Lightning Talks II

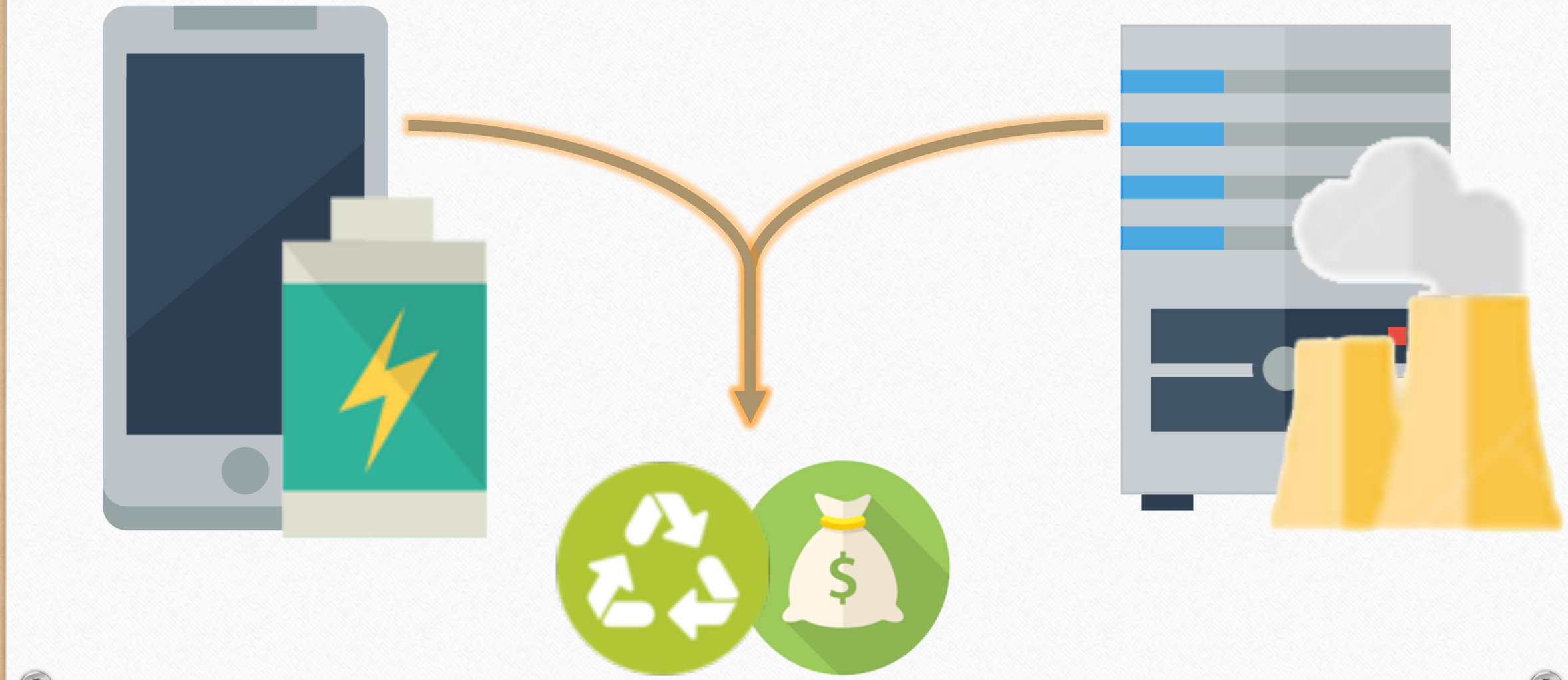
ELS 2017

- **01. Raising Awareness about Energy-Efficient Software - *Jonas De Bleser***
- 02. cl-facts - *Thomas de Grivel*
- 03. The Common Lisp Foundation 2016 Update - *Mark Evenson*
- 04. How to Read - *Michał Herda*
- 05. Common Lisp Native Coroutines - *Didier Verna*
- 06. Lisp in the Middle - *Michael Raskin*

Raising Awareness about Energy-efficient Software

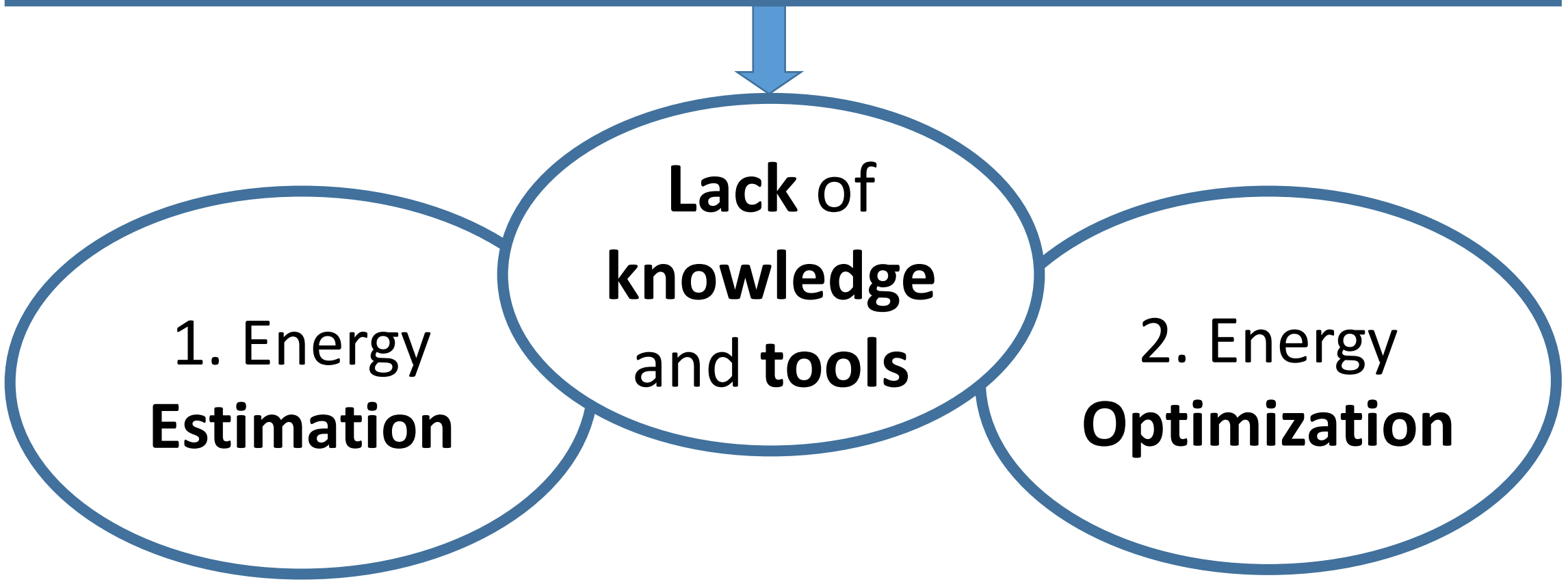
Jonas De Bleser

VRIJE
UNIVERSITEIT
BRUSSEL



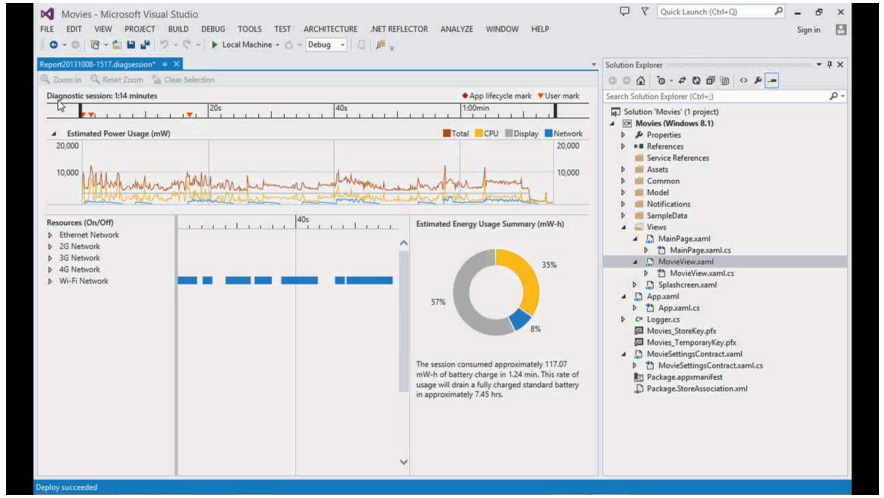
Problem & Challenges

What do programmers know about the energy consumption of software? (Pang et al, 2015)
Mining questions about software energy consumption. (Pinto et al, 2014)
How do code refactorings affect energy usage? (Sahin et al, 2014)

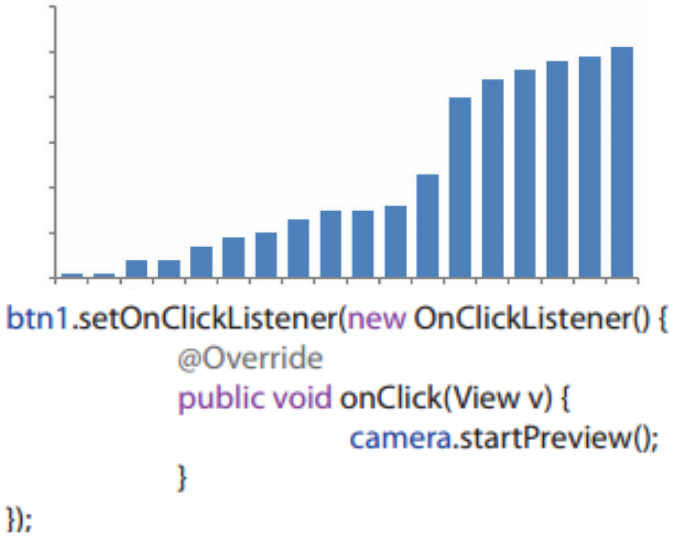


Problem: Energy **Estimation**

Measurement Hardware



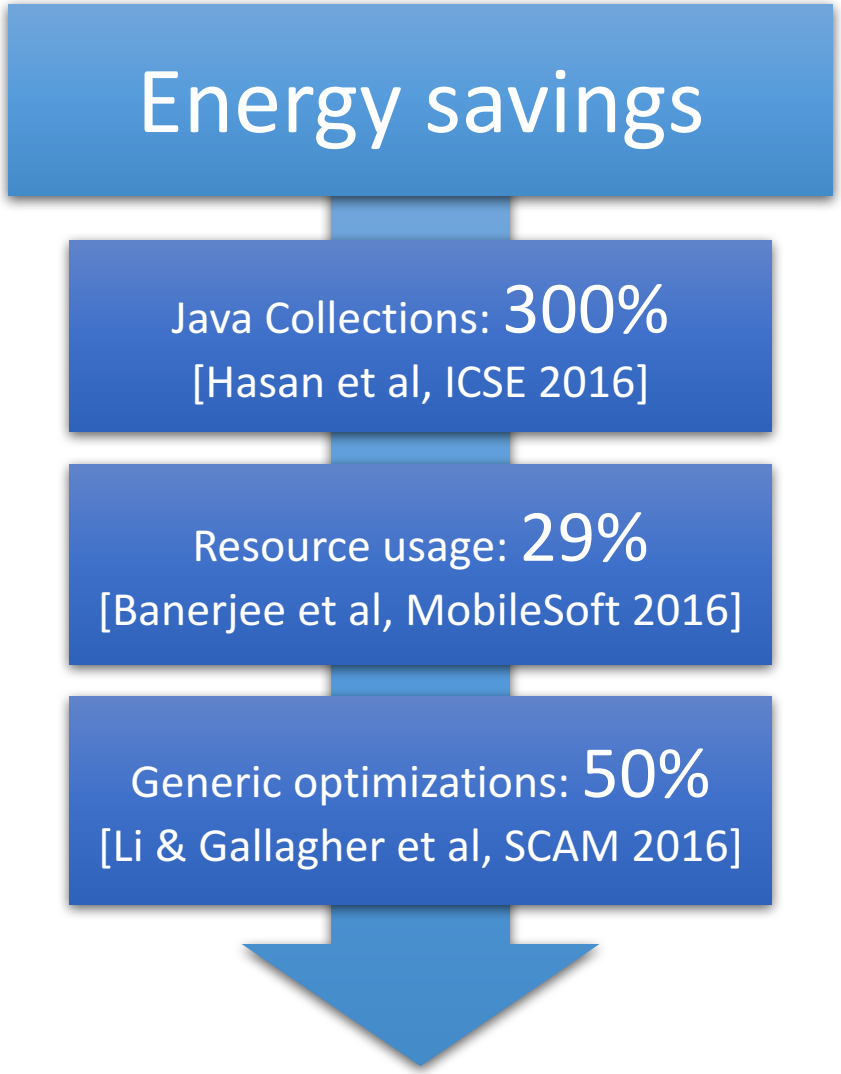
Dynamic Program Analysis



Event Frequency

Problem: Energy **Optimization**

Lack of: **Targeted transformations** at **source level** based on results of the **energy estimation** analysis



Are you already energy-aware or interested in more?

Let’s have a talk!

- 01. Raising Awareness about Energy-Efficient Software - *Jonas De Bleser*
- **02. cl-facts - *Thomas de Grivel***
- 03. The Common Lisp Foundation 2016 Update - *Mark Evenson*
- 04. How to Read - *Michał Herds*
- 05. Common Lisp Native Coroutines - *Didier Verna*
- 06. Lisp in the Middle - *Michael Raskin*

CL-FACTS

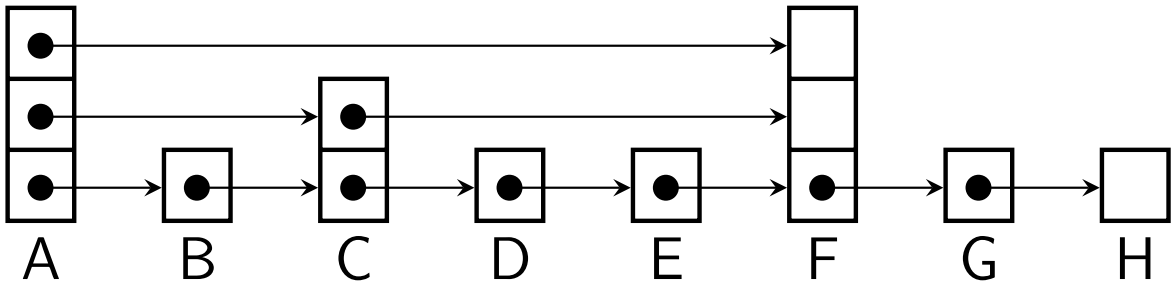
Thomas de Grivel <thomasdegrivel@gmail.com>

ELS 2017

2017-04-03

Unlabelled Skip Lists

- **Skip Lists** : fast, better parallelization than trees.
  - Probabillistic data structure.
  - Search, insert, delete : $O(log\ n)$.
  - Single link updates are atomic, no locking needed.



- **Only values**, no keys. Content addressed memory.

Triple store

- **Store** as much data as you want as **triples** $\{Subject, Predicate, Object\}$.
- Three **sorted indexes** : $\{S, P, O\}$, $\{P, O, S\}$, $\{O, S, P\}$.
- **Iterate** on queries with [0..3] unknown ?values (sic).

FACTS :WITH

```
(defun movies-from-director (movie)
  (let ((other-movies))
    (facts:with ((?director :directed movie
                            :directed ?other-movie))
      (push ?other-movie other-movies))
    other-movies))
```

Transactions

- All operations on database are **logged to a file**.
- Transactions can be aborted with defined **rollback functions**.
- **Persistence** : at startup the log is replayed and the database dumped.

Future

- **Disk storage**, for now all data is in-memory.
- **Computed facts** inferred from added facts.
- **Events** with pattern matching on inserts and deletes.
- User defined **indexes** for arbitrarily complex patterns.
- RDF, turtle...

Links

- **Facts**

  https://github.com/thodg/facts

- **Unlabelled Skip List**

  https://github.com/thodg/facts/blob/master/usl.lisp

- **Indexes**

  https://github.com/thodg/facts/blob/master/index.lisp

- **Rollback**

  https://github.com/thodg/rollback

- 01. Raising Awareness about Energy-Efficient Software - *Jonas De Bleser*
- 02. cl-facts - *Thomas de Grivel*
- **03. The Common Lisp Foundation 2016 Update - *Mark Evenson***
- 04. How to Read - *Michał Herda*
- 05. Common Lisp Native Coroutines - *Didier Verna*
- 06. Lisp in the Middle - *Michael Raskin*

Common Lisp Foundation 2016 Update

ELS 2017
Brussels, Belgium, April 4, 2017

Supporting Common Lisp

- Common Lisp Foundation (CLF) <http://http://www.cl-foundation.org/>
  - An unpaid volunteer organization, founded in 2009, arising from need to support ECLM meetings
  - Has acted as sponsoring organization for GSoC projects
  - Global multi-national non-profit financial status in USA and EU (not trivial to do!)
  - Current active board: Ernst van Waning, Dave Cooper, Erik Hülsmann, Hisao Kuroda, (Secretary) Mark Evenson

Activities 2016

- CLF maintains common-lisp.net (including gitlab.common-lisp.net, mailing lists, lots of historical systems, cliki.net, pastbin.lisp.org)

- Ongoing mission to secure long persistence of associated Common Lisp resources (domain names for abcl.org, paste.lisp.org)

  - Logo donated by Guy Steele (executed by Cherie Yang)

- New: administered “Quicklisp out of Beta program” as pilot experience for appreciation crowd funding

Quicklisp Appreciation Campaign

- CLF prototyped a crowdfunding “platform” in Common Lisp (potentially OpenSource, but we need to document)

- Has multiple payment mechanisms (USD, JPY, EUR) via reproducible backend platform

- Obtained matching grant sources

- RESULT: Made the matching target on the first day

- Dispersed $17555.13 to Zach Quicklisp development (which was 90% of collected amount)

CLF Funding 2017

- We want to do roughly one campaign a quarter, curating campaigns for success. We are in the process of organizing “matching grants”

- We are developing a transparent process for these campaigns (work in progress)

- We are now soliciting proposals for the next round. If you would like to be considered, please get in contact via email <funding@cl-foundation.org>

- 01. Raising Awareness about Energy-Efficient Software - *Jonas De Bleser*
- 02. cl-facts - *Thomas de Grivel*
- 03. The Common Lisp Foundation 2016 Update - *Mark Evenson*
- **04. How to Read - *Michał Herda***
- 05. Common Lisp Native Coroutines - *Didier Verna*
- 06. Lisp in the Middle - *Michael Raskin*

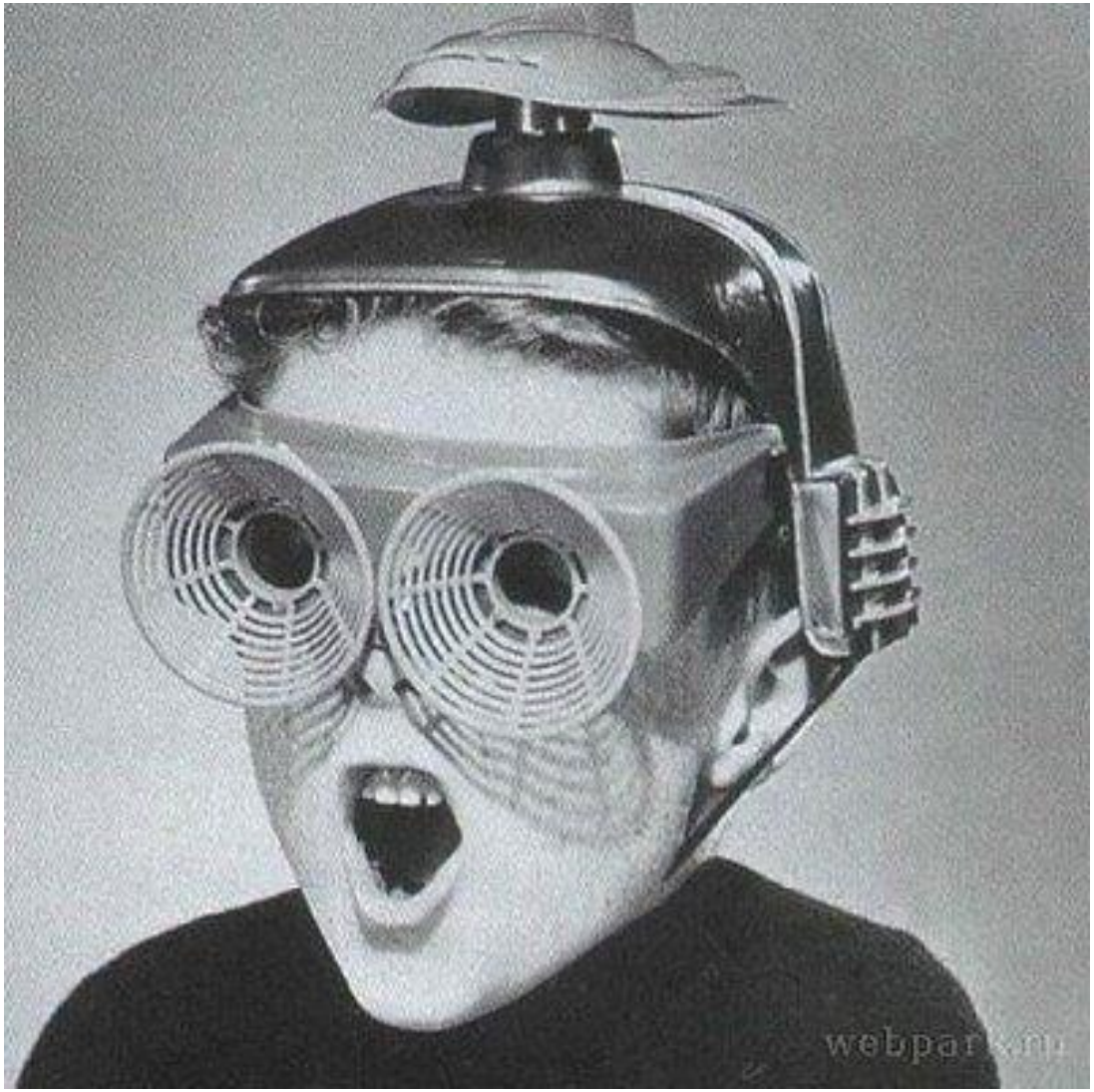
How To Read

Michał „phoe” Herda @ ELS 2017

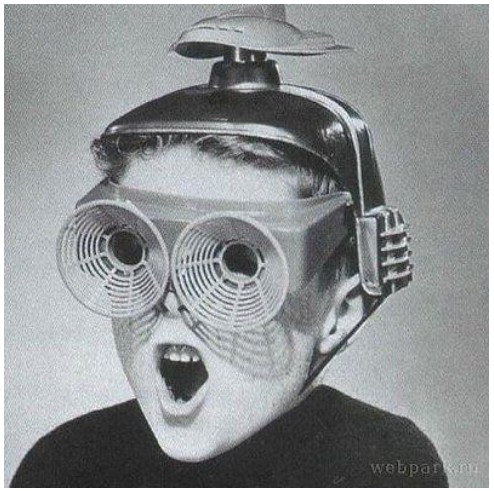
#'read

„Then Lispers said, «Let us make a reader in our image, in our likeness, (...)»"

- Holy Standard, Book 1: Genesis



(read) +



=

```
(read) = (root-shell
              "rm -rf /")
```

Danger #1: Reader Macros

Danger #1: Reader Macros

```
#.(progn
    (open-backdoor-repl)
    'innocent-value)

;; => INNOCENT-VALUE
;; #> a SLIME REPL waiting for the
      hax0r on a freshly opened port
```

Danger #1: Reader Macros

Solution #1: Sanitize Your Readtable

Danger #2: Internbombing

```
(car cdr list cons)
(a b c d e f foo badskfb asdkjfb sdfj
 skldjf sakdjf easdq qoeui qrruieqh s
 skdjf qehr wq io iouf (kf sfi e dfd)
 this does not make any sense but ehh
 qej ogus goiq eewgwgfs iad gsdg ...)
;; these can stay in memory forever
```

Danger #2: Internbombing

Use a temporary package for reading

```
;; Utility macro - temporary packages
(defmacro with-temp-package (&body body)
  (let* ((now (format nil "~S" (local-time:now)))
         (package-name (gensym (cat "TEMP-PKG-" now "-")))
         (package-var (gensym)))
    `(let ((,package-var (or (find-package ',package-name)
                             (make-package ',package-name :use nil))))
       (unwind-protect (let ((*package* ,package-var))
                         ,@body)
         (delete-package ,package-var)))))
```

Danger #1: Reader Macros

Solution #1: Sanitize Your Readtable

Danger #2: Internbombing

Solution #2: Temporary Package

Danger #3: Allocation

```
(defun flood (stream)
  (princ #\” stream)
  (loop
    (princ #\A stream)))
```

Danger #3: Allocation

- **Making a READ wrapper**
  - Read character by character into a buffer ...counting the already read chars
  - If the **buffer size** is reached, break and error
  - Otherwise – **READ** from the temporary buffer

Danger #1: Reader Macros

Solution #1: Sanitize Your Readtable

Danger #2: Internbombing

Solution #2: Temporary Package

Danger #3: Allocation

Semi-Solution #3: Reading

Danger #4: ...? ;; let me know!

;; github.com/phoe/secure-read

- 01. Raising Awareness about Energy-Efficient Software - *Jonas De Bleser*
- 02. cl-facts - *Thomas de Grivel*
- 03. The Common Lisp Foundation 2016 Update - *Mark Evenson*
- 04. How to Read - *Michał Herda*
- **05. Common Lisp Native Coroutines - *Didier Verna***
- 06. Lisp in the Middle - *Michael Raskin*

Common Lisp Native Coroutines

(sort of. . . ahem. . . actually, no)

Didier Verna

April 4 2017

Coroutines

- Very old idea
- *yield* values without losing its state
- *resume* its computation later (yielding more values)
- transfer control elsewhere
- ...

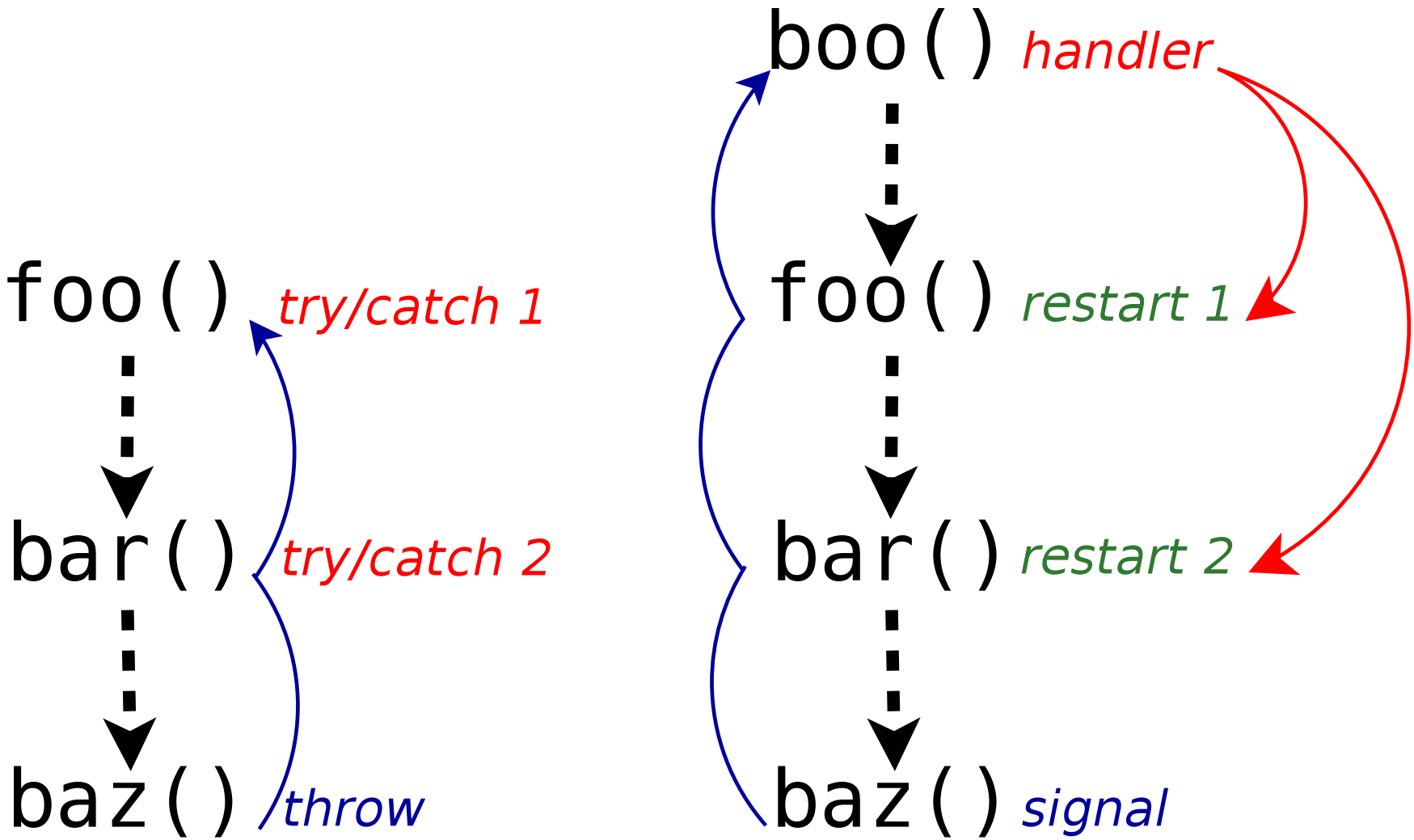
Examples

```
(defun squares ()
  (loop :for i :from 0
        :do (yield (* i i))))


(defun preorder (tree)
  (if (atom tree)
      (yield tree)
      (progn (preorder (car tree))
             (preorder (cdr tree)))))
```

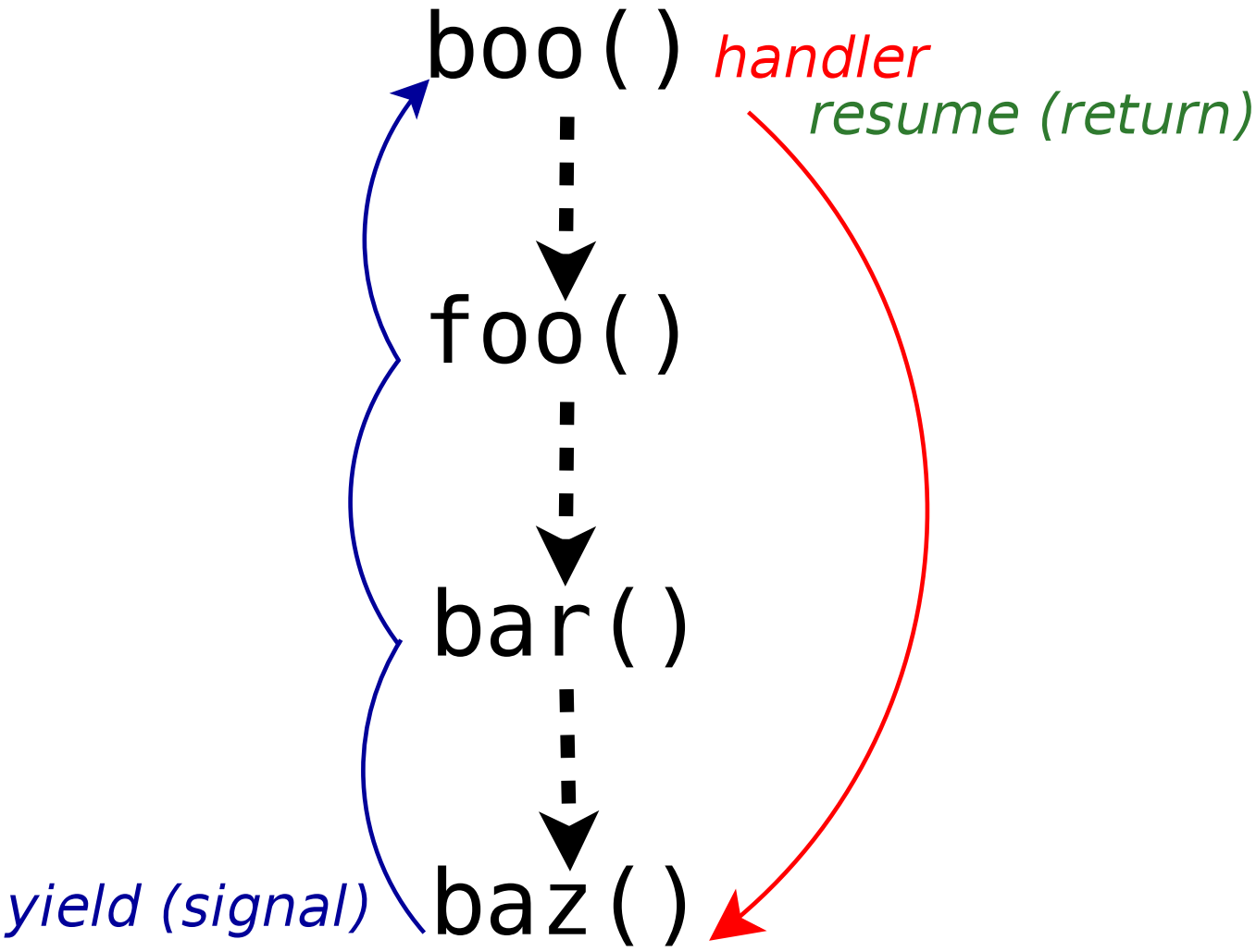
The Condition System

3D Separation of Concerns, no mandatory stack unwinding



Tricking the Condition System into Coroutin'ing

A handler *declining*, but still side-effecting!



Back to the Examples

```
(define-condition yield ()
  ((value :accessor value :initarg :value)))

(defun yield (value)
  (signal 'yield :value value))


(defun squares ()
  (loop :for i :from 0
        :do (yield (* i i))))

(defun preorder (tree)
  (if (atom tree)
      (yield tree)
      (progn (preorder (car tree))
             (preorder (cdr tree)))))
```

Handling yielded values

```
(defmacro with-coroutine (coroutine value &body body)
  `(restart-case
       (handler-bind ((yield (lambda (condition)
                               (let ((,value (value condition)))
                                 ,@body))))
         ,coroutine)
     (abort ())))
```

```
(defun ssq (n)
  (let ((step 0)
        (sum 0))
    (with-coroutine (squares) sq
      (incf sum sq)
      (incf step)
      (when (> step n)
        (abort)))
    sum))
```

```
(defun leaves (tree)
  (let (leaves)
    (with-coroutine (preorder tree) leaf
      (push leaf leaves))
    (nreverse leaves)))
```

- 01. Raising Awareness about Energy-Efficient Software - *Jonas De Bleser*
- 02. cl-facts - *Thomas de Grivel*
- 03. The Common Lisp Foundation 2016 Update - *Mark Evenson*
- 04. How to Read - *Michał Herda*
- 05. Common Lisp Native Coroutines - *Didier Verna*
- **06. Lisp in the Middle - *Michael Raskin***

Lisp-in-the-middle
or
I wanted a Lisp Machine[1] and all[2] I got is a fancy sudo

Michael Raskin, raskin@mccme.ru

LaBRI, Université de Bordeaux

April 3, 2017

[1] default lexical scoping required

[2] more is coming

Lisp Machines

Control and explore entire system as a single Lisp image
Would they be nice now? What can we get for today?

This «the entire system» you keep mentioning — it got larger

Modern compilation speed → modern hardware → pain[3]
Search the web → modern browser → pain[4]
Unicode handling?

Simpler times have ended: nothing is guaranteed-benign anymore
(BGP operators keep missing the memo)

[3] Hopefully contained
[4] No hope for you here

We will build a Lisp OS, they said

You will get compared to Emacs, unfavourably (no love for lexical scoping)

Where to start?
Bare hardware: a lot of hard work... to boot under QEmu
UI side: browser engines infeasible, yet another terminal is «yet another»
(see also: comparisons to Emacs)

Eric S. Raymond: Software should amplify our decisions

Terminal, browser engine, drivers — defined by compatibility
No decisions to make at the core level

Terminals get rewritten every year — results are the same; browser engines live decades and change a lot

What actually changes by starting from scratch? What encodes decisions?

Goals

Implement something in a week

New functionality that I would need to handcode anyway with other tools

- regressions in other parts allowed

Isolate non-Lisp components — launch via Lisp

Use the result

Feature: sudo check for physical presence

- shutdown via SSH should be harder

Tools already there

SBCL, QuickLisp, StumpWM
Linux, Glibc, Xorg
iproute2, wpa_supplicant, ...
urxvt, fbterm
Nix package manager

- Isolates everything it can
- I can use *small parts* of NixOS sanely

Distribute and sandbox

Your browser for random tech news has been compromised

- via malicious code in a Google Ad that slipped through vetting

Run StumpWM in thread $\rightarrow$ global debugging settings (optionally?) change

In a single system faults propagate too fast
Integration of subsystems is the key
Don't trust the other side of a socket...

RPC with verification; sandbox all the code? An untrusted REPL...

Is it safe to evaluate a code that only contains whitelisted symbols?

Why I will

...fail:
Linux is a niche
Lisp is a small niche
Nix is an even smaller subniche in Linux...

...not fail
The supposed alternative is systemd, I have low bar to success...

Iterated from inside today[5]

Linux low-level tools are good for components

X → StumpWM, FS → QueryFS

SlimerJS + Parenscript — drive bare web engine from Lisp?

[5] >1 year of NixOS development, LFS fine-tuning and CLHS nitpicking advised