Revenge of the Pragmatists

or, lessons learnt from running a Clojure startup

```
(info :BG)
=> {:name "BG",
    :govt-id "Baishampayan Ghose",
    :twitter "@ghoseb"}
```

whoami

- Background in Lisp, Distributed Systems, Information Retrieval
- Built Air travel booking system in Common Lisp & a Sport Social Network in Python
- Dissatisfied with mutable, inconsistent, crippled, slow languages
- Early adopter of Clojure in late-2008
- Co-founder & Ex-CTO of Helpshift, Inc.

Most business software is

- Automation of repetitive business processes
- Data processing heavy
- Potentially high-throughput (but *not* necessarily low latency)
- Multiple database technologies
- Cloud deployed
- Constantly changing requirements
- Held together with duct tape

The language I wanted

- Dynamically typed
- Lisp-like
- First-class functions
- Good library ecosystem
- Immutable data-structures
- I wanted the language to *get out of my way*

Discovering Clojure

- At first, I saw the J(ava) in *Clojure* and I ran
- Later, I found the jewels
  - Extremely well-designed and simple language
  - Designed to be practical
  - Experienced & welcoming community members
  - It's a Lisp!
  - The JVM is an excellent runtime host

But it was a risky bet!

- A <1 year old language
- Tooling was lacking (no Leiningen!)
- No powerful features like Protocols, **`clojure.spec`**, etc.
- It is a Lisp! Meh.

However, I was convinced enough to invest in Clojure. Now onto the bigger problem...

Businesses often don't choose technologies for their power of leverage, but for their risk mitigation properties.

— *Anonymous Coward, Ca. 2017*

Convincing Business Stakeholders

- Ease of hiring
- Industry reports
- Blog posts from competition
- Hacker News frenzy

Convincing Business Stakeholders

- ~~Ease of hiring~~
- ~~Industry reports~~
- ~~Blog posts from competition~~
- ~~Hacker News frenzy~~
- Authority
- Responsibility

Even if you want to fix just one thing, you may need to fix the whole system.

— *Anonymous Coward, Ca. 2017*

Recruiting a team

- Juniors don't know what to learn
- Mid-levels don't know how to learn
- Seniors don't want to learn
- May be go for the middle to start with then?

What didn't matter

- Degree
- Major
- Years of experience with **`foobar`**

What mattered

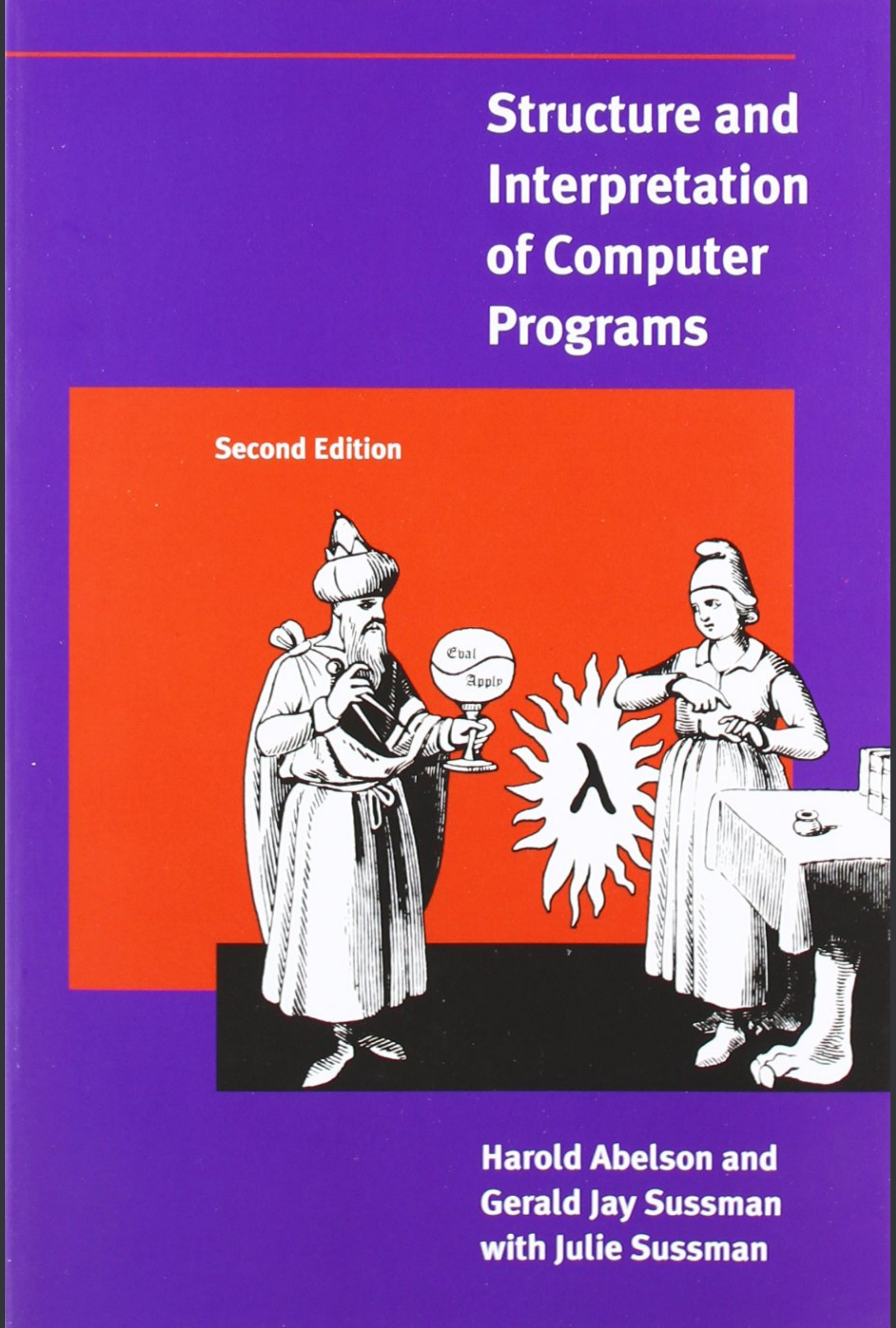
- Fundamentals
- Passion to do better
- Humility to accept feedback
- Patience to keep practising
- Culture-fit (more on this later)
- Recruiting in a sustained vs. burst mode

Don't *hire* teams, *grow* them.

— *Anonymous Coward, Ca. 2017*

Training

- Start with the book!
- Mentoring (especially marginalized groups)
- Bootcamps
- Conferences
  - Helped organise two, supported many
- Book reading groups
- Meetups



Structure and Interpretation of Computer Programs

Second Edition

Harold Abelson and Gerald Jay Sussman with Julie Sussman

Culture

- Culture is like scalable training, with feedback
- In case of conflict, culture is the arbiter
- Built through repeatable processes
    - Code reviews
    - Upfront design
    - Style guides, linter tools
    - Curated list of libraries (internal/external)

At **scale**, all problems become **people** problems.

— *Anonymous Coward, Ca. 2017*

Challenges Faced

- Clojure demands up-front design that's sometimes hard to justify
- Sometimes lack of good/tested libraries add to the pressure
- Language never tripped us up, sometimes had runtime issues
  - GC pauses
  - Memory leaks
  - Laziness
- Championing change

Lessons Learnt

- Small, dedicated teams *can* do a lot with FP & Clojure
- Functional Programming *does* help in building reliable software
  - Never had memory corruption or inconsistent state bugs
  - Concurrency and parallelism was simpler
- Functional Programming is ideally suited for data processing
- Estimation is still hard
- Must try ClojureScript

Clojure made me do this

- Parsers
- Interpreters, Compilers
- State Machines
- Lockless concurrency
- Monadic patterns
- Immutable everywhere
- Weild the power of the JVM (V8, etc.)

Big Wins

- Was able to assemble an amazing team (100+)
- Raised $40MM in funding till date
- SDK installed 2 Billion+ times
- Company went on to having 600MM+ monthly active users
- 50K+ requests per second
- Multi-datacenter, multi-tenant, SaaS
- 400K+ SLOC Clojure (largest repo had ~250K SLOC)

Clojure is the ideal language for pragmatic programmers with deadlines.

— *Anonymous Coward, Ca. 2017*

Some parting thoughts...

- Building software is unnecessarily difficult
- Distributed Systems == Distributed Problems
- **`Process : Clojure :: System : ???`**
- Developers need to learn Ethics
- Clojure will probably never be a TIOBE #1 language
  - But does it matter?

Stop coding. Let's build software!

— *Anonymous Coward, Ca. 2017*

Thank you!

Questions?

@ghoseb on Twitter & Github