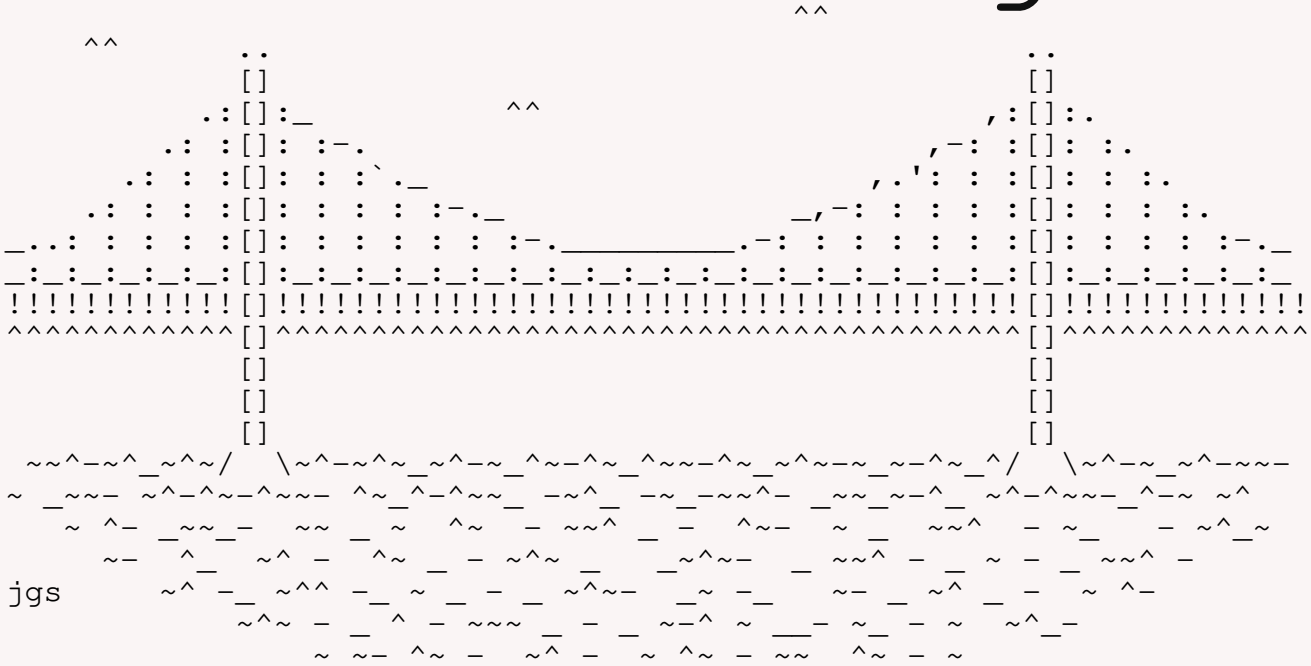
There are so many bridges in Software Architecture

.. but only few architects in bridge design

Structural Engineering

.. is in its core the study of how forces travel through a system and how that system reacts to them.

The Software Structural Engineer



Joan G. Stark (Spunk) via https://www.asciiart.eu/

Agenda

- Modeling to reduce cognitive load
- Designing for failure
- Building dependable & economic
- Simplicity & collaboration

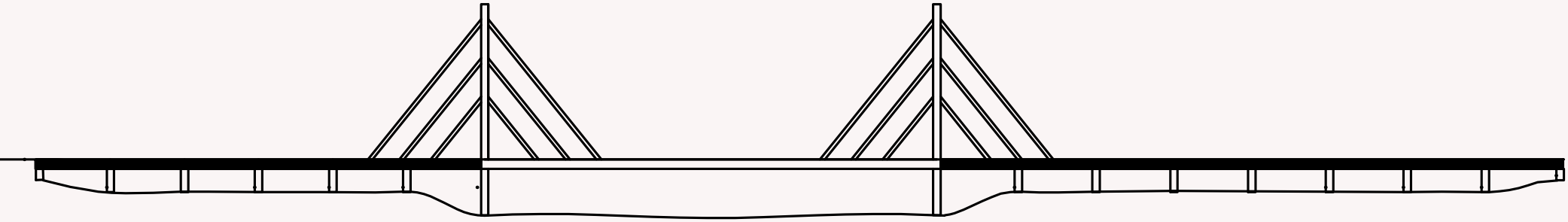
Modeling to reduce cognitive load



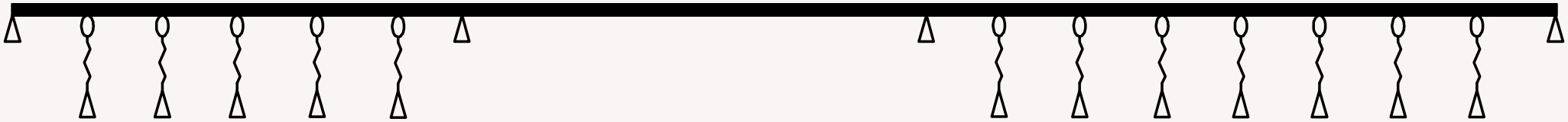
Beeckerwerther bridge, Duisburg (Germany)



Schematic

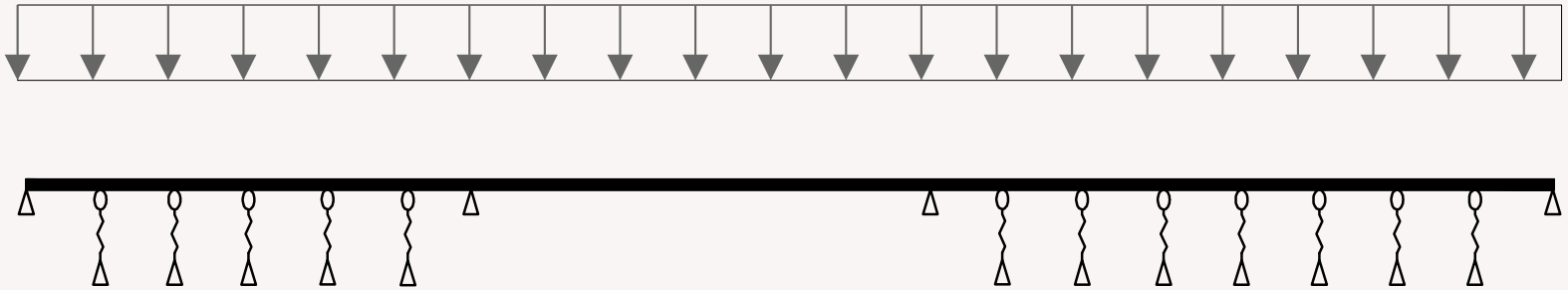


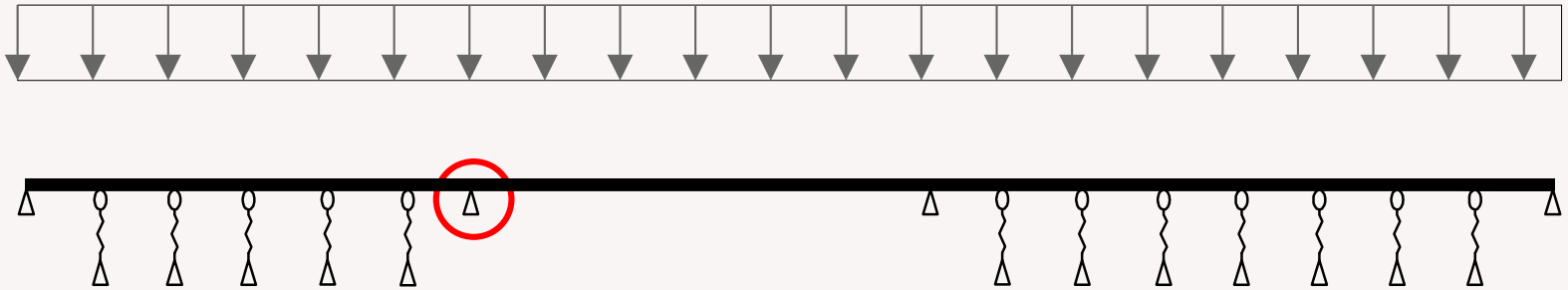
'Global Analysis'



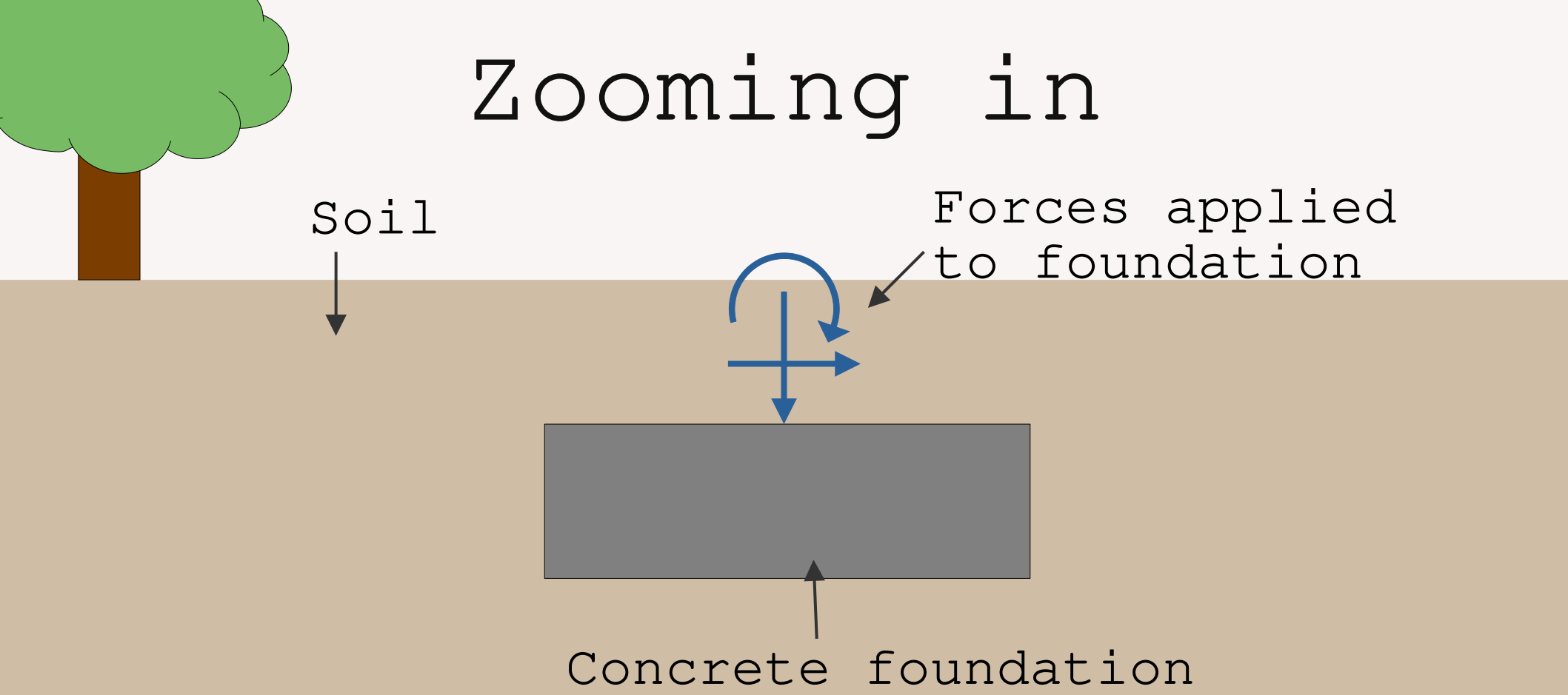
Structural Idealization

The *"Global Analysis"*

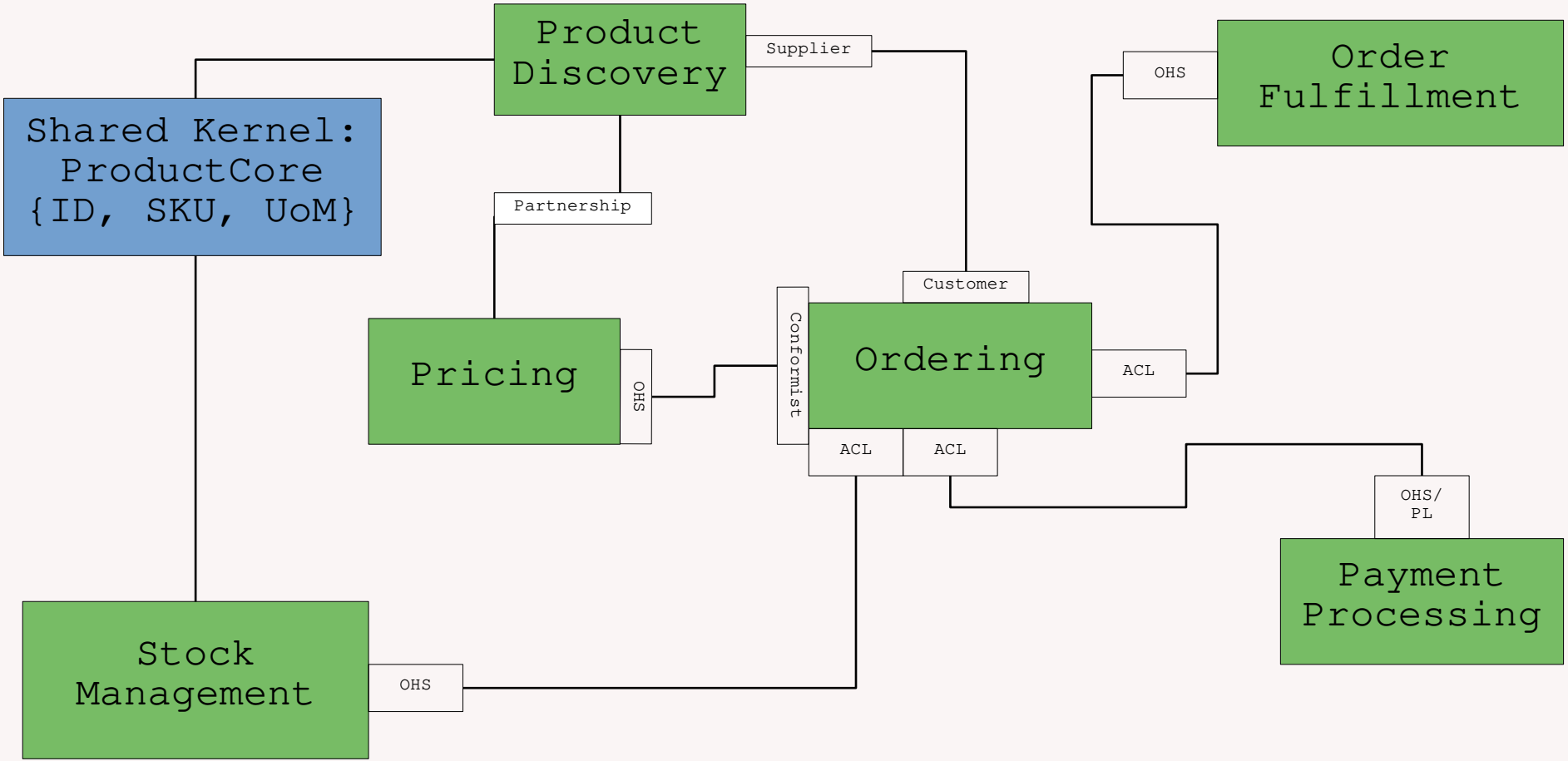




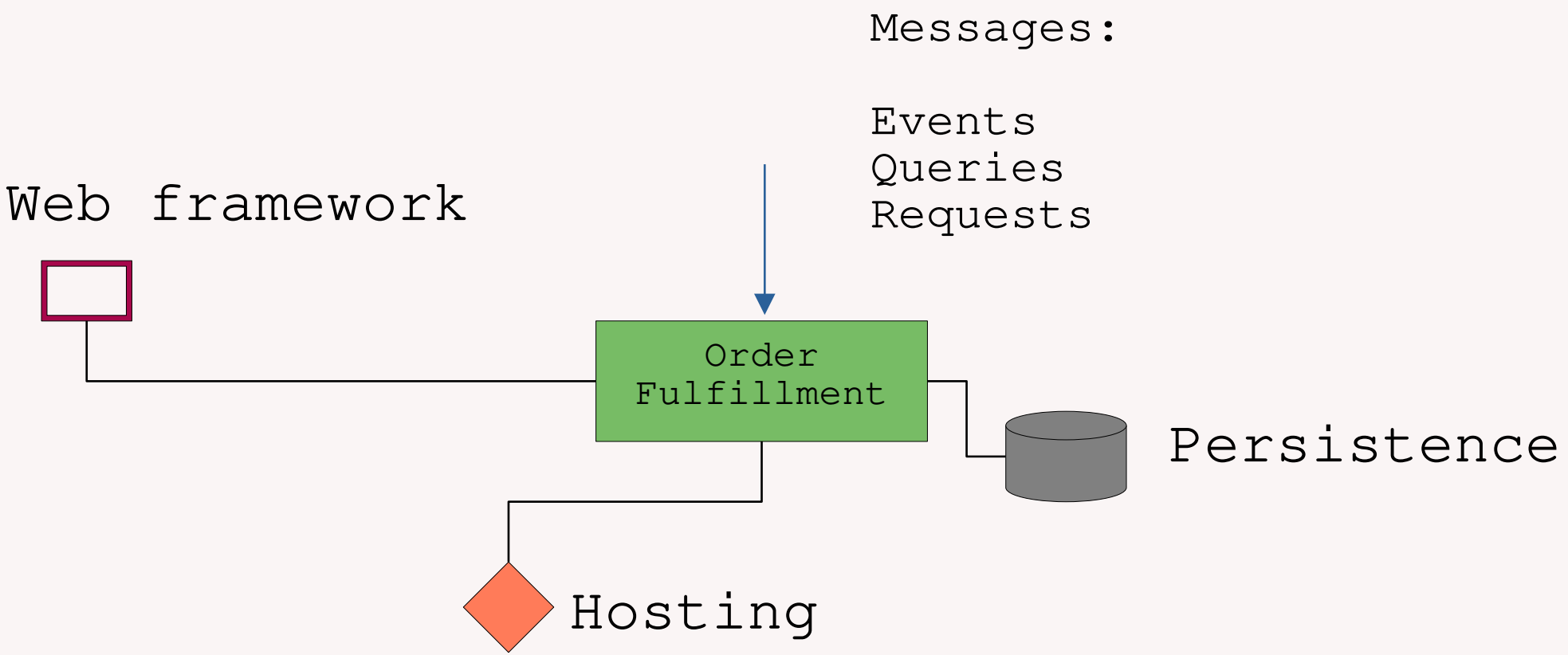
Zooming in



Context Map: E-Commerce



Zooming in


Messages:
Events
Queries
Requests
Web framework
Order
Fulfillment
Persistence
Hosting

Shared Principles

- Avoid God models
- Define Bounded Contexts to restrict cognitive load.
- Ensure contexts do not leak internals.
- Not all problems can be solved by looking closer at the details

Designing for Failure



Photo by Ben Wicks on Unsplash

Graceful Degradation



Photo by Mohit Tomar on Unsplash



Photo by Erik Mclean on Unsplash

Clarify Boundaries

Reduce Mean time to recover



Advantages that go beyond the system not failing

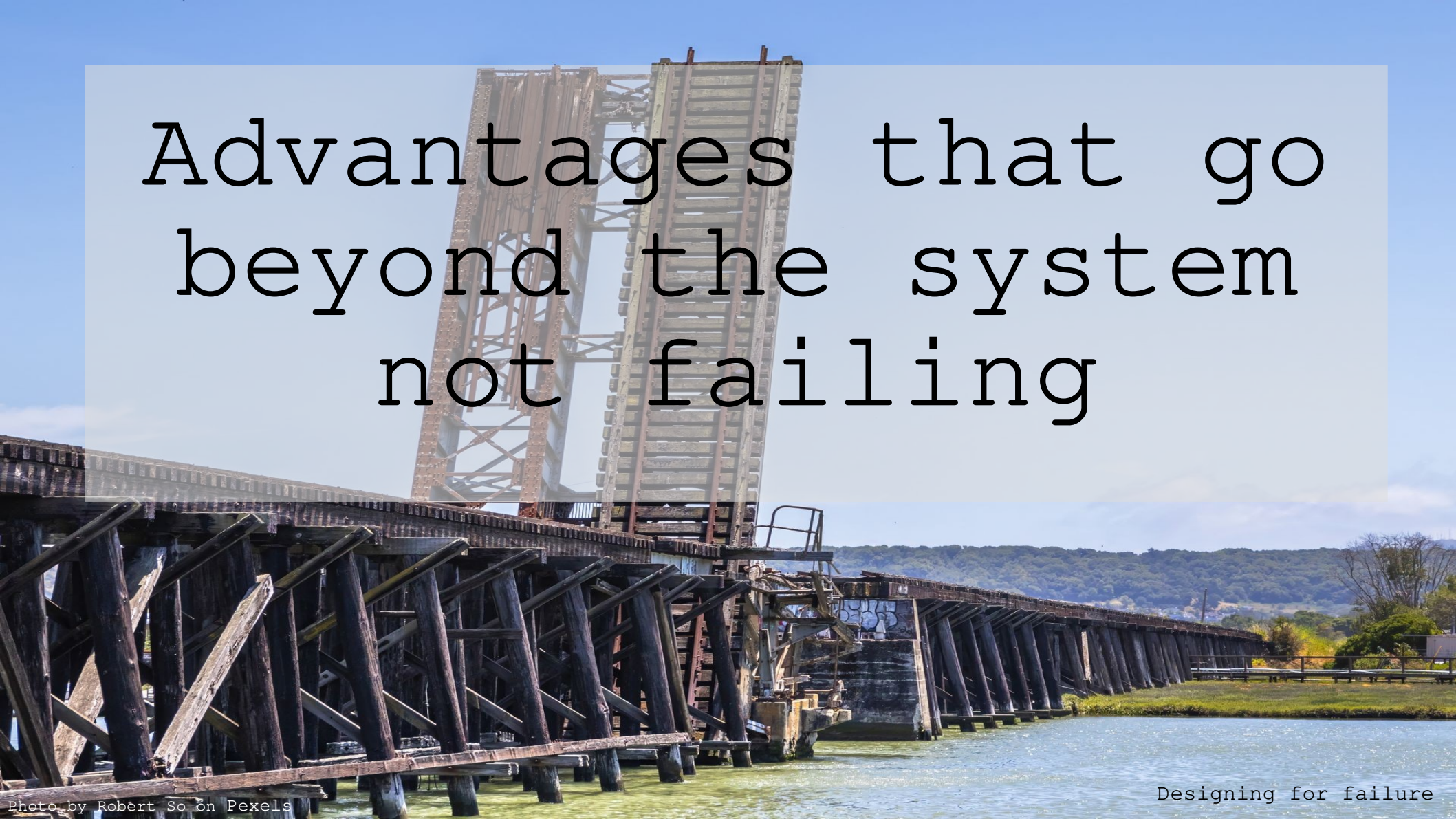
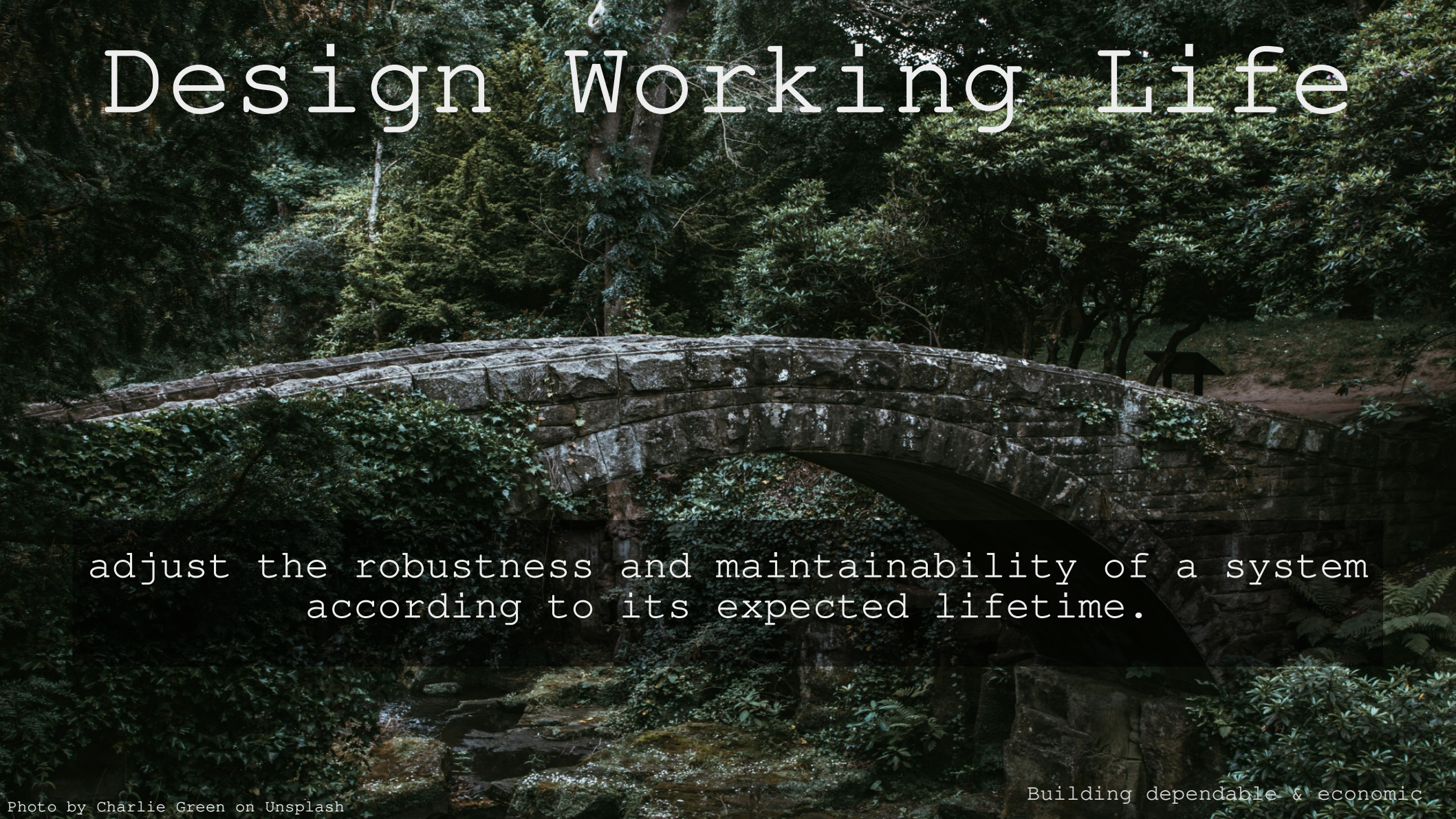


Photo by Robert So on Pexels



Seiya Maeda on Unsplash
ZEKERIYA SEN on Unsplash
Eyüpcan Timur on pexels
Dirk Erasmus on Unsplash

Design Working Life



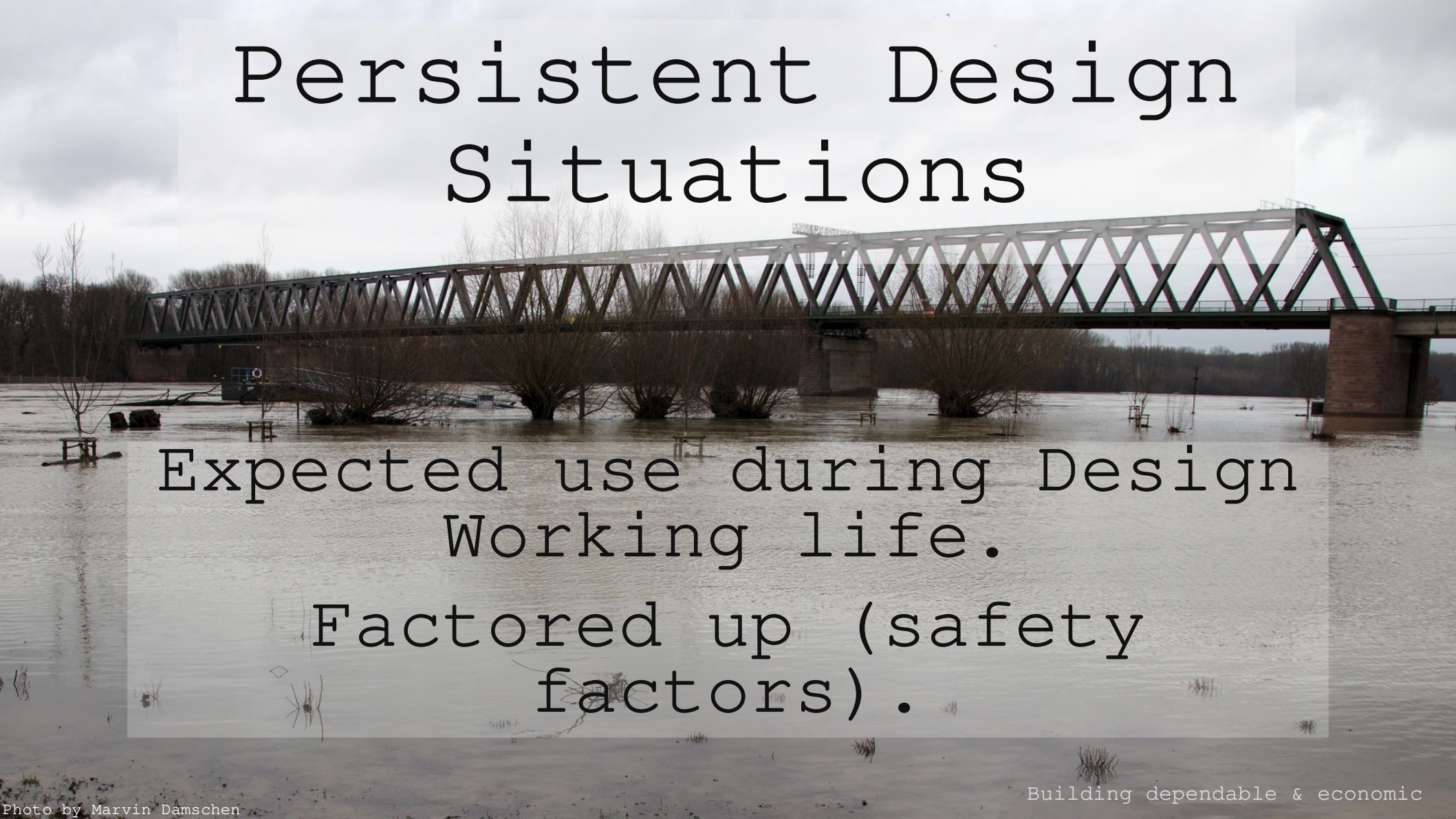
adjust the robustness and maintainability of a system according to its expected lifetime.

Photo by Charlie Green on Unsplash

Building dependable & economic

Design Situations

Persistent Design Situations



Expected use during Design Working life.

Factored up (safety factors).

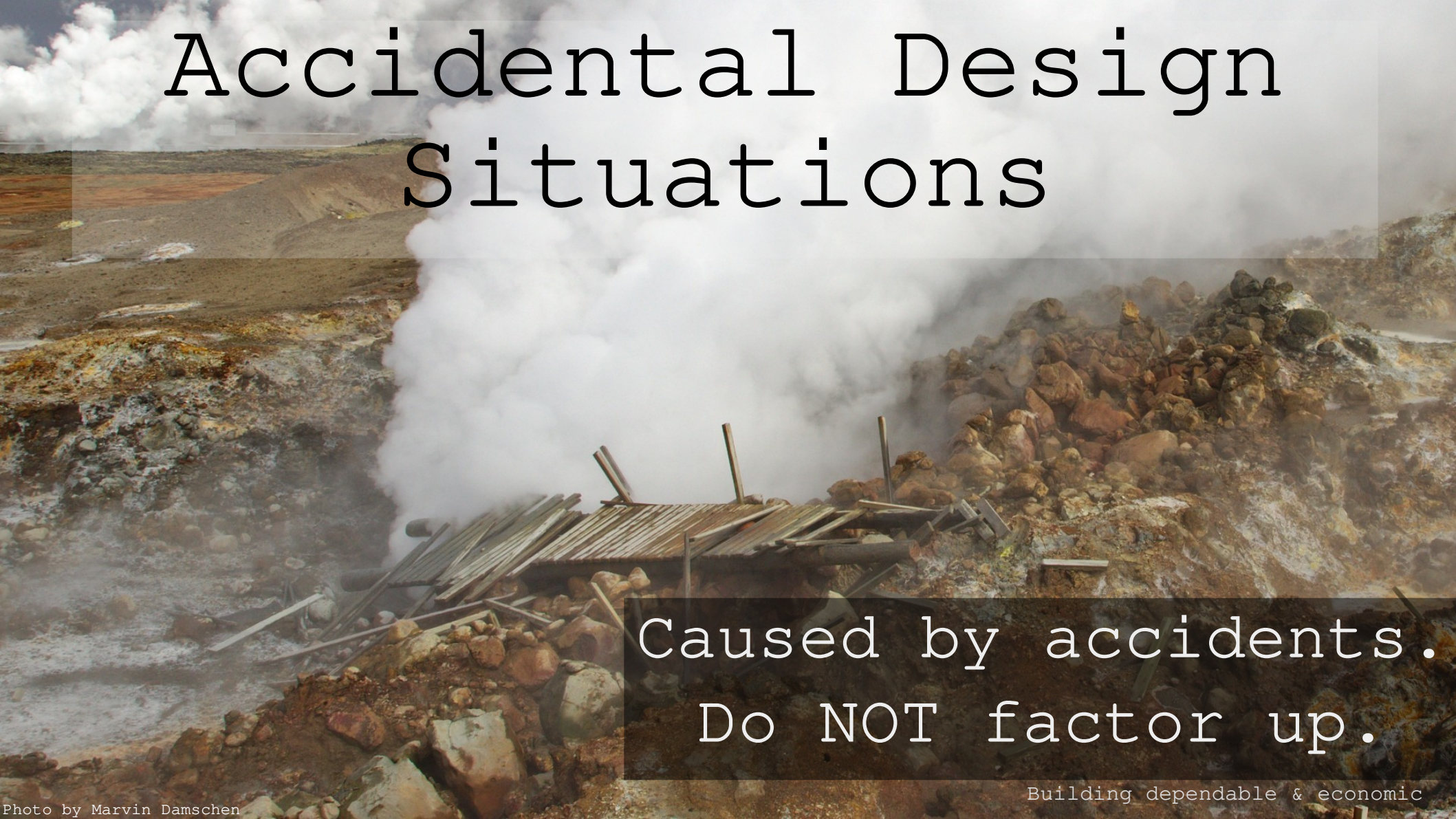
Photo by Marvin Damschen

Persistent Design Situations



Image by Maicon Fonseca Zanco on Pixabay

Accidental Design Situations

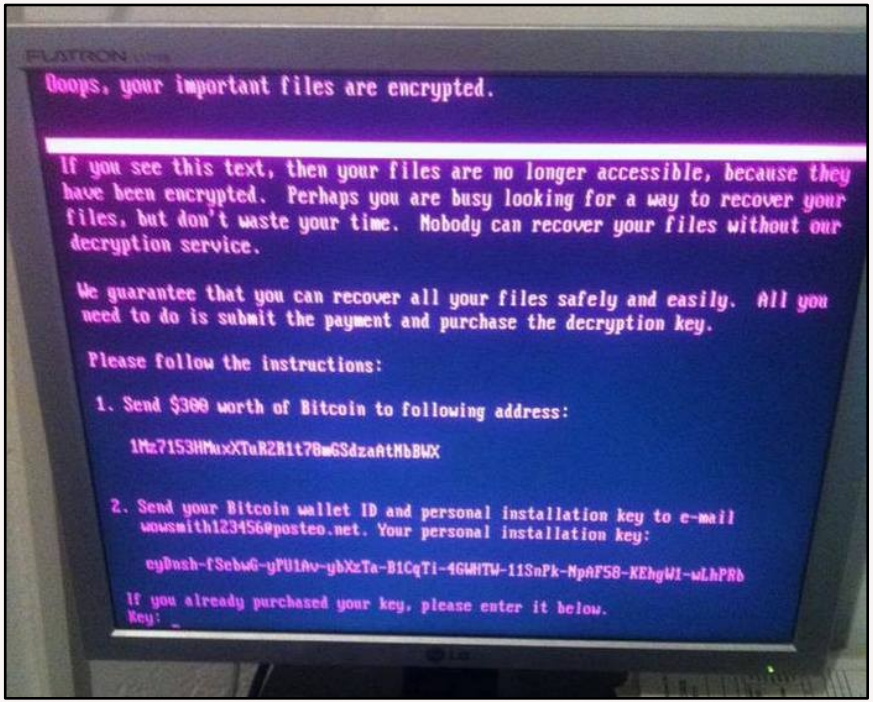


Caused by accidents.
Do NOT factor up.

Building dependable & economic

Photo by Marvin Damschen

Accidental Design Situations



By User:Jbuket - https://uain.press/blogs/yevgen-buket-vitannya-petru-o-vid-pyeti-dnya-konstytutsiyi/, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=60473519

In which situations do you want your system **to work as intended,** and when is it okay for the system to stop working as long as the **minimum guaranteed behavior** is met?

Limit States



Photo by Brian Kelly on Unsplash

Ultimate Limit States

Building dependable & economic

Ultimate Limit States

Geotechnical → External Services

Equilibrium → High Coupling

Strength → Resource Constraints/Bugs

Fatigue → Cruft, Software Rot

Deliberate Threats

.. introduce an additional Ultimate Limit State:

Security



Serviceability Limit States

Building dependable & economic

Photo by Zülfü Demir on Pexels

Kritiken handlar dels om utformningen av systemet i sig, som att det krävs nio klick för att skicka i väg ett brev, dels om att utbildningen i det nya systemet är otillräcklig. Förutom det 50-tal

From: https://lakartidningen.se/nyheter/oro-infor-millenniumstarten-pa-sas-bromsa-taget-innan-det-kraschar/ 26-04-10

The criticism concerns, on one hand, the design of the system itself,such as the fact that it takes nine clicks to send a letter,and on the other, that the training for the new system is insufficient.

already halted the planned introduction of Millennium in 2021 after hundreds of doctors voiced fierce criticism. Millennium had been developed for American and not Nordic conditions, they said.

: “serious performance issues”

From: https://www.heise.de/en/news/Scrapping-the-millennium-introduction-of-a-health-record-in-Sweden-fails-10323142.html 2026-04- 10

Both

the Ultimate and Serviceability limit states **must be satisfied.**

"Nines don't matter when users aren't happy."

Charity Majors

Consequence Classes

Build in a buffer into systems whose failure would have extraordinary consequences.

Deliberately accept lower robustness for lower-consequence systems.

Recap

- Decide what to design for -> Design Working Life & Design Situations
- Decide what must not happen -> Ultimate & Serviceability Limit States
- Decide how much buffer is justified -> Consequence Classes

Zooming out

Changing the strategy from **designing for all possible accidents**

to

Risk Avoidance

Simplicity & Collaboration



Tiger and Turtle, Duisburg, Germany

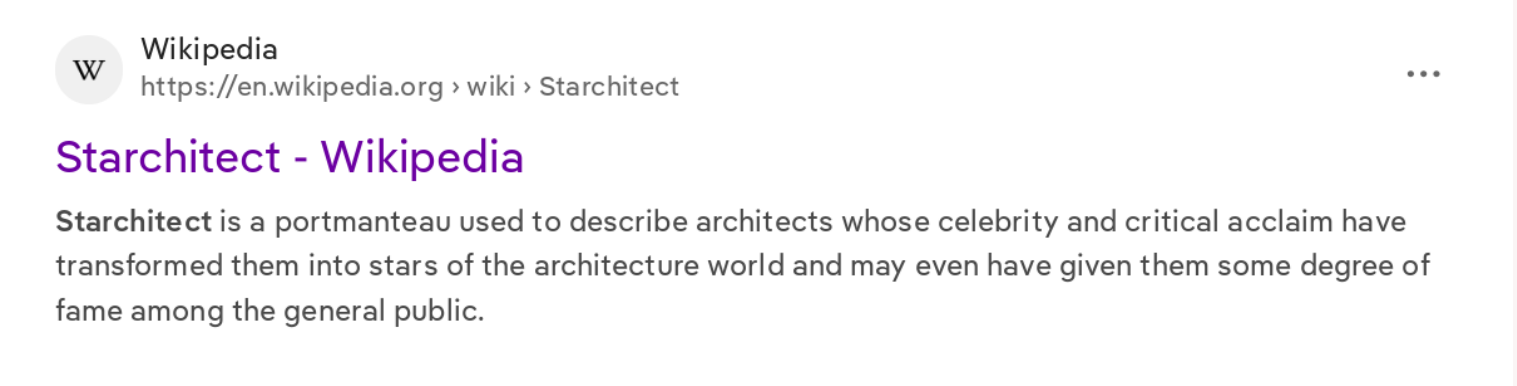
Photo by Marvin Damschen

Make it easy to do the right thing.

The code is the architecture

Systems designed in an ivory tower are unlikely to survive contact with reality.

Starchitects exist



..but (Structural) Engineering is a team effort

Empirical knowledge

Have
Postmortems!



Photo by Helmut Krcal (slightly modified by User:Springbank: Levels, Color, Contrast) - [1], CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3216104



Photo by Secretaría de Gestión Integral de Riesgos y PC CDMX - Secretaría de Gestión Integral de Riesgos y PC CDMX (https://twitter.com/SGIRPC_CDMX/status/1389631360307011586/photo/2)License: https://www.proteccioncivil.cdmx.gob.mx/terminos-y-condiciones, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=104902508

Design deliberately.
Fail predictably.
Learn collectively.
Iterate mercilessly.

Thank you!

Mail : katharina@damschen.se

Blog : katharina.damschen.net
Business: damschen.se

Mastodon: mastodon.nu/@katharina
LinkedIn: katharina-damschen



This presentation was developed during my employment at factor10