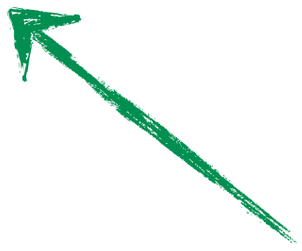
Get out of technical debt now!

Andy Lester



Perl guy
(Thank you!)

http://petdance.com/perl/technical-debt/
(with a couple of updates by Carsten Bormann)

Signs of debt

"Don't we have docs on the file layouts?"

"I thought we had a test for that..."

"If I change X, it's going to break Y. I think."

"Don't touch that code.
Last time we did, we
spent a week fixing it."

"We just lost the drive.
Where are our
backups?"

"Where's the email about that bug?"

"Just put in a comment with XXX. We'll see it later."

"Just put in a comment
with TODO.
We'll see it later."

"Just put in a TODO.
Right above the XXXs."

The costs of debt

"I don't know what happened! I just changed one line!"

"We can't upgrade. It will break code."

"We can't update the code. No time."

"We can't update the code. No one understands it."

The interest on
technical debt
will kill you.

So, either pay your technical debt, or else in time...

When you can't do the projects you must do, you are bankrupt.

The solution is simple

Technical debt

- Identify your debts
- Determine the costs
- Pay the most profitable
- Stop incurring new debt
- Repeat as necessary

Simple is not
the same as easy.

Identify your debts

Clutter

Clutter

```
$ ls /etc/apache
apache.conf
apache.conf.bak
apache.conf.save
apache.conf.mar-27
apache.conf.orig
apache.conf.andy
apache.conf.good
apache.conf.good2
apache.conf.test
apache.conf.temp
apache.conf.delete-me
```

Anything we plan to do

"later" is clutter.

Failing tests

Failing test = car alarm

Unfixed bugs

Fragile code

Ugly code

Code without comments

Outdated comments

```
=head1 wango( $ted )

Appends the $ted to the tango.

=cut

sub wango {
    my ($nuge,$motor) = @_;
    ...
}
```

Bus-sensitive knowledge

Unpatched security holes

TODO and XXX

Missing infrastructure?

- Ticketing system (e.g., Redmine)
- Version control system (e.g., git)
- Backups
- Automation
  - tests
  - deployment
  - other (cron...)

Relying on non-open-source products

Jerks on the team

Jerks in management

No coding standards

Code ownership

Not realizing you have technical debt

Ignoring your technical debt

Determine the costs

Think time.

Think dollars, too.

Think risk.

Determine the cost of paying the debt.

Think time & dollars.

You are creating the justification to sell the PHB on your changes.

"We don't have a backup server"

- What is cost of a new server? Hardware? Software? Manpower? Training?
- What if it's just risk?
- What are costs of paying the debt?
  - Dollars to buy
  - Install time
  - Opportunity cost

All cost assessments must include opportunity costs.

Opportunity cost

- What could be done if this change was not made?
- What if the person making the change is critical path on another project?
- Do you push out a project release date in order to document a database schema?

Pay the most profitable

Pay off the most profitable

- Pick **one**.
- Don't pick the easiest.
- Don't pick the most fun, either.
- Think improvement, not perfection.
- **JFDI**

How do I sell the boss?

- **QUANTIFY** these things
  - Improved turnaround
  - Improved quality
  - Decreased risk

Land one plane at a time.

Avoid future debt

Don't incur more debt.

Note when you do anyway.

Understand your cash flow.

Watch the corners

And automate that:
— CI system
— coverage tools
— heckle!

Sharpen the saw

Tools