LAST NIGHT I DRIFTED OFF
WHILE READING A LISP BOOK.

HUH?

SUDDENLY, I WAS BATHED
IN A SUFFUSION OF BLUE.

AT ONCE, JUST LIKE THEY SAID, I FELT A
GREAT ENLIGHTENMENT. I SAW THE NAKED
STRUCTURE OF LISP CODE UNFOLD BEFORE ME.

MY GOD

IT'S FULL OF ZEROS

THE PATTERNS AND META PATTERNS DANCED.
SYNTAX FADED, AND I SWAM IN THE PURITY OF
QUANTIFIED CONCEPTION. OF IDEAS MANIFEST.

TRULY, THIS WAS
THE LANGUAGE
FROM WHICH THE
GOIDS WROUGHT
THE UNIVERSE.

NO, IT'S NOT.

IT'S NOT?

I MEAN, OSTEASIBLY, YES.
HONESTLY, WE HACKED MOST
OF IT TOGETHER WITH PERL.
Perl
Baseless Myths & Startling Realities

by Tim Bunce, February 2008

Parrot and Perl 6 portion incomplete
due to lack of time
(not lack of myths!)
Realities

I'm positive about Perl
Not negative about other languages

Pick any language well suited to the task

Good developers are always *most important*,
whatever language is used
Who am I?

Tim Bunce
Author of the Perl DBI module
Using Perl since 1991
Involved in the development of Perl 5
“Pumpkin” for 5.4.x maintenance releases
http://blog.timbunce.org
~ Myths ~

~ Myths ~

Perl is dead

Perl is hard to read / test / maintain

Perl 6 is killing Perl 5

Another myth: Perl is slow:
http://www.tbray.org/ongoing/When/200x/2007/10/30/WF-Results
~ Myths ~

Perl is dead

Perl is hard to read / test / maintain

Perl 6 is killing Perl 5
Perl 5 isn’t the new kid on the block
Perl is 21 years old
Perl 5 is 14 years old

A mature language with a mature culture

How many times Microsoft has changed developer technologies in the last 14 years...
Perl 0: embryo
Perl 1: infant
Perl 2: toddler
Perl 3: child
Perl 4: preteen
Perl 5: adolescent

You can guess where that’s leading...

From “The State of the Onion 10” by Larry Wall, 2006
Buzz != Jobs

Perl5 hasn’t been generating buzz recently
It’s just getting on with the job
Lots of jobs
just not all in web development

Web developers tend to have a narrow focus.

“At a recent finance technology conference in New York, the top 3 foundational technologies (by number of references) mentioned over the course of the conference by its speakers were:
#3 - XML
#2 - SQL
#1 - Perl
By "foundational technology" I mean a building block technology.
Guess the Languages

![Job Trends](image)

www.indeed.com

“web developer”

Yes, Perl is growing more slowly than others but these are just “web developer” jobs

I think this graph captures the essence of why people think Perl is stagnant. It's because Perl hasn't been growing much "in the 'web developer' world"

PHP - I'm not going to focus on PHP because it's not a general purpose language.
Perl is mentioned in many more software engineer/developer jobs.
Perl is the primary focus of more developer jobs. Want a fun new job? Become a Perl developer!

The existence of “foo developer” job titles is a sign of maturity.
Massive Module Market

Large and vibrant developer community
Over 14,000 distributions (54,000 modules)
Over 6,400 ‘authors’ (who make releases)
Thousands of new releases every month to hundreds of modules

Libraries are more important than languages.
Large user community leads to large contributor community (given good community tools - more on that later)

http://search.cpan.org/recent
Top Modules

Many gems, including...

DBI  DBD::*  DBIx::Class  Rose::DB::Object
Catalyst  Moose  DateTime
Algorithm::*  Statistics::*  Thread::*
XML::*  HTML::*  WWW::*  Parse::*
Net::*  Email::*  POE::*  Locale::*
Test::*  Devel::Cover  Perl::Critic  perltidy

Quality varies on CPAN, naturally.
(See Acme::* for some fun.)

http://www.serpentine.com/blog/2008/02/19/peruse-popular-perl-packages/

“Moose is pretty much the most exciting thing I’ve seen come out of the Perl 5 world in quite some time, and I’m really enjoying using it for my projects”

(My apologies if your favourite module isn’t included here.)
Comprehensive Perl Archive Network

360 mirrors in 51 regions (TLDs)

CPAN handles the global distribution.

Mirror status http://www.cs.uu.nl/stats/mirmon/cpan.html
Developer Services

Upload a perl module distribution and you automatically get...

- global distribution and archiving
- namespace ownership and management
- a bug tracking queue at rt.cpan.org
- a forum at cpanforum.com
- smoke testing on many platforms

This is a mature environment with rich services.

I'll discuss smoke testing and quality a little later
For each and every distribution:
Browse well formatted inter-linked docs
Links to forums, bug tracking, ratings, annotated documentation, dependency analysis, smoke test results, and more...

620,000 unique visitors per month
150,000 page views per day
33,000 visits per day

The primary face of CPAN
TOOLS Also has other tools, like grep'ing and diff'ing the distributions without downloading them.
DOWNLOAD You can download distributions but most people use automated tools for that.
Source for stats: google analytics (added 31st Jan, so unique visitor count is probably too low and its too soon to see trends)
Example page for a distribution
Showing rich set of features
<table>
<thead>
<tr>
<th>Module</th>
<th>Successfully Tested</th>
<th>Dependencies need not be hell!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moose</td>
<td>16 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td>Shows tree of dependencies</td>
</tr>
<tr>
<td>Test::LongString</td>
<td>108 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td>Identify risks.</td>
</tr>
<tr>
<td>Test::Builder</td>
<td>Core module</td>
<td>Refine the view to match your particular operating system and Perl version</td>
</tr>
<tr>
<td>Scalar::Util</td>
<td>Core module</td>
<td></td>
</tr>
<tr>
<td>Test::Exception</td>
<td>28 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td></td>
</tr>
<tr>
<td>Test::Harness</td>
<td>Core module</td>
<td></td>
</tr>
<tr>
<td>Sub::Uplevel</td>
<td>34 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td></td>
</tr>
<tr>
<td>Exporter</td>
<td>Core module</td>
<td></td>
</tr>
<tr>
<td>Carp</td>
<td>Core module</td>
<td></td>
</tr>
<tr>
<td>Filter::Simple</td>
<td>Core module</td>
<td></td>
</tr>
<tr>
<td>Class::MOP</td>
<td>20 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td></td>
</tr>
<tr>
<td>File::Spec</td>
<td>Core module</td>
<td></td>
</tr>
<tr>
<td>Sub::Name</td>
<td>16 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td></td>
</tr>
<tr>
<td>Sub::Exporter</td>
<td>81 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td></td>
</tr>
<tr>
<td>Params::Util</td>
<td>99 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td></td>
</tr>
<tr>
<td>Sub::Install</td>
<td>102 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td></td>
</tr>
<tr>
<td>Data::OptList</td>
<td>89 PASSes, 0 UNKNOWNS, 0 FAILs, 0 NAs</td>
<td></td>
</tr>
<tr>
<td><strong>Chance of success</strong></td>
<td><strong>100%</strong></td>
<td></td>
</tr>
</tbody>
</table>
Did I say it was a vibrant and growing developer community?

(2008 may be slightly exaggerated as it’s extrapolated from January data.)
The trend is clear.

Certainly doesn’t look like a dead language!

http://services.alphaworks.ibm.com/manyeyes/view/SmAgULsOtha6C7W~7CInL2~
Perl is dead

Perl is hard to read / test / maintain

Perl 6 is killing Perl 5
~ Myths ~

Perl is dead

BUSTED!

Perl is hard to read / test / maintain

Perl 6 is killing Perl 5
“How Not to Write FORTRAN in Any Language”

by Donn Seeley
ACM Queue Vol. 2 No. 9 - Dec/Jan 2004-2005

False: “There’s no obfuscated Perl contest because it’s pointless.”
—Jeff Polk

True: “Whatever language you write in, your task as a programmer is to do the best you can with the tools at hand.”
—Kernighan and Pike

Perl has a bad rap in this area.

Spaghetti code is fact of life in all languages. Perl, Python, Ruby, PHP, Java etc.
The skills, and especially the attitude of developers, makes the biggest difference.

Culture is part of the issue.
Perl's use in the dot-com era grew faster than the culture matured.
That's changed.

“True greatness is measured by how much freedom you give to others, not by how much you can coerce others to do what you want.”
—Larry Wall
Guidelines and Tools

Perl Best Practices
Perl::Tidy
Perl::Critic
Test::*
Devel::Cover
Already a growing culture of quality and testing when the book came out.
July 2005
256 guidelines

Agreeing on one set of sane guidelines is more important than the exact details of the guidelines.

PBP makes it easy for a team to agree “We’ll just follow PBP guidelines”.
Perl::Tidy

Perl code beautifier
Works beautifully - can be trusted
Supports *many* options for personal styles
Perl Best Practices recommended options

Normalise the coding style of existing code.
Very simple and effective way to add clarity to a code base.

Now you’ve got pretty code,
    but is it good code?
CRITIC (next)
Perl::Critic

Static Code Analysis for Perl
Includes over 120 policies
Most based on Perl Best Practices
Grouped into levels and themes
Configurable and extensible
for local policies and styles

An extensible framework for creating and applying coding standards to Perl source code
Perltidy address the layout of code.
Perlcritic addresses the semantics.

Now you've got pretty code,
that follows best practices,
but does it work?
... TESTING (next)
Perl culture takes testing seriously
Excellent mature tools for testing
Test Anything Protocol - producers/consumers
Test::* modules make it easy to write tests
Over 200 Test::* distributions on CPAN

Also many 'mock' modules for mocking objects and other functionality to ease testing
Test::Class provides xUnit styles tests
The test modules work together.
http://search.cpan.org/search?m=dist&q=Test%3A%3A&s=1&n=100

Now...
you've got pretty code, that follows best practices, has tests and the tests pass, but how much code is exercised by the tests?
COVERAGE (next)
Devel::Cover

Coverage Analysis for Perl
Tells you what code has been executed
Statement, branch, condition, subroutine, pod
Produces drill-down reports in HTML

Includes documentation coverage analysis.
Devel::Cover Reports

http://pjcj.sytes.net/cover/latest/

<table>
<thead>
<tr>
<th>File</th>
<th>stmt</th>
<th>bran</th>
<th>cond</th>
<th>sub</th>
<th>pod</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcePerl-1.91</td>
<td>72.48</td>
<td>52.74</td>
<td>43.18</td>
<td>74.66</td>
<td>56.36</td>
<td>64.15</td>
</tr>
<tr>
<td>Algorithm-Annotate-0.10</td>
<td>100.00</td>
<td>50.00</td>
<td>n/a</td>
<td>100.0</td>
<td>0.00</td>
<td>87.80</td>
</tr>
<tr>
<td>Algorithm-C3-0.06</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Algorithm-Dependency-1.102</td>
<td>75.13</td>
<td>41.60</td>
<td>31.43</td>
<td>85.00</td>
<td>92.11</td>
<td>67.14</td>
</tr>
<tr>
<td>Algorithm-Diff-1.1902</td>
<td>87.47</td>
<td>78.68</td>
<td>77.66</td>
<td>80.77</td>
<td>100.00</td>
<td>83.85</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>file</th>
<th>stmt</th>
<th>bran</th>
<th>cond</th>
<th>sub</th>
<th>pod</th>
<th>time</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>blib/lib/Algorithm/Dependency.pm</td>
<td>95.2</td>
<td>83.3</td>
<td>n/a</td>
<td>93.3</td>
<td>100.0</td>
<td>34.4</td>
<td>92.7</td>
</tr>
<tr>
<td>blib/lib/Algorithm/Dependency/Item.pm</td>
<td>100.0</td>
<td>50.0</td>
<td>33.3</td>
<td>100.0</td>
<td>100.0</td>
<td>8.2</td>
<td>86.8</td>
</tr>
<tr>
<td>blib/lib/Algorithm/Dependency/Ordered.pm</td>
<td>100.0</td>
<td>75.0</td>
<td>n/a</td>
<td>100.0</td>
<td>100.0</td>
<td>9.3</td>
<td>93.7</td>
</tr>
<tr>
<td>blib/lib/Algorithm/Dependency/Source.pm</td>
<td>86.3</td>
<td>55.0</td>
<td>46.7</td>
<td>90.9</td>
<td>100.0</td>
<td>22.2</td>
<td>75.5</td>
</tr>
<tr>
<td>blib/lib/Algorithm/Dependency/Source/File.pm</td>
<td>100.0</td>
<td>50.0</td>
<td>n/a</td>
<td>100.0</td>
<td>100.0</td>
<td>7.8</td>
<td>85.2</td>
</tr>
<tr>
<td>blib/lib/Algorithm/Dependency/Source/HoA.pm</td>
<td>100.0</td>
<td>50.0</td>
<td>n/a</td>
<td>100.0</td>
<td>100.0</td>
<td>0.5</td>
<td>90.5</td>
</tr>
<tr>
<td>blib/lib/Algorithm/Dependency/Weight.pm</td>
<td>97.7</td>
<td>44.4</td>
<td>n/a</td>
<td>100.0</td>
<td>100.0</td>
<td>13.6</td>
<td>85.9</td>
</tr>
<tr>
<td>inc/Class/Inspector.pm</td>
<td>41.1</td>
<td>19.6</td>
<td>0.0</td>
<td>43.5</td>
<td>83.3</td>
<td>2.1</td>
<td>35.6</td>
</tr>
<tr>
<td>inc/Test/ClassAPI.pm</td>
<td>77.2</td>
<td>40.5</td>
<td>28.6</td>
<td>100.0</td>
<td>50.0</td>
<td>1.9</td>
<td>68.4</td>
</tr>
<tr>
<td>Total</td>
<td>75.1</td>
<td>41.6</td>
<td>31.4</td>
<td>85.0</td>
<td>92.1</td>
<td>100.0</td>
<td>67.1</td>
</tr>
</tbody>
</table>
### Table: Criteria Coverage

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Covered</th>
<th>Total</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>statement</td>
<td>42</td>
<td>43</td>
<td>97.7</td>
</tr>
<tr>
<td>branch</td>
<td>8</td>
<td>18</td>
<td>44.4</td>
</tr>
<tr>
<td>condition</td>
<td>n/a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subroutine</td>
<td>12</td>
<td>12</td>
<td>100.0</td>
</tr>
<tr>
<td>pod</td>
<td>5</td>
<td>5</td>
<td>100.0</td>
</tr>
<tr>
<td>total</td>
<td>67</td>
<td>78</td>
<td>85.9</td>
</tr>
</tbody>
</table>

### Code Snippet

```perl
package Algorithm::Dependency::Weight;

sub weight_all {
    my $self = shift;
    my @items = $self->source->items;
    defined $items[0] or return undef;
    $self->weight_hash( map { @$_->id } @items );
}
```
and 3 conditions

<table>
<thead>
<tr>
<th>line</th>
<th><code>!</code></th>
<th><code>&amp;&amp;r</code></th>
<th><code>&amp;&amp;l</code></th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>139</td>
<td>0</td>
<td>0</td>
<td>1037</td>
<td>defined $_[0] &amp;&amp; !<em>ref($</em>[0])</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1037</td>
<td>defined $<em>[0] &amp;&amp; !<em>ref($</em>[0]) &amp;&amp; $</em>[0] ne ''</td>
</tr>
</tbody>
</table>

or 3 conditions

<table>
<thead>
<tr>
<th>line</th>
<th><code>l</code></th>
<th><code>!!&amp;r</code></th>
<th><code>!!!i</code></th>
<th>condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>1037</td>
<td>0</td>
<td>0</td>
<td>$$self{'loaded'} or $self-&gt;load</td>
</tr>
<tr>
<td>161</td>
<td>11</td>
<td>3</td>
<td>0</td>
<td>$$self{'loaded'} or $self-&gt;load</td>
</tr>
<tr>
<td>182</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>$$self{'loaded'} or $self-&gt;load</td>
</tr>
</tbody>
</table>
~ Myths ~

Perl is hard to read / test / maintain

Perl 6 is killing Perl 5
~ Myths ~

Perl is dead

BUSTED!

Perl is hard to read / test / maintain

BUSTED!

Perl 6 is killing Perl 5
Perl 6 saved Perl 5!

“Perl 5 had already started dying, because people were starting to see it as a dead-end language.

It seemed odd at the time, but when we announced Perl 6, Perl 5 suddenly took on a new life.”

—Larry Wall, 2002
Perl 6 saved Perl 5!

In 2000 perl was dying from the inside out
The Perl 6 RFC process “vented speen”
Perl 5 development has gone smoothly since
Much refactoring driven by Perl5-on-Parrot
Many new features inspired by Perl 6 work

Back around 2000...
Perl5 was getting very hard to maintain
Bickering on the mailing lists
Few volunteers for any real work
Lack of direction

Using Perl 5 as one of the backends for Perl 6.
Perl 5.10

Perl 5.10 was released in December
Five years after 5.8.0, two years after 5.8.8

Refactored internals

*many fixes, more speed, less memory*

Switch statement, smart matching, named captures, state variables, defined-or, say, field hashes, pluggable regex engines, trie-based non-recursive pattern matching, and more...

A Culture of Testing

Another bonus from Perl 6:

*Strong test suites for Perl 5 code are needed to ensure backwards compatibility*

Strengthened culture of testing
2002: Perl 5.8.0 had 26,725 core tests
+41,666 more for bundled libraries etc.

2007: Perl 5.10.0 has 78,883 core tests
+109,427 more for bundled libraries etc.

(I couldn’t find code coverage stats.)

For perspective: Ruby has ~1,400 core tests plus ~14,000 for bundled libraries etc.

See comments in http://www.oreillynet.com/onlamp/blog/2007/05/trust_but_verify.html
and http://reddit.com/info/1uzda/comments/
Module Test Suites

CPAN Testers Network:

Automated smoke testing of CPAN uploads
Runs the test suite included in upload
61 different platforms
20 different versions
Immediate feedback for developers

Some by completely automated robots in virtual machines.

Some by users who submit reports as they download and test via the installer.
This one is for the DBI
~90,000 Reports/Month

http://services.alphaworks.ibm.com/manyeyes/view/SmAgULsOtha6g7GcD5KnL2-

Is perl6 killing perl5? Certainly doesn't look like it!
Myths

Perl is dead

Perl is hard to read/test/maintain

Perl 6 is killing Perl 5

BUSTED!
~ Myths ~

Perl is dead

Perl is hard to read / test / maintain

Perl 6 is killing Perl 5

BUSTED!
Community Resources

“Awesome community. 
Perl people tend to be laid-back and friendly.”

perlmonks.org - Meditations and wisdom
use.perl.org - News and blogs
perlbuzz.com - Headlines and articles
comp.lang.perl.* - Newsgroups

Quote taken from comment on http://blog.tim bunce.org/2008/02/14/perceptions-of-perl-views-from-the-edge/
Any questions?

http://perlmonks.org
http://search.cpan.org
http://blog.tim Bunce.org