Why simplicity?

Designing and Maintaining Software (DAMS)

Louis Rose

Habitable Software

Leaner

Less **Complex**

Loosely **Coupled**

More **Cohesive**

Avoids **Duplication**

Clearer

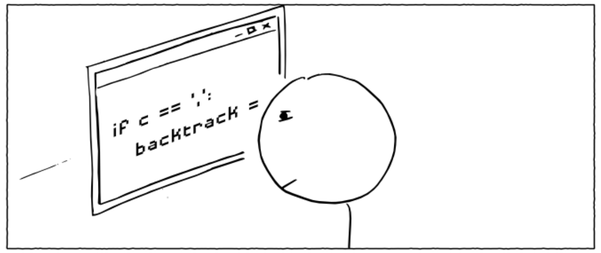
More **Extensible**

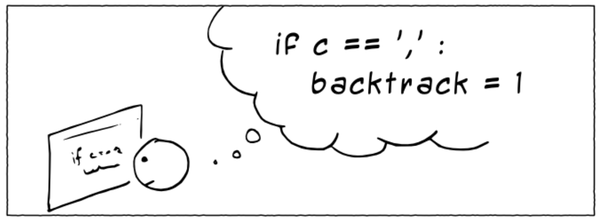
???

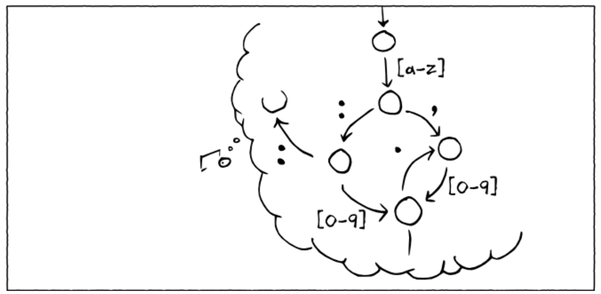
What is software complexity?

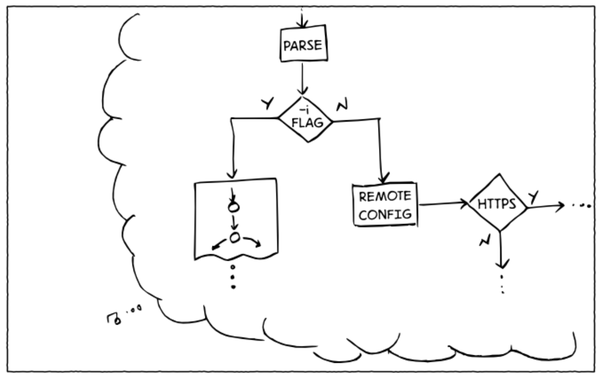
Partly about size (lean)

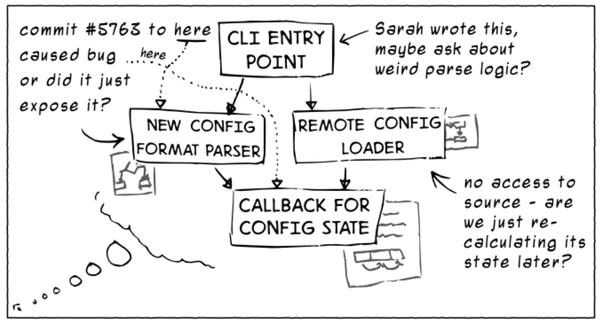
Partly about understandability (clarity)

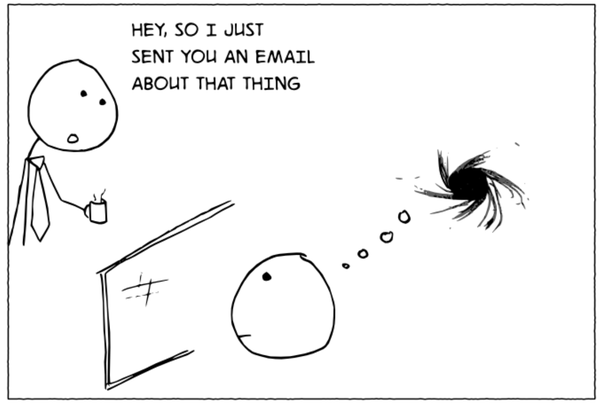


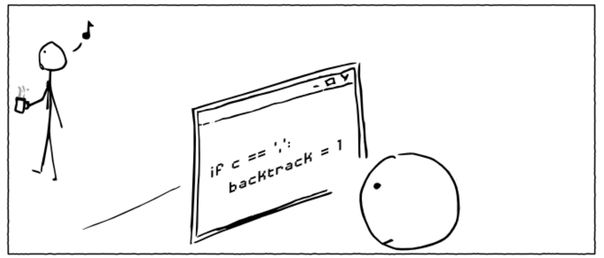












 heeris.id.au

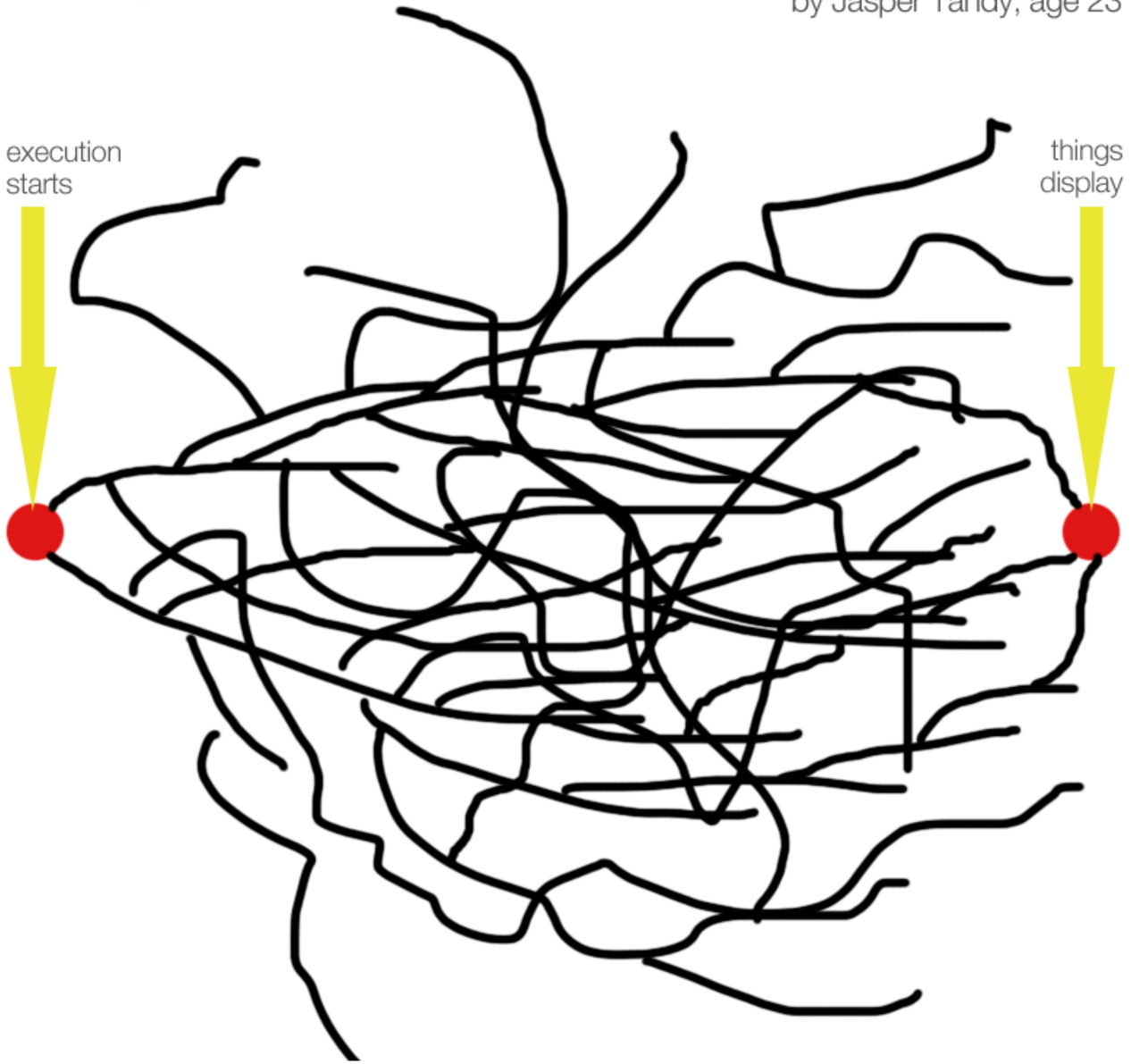
Some examples

A method with many statements, conditionals & loops
(control flow states and transitions)

A program which inputs, processes and outputs
lots of interrelated data

Spaghetti-code; A visualisation

by Jasper Tandy, age 23

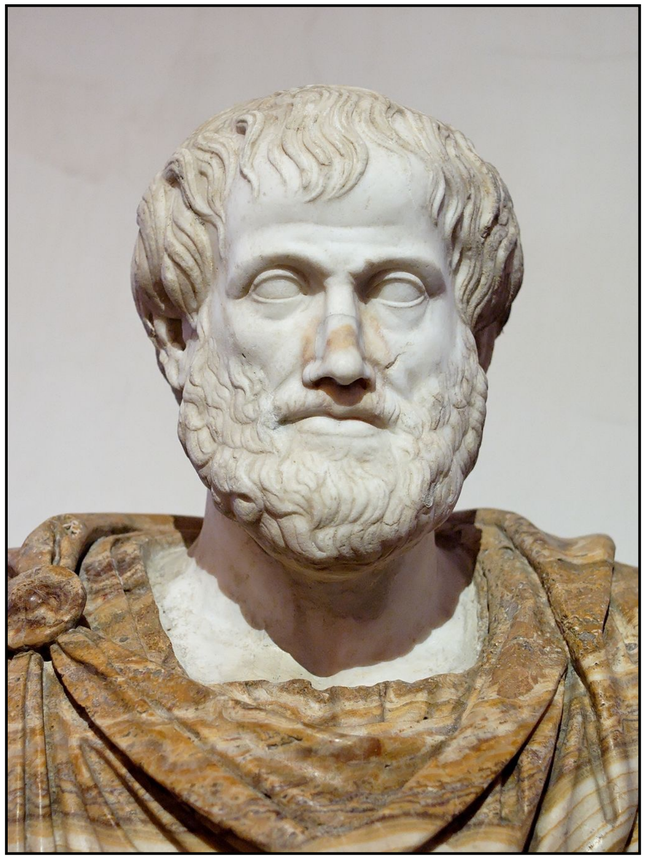


No Silver Bullet

“The very nature of software makes it unlikely that there will ever be [any silver bullets for software engineering]”

- Frederick P. Brooks, Jr.
The Mythical Man-month
Addison-Wesley, 1995

Essence vs Accident

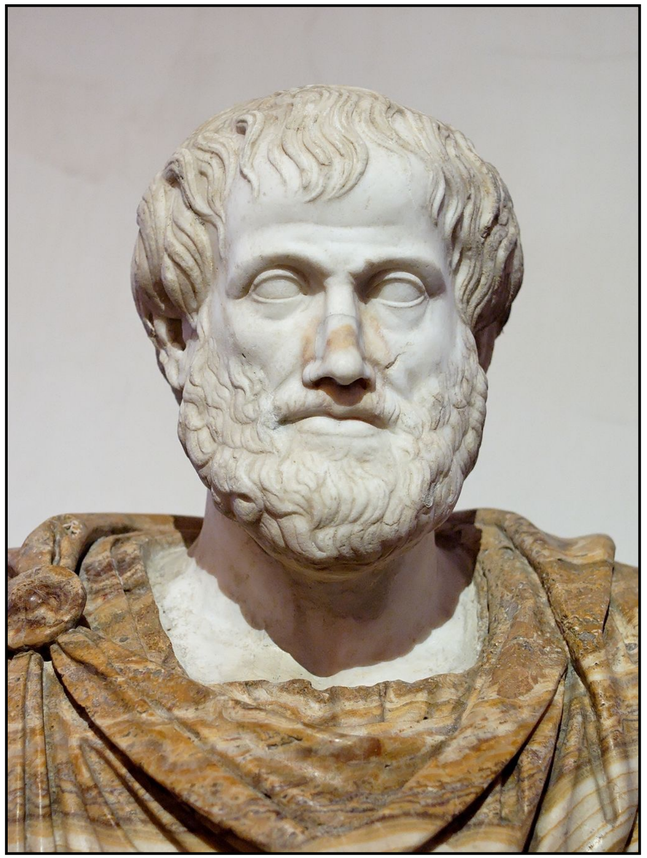


Aristotle
Taken from Wikipedia

Essence: attributes that are necessary for an entity to have its identity

Accident: attributes without which an entity can retain its identity

Essence vs Accident



Aristotle
Taken from Wikipedia

Essence: a chair must be “sitable” to be a chair

Accident: many chairs are (partially) wooden

Essential Complexity

Mostly unavoidable

Largely related to
“identifying what to build”

Accidental Complexity

Mostly avoidable

Largely related to
“doing the building”

Tackling accidents

Some tactics for reducing accidental complexity

Idea: Abstractions

Favour specifying the problem over
specifying the implementation

Example

```
def twitter_handles(authors, company)
  result = []
  authors.each do |a|
    if (a.company == company)
      handle = a.twitter_handle;
      result << handle unless handle.nil?
    end
  end
  result
end
```

http://martinfowler.com/articles/refactoring-pipelines.html#Identifiers

Example

```
def twitter_handles(authors, company)
  authors.select { |a| a.company == company }
         .map    { |a| a.handle }
         .reject { |h| h.nil? }
end
```

http://martinfowler.com/articles/refactoring-pipelines.html#Identifiers

Challenge: Leaky Abstractions

“Abstractions fail. Sometimes a little, sometimes a lot. There's leakage. Things go wrong.”

- Joel Spolsky
http://www.joelonsoftware.com/articles/LeakyAbstractions.html

Tackling essentials

Life without a silver bullet

Buy vs Build

Eradicate software complexity by
not constructing any software

Buy vs Build

Reuse software wherever possible

Aim to make our own software more reusable where possible

Building the Right Thing

Essential complexity arises in part due
to unknowns and uncertainties

Building the Right Thing

Rapid prototyping needed to increase
frequency and quality of feedback

Great Designers

As always, people are the key to
(great) software engineering

Great Designers

Train, foster and develop great
designers from school age onwards

Organisations that rely on great software need to
value great designers more than great managers

Summary

Complexity is partly about size and
partly about understandability

Accidental complexity mostly can and should be
avoided: often via (non-leaky) abstractions

Essential complexity is mostly unavoidable,
though might be helped by sharing practice and
code, and by getting feedback faster