The role of **data** in **Green Software**

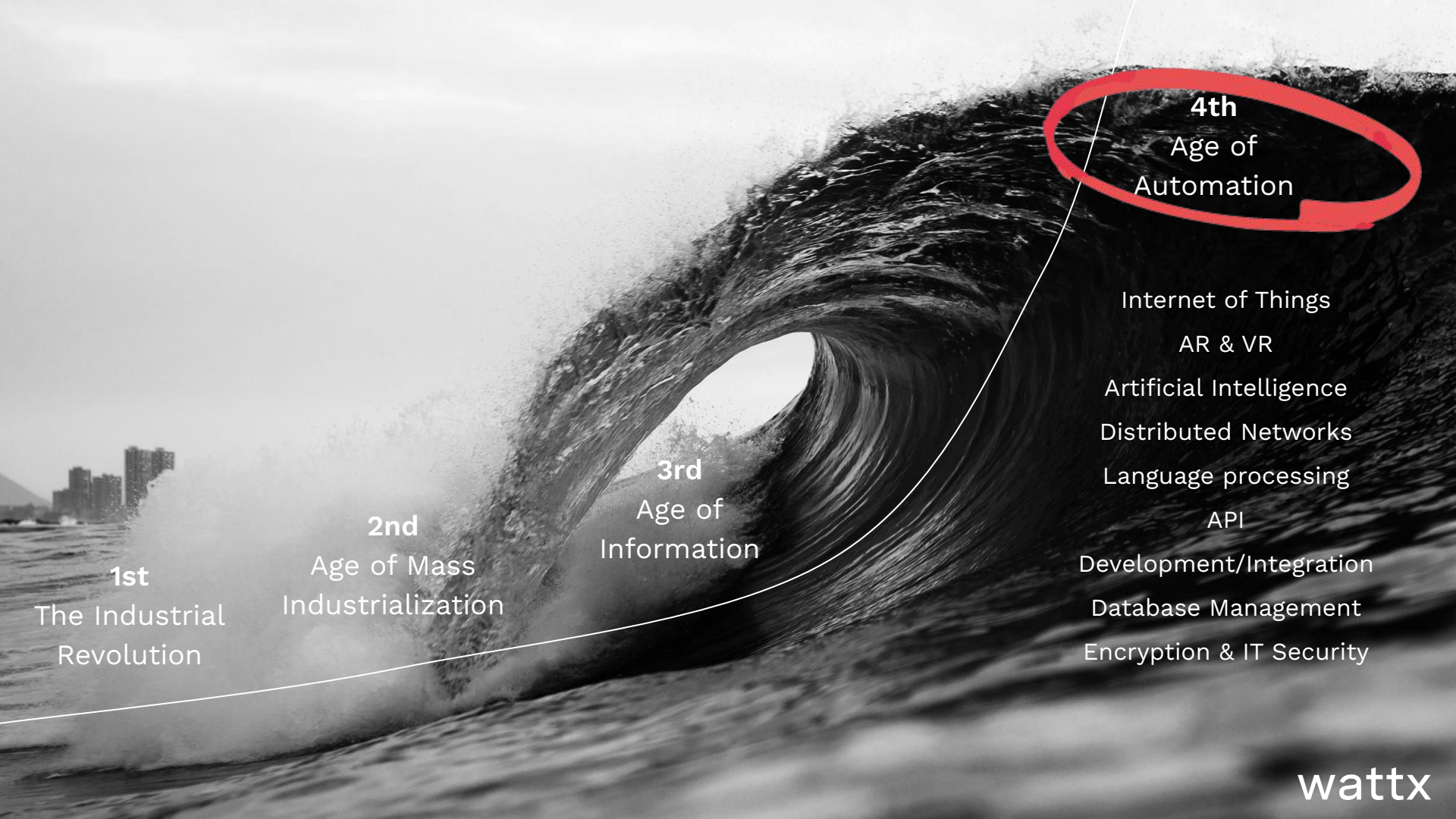
A challenge for every business

wattx

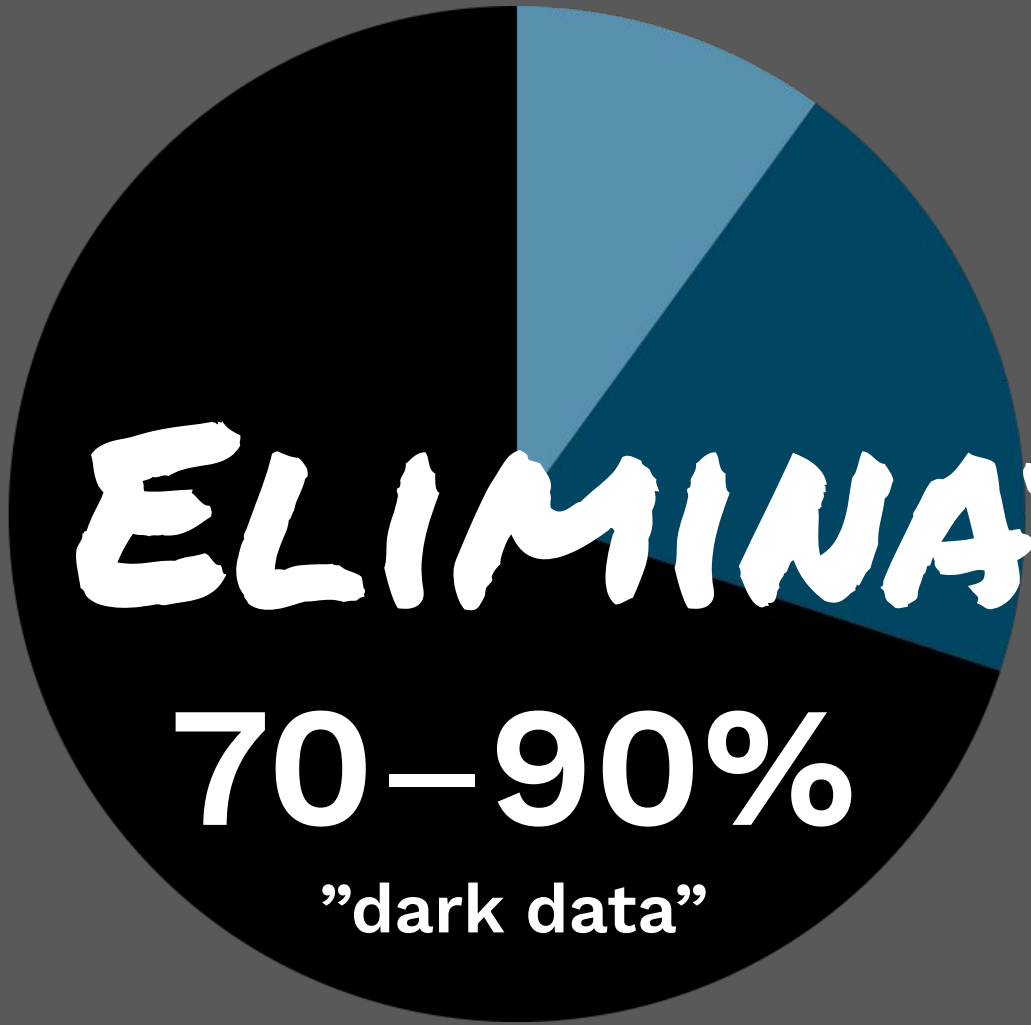

Data is the new oil!

wattx


~200 TWh
wattx


1st
The Industrial Revolution
2nd
Age of Mass Industrialization
3rd
Age of Information
4th
Age of Automation
Internet of Things
AR & VR
Artificial Intelligence
Distributed Networks
Language processing
API
Development/Integration
Database Management
Encryption & IT Security
wattx

1. Eliminate
2. Utilize
3. Allocate

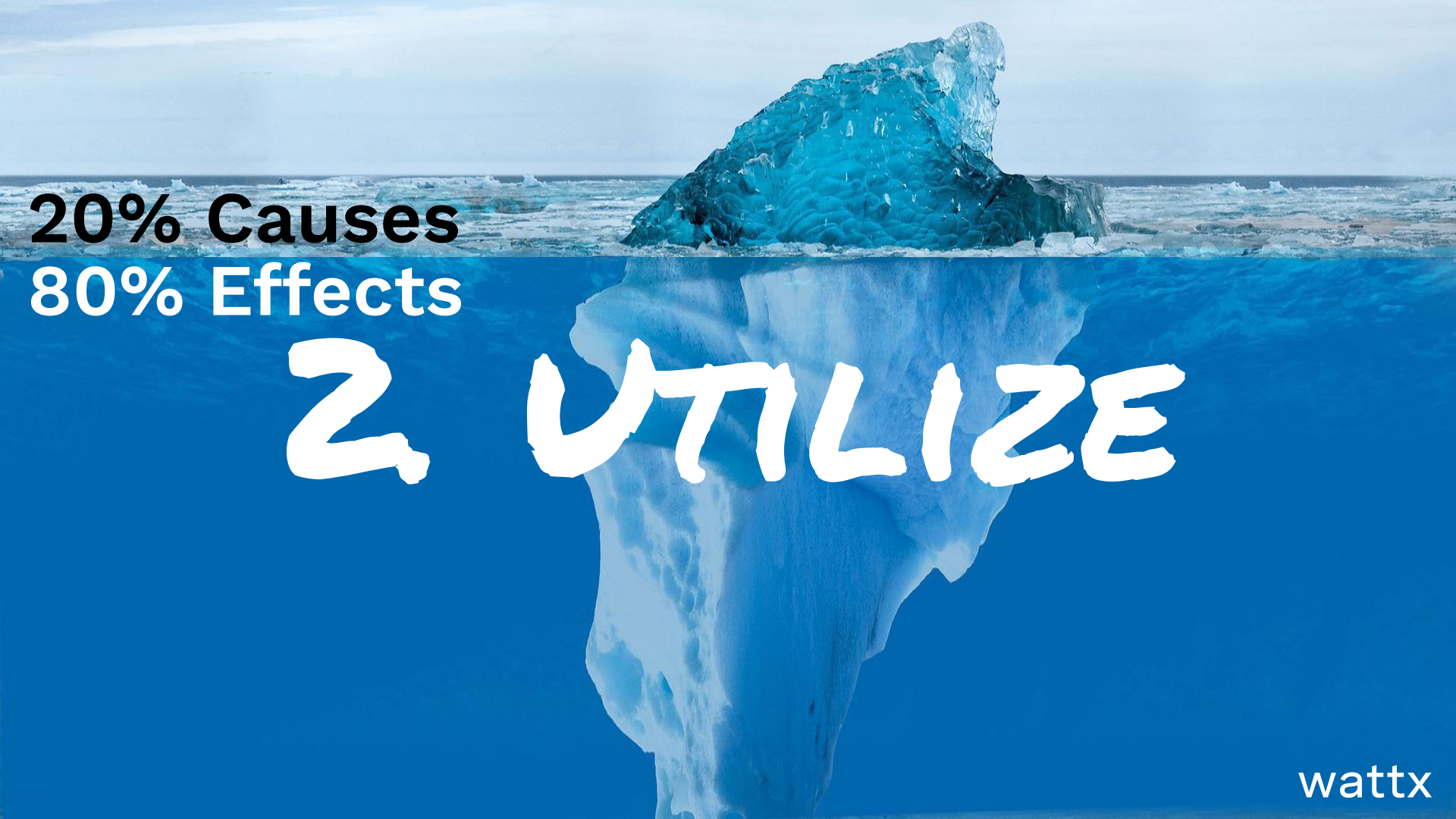


1. ELIMINATE

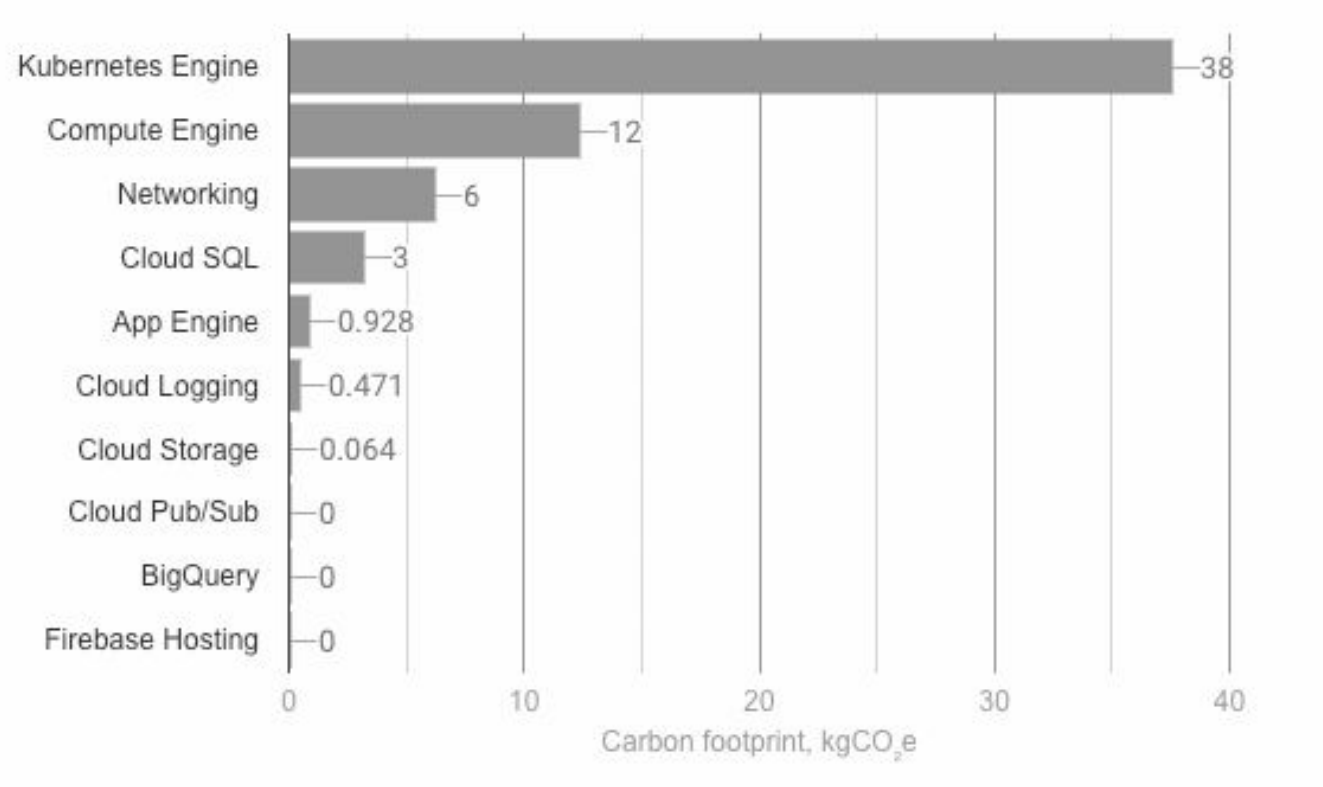
70–90%

”dark data”

wattx


20% Causes
80% Effects
2. UTILIZE
wattx


3. ALLOCATE
wattx



Source: GCP Console

wattx

Carbon data across GCP regions

Google Cloud Region	Location	Google CFE%	Grid carbon intensity (gCO_2eq/kWh)	Google Cloud net operational GHG emissions	
europe-central2	Warsaw	20%	576	0	
europe-north1	Finland	91%	127	0	Low CO_2
europe-southwest1	Madrid	*	121	0	Low CO_2
europe-west1	Belgium	82%	110	0	Low CO_2
europe-west2	London	57%	172	0	
europe-west3	Frankfurt	60%	269	0	
europe-west4	Netherlands	53%	282	0	
europe-west6	Zurich	85%	86	0	Low CO_2
europe-west8	Milan	*	298	0	
europe-west9	Paris	*	59	0	Low CO_2

Source: “Carbon free energy for Google Cloud regions”, Google

wattx

1. ELIMINATE

2. UTILIZE

3. AUTOMATE

FOR LEADS:

4. EDUCATE

Sustainability in Tech

Checklist for wattx projects

wattx

wattx

INSIGHTS

July 2022

Building Sustainable Digital Products

Marcin Balinski
Head of Engineering
wattx

Manuel Lopes
Venture Developer
wattx

wattx

wattx

Discuss 💬

Example: Languages & Algorithms

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Source: “Energy Efficiency across Programming Languages”, Pereira et al. 2017

wattx

The Computer Language Benchmarks Game

Node js versus TypeScript fastest programs

vs C++ vs Dart vs Java **vs TypeScript**

Always look at the source code.

These are only the fastest programs. Look at the other programs. They may seem more-like a *fair* comparison to you.

spectral-norm

source	secs	mem	gz	busy	cpu load
Node js	**1.68**	62,068	999	6.05	89% 90% 91% 90%
TypeScript	5.38	32,860	441	5.43	100% 0% 0% 0%

JavaScript:

source code

```
/* The Computer Language Benchmarks Game
   http://benchmarksgame.alioth.debian.org/

   contributed
*/

const cluster = require('cluster');
const numCPUs = require('os').cpus().length * 2;
var fs = require('fs');

const d = parseInt(process.argv[2])
```

TypeScript:

source code

```
/* The Computer Language Benchmarks Game
   http://benchmarksgame.alioth.debian.org/

   direct transliteration of Greg Buchholz's C program
   contributed by Isaac Gouy
*/

/// <reference path="./Include/node/index.d.ts" />

const w = +process.argv[2]
const h = w

let bit_num = 0, i = 0, byte_acc = 0
```

Source: “The Benchmarks Game”

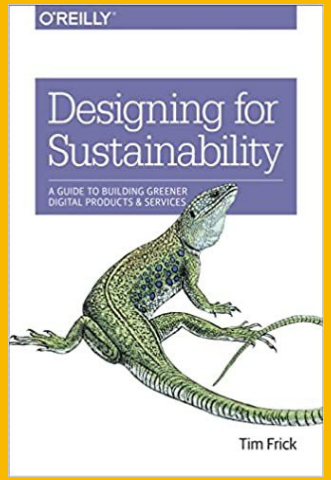
wattx

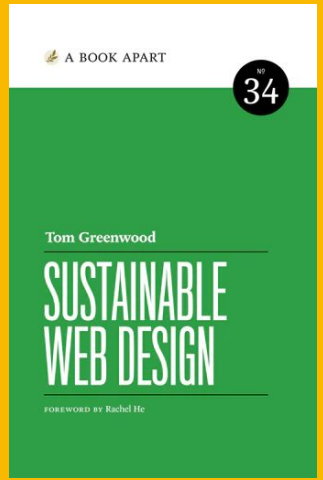
More insights from...

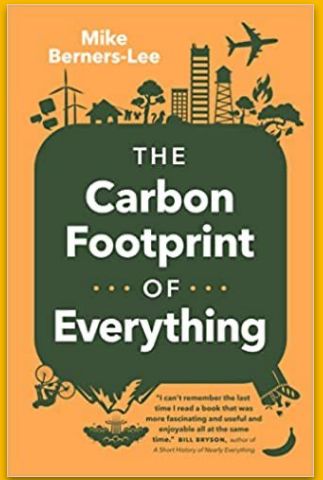
Communities

- 👩‍💻 climateaction.tech
- 📡 greensoftware.foundation
- 🤖 climatechange.ai
- 💬 lfca.earth
- 👩‍🎨 climatedesigners.org

Books







wattx

1. ELIMINATE
2. UTILIZE
3. ALLOCATE
4. EDUCATE

Data is the new oil!

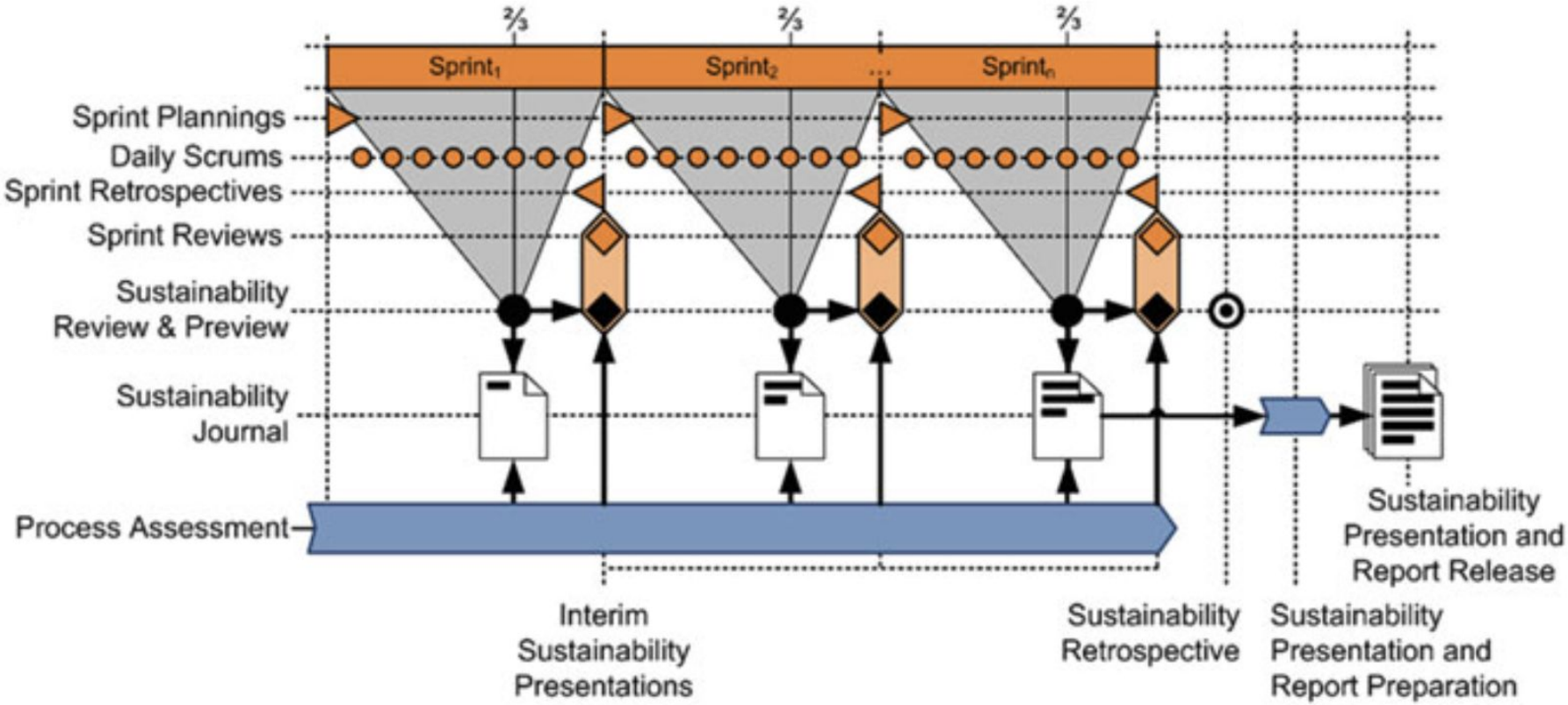
Let’s NOT just build refineries again!

Talk to me about details!

Dr. Simon Müller
CTO & Managing Director

simon@wattx.io

wattx



Source: “Processes for Green and Sustainable Software Engineering”, Kern et al. 2015

wattx