Why lean?

Designing and Maintaining Software (DAMS)

Louis Rose

Habitable Software

Leaner

Less **Complex**

Loosely **Coupled**

More **Cohesive**

Avoids **Duplication**

Clearer

More **Extensible**

???

Lean

“Perfection is finally achieved not when there is no longer anything to add, but when there is no longer anything to take away.”

- Antoine de Saint-Exupery

Lean software…

Has no extra parts

Solves the problem at hand and no more

Is often easier to change (i.e., is more habitable)

YAGNI

The argument in favour of lean software (part 1)

You Ain’t Gonna Need It!

*“Always implement things when you **actually** need them, never when you just **foresee** that you need them.”*

- Ron Jeffries

YAGNI Examples

Don’t implement a feature today that you won’t be shipping for a long time.

Don’t use a database when flat files will do.

Avoid “speculative generality”.

UNIX Philosophy

Favour small, focused and composable tools
over large, monolithic and integrated tools.

UNIX Example

Problem:
Count number of DAMS students named “Bob”

Solution:
Write a bespoke program

UNIX Philosophy

Problem:
Count number of DAMS students named “Bob”

Solution:
~~Write a bespoke program~~

UNIX Philosophy

Problem:
Count number of DAMS students named “Bob”

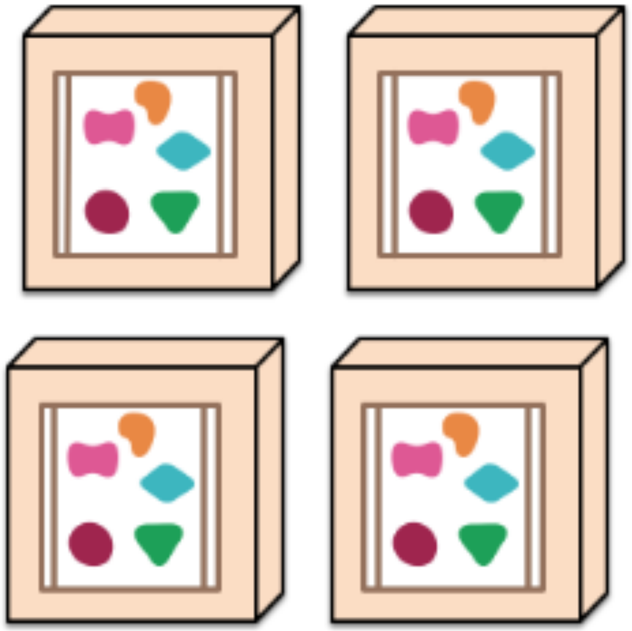
Solution:
Reuse existing programs:

```
cat dams_students.csv | grep Bob | wc -l
```

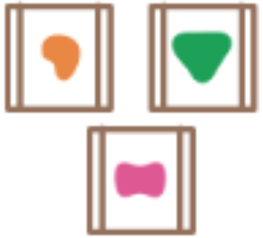
Microservices

A monolithic application puts all its functionality into a single process...

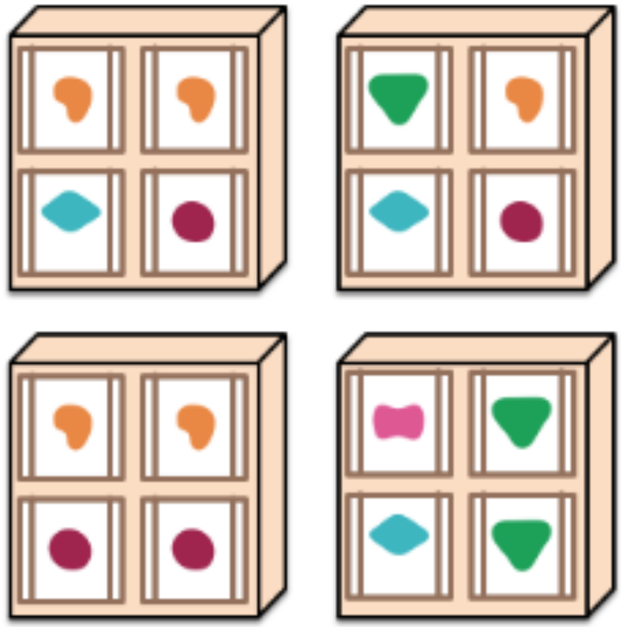
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



http://martinfowler.com/articles/microservices.html

Worse is Better

The argument in favour of lean software (part 2)

The Right Thing

Simplicity	The implementation and the interface must be simple.
Correctness	The design must be correct in all observable aspects. Incorrectness is not allowed.
Consistency	The design must not be inconsistent. A design can be less simple / complete to allow consistency.
Completeness	The design must cover as many use cases as is practical. Simplicity is not allowed to overly reduce completeness.

http://dreamsongs.com/WIB.html

Worse is Better

Simplicity	The implementation and the interface must be simple. **Simplicity is the most important design consideration.**
Correctness	The design must be correct in all observable aspects. **It is slightly better to be simple than correct.**
Consistency	The design must not be **overly** inconsistent. A design can **sometimes be inconsistent to allow simplicity**.
Completeness	The design must cover as many use cases as is practical. **Completeness can be sacrificed for any other quality.**

http://dreamsongs.com/WIB.html

“Worse is Better” is Better

Quality does not necessarily increase
with functionality

WIB software is more likely to “survive”
and be improved over time

“Unix and C are the the ultimate computer viruses”

KISS

Keep it simple, stupid!

When lean is mean

The argument against lean software

Overheads

If I split a class, I have to edit >1 file

If I split a program, I have to manage >1 project

If I split an application, I have to manage >1 server

Legacy Systems

I have a huge, bloated, monolith.
Where do I start?

Huge: refactor as you go.
Bloated: run coverage in production.
Monolith: start to identify boundaries in the domain.