Perl's Diaspora

Should we fear the future?

Elizabeth Mattijsen
Brussels, 2 February 2013
Perl is DEAD!

• Haven't you heard?
• Or at least it's *abdicating*.
• Or its found its *niche*.
• It provides *job security*.
• But *everybody* is using it.
Not happy with Perl 5

• It does not have a number of features of newer languages.

• It is very hard to add these new features because of back compatibility issues.

• It is even harder to add new features because of the innards of Perl 5.

• Macro-infused C-like language: Jenga!
Example

• ithreads.

• They are not threads as most people know them.

• They are an emulation of fork() for Windows backported to Unixes.

• Why?

• The architecture of Perl.
Alternatives?

• Perl 6 is an alternative.
• Better runtime for Perl 5 is an alternative.
• Other languages are an alternative.
• A new Perl 5 might be an alternative.
New Perl 5 initiatives

• Will they not take away attention from the real Perl?
• Will they not fragment the developer base?
• Will it not be a bad thing all around?
• Perhaps, but it will be -Ofun.
• And it has happened many times before.
Some History first

• First, Larry made Perl 1.
• Then Perl 2.
• And then Perl 3.
• With Perl 4 things started to get hairy.
1991 - Versions of Perl 4

- No extension mechanism.
- Extensions hardcoded in the core.
- `oraperl`, `sybperl` were most used.
- Hard to maintain with core changes.
- Fixed in Perl 5!
Perl 5 in 1994

- Design started in 1993.
- Modules, objects, extensions.
- Easy language for scripting CGI.
- Perl becomes mainstream.
- Core development relatively easy.
- But Jenga develops quickly.
1998 - Topaz

- "Perl is hard to maintain"
- Written in C++ rather than C.
- Perl for the 22nd century!
- Abandoned in 2000.
- But became one of the inspirations of Perl 6.
2000 - Perl 6

• A Community rewrite of Perl.
• RFC input from all over the world.
• Still being digested by Larry in some parts.
• Result: a design document for Perl 6.
• But how to implement?
2001 - Parrot

- The Runtime (more modernly VM)
- Perl 6 and maybe other scripting languages.
- Initially an April Fool's joke.
- It got out of hand. Seriously.
- It is now an Edsel.
2005 - Pugs

• By Audrey Tang.

• Prototype Perl 6 implementation in Haskell.

• Provided many pointers for Rakudo.

• Not many core developers versed enough in Haskell to be able to contribute.

• Stalled in 2006.
2006 - Perlito

- Research project of Flavio Glock.
- Compile (subset of) Perl 5 / 6 code.
- Execute in Javascript, Python, Ruby, Common Lisp, Go.
- Execute Perl 5 / 6 inside browser.
- Considered complete in 2013.
2006 - Moose

• New object system for Perl 5.
• By Stevan Little et al.
• Inspired by Perl 6 and many others.
• Bolted on Perl 5, requires many modules.
• Lighter versions: Moo, Mo.
• Widely in production.
2009 - Rakudo

- Split from the Parrot project by Patrick Michaud & Jonathan Worthington.
- Further development of Perl 6.
- 6model abstracted object system.
- Distancing from Parrot.
- Other VM's should be possible.
2010 - Niecza

- Perl 6 implementation by Stephen O'Rear.
- From scratch.
- Using .NET / mono as VM.
- Potentially more core developers.
- But stuck with a single VM.
• **Not Quite Perl** by Patrick Michaud & Jonathan Worthington.

• Subset of Perl 6.

• The "miniperl" of Perl 6.

• Bootstrap the "real" Perl 6.

• VM agnostic (not quite yet).
2011 - p5-mop

- Integrate Moose features into Perl 5 core.
- Stalled in 2013 because of difficulty in implementation in Perl 5.
- Jenga strikes again.
2012 - STD5

- Inspired by Perl Reunification Summit.
- Parse Perl 5 code inside Rakudo.
- Will not include indirect object syntax.
- Stalled for lack of tuits.
2012 - nqp-jvm

- By Jonathan Worthington.
- Writes Java Bytecode for Rakudo.
- Allows Perl 6 to run on JVM.
- Moving forward very fast now.
2013 - Moe

- By Stevan Little et al.
- "Pugs for Perl 5".
- p5-mop frustrations coming out.
- May turn out to be just a thought experiment or a research project.
2013 - p2

• p2 by Reini Urban.
• Perl 5+i like syntax.
• Using potion as a backend.
• Directly writes machine code, so fast!
• Potential Rakudo backend.
• Community development uncertain.
Now

• Classic Perl 5 (p5p).
• Rakudo Perl 6 (on Parrot & JVM).
• Niecza Perl 6 (on .NET / mono).
• Moe (Pugs for Perl 5).
• p2 (Perl 5+i on potion).
Fear the Future?

• **No**, but we should be vigilant.
• We should **do** more rather than talk.
Classic Perl 5 (p5p)

- Suffering from major Jenga.
- Codebase was bad in 1998 (Topaz).
- In some ways better, in some ways worse.
- Stopped p5-mop effort.
- Still on yearly release schedule.
Rakudo

• Moving away from Parrot.
• Seems like running on JVM before summer.
• Has a healthy developers community.
• Has monthly Rakudo * updates.
Nieczza

• In some ways more complete Perl 6.
• And faster than Rakudo on Parrot.
• Healthy developer community.
• But stuck to single VM.
Moe / p2

- Very early in their lifecycle.
- Who knows what they will bring?
Final word of warning

• CPU's are not getting faster.
• But we will get more CPU's per box.
• Writing threaded programs is hard.
• Perl will need auto-threading capabilities.
• Should be a USP of modern Perl.
Perl 6 can auto-thread

- Perl 6 specification defines auto-threading.
- System uses multiple threads when it can.
- No code changes, maybe some hints.
- Check out "junction", "hyper" and "race".
Future of Perl 6?

• Check out #perl6 on freenode.
• Friendly people working hard on Perl 6.
Future of Perl 5?

- Follow closely where Moe is going.
Questions?

Perl's Diaspora

Should we fear the future?

Elizabeth Mattijsen
Brussels, 2 February 2013