Porting a Web Application from Perl 5 / Mojolicious to Perl 6 / Cro

Content

- Background
- Implementation
- Experiences with porting Perl 6
- Getting help

This is neither a Perl 5 nor a Perl 6 talk …

… or may be both.

NH_3 Emissions from Agriculture

- Yearly loss of **40,000 tonnes** of nitrogen (N), almost 1/3 of farmyard manure **in Switzerland**
- Diminished productivity → financial loss for farmers

- Ammonia immissions damage natural ecosystems
- $N/NH_3/NO_x$ are air pollutants

Agrammon Simulation Model

- NH_3 production from livestock (cows, pigs, poultry, …)
- Emissions from
  - housing, yard, pasture/grazing
  - storage (manure, slurry)
  - manure/slurry application to the field
  - additional fertilizers for plant production

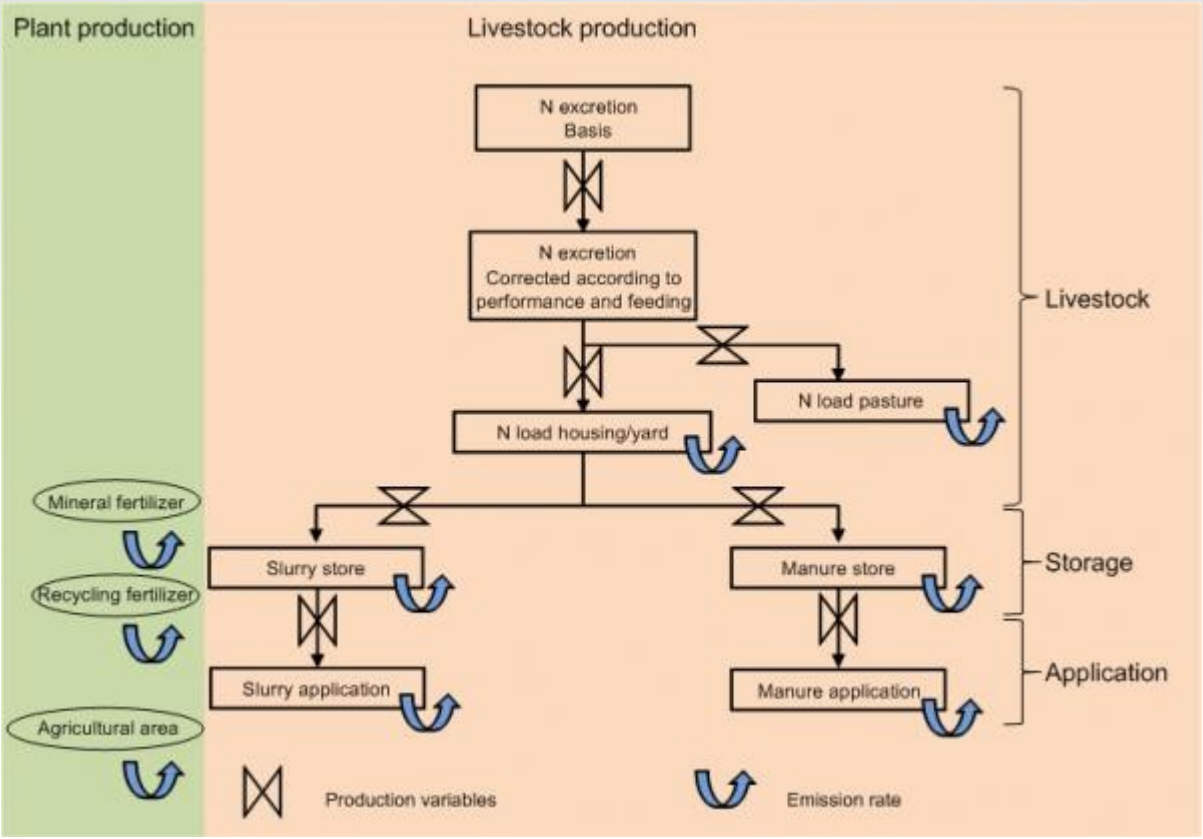
Agrammon Simulation Model

- Emissions depend on structural parameters:
  - number and types of animals
  - Types of barns and storage areas

- Production methods:
  - feeding
  - application methods

Can you Spot the Difference?

Agrammon Simulation Modell



Agrammon Simulation Model

- Created by agricultural scientists
- … implemented in Excel
- … became unmaintainable and undebuggable
- … especially after the implementor left

- ”Deployment” / Distribution …

Agrammon Application

- Single-page web application
  - individual farms
  - regional simulations from statistical data

- Command line application
  - national projections
  - thousands of individual farms
  - Post-processing of results

Vision

- Simulation model is developed and extended by scientists / students …

  … with limited/no programming skills

- Self-documenting … easily reviewable

Design

- Simulation model and GUI generated from
  - modular model descriptions
  - structured plain text files
  - generated/uploaded by scientists
- Documentation
  - generated
  - additional background informationen
  - easily reviewable

Agrammon Modules

- Description (documentation)
- Input parameters (users)
    - e.g. animal and farm infos
- Technical parameters (model)
    - e.g. emission rates
- Used sub-modules
- Formulas (simulation) → Output

Livestock::DairyCow::Excretion

*** general ***

+short description

Computes the annual N excretion of a number of dairy cows as a function of the milk yield and the feed ration.

+description

This process calculates the annual N excretion ….

\subsubsection{References}

Burgos SA, Robinson PH, Fadel JG, DePeters EJ 2005.
Ammonia volatilization potential: Prediction of urinary urea nitrogen output on lactating dairy cows.
Agriculture, Ecosystems and Environment 111:261-269.

Livestock::DairyCow::Excretion

*** input parameters ***

```
+dairy_cows
  type      = integer
  validator = ge(0)
  ++labels
      en = Number of animals
      de = Anzahl Tiere
      fr   = Nombre d'animaux
  ++units
     en = -
  ++description
      Number of dairy cows in barn.
  ++help
     +++en
        <p>Actual number of animals
             in the barn.</p>
      +++de  ...
      +++fr    ...
```

*** technical parameters ***

```
+standard_N_excretion
   value = 115
  ++units
      en = kg N/year
      de = kg N/Jahr
      fr   = kg N/an
  ++description
    Annual standard N excretion for a
    dairy cow according to
    Flisch et al. (2009).
```

Livestock::DairyCow::Excretion

*** external ***

+**Excretion::CMilk**

+**Excretion::CFeed**

*** output ***

+**n_excretion**

print = 7

++units

en = kg N/year
de = kg N/Jahr
fr = kg N/an

++formula

Tech(**standard_N_excretion**)
* Val(cmilk_yield, **Excretion::CMilk**)
* Val(c_feed_ration,**Excretion::CFeed**)
* In(**dairy_cows**);

LaTeX / PDF documentation

++description

Annual total N excreted by a specified number of animals.

Module parsing

- Config::Grammar
  - hierarchical structure (sections, subsections)
  - key / value pairs
  - free text
  - input validation

  → HASH

Calculations

- Perl5 formulas with ”simplified” hash accessors
  - `In(variableName)`

    → `$inputs->{variableName}`
  - `Tech(parameterName)`

    → `$tech->{parameterName}`
  - `Val(outputName, moduleName)`

    → `$outputs->{moduleName}{outputName}`

- `eval()` ed

GUI generation

Forms and and navigation from module files:

Livestock.nhd:

```
name      = Livestock
gui       = Livestock, Tierhaltung, Production animale
```

Livestock/DairyCow.nhd:

```
name      = Livestock::DairyCow
gui       = DairyCow, Milchkühe, Vâches laitières
instances = multi
```

Livestock/DairyCow/Excretion.nhd:

```
name      = Livestock::DairyCow::Excretion
```

GUI generation

Form fields from input parameters:

```
+dairy_cows
  type = integer
  validator = ge(0)
  branch = true
  ++labels
    en = Number of animals
    de = Anzahl Tiere
    fr = Nombre d'animaux
  ++units
    en = -
  ++description
    Number of dairy cows in barn.
  ++help
    +++en
       <p>Actual number of animals in the barn.</p>
    +++de
       <p>Tatsächliche Anzahl Tiere im Stall.</p>
    +++fr
       <p>Nombre effectif d'animaux dans la stabulation.</p>
```

Reality

- 128 module descriptions with
  - ≈ 30,000 lines (including comments)
- maintained
  - by a colleague with some programming skills for the last 8 years
  - by me since this year
- Documentation infrequently looked at

Results

- On screen:
  - Summary with live-ipdate
  - Detailed tables
  - Graphs (histograms, pie)
- Export:
  - PDF and Excel
  - Transmittal of PDF report by eMail
  - Sending of datasets to other users

Agrammon Perl 6

- Several pending change requests
- ”Grown” 10 year old code base, little testing

- Performance not critical
- Parsing / compiling important

- Why Perl 6?

Because we can!

Agrammon 6

META6.json:

```
{
  "name"        : "Agrammon",
  "description" : "Simulation model for calculating \
                   ammonia emissions from agriculture.",
  "tags"        : ["Agriculture"],
  "perl"        : "6.*",
  "version"     : "0.1",
  "authors"     : [
      "Fritz Zaucker <fritz.zaucker@oetiker.ch>",
      "Jonathan Worthington <jonathan@edument.se>"
  ],
  "license"     : "Artistic-2.0",
  "source-url"  : "https://github.com/oposs/agrammon.git",
  "resources"   : [],
```

Agrammon 6

```
    "depends" : [
        "Cro::HTTP",
        "Cro::OpenAPI::RoutesFromDefinition",
        "DB::Pg",
        "Digest::SHA1::Native",
        "OO::Monitors",
        "Text::CSV",
        "YAMLish",
        "Test::NoTabs"
    ],
    "provides": {
       /*
          49 class files
       */
    }
}
```

agrammon.pl6

- dev/agrammon6/bin/agrammon.pl:

```
#!/usr/bin/env perl6

use Agrammon::UI::CommandLine;
```

- That's all there is!

Do you know why?

agrammon.pl6

- cd dev/agrammon6
  perl6 -Ilib bin/agrammon.pl6

```
Usage:
  bin/agrammon.pl6 web <cfg-filename> <model-filename>
                       [<tech-file>]
                       -- Starts the web interface
  bin/agrammon.pl6 [--language=<SupportedLanguage>]
                   [--prints=<Str>] [--csv]
                   [--batch=<Int>]  [--degree=<Int>] run
                   <filename> <input> [<tech-file>]
                   -- Run the model
  bin/agrammon.pl6 latex <filename> -- Create LaTeX docu
  bin/agrammon.pl6 create-user <username>
                               <firstname> <lastname>
                   -- Create an Agrammon user

  See https://www.agrammon.ch for more information.
```

Agrammon::UI::CommandLine

```
#| Start the web interface
multi sub MAIN('web',
                ExistingFile $cfg-filename,
                ExistingFile $model-filename,
                Str $tech-file?) is export {
    my $http = web $cfg-filename, $model-filename,
                   $tech-file;
    react {
        whenever signal(SIGINT) {
            say "Shutting down ...";
            $http.stop;
            done;
        }
    }
}
```

Agrammon::UI::CommandLine

```
sub web(Str $cfg-filename, Str $model-filename,
        Str $tech-file?) is export {
    ...
    my $ws = Agrammon::Web::Service.new(
        :$cfg, :$model, :%technical-parameters);
    my $host = %*ENV<AGRAMMON_HOST> || '0.0.0.0';
    my $port = %*ENV<AGRAMMON_PORT> || 20000;
    my Cro::Service $http = Cro::HTTP::Server.new(
        :$host, :$port,
        application => routes($ws),
        after => [
            Cro::HTTP::Log::File.new(logs => $*OUT, errors => $*ERR)
        ],
        Before => [
          Cro::HTTP::Session::InMemory[Agrammon::Web::SessionUser].new(
               expiration  => Duration.new(60 * 15),
            )
        ]
    );
    $http.start;
    say "Listening at http://$host:$port";
    return $http;
}
```

Agrammon::UI::CommandLine

```
sub run ( ... $batch, $degree)  is export {
    ...

    my $model = load-model-using-cache(...);

    my $rc = Agrammon::ResultCollector.new;
    my atomicint $n = 0;

    race for $ds.read($fh).race(:$batch, :$degree)
        -> $dataset {
        ++$n;
        my $outputs = $model.run(...);
        $result = output-as-csv( ... $outputs ...);
        $rc.add-result( ... $result ...);
    }
    return $rc.results;
}
```

Agrammon Backends

- Perl 6
  - 49 .pm6 files with 4811 lines (some GUI interface code missing)
  - 42 .t files with 3488 lines
  - 6 CPAN modules
- Perl 5
  - 21 .pm files with 7766 lines
  - Some result tests
  - 30 CPAN modules

Agrammon6 Parser/Compiler

```
find lib/Agrammon/ -name \*Builder*
     -o -name \*Parser\* | xargs wc | magic_sort

   68   184  1556 lib/Agrammon/CommonParser.pm6

  257   706  5896 lib/Agrammon/Formula/Parser.pm6
  407   822 10123 lib/Agrammon/Formula/Builder.pm6

   73   111  1446 lib/Agrammon/ModuleParser.pm6
   92   196  2489 lib/Agrammon/ModuleBuilder.pm6

   27    57   637 lib/Agrammon/TechnicalParser.pm6
   17    40   440 lib/Agrammon/TechnicalBuilder.pm6

  941  2116 22587 total
```

Agrammon6 Performance

8 x Intel(R) Xeon(R) CPU E5-2637 v4 @ 3.50GHz

0.5 sec per data set

- 55% calculation
- 45% output generation

Good enough for interactive GUI usage (live update)

Performance (CH simulation)

- CLI, 2688 datasets (farms) with 425 input variables
- Wall-clock time [secs]

CPU core(s)	Perl 5.22	Rakudo 2018.06	Rakudo 2018.08	Rakudo 2018.0x eaca68d5a5f62
1	402	**2035 (507%)**	**1167 (290%)**	**994 (247%)**
2	n/a	1017 (253%)	677 (169%)	601 (150%)
4	n/a	550 (137%)	410 (102%)	399 (99%)
8	n/a	358 (89%)	303 (75%)	298 (74%)
16	n/a			341 (85%)

Performance (CH simulation)

- CLI, 2688 datasets (farms) with 425 input variables
- Wall-clock time [secs]

CPU core(s)	Perl 5.22	Rakudo 2018.06	Rakudo 2018.08	Rakudo 2018.0x eaca68d5a5f62	***Rakudo 20??.??***
1	402	**2035 (507%)**	**1167 (290%)**	**994 (247%)**	**402 (100%)**
2	n/a	1017 (253%)	677 (169%)	601 (150%)	
4	n/a	550 (137%)	410 (102%)	399 (99%)	
8	n/a	358 (89%)	303 (75%)	298 (74%)	
16	n/a			341 (85%)	

Current state

- Parses current model descriptions
- Identical simulation results as Perl5
- Simulation ”as fast” as Perl 5 (4 versus 1 core)
- Basic backend/GUI interaction working

TODO:

- 1 parser extension (detailed report just added)
- add missing GUI functionality

Perl 6 Experiences

- Rich new syntax
  - Similarities with Perl5, e.g.
    - `.method()` instead of `->method()`
  - Completely different things
  - New things, e.g.
    - `:$var` short for `var => $var`
    - many operators
    - many builtins
    - ...

Perl 6 Experiences

- Many ”new” features and concepts, e.g.
  - signatures
  - native OO
  - async and concurrency, lazy evaluation
  - type system
  - references, bindings, assignments, containers
  - multi-paradigm programming

Perl 6 Experiences

- Performance (tricks)
  - Slow startup → pre-compilation
    - annoying during development !
  - Some things are slower than others
  - Regexes are slow, as is Text::CSV
  - Parallelization is ”easy”

Perl 6 Experiences

- Great error messages
  - e.g. Perl5isms
- Profiler didn’t work for me, but looks promising
- Getting the latest Rakudo release is simple:
  - wget tar-ball, perl Configure, make
  - git clone MoarVM, nqp, rakudo
  - git clone zef and install

Perl 6 Experiences

There is no excuse to use an ”old” Rakudo version!

Getting help with Perl 6

- Online documentation, plenty of books
- Stack Overflow
- #perl6 on irc.freenode.net
- Perl 6 Weekly --- a must
- Blogs
  - Liz → for mere humans
  - Jonathan → for the brave and bold
  - …

Getting help with Perl 6

- Learn from a pro:
  - discuss your architecture
  - get code review (e.g. through GitHub)
  - have things implemented that currently are too hard to get right yourself

  → Don’t write Perl 5 in Perl 6

Getting help with Perl 6

- Agrammon 6 would not have been possible without **Jonathan Worthington**
  - friendly
  - patient
  - responsive
  - brilliant

Benefits

- Better architecture
- Clean data structures and flow
- Grammar based parser/compiler done right
- ”real-world” ***idiomatic*** code examples to **study**

Thanks, Jonathan!

Side effect: providing income to Jonathan

→ faster (progress with) Perl 6

Demo

https://agrammon.ch

https://model.agrammon.ch/single

When will Agrammon 6 go online?

Christmas
… 2018
… I think.

Thanks for listening.

Questions?