Embracing Simplicity

* Min res 56x16 char

1

Why?

Embrace simplicity as

- reduction of complexity
- a way to save resources
- human reaction to "AI"

How?

Embrace simplicity by

- learning from the pioneers
- using simple systems/code
- building your own tools

Reducing Complexity

Structure your system to

- make it easy to understand
- enable others to modify it
- keep simple changes local
- reduce cognitive load

* J. Ousterhout, A Philosophy of Software Design

Saving Resources

Hope for the best, prepare for the worst

- care for all hardware, especially the chips
- observe first, consider not doing, expose seams
- question scaling up, incomplete might work fine
- keep it flexible, build on solid ground, go bio

* https://permacomputing.net/principles

Staying Human

- "AI" tools generate infinite complexity
- original insight requires understanding
- prevent (non-intentional) dehumanisation

> "not only does AI risk inducing harm to the used-upon, but also to the user"

* L. Van der Gun, O. Guest, 2024 in Journal of Human-Technology Relations https://doi.org/10.59490/jhtr.2024.2.7272

Pioneers

Study the work of, among others

- David Parnas
- Niklaus Wirth & Jurg Gutknecht
- Dennis Ritchie & Ken Thompson
- Adele Goldberg, Dan Ingalls & Alan Kay
- Don Norman
- John Maeda

* ordered by personal preference

Modularity

Decomposing systems into modules

- D. Parnas wrote the classic paper on
- modularization, minimizing dependencies
- leading to flexible, comprehensible systems

* https://dl.acm.org/doi/10.1145/361598.361623

Project Oberon

Design & implement an entire system from scratch

- N. Wirth and J. Gutknecht developed Oberon
- a programming language with modules, exports
- an OS fully documented in less than 500 pages

* https://people.inf.ethz.ch/wirth/ProjectOberon/PO.System.pdf

Unix Philosophy

Make each program do one thing well

- D. Richie and K. Thompson developed C/Unix
- a programming language close to the metal
- an OS where almost everything is a file

```
$ ls | wc -l
```

* https://en.wikipedia.org/wiki/Unix_philosophy

Powerful Primitives

Build with a minimum set of general parts

- D. Ingalls implemented the Bit blit operation
- enabling popups and overlapping windows
- paving the way for modern GUIs

* https://www.cs.virginia.edu/~evans/cs655/readings/smalltalk.html

Object Orientation

Iterate, prototype & evolve towards understanding

- A. Goldberg, D. Ingalls & A. Kay made Smalltalk
- introducing the world to object orientation
- everything is an object (even classes)
- user can "change a running system"

* https://spectrum.ieee.org/qa-adele-goldberg-on-the-legacy-of-smalltalk

User-centered Design

The design of everyday things

- D. Norman wrote this classic on product design
- simplify task structure, make things visible
- get the mapping right, exploit constraints
- design for error, understand affordances

* ISBN 978-0-465-06710-7

Laws of Simplicity

Subtracting the obvious, adding the meaningful

- J. Maeda identified these "laws of simplicity"
- thoughtful reduction, organization, time saved
- knowledge, differences (vs.), context (focus)
- more emotions, trust, failure (to simplify)

* https://lawsofsimplicity.com

Using Simple Systems/Code/Ideas

Favourites from the past 30 years

- the Web, i.e. HTTP over TCP/IP
- Oberon.Sdb, a simple database
- Yaler, for secure remote access
- IoT Bricks, for IoT prototyping

* ordered chronologically

Getting Rid of the Save Button

Oberon.Sdb – a simple DB for PDAs

- .NET CF, 2005, ~150 lines of code
- serialize operations, append-only
- load from file, replay everything
- compaction "not yet implemented"

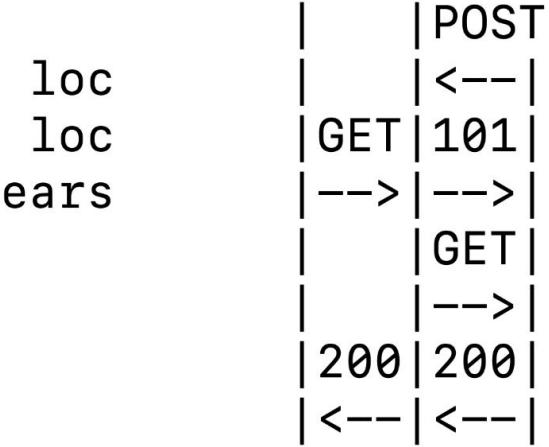
* e.g. http://www.tamberg.org/po

Reversing HTTP Requests

Yaler – access devices from the Web

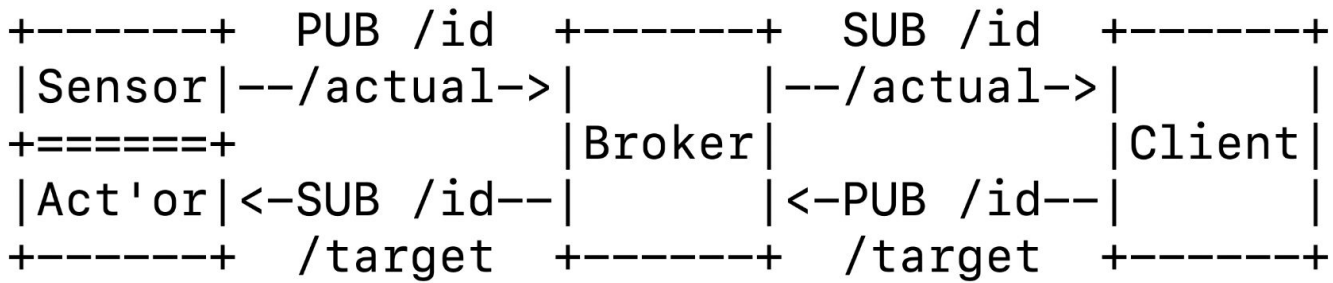
- relay server in Java, ~3.7k loc
- portable daemon in C, ~3.3k loc
- paying customers since 15 years
- highly reliable and stable

* https://github.com/yaler



Moving Logic to the Client

Orchestrate IoT devices using glue code



* https://github.com/tamberg/fhnw-iot

Prototyping Room-scale IoT Apps

```
var proxy = MqttProxy.fromConfig(CFG_URI);
var button = ButtonBrick.connect(proxy, BTN_ID);
var buzzer = BuzzerBrick.connect(proxy, BZR_ID);

while (true) {
    var value = button.isPressed();
    buzzer.setEnabled(value);
    proxy.waitForUpdate();
}
```

* https://github.com/tamberg/fhnw-iot-bricks

Building Your Own Tools

Less dependencies, more joy :)

- a static link shortener on S3
- a text to braille converter
- a simple idea generator
- a basic slide stepper

* not "the", just "a"

Shortening Links With a Hack

Client-side redirects in static HTML

- add meta header to HTML hosted on AWS S3
- server returns /id => /id/index.html
- client loads HTML, refreshes url

```
<meta http-equiv="refresh" content="0; url=...">
```

* e.g. tmb.gr/0 => http://www.tamberg.org

Converting Text to Braille

Some characters are grouped/contracted

- braille.c, 640 loc, no dependencies
- parser is a 250 loc if-statement
- using a simple state machine

```
$ ./braille -scad "text" > file.scad
```

* https://github.com/tamberg/braille

Generating Input for Ideation

Pick a random line each from n files

- idea.c, 70 loc, no dependencies
- file with options per category

```
$ echo "cat\ncow\ndog" > animal
$ echo "walk\nrun\ndance" > activity
$ ./idea animal activity
```

* https://github.com/tamberg/idea

Presenting From a Text File

Step through slides, in a terminal

- step.c, 66 loc, no dependencies
- slides.txt, + as page separator
- like the cat tool, plus paging

```
$ ./step slides.txt
```

* https://github.com/tamberg/step

Thanks

thomas.amberg@fhnw.ch

* tmb.gr/smp => simplicity.txt